



- (51) International Patent Classification:
G06K 9/00 (2006.01)
- (21) International Application Number:
PCT/IB2018/000386
- (22) International Filing Date:
07 February 2018 (07.02.2018)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
62/456,050 07 February 2017 (07.02.2017) US
62/553,953 04 September 2017 (04.09.2017) US
62/598,487 14 December 2017 (14.12.2017) US
- (71) Applicant: **MINDMAZE HOLDING SA** [CH/CH];
Chemin De Roseneck 5, 1006 Lausanne (CH).
- (72) Inventors: **TADI, Tej**; Chemin De Roseneck 5, 1006 Lausanne (CH). **BOLOMEY, Léandre**; Chemin De Roseneck 5, 1006 Lausanne (CH). **FREMAUX, Nicolas**; Chemin De Roseneck 5, 1006 Lausanne (CH). **RUBIO, José**; Chemin De Roseneck 5, 1006 Lausanne (CH). **CARDIN, Sylvain**; Chemin De Roseneck 5, 1006 Lausanne (CH). **ROTH, Flavio**; Chemin De Roseneck 5, 1006 Lausanne (CH).

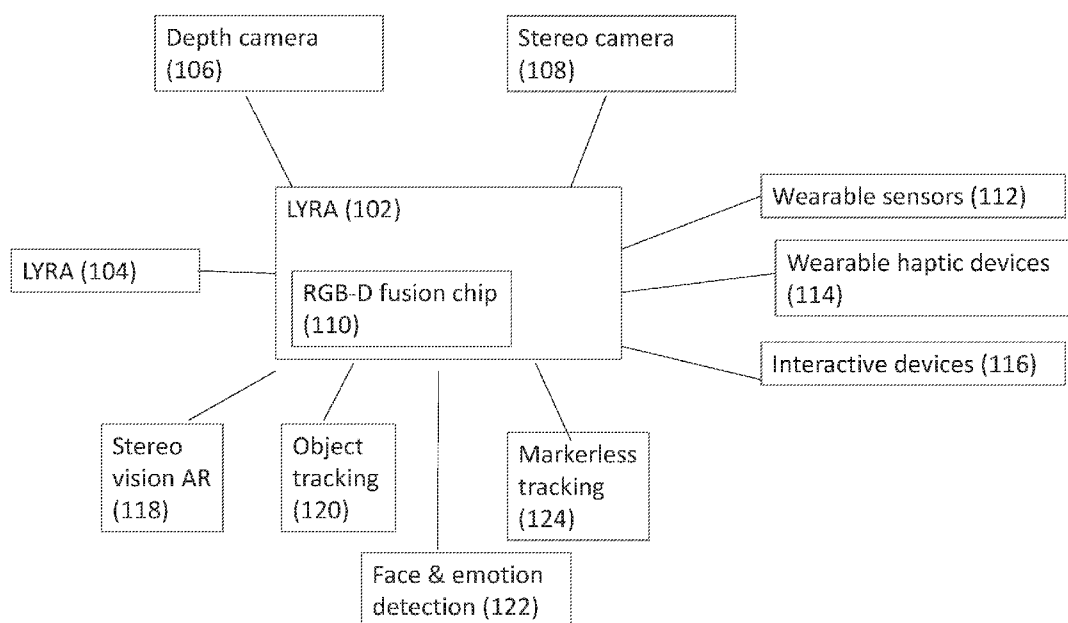
(CH). **OSTLUND, Jonas**; Chemin De Roseneck 5, 1006 Lausanne (CH). **OTT, Renaud**; Chemin De Roseneck 5, 1006 Lausanne (CH). **CONDOLO, Frédéric**; Chemin De Roseneck 5, 1006 Lausanne (CH). **BOURDAUD, Nicolas**; Chemin De Roseneck 5, 1006 Lausanne (CH). **LEVI CAPITAO CANTANTE, Flavio**; Chemin De Roseneck 5, 1006 Lausanne (CH). **BARBIER, Corentin**; Chemin De Roseneck 5, 1006 Lausanne (CH). **GOMEZ LORENZO, Ieltxu**; Chemin De Roseneck 5, 1006 Lausanne (CH).

(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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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,

(54) Title: SYSTEMS, METHODS AND APPARATUSES FOR STEREO VISION AND TRACKING

Figure 1



(57) Abstract: A system, method and apparatus for stereo vision and tracking with a plurality of coupled cameras and optional sensors.



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).

Published:

- *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

SYSTEMS, METHODS AND APPARATUSES FOR STEREO VISION AND TRACKING

FIELD OF THE DISCLOSURE

[0001] The present disclosure is directed to systems, methods and apparatuses for stereo vision and tracking, and in particular, to systems, methods and apparatuses for stereo vision which include a plurality of image sensors (e.g., cameras), as well as (in some embodiments) additional sensors that also includes tracking of at least part of a user.

BACKGROUND OF THE DISCLOSURE

[0002] Stereoscopic cameras provide a stereo view and are well known. For example, International Patent Publication no. WO2014154839 is understood to describe a camera system for capturing stereo data using two RGB cameras combined with a depth sensor for tracking the motion of an object (e.g., a person). The computations of the system are performed by a separate computer, which can lead to lag. Other examples include:

- The Persee product of Orbbec 3D (also known as Shenzhen Orbbec Co., Ltd.; <https://orbbec3d.com/>) combines camera functions with an ARM processor in a single apparatus. The apparatus includes a single RGB camera, a depth sensor, an infrared receiving port and a laser projector to provide stereo camera information;
- International Patent Publication no. WO2016192437, describes a system in which infrared sensor data is combined with RGB data to create a 3D image; and
- The Zed product of Stereolabs Inc (<https://www.stereolabs.com/zed/specs/>) provides a 3D camera with tracking capabilities.

BRIEF SUMMARY OF THE DISCLOSURE

[0003] Embodiments of the present disclosure are directed to systems, methods and apparatuses for stereo vision which include tracking, and in particular, to systems, methods and apparatuses for stereo vision which include a plurality of image sensors (e.g., cameras), as well as (in some

embodiments) additional sensors.

[0004] According to at least some embodiments there is provided a stereo vision procurement apparatus for obtaining stereo visual data, comprising: a stereo RGB camera; a depth sensor; and an RGB-D fusion module, wherein: each of said stereo RGB camera and said depth sensor are configured to provide pixel data corresponding to a plurality of pixels, said RGB-D fusion module is configured to combine RGB pixel data from said stereo RGB camera and depth information pixel data from said depth sensor to form stereo visual pixel data (SVPD), and said RGB-D fusion module is implemented in an FPGA field-programmable gate array).

[0005] Optionally the apparatus further comprises a de-mosaicing module configured to perform a method comprising: averaging the RGB pixel data associated with a plurality of green pixels surrounding red and blue sites for R(B) at B-G(R-G) sites or R(B) at R-G(B-G) sites, and reducing a number of green pixel values from the RGB pixel data to fit a predetermined pixel array (e.g., a 5 x 5 window) for R(B) at B(R) sites.

[0006] Optionally said stereo RGB camera comprises a first camera and a second camera, each of said first and second cameras being associated with a clock on said FPGA, and said FPGA including a double clock sampler for synchronizing said clocks of said first and right cameras.

[0007] Optionally the apparatus further comprises a histogram module comprising a luminance calculator for determining a luminance level of at least said RGB pixel data; and a classifier for classifying said RGB pixel data according to said luminance level, wherein said luminance level is transmitted to said stereo RGB camera as feedback.

[0008] Optionally the apparatus further comprises a white balance module configured to apply a smoothed GW (gray world) algorithm to said RGB pixel data.

[0009] Optionally the apparatus further comprises a processor; and a biological sensor configured to provide biological data, wherein: said biological sensor is selected from the group consisting of: an EEG sensor, a heartrate sensor, an oxygen saturation sensor, an EKG sensor, or EMG sensor, and a combination thereof, the processor is configured to process the biological data to form a plurality of sub-features, said sub-features are combined by the FPGA to form a feature.

[0010] Optionally said FPGA is implemented as a field-programmable gate array (FPGA) comprising a system on a chip (SoC), including an operating system as a SOM (system on module).

[0011] Optionally the apparatus further comprises a CPU SOM for performing overflow operations from said FPGA.

[0012] Optionally the apparatus further comprises a processor; and a plurality of tracking devices to track movement of a subject, wherein: the processor is configured to process data from the tracking devices to form a plurality of sub-features, and said sub-features are combined by said FPGA to form a feature to track movements of the subject.

[0013] Optionally the tracking devices comprise a plurality of wearable sensors.

[0014] Optionally the apparatus further comprises a processor; and a multi-modal interaction device in communication with a subject, said multi-modal interaction device comprising said plurality of tracking devices and at least one haptic feedback device, wherein: the processor is configured to process data from the tracking devices to form a plurality of tracking sub-features, and said sub-features are combined by said FPGA to form a feature to track movements of the subject and to provide feedback through said at least one haptic feedback device.

[0015] Optionally the apparatus further comprises a processor configured to perform a defined set of operations in response to receiving a corresponding instruction selected from an instruction set of codes; and a memory; wherein: said defined set of operations including: a first set of codes for operating said RGB-D fusion module to synchronize RGB pixel data and depth pixel data, and for creating a disparity map; and a second set of codes for creating a point cloud from said disparity map and said depth pixel data.

[0016] Optionally said point cloud comprises a colorized point cloud.

[0017] Optionally the apparatus further comprises a memory; and a processor configured to perform a defined set of operations for performing any of the functionality as described herein in response to receiving a corresponding instruction selected from an instruction set of codes.

[0018] Optionally said processor is configured to operate according to a set of codes selected from the instruction set for a de-noising process for a CFA (color filter array) image according to a W-means process.

[0019] Optionally said computational device comprises a second set of codes selected from the instruction set for operating a bad pixel removal process.

[0020] According to at least some embodiments there is provided a system comprising the

apparatus as described herein, further comprising a display for displaying stereo visual data.

[0021] Optionally the system further comprises an object attached to a body of a user; and an inertial sensor, wherein said object comprises an active marker, input from said object is processed to form a plurality of sub-features, and said sub-features are combined by the FPGA to form a feature.

[0022] Optionally the system further comprises a processor for operating a user application, wherein said RGB-D fusion module is further configured to output a colorized point cloud to said user application.

[0023] Optionally said processor is configured to transfer SVPD to said display without being passed to said user application, and said user application is additionally configured to provide additional information for said display that is combined by said FPGA with said SVPD for output to said display.

[0024] Optionally said biological sensor is configured to output data via radio-frequency (RF), and wherein: the system further comprises an RF receiver for receiving the data from said biological sensor, and said feature from said FPGA is transmitted to said user application.

[0025] Optionally the system further comprises at least one of a haptic or tactile feedback device, the device configured to provide at least one of haptic or tactile feedback, respectively, according to information provided by said user application.

[0026] According to at least some embodiments there is provided a stereo vision procurement system comprising: a first multi-modal interaction platform configurable to be in communication with one or more additional second multi-modal interaction platforms; a depth camera; a stereo RGB camera; and an RGB-D fusion chip; wherein: each of said stereo RGB camera and said depth camera are configured to provide pixel data corresponding to a plurality of pixels, the RGB-D fusion chip comprises a processor operative to execute a plurality of instructions to cause the chip to fuse said RGB pixel data and depth pixel data to form stereo visual pixel data.

[0027] Optionally the depth camera is configured to provide depth pixel data according to TOF (time of flight).

[0028] Optionally the stereo camera is configured to provide SVPD from at least one first and at least one second sensor.

[0029] Optionally the RGB-D fusion chip is configured to preprocess at least one of SVPD and depth pixel data so as to form a 3D point cloud with RGB pixel data associated therewith.

[0030] Optionally the fusion chip is further configured to form the 3D point cloud for tracking at least a portion of a body by at least the first multi-modal interaction platform.

[0031] Optionally the system further comprises at least one of a display and a wearable haptic device, wherein at least the first multi-modal interaction platform is configured to output data to at least one of the display and the haptic device.

[0032] Optionally the system further comprises one or more interactive objects or tools configured to perform at least one of giving feedback, receiving feedback, and receiving instructions from at least one of the multi-modal interaction platforms.

[0033] Optionally the system further comprises one or more sensors configured to communicate with at least one of the multi-modal interaction platforms.

[0034] Optionally the one or more sensors include at least one of: a stereo vision AR (augmented reality) component configured to display an AR environment according to at least one of tracking data of a user and data received from the first multi-modal interaction platform, and a second additional multi-modal interaction platform; an object tracking sensor; a facial detection sensor configured to detect a human face, or emotions thereof; and a markerless tracking sensor in which an object is tracked without additional specific markers placed on it.

[0035] According to at least some embodiments there is provided a multi-modal interaction platform system comprising: a multi-modal interaction platform; a plurality of wearable sensors each comprising an active marker configured to provide an active signal for being detected; an inertial sensor configured to provide an inertial signal comprising position and orientation information; at least one of a heart rate and oxygen saturation sensor, or a combination thereof; an EEG sensor; and at least one wearable haptic devices, including one or more of a tactile feedback device and a force feedback device.

[0036] According to at least some embodiments there is provided a method for processing image information comprising: receiving SVPD from a stereo camera; performing RGB preprocessing on the input pixel data to produce preprocessed RGB image pixel data; using the RGB preprocessed image pixel data in the operation of the stereo camera with respect to at least one of

an autogain and an autoexposure algorithm; rectifying the SVPD so as to control artifacts caused by the lens of the camera; and calibrating the SVPD so as to prevent distortion of the stereo pixel input data by the lens of the stereo camera.

[0037] Optionally the method further comprises colorizing the preprocessed RGB image pixel data, and creating a disparity map based on the colorized, preprocessed RGB image pixel data.

[0038] Optionally calibration comprises matching the RGB pixel image data with depth pixel data.

[0039] Optionally the disparity map is created by: obtaining depth pixel data from at least one of the stereo pixel input data, the preprocessed RGB image pixel data, and depth pixel data from a depth sensor, and checking differences between stereo images.

[0040] Optionally said disparity map, plus depth pixel data from the depth sensor in the form of a calibrated depth map, is combined for the point cloud computation.

[0041] According to at least some embodiments there is provided an image depth processing method for depth processing of one or more images comprising: receiving TOF (time-of-flight) image data of an image from a TOF camera; creating at least one of a depth map or a level of illumination for each pixel from the TOF data; feeding the level of illumination into a low confidence pixel removal process comprising: comparing a distance that each pixel is reporting; correlating said distance of said each pixel to the illumination provided by said each pixel, removing any pixel upon the illumination provided by the pixel being outside a predetermined acceptable range such that the distance cannot be accurately determined; processing depth information to remove motion blur of the image, wherein motion blur is removed by removing artifacts at edges of moving objects in depth of the image; and applying at least one of temporal or spatial filters to the image data.

[0042] According to at least some embodiments there is provided a stereo image processing method comprising: receiving first data flow of at least one image from a first RGB camera and second data flow of at least one image from a second RGB camera; sending the first and second data flows to a frame synchronizer; and synchronizing, using the frame synchronizer, a first image frame from the first data flow and a second image frame from the second data flow such that time shift between the first image and frame and the second image frame is substantially eliminated.

[0043] Optionally sampling, before sending the first and second data flows to the frame

synchronizer, the first and second data flows such that each of the first and second data flows are synchronized with a single clock; and detecting which data flow is advanced of the other, and directing the advanced data flow to a First Input First Output (FIFO), such that the data from the advanced flow is retained by the frame synchronizer until the other data flow reaches the frame synchronizer.

[0044] Optionally the method further comprises serializing frame data of the first and second data flows as a sequence of bytes.

[0045] Optionally the method further comprises detecting non-usable pixels.

[0046] Optionally the method further comprises constructing a set of color data from each of the first and second data flows.

[0047] Optionally the method further comprises color correcting each of the first and second data flows.

[0048] Optionally the method further comprises corresponding the first and second data flows into a CFA (color filter array) color image data; applying a denoising process for the CFA image data, the process comprising: grouping four (4) CFA colors to make a 4-color pixel for each pixel of the image data; comparing each 4-color pixel to neighboring 4-color pixels; attributing a weight to each neighbor pixel depending on its difference with the center 4-color pixel; and for each color, computing a weighted mean to generate the output 4-color pixel.

[0049] Optionally said denoising process further comprises performing a distance computation according to a Manhattan distance, computed between each color group neighbor and the center color group.

[0050] Optionally the method further comprises applying a bad pixel removal algorithm before said denoising process.

[0051] Optionally the apparatus as described herein is able to obtain SVDP and to track a user, wherein the apparatus further comprises: a body model; and one or more processors having computer instructions operating thereon configured to cause the processor to: fit data points from the depth sensor to the body model according to a probabilistic fitting algorithm, said probabilistic fitting algorithm being constrained according to at least one constraint defined according to human anatomy, identifying a plurality of data points as super points and assigning each of said super

points an additional weight; wherein: a plurality of said data points are identified with joints of the anatomy, said super points are defined according to one or more objects attached to a body, each of said stereo RGB camera and said depth sensor are configured to provide data as a plurality of pixels, said RGB-D fusion module is configured to combine RGB data from said stereo RGB camera and depth information from said depth sensor to SVPD, and the depth sensor provides data to determine a three-dimensional location of a body in space according to a distance of the body from the depth sensor.

[0052] Optionally said one or more objects attached to the body comprise one or more of at least one active marker configured to provide a detectable signal and a passive object.

[0053] Optionally said data points identified with joints of the human body are identified according to a previously determined position as an estimate.

[0054] Optionally the body model comprises a template, said template including a standard model of a skeleton according to a hierarchy of joints as vertices and skinning, and a first determination of a position of at least one of the joints of the hierarchy of joints according to said template.

[0055] According to at least some embodiments there is provided a system comprising an apparatus as described herein, optionally comprising a characteristic of a system as described herein, further comprising a display for displaying SVPD.

[0056] Optionally the system further comprises one or more objects attached to the user; and an inertial sensor, wherein said one or more objects comprises an active marker, the computer instructions are configured to cause the processor to form a plurality of sub-features from input from said one or more objects and combining said sub-features into a feature.

[0057] Optionally the system further comprises at least one of a haptic feedback device and a tactile feedback device configured to provide at least one of haptic and tactile feedback according to information provided by said user application.

[0058] Optionally computer instructions include instructions which cause the processor to perform as a tracking engine.

[0059] Optionally the tracking engine is configured to track at least one of the position of the user's body and the position of one or more body parts of the user, including but not limited, to one or more of an arm, a leg, a hand, a foot, and a head.

[0060] Optionally the tracking engine is configured to decompose signals representing physical actions made by the user into data representing a series of gestures.

[0061] Optionally the tracking engine is configured to decompose signals representing physical actions into data representing a series of gestures via classifier functionality.

[0062] Optionally the system further comprises a plurality of templates, wherein the computer instructions are further configured to cause the processor to initialize a template of the plurality of templates. Optionally the template features a model of a human body configured only as a plurality of parameters, only as a plurality of features, or both.

[0063] Optionally the plurality of parameters and/or features include a skeleton, and one or more joints.

[0064] Optionally the computer instructions are additionally configured to cause the processor to utilize the plurality of parameters and/or features to assist in tracking of the user's movements.

[0065] Optionally the computer instructions are configured to map the sensor data onto a GMM (Gaussian mixture model).

[0066] Optionally the body model includes a sparse-skin representation.

[0067] Optionally the computer instructions are additionally configured to cause the processor to suppress corresponding Gaussians.

[0068] Optionally data is mapped to a GMM.

[0069] Optionally said data is mapped to said GMM by a classifier.

[0070] Optionally the tracking engine includes a template engine configured to read a template from a template database, and the computer instructions are additionally configured to: cause the processor to operate as a GMM mapper, and send the template into the GMM mapper.

[0071] Optionally the computer instructions are additionally configured to cause the processor to operate as a point cloud decomposer, and the GMM mapper is configured to receive point cloud information from the point cloud decomposer.

[0072] Unless otherwise defined, all technical and scientific terms used herein have the same meaning as commonly understood by one of ordinary skill in the art to which this disclosure belongs. The materials, systems, apparatuses, methods, and examples provided herein are

illustrative only and not intended to be limiting.

[0073] Implementation of the embodiments of the present disclosure include performing or completing tasks, steps, and functions, manually, automatically, or a combination thereof. Specifically, steps can be implemented by hardware or by software on an operating system, of a firmware, and/or a combination thereof. For example, as hardware, steps of at least some embodiments of the disclosure can be implemented as a chip or circuit (e.g., ASIC). As software, steps of at least some embodiments of the disclosure can be implemented as a number of software instructions being executed by a computer (e.g., a processor) using an operating system. Thus, in any case, selected steps of methods of at least some embodiments of the disclosure can be performed by a processor for executing a plurality of instructions.

[0074] Software (e.g., an application, computer instructions, code) which is configured to perform (or cause to be performed) certain functionality of some of the disclosed embodiments may also be referred to as a “module” for performing that functionality, and also may be referred to a “processor” for performing such functionality. Thus, processor, according to some embodiments, may be a hardware component, or, according to some embodiments, a software component.

[0075] Further to this end, in some embodiments, a processor may also be referred to as a module, and, in some embodiments, a processor may comprise one more modules. In some embodiments, a module may comprise computer instructions - which can be a set of instructions, an application, software, which are operable on a computational device (e.g., a processor) to cause the computational device to conduct and/or achieve one or more specific functionality. Furthermore, the phrase “abstraction layer” or “abstraction interface”, as used with some embodiments, can refer to computer instructions (which can be a set of instructions, an application, software) which are operable on a computational device (as noted, e.g., a processor) to cause the computational device to conduct and/or achieve one or more specific functionality. The abstraction layer may also be a circuit (e.g., an ASIC see above) to conduct and/or achieve one or more specific functionality. Thus, for some embodiments, and claims which correspond to such embodiments, the noted feature/functionality can be described/claimed in a number of ways (e.g., abstraction layer, computational device, processor, module, software, application, computer instructions, and the like).

[0076] Some embodiments are described with regard to a "computer", a "computer network,"

and/or a “computer operational on a computer network,” it is noted that any device featuring a processor (which may be referred to as “data processor”; “pre-processor” may also be referred to as “processor”) and the ability to execute one or more instructions may be described as a computer, a computational device, and a processor (e.g., see above), including but not limited to a personal computer (PC), a server, a cellular telephone, an IP telephone, a smart phone, a PDA (personal digital assistant), a thin client, a mobile communication device, a smart watch, head mounted display or other wearable that is able to communicate externally, a virtual or cloud based processor, a pager, and/or a similar device. Two or more of such devices in communication with each other may be a "computer network."

BRIEF DESCRIPTION OF THE DRAWINGS

[0077] Embodiments of the present disclosure are herein described, by way of example only, with reference to the accompanying drawings. With specific reference now to the drawings in detail, it is stressed that the particulars shown are by way of example and for purposes of illustrative discussion of the preferred embodiments of inventions disclosed herein, and are presented in order to provide what is believed to be the most useful and readily understood description of the principles and conceptual aspects of various embodiments of the inventions disclosed herein.

[0010] Figure 1 shows a non-limiting example of a system according to at least some embodiments of the present disclosure;

[0011] Figures 2A, 2B and 2C show additional details and embodiments of the system of Figure 1;

[0012] Figure 3 shows a non-limiting example of a method for preprocessing according to at least some embodiments of the present disclosure;

[0013] Figures 4A and 4B shows a non-limiting example of a method for depth preprocessing according to at least some embodiments of the present disclosure;

[0014] Figures 5A-5D show a non-limiting example of a data processing flow for the FPGA (field-programmable gate array) according to at least some embodiments of the present disclosure;

[0015] Figures 6A-6E shows a non-limiting example of a hardware system for the camera according to at least some embodiments of the present disclosure;

[0016] Figure 7 shows a non-limiting example of a method for stereo processing according to at least some embodiments of the present disclosure;

[0017] Figure 8 shows a non-limiting example of a MCU configuration according to at least some embodiments of the present disclosure;

[0018] Figure 9 shows a non-limiting example of a camera according to at least some embodiments of the present disclosure;

[0019] Figure 10 shows a non-limiting example of a configuration for double clock sampler functions according to at least some embodiments of the present disclosure;

[0020] Figures 11A and 11B show a non-limiting example of a buffer configuration according to at least some embodiments of the present disclosure;

[0021] Figure 12 shows a non-limiting example of an internal buffer cells arrangement: a) global structure, b) mask for defective pixel detection, c) mask for de-mosaic task; and Figure 12D, which shows exemplary state machines;

[0022] Figures 13A-13IH show a non-limiting example of a method for de-mosaic according to at least some embodiments of the present disclosure;

[0023] Figure 14 shows a non-limiting example of a method for white balance correction according to at least some embodiments of the present disclosure;

[0024] Figure 15 shows a non-limiting example of a method for performing the histogram adjustment according to at least some embodiments of the present disclosure;

[0025] Figure 16 shows an illustrative, exemplary, non-limiting process for stereo rectification according to at least some embodiments of the present disclosure;

[0026] Figure 17A shows an illustrative, exemplary, non-limiting system for stereo rectification according to at least some embodiments of the present disclosure;

[0027] Figure 17B shows an illustrative, exemplary, non-limiting mapper module for use with the system of Figure 17A according to at least some embodiments of the present disclosure;

[0028] Figure 17C shows an illustrative, exemplary, non-limiting memory management for use with the system of Figure 17A according to at least some embodiments of the present disclosure;

[0029] Figure 17D shows a non-limiting example of an image;

[0030] Figure 17E shows the memory filling scheme for this image;

[0031] Figure 17F shows a non-limiting, exemplary finite state machine for use with the system of Figure 17A according to at least some embodiments of the present disclosure;

[0032] Figure 18A shows an illustrative, exemplary, non-limiting disparity map method according to at least some embodiments of the present disclosure;

[0033] Figure 18B shows an illustrative, exemplary, non-limiting method for calculating a cost for the disparity map method according to at least some embodiments of the present disclosure;

[0034] Figure 19A shows an example of image representation for “W-means” algorithm;

[0035] Figure 19B shows the effects of parameters on “W-means” weight;

[0036] Figure 19C shows taxicab geometry versus Euclidean distance: In taxicab geometry, the red, yellow, and blue paths will have the shortest length of $|6| + |6| = 12$. In Euclidean geometry, the green line has length $\sqrt{72}$, and is the unique shortest path;

[0037] Figure 19D shows the W-means algorithm, in a non-limiting example;

[0038] Figure 20 shows the results of state of the art and “W-means” algorithms, after application of the debayer. Image size (150 × 80) (zoom). Algorithm parameters are : NLM(h = 6, f = 3, r = 10), Vinh(p = 8), PSWFA(n = 5), W_means(h = 16, σ = 4), W_means_1stOrd(h = 32, σ = 2), W_means_thr(σ = 12), W_means_thr_optdiv(σ = 12);

[0039] Figure 21A shows required ports of the filter to be added in the image pipeline, while Figure 21B shows a pixel stream interface chronogram;

[0040] Figure 22A shows a schematic of the Bailey and Jimmy method, while Figure 22B shows an exemplary implementation thereof;

[0041] Figure 23 shows an exemplary bad pixel removal method FPGA implementation diagram, in which each yellow unit is a VHDL component;

[0042] Figure 24 shows an exemplary, illustrative non-limiting data flow for bad pixel removal;

[0043] Figure 25 shows an exemplary, illustrative non-limiting diagram for “W-means” unit FPGA implementation;

[0044] Figure 26 shows an exemplary, illustrative non-limiting generate kernel component diagram for “W-means” algorithm, where the red annotations are color groups;

[0045] Figure 27 shows an exemplary, illustrative non-limiting distance computation component diagram for “W-means” algorithm, in which “c_{cg(i)}” is the center color group with color number i , “c_{cg(x)(i)}” is the neighbor number x with color number i and “d(x)” is the result distance for the neighbor number x . $i \in [1, 4]$, $x \in [1, 8]$;

[0046] Figure 28 shows an exemplary, illustrative non-limiting filter core “thr_optdiv” component diagram for “W-means” algorithm, in which “c_{cg(i)}” is the center color group with color number i , “c_{cg(x)(i)}” is the neighbor number x with color number i , and “f_{cg(i)}” is the center color group with color number i . $i \in [1, 4]$, $x \in [1, 8]$;

[0047] Figure 29A shows an exemplary, illustrative non-limiting format output component diagram for “W-means” algorithm, while Figure 29B shows an exemplary, illustrative valid output color group for “W-means” algorithm in a CFA (color filter array) image. In this example the CFA colors are “GBRG” (first image row start with green then blue and the second row starts with red then green);

[0048] Figure 30 shows an exemplary, illustrative non-limiting data flow for bad pixel removal and denoising;

[0049] Figures 31A and 31B show final test results on the camera module for both the bad pixel and “W-means” algorithms. Image size (150 × 150) (zoom);

[0050] Figure 32 shows a non-limiting exemplary method for color correction according to at least some embodiments;

[0051] Figures 33A-33D show a non-limiting exemplary FPGA configuration according to at least some embodiments;

[0052] Figure 34 shows a non-limiting example of a method for tracking the user, optionally performed with the system of Figures 1 or 2, according to at least some embodiments of the present disclosure;

[0053] Figure 35 shows a non-limiting example of a tracking engine, optionally for use with the system of Figures 1 or 2, or the method of Figure 34, according to at least some embodiments of

the present disclosure;

[0054] Figure 36 shows templates and a template engine, according to at least some embodiments of the present disclosure;

[0055] Figure 37 shows a non-limiting example of a method for creating and using templates, according to at least some embodiments of the present disclosure;

[0056] Figures 38A to 38E show non-limiting examples of methods for mapping data to track a user, according to at least some embodiments of the present disclosure;

[0057] Figure 39 shows a non-limiting example of a method for applying a deformation model, according to at least some embodiments of the present disclosure;

[0058] Figure 40 shows a non-limiting example of a method for pose recovery, according to at least some embodiments of the present disclosure;

[0059] Figure 41 shows a non-limiting example of a method for segmentation of a background object, according to at least some embodiments of the present disclosure;

[0060] Figure 42 shows a non-limiting example of a method for joint detection, according to at least some embodiments of the present disclosure;

[0061] Figures 43 and 44 show two non-limiting example methods for applying VR to medical therapeutics according to at least some embodiments of the present disclosure;

[0062] Figure 45 shows a non-limiting example method for applying VR to increase a user's ability to perform ADL (activities of daily living) according to at least some embodiments; and

[0063] Figure 46 shows a non-limiting example method for applying AR to increase a user's ability to perform ADL (activities of daily living) according to at least some embodiments.

DETAILED DESCRIPTION OF AT LEAST SOME EMBODIMENTS

[0064] Figure 1 shows a non-limiting example of a system according to at least some embodiments of the present disclosure. As shown, a system 100 features a multi-modal interaction platform 102, which can be chained to one or more additional multi-modal interaction platforms 104 as shown. Multi-modal interaction platform 102 can in turn be in communication with a depth sensor (e.g., camera) 106, a stereo sensor (e.g., camera) 108, and an RGB-D fusion chip 110. Depth camera

106 is configured to provide depth sensor data, which may be pixel data, for example, according to TOF (time of flight) relative to each pixel. Stereo camera 108 is configured to provide stereo camera data (pixel data) from left (first) and right (second) camera sensors (not shown). The functions of RGB-D fusion chip 110 are described in greater detail with regard to Figure 3, but preferably include preprocessing of stereo camera data and depth data, to form a 3D point cloud with RGB data associated with it. The formation of the point cloud enables its use for tracking a body or a portion thereof, for example (or for other types of processing), by multi-modal interaction platform 102. Multi-modal interaction platform 102 can then output data to a visual display (not shown) or a wearable haptic device 114, for example to provide haptic feedback. One or more interactive objects or tools 116 may be provided to give or receive feedback or instructions from multi-modal interaction platform 102, or both.

[0065] A plurality of additional functions may be provided through the components described herein, alone or in combination, with one or more additional sensors, provided through outputs from multi-modal interaction platform 102. For example, a stereo vision AR (augmented reality) component 118 can be provided to display an AR environment according to tracking data of the subject and other information received from multi-modal interaction platform 102. Such object tracking can be enabled by an object tracking output 120. Detection of a human face, optionally with detection of emotion, may be provided through such an output 122. Markerless tracking 124, in which an object is tracked without additional specific markers placed on it, may also be provided. Other applications are also possible.

[0066] Figure 2A shows a detail of the system of Figure 1, shown as a system 200. In this figure, multi-modal interaction platform 102 is shown as connected to a plurality of different wearable sensors 112, including, but not limited, to an active marker 202, which can, for example, provide an active signal for being detected, such as an optical signal (for example) which would be detected by the stereo camera; an inertial sensor 204, for providing an inertial signal that includes position and orientation information; a heart rate/oxygen saturation sensor 206; EEG electrodes 208; and/or one or more additional sensors 210. Operation of some wearable sensors 112 in conjunction with multi-modal interaction platform 102 is described in greater detail below.

[0067] Multi-modal interaction platform 102 is also shown as connected to a plurality of different wearable haptic devices 114, including one or more of a tactile feedback device 212 and a force

feedback device 214. For example and without limitation, such wearable haptic devices 114 could include a glove with small motors on the tips of the fingers to provide tactile feedback or such a motor connected to an active marker. Without wishing to be limited to a single benefit or to a closed list, connecting such sensors/feedback devices on a hardware platform enables better data synchronization, for example with timing provided by the same hardware clock signal, which can be useful for analysis.

[0068] Figure 2B shows a non-limiting example of a system according to at least some embodiments of the present disclosure. As shown, a system 220 features a camera 222, a depth sensor 224 and optionally an audio sensor 226. Optionally an additional sensor 240 is also included. Optionally camera 222 and depth sensor 224 are combined in a single product (e.g., Kinect® product of Microsoft®, and/or as described in US Patent No. 8,379,101). Figure 1B shows an exemplary implementation for camera 222 and depth sensor 224. Optionally, camera 222 and depth sensor 224 can be implemented with the LYRA camera of Mindmaze SA. The integrated product (i.e., camera 222 and depth sensor 224) enables, according to some embodiments, the orientation of camera 222 to be determined with respect to a canonical reference frame. Optionally, three or all four sensors (e.g., a plurality of sensors) are combined in a single product.

[0069] The sensor data, in some embodiments, relates to physical actions of a user (not shown), which are accessible to the sensors. For example, camera 222 can collect video data of one or more movements of the user, while depth sensor 224 may provide data to determine the three dimensional location of the user in space according to the distance of the user from depth sensor 224 (or more specifically, the plurality of distances that represent the three dimensional volume of the user in space). Depth sensor 224 can provide TOF (time of flight) data regarding the position of the user, which, when combined with video data from camera 222, allows a three dimensional map of the user in the environment to be determined. As described in greater detail below, such a map enables the physical actions of the user to be accurately determined, for example, with regard to gestures made by the user. Audio sensor 226 preferably collects audio data regarding any sounds made by the user, optionally including, but not limited to, speech. Additional sensor 240 can collect biological signals about the user and/or may collect additional information to assist the depth sensor 224.

[0070] Sensor data is collected by a device abstraction layer 228, which preferably converts the

sensor signals into data which is sensor-agnostic. Device abstraction layer 228 preferably handles the necessary preprocessing such that, if different sensors are substituted, only changes to device abstraction layer 228 would be required; the remainder of system 220 can continue functioning without changes (or, in some embodiments, at least without substantive changes). Device abstraction layer 228 preferably also cleans signals, for example, to remove or at least reduce noise as necessary, and can also be used to normalize the signals. Device abstraction layer 228 may be operated by a computational device 250, and any method steps may be performed by a computational device (note - modules and interfaces disclosed herein are assumed to incorporate, or to be operated by, a computational device, even if not shown).

[0071] The preprocessed signal data from the sensors can then be passed to a data analysis layer 230, which preferably performs data analysis on the sensor data for consumption by an application layer 236 (according to some embodiments, “application,” means any type of interaction with a user). Preferably, such analysis includes tracking analysis, performed by a tracking engine 232, which can track the position of the user’s body and also can track the position of one or more body parts of the user, including but not limited, to one or more of arms, legs, hands, feet, head and so forth. Tracking engine 232 can decompose physical actions made by the user into a series of gestures. A “gesture” in this case may include an action taken by a plurality of body parts of the user, such as taking a step while swinging an arm, lifting an arm while bending forward, moving both arms, and so forth. Such decomposition and gesture recognition can also be done separately, for example, by a classifier trained on information provided by tracking engine 232 with regard to tracking the various body parts. Tracking engine 232 may be adjusted according to a presence or absence of each limb of the user. For example, if the user is an amputee who is missing a leg, tracking engine 232 can be calibrated to take such a loss into account. Such calibration may take place automatically or may occur as part of a user directed calibration process at the start of a session with a particular user.

[0072] It is noted that while the term “classifier” is used throughout, this term is also intended to encompass “regressor”. For machine learning, the difference between the two terms is that for classifiers, the output or target variable takes class labels (that is, is categorical). For regressors, the output variable assumes continuous variables (see for example <http://scottge.net/2015/06/14/ml101-regression-vs-classification-vs-clustering-problems/>).

[0073] The tracking of the user's body and/or body parts, optionally decomposed to a series of gestures, can then be provided to application layer 236, which translates the actions of the user into a type of reaction and/or analyzes these actions to determine one or more action parameters. For example, and without limitation, a physical action taken by the user to lift an arm is a gesture which could translate to application layer 236 as lifting a virtual object. Alternatively or additionally, such a physical action could be analyzed by application layer 236 to determine the user's range of motion or ability to perform the action.

[0074] To assist in the tracking process, optionally, one or more markers 238 can be placed on the body of the user. Markers 238 optionally feature a characteristic that can be detected by one or more of the sensors, such as by camera 222, depth sensor 224, audio sensor 226 or additional sensor 240. Markers 238 can be detectable by camera 222, for example, as optical markers. While such optical markers may be passive or active, preferably, markers 238 are active optical markers, for example featuring an LED light. More preferably, each of markers 238, or alternatively each pair of markers 238, can comprise an LED light of a specific color which is then placed on a specific location of the body of the user. The different colors of the LED lights, placed at a specific location, convey a significant amount of information to the system through camera 222; as described in greater detail below, such information can be used to make the tracking process efficient and accurate. Additionally, or alternatively, one or more inertial sensors can be added to the hands of the user as a type of marker 238, which can be enabled as Bluetooth or other wireless communication, such that the information would be sent to device abstraction layer 228. The inertial sensors can also be integrated with an optical component in at least markers 238 related to the hands, or even for more such markers 238. The information can then optionally be integrated to the tracking process, for example, to provide an estimate of orientation and location for a particular body part, for example as a prior restraint.

[0075] Data analysis layer 230, in some embodiments, includes a system calibration module 234. As described in greater detail below, system calibration module 234 is configured to calibrate the system with respect to the position of the user, in order for the system to track the user effectively. System calibration module 234 can perform calibration of the sensors with respect to the requirements of the operation of application layer 236 (although, in some embodiments – which can include this embodiment - device abstraction layer 228 is configured to perform sensor specific calibration). Optionally, the sensors may be packaged in a device (e.g., Microsoft® Kinect), which

performs its own sensor specific calibration.

[0076] Figure 2C shows a non-limiting example of a system according to at least some embodiments of the present disclosure. As shown, a system 252 includes the components of the system of Figure 2B, and additionally features a second camera 222B and a second depth sensor 224B. As a non-limiting example of a use for system 252, it could be used to provide additional information about the movements of a user. For example, camera 222B and depth sensor 224B could be attached to the user, for example and without limitation to headgear worn by the user. Camera 222A and depth sensor 224A would be placed external to the user, for example at a short distance from the user. Such a configuration would enable the hands of the user to be tracked separately from the body of the user.

[0077] For this implementation, one of camera 222A and camera 222B, and one of depth sensor 224A and depth sensor 224B, is preferably selected as the master while the other is the slave device. For example, preferably camera 222B and depth sensor 224B would be the master devices, such that control would be provided according to the movements of the user. Optionally only one of camera 222B and depth sensor 224B is provided; if so, then preferably at least depth sensor 224B is provided.

[0078] Another non-limiting implementation would use system 252 to extend the range of operation. Each of camera 222A,B and depth sensor 224A,B has a trade off between field of view and resolution: the greater the field of view, the lower the angular resolution is, and vice versa. In order for the range of operation to be extended to 10 meters, for example, it would be necessary to provide a plurality of cameras 222 and a plurality of depth sensors 224, stationed at various points along this range. The data would therefore have the necessary resolution and field of view.

[0079] Figure 3 shows a non-limiting example of a method for preprocessing according to at least some embodiments of the present disclosure. As shown, preprocessing starts at 302 with input from the stereo camera, provided as stereo data 304. Stereo data 304 undergoes RGB preprocessing 306, which in turn feeds back to the operation of stereo camera 302, for example, with regard to the autogain and autoexposure algorithm, described in greater below. In 308, image rectification is performed, to control artifacts caused by the lens of the camera. In some embodiments, a calibration process can be performed to prevent distortion of the image data by the lens, whether at the time of manufacture or at the time of use.

[0080] Optionally, the camera calibration process is performed as follows. To perform all these steps, intrinsic and extrinsic parameters of the cameras are needed to know how they are positioned one to each other, to know their distortion, their focal length and so on. These parameters are often obtained from a calibration step. This calibration step optionally comprises taking several pictures of a chessboard pattern with the cameras and then computing the parameters by finding the pattern (of known size) inside the images.

[0081] From the intrinsic calibration process, the intrinsic parameters of each camera are extracted and may comprise the following:

- Focal length: in pixels, (f_x, f_y) ;
- Principal point: in pixels, (c_x, c_y) ;
- Skew coefficient: defines the angle between the horizontal and vertical pixels axes, α_c ;
- Distortion coefficients: radial $(k_1, k_2, k_3, k_4, k_5, k_6)$ and tangential (p_1, p_2) distortion coefficients.

[0082] Then, from the extrinsic calibration process, the position of one camera to the other can be extracted by having a 3x3 rotation matrix r and a 3x1 translation vector t .

[0083] In 310, stereo RGB images that have been preprocessed may then be processed for colorization and for creating a disparity map, such may then be fed to a colored point cloud formation process 312. The process in 312 may be performed, for example, as described in the paper "Fusion of Terrestrial LiDAR Point Clouds with Color Imagery", by Colin Axel, 2013, available from <http://www.cis.rit.edu/DocumentLibrary/admin/uploads/CIS000202.PDF>. However, optionally, determination of the sensor position and orientation may be dropped, since the stereo camera and depth sensor can both be calibrated, with their position and orientation known before processing begins. In addition, pixels from the RGB camera can be matched with pixels from the depth sensor, providing an additional layer of calibration. The colored point cloud can then be output as the 3D point cloud with RGB data in 314.

[0084] Turning back to 310, the disparity map is created in 312 by obtaining the depth information from the stereo RGB images and then checking the differences between stereo images. The disparity map, plus depth information from the depth sensor in the form of a calibrated depth map (as described in greater detail below), is combined for the point cloud computation in 318, for a

more robust data set.

[0085] Depth information from the depth sensor can be obtained as follows. Depth and illumination data is obtained in 320, from TOF (time of flight) camera 326. The depth and illumination data may then be processed along two paths, a first path for TOF control 322, which in turn feeds back to TOF camera 326 to control illumination and exposure time according to the illumination data. A second path for TOF calibration 324 can then be used to correct the TOF image, by applying the factory calibration, which in turn feeds corrected TOF depth data into the depth map 328. Calibration of the TOF function may be required to be certain that the depth sensor data is correct, relative to the function of the depth sensor itself. Such calibration increases the accuracy of depth map 328. Depth map 328 can then be fed into 318, as described above, to increase the accuracy of creating the colorized point cloud.

[0086] Figures 4A and 4B show a non-limiting example of a method for depth preprocessing according to at least some embodiments of the present disclosure, which shows the depth processing method of Figure 3 in more detail. Accordingly, as shown in Figure 4A, a depth preprocessing process 400 starts with image (e.g., pixel) data being obtained from a TOF camera in 402, which may be used to create a depth map in 406, but may also may be used to determine a level of illumination in 414 for each pixel. The level of illumination can then be fed into a low confidence pixel removal process 408. This process compares the distance that a pixel in the image is reporting and correlates this reported distance to the illumination provided by that pixel. The settings for process 408 can be decided in advance, according to the acceptable noise level, which may for example be influenced by the application using or consuming the data. The lower the acceptable noise level, the lower the amount of data which is available. If the illumination is outside of a predetermined acceptable range, the distance cannot be accurately determined. Preferably, if this situation occurs, the pixel is removed.

[0087] A histogram process 416, which enables autoexposure and autogain adjustments, is described in greater detail below.

[0088] After removal of low confidence pixels in 408, the depth processing can continue with motion blur removal in 410, which can remove artifacts at edges of moving objects in depth (i.e., removing the pixels involved). The application of temporal and spatial filters may be performed in 412, which are used to remove noise from the depth (spatial) and average data over time to

remove noise (temporal). Spatial filters attenuate noise by reducing the variance among the neighborhood of a pixel, resulting in a smoother surface, but potentially at the cost of reduced contrast. Such a spatial filter may be implemented as a Gaussian filter for example, which uses a Gaussian weighting function, $G(p - p')$ to average the pixels, p' , within a square neighborhood, w , centered about the pixel, p .

[0089] Turning back to histogram process 416, the information obtained therefrom may also be passed to an exposure and illumination control process 418 as previously described, which is used to adjust the function of TOF camera 402. Figure 4B shows an exemplary illustrative non-limiting method for detecting defective pixels according to at least some embodiments of the present disclosure, which can be used for example with the method of Figure 4A, for example to remove low confidence pixels as previously described. The process 450 can be divided into three steps: interpolation, defect screening, candidate screening (for example).

[0090] While each incoming pixel (452) reaches the center of the moving window obtained in the buffer of the FPGA (field-programmable gate array), it is checked to determine if it was previously stored (in memory) as being defective (454). If not previously stored, the module proceeds to perform the candidate screening process (456) where the value of the pixel under test is compared toward surrounding neighbors average. If a certain threshold, TH_NEIGH , is exceeded, the inspected pixel is suspected to be defective, hence its data (value, position, neighbor average) are stored for further analysis.

[0091] A stored pixel is checked to determine whether it was previously labeled as defective (458), which leads to interpolation (460). If not previously labeled as defective, the pixel undergoes defect screening (462) by comparing its actual and previous values. A higher difference between these values as compared to the threshold TH_DIFF (to cancel effects of noise) corresponds to the pixel changing regularly, such that the pixel is no longer suspected as being defective. A time constant is incremented for each period of time that the pixel remains under suspicion of being defective. Another threshold, TH_FRAME , is defined and used to compare the value of the time constant. Once a pixel value (excluding noise) remains unchanged for a certain number of frames, such that the value of the time constant is equal to the second threshold of TH_FRAME , the pixel is determined to be defective. Now the interpolation step becomes active, so that defective pixel is corrected before it slides toward first mask_2 memory cell. Interpolation may be performed by

substituting investigated pixel value by average of its surrounding pixel. The average can be calculated among those pixels having the same filter color as the one in the center of the mask. An example of such a process is demonstrated in following pseudo-code form:

```

for pixel=1 to endFrame do
  if pixel already stored then
    if pixel already defective then
      Interpolate pixel
    else if  $|pixel - previousPixelValue| \leq TH\_DIFF$ 
      then if timeConst=TH_FRAME then
        Add pixel to defects list
      else
        Increment timeConst
      end
    else
      Remove pixel from candidate list
    end
  else if memory not full then
    if  $pixel - neighborsAverage \geq TH\_NEIGH$  then
      Add pixel to candidate list
    end
  end
end

```

[0092] Figures 5A-5D show a non-limiting example of a data processing flow for the FPGA according to at least some embodiments of the present disclosure. Figure 5A shows the overall flow 500, which includes input from one or more sensors 504, and output to one or more output devices 530. Input from sensors 504 can be processed through FPGA process 502 and then sent to a user application 506. User application 506 may then return output to output devices 530.

[0093] Figure 5B describes the detailed flow for some exemplary input sensors 504. Thus, and for example, as shown, exemplary input sensors 504 include one or more of a stereo camera 508, a ToF camera 510, an inertial sensor 512 and a sound input device 514. A non-limiting example of

sound input device 514 could include a microphone for example. Input from input sensors 504 may be received by a data sync and buffer 516, which operates as described in greater detail below, to synchronize various data streams (including without limitation between inputs of stereo camera 508, and between stereo camera 508 and ToF camera 510) according to a plurality of clocks. Data sync and buffer 516 can also buffer data as described in greater detail below. In terms of buffering functions, the buffer part of data sync and buffer 516 is configured to provide a moving window. This allows data processing to be performed on a portion of a frame when data are serially sent.

[0094] Optionally one or more input sensors 504 are asynchronous sensors. As a non-limiting example, an asynchronous sensor implementation for a camera does not send data at a fixed frame rate. Instead, such a sensor would only send data when a change had been detected, thereby only sending the change data.

[0095] Data may then pass to an RGB-D fusion chip process 518, the operation of which was described with regard to Figure 3, and which preprocesses the data for depth and RGB processing. Data can also pass to a sensor specific preprocess and control 520 for sensors other than stereo camera 508 and ToF camera 510, to prepare the sensor data for further use (for example, in regard to calibration of the data).

[0096] Next, data may pass to a layer of feature specific kernels 520, which receive data from RGB-D fusion chip process 518, and sensor specific preprocess and control 520. Feature specific kernels 520 may be operated according to the OPENCL standard, which supports communication between the FPGA and the CPU of the computational device operating user application 506 (not shown). Feature specific kernels 520 may also receive data directly from data sync and buffer 516, for example, to control the sensor acquisition and to provide feedback to data sync and buffer 516, to feed back to sensors 504.

[0097] Feature specific kernels 520, according to some embodiments, take data related to particular features of interest to be calculated, such as the previously described point cloud of 3D and RGB data, and calculate sub-features related to the feature. Non-limiting examples of such features may also include portions of processes as described herein, such as the de-mosaic process, color correction, white balance and the like. Each feature specific kernel 520 may have an associated buffer (not shown), which is preferably designed in order to provide a moving window. This allows data processing to be performed on a portion of a frame when data is serially sent.

[0098] Next, the sub-features can be passed to a plurality of fusion kernels 522, to fuse the sub-features into the actual features, such as the previously described point cloud of 3D and RGB data. Specific feature specific kernels 520 and fusion kernels 522 processes are described in greater detail below. Fusion kernel 522 can also report that a particular feature specific kernel 520 is missing information to the feature specific kernel that reports any missing information to sensors 504 through data sync and buffer 516. These features 526 may then be passed to user application 506 which may request specific features 526, for example, by enable specific fusion kernels 522, as needed for operation.

[0099] Among the advantages of calculation by feature specific kernels 520 and fusion kernels 522 according to some embodiments, is that both are implemented in the FPGA (field programmable array), and hence may be calculated very quickly. Both feature specific kernels 522 and fusion kernels 524 may be calculated by dedicated elements in the FPGA which can be specifically created or adjusted to operate very efficiently for these specific calculations. Even though features 526 may require intensive calculations, shifting such calculations, away from a computational device that operates user application 506 (not shown) and to the FPGA process 502, significantly increases the speed and efficiency of performing such calculations.

[0100] Optionally the layer of feature specific kernels 520 and/or the layer of fusion kernels 522 may be augmented or replaced by one or more neural networks. Such neural network(s) could be trained on sensor data and/or on the feature data from the layer of feature specific kernels 520.

[0101] Figure 5C shows the operation of the process 500 as it relates to additional external sensors 504 and output devices 530. Input from additional external sensors 504 may be transmitted to data sync and buffer 516, and then to a raw data processor 540, for example, for the display or other output device 530, that requires a raw pipe of data, optionally with minor modifications, to avoid sending all of the data to user application 506, which is operated by a slower computational device (thereby avoiding delay). Raw processor 540 could also optionally receive data from stereo camera 508 (not shown) as a raw feed. From raw data processor 540, the sensor input data can be sent to a user output controller 542 for being output to the user.

[0102] Output from user application 506 can also be sent to user output controller 542, and then to output devices 530. Non-limiting examples of output devices 530 include a tactile feedback device 532, a display 534, a sound output device 536 and optionally other output devices 538.

Display 534 can display visual information to the user, for example, as part of a head mounted device, for example for VR (virtual reality) and AR (augmented reality) applications. Similarly, other output devices 530 could provide feedback to the user, such as tactile feedback by tactile feedback device 532, as part of VR or AR applications.

[0103] Figure 5D shows the operation of a process 550 which features an additional stereo camera 508B and an additional ToF camera 510B. Stereo camera 508B and ToF camera 510B may be mounted on the head of the user as previously described. For this implementation, stereo camera 508B and ToF camera 510B would be the master devices, while stereo camera 508A and ToF camera 510A would be the slave devices. All of the devices would send their data to data sync and buffer 516; the process would then proceed as previously described. Again optionally only one of stereo camera 508B and ToF camera 510B is present, in which case preferably ToF camera 510B is present.

[0104] Figures 6A-6E show an exemplary, illustrative, non-limiting hardware system for the camera according to at least some embodiments of the present disclosure. Figure 6A shows the overall hardware system 600, featuring a plurality of layers 602, 604, 606 and 608. Layer 602 features a plurality of inputs. Layer 604 features FPGA hardware, which may optionally function as described with regard to Figure 5. Layer 606 relates to CPU hardware and associated accessories. Layer 608 relates to a host computer. Figure 6B shows layer 602 in more detail, including various inputs such as a stereo camera 609, featuring a left camera 610 and a right camera 612, which in this non-limiting example, feature 720 pixels and 60 fps (frames per second). Each of left camera 610 and right camera 612 may communicate with the FPGA (shown in the next layer) according to a standard such as MIPI (Mobile Industry Processor Interface) or parallel communication.

[0105] A depth sensor 614 is shown as a ToF camera, in this non-limiting example implemented as a QVGA (Quarter Video Graphics Array) camera operating at 60 fps, which communicates with the FPGA according to parallel communication. Audio input may be obtained from a stereo microphone 616 as shown. An inertial sensor 618 may be used to obtain position and orientation data. A radio-frequency (RF) receiver 620 may be used to collect data from other external sensors, which may be worn by the user for example, such as a bio sensor 622 and an AM (active marker) sensor 624, as previously described.

[0106] Figure 6C shows layer 604, which includes a FPGA 626, which may operate as described with regard to Figure 5. FPGA 626 may be implemented as an FPGA SoC SOM, which is a field-programmable gate array (FPGA) which features an entire system on a chip (SoC), including an operating system (so it is a “computer on a chip” or SOM – system on module). FPGA 626 includes a color preprocessing unit 628 which receives data from stereo camera 609, and which preprocesses the data as previously described, for example with regard to Figure 3. A depth preprocessing unit 630 receives depth data from depth sensor 614, and preprocesses the data as previously described, for example with regard to Figures 3 and 4.

[0107] A sensor config 646 optionally receives configuration information from stereo camera 609 and depth sensor 614, for example, to perform the previously described synchronization and calibration of Figure 3. Similarly, sensor config 646 optionally receives configuration information from the remaining sensors of layer 602, again to perform synchronization and calibration of the data, and also the state and settings of the sensors. Synchronization is controlled by a data sync module 648, which instructs all sensors as to when to capture and transmit data, and which also provides a timestamp for the data that is acquired. A route module 632 can receive input from stereo microphone 616, to convert data for output to USB 640 or data transceiver 644.

[0108] Inertial sensor 618 may communicate with FPGA 626 according to the I2C (Inter Integrated Circuit) protocol, so FPGA 626 includes an I2C port 634. Similarly, RF receiver 620 may communicate with FPGA 626 according to the UART (universal asynchronous receiver/transmitter) protocol, so FPGA 626 features a UART port 636. For outputs, FPGA 626 can include one and/or another of a MIPI port 638, a USB port 640, an Ethernet port 642 and a data transceiver 644.

[0109] Turning now to Figure 6D, the elements of layer 606 are shown, which can include one and/or another of a CPU 650, an Ethernet switch 652, and a USB transceiver 654. CPU 650 may handle calculations otherwise handled by FPGA 626 if the latter is temporarily unable to process further calculations, or to perform other functions, such as functions to assist the more efficient operation of a user application (which would be run by the host computer of layer 608). CPU 650 may be implemented as a SOM. Inputs to CPU 650 optionally include a CSI port 656 (for communicating with MIPI port 638 of FPGA 626); a USB port 658 (for communicating with USB port 640 of FPGA 626); an I2S 660 for transferring sound from the microphone; and UART/SPI

master 662 for providing the RF receiver data to the CPU processors.

[0110] Also shown in Figure 6D, a Bluetooth output 666 may be used to communicate with a Bluetooth port 678 of host computer 676 (shown in layer 608, Figure 6E). Similarly, a WiFi output 668 may be used to communicate with a WiFi port 680 of host computer 676. USB port 670 may be used to communicate with external accessories through their ports 672. HDMI 674 can also be available for display connection. Ethernet switch 652 may be configured to handle communication from any one or more of Ethernet port 642 of FPGA 626, Ethernet port 664 of CPU 650, and also Ethernet port 682, of host computer 676 (shown in layer 608, Figure 6E). Such communication may be bidirectional in these cases. Similarly USB transceiver 654 handles communication from data transceiver 644 of FPGA 626, as well as from USB port 684 of host computer 676 (shown in layer 608, Figure 6E). Such communication may be bidirectional in both cases. Figure 6E shows layer 608, the functions of which were previously described.

[0111] Figure 7 shows a non-limiting example of a method for stereo processing according to at least some embodiments of the present disclosure, the functionality of which may be contained within the FPGA of Figure 6. As shown, a process 700 can start with input from left RGB camera 702 and right RGB camera 704, of RGB data as previously described. Such input may be sent to a frame synchronizer 706, which synchronizes frames between the two cameras to eliminate time shift. This task may be performed in two stages. In a first stage, the input flows are sampled in such a way that they are synchronized with the same clock. In a second stage, a state machine detects which flow is in advance with respect to the other one so that it directs this flow toward a First Input First Output (FIFO). In this way, the first flow reaching frame synchronizer 706 is delayed until the other data flow reaches frame synchronizer 706 as well. Additional details are provided below.

[0112] A frame serializer 708 serializes the frame data as a sequence of bytes and the serialized data is passed to a stereo detect module 710, which performs the previously described “bad” or non-usable pixel detection. The data then undergoes a de-mosaic process 716, which is described in greater detail below and which involves constructing a complete set of color data from the incomplete color samples obtained previously. Thereafter, the data may then pass to a CCM (color correction matrix) process 718, described in greater detail below, which corrects deficiencies in the color data. Thereafter, the data may be adjusted for white balance in a white balance process

722, also described in greater detail below, and thereafter, can undergo a frame deserialization process 724 to restore the frame structure of the data.

[0113] Data from CCM process 718 can then be passed to a histogram process 720, which enables autoexposure and/or autogain adjustments (see below). Histogram data may be sent to an MCU 710, which performs any necessary adjustments to histogram process 720. MCU 710 also sends feedback to left RGB camera 702 and right RGB camera 704, to adjust their function according to the histogram data.

[0114] As shown in Figure 7, I2C 712 can be configured to control the register of the camera. An I2C is a multi-master, multi-slave, packet switched, single-ended, serial computer bus which is well known in the art.

[0115] Figure 8 shows a non-limiting example of a MCU (microcontroller, i.e., a processor) configuration according to at least some embodiments of the present disclosure. Optionally, a similar configuration could be used for a CPU structure (additionally or alternatively). As shown, MCU 710, which may for example be implemented with the process of Figure 7, features a bus 800, which is connected to a master 802 and a plurality of slave units 804, shown as slave units 804a to 804e, which handle custom parameters to communicate with custom cores. The custom cores can, for example, be used for RGB preprocessing, to configure and control the various components and functions of the RGB preprocessing (as previously described). MCU 710 can also be configured to control each kernel as previously described with regard to Figure 5.

[0116] Master 802 may be implemented by using, for example, the Lattice Semiconductors™ product, in which case the GPIO (General Purpose Input Output) core is implemented for slave units 804. Bus 800 may be implemented according to the Wishbone protocol, which is an open source interconnect architecture maintained by OpenCores organization (<https://opencores.org/opencores.wishbone>).

[0117] Configurable parameters can be sent to custom cores by means of the hardware implemented processor, e.g., LatticeMico32™ as master 802, which is based on a 32-bit Harvard RISC architecture and the open bus WISHBONE. For communication within MCU 710, such communication always occurs between a MASTER interface and a SLAVE interface. In some embodiments, only MASTER unit 802 can begin communications. Master unit 802 performs a handshake with slave 804 through bus 800, after which communication can occur.

[0118] Figure 9 shows illustrative aspects of an example of a camera according to at least some embodiments of the present disclosure, including a camera readout schematic 900, a frame active area 902, horizontal blanking 904, vertical blanking 906 and horizontal/vertical blanking 908.

[0119] Figure 10 shows a non-limiting example of a configuration for double clock sampler functions according to at least some embodiments of the present disclosure. Such functions are desirable because of the need to synchronize different clocks, for example between the right and left cameras as described herein. In order to perform clock synchronization, a double clock module 1000 is provided in which a first layer of registers (Xreg1 (1002) and Yreg1 (1008)) sample data from the right camera (not shown) using its own clock signal (clk_Rt), while second layer of registers (Xreg2 (1004) and Yreg2 (1010)) sample data from the left camera (not shown) using left clock instead (clk_Lt). The left clock can be used as the overall module clock for double clock module 1000. Signal sel (1006) alternatively activates a pair of registers Xreg1-Yreg2 or Yreg1-Xreg2. In this way, data has time to reach stable state in first layer before to be sampled by second one. Finally data can be synchronized to the first left camera clock when outputted from multiplexer, which selector is connected to the sel signal from 1006.

[0120] Figures 11A and 11B show non-limiting buffer configurations according to at least some embodiments of the present disclosure, which for example may be used to fulfill the buffer requirements of the FPGA and/or optionally of various modules as described herein. Figure 11A shows an exemplary buffer configuration 1100, featuring muxs, highlighted in circles, generating int2_2_2 (1102), int2_2_3 (1104), and int2_2_4 (1106) signals, which are replications of moving window cells. When a moving window has its center placed on the edge of a frame, outside corner information may be missing from the frame. For this reason, the replication of the last 2 rings can be chosen as the strategy to avoid data loss. Replication includes providing the same information to more than one cell of a moving window a plurality of times, which can be accomplished by using muxs, as shown in Figure 11A. Such a buffering system is used for example for the de-mosaic and detect modules.

[0121] Figure 11B shows a portion of the internal structure of the buffer module 1150 - the first lines of the internal structure of buffer module. The mask is realized with registers 1152, while the rest of the line makes use of EBRs 1154.

[0122] The moving windows can comprise data registers 1152, which allows moving mask to have

all cells accessible at same time. The remaining part of each line may be realized with EBRs 1154, which behave as FIFO registers. Each EBR 1154 preferably comprise 18 Kbit RAM. According to available memory configuration, this buffer is capable of handling a frame having maximum width of 2053 pixels (2 EBRs 1154 per line are adopted in configuration 1024×18). In order to maintain original synchronization, FV and LV signals entering in the buffer have to be properly delayed at output. In some embodiments, the first entering pixel through pix_in input comes out from pix_TEST after about 2 frame lines (see Figure 12). FV and LV time shift is achieved by using EBRs 1154 and a control state machine. The control state machine could be implemented for example as shown in Figure 11A; and may be configured to control a counter connected to read/write address input of EBRs 1154.

[0123] Figures 12A-C show non-limiting examples of an internal buffer cells arrangement. Figure 12A shows a global structure 1200 in which the previously described EBRs are implemented as embedded block RAM 1202. A general mask 1204 is shown as implemented in LUT4 based cells including mask cells (gray) and cells that are not externally accessible (blue). The specific details of the mask cells vary according to the functions of the mask. For example, Figure 12B shows a non-limiting mask 1220 for defective pixel detection, which is performed as previously described. Figure 12C shows a non-limiting mask 1222 for the de-mosaic task, which is performed as previously described and also as described below.

[0124] Figure 12D shows exemplary state machines of the output synchronization signals, according to some embodiments. Diagrams of state machines 1250 and 1252 show the waveforms of sync signals and logic state of the controller. State machine 1250 relates to states of state-machine handling sync signals delay, while state machine 1252 relates to FV and LV signals at the output of time shift EBR. Each state machine starts in W_H_FV, waiting for FV being asserted. Once this occurs, the state changes to W_H_FV_LV, to wait for both FV and LV being asserted. When FV and LV are in high state, a counter is started keeping track of how many clocks are needed to delay signals of 2 frame lines. This can be achieved by alternatively jumping between W_DeIFVCount2 and W_DeIFVCount2bis (which increments the counter). The counter stops when two entire rows (horizontal blanking included) are output. The number of clocks can be stored in register MAXADDSYNC. In order to take into account possible resolution changes when a new frame starts, the state machine always resets the counter to update MAXADDSYNC.

[0125] Figures 13A-13H show non-limiting examples related to a method for performing the de-mosaic task according to at least some embodiments of the present disclosure, involving constructing a complete set of color data from the incomplete color samples obtained previously. This module uses moving windows to perform its task and is equipped with a buffer module to coordinate signals used to identify the formula to apply on pixel under test. In particular the cases incurred are:

- R pixels: G and B values will be calculated;
- G pixels at rows containing R pixels: R and B values will be calculated;
- G pixels at rows containing B pixels: R and B values will be calculated;
- B pixels: G and R values will be calculated.

[0126] The operation of the de-mosaic module (described below), but briefly a set of formulas are given below. Figures 13A and B show masks on which the algorithm is performed, including in Figure 13A G values at R(B) place or B(R) values at R(B) place; and in Figure 13B, R(B) at G places. B(R) values in R(B) sites, fig. 13A:

$$B_d/R_d = G_d + \frac{1}{4}(2_2 + 2_4 + 4_2 + 4_4) - \frac{1}{4}(2_3 + 4_3) - \frac{1}{4}(3_2 + 3_4)$$

[0070] G values in R(B) sites, fig. 13 A:

$$G_d = \frac{1}{2}TEST + \frac{1}{4}(2_3 + 4_3) + \frac{1}{4}(3_2 + 3_4) - \frac{1}{8}(1_3 + 5_3) - \frac{1}{8}(3_1 + 3_5)$$

[0071] R(B) in RG(BG) rows at G sites, fig. 13B:

$$R_{d_rg}/B_{d_bg} = \frac{1}{2}TEST + \frac{1}{2}(3_2 + 3_4) - \frac{1}{8}(2_2 + 2_4 + 4_2 + 4_4) + \\ - \frac{1}{8}(3_1 + 3_5) + \frac{1}{4}TEST$$

[0072] Figure 13C shows the de-mosaic algorithm in an exemplary implementation, in more detail, to determine the missing green values. This implementation simplifies multiplications and division by reducing them to only shift operations. A de-mosaic process 1300 starts with classifying a pixel 1302. For the value of G (green) at R (red) and B (blue) sites (classification a),

matrix A is used in 1304. All matrices are shown in Figure 13D. The convolution matrices shown as matrices B1 and B2 are used for classification b, for R(B) at B-G(R-G) sites (matrix B1) and for R(B) at R-G(B-G) sites (matrix B2), to take the average of the green pixels surrounding the red and blue sites in 1306 and to apply the convolution matrices B1 and B2 in 1308. The method as performed on the pixels is shown in Figure 13E.

[0073] The remaining classification is classification c, in which the number of green pixel values is reduced to fit in a 5x5 window in 1310, and matrix C is applied as the convolution matrix in 1312. This classification is applied for R(B) at B(R) sites, which are the remaining cases. The method as performed on the pixels is shown in Figure 13F.

[0074] Figure 13G shows the checking phase of summation obtained at a numerator of formulas used in a de-mosaicing process, for handling truncation, according to some embodiments. When maximum pixel width is adopted (12 bits) pixel, values can range from 0 up to 4095. In order to avoid premature truncation during partial calculations steps, each term can be carefully sized as to contain signed summations. Nevertheless, under certain conditions, overflow or underflow may occur in final result, hence a truncation mechanism can be required.

[0075] Process 1354 features a truncation mechanism, in the last calculation phase: a vector 1356 containing the summation resulting from operation performed on numerator of one of the above equations for de-mosaicing, which is right shifted. The control may be performed on the most left bits 1358 just before final color value begins. First, it is determined whether the bits are all equal to zero, so as to ensure that the result is on the correct range. As 2's complement convention is used for negative number representation, if first bit is 1 the final value will be set to 0 (as a negative color value does not make sense). On the other hand, if first bit is null, but the others bits preceding final result interval are not all zero, then result incurred in overflow. In this case, the final result will be truncated to 4095 (if 12 bits format is used). The final color value is shown in 1360, while suppressed bits are shown in 1362.

[0076] Figure 13H shows a non-limiting example of a DSP configuration for RGB processing as described above according to at least some embodiments of the present disclosure. Accordingly, a configuration 1370 features DSP modules red-blue_sum 1372 and 4_sum 1374 which are optimized adders accepting 8 and 4 terms respectively. A trivial shift by 8 (1378a) allows to obtain the average at R and B pixel sites. On the other hand, average of G sites is obtained through

green_mult-sum 1376 which sums the results of the two adders and multiplies their result by aav; a shift (1378b) finalizes the calculation. Depending on the color of the investigated pixel, sel signal 1380 assumes high or low logic state to select the right average to compare it with. The average in G sites requires more process stage cycles than for R/B ones. Hence in order to obtain both results at same time, the latter average is delayed using register sequence, controlled by 1380.

[0077] Figure 14 shows a non-limiting of a method for white balance correction according to at least some embodiments of the present disclosure, showing a state machine time diagram for coefficient updating in the white balance module. To this end, a white balance algorithm, e.g., the GW (gray world) algorithm, assumes that in a normal well color balanced photo, the average of all the colors is a neutral gray. Therefore, the illuminant color cast can be estimated by looking at the average color and comparing it to gray (see <https://web.stanford.edu/~sujason/ColorBalancing/grayworld.html> for a detailed explanation and implementation). However, while the computational simplicity associated therewith is attractive, the present inventors found that the GW algorithm did not provide sufficiently robust results, in particular, proving to be unstable under certain circumstances. Instead, a smoothed GW algorithm was chosen to implement the white balance module.

[0078] The smoothed GW algorithm was implemented according to the following equations:

$$\begin{cases} \text{corr}_{R,i} = \text{corr}_{R,i-1} \\ \text{corr}_{B,i} = \text{corr}_{B,i-1} \end{cases} \quad \text{if } |d_{RG}| = |d_{BG}| = 0$$

$$\begin{cases} \text{corr}_{R,i} = \text{corr}_{R,i-1} \\ \text{corr}_{B,i} = \text{corr}_{B,i-1} + \mu \times \text{sign}(-d_{BG}) \end{cases} \quad \text{if } |d_{BG}| \geq |d_{RG}|$$

$$\begin{cases} \text{corr}_{R,i} = \text{corr}_{R,i-1} + \mu \times \text{sign}(-d_{RG}) \\ \text{corr}_{B,i} = \text{corr}_{B,i-1} \end{cases} \quad \text{if } |d_{BG}| < |d_{RG}|$$

Where $d_{RG} = \bar{R} - \bar{G}$ and $d_{BG} = \bar{B} - \bar{G}$.

[0079] Per channel frame average can be obtained by using a DSP adder in self-accumulation configuration (as shown), which can be activated only when both synchronization signals (FV_whb and LV_whb) are in high logic state, so that only valid pixel values are added. Obtained

summation can then be divided by total number of pixels composing a frame. Co-efficients n_{av} and a_{av} are chosen by running a function in Scilab called $nAvMinErr()$, which need the number of bits to represent a pixel and the resolution of used camera. Averages are calculated on corrected channels, in order to have a feedback on the effect of last values assumed by coefficients. Each coefficient is initialized to 1 in order to directly estimate real image situation. A state machine can be implemented as to adjust multiplying coefficients during vertical blanking time intervals (FV_whb at logic '0'), its associated time diagram being depicted in Figure 14.

[0080] The adjustment of coefficients, according to which the R and B channels are multiplied, requires few clock cycles, and it is performed at the end of a frame, right after FV_whb goes to logic '0'. Here two states follow one another: AV_CALC causes finalization of the calculation of averages, UPDATE allows the update of both coefficients. Comparison of averages B and R toward G can be done in parallel. During remaining time, state machine stays in W_L_FV or W_H_FV states in order to catch the end and the beginning of a frame.

[0081] Multiplication of R and B channels can be performed converting to fixed point convention (multiplication by 2^{nres} , with $nres$ number of fractional digits) followed by integer part selection, by taking off fractional digits (right shift). The minimum possible step increment may be 0.001, preferably up to and including 0.01). The closest resolution obtainable is 0.000977 using $nres = 10$. To ensure a good range, the integer part is fixed to two bits (3 is the maximum integer part can be represented). Moreover, as the adjustment can be both an increment or a decrease, an additional bit for 2's complement representation is needed. Hence $ampl_step$ input is 10 bits wide.

[0082] Figure 15 shows a non-limiting example of a method for performing the histogram adjustment according to at least some embodiments of the present disclosure. The functions are shown as being performed on MCU 710, while the histogram functionality may, for example, be provided for histogram 720. As shown, the process can be controlled by a control 1500. Luminance can be calculated by a luminance calculation module 1502 as previously described.

[0083] A classification module 1504 classifies each pixel according to a different range of luminances, as the histogram is configured to show a set of ranges of such luminances. The histogram application therefore involves the classification of each pixel according to its relevant luminance range. The classified pixel may then be stored in a memory 1506, from which the data may be retrieved for use in other procedures. To permit both the FPGA (not shown) and MCU 710

to access the luminance data, a pseudo dual port RAM may be used to updates the luminance data (not shown).

[0084] Figure 16 shows an illustrative, exemplary, non-limiting process for stereo rectification according to at least some embodiments of the present disclosure. The method is optionally implemented as an inverse mapping algorithm that computes for each pixel coordinates in the rectified image the corresponding pixels coordinates in the raw, unrectified and distorted image. Let r and c be the pixel coordinates in the rectified image.

[0085] As shown, a method 1600 begins in stage 1602 with computing the projection of the rectified image on the aligned camera reference frame through the new camera matrix computed with the intrinsic parameters (focal length and principal point) and the extrinsic parameters (rotation matrix and translation vector).

[0086] Let Pose be a matrix resulting from the computation of a matrix composed of the intrinsic camera parameters and of a matrix composed from the rotation and the translation matrixes between the 2 cameras. So the projection is:

$$\begin{pmatrix} ray\ 1 \\ ray\ 2 \\ ray\ 3 \end{pmatrix} = Pose \cdot \begin{pmatrix} c \\ r \\ 1 \end{pmatrix}$$

[0087] From this point, the pixels coordinates of the projection of the r and c pixel coordinates on the new coordinates system become:

$$r_{new} = \frac{ray\ 2}{ray\ 3} \qquad c_{new} = \frac{ray\ 1}{ray\ 3}$$

[0088] Stage 1604 includes correcting the distortion of the lenses of the cameras with their distortion parameters.

[0089] With $q^2 = r_{new}^2 + c_{new}^2$, the radial distortion is taken in account in this way:

$$\begin{pmatrix} r_r \\ c_r \end{pmatrix} = \frac{(1+k_1 \cdot q^2 + k_2 \cdot q^4 + k_3 \cdot q^6)}{(1+k_4 \cdot q^2 + k_5 \cdot q^4 + k_6 \cdot q^6)} \begin{pmatrix} r_{new} \\ c_{new} \end{pmatrix}$$

[0090] The tangential distortion is taken in account in this way:

$$\begin{pmatrix} r_t \\ c_t \end{pmatrix} = \begin{pmatrix} p_2 \cdot (q^2 + 2 \cdot r_{new}^2) + 2 \cdot p_1 \cdot c_{new} \cdot r_{new} \\ 2 \cdot p_2 \cdot c_{new} \cdot r_{new} + p_1 \cdot (q^2 + 2 \cdot c_{new}^2) \end{pmatrix}$$

[0091] Finally, the undistorted pixel coordinates are the sum of the radial and the tangential distortion computations:

$$\begin{pmatrix} r_{undist} \\ c_{undist} \end{pmatrix} = \begin{pmatrix} r_r + r_t \\ c_r + c_t \end{pmatrix}$$

[0092] Stage 1606 includes projecting the undistorted pixel coordinates on the real camera reference frame using the KK camera matrix. This matrix is defined as follows:

$$KK = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

[0093] So the final pixel coordinates are:

$$\begin{pmatrix} r_p \\ c_p \end{pmatrix} = KK \cdot \begin{pmatrix} r_{undist} \\ c_{undist} \end{pmatrix}$$

[0094] Figure 17A shows an illustrative, exemplary, non-limiting system for stereo rectification according to at least some embodiments of the present disclosure. A system 1700 features a mapper 1702, a memory management unit 1704, a bilinear interpolator 1706 and a finite state machine 1708.

[0095] Mapper 1702 is in charge of executing the rectification algorithm and generating the rectified pixel coordinates. The operation of mapper 1702 is described in more detail in Figure 17B.

[0096] The purpose of the Memory Management Unit 1704, in some embodiments, is to first store the incoming raw pixels, and second, to output the pixels corresponding to the rectified pixels coordinates given by the Mapper 1702. The operation of Memory Management Unit 1704 is described in more detail in Figure 17C.

[0097] The Bilinear Interpolator 1706 may be used to compute the bilinear interpolation of 4 pixels. The rectified pixels coordinates aim at 4 pixels as they are non-integer. A strategy to retrieve a value for the rectified pixel could be to choose one pixel among these 4 but to be as accurate as possible, a better strategy is to compute the bilinear interpolation of these 4 pixel values according the relative position of the rectified pixel among these 4 pixels. The following equation describes this operation:

$$I_{\text{pix_out}} = (1 - c_{\text{p.f}} \quad c_{\text{p.f}}) \begin{pmatrix} I_{\text{sw}} & I_{\text{ss}} \\ I_{\text{sw}} & I_{\text{se}} \end{pmatrix} \begin{pmatrix} 1 - r_{\text{p.f}} \\ c_{\text{p.f}} \end{pmatrix}$$

[0098] Hence, this block takes as inputs the 4 four pixels values pointing by the rectified coordinates as well as the fractional parts of these rectified coordinates and outputs the pixel value out the rectified pixel as their bilinear interpolation.

[0099] A Finite-State Machine 1708 may be used to control the block(s) according to, for example, an imposed 1280*720p @ 60 fps protocol (the Line Valid and Frame Valid signals define this protocol). For example, at the beginning of a sequence, the state machine is in a WAITING state and waits for a new frame to arrive, i.e. for a rise of the Frame Valid and Line Valid signals. When this happens, the state machine proceed to a MMU_FILLING state. In this state, a Writing Controller can be enabled and can fill the first buffer with the incoming pixels at the input Live Valid signal rate. When this buffer is full, a control signal may be raised to the state machine to proceed into a COMPUTING state to enable the Mapper, for initiating computation of the rectified pixels coordinates. When the first rectified pixels coordinates are computed, the state machine proceeds to a RECTIFYING state and a reading process starts with the Coord2memAddr_converter being enabled. There may be two (2) additional states,

LV_DELAYING and FV_DELAYING, which may be used to make sure that the outputted frame will follow the same protocol as the input frame.

[0100] Figure 17B shows an illustrative, exemplary, non-limiting mapper module for use with the system of Figure 17A according to at least some embodiments of the present disclosure, and Figure 17F shows a non-limiting, exemplary finite state machine for use with the system of Figure 17A according to at least some embodiments of the present disclosure.

[0101] With respect to Figure 17B, mapper 1702 may feature four blocks: a coordinates generator 1720, a projection 1 (shown as 1722), an undistortion module 1724 and a projection 2 (shown as 1726). The first block, the Coordinates Generator 1720, corresponds to the generation of all possible pixels coordinates in row order from (1,1) (top-left of the image) to (720,1280) (bottom-right of the image) at the pixel clock rate (i.e., it can be considered a counter). Then, these coordinates will be the inputs of the 3 remaining blocks that correspond to the 3 steps of the rectification algorithm of Figure 16. This block therefore may be used to output rectified pixels coordinates, and according to all the calculus the rectification algorithm requires, the rectified pixels coordinates are non-integer. Hence, and in order to facilitate both the work of the Bilinear Interpolator and the work of the Coord2memAddr_converter, the Mapper separates the integer (r_p_i and c_p_i) and the fractional (r_p_f and c_p_f) parts of the rectified pixels coordinates.

[0102] Figure 17C shows an illustrative, exemplary, non-limiting memory management for use with the system of Figure 17A according to at least some embodiments of the present disclosure. As shown, memory management unit 1704 can perform two processes (and at least one), including storing the incoming pixels of the image at the pixel clock rate and being able to make these stored pixels available at any time for the bilinear interpolator. Hence, these 2 processes can be seen as a writing process and a reading process.

[0103] In order to avoid data corruption, the buffering process can use a “ping-pong” scheme so that while data is being written in one buffer, and data can be read into the other buffer. A change of buffer can occur every time the writing process reach the end of a buffer. With this scheme, the architecture starts filling one buffer as soon as it receives the first pixels of an image (indicated by the FV and LV signals) and waits for this buffer to be full before starting to rectify the first pixels coordinates and allow the reading process to read in this buffer. A small delay may be therefore added at the launching of the architecture, but then the latter may be able to output pixels at the

requested frame rate.

[0104] As the rectified pixels coordinates are non-integer, and as four pixels from the unrectified image are needed at the same time to interpolate the intensity of one rectified pixel, four dual-port memories can be used in each buffer so that four pixels at the same clock cycle may be output when requested. To insure that the 4 adjacent pixels targeted by the non-integer pixel coordinates are situated in different dual-port memories, pixels may be simply cyclically stored in the 4 memories following the row order.

[0105] An illustrative example of how this operates is shown in Figures 17D and 17E. Figure 17D shows a non-limiting example of an image. Figure 17E shows the memory filling scheme for this image.

[0106] If the pixel coordinates couple requested by the Mapper is the green point on the image (shown in Figures 17D and 17E as a non-limiting example), then the four pixels that need to be interpolated are the pixels p8, p9, p26 and p27. This can be done at the same clock cycle since they are all in different memory: p8 is in m3, p9 is in m0, p26 is in m1 and p27 is in m2. This process of filling the memories, in some embodiments, works upon the width of the image being a multiple of 6 and not multiple of 4. Since 1280 is multiple of 4, a padding process may be used to “fake” an image width of 1290. This way, the adjacent pixels may always be located in different memories and the process remains simple for the reading process.

[0107] The writing process may be managed by the Writing Controller which can generate the writing addresses of the four memories and cyclically activates their write enable signals while skipping the addresses that need to be to fit with the padding process. A demultiplexer may then be used to redirect the write enable signals to the right buffer (the one that is currently in the writing process).

[0108] The reading process is managed by the Coord2memAddr_converter, which may be used to turn pixel coordinates couples coming from the Mapper into reading memory addresses for the Bilinear Interpolator (BI) -- the four pixels values required to compute the rectified pixel value. The BI is facilitated by cyclically storing the pixels because, from a pixel coordinates couple, the BI need merely compute the linear address, and then divide it by 4 (for example). This calculus is described below:

$$\begin{aligned}
 NW_{addr} &= \left\lfloor \frac{(r_{p_i-1}) * ImageWidth + (c_{p_i-1})}{4} \right\rfloor \\
 NE_{addr} &= \left\lfloor \frac{(r_{p_i-1}) * ImageWidth + (c_{p_i-1}) + 1}{4} \right\rfloor \\
 SW_{addr} &= \left\lfloor \frac{(r_{p_i-1}) * ImageWidth + (c_{p_i-1}) + ImageWidth}{4} \right\rfloor \\
 SE_{addr} &= \left\lfloor \frac{(r_{p_i-1}) * ImageWidth + (c_{p_i-1}) + ImageWidth + 1}{4} \right\rfloor
 \end{aligned}$$

[0109] Based upon Figures 17D and 17E, the calculation would be performed as follows:

$$NW_{addr} = \left\lfloor \frac{0 * 18 + 7}{4} \right\rfloor = 1 \quad NE_{addr} = \left\lfloor \frac{0 * 18 + 7 + 1}{4} \right\rfloor = 2 \quad SW_{addr} = \left\lfloor \frac{1 * 18 + 7}{4} \right\rfloor = 6 \quad SE_{addr} = \left\lfloor \frac{1 * 18 + 7 + 1}{4} \right\rfloor = 6$$

[0110] As shown, p8 that is in m3 is at the linear address 1, p9 in m0 is at linear address 2, and p26 and p27 both are at linear address 6 in m1 and m2 respectively. In this architecture, using the padding process, ImageWidth is replaced by the width of the padded image, 1290 in the present case, so that the memory addresses skipped by the Writing Controller during the writing process may the never be achieved.

[0111] Also, in order to know which memory corresponds to which linear address, a modulo 4 operation may be computed on the column number (c_p_i). This information may also benefit the Router block that match the incoming pixels value from m0, m1, m2 and m3, with their position in the image (which may be important for the bilinear interpolation).

[0112] The linear addresses computed with the above equations may comprise absolute addresses (according to some embodiments). Indeed, in some embodiments, the processes work for buffer size being the same as an entire image. However, since the buffer size may comprise several lines, the Coord2memAddr_converter requires the identification of the line which is currently stored at the beginning of the buffer, so that the linear absolute addresses may be processed into relative addresses. Such information may be provided by the Writing Controller through a first_row signal. Thus, the process, in some embodiments, should take this into account by, for example, subtracting the numerator by this signal.

[0113] Figure 18A shows an illustrative, exemplary, non-limiting disparity map method according

to at least some embodiments of the present disclosure. Once the stereo images are rectified, matching becomes a one-dimensional issue and the disparity map can be computed. Accordingly, the following is a non-limiting exemplary process for producing a disparity map.

[0114] Step 1: Matching Cost Computation. In this step, the similarity of pixels in left and right image are measured by producing a cost. Various non-limiting, exemplary algorithms are described below.

[0115] Absolute Differences (AD)

$$AD(x, y, d) = |L(x, y) - R(x-d, y)|$$

[0116] This algorithm can be used to compute the absolute difference of a pixel in the left image and a pixel in the right image on the same row, and with an offset in the column index (corresponding to the disparity). It has a low complexity due to its simplicity but does not produce smooth disparity map for highly textured images.

[0117] Squared Differences (SD)

$$SD(x, y, d) = (L(x, y) - R(x-d, y))^2$$

[0118] This algorithm is very similar to the Absolute Differences by its definition and by its results in term of speed and accuracy. It also can be used to compute the difference of the intensity of a pixel in the left image and a pixel in the right image and then elevates it to the power of 2. AD and SD produce almost the same disparity maps.

[0119] Sum of Absolute Differences (SAD)

$$SAD(x, y, d) = \sum_{i,j \in \omega} |L(i, j) - R(i-d, j)|$$

[0120] This algorithm gathers data as in step 1 and step 2 of the taxonomy (above), in one step. Indeed, this algorithm is the same as the AD, it operates on a square window around the pixel of interest. Therefore, it has a bigger computational time than the AD, but it smooths the disparity map produced due to the window-based method which acts like a filter and it decreases the error rate of the disparity map produced by better finding some occluded disparities.

[0121] Sum of Squared Differences (SSD)

$$SSD(x, y, d) = \sum_{(i,j) \in \omega} (L(i, j) - R(i-d, j))^2$$

[0122] The SSD is to the SD, as the SAD is to the AD. Again, the SAD and the SSD are very similar and produce almost the same disparity maps.

[0123] Normalized Cross Correlation (NCC)

$$NCC(x, y, d) = \frac{\sum_{(i,j) \in \omega} L(i, j) \cdot R(i-d, j)}{\sqrt{\sum_{(i,j) \in \omega} L(i, j)^2 \cdot R(i-d, j)^2}}$$

[0124] If an algorithm that computes the disparity based on the intensity of the pixels in the images is used with images that come from cameras that do not have the same gain and/or bias, the produced disparity map can be incorrect. Thus, to compensate for differences in gain and/or bias, the normalized cross correlation algorithm can be applied. It normalizes the intensity of the pixels from the left and the right images so that a difference in gain and/or bias does not come into account anymore. Accordingly, this algorithm may be required if the cameras do not have the same gain/bias, but it can blur regions of discontinuity and also requires considerable computational resources to obtain a high- accuracy disparity map.

[0125] Census Transform (CT)

$$CT(x, y, d) = \text{Hamming}(\text{Census}_L(x, y), \text{Census}_R(x-d, y))$$

With:

$$\text{Census}(x, y) = \text{bitstring}_{(i,j) \in \omega} \{I(i, j) \geq I(x, y)\}$$

[0126] This algorithm is based the Census transform and it computes a bitstring based on a square window centered on the pixel of interest and where each bit of this bitstring is the result of the comparison between the intensity of a pixel inside the window and the intensity of the pixel of interest. The Hamming distance between the Census transform computed in the left image and the Census transform computed in the right image is performed and considered, but may be at a cost. This algorithm is robust to disparity discontinuities and it can show very high matching quality at

object borders. However, in some embodiments, it may produce incorrect matching in regions with repetitive structures.

[0127] Mini-Census Transform (miniCT)

[0128] This algorithm is the same as the Census transform, though the a different window on which the algorithm operates. In the mini-Census transform, the bitstring is not computed on a square window, but rather, on a cross centered on the pixel of interest. The resulting bitstring is 6-bit long (2 bits up and 2 bits down the pixel of interest and 1 pixel left with an offset of 1 and 1 pixel right with an offset of 1). This cross with an example of the application of the algorithm is shown in Figure 18B, which shows an illustrative, exemplary, non-limiting method for calculating a cost for the disparity map method according to at least some embodiments of the present disclosure.

[0129] Step 2: Cost Aggregation

[0130] From step 1, a 3-D costs map is produced. Indeed, for each pixel in the image, a cost is computed for each disparity (shift between the 2 images). But these costs can be considered as raw (except for some algorithms) since they are computed with local information. In order to minimize the matching uncertainties, the step 2 aggregates the raw costs according to several possible schemes.

[0131] Furthermore, only local methods will be described here as global methods often skip this step. Local methods are window based methods and the disparity of each pixel depends only on the intensity values of the surroundings pixels within the predefined window. Hence, as this method takes in account only local information, it has a low computational complexity and a short run time so that architectures implementing it can be real-time (sometimes using additional hardware). Finally, local methods use all 4 steps of the process.

[0132] Global methods are, in contrast, methods that generate a disparity map that optimizes a global energy function. This global energy function contains mainly 2 terms. One penalizes disparity variations and the other measures the pixel similarity. Global methods has a high computational complexity and a longer run time than local methods. By the way, software-based global methods are almost impossible to be implemented in a real-time architecture so additional hardware would be needed to address this constraint. Another difference with local methods is that

global methods usually skip step 2 of the 4-step process.

[0133] Turning back to cost aggregation, these methods aggregate the matching cost by summing them over a support region which is usually a square window centered on the current pixel of interest. The simplest aggregation method is to apply a low-pass filter in the square support window. This window can be fixed-size (FW) but the error rate increases when the size of this window becomes too big and the parameters must fit the particular input dataset. Or this window can also be adaptive (AW), in terms of size, or in terms of weight: adaptive support weight (ASW), or there can be multiple windows (MW). The MW technique shows weaknesses at objects boundaries but the AW technique reduced the errors caused by boundary problems. AW can achieve high quality results near depth discontinuities and in homogenous regions. The ASW technique first computes for each pixel an adaptive cross based on its intensity similarity to its consecutive neighbours in the four directions. Then the adaptive support weight window on which the raw costs will be sum over is created by merging the horizontal arms of the cross of its vertical neighbours.

[0134] This technique is said to produce quality results of the generated disparity map but may be time consuming than the fixed-size (FW) technique for instance.

[0135] Step 3: Disparity Selection

[0136] Now that the costs are aggregated and that the matching uncertainties have been addressed, it is time to go from this 3-D aggregated costs map to a 2-D disparity map. In other words, it is time to find for each pixel the correct disparity among all the disparities that were used to build this 3-D costs map.

[0137] As local and global methods exist for this step, both will be described briefly.

[0138] For the local methods, the most used disparity selection method is a Winner Takes All (WTA) strategy so that the disparity $d(x,y)$ for each pixel corresponds to the minimum aggregated cost in the range of the aggregated cost obtained after step 2 (or step 1 if step 2 skipped) over all allowed disparities (D):

$$d(x,y) = d|C = \min_{d \in D} Cost(x,y,d)$$

Where $D = [\min_disp, \max_disp]$ is the range of shift used in step 1 and 2.

[0139] This method works for the algorithms described in step 1, except for the normalized cross correlation (NCC) where the Winner Takes All method consists of choosing the disparity that corresponds to the maximum aggregated cost.

[0140] For global methods, a global energy function may be used:

$$E(d) = E_{\text{data}}(d) + \beta \cdot E_{\text{smooth}}(d)$$

Where $E_{\text{data}}(d(x,y))$ is the matching cost of pixel (x,y) , β is a weighting factor and $E_{\text{smooth}}(d(x,y))$ penalizes the disparity variation.

[0141] Some algorithms that perform this disparity selection as global methods are:

- belief propagation (BP)
- graph cut (GC)
- dynamic programming (DP)

As previously noted, the local method can be retained for this step also.

[0142] Step 4: Disparity Refinement

[0143] In this step, the goal is to reduce noise generated through the previous steps and to refine the final disparity map. Among known techniques to do so include:

- Gaussian convolution: reduces noise in the disparity map and can also reduce the amount of fine detail. Disparity is estimated using the one of neighboring pixels in compliance with weights of a Gaussian distribution
- Median filter: removes small and isolated mismatches in disparity. Low computational complexity
- Anisotropic diffusion: Applies smoothing without crossing any edges, unlike Gaussian convolution

[0144] These techniques are quite similar in their concept. Another way of improving the quality of the produced disparity map, according to some embodiments, is by doing a consistency check. In some embodiments, 2 disparity maps can be computed from the same stereo image pair. One by looking for matching pixels of the left image in the right image, and another by looking for the

matching pixels of the right image in the left image. Due to at least occlusions, these 2 disparity maps of a same stereo image pair will not be the same. But with these 2 disparity maps, a left to right consistency check (LRC) can be performed in order to detect outliers and then several strategies exist to try to refine them.

[0145] This left to right consistency check consists of checking all the pixels in the left disparity map if the disparities correspond to the disparities in the right disparity map. For instance, let k be the disparity in the left disparity map at pixel (x,y) : $DL(x,y) = k$. This means that pixel (x,y) in left original image best corresponds to pixel $(x-k,y)$ in right original image when the disparity map is computed for the left image. On the other hand, it can be expected that pixel $(x-k,y)$ in right original image best corresponds to pixel (x,y) in left original image when the disparity map is computed for the right image. Which can be expressed as: $DR(x-k,y) = k$. Thus, if $DL(x,y) = k$ and $DR(x-k,y) = k$ then disparity at pixel (x,y) in left disparity map can be considered as correct. Otherwise disparity at pixel (x,y) in left disparity map is considered as an outlier.

[0146] This LRC permits to detect occlusion problems or simple mismatches and several strategies to address the problems/mismatches are highlighted. For example, the non-trusted disparity may be interpolated with the neighbor disparities if such is considered as correct and if the neighboring pixels have a similar intensity to the pixel corresponding to this non-trusted disparity in the original image. Outliers can also be dealt with by using the information of another technique to determine the depth of a scene like using the data coming from a Time-of-Flight sensor for instance.

[0147] Various of these algorithms and methods have been tested. In certain instances, it has been found that for step 3, the Winner-Take-All method provided the best results, including with regard to simplicity. For step 1, the two best algorithms were found to be the AD algorithm and the SAD algorithm. In some embodiments, the AD algorithm was enhanced. In step 1, the matching cost computation, instead of computing the absolute differences of only one pixel in the left image and one pixel in the right image, in this improved version, the absolute differences of 2 consecutive pixels are computed. Then, knowing that the disparity that produces the smallest cost will be selected as the good one in step 3, a check is carried out on the value of the 2 cost resulting from the 2 absolute differences computation, and if both of them are smaller than a certain threshold, then the retained cost, which is the sum of these 2, is reduced. Otherwise, if one of them or the 2 of them are bigger than this threshold, the final cost is increased.

[0148] This change improves the function of step 3 and improves the quality of the produced disparity map while keeping a low computational cost compared to the SAD algorithm.

[0149] Figures 19-20 relate to a de-noising algorithm for a CFA (color filter array) image, termed herein a “W-means” for “Weighted means”. Figure 19D shows a non-limiting example of such an algorithm. The algorithm groups the 4 CFA colors to make a so-called “4-color pixel”. Each one of these 4-color pixels in the input image is compared to its neighbors. A weight is attributed to each neighbor depending on its difference with the center pixel. Then, for each color separately, the weighted mean is computed to generate the output 4-color pixel.

[0150] First, consider the following CFA image X with size (w × h) and a (2 × 2) color pattern size (the colors shows an example for the Bayer pattern “Green1-Blue-Red-Green2 (GBRG)”):

$$X = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & \dots & x_{0,w-1} \\ x_{1,0} & x_{1,1} & x_{1,2} & \dots & x_{1,w-1} \\ x_{2,0} & x_{2,1} & x_{2,2} & \dots & x_{2,w-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{h-1,0} & x_{h-1,1} & x_{h-1,2} & \dots & x_{h-1,w-1} \end{bmatrix},$$

where $x_{i,j}$ are pixels intensity values.

$$\left(\frac{w}{2} \times \frac{h}{2} \right)$$

[0151] The same image can be represented as a four color image U with size (m × n)=

$$U = \begin{bmatrix} U_{0,0} & U_{0,1} & \dots & U_{0,m-1} \\ U_{1,0} & U_{1,1} & \dots & U_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ U_{n-1,0} & U_{n-1,1} & \dots & U_{n-1,m-1} \end{bmatrix}$$

[0152] Where $U_{i,j} = [x_{2i,2j}, x_{2i+1,2j}, x_{2i,2j+1}, x_{2i+1,2j+1}]$.

[0153] Figure 19A shows a simple example of this alternative representation.

[0154] The filtered image V with size m × n (same format as U), is given by the equations below.

$$V_{i,j} = \frac{1}{C_{i,j}} \sum_{U_{k,l} \in B(i,j,f)} U_{k,l} \cdot w(U_{k,l}, U_{i,j}),$$

$$C_{i,j} = \sum_{U_{k,l} \in B(i,j,f)} w(U_{k,l}, U_{i,j}),$$

$$w(U_{k,l}, U_{i,j}) = e^{-\frac{\max(d(U_{k,l}, U_{i,j}) - 2\sigma, 0)}{h}},$$

$$d(U_{k,l}, U_{i,j}) = \sqrt{\frac{1}{4} \sum_{q \in U_{k,l}, x \in U_{i,j}} (q - x)^2},$$

where $B(i, j, f)$ is the square neighborhood centered at $U_{i,j}$ with size $(2f + 1) \times (2f + 1)$ from U image, σ and h are constant parameters. The weight $w \in [0, 1]$ depends on the color distance d (there are 4 colors so this is a 4- dimensional distance). This allows application of a bigger weight on similar pixels.

[0155] The σ parameter can work as a threshold to ignore noise effect on distances, when its value equals the standard deviation of the noise. Distances smaller than 2σ have their weights set to 1, while larger distances decrease at an exponential rate. The h parameter controls the strength of this exponential function, thus the weights of non-similar pixels. The effect of parameters on the weights relative to the distance can be seen in Figure 19B.

[0156] The main difference with the NLM (Non-Local Means) algorithm (see Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. “Non-Local Means Denoising”. In: *Image Processing On Line* 1 (2011), pp. 208–212. DOI: 10.5201/ipol.2011.bcm_nlm), which makes “W-means” algorithm a lot less iterative, is the computing of the distance d (last equation above). Instead of computing the distance with all $U_{k,l}$ and $U_{i,j}$ neighbors, this algorithm only cares about $U_{k,l}$ and $U_{i,j}$ colors. The advantage of having 4 colors is to be more accurate than with only 3 colors.

[0157] Various adjustments can then be performed to decrease the computational resources necessary to perform the W-means algorithm for noise reduction. The Euclidean distance in the third of three equations above, where the square factor requires a multiplier for each recursive step

(for each color of each neighbor) and a square root for each neighbor, the following optimization was performed. The Euclidean distance can be replaced by the Manhattan (Taxicab) distance. Compared to the Euclidean distance, it is computed by removing the square root and computing an absolute value instead of the square, which improves the resource consumption a lot. A simple 2D visualization of these distances can be seen in Figure 19C.

[0158] The Euclidean distance gives the best estimation for the difference between 2 pixels. But, being compared to other differences, this algorithm only requires to have comparable difference values. The Manhattan distance also quantifies the difference between 2 pixels, thus it can also be used for this application.

[0159] With this optimization, the last of the above three equations becomes the below equation:

$$d(U_{k,l}, U_{i,j}) = \frac{1}{4} \sum_{q \in U_{k,l}, x \in U_{i,j}} |q - x|$$

[0160] The division by the parameter h in the second of the three above equations may optionally be handled by restricting h values to powers of 2. This way, only multiplexers and/or shifters are required. However, it is preferred to divide by a constant, from 1 to 8, even if that requires more logic elements. The exponential in the second of the three above equations may optionally be handled with threshold based binary weights. Binary weights may optionally be used generally to optimize the above equations.

[0161] Figure 20 demonstrates the effectiveness of the W-means method, which also consumes fewer resources than the art-known methods. For further optimization, optionally a parameter is set so that the denoising increases as the analog amplifier increases its activity. The analog amplifier increases its amplification as the amount of light decreases. In low light conditions, noise can increase significantly. Therefore, increasing denoising as amplification increases can offset this problem, without adding blur in the image.

[0162] Figures 21-31 relate to an exemplary, optional implementation system, and flow, according to some embodiments, that is interoperative with the previously described systems. This system and flow can allow correction according to the W-means method described above, as well as bad pixel correction, described below.

[0163] The corrections will be implemented on raw CFA images, just before the debayer process. The input pixel stream consists in the following standard signals:

- Pixel clock 1-bit : clock for following signals.
- Pixel Data 12-bit : pixel intensity value.
- Frame valid 1-bit : used to synchronize the start and the end of the frame.
- Line valid 1-bit : means that the pixel data is valid, otherwise it is blanking data. This signal takes the value '1' continuously for the entire row with.

[0164] The process units can have, at least, the interfaces shown in Figure 21A. The chronogram in Figure 21B shows an example of data transfer.

[0165] The method used for defective pixel detection and correction is an adaptation of the algorithm proposed by Bailey and Jimmy (Single shell version; D. Bailey and J. S. Jimmy. "FPGA based multi-shell filter for hot pixel removal within colour filter array demosaicing". In: 2016 International Conference on Image and Vision Computing New Zealand (IVCNZ). Nov. 2016, pp. 1–6. DOI: 10 . 1109 / IVCNZ . 2016 . 7804450) is low resource consuming and produced good results during the tests. It is a spatial filter especially made for CFA images. A schematic of the method is shown in Figure 22A.

[0166] Algorithms could be described by the below equation which is applied for all pixels in the image. The proposed implementation diagram is shown in Figure 22B. In the equation:

$$y_{i,j} = \text{med}\{\min(S_{CFA}), x_{i,j}, \max(S_{CFA})\},$$

[0167] where $y_{i,j}$ is the output pixel that depends on the input pixel $x_{i,j}$ and neighbors of same color S_{CFA} represented by black dots in Figure 22A.

[0168] The filter can remove defective pixels that do not belong in a defective pixel cluster (two or more defective neighbors). The sensor data sheet specifies that there are no clusters of defective pixels. Pixels in borders that cannot be processed (two rows on top and two on the bottom, and two columns on each side) are copied from the input to the output.

[0169] The diagram of the exemplary, illustrative FPGA implementation is shown in Figure 23,

features various adaptations of the published method. Compared to proposed implementation diagram by the authors, the second to last register was added to solve timings issues. A multiplexer has been added to copy the input pixel right to the output when the pixel belongs to the image border.

[0170] The Create rows stream component allows to turn the single row stream into a three color-neighbors rows streams called $rs1$, $rs3$, and $rs5$. Due to the CFA image, the filter must process one in two rows. To do this the “2x rows buffer” stores 2 lines instead of one. Then, the Quad-register component can be used to extract the kernel, as in Figure 22A. Other components include Sort min/max combinatorial units, to perform the process shown in above equation.

[0171] Control signals: the pixel data is delayed by approximately two rows, so control signals (frame valid and line valid) must also have this delay. To do that, two more components were created: frame valid delay, that simply runs a counter on each frame valid input transition (when the counter reaches the required delay value, the output is inverted), .and a line valid generator that is also based on a counter. When the counter starts, the valid signal is set. Then, when it reaches the image width, the valid signal is cleared.

[0172] Based on row and column counters, the line valid generator can be enabled on the second row of the input image and disabled two rows after the end. The copy signal is enabled when the output pixel corresponds to a border in the output image. Pixels residing in the image border are : 1st and 2nd row; 1st and 2nd column; 2nd last and last column; and 2nd last and last row.

[0173] The exemplary implementation of the bad pixel removal method in a camera system as described herein is shown in Figure 24. Because of the stereo camera pipeline, the method can be instantiated twice, one for each pixel stream. However, the memory is preferably allocated in such a way to avoid employing double the amount of memory.

[0174] Turning now to the architecture of the W-means method, shown in Figure 25, the design was made to be reusable. Indeed, if in the future the resource optimization level does not produce sufficient denoising accuracy, some parts of the algorithm can easily be changed to a more resource intensive version. The control signals are generated as the previous implementation. They are based on row and column counters.

[0175] The four components shown in Figure 25 include a generate kernel module, a compute

distances module, a filter core module and a format output module.

[0176] Generate kernel - this component permits to extract the image kernel to be processed. Figure 26 shows an exemplary diagram. The “Create row stream” component follows the same principle as the previous implementation of the bad pixel method. The kernel contains a 3×3 color group zone, which corresponds to a 6×6 pixel zone.

[0177] Distance computation - the distance is computed following the Manhattan distance described in the previous equation. Figure 27 shows an exemplary hardware implementation. The Manhattan distance is computed between each color group neighbor and the center color group.

[0178] Filter core “thr_optdiv” – a non-limiting, exemplary diagram of the main component of the filter is shown in Figure 28. The implementation features a number of components. Compute weights (threshold version), where the binary weights are computed. Bit addition and compare, which permits the addition of weights and prepares control signals for the division optimization. It sums bits in the weights vector and compares the sum with all possible power of 2 values (except 1).

[0179] Division optimization: This process applies a division optimization, if the sum of weights is equal to a power of 2, the weight does not change. Else, all weights that overflow after the power of 2 are forced to 0.

[0180] Apply weights: Applying weights is simply done by a multiplexer. If the weight equal to 1 the associated pixel value is outputted, else it is 0. Then all multiplexers outputs are summed. Division - here the power of 2 divisions are made, where each divisor unit is only wiring.

[0181] Format output - the denoised color group stream needs to be formatted to a pixel stream. This component permits the algorithm to choose the valid color group to be outputted as a pixel stream. Figure 29B shows an example of valid and not valid color groups. A color group is valid when its top left color is the same as the first color of the image (1st row, 1st column).

[0182] Figure 29A shows an exemplary, illustrative, non-limiting diagram for this process. Control signals are generated from column and rows counters. Row and column selection are simply least significant bits (LSBs) of these counters. As a color group belongs to two rows. it is required to use a row buffer to delay the second row of the color group. The copy_pix signal is the copied pixel value from the input image. It is used to copy image borders.

[0183] An exemplary implementation of the “W-means” algorithm in the stereoscopic pixel stream can be added while keeping the bad pixel removal algorithm in a camera system as described herein is shown in Figure 30. The visual result, similar to previous tests, is shown in Figures 31A (pre) and 31B (post).

[0184] As tested on a Cyclone V FPGA, the system consumes only 5% of combinatorial logic and 7% of the memory. The FPGA tested was the Altera Cyclone V SOC (5CSTFD6D5F31I7N) FPGA). Optionally, the debayer method and the “W-means” algorithm could be combined or interwoven, to decrease resource usage. For every 4 clock cycles the “W-means” implementation only needs 1 to output 4 denoised pixels (only when color groups are valid). This means that during 3 clock cycles the algorithm does not need to filter the image. To improve resource consumption, instead of using a separate unit per pixel stream, both streams can be used in the same computing pipeline.

[0185] Figure 32 shows a non-limiting exemplary method for color correction according to at least some embodiments. Selectivity of filters, by which CFA (color filter array) can be applied as described above, is not narrow; moreover tails of each R G B spectra response usually overlap each other. These problems can lead to the wrong colors in output frame. A matrix of coefficients multiplied by each color channel tends to mitigate such effect. Each coefficient is obtained by a calibration camera process, which can be performed once, and resulting matrix is called Color Correction Matrix (CCM).

[0186] CMOS image sensors are sometimes characterized by quantum efficiency response. Hence, such sensors are monochromic by nature. In order to obtain a color image, a CFA is applied to the sensor output. Depending on the quantum efficiency of the filter, each pixel stores a single color information point. The particular selection of materials, used to realize the CFA, are usually not faithful to natural colors. The problem is typically due to an imperfect frequency range selectivity as well as cross color effect. In particular, each curve does not have a tight Gaussian shape (low selectivity), moreover the tails of each curve overlap each other (cross color effect). In order to correct the color appearance, each channel of the de-mosaiced image has to be multiplied by certain coefficients:

$$\begin{bmatrix} R_{corr} \\ G_{corr} \\ B_{corr} \end{bmatrix} = \begin{bmatrix} r1 & g1 & b1 \\ r2 & g2 & b2 \\ r3 & g3 & b3 \end{bmatrix} \times \begin{bmatrix} R_{cam} \\ G_{cam} \\ B_{cam} \end{bmatrix}$$

Where X_{cam} are R, G, B data coming from camera and X_{corr} are R, G, B channel corrected values. The terms r_j, g_j, b_j (with j assuming values 1, 2, 3) compose the color correction matrix.

[0187] Turning now to Figure 32, a method 3200 is performed for color correction (according to some embodiments). In stage 3202, the camera calibration to retrieve color correction matrix coefficients is performed by processing a frame portraying a color checker board. The regions of the frame belonging to the color checker may be manually selected. For each region, the median is performed to evaluate the R, G, B camera channels' response. An example of the reference colors information characterizing color checker board may be found for example in ColorChecker classic for image reproduction | x-rite.

[0188] A first estimation of the coefficients is obtained in stage 3204, for example by computing the minimum norm least square solution satisfying system in Tsung-Huang Chen and Shao-Yi Chien (Cost effective color filter array de-mosaicking with chrominance variance weighted interpolation. In IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007, pages 1277–1280); where X_{ref} terms are R, G, and B reference colors values in checker board, while X_{cam} terms are R, G, and B camera colors values sent by camera. Applying these coefficients to the image causes the response of each channel to better adhere to the ideal characteristics of the image. Nevertheless test output images featured large saturated regions (data not shown).

$$\begin{bmatrix} R_{ref} \\ G_{ref} \\ B_{ref} \end{bmatrix} = \begin{bmatrix} r1 & g1 & b1 \\ r2 & g2 & b3 \\ r3 & g2 & b3 \end{bmatrix} \times \begin{bmatrix} R_{cam} \\ G_{cam} \\ B_{cam} \end{bmatrix}$$

[0189] This is due a lack of compensation of the luminance component, defined, according to ITU-R recommendation BT.709, as:

$$Y' = \begin{bmatrix} 0.2126 & 0.7152 & 0.0722 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

[0190] When a direct correction is performed, the resulting luminance is higher than in original frame. In order to maintain an unaltered luminance component the following calculation is performed in stage 3206:

[0191] Consider x as pixels from original frame, y as pixels from directly corrected frame and y^* as pixels from luminance corrected frame. These pixels are related one to the other by the two below equations, where A and C are 3×3 matrices.

$$y = Ax$$

$$y^* = Cx$$

These matrices are linked by the relation:

$$A = \alpha C$$

then

$$y = \alpha y^*$$

$$\alpha = \frac{lum(Ax)}{lum(y^*)}$$

Where $lum()$ is a function defined to calculate luminance component of input pixels. Because we are looking for α such that the luminance components of the original and final frames are equal, $lum(y^*) = lum(x)$, then α is:

$$\alpha = \frac{lum(Ax)}{lum(x)}$$

[0192] The color correction matrix is then established in stage 3208. Multiplying the frame by the obtained C color correction matrix, a natural color frame image is obtained in stage 3210. Moreover, the image sensor response is more similar to an ideal one adjusted with original luminance.

[0193] Figures 33A-33D show a non-limiting exemplary FPGA configuration according to at least some embodiments. Figure 33A shows an FPGA 3300 system, while Figure 33B shows the top of

FPGA system 3300 in more detail, and Figures 33C and 33D show the left and right sides, respectively, of the bottom of FPGA system 3300 in more detail. Reference is made to all of Figures 33A-33D in the below discussion.

[0194] FPGA system 3300 features an FPGA 3302, receiving input from a right sensor 3304 and a left sensor 3306. Data from each sensor 3304 and 3306 is fed to a preprocessing stage 3308, which runs preprocessing for data from each sensor separately as shown. For each sensor, preprocessing stage 3308 begins with denoising and bad pixel detection 3310, performed as previously described. Next the previously described debayer process 3312 is performed .

[0195] The results of the debayer process 3312 are then fed to the previously described color correction matrix (CCM) process 3314. The data from CCM process 3314 is used to determine the histogram 3318. The histogram then feeds to the previously described white balance correction process 3316. After that a rectify process 3320 is performed for stereo rectification as previously described.

[0196] FPGA system 3300 is shown with three branches, in Figures 33B-33D. There are two links shown between the top and bottom branches, labeled as “to A” and “to B”. There are two links shown between the left and right bottom branches, labeled as “to B” and “to C”.

[0197] Turning to the first branch, “to A” (in Figure 33B) and “A” (in Figure 33C), sensors 3304 and 3306 have a bi-directional flow with a trigger 3322 for controlling and syncing between the inputs from both sensors 3304 and 3306, so that timing is synchronized between the frames. In addition, sensors 3304 and 3306 have a bi-directional flow with an I2C (Inter Integrated Circuit) 3324. I2C 3324 includes an I2C controller 3326 and a memory map 3328. I2C controller 3326 in this example is a master microcontroller (slave microcontrollers and other slave components may also be featured; not shown). Memory map 3328 is a map of the memory registers in the various slave components which allow the one or more controllers to write to the registers of the slave devices. Memory map 3328 is a register for controlling the values for the variables for FPGA system 3300.

[0198] I2C controller 3326 is also in communication with a depth controller 3330 for synchronizing the timing of the depth sensor data. Optionally all sensor data passes through I2C controller 3326, including but not limited to sensors 3304 and 3306, and sensors 3346.

[0199] In the second branch, “to B” (in Figures 33B or 33C) or “B” (in Figure 33D), preprocessing stage 3308 transmits preprocessed RGB sensor data to two FIFO buffers 3334A and 3334B on a GPIF (General Programmable Interface) IF (interface) module 3336. GPIF IF module 3336 implements a 32 bit bus interface, which is used to communicate with the USB3 chip 3350. FIFO buffers 3334A and 3334B operate as previously described. Depth data from depth controller 3330 is fed to a depth FIFO buffer 3338. GPIF IF module 3336 also has a controller 3340 and a GPIF IF 3342. GPIF IF 3342 is the interface for the bus.

[0200] GPIF IF 3342 also receives additional sensor data from an additional sensors FIFO buffer 3344, which in turn optionally receives sensor data from multiple sensors 3346, of which two examples are shown for the purpose of illustration and without any intention of being limiting. Non-limiting examples that are shown include a MCU inertial sensor 3346A and a MCU coordinator 3346B. This data is optionally fed through a controller 3348, which may be an SPI (serial peripheral bus) controller for example.

[0201] Processed information is then output from GPIF IF 3342 to the USB chip 3350 for example.

[0202] The actions of GPIF IF 3342 may be assisted by computations performed by SOC (system on chip) 3360, optionally with an external memory 3362. SOC 3360, using external memory 3362, is able to increase the speed of performance of GPIF IF 3342 by performing computations more quickly. SOC 3360 acts as embedded processor with a DMA (direct memory access) module 3361. For example, SOC 3360 can perform calculations related to stereo data (including depth and RGB data) through sensor FIFOs 3334A, 3334B and 3338.

[0203] Turning now to the third branch, labeled “to C” in Figure 33C and “C” in Figure 33D, trigger 3322 may control the action(s) of sensors 3346 as shown, to trigger their activation for data collection for example. Trigger 3322 may, alternatively or additionally, synchronize the various sensors 3346 with a timestamp. I2C 3324 receives data from the various sensors, including sensors 3346, and sensors 3304 and 3306, as previously described.

[0204] Figures 34-46 describes various non-limiting examples of methods for tracking a user. These methods may, for example and without limitation, be performed by the FPGA as shown in Figures 5 or 6, and/or the CPU of Figure 6. As a non-limiting example, these methods may be an example of a feature that may be provided to the user application of Figure 5 and that may be calculated by the fusion kernels of Figure 5.

[0205] These methods may be used for tracking at least a portion of a body by fitting data points received from a depth sensor and/or other sensors and/or “markers” as described herein to a body model. For example, in some embodiments, certain of such data points are identified as “super points,” and apportioned greater weight as compared to other points. Such super points can be obtained from objects attached to the body, including, but not limited to, active markers that provide a detectable signal, or a passive object, including, without limitation, headgear or a mask (for example for VR (virtual reality)), or a smart watch. Such objects are described for example with regard to Figure 2B. Such super points may also be obtained from specific data points that are matched to the model, such as data points that are matched to vertices that correspond to joints in the model.

[0206] Figure 34 shows an exemplary, illustrative non-limiting method for tracking the user, optionally performed with the system of Figures 1 or 2, according to at least some embodiments of the present disclosure.

[0207] As shown, at 3402, the system initiates activity, for example, by being powered up (i.e., turned on). The system can be implemented as described in Figure 1 but may also optionally be implemented in other ways. At 3404, the system performs system calibration, which can include determining license and/or privacy features. System calibration may also optionally include calibration of one or more functions of a sensor.

[0208] At 3406, an initial user position is determined, which (in some embodiments), is the location and orientation of the user relative to the sensors (optionally at least with respect to the camera and depth sensors). For example, the user may be asked to or be placed such that the user is in front of the camera and depth sensors. Optionally, the user may be asked to perform a specific pose, such as the “T” pose for example, in which the user stands straight with arms outstretched, facing the camera. The term “pose” relates to position and orientation of the body of the user.

[0209] At 3408 the template is initialized. As described in greater detail below, the template features a model of a human body, configured as only a plurality of parameters and features, such as a skeleton, joints and so forth, which are used to assist in tracking of the user’s movements. At 3410, sensor data is received, such as for example, one or more of depth sensor data and/or camera data. At 3412 and 3414, the sensor data is analyzed to track the user, for example, with regard to the user’s movements. Optionally, the sensor data can be mapped onto a body model, e.g., the

body model features an articulated structure of joints and a skin defined by a mesh of vertices that are soft-assigned to the joints of the model with blending weights. In this way, the skin can deform accordingly with the body pose to simulate a realistic human shape.

[0210] Optionally, the sensor data is analyzed by mapping onto a GMM (Gaussian mixture model) as described herein. As described in greater detail below, optionally, a classifier can be used. Because the user's pose is not likely to change significantly between frames, optionally, the process at 3412, 3414, while performed iteratively, can only be performed with regard to a limited number of iterations. For example, the present inventors have found that, surprisingly, as few as 3-10 iterations may be used to map the data. If a GMM is used, each vertex of the skin defines an isotropic Gaussian, whose mean location in the 3D space is a function of the rotation parameters of the joints to which the vertex is attached (rotating the left wrist won't affect the position of the vertices on the right hand skin).

[0211] The body model preferably features a sparse-skin representation. Having a sparse-skin representation is convenient to handle occlusions. Both self-occlusions or occlusions of body parts due to clutter or because the user exits the camera frame. One dynamically enables or disables the Gaussians that are considered to be occluded at a given frame, so that those disabled won't influence the optimization.

[0212] In a different direction, it is also straightforward to model amputee users by suppressing the corresponding Gaussians. This can be done online during a calibration process or having a therapist manually configuring the body model. In 3412, if a GMM is used, the sensor data is mapped as a point cloud to the GMM. The GMM and mapping are optionally implemented as described with regard to "Real-time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera" by Mao Ye and Ruigang Yang, IEEE Transactions on Pattern Analysis & Machine Intelligence 2016 vol. 38 Issue No. 08. In this paper, an energy function is described, which is minimized according to the mapping process.

[0213] Optionally, only the depth sensor data is used, but alternatively, both the depth sensor and the camera data are used. For example, the calculations may be performed as follows. Given a set of N points $x \in X$ it is desired to fit a GMM with M components (v_m).

$$p(x_n) = (1 - u) \sum_{m=1}^M p(v_m) p(x_n | v_m) + u \frac{1}{N} \quad (1)$$

[0214]

[0215] At 3414, one or more constraints are imposed on the GMM as described in greater detail below. For example, optionally the model is constrained so that the body parts of the user are constrained in terms of the possible angles that they may assume. At 3416, the mapped data is optionally integrated with video data.

[0216] Figure 35 shows an exemplary, illustrative non-limiting tracking engine, optionally for use with the system of Figures 1 or 2, or the method of Figure 34, according to at least some embodiments of the present invention. For this embodiment of the tracking engine, the data is assumed to be mapped to a GMM, but as described herein, optionally a classifier is used instead. As shown, the tracking engine features a template engine 3500, which reads a template from a template database 3502, and then feeds the template to a GMM mapper 3508. GMM mapper 3508 also receives point cloud information from a point cloud decomposer 3504, which receives the depth sensor data as an input in 3506. Optionally color camera data could also be provided to point cloud decomposer 3504. For example, stereo RGB could be used to assist with the assignment of points to body parts and/or to improve the depth sensor data. Solutions to the problem of configuring depth sensor data to a point cloud is well known in the art and could optionally be performed according to any suitable method. One non-limiting example of a suitable method is provided in “Alignment of Continuous Video onto 3D Point Clouds” by Zhao et al, available at <https://pdfs.semanticscholar.org/124c/0ee6a3730a9266dae59d94a90124760f1a5c.pdf>.

[0217] To increase the speed of processing, the depth sensor data may be configured as follows. To do so a KD-tree of the scene is built for each frame, so that when computing correspondences from vertices to cloud one only uses the K nearest neighbors and assume a zero-posterior for the rest. As a consequence, the algorithm runs several orders of magnitude faster. The gating of correspondences allows sparsification of both the distance and the posterior matrix with huge gains on computation speed.

[0218] As compared to “Real-time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera” by Mao Ye and Ruigang Yang, IEEE Transactions on

Pattern Analysis & Machine Intelligence 2016 vol. 38 Issue No. 08., which reached real time performance with a GPU (graphics processing unit), the presently described algorithm, according to some embodiments, can reach real-time performance (100+ fps in a i7 processor) with CPU (central processing unit) only, which is a significant computational advantage.

[0219] GMM mapper 3508 features a GMM data mapping module 3510, a mapping constraint module 3512 and a template deformation module 3514. GMM data mapping module 3510 receives the point cloud data from point cloud decomposer 3504 and maps this data onto the GMM, as adjusted by the input template from template engine 3500. Next one or more constraints from mapping constraint module 3512, for example in regard to the angle range that body parts of the user can assume, are applied to the mapped data on the GMM by mapping constraint module 3512. Optionally, such information is augmented by deforming the template according to information from template deformation module 3514; alternatively, such deformations are applied on the fly by GMM data mapping module 3510 and mapping constraint module 3512. In this case, template deformation module 3514 is either absent or alternatively may be used to apply one or more heuristics, for example according to pose recovery as described in greater detail below.

[0220] Figure 36 shows templates and the template engine in more detail, according to at least some embodiments of the present disclosure. Template engine 3500, according to some embodiments, features an adjust template for initial user position module 3614, which receives information regarding the initial position of the user from an initial user position input 3612. For example, if the template is constructed with a standing user, but the user is lying down or sitting, then the template is preferably adjusted according to the actual position of the user. Even if the user's position is the same as that intended by the template, such as standing in a T pose, the template can be adjusted according to the actual user, such as the actual dimensions of the user.

[0221] Template database 3502 can contain a variety of different types of information in each template. For example, each template preferably includes a skeleton 3600, a hierarchy of joints 3602 and a joint detection 3604. Skeleton 3600 defines the virtual skeleton of the user, as determinable from the point cloud data. Each skeleton 3600 includes a hierarchy of joints 3602, which is (as known) representation of a virtual skeleton with its virtual parts. A hierarchy of joints 3602 enables each joint of the user's body to be located and for the location to be determined in context to the location of other joints. Joint detection 3604 can relate to information for detecting

the joints of the user as specific points, or groups of points, from the point cloud as described in greater detail below.

[0222] Each template can also feature surface vertices 3606, surface mesh connectivity 3608, skinning weights 3610 and constraints 3616. Surface vertices 3606 relate to the vertices on the external body of the user (as opposed to the internal representation of the user's body structure by the virtual skeleton), while surface mesh connectivity 3608 defines the external body surface of the user according to a mesh. Skinning weights 3610 determine the weights for how the skinning is performed, for example in terms of determining the correct external surface of the user.

[0223] Constraints 3616 can be used to determine which body positions are not allowed, for example, according to the possible angle ranges of different body part positions. For example, and without limitation, constraints may include one or more of self-intersection, angle and pose prior. These constraints can be weighted differently. Weighting is preferred, rather than setting the constraints as an absolute bar, as the inventors have determined that surprisingly, the GMM model does not operate efficiently if the search space is too restricted. Therefore, weighting allows the importance of the constraints to be captured without excessively restricting the search space. The constraints can be applied to the previously described energy function.

[0224] The self-intersection constraint is optional and may not be applied, as the angle constraints may effectively account for self-intersection according to how they are applied. If applied, the self-intersection constraint may be calculated as follows:

$$E_{self} = \frac{1}{|P|} \sum_{(s,t) \in P} \max(0, h_{st}(\Theta^{t-1} + \Delta\Theta)) \quad (2)$$

$$h_{st}(\theta) = (r_s + r_t)^2 - \|v_s(\theta) - v_t(\theta)\|^2 \quad (3)$$

$$v_m(\Theta^{t-1} + \Delta\Theta) = v_m + I_m \Delta\xi_g + \sum_{k=1}^K \beta_{mk} \xi_k^t v_m \Delta\theta_k \quad (4)$$

$$h_{st}(\theta) = (r_s + r_t)^2 - \|v_s + \sum_{k=1}^K \beta_{sk} \xi_k^t v_s \Delta\theta_k - v_t - \sum_{k=1}^K \beta_{tk} \xi_k^t v_t \Delta\theta_k\|^2 \quad (5)$$

[0225]

[0226] where r_s , r_t represent the radius of the intersecting Gaussians. The set P defines the set of

pairs of Gaussians that should be forbidden to intersect.

[0227] In some embodiments, the angle constraints are important for correct operation of the model and are heavily weighted. Constraints are currently applied per angle, but could optionally be applied for all angles jointly. The angle constraints relate to the ranges of angles that various body parts are allowed to achieve.

[0228] For a given joint, the angle constraints are preferably determined according to a rotational model, such that for example optionally 1, 2 or 3 degrees of freedom are determined. For example the elbow may be determined to have 1 degree of freedom and the shoulder may be determined to have 3 degrees of freedom. For each degree of freedom, a minimum and maximum angle is determined. As another example of the ranges, the Appendix provides notation in regard to the angles, such that `limit_rotation_x` relates to the minimum and maximum angle for a first degree of freedom, `limit_rotation_y` relates to the minimum and maximum angle for a second degree of freedom, and `limit_rotation_z` relates to the minimum and maximum angle for a third degree of freedom. If the values are set to zero for a limit rotation, then that degree of freedom is not available to the joint and those angles would not change at all. If no specific limit rotations are set, then the joint or bone achieves the rotation of the entire body.

[0229] The angle constraints are optionally imposed with a max/mean operator. This operator is non-linear and involves gradient based optimizing. Such angle constraints can be imposed as rotation limits, that are applied using a non-linear term. Non-limiting examples of such terms are described in equations 11, 12 of this paper: “Efficient and Precise Interactive Hand Tracking Through Joint, Continuous Optimization of Pose and Correspondences” by Taylor et al, 2016, available from <http://www.samehkhams.com/taylor-siggraph2016.pdf>. One example of a non-limiting method to calculate angle constraints is as follows:

$$E_{limit} = \frac{1}{K} \sum_{i=1}^K v(\theta_i^{min}, \theta, \theta_i^{max})^2 \quad (6)$$

[0230]

[0231] where $v(a, x, b) = \max(0, a-x) + \max(x-b, 0)$. Since it is desired to optimize with respect to $\Delta\theta$, the following equations should be used:

$$E_{\text{limit}} = \frac{1}{K} \sum_{i=1}^K \max(0, \theta_i^{\text{min}} - (\theta_i^{t-1} + \Delta\theta_i)) + \max((\theta_i^{t-1} + \Delta\theta_i) - \theta_i^{\text{max}}, 0) \quad (7)$$

Applying subgradient, and assuming $\Delta\theta$ is very small:

$$\frac{\partial E}{\partial \Delta\theta} = \begin{cases} 0 & \text{if } \theta^{t-1} \geq \theta^{\text{min}} \wedge \theta^{t-1} \leq \theta^{\text{max}} \\ 2(\theta - \theta^{\text{max}}) & \text{if } \theta^{t-1} \geq \theta^{\text{min}} \wedge \theta^{t-1} > \theta^{\text{max}} \\ -2(\theta^{\text{min}} - \theta) & \text{if } \theta^{t-1} < \theta^{\text{min}} \wedge \theta^{t-1} \leq \theta^{\text{max}} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

[0232]

[0233] The pose prior constraints are optionally applied to avoid problems caused by depth sensor noise, like not correctly detecting dark hair or dark clothing. One example of a pose prior constraint is keeping the pelvic bone more or less straight. If the user is sitting at a table or lying down, the lower body can be more or less hidden by the furniture and, as a consequence, the pelvic bone may try to fit outlier points, usually located in the region of the furniture, and the pelvic bone rotates towards the furniture, which is not correct. Another pose prior constraint is to keep the head more or less straight. This is applied to solve noisy rotations of the head that happen if the hair points are not showing due to depth-sensor issues. These pose prior constraints may be applied a priori and/or added to the energy function on the fly.

[0234] Both pose prior and angle limit constraints, in some embodiments, are implemented with a weighted sum of components in the energy function. The angle limit energy component preferably has a larger weight relative to the pose prior. Data from inertial sensors can also be used for one or more constraints. For example, and without limitation, the orientation of the inertial sensor may be applied as a prior on the orientation of the bones (preferably the hands). Alternatively, the orientation data could be used in the optimization, not as a prior but as an additional specific term.

[0235] The inertial data can be introduced as a "super point" as described herein, additionally or alternatively to the above uses. The use as a super point increases the accuracy of the absolute position data, even without the orientation data. However the orientation data is helpful for determining the position orientation of the hands.

[0236] Different templates can be created for different user positions, such as sitting, standing, lying down and so forth. Different templates may also optionally be created according to gender or other demographic information, and/or according to body dimensions (such as for various ranges of height, girth and so forth).

[0237] Template engine 3500 can be updated as to an initial position of the user, such as sitting, standing, lying down or so forth, as well as about other user characteristics which would enable a particular template to be selected and/or adjusted, such as demographic information for example. Template engine 3500 can then select a particular template from template database 3502 according to this information. The Adjust template for initial user position module 3614 then adjusts the template according to the initial detected user position, which can also relate to initialization of the template for that particular user from the first set of input data.

[0238] Figure 37 shows a non-limiting example of a method for creating and using templates, according to at least some embodiments of the present disclosure. As shown, an example user's entire body (or optionally a portion thereof) is scanned in a fixed position at 3702, to form a standard body. At 3704, the body of the user is modeled in 3D modeling software, for example, according to the Blender software as a non-limiting example (created by the Blender Foundation). A mesh is created which represents the human body. Each vertex of the mesh, which is each vertex of the skin, is incorporated. Certain vertices can be assigned to "joints" and/or "bones". For example, the vertices of the left forearm would be associated with the left elbow, so that when the left elbow rotates, the vertices of the left forearm also rotate. The model, according to some embodiments, imposes a type of constraint on the possible positions of the vertices, as well as repositioning the skin vertices in terms of joint positions. In 3706, the modeled body is corresponded to the template parameters, according to the operation of the specific 3D modeling software. The template parameters in terms of what is included (according to some embodiments) are described with regard to Figure 36.

[29] At 3708, the template parameters can be exported to a file, a portion of a non-limiting example of which is given in the Appendix. The example in the Appendix shows a portion of a standard male body in a T pose. The features are generally self-explanatory, but a few examples are described here for completeness. The global scale parameter indicates the global scaling in terms of the "overall size" of the body. Next a list of bones is provided, with their respective names, their initial locations (provided as "head" and "tail" parameters). The rotations can be as previously described. Each bone can be a parent to another bone, such that "sacrum" is the parent to bones such as "l. thigh" or left thigh for example. The skinning_weights provide the blending weights, or weights to be assigned for skinning; each data point represents a vertex in the model, with the one or more bone(s) that are influencing the vertex plus the weight(s) for that influence.

[30] At 3710, sensor data is obtained for the subject in a given position, which is preferably manually communicated to the template engine as described herein. At 3712, at least one measurement of the subject is determined as described herein, for example, with regard to bone length. Such information can be determined from the point cloud. Optionally, as described with regard to Figure 38 below, different types of point cloud data are used. For example, and without limitation, optical markers can provide “super point” data, as their locations are known according to the different colors of the LED lights featured in the markers. Other non-limiting examples of super point data include points associated with a VR headset, a smart-watch or some other appliance.

[31] Such “super point” data may also optionally be performed with joint detection as described in greater detail below, such as for example with regard to Figure 42, or alternatively as described in “Random Tree Walk toward Instantaneous 3D Human Pose Estimation” by Jung et al, June 2015 CVPR conference (available from https://www.researchgate.net/publication/275771971_Random_Tree_Walk_toward_Instantaneous_3D_Human_Pose_Estimation) and/or according to the description provided in “Efficient Human Pose Estimation from Single Depth Images” by Shotton et al, 2011 (available from <https://people.eecs.berkeley.edu/~rbg/papers/pose-from-depth-pami.pdf>).

[32] Alternatively the point cloud data can be used, alone or in combination with one or more types of “super point” data. The measurement that is obtained from the subject is used to initialize the tracking calculations, to adjust the template according to the specific parameters of the user.

[33] At 3714, the template is scaled or otherwise adjusted according to the specific parameters of the user, for example, and without limitation, according to one or more of user position, demographic data, size data, particular measurements within the body of the user such as bone length, and the like. Preferably, the dimensions of the user are scaled, not the constraint angles. To adjust the template for scaling, preferably the cost function can be adjusted for the parameter space of bone scales, rather than joint constraints.

[34] Figures 38A to 38E relate to different exemplary, illustrative non-limiting methods for mapping data to track the user, according to at least some embodiments of the present invention. Figure 38A shows such an exemplary method which uses the point cloud data without weighting any particular points and/or without a priori knowledge regarding these points, to map to a GMM

model. At 3802A, the sensor data from the depth sensor is decomposed to form point cloud data. At 3804A, the template vertices are adjusted, which also includes estimating an initial pose of the user. The adjustment can include adjusting the template according to the user position and scale, and optionally according to other parameters. If the process is being performed on a series of video frames and has already been performed on at least one frame, then optionally such an estimate may take into account a previous pose of the user, as the pose of the user is unlikely to change dramatically between frames. The adjustment of the template vertices preferably also includes the inclusion of the previously described constraints, which can then be applied later on to increase the probability of correct minima being obtained.

[35] At 3806A, the cost function gradient can be determined. The cost function gradient can be calculated as described with regard to “Real-time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera” by Mao Ye and Ruigang Yang, IEEE Transactions on Pattern Analysis & Machine Intelligence 2016 vol. 38 Issue No. 08. In this paper, an energy function is described, which is minimized according to the mapping process. The distance needs to be calculated for each point in the point cloud to each vertex for the cost function gradient.

[36] At 3808A, one or more constraints can be applied to the energy equation, as previously described, although this process can be performed concurrently at 3806A. At 3810A, a new location in the parameter space is solved for with regard to probability density, to determine a most likely location, according to the EM algorithm, which provides an iterative solution. At 3812A, the deformation model can be applied, for example, as described with regard to Figure 39, according to the iterative solution from the EM algorithm. At 3814, steps 3806A, 3808A, 3810A and 3812A can be repeated until the problem is solved.

[37] Figure 38B shows such an exemplary method which uses only optical marker point data, to map to a GMM model. In 3802B, only the optical marker data is detected from the point cloud data. The optical marker point data are an example of super points, in that their location on the body can be detected and tracked accurately, because specific colors of LED lights are associated with specific locations. Apart from only using the optical marker point data, the method proceeds as described with regard to Figure 38A, except that, optionally, the distance from each point to each vertex does not need to be calculated, which in some embodiments, is due to the possibility

to identify each marker with a specific location on the body with a very high degree of likelihood, so that it is known which marker is related to which location on the body.

[38] Figure 38C shows such an exemplary method which uses only joint detection point data, to map to a GMM model. For 3802C, only the joint data is detected from the point cloud data. The joint point data are an example of super points, in that their location on the body can be detected and tracked accurately, because of the special joint detection method that can be used as described herein. For example, the joint detection method can be performed according to a classifier as described herein. The joint detection algorithm can permit joints to be detected which can be difficult to label with optical markers (e.g., the jaw or neck). Apart from only using the optical marker point data, the method proceeds as described with regard to Figure 38B.

[39] Figure 38D shows such an exemplary method which uses a combination of super points, with higher weighting, in combination with the remainder of the point cloud data, to map to a GMM model. For 3802D, the sensor data is decomposed to a point cloud as described with regard to 3802A of Figure 38A. At 3804D, one or more super points are detected and are given higher weighting. The super points may be detected according to the previously described optical markers and/or according to joint detection. The position of the super points may not be reliable but the correspondences to various vertices are known with a high degree of probability, even if the data relating to the position of the super points is noisy. Steps 3806D, 3808D, 3810D, 3812D, 3814D and 3816D can be performed as described with regard to steps 3804A, 3806A, 3808A, 3810A, 3812A and 3814A of Figure 38A, except that the super points are given a higher weighting for the various calculations performed, over that of regular point cloud data.

[40] Figure 38E shows an exemplary method which uses a classifier. At 3802E is optionally performed as described with regard to Figure 38A. At 3804E, a trained classifier is applied to the point cloud. The trained classifier preferably features some type of body part classification model, such as described for example in “Efficient Human Pose Estimation from Single Depth Images” by Shotton et al, 2011 (available from <https://people.eecs.berkeley.edu/~rbg/papers/pose-from-depth-pami.pdf>). The described method operates as follows. Body part classification (BPC) first predicts a body part label at each pixel, and then uses these inferred labels to localize the body joints. The number and location of body parts are selected to provide the most information for determining the joints while at the same time, not overwhelming the trained classifier with data

requirements. The trained classifier may be constructed and trained according to any known method, such as for example the random trees method of classification. The location of the joints may then be fed into the previously described generative model of “Real-time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera” by Mao Ye and Ruigang Yang, IEEE Transactions on Pattern Analysis & Machine Intelligence 2016 vol. 38 Issue No. 08. Such a classification would create a plurality of super points from regular point cloud data, by significantly increasing the probability of a correct assignment to a particular vertex. Therefore, optionally, the remainder of the process would continue only with the super points or alternatively, the super points are implemented with higher weights than other points. The process then continues as for Figure 38B with the classified points acting as super points, such that steps 3806E, 3808E, 3810E, 3812E, 3814E and 3816E of Figure 38E correspond to steps 3804B-3814B of Figure 38B.

[41] Figure 39 shows a non-limiting example of a method for applying the deformation model, according to at least some embodiments of the present disclosure. At 3902, the position of the joints is initialized according to the template. At 3904, the position of each joint is detected separately. At 3906, the surface vertices are deformed according to the joint position. At 3908, the surface vertices are adjusted by skinning weights, to smooth the skin over a plurality of bones and joints. The skinning weights are blending weights as previously described, to determine how much each bone or joint determines an input to the skin surface vectors. An exemplary implementation is described in Real-time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera” by Mao Ye and Ruigang Yang, IEEE Transactions on Pattern Analysis & Machine Intelligence 2016 vol. 38 Issue No. 08. In 3910, the deformation model is updated from the last known position.

[42] With regard to the modeling of the data as described herein, a specific case relates to modeling of the shoulder joint. Ball-joints are a way of improving the present modeling of 3-DOF (3 degrees of freedom) joints (shoulders) that suffer from several issues due to the representation of rotations. The shoulder joint may be modeled by applying three rotations sequentially, for each DOF. This presents problems mainly related to gimbal lock and that such parameter space is not very well suited for optimization.

[43] An optional implementation uses twists that jointly encode angle, axis, and axis location. It is necessary to decouple the axis location in order to leave the axis vector free, as described for

example in Pons-Moll and Rosenhahn, “Ball Joints for Marker-less Human Motion Capture”, WACV 2009 (available from https://ps.is.tuebingen.mpg.de/uploads_file/attachment/attachment/140/PonsWACV2010.pdf).

One exemplary way to do so is to subtract the axis location from the points rotating wrt to the ball joint, and assume in the twist that the axis is in (0, 0, 0).

[44] This adjustment may be applied to any suitable method to model the shoulder joint. For example it can be used to adjust the deformation model of “Real-time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera” by Mao Ye and Ruigang Yang, IEEE Transactions on Pattern Analysis & Machine Intelligence 2016 vol. 38 Issue No. 08 as follows. In this paper, the basic deformation model is given by:

[45]
$$v_m^{t+1} \approx v_m^t + I_m^t \Delta \xi_g^t + \sum_{k=1}^K \beta_{mk} \xi_k^t v_m^t \Delta \theta_k^t$$

[46] In the above equation, v_m is a vertex of the previously described model.

[47] $\{\Delta \theta_k^t\}$ relates to the joint angles.

[48] $\Delta \xi_g^t$ relates to the global transformation.

[49] To accommodate the improved modeling of the shoulder, the previously described ball-joint information is plugged in and the constraint of ω having unit length is dropped, such that it is possible to represent jointly the rotation axis and the angle as the norm of ω :

[50]
$$v_m^{t+1} = v_m^t + I_m^t \Delta \xi_g^t + \sum_{k=1}^K \beta_{mk} \xi_k^t v_m^t \Delta \theta_k^t + \sum_{l=1}^L (\beta_{ml} (v_m^t - q_l) \Delta \omega^l + q_l)$$

[51] It is therefore possible to make a distinction between the indices k which represent degrees of freedom of 1D and 2D joints, and the indices l which represent whole ball-joints. The optimization parameters are transformed from three scalar angles $\Delta \theta$ to a 3-dim vector $\Delta \omega$ that represents both angle and axis orientation. The vector q_l refers to the location of the axis of joint

1. Note that the twist is primed, so to recover the original coordinate transform needs to be undone:

$$\Delta\omega_i\theta_i = ((T_i)^{-1}\Delta\omega'_i\theta_i)^\top T_i$$

[52]

[53] Figure 40 shows a non-limiting example of a method for pose recovery, according to at least some embodiments of the present disclosure. At 4002, a separate pose recovery thread is operated, which is separate from the main thread which is tracking the user and performing the above described pose calculations. To avoid overwhelming available computational resources, the pose recovery process is preferably operated intermittently rather than continuously. At 4004, for every n frames, the recovery pose is calculated from the point cloud data to determine the energy level of that pose. The number n can be any plurality of frames, such as from 2 to 10 frames, but is preferably from 3 to 5 frames. The initialization is applied with the previous frame's pose as the seed point but rerun to fit the data according to the GMM. The algorithm is then rerun with the recovery pose in 4006. At 4008, the current calculated pose of the user is compared to the recovery pose; if the recovery pose has lower energy in terms of the calculation of the point cloud, then the current calculated pose is replaced with the recovery pose. The recovery pose can be a resting pose of the user, for example.

[54] Figure 41 shows a non-limiting example of a method for segmentation of a background object, according to at least some embodiments of the present disclosure. Here, the background object can be an inanimate object (which should not be confused with the user), such as a table, chair, bed or other piece of furniture. At 4102, the point cloud is generated from the sensor data as previously described, followed by 4104, where a segmentation model is selected. For example, the segmentation model can be a simple model, such as a planar model, for certain types of background objects such as a table for example.

[55] At 4106, the RANSAC (random sample consensus) algorithm is applied to the point cloud data according to the selected segmentation model. This algorithm is probabilistic and iterative, and is used to detect (and remove) outliers from a point cloud data, in order to fit this data to the segmentation model. A non-limiting example of the application of the RANSAC model to segmentation according to a planar model is described in "RANSAC algorithm and elements of graph theory for automatic plane detection in 3D point clouds", by Martyna Poreba and François

Goulette, Symposium de PTFiT (Polish Society for Photogrammetry and Remote Sensing), Sep 2012, Poland. 24, pp.301-310, 2012.

[56] At 4108, the group of points that are most likely to belong to the selected segmentation model are located, and optionally, at 4110, if the method described herein has been performed more than once, then points which were previously detected as belonging to the model are up-voted. Because the object being segmented is a background, inanimate object, its position is not likely to change during the tracking process. Therefore, once a point has been determined to belong to the background object, it is likely to continue to be correctly associated with that object. At 4112, points that have been determined to belong to the model are removed, as such points could interfere with the correct tracking of the user.

[57] Figure 42 describes a non-limiting example of a method for joint detection, according to at least some embodiments of the present disclosure. As shown, the point cloud is received as input at 4202; the point cloud could optionally be generated as previously described. At 4204, an estimated location is determined for each joint. Preferably each joint is determined as a point in the point cloud, such that the estimated location relates to the location of the point that would correspond to that joint. For clarity, the joint locations are determined as vertices in the skeletal model, such that each joint is mapped as a discrete point (vertex) in the skeletal model. For the first frame, a mean position may be used from the training data determined from training the trees (as described in greater detail below), or from the template data. For subsequent frames, it is possible to use the position from the previous frame as an estimate. At 4206, a plurality of points close to the estimated location of the point corresponding to the joint is sampled. For example, the points can be sampled within a sphere having a particular radius from that point, such as 0.5 meter for example. Other distances are also possible, including any distance within a range from 0.005 meter to 2 meters for example. The radius could optionally be different for each joint.

[58] In terms of the mathematical representation of the sampled points, every candidate that is considered $c_{i,j}$ has a position $p_{i,j} \in \mathbb{R}^3$ that can be generated in a number of different ways. For a given joint j , candidate 3D positions can be generated for example as follows. Given training data of likely 3D poses of the skeleton, a multivariate 3D normal distribution of positions for every joint is estimated. A position can be generated by simply sampling a point from that distribution. This is optionally and preferably refined according to data from previous frames.

[59] With regard to subsequent processes, assume the joints are indexed by integers $j \in \mathbf{J}$, with \mathbf{J} representing the set of all joints. It is also assumed that two trained trees are trained as classifiers or regressors in advance, described in greater detail below with regard to 2412. These two trees include a tree $T_{\text{dist},j}$ that can estimate the distance $|x - y|$, as well as a tree $T_{\text{offset},j}$ that can directly estimate the offset $y - x$ to the joint. The training of these trees seeks to minimize a least-squares objective function. The trees can be trained as described with regard to “Random Tree Walk toward Instantaneous 3D Human Pose Estimation” by Jung et al, June 2015 CVPR conference (available from https://www.researchgate.net/publication/275771971_Random_Tree_Walk_toward_Instantaneous_3D_Human_Pose_Estimation).

[60] At every leaf node of the tree $T_{\text{dist},j}$, there is a set of distances. Once the distance tree has been traversed, a leaf node is reached where one takes the median distance at the leaf node to be the estimated distance. For the tree $T_{\text{offset},j}$, clusters are computed at every leaf node. Once at a leaf node of that tree, one takes the mean position of a randomly selected cluster, with the probability of choosing a cluster being proportional to the number of samples in the cluster. At 4208, the distance from each of the sampled points to the estimated location is determined. Preferably, the distance is calculated by using a tree as follows. There are several candidate positions for each joint whose position is to be estimated.

[61] One assumes there are N_j joints $j \in \mathbf{J}$ with $\mathbf{J} = \{1, \dots, N_j\}$. For every joint, one generates N_C candidates indexed by $\mathbf{C} = \{1, \dots, N_C\}$. Every candidate $c_{i,j} = (p_{i,j}, d_{i,j})$, with $i \in \mathbf{C}$ consists of a position $p_{i,j} \in \mathbb{R}^3$ of the proposed position for a joint in the camera coordinate system, and a distance $d_{i,j} \in \mathbb{R}$ in meters which is an estimate of how far away the position $p_{i,j}$ is from the actual, unknown joint position. A distance of $d_{i,j} = 0$ would mean that the candidate is a very good estimate of the true unknown 3d position. In other words, it is an estimated measure of how faithfully $p_{i,j}$ approximates the true unknown location.

[62] Let $r \in \mathbb{R}$ be a randomly generated number from a set of random numbers \mathbb{R} . Let $f \in \mathbf{F}$ represent an input depth frame from a set of frames \mathbf{F} , consisting of pixels with a depth

value associated with every pixel. One lets $S_{dir,j} : \mathbb{R}^3 \times \mathbb{F} \times \mathbb{R} \rightarrow \mathbb{R}^3$ represent the process of given a position $x \in \mathbb{R}^3$ using the tree $T_{dir,j}$ at that position to compute a direction q , and the tree $T_{dist,j}$ at that position to compute a distance d , one obtains a new position $y = x + dq$. In other words one can estimate the joint position y given some position x by first computing the direction towards the new position, then scaling that direction by an estimated distance, and finally adding it to the current position. So if one has a position x , one obtains the estimated position y as $y = S_{dir,j}(x, f, r)$ for a given frame f and random number r . The random number r is used to randomly select the cluster at the leaf node of the tree $T_{dir,j}$.

[63] It is possible to also define $S_{offset,j} : \mathbb{R}^3 \times \mathbb{F} \times \mathbb{R} \rightarrow \mathbb{R}^3$ to represent the process of given a position x , estimating the offset $p \in \mathbb{R}^3$ using the tree $T_{offset,j}$, and then computing the estimated joint position $y \in \mathbb{R}^3$ as $y = x + p$. For every candidate position $p_{i,j} \in \mathbb{R}^3$, one estimates the distance $d_{i,j}$ to the true position by evaluating the tree $T_{dist,j}$ at that candidate position.

[64] The optimization problem for determining the true location of the joint, and for selecting a candidate point, is then preferably performed at 4210. Optimization can be performed with regard to relative length constraints between each pair of joints and/or according to constraints on the angles. Step 4210 can be performed according to dynamic optimization, with relative length constraints between each pair of joints, as follows. The skeletal model determines the range of lengths that are possible. The dynamic optimization algorithm then can solve the location of the candidate point representing a joint for each pair of joints, according to the relative lengths that are possible between each pair of joints, after which the algorithm determines a global solution. It is not necessary to consider all possible assignments, but only assignments for every pair of connected joints. To perform this calculation, templates for modeling the human skeletal structure are used. These templates can optionally be the previously described templates or alternatively may be obtained from <http://h-anim.org/Specifications/H-Anim1.1/appendices.html#appendixa>. The below described objective function is used, which takes an edge (distance between each pair of joints) and information from the template, and returns the distance between two joints in the template.

[65] Specifically, the objective function may be defined as follows. One assumes that the joints are vertices of a tree-structured graph, connected by a set of edges $E \subset J \times J$. Furthermore, one considers N_t templates indexed by $t \in T$ with $T = \{1, \dots, N_t\}$ and introduce a function $D : E \times T \rightarrow R$ which provides the length in meters of an edge in the skeleton template. The solution of the optimization problem is a vector $I \in J^{N_c}$ which is the index of the candidate associated with every joint, with i_j representing the candidate index associated with joint j : $I = (i_1, i_2, \dots, i_{N_j})$.

$$\begin{array}{c}
 \text{Minimize} \\
 I \in J^{N_c}, t \in T
 \end{array}
 \underbrace{\sum_{j=1}^{N_j} d_{i_j, j}}_{\text{Data term}} + \alpha \underbrace{\sum_{(m,n) \in E} \left[\underbrace{D((m,n), t)}_{\text{Template edge length}} - \underbrace{|p_{i_m, m} - p_{i_n, n}|}_{\text{Actual edge length}} \right]}_{\text{Edge length violation}}$$

Regularization

[66] The parameters that are to be optimized are the skeleton template index t and the vector I of candidate assignments for all joints. The left sum of the objective function is the *data term* of the optimization problem: For a candidate i_j and a joint j , that estimated distance is $d_{i_j, j}$. The right sum of the objective function is the *regularization term* of the optimization problem: Given a template t , for every edge (m, n) of the tree-structured skeleton, where m and n refer to joints, by how much the template distance $D((m, n), t)$ is violated by the two positions $p_{i_m, m}$ and $p_{i_n, n}$ of the candidates. All solutions of the above optimization problem can be enumerated and

the number of possible solutions is $N_c^{N_j} \cdot N_t$

[66] The alpha parameter is heuristically determined and has been found to have useful values particularly between 0.4 and 1. Alpha is set higher for noisy data, and lower for clean data. The alpha term is at least 0.001 for clean data. To solve the optimization problem in a computationally efficient manner, because of the tree-structure of the skeleton, it is possible to use the *dynamic optimization* approach to compute the global optimum in a reasonable amount of time, as noted above. One non-limiting example of a suitable dynamic optimization method that could be applied

to this problem is described in Felzenszwalb, Pedro F., and Ramin Zabih. "Dynamic programming and graph algorithms in computer vision." IEEE transactions on pattern analysis and machine intelligence 33.4 (2011): 721-740 (available at <http://www.cs.cornell.edu/~rdz/Papers/FZ-survey.pdf>).

[67] At 4212, optionally at least one trained classifier tree is used to assess the appropriateness of each candidate point as representing the location of the joint. Preferably, the trees are trained to determine a direction from a point to a joint and also to estimate the distances from a given 3D position to a particular joint location. The trees can be used to determine, for each candidate position, how good the candidate is for explaining the actual location of the joint. Two different sets of trees, in some embodiments, can be used as previously described. Since direction alone doesn't determine how close a candidate is to a particular location, in some embodiments, distance trees are also used. Optionally, the trees can be trained such that, given a particular data point, how close is that point to a particular joint. With respect to the notation used herein, the assignment is further refined by applying either $S_{dirdist,j}$ or $S_{offset,j}$ to that position, or preferably both, in order to obtain a position which could be closer to the true joint position.

[68] Optionally, at 4214, the candidate points can be used as features to a second process for refinement, to better localize the joints. Optionally, steps 4206-4212 of the algorithm can be repeated but with these candidate points as the initial estimates (according to some embodiments). The process can optionally be repeated more than once, optionally with additional, different features.

[69] Also optionally, the initial training process for training the previously described trees could be improved from the method described in "Random Tree Walk toward Instantaneous 3D Human Pose Estimation" by Jung et al, June 2015 CVPR conference (available from https://www.researchgate.net/publication/275771971_Random_Tree_Walk_toward_Instantaneous_3D_Human_Pose_Estimation). In that paper, the feature used for partitioning the training samples was determined as shown in equation 6:

$$f_{\theta}(F, x) = d_F \left(x + \frac{t_1}{d_F(x)} \right) - d_F \left(x + \frac{t_2}{d_F(x)} \right)$$

where F is the current frame, $x \in \mathbb{R}^3$ is the 3d position where we evaluate the feature, $d_F(x)$ is the image depth at x and $t_1 \in \mathbb{R}^3$ and $t_2 \in \mathbb{R}^3$ are the two offsets.

[70] A new feature f_{θ}^1 can be added, for example, as follows, assuming that the current position has components x_1, x_2 and x_3 , that is $x = (x_1, x_2, x_3)$, thereby comparing the depth of the current point with the depth at some random nearby point:

$$f_{\theta}^1(F, x) = d_F \left(x + \frac{t_1}{d_F(x)} \right) - x_3$$

where $t_1 \in \mathbb{R}^3$ is a random offset.

[71] A further refinement could optionally feature training a new set of trees that also include previous estimates. This would include a feature:

$$f_{u,j}^2(x) = u \cdot (x - p_j^*)$$

[72]

[73] where $u \in \mathbb{R}^3$ with $|u| = 1$ is a randomly chosen unit vector, j is the joint index, $x \in \mathbb{R}^3$ is the position where one evaluates the feature and the symbol “ \cdot ” represents the dot product of two vectors. In other words, this feature computes the offset with a given point $x \in \mathbb{R}^3$ and a previous estimate p_j^* of that point to obtain a feature which takes previous knowledge into account. This can be used to train trees as before using both this new feature for different joints j and random unit vectors u , in addition to (or in place of) the features provided in the previously described paper.

[74] For a given frame, first position p_j^* is detected using the trees that were trained. Then the algorithm is rerun, but using the second set of trees that were trained, which in addition to

image data also uses the detections p_j^* for the second feature. This refined position can be used as a new input feature and then obtain for every joint an even better refinement p_j^{*3} . It is possible to repeat this process as many times as desired. If one performs this process n times, for every joint j there will be a position p_j^{*n} .

[75] Without wishing to be limited by a single hypothesis, the method of Figure 42 has a number of advantages over “Random Tree Walk toward Instantaneous 3D Human Pose Estimation” by Jung et al, June 2015 CVPR conference (available from https://www.researchgate.net/publication/275771971_Random_Tree_Walk_toward_Instantaneous_3D_Human_Pose_Estimation) and/or according to the description provided in “Efficient Human Pose Estimation from Single Depth Images” by Shotton et al, 2011 (available from <https://people.eecs.berkeley.edu/~rbg/papers/pose-from-depth-pami.pdf>). The method described in the Jung et al paper suffers from a number of disadvantages, including with regard to the core random walk algorithm, which tends to diverge significantly from the actual desired fitted model. The method described in the Shotton et al. paper requires each pixel to vote on a particular location, first to determine background as opposed to the subject, and then to determine the location of each joint within the subject, as the points are determined as offset from the joints; this method is computationally very expensive.

[76] Optionally, at least some of the method embodiments disclosed herein may be combined with SLAM (simultaneous location and mapping), as described in, for example, “SYSTEMS, METHODS, DEVICE AND APPARATUSES FOR PERFORMING SIMULTANEOUS LOCALIZATION AND MAPPING”, owned in common with the present application and having at least one inventor in common with the present application, filed on January 19, 2018 as PCT Application No. PCT/US18/14508, the entire disclosure of which is hereby incorporated by reference as if fully set forth herein.

[77] For example and without limitation, a SLAM system as described therein is provided for a wearable device, including without limitation, a head-mounted wearable device that optionally includes a display screen. Such systems, methods and apparatuses can be configured to accurately (and in some embodiments, quickly) localize a wearable device within a dynamically constructed map, e.g.,

through computations performed with a computational device. A non-limiting example of such a computational device is a smart cellular phone or other mobile computational device. To this end, according to at least some embodiments, SLAM systems, methods and apparatuses can support a VR (virtual reality) application, an AR (augmented reality) application, and/or the like.

[78] Also optionally, at least some of the method and system embodiments as described herein may be combined with a facemask, as described for example in any of “SYSTEM, METHOD, DEVICES AND APPARATUSES FOR DETECTING FACIAL EXPRESSION IN A VIRTUAL REALITY SYSTEM”, filed on January 19, 2018 as U.S. Patent Application No. 15875306; “SYSTEM, METHOD AND APPARATUS FOR DETECTING FACIAL EXPRESSION IN AN AUGMENTED REALITY SYSTEM”, filed on January 19, 2018 as U.S. Patent Application No. 15875382; and/or “SYSTEMS, METHODS, DEVICES AND APPARATUSES FOR DETECTING FACIAL EXPRESSION”, filed on January 19, 2018 as U.S. Patent Application No. 15875227; each of which is owned in common with the present application and having at least one inventor in common with the present application, and each of which is hereby incorporated by reference as if fully set forth herein.

[79] For example and without limitation, such a facemask may be provided as apparatuses, methods, and systems which facilitate a rapid, efficient mechanism for facial expression detection according to electromyography (EMG) signals. In some implementations, such apparatuses, methods and system can detect facial expressions according to EMG signals that can operate without significant latency on mobile devices (including but not limited to tablets, smartphones, and/or the like).

[80] Accordingly, such a facemask may be used for example for VR (virtual reality) or AR (augmented reality) applications, in which the face of the user may be partially or completely obscured by headgear or other equipment for facilitating the VR or AR experience. The facemask may then be used for detecting an emotional state of the user.

[81] Figures 43-46 relate to combined methods which may combine various of the previously described methods and which may be implemented with any of the above described systems.

[82] **Figures 43** and **44** show non-limiting examples of methods for applying VR (via, for example a VR application) to medical therapeutics according to at least some embodiments of the present disclosure, which also feature tracking of the user’s body and SLAM for localizing the user within the surrounding environment. **Figure 43** shows an example of a method for applying

VR to medical therapeutics – e.g., assisting an amputee to overcome phantom limb syndrome. The method includes, at 4302, the morphology of the body of the user (e.g., an amputee) or a portion thereof, such as the torso and/or a particular limb, may be determined, through scanning (for example). Such scanning may be performed in order to create a more realistic avatar for the user to view in the VR environment, enabling the user when “looking down” in the VR environment, to see body parts that realistically appear to “belong” to the user’s own body.

[83] At 4304, optionally, a familiar environment for the user is scanned, where such scanning may be performed to create a more realistic version of the environment for the user in the VR environment. The user may then look around the VR environment and see virtual objects that correspond in appearance to real objects with which the user is familiar.

[84] The user enters the VR environment (at 4306), for example, by donning a wearable device (as described herein) and/or otherwise initiating the VR application. For this non-limiting method, optionally, a tracking sensor may be provided to track one or more physical actions of the user, such as one or more movements of one or more parts of the user’s body. A non-limiting example of such a tracking sensor is the Kinect of Microsoft, or the Leap Motion sensor, as previously described, but preferably as any of the tracking implementations described herein. Also as described herein, any of the tracking implementations described herein may be adjusted or calibrated for the absence of the missing limb, in order to provide accurate tracking of the user’s body.

[85] At 4308, the user “views” the phantom limb – that is, the limb that was amputated – as still being attached to the body of the user. For example, if the amputated limb was the user’s left arm, then the user then sees his/her left arm as still attached to his/her body as a functional limb, within the VR environment. Optionally, in order to enable the amputated limb to be actively used, the user’s functioning right arm can be used to create a “mirror” left arm. In this example, when the user moved his/her right arm, the mirrored left arm appears to move and may be viewed as moving in the VR environment. If a familiar environment for the user was previously scanned, then the VR environment can be rendered to appear as that familiar environment, which can lead to powerful therapeutic effects for the user, for example, as described below in regard to reducing phantom limb pain. At 2510, the ability to view the phantom limb is optionally and preferably incorporated into one or more therapeutic activities performed in the VR environment.

[86] The facial expression of the user may be monitored while performing these activities, for example to determine whether the user is showing fatigue or distress (4312). Optionally, the user's activities and facial expression can be monitored remotely by a therapist ready to intervene to assist the user through the VR environment, for example, by communicating with the user (or being an avatar within the VR environment).

[87] At 4314, the body of the user is tracked, so that the location of one or more body parts of the user may be determined over time. Such tracking may be used for example to increase the realistic view of the user's avatar body and phantom limb, as well as to more realistically render the interactions of the user's avatar with the environment. Optionally SLAM is also included, again to more realistically locate the user within the environment. The combination of SLAM and tracking may for example be used to evaluate the movements of the user more effectively, for example to determine the range and/or quality of motion of the user, and the like.

[88] One of skill in the art will appreciate that the above described method may be used to reduce phantom limb pain (where an amputee feels strong pain that is associated with the missing limb). Such pain has been successfully treated with mirror therapy, in which the amputee views the non-amputated limb in a mirror (see, for example, the article by Kim and Kim, "Mirror Therapy for Phantom Limb Pain", Korean J Pain. 2012 Oct; 25(4): 272–274). The VR environment described herein can provide a more realistic and powerful way for the user to view and manipulate the non-amputated limb, and hence to reduce phantom limb pain.

[89] **Figure 44** shows another non-limiting example method for applying VR to medical therapeutics according to at least some embodiments of the present disclosure, which can provide a therapeutic environment to a subject who has suffered a stroke, for example (e.g., brain injury). In this non-limiting example, the subject is encouraged to play the game of "Simon says" in order to treat hemispatial neglect. In the game of "Simon says", one player (which in this example may be a VR avatar) performs an action which the other players are to copy – but only if the "Simon" player says "Simon says (perform the action)". Of course, this requirement may be dropped for this non-limiting example, which is described only in terms of viewing and copying actions by the user. 4402-4406 may be similar to 4302-4306 of **Figure 43**.

[90] At 4408, the user views a Simon avatar, which is optionally another player (such as a therapist) or alternatively is a non-player character (NPC) generated by the VR system. Preferably

the user perceives the Simon avatar as standing in front of him or her, and as facing the user. The user optionally has his or her own user avatar, which represents those parts of the user's body that is normally be visible to the user according to the position of the user's head and body. This avatar is referred to in this non-limiting example as the user's avatar.

[91] At 2610, the Simon avatar can initiate an action, which the user is to mimic with the user's own body. The action includes movement of at least one body part and optionally includes a facial expression as well. At 2612, the user copies – or at least attempts to copy – the action of the Simon avatar. The user can see the Simon avatar, as well as those parts of the user's avatar that are expected to be visible according to the position of the user's head and body. Optionally, for 2610 and 2612, the user's avatar can also be placed in front of the user, for example, next to the Simon avatar. The user can then see both the Simon avatar, whose visual action(s) the user would need to copy, and how the user's body is actually performing those actions with the user's avatar. The motions of the user may be tracked as described herein, in order to increase the realism of the rendering of the user's avatar and the corresponding movements displayed by that avatar. SLAM may also be used, alone or in combination with tracking, as previously described, to increase the realism of the environment and the actions of the user within that environment, also optionally in regard to movements of the avatar.

[92] At 2614, if the user fails to accurately/correctly copy the action of the Simon avatar, that avatar preferably repeats the action. This process may continue for a predetermined period of rounds or until the user achieves at least one therapeutic goal. At 2616, the ability of the user to perform such actions may be optionally scored, such scoring may include separate scores for body actions and facial expressions. Such scoring may be augmented or performed with the assistance of tracking that is performed as described herein, in order to more accurately assess the actions of the user. SLAM may also be combined to further increase the accuracy of the assessment. At 2618, the facial expressions of the user while performing the actions can be monitored, even if the actions do not include a specific facial expression, so as to assess the emotions of the user while performing these actions.

[93] **Figure 45** shows a non-limiting example method for applying VR to increase a user's ability to perform ADL (activities of daily living) according to at least some embodiments. 4502-4506 may be similar to 4302-4306 of **Figure 43**.

[94] In 4508, the user's action range is optionally calibrated as previously described, in order to determine the user's range of motion for a particular action or set of actions, such as for example for a particular gesture or set of gestures. For example, and without limitation, if the user is not capable of a normal action range, then the system may be adjusted according to the range of action of which the user is capable. In 4510, the user reaches for a virtual object in the VR environment, as a non-limiting example of an activity to be performed in the VR environment, for example as a therapeutic activity.

[95] In 4512, the user's capabilities are assessed, for example in terms of being able to reach for and grasp the virtual object, or in terms of being able to perform the therapeutic task in the VR environment. Optionally, in 4514, the user is asked to copy an action, for example being shown by a system or "Simon" avatar. Such an action may be used to further determine the user's capabilities. The performance of such an evaluation may be performed with the assistance of tracking that is performed as described herein, in order to more accurately assess the actions of the user. SLAM may also be combined to further increase the accuracy of the assessment.

[96] The system may then determine which action(s) need to be improved in 4516, for example in order to improve an activity of daily living. For example, and without limitation, the user may need to improve a grasping action in order to be able to manipulate objects as part of ADL. One or more additional therapeutic activities may then be suggested in 4518. The process may be repeated, with the user being assessed in his/her ability to perform ADL actions and also in terms of any improvement thereof.

[97] **Figure 46** shows a non-limiting example method for applying AR to increase a user's ability to perform ADL (activities of daily living) according to at least some embodiments. Steps 4602-4606 may be similar to 4302-4306 of **Figure 25**.

[98] In 4608, the user's action range is optionally calibrated as previously described, in order to determine the user's range of motion for a particular action or set of actions, such as for example for a particular gesture or set of gestures. For example, and without limitation, if the user is not capable of a normal action range, then the system may be adjusted according to the range of action of which the user is capable. In 4610, the user reaches for an actual object or a virtual object in the AR environment, as a non-limiting example of an activity to be performed in the AR environment, for example as a therapeutic activity. However, optionally the user reaches at least once for a

virtual object and at least once for an actual object, in order to determine the capabilities of the user in terms of interacting with actual objects. Furthermore, by doing both, the user's abilities can be assessed in both the real and the virtual environments. Optionally, in some embodiments, the AR environment is used for diagnosis and testing, while the VR environment is used for training and other therapeutic activities.

[99] In 4612, the user's capabilities are assessed, for example in terms of being able to reach for and grasp the virtual and/or real object, or in terms of being able to perform the therapeutic task in the AR environment. Optionally, in 4614, the user is asked to copy an action, for example being shown by a system or "Simon" avatar. Such an action may be used to further determine the user's capabilities. The performance of such an evaluation may be performed with the assistance of tracking that is performed as described herein, in order to more accurately assess the actions of the user. SLAM may also be combined to further increase the accuracy of the assessment. Tracking and/or SLAM may also be used to assess the actions of the user between the VR and AR environments, for example to determine whether the assessment in the VR environment is being performed accurately.

[100] The system may then determine which action(s) need to be improved in 4616, for example in order to improve an activity of daily living. For example, and without limitation, the user may need to improve a grasping action in order to be able to manipulate objects as part of ADL. One or more additional therapeutic activities may then be suggested in 4618. The process may be repeated, with the user being assessed in his/her ability to perform ADL actions and also in terms of any improvement thereof.

```

[101] APPENDIX (partial description of a skeleton)
[102]
[103] # Everything is expressed in world coordinate
[104] ArticulatedModel =
[105] {
[106]     global_scale = 0.80;
[107]     dense_mesh = true;
[108]     bones = (
[109]     {
[110]         name = "sacrum";
[111]         head = [-0.03382374346256256, -0.907242476940155, -
0.39482375979423523];
[112]         tail = [-0.03382374346256256, -1.063200831413269, -
0.40458834171295166];
[113]         x_axis = [0.999997615814209, -
0.00013603654224425554, 0.0021727425046265125];

```

```

[114]         y_axis = [-2.764863893389702e-10, -
0.9980457425117493, -0.06248803436756134];
[115]         z_axis =
[0.0021769972518086433, 0.062487829476594925, -0.9980433583259583];
[116]         parent = "None";
[117]         ndof = 3;
[118]         initial_theta = [0.0, 0.0, 0.0];
[119]         offset = [0.0, 0.0, 0.0];
[120]         active = True;
[121]     },
[122]     {
[123]         name = "l_thigh";
[124]         head = [0.07265270501375198, -0.9023483991622925, -
0.3843427300453186];
[125]         tail = [0.11466878652572632, -0.49611833691596985, -
0.41821298003196716];
[126]         x_axis = [0.9938485026359558, -0.10558056086301804, -
0.03343289718031883];
[127]         y_axis = [0.10252843052148819, 0.9912905097007751, -
0.08265086263418198];
[128]         z_axis =
[0.04186808317899704, 0.07871462404727936, 0.996017575263977];
[129]         limit_rotation_x = [-
1.5707963705062866, 0.7853981852531433];
[130]         limit_rotation_y = [-
0.7853981852531433, 0.7853981852531433];
[131]         limit_rotation_z = [-
0.7853981852531433, 0.7853981852531433];
[132]         parent = "sacrum";
[133]         ndof = 3;
[134]         initial_theta = [0.0, 0.0, 0.0];
[135]         offset =
[0.10647644847631454, 0.16085243225097656, 0.020245611667633057];
[136]         active = False;
[137]     },

```

[0239]

[0240] While various inventive embodiments have been described and illustrated herein, those of ordinary skill in the art will readily envision a variety of other means, structures, steps, and/or functionality for performing the function and/or obtaining the results and/or one or more of the advantages described herein, and each of such variations and/or modifications is deemed to be within the scope of the inventive embodiments described herein. More generally, those skilled in the art will readily appreciate that all parameters, dimensions, materials, structure, functionality, steps, processes, and configurations described herein are meant to be exemplary and that the actual parameters, dimensions, materials, structure, functionality, steps, processes, and configurations will depend upon the specific application or applications for which the inventive teachings is/are

used. Those skilled in the art will recognize, or be able to ascertain using no more than routine experimentation, many equivalents to the specific inventive embodiments described herein. It is, therefore, to be understood that the embodiments disclosed herein are presented by way of example only and that, such embodiments (and any embodiments supported by the present disclosure either expressly, implicitly or inherently) may be practiced otherwise than as specifically described and claimed. Some embodiments of the present disclosure are directed to each individual feature, system, function, article, material, instructions, step, kit, and/or method described herein, and any combination of two or more such features, systems, functions, articles, materials, kits, steps, and/or methods, if such features, systems, functions, articles, materials, kits, steps and/or methods are not mutually inconsistent, is included within the inventive scope of the present disclosure. Additionally, some embodiments of the present disclosure are inventive over the prior art by specifically lacking one and/or another feature/functionality disclosed in such prior art (i.e., claims to such embodiments can include negative limitations to distinguish over such prior art).

[0241] Also, various inventive concepts may be embodied as one or more steps/methods, of which examples have been provided. The acts performed as part of the method may be ordered in any suitable way. Accordingly, embodiments may be constructed in which acts are performed in an order different than illustrated, which may include performing some acts simultaneously, even though shown as sequential acts in illustrative embodiments.

[0242] Any and all references to publications or other documents, including but not limited to, patents, patent applications, articles, webpages, books, etc., presented in the present application, are herein incorporated by reference in their entirety.

[0243] All definitions, as defined and used herein, should be understood to control over dictionary definitions, definitions in documents incorporated by reference, and/or ordinary meanings of the defined terms.

WHAT IS CLAIMED IS:

1. A stereo vision procurement apparatus for obtaining stereo visual data, comprising:
 - a stereo RGB camera;
 - a depth sensor; and
 - an RGB-D fusion module,wherein:
 - each of said stereo RGB camera and said depth sensor are configured to provide pixel data corresponding to a plurality of pixels,
 - said RGB-D fusion module is configured to combine RGB pixel data from said stereo RGB camera and depth information pixel data from said depth sensor to form stereo visual pixel data (SVPD), and
 - said RGB-D fusion module is implemented in an FPGA field-programmable gate array).

2. The apparatus of claim 1, further comprising a de-mosaicing module configured to perform a method comprising:
 - averaging the RGB pixel data associated with a plurality of green pixels surrounding red and blue sites for R(B) at B-G(R-G) sites or R(B) at R-G(B-G) sites, and
 - reducing a number of green pixel values from the RGB pixel data to fit a predetermined pixel array (e.g., a 5 x 5 window) for R(B) at B(R) sites.

3. The apparatus of claim 2, wherein:
 - said stereo RGB camera comprises a first camera and a second camera,
 - each of said first and second cameras being associated with a clock on said FPGA, and
 - said FPGA including a double clock sampler for synchronizing said clocks of said first and right cameras.

4. The apparatus of claim 3, further comprising:
 - a histogram module comprising a luminance calculator for determining a luminance level of at least said RGB pixel data; and
 - a classifier for classifying said RGB pixel data according to said luminance level, wherein said luminance level is transmitted to said stereo RGB camera as feedback.
5. The apparatus of claim 4, further comprising a white balance module configured to apply a smoothed GW (gray world) algorithm to said RGB pixel data.
6. The apparatus of any of the above claims, further comprising:
 - a processor; and
 - a biological sensor configured to provide biological data,wherein:
 - said biological sensor is selected from the group consisting of: an EEG sensor, a heartrate sensor, an oxygen saturation sensor, an EKG sensor, or EMG sensor, and a combination thereof,
 - the processor is configured to process the biological data to form a plurality of sub-features,
 - said sub-features are combined by the FPGA to form a feature.
7. The apparatus of any of the above claims, wherein said FPGA is implemented as a field-programmable gate array (FPGA) comprising a system on a chip (SoC), including an operating system as a SOM (system on module).
8. The apparatus of claim 7, further comprising a CPU SOM for performing overflow

operations from said FPGA.

9. The apparatus of any of the above claims, further comprising:
a processor; and
a plurality of tracking devices to track movement of a subject,
wherein:

the processor is configured to process data from the tracking devices to form a plurality of sub-features, and

said sub-features are combined by said FPGA to form a feature to track movements of the subject.

10. The apparatus of claim 9, wherein said tracking devices comprise a plurality of wearable sensors.

11. The apparatus of claims 9 or 10, further comprising:

a processor; and

a multi-modal interaction device in communication with a subject, said multi-modal interaction device comprising said plurality of tracking devices and at least one haptic feedback device,

wherein:

the processor is configured to process data from the tracking devices to form a plurality of tracking sub-features, and

said sub-features are combined by said FPGA to form a feature to track movements of the subject and to provide feedback through said at least one haptic feedback device.

12. The apparatus of any of the above claims, further comprising

a processor configured to perform a defined set of operations in response to receiving a corresponding instruction selected from an instruction set of codes; and

a memory;

wherein:

said defined set of operations including:

a first set of codes for operating said RGB-D fusion module to synchronize RGB pixel data and depth pixel data, and for creating a disparity map; and

a second set of codes for creating a point cloud from said disparity map and said depth pixel data.

13. The apparatus of claim 12, wherein said point cloud comprises a colorized point cloud.
14. The apparatus of any of claims 1-11, further comprising:
a memory; and
a processor configured to perform a defined set of operations for performing any of the functionality recited in any of claims 1-11 in response to receiving a corresponding instruction selected from an instruction set of codes.
15. The apparatus of claim 14, wherein said processor is configured to operate according to a set of codes selected from the instruction set for a de-noising process for a CFA (color filter array) image according to a W-means process.
16. The apparatus of claims 14 or 15, wherein said computational device comprises a second set of codes selected from the instruction set for operating a bad pixel removal process.
17. A system comprising the apparatus of any of the above claims, further comprising a display

for displaying stereo visual data.

18. The system of claim 17, further comprising:

an object attached to a body of a user; and

an inertial sensor,

wherein

said object comprises an active marker,

input from said object is processed to form a plurality of sub-features, and

said sub-features are combined by the FPGA to form a feature.

19. The system of any of the above claims, further comprising a processor for operating a user application, wherein said RGB-D fusion module is further configured to output a colorized point cloud to said user application.

20. The system of claim 19, wherein:

said processor is configured to transfer SVPD to said display without being passed to said user application, and

said user application is additionally configured to provide additional information for said display that is combined by said FPGA with said SVPD for output to said display.

21. The system of claims 19 or 20, wherein said biological sensor is configured to output data via radio-frequency (RF), and wherein:

the system further comprises an RF receiver for receiving the data from said biological sensor, and

said feature from said FPGA is transmitted to said user application.

22. The system of any of claims 17-21, further comprising at least one of a haptic or tactile feedback device, the device configured to provide at least one of haptic or tactile feedback, respectively, according to information provided by said user application.
23. A stereo vision procurement system comprising:
a first multi-modal interaction platform configurable to be in communication with one or more additional second multi-modal interaction platforms;
a depth camera;
a stereo RGB camera; and
an RGB-D fusion chip;
wherein:
each of said stereo RGB camera and said depth camera are configured to provide pixel data corresponding to a plurality of pixels,
the RGB-D fusion chip comprises a processor operative to execute a plurality of instructions to cause the chip to fuse said RGB pixel data and depth pixel data to form stereo visual pixel data.
24. The system of claim 23, wherein the depth camera is configured to provide depth pixel data according to TOF (time of flight).
25. The system of claims 23 or 24, wherein the stereo camera is configured to provide SVPD from at least one first and at least one second sensor.
26. The system of any of claims 23-25, wherein the RGB-D fusion chip is configured to preprocess at least one of SVPD and depth pixel data so as to form a 3D point cloud with RGB pixel data associated therewith.
27. The system of claim 26, wherein the fusion chip is further configured to form the 3D point

- cloud for tracking at least a portion of a body by at least the first multi-modal interaction platform.
28. The system of any of claims 23-27, further comprising at least one of a display and a wearable haptic device, wherein at least the first multi-modal interaction platform is configured to output data to at least one of the display and the haptic device.
29. The system of any of claims 23-28, further comprising one or more interactive objects or tools configured to perform at least one of giving feedback, receiving feedback, and receiving instructions from at least one of the multi-modal interaction platforms.
30. The system of any of claims 23-29, further comprising one or more sensors configured to communicate with at least one of the multi-modal interaction platforms.
31. The system of claim 30, wherein the one or more sensors include at least one of:
- a stereo vision AR (augmented reality) component configured to display an AR environment according to at least one of tracking data of a user and data received from the first multi-modal interaction platform, and a second additional multi-modal interaction platform;
 - an object tracking sensor;
 - a facial detection sensor configured to detect a human face, or emotions thereof; and
 - a markerless tracking sensor in which an object is tracked without additional specific markers placed on it.
32. A multi-modal interaction platform system comprising:
- a multi-modal interaction platform;
 - a plurality of wearable sensors each comprising an active marker configured to provide an

- active signal for being detected;
 - an inertial sensor configured to provide an inertial signal comprising position and orientation information;
 - at least one of a heart rate and oxygen saturation sensor, or a combination thereof;
 - an EEG sensor; and
 - at least one wearable haptic devices, including one or more of a tactile feedback device and a force feedback device.
33. A method for processing image information comprising:
- receiving SVPD from a stereo camera;
 - performing RGB preprocessing on the input pixel data to produce preprocessed RGB image pixel data;
 - using the RGB preprocessed image pixel data in the operation of the stereo camera with respect to at least one of an autogain and an autoexposure algorithm;
 - rectifying the SVPD so as to control artifacts caused by the lens of the camera; and
 - calibrating the SVPD so as to prevent distortion of the stereo pixel input data by the lens of the stereo camera.
34. The method of claim 33, further comprising colorizing the preprocessed RGB image pixel data, and creating a disparity map based on the colorized, preprocessed RGB image pixel data.
35. The method of claims 33 or 34, wherein calibration comprises matching the RGB pixel image data with depth pixel data.
36. The method of any of claims 33-35, wherein the disparity map is created by:

- obtaining depth pixel data from at least one of the stereo pixel input data, the preprocessed RGB image pixel data, and depth pixel data from a depth sensor, and checking differences between stereo images.
37. The method of claim 36, wherein said disparity map, plus depth pixel data from the depth sensor in the form of a calibrated depth map, is combined for the point cloud computation.
38. An image depth processing method for depth processing of one or more images comprising:
- receiving TOF (time-of-flight) image data of an image from a TOF camera;
 - creating at least one of a depth map or a level of illumination for each pixel from the TOF data;
 - feeding the level of illumination into a low confidence pixel removal process comprising:
 - comparing a distance that each pixel is reporting;
 - correlating said distance of said each pixel to the illumination provided by said each pixel,
 - removing any pixel upon the illumination provided by the pixel being outside a predetermined acceptable range such that the distance cannot be accurately determined;
 - processing depth information to remove motion blur of the image, wherein motion blur is removed by removing artifacts at edges of moving objects in depth of the image;
 - and
 - applying at least one of temporal or spatial filters to the image data.
39. A stereo image processing method comprising:
- receiving first data flow of at least one image from a first RGB camera and second data flow of at least one image from a second RGB camera;

- sending the first and second data flows to a frame synchronizer; and
- synchronizing, using the frame synchronizer, a first image frame from the first data flow and a second image frame from the second data flow such that time shift between the first image and frame and the second image frame is substantially eliminated.
40. The method of claim 39, wherein:
- sampling, before sending the first and second data flows to the frame synchronizer, the first and second data flows such that each of the first and second data flows are synchronized with a single clock; and
- detecting which data flow is advanced of the other, and directing the advanced data flow to a First Input First Output (FIFO), such that the data from the advanced flow is retained by the frame synchronizer until the other data flow reaches the frame synchronizer.
41. The method of claim 40, further comprising serializing frame data of the first and second data flows as a sequence of bytes.
42. The method of any of claims 39-41, further comprising detecting non-usable pixels.
43. The method of any of claims 39-42, further comprising constructing a set of color data from each of the first and second data flows.
44. The method of claim 43, further comprising color correcting each of the first and second data flows.
45. The method of any of claims 39-44, further comprising:
- corresponding the first and second data flows into a CFA (color filter array) color image

data;

applying a denoising process for the CFA image data, the process comprising:

grouping four (4) CFA colors to make a 4-color pixel for each pixel of the image data;

comparing each 4-color pixel to neighboring 4-color pixels;

attributing a weight to each neighbor pixel depending on its difference with the center 4-color pixel; and

for each color, computing a weighted mean to generate the output 4-color pixel.

46. The method of claim 45, wherein said denoising process further comprises performing a distance computation according to a Manhattan distance, computed between each color group neighbor and the center color group.

47. The method of claims 45 or 46, further comprising applying a bad pixel removal algorithm before said denoising process.

48. An apparatus according to any of the above claims, for obtaining SVPD and tracking a user, further comprising:

a body model; and

one or more processors having computer instructions operating thereon configured to cause the processor to:

fit data points from the depth sensor to the body model according to a probabilistic fitting algorithm, said probabilistic fitting algorithm being constrained according to at least one constraint defined according to human anatomy,

identifying a plurality of data points as super points and assigning each of said super points an additional weight;

wherein:

a plurality of said data points are identified with joints of the anatomy,
said super points are defined according to one or more objects attached to a body,
each of said stereo RGB camera and said depth sensor are configured to provide data as a plurality of pixels,
said RGB-D fusion module is configured to combine RGB data from said stereo RGB camera and depth information from said depth sensor to SVPD, and
the depth sensor provides data to determine a three-dimensional location of a body in space according to a distance of the body from the depth sensor.

49. The apparatus of claim 48, wherein said one or more objects attached to the body comprise one or more of at least one active marker configured to provide a detectable signal and a passive object.
50. The apparatus of claims 48 or 49, wherein said data points identified with joints of the human body are identified according to a previously determined position as an estimate.
51. The apparatus of any of claims 48-50, wherein
the body model comprises a template,
said template including a standard model of a skeleton according to a hierarchy of joints as vertices and skinning, and
a first determination of a position of at least one of the joints of the hierarchy of joints according to said template.
52. A system comprising the apparatus of any of the above claims, optionally comprising a system of any of the above claims, further comprising a display for displaying SVPD.
53. The system of claim 52, further comprising:
one or more objects attached to the user; and

an inertial sensor,

wherein

said one or more objects comprises an active marker,

the computer instructions are configured to cause the processor to form a plurality of sub-features from input from said one or more objects and combining said sub-features into a feature.

54. The system of any of the above claims, further comprising at least one of a haptic feedback device and a tactile feedback device configured to provide at least one of haptic and tactile feedback according to information provided by said user application.
55. The system of any of the above claims, wherein computer instructions include instructions which cause the processor to perform as a tracking engine.
56. The system of claim 55, wherein the tracking engine is configured to track at least one of the position of the user's body and the position of one or more body parts of the user, including but not limited, to one or more of an arm, a leg, a hand, a foot, and a head.
57. The system of claims 55 or 56, wherein the tracking engine is configured to decompose signals representing physical actions made by the user into data representing a series of gestures.
58. The system of claim 57, wherein the tracking engine is configured to decompose signals representing physical actions into data representing a series of gestures via classifier functionality.
59. The system of any of the above claims, further comprising a plurality of templates, wherein the computer instructions are further configured to cause the processor to initialize a template of the plurality of templates.

60. The system of claim 59, wherein the template features a model of a human body configured only as a plurality of parameters, only as a plurality of features, or both.
61. The system of claim 60, wherein the plurality of parameters and/or features include a skeleton, and one or more joints.
62. The system of claims 60 or 61, wherein the computer instructions are additionally configured to cause the processor to utilize the plurality of parameters and/or features to assist in tracking of the user's movements.
63. The system of any of the above claims, wherein the computer instructions are configured to map the sensor data onto a GMM (Gaussian mixture model).
64. The system of claim 63, wherein the body model includes a sparse-skin representation.
65. The system of any of claims 60-64, wherein the computer instructions are additionally configured to cause the processor to suppress corresponding Gaussians.
66. The system of claim 65, wherein data is mapped to a GMM.
67. The system of claim 66, wherein said data is mapped to said GMM by a classifier.
68. The system of any of claims 65-67, wherein:
the tracking engine includes a template engine configured to read a template from a template database, and
the computer instructions are additionally configured to:
cause the processor to operate as a GMM mapper, and send the template into the GMM

mapper.

69. The system of any of claims 65-68, wherein the computer instructions are additionally configured to cause the processor to operate as a point cloud decomposer, and the GMM mapper is configured to receive point cloud information from the point cloud decomposer.

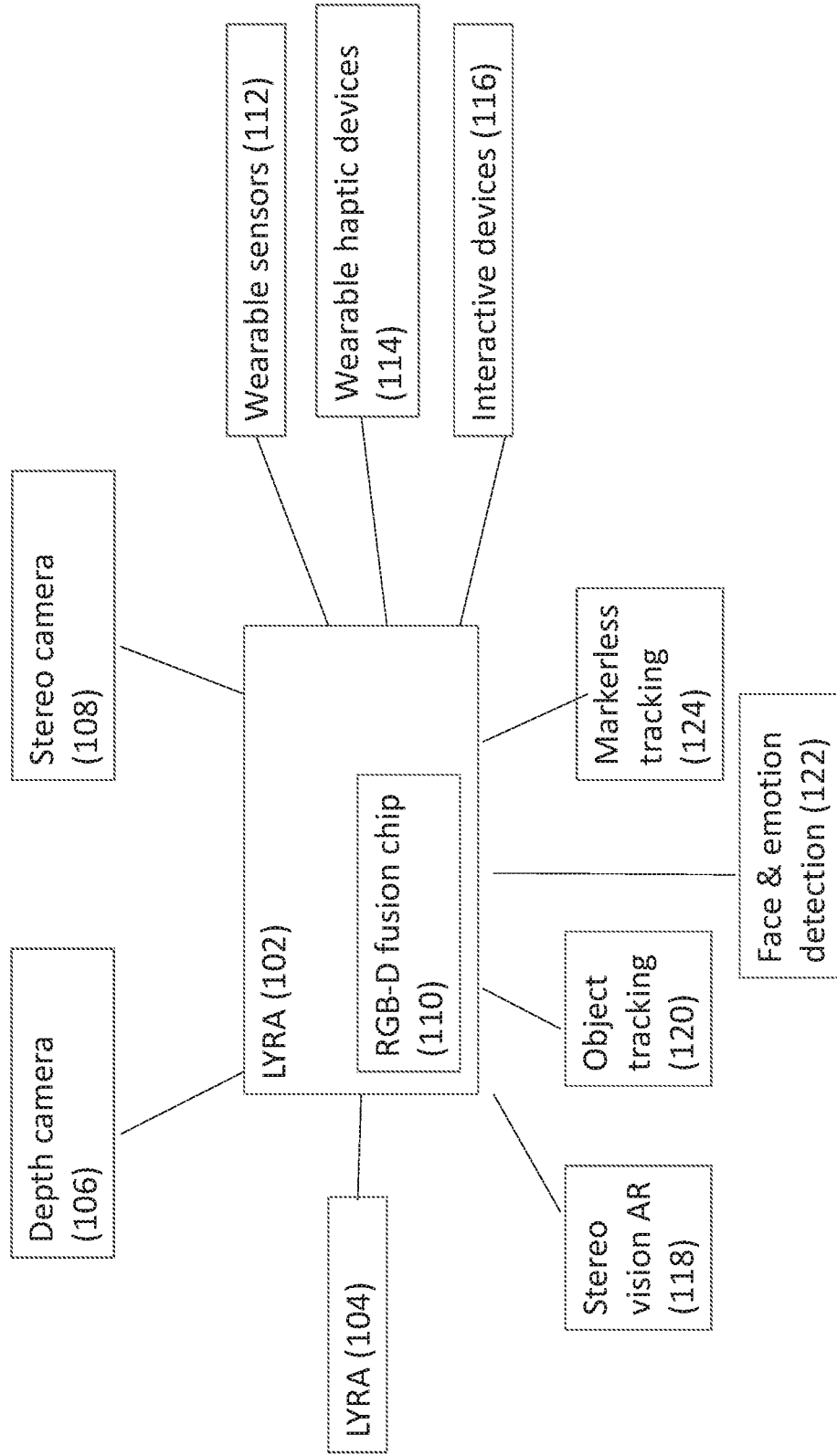
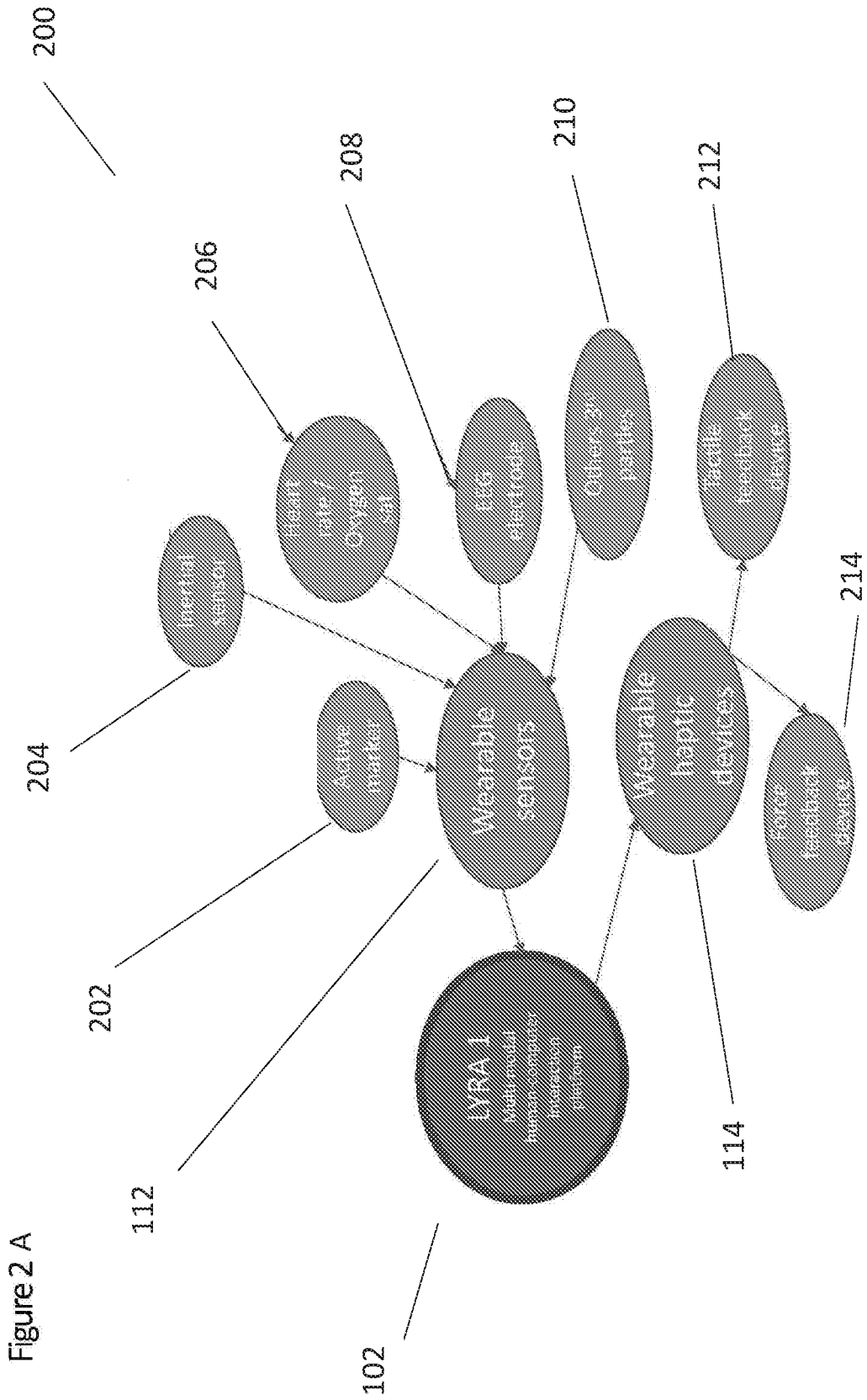


Figure 1



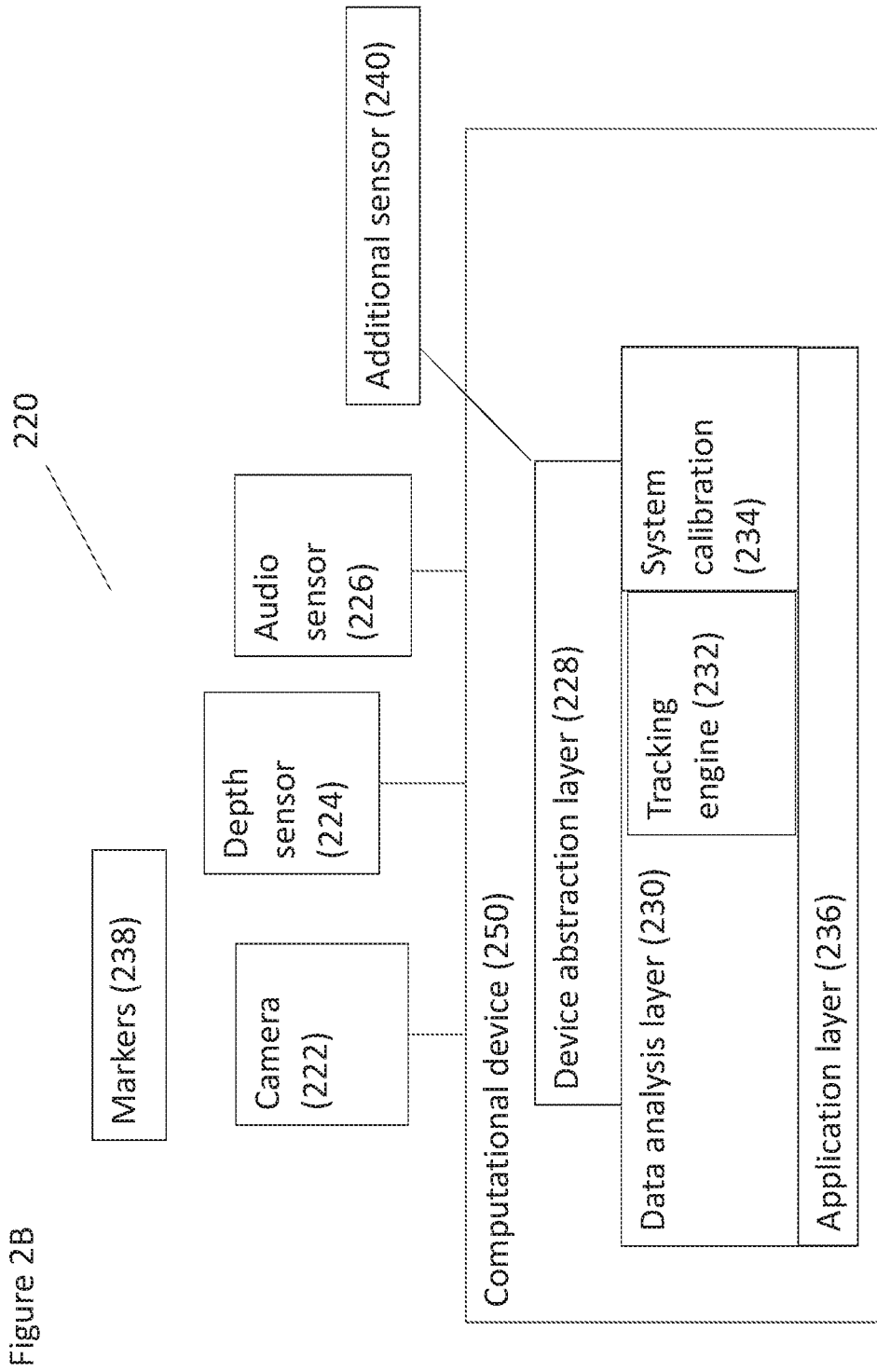


Figure 2C

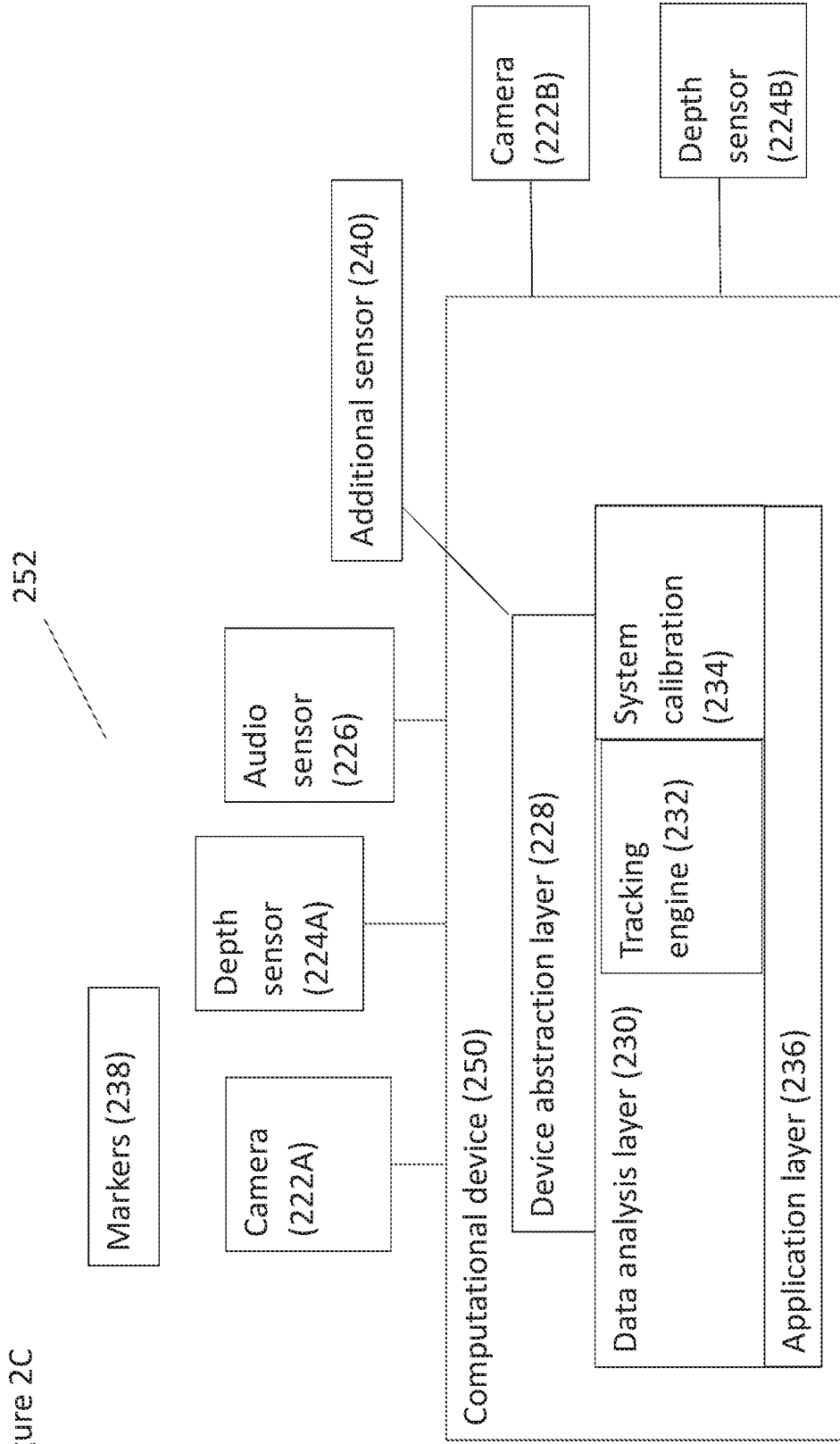


Figure 3

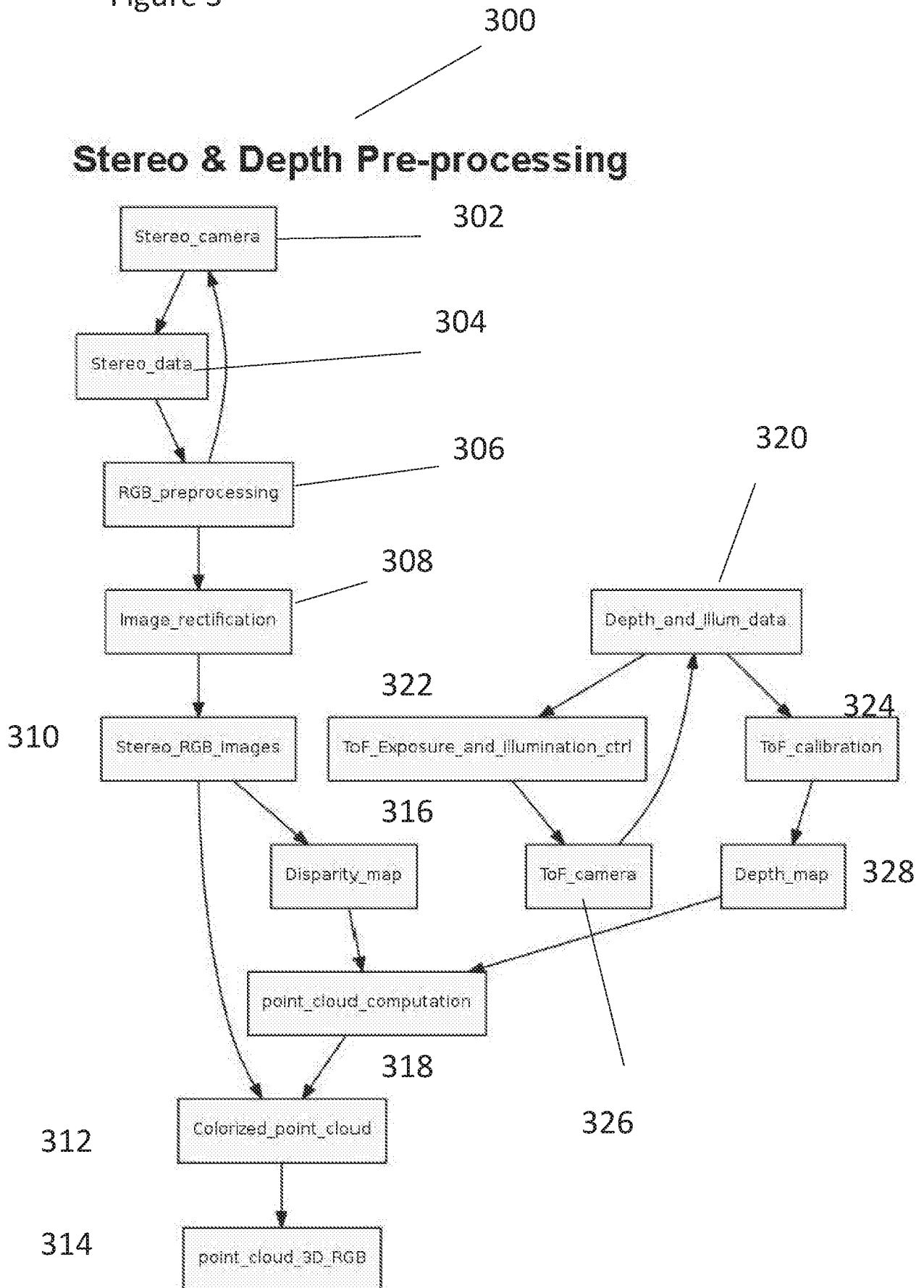


Figure 4A

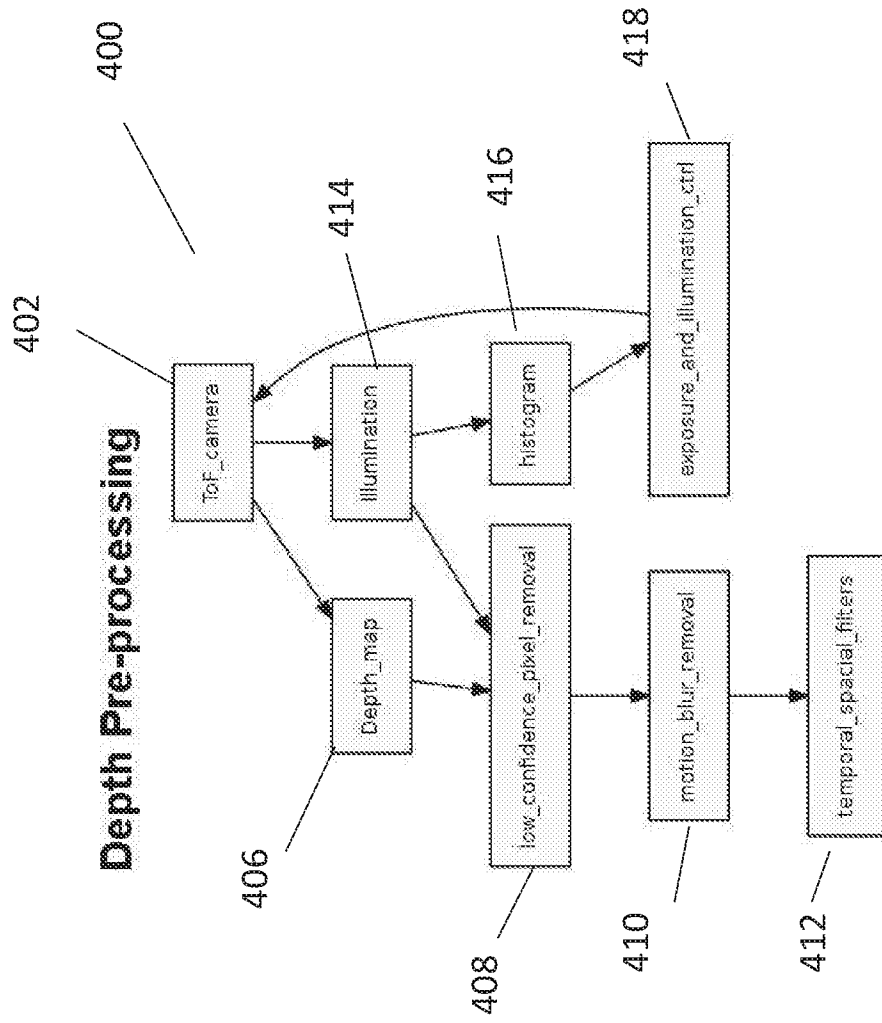
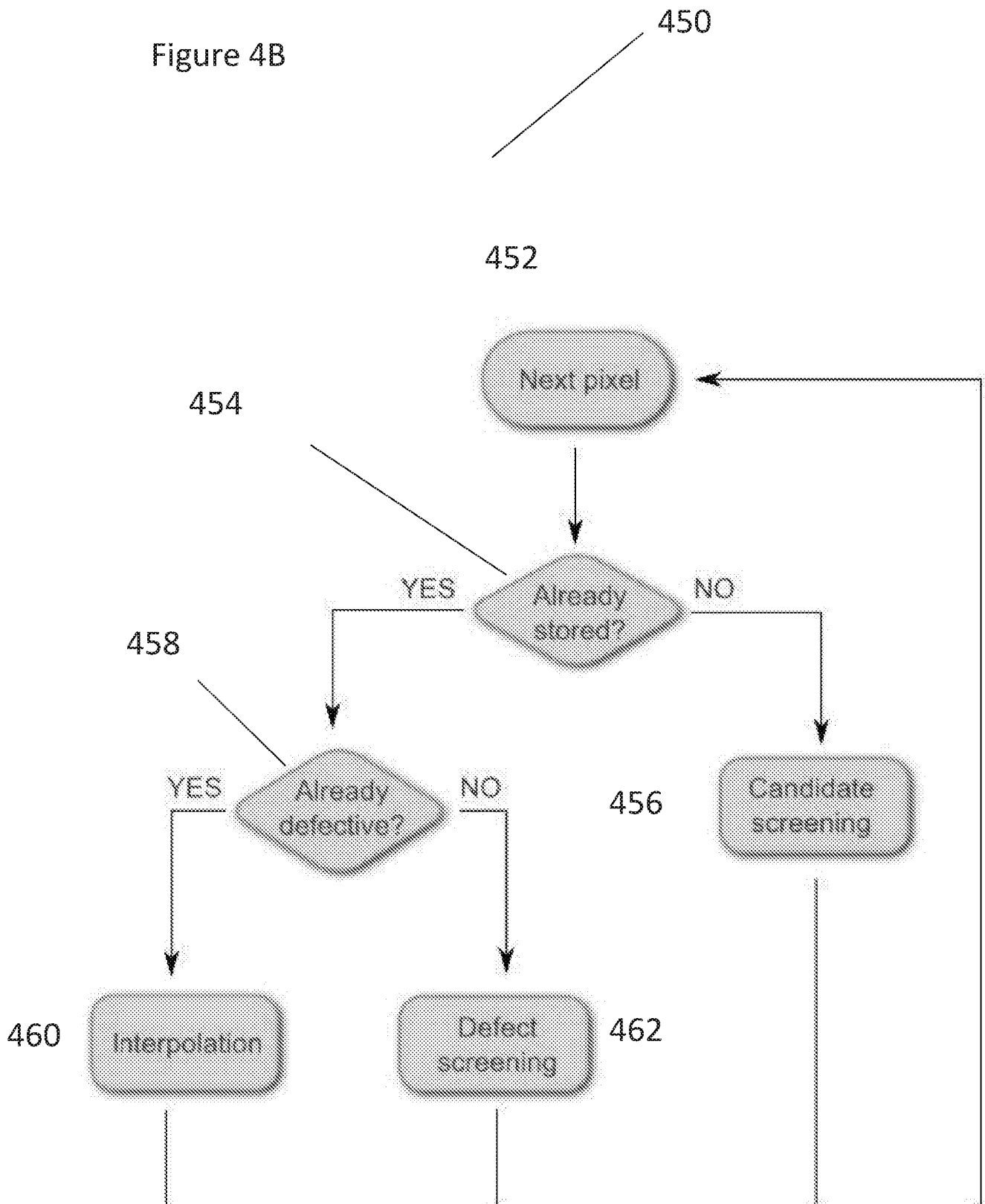


Figure 4B



State diagram to detect defective pixels.

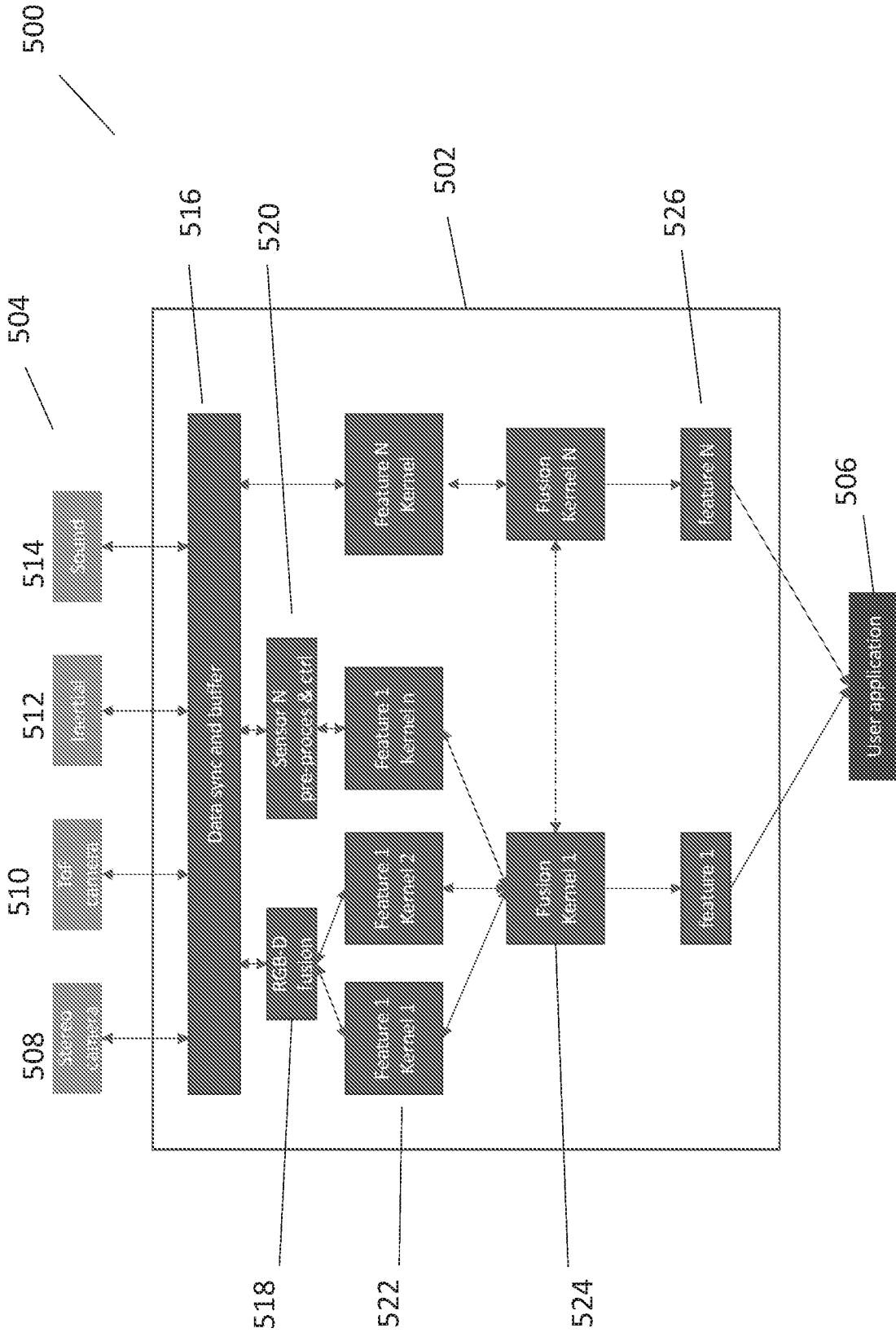


Figure 5B

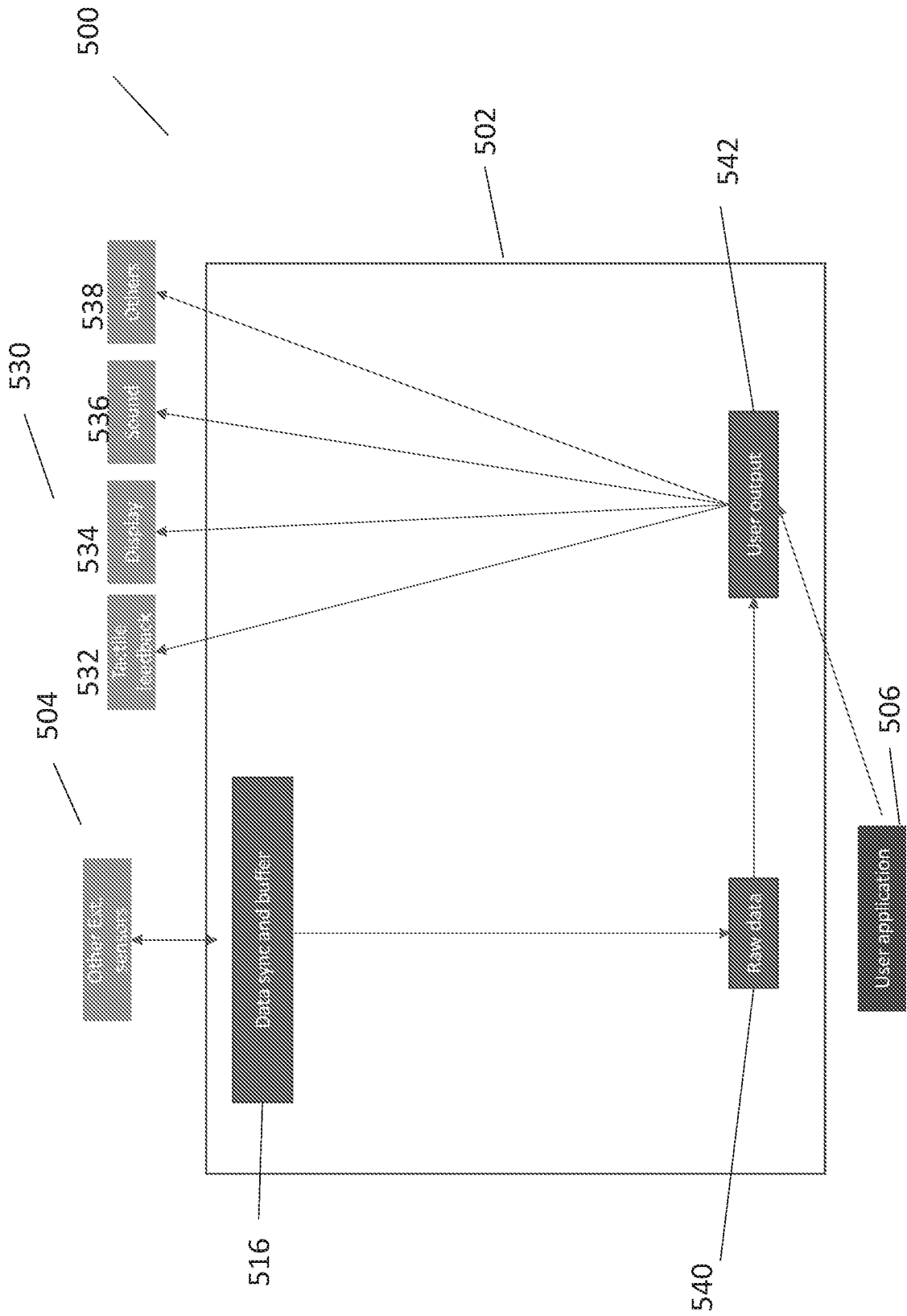


Figure 5C

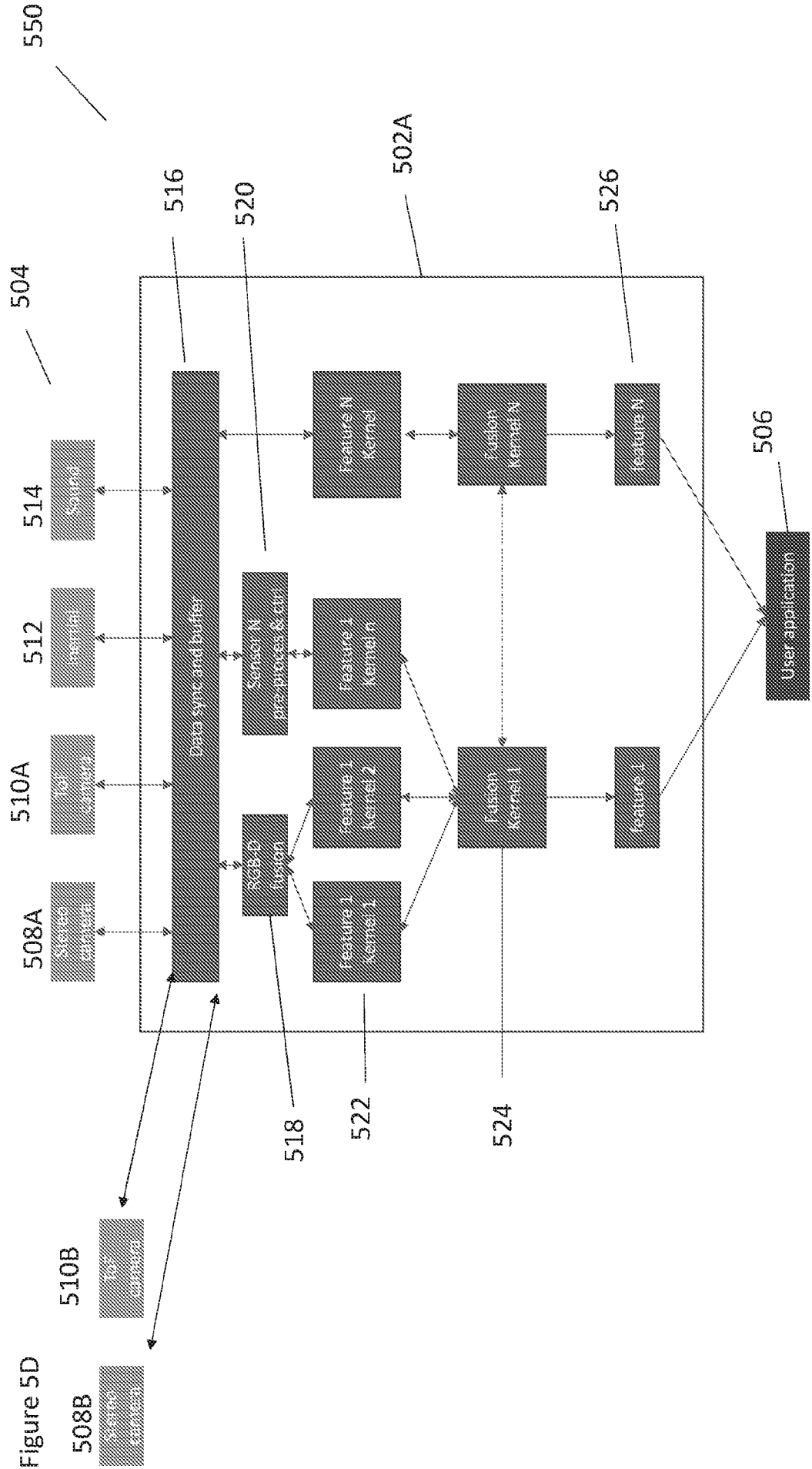


Figure 6A

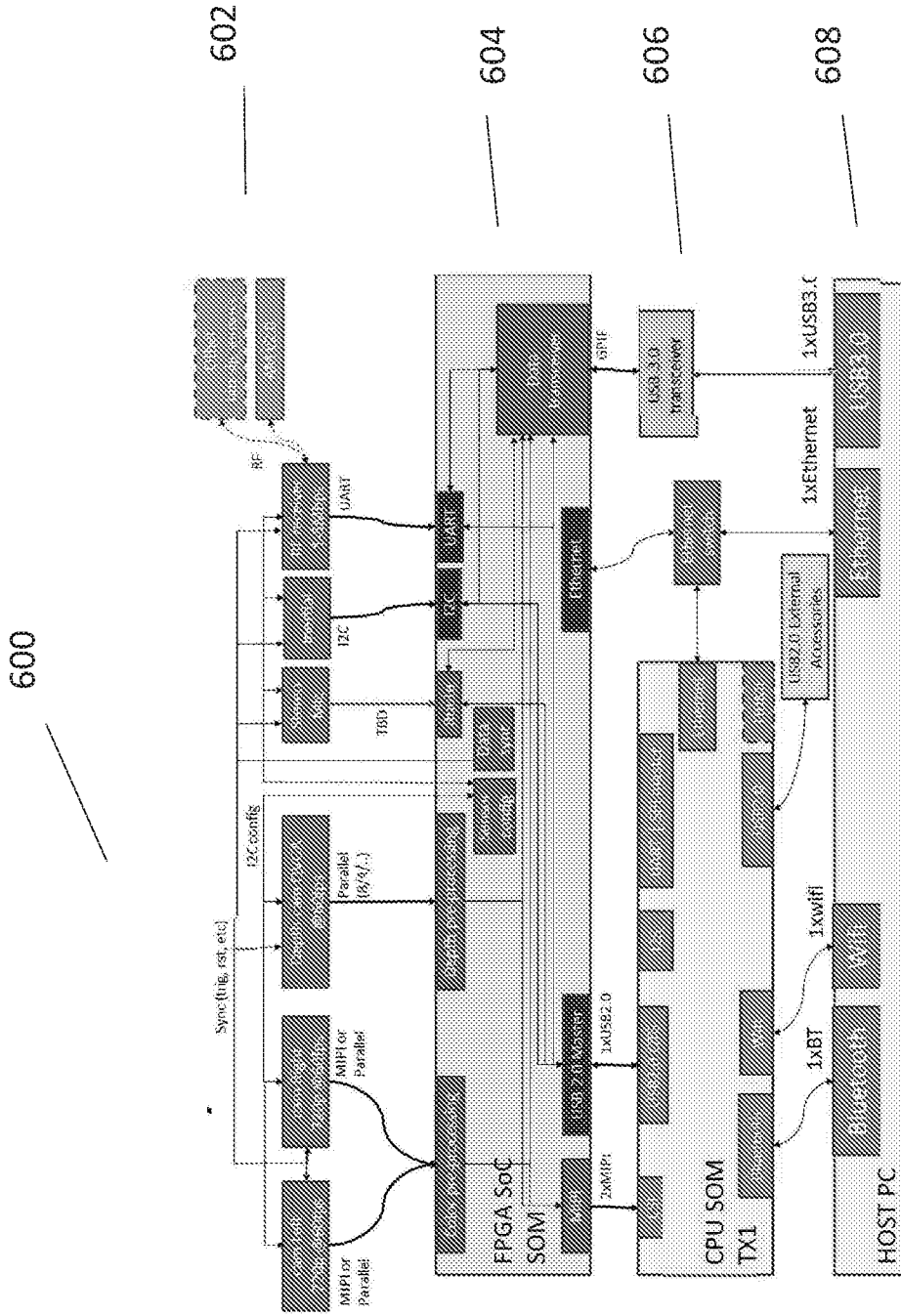
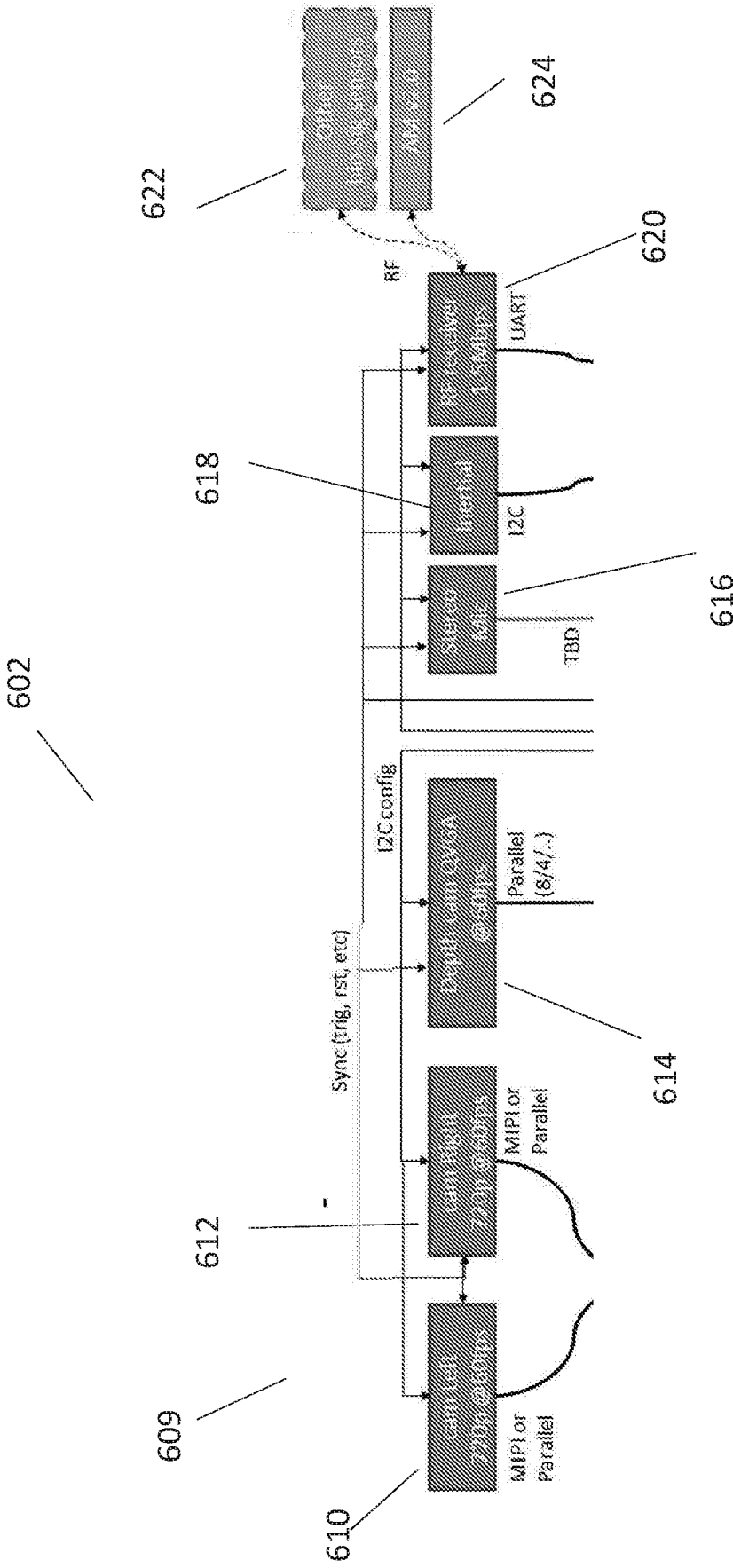


Figure 6B



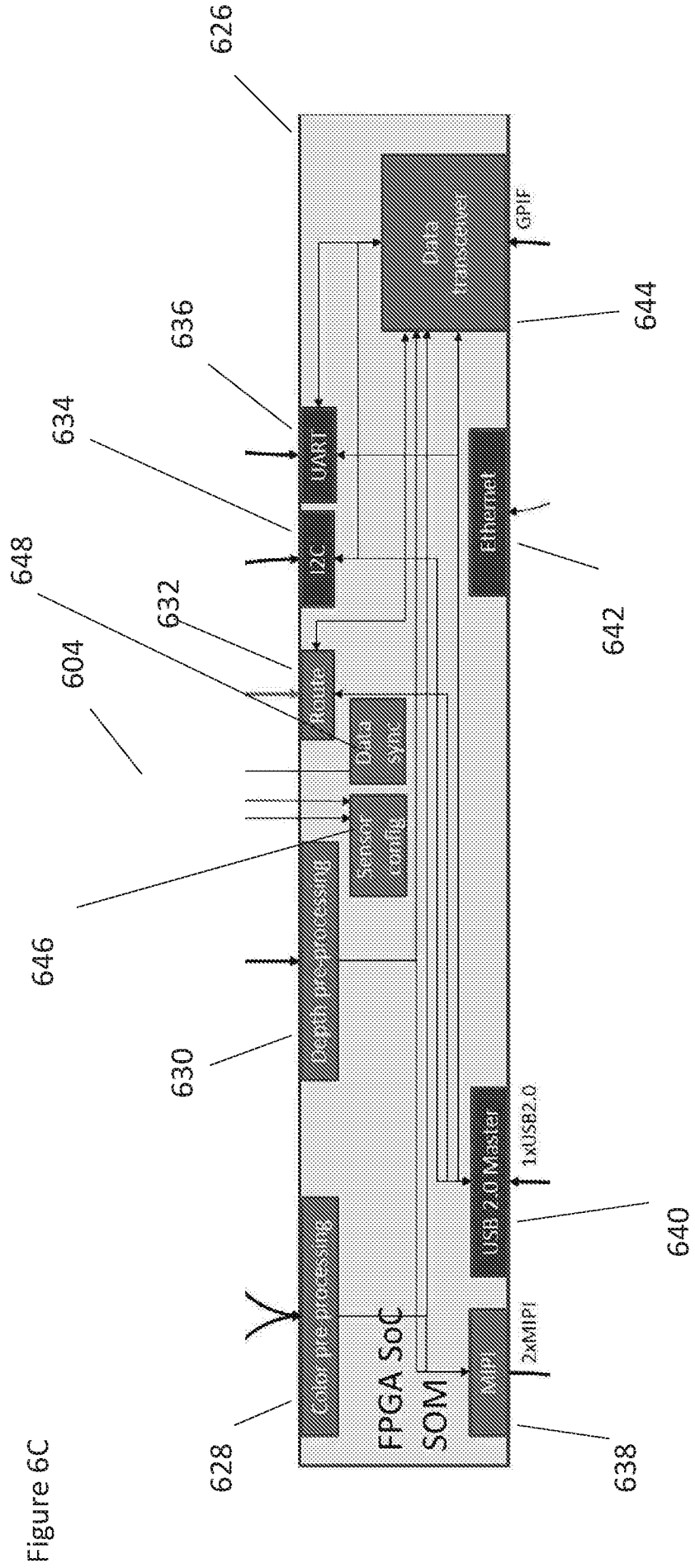


Figure 6C

Figure 6D

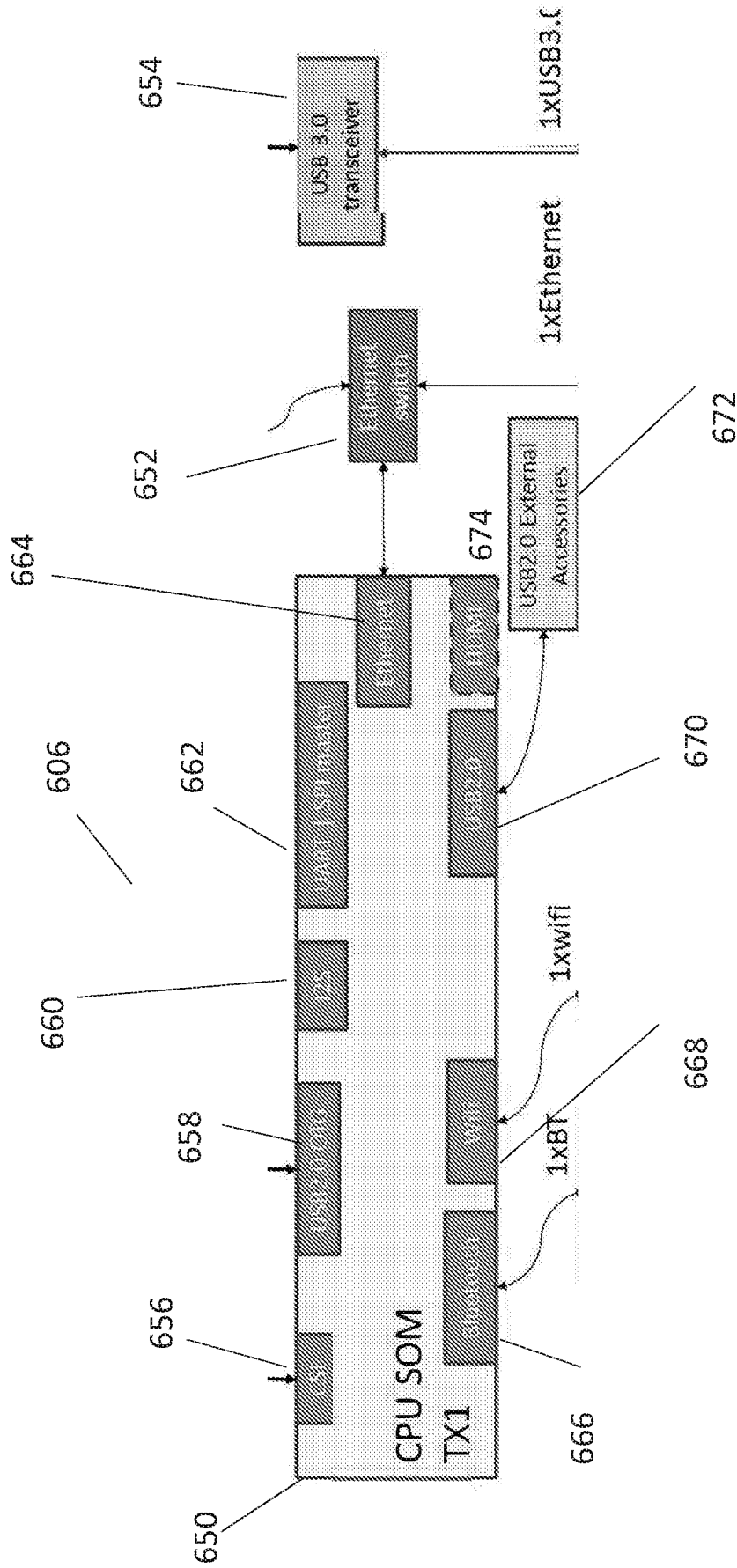
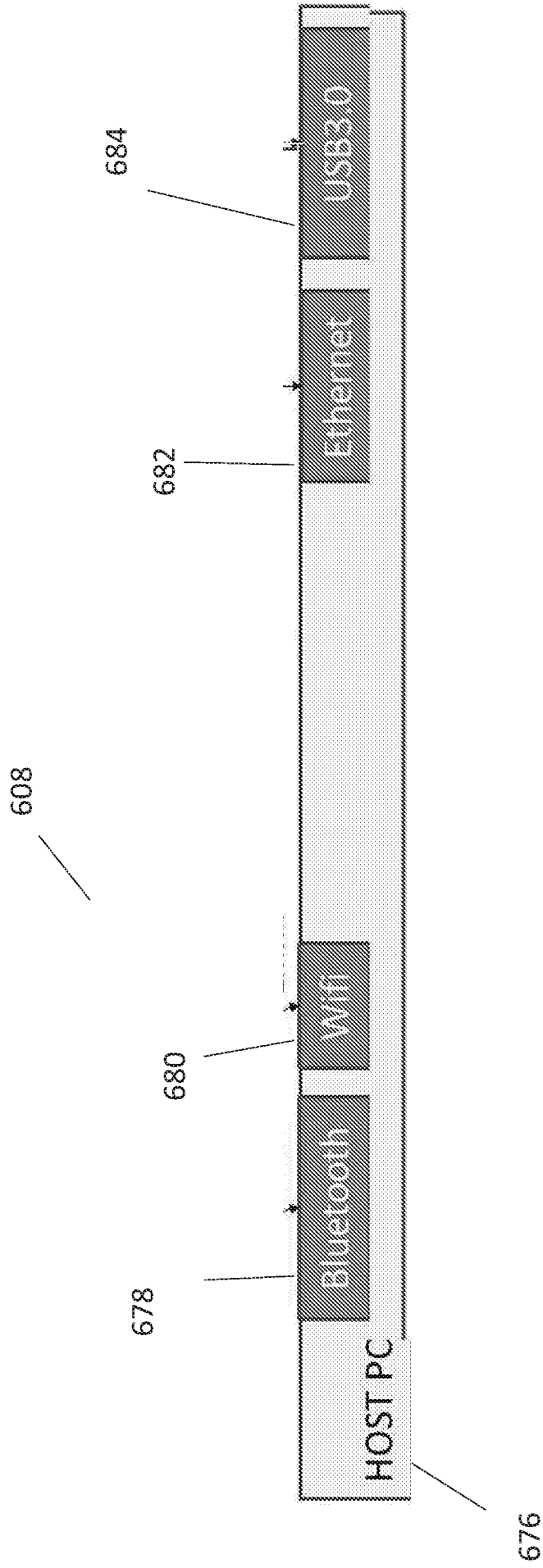
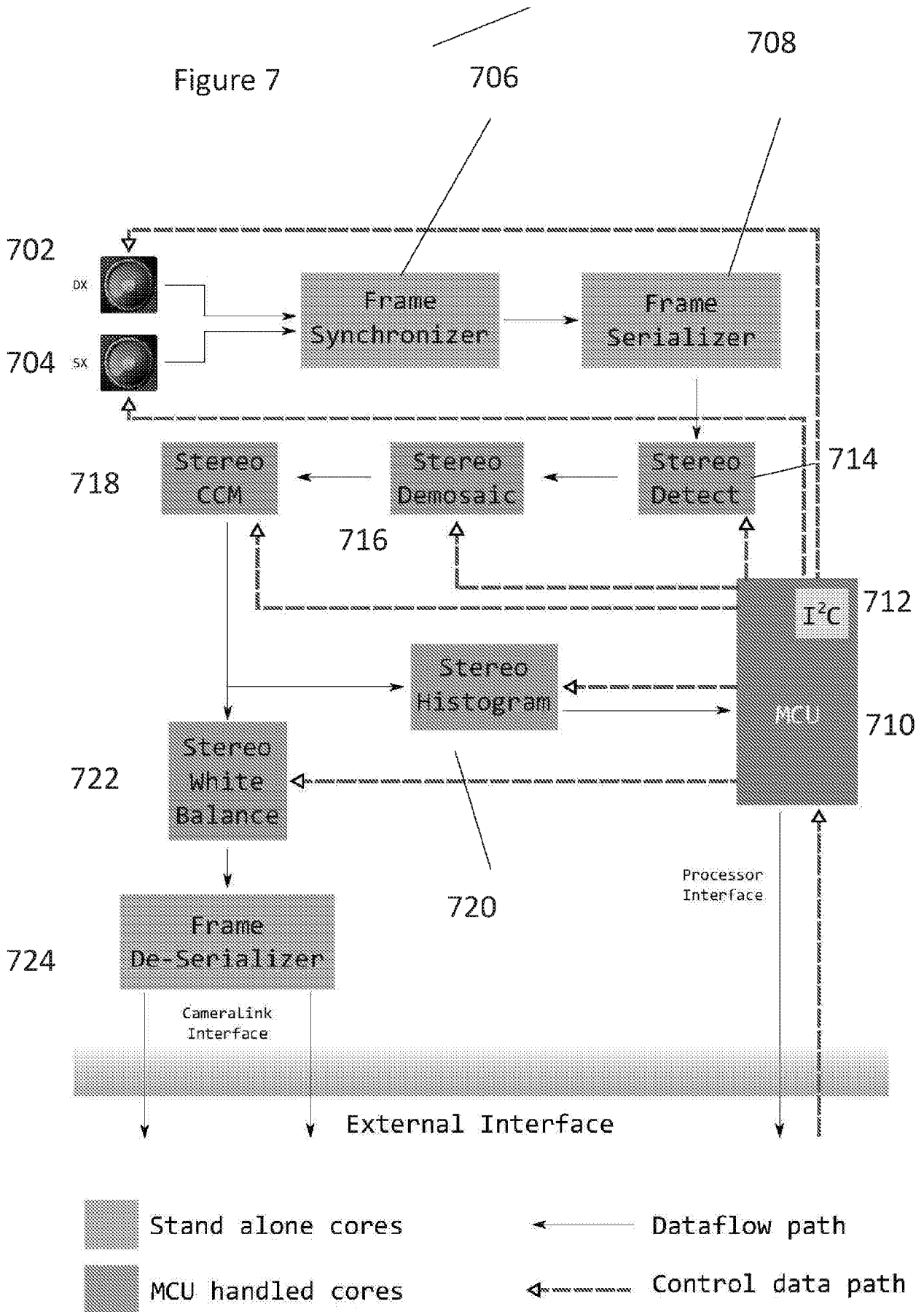


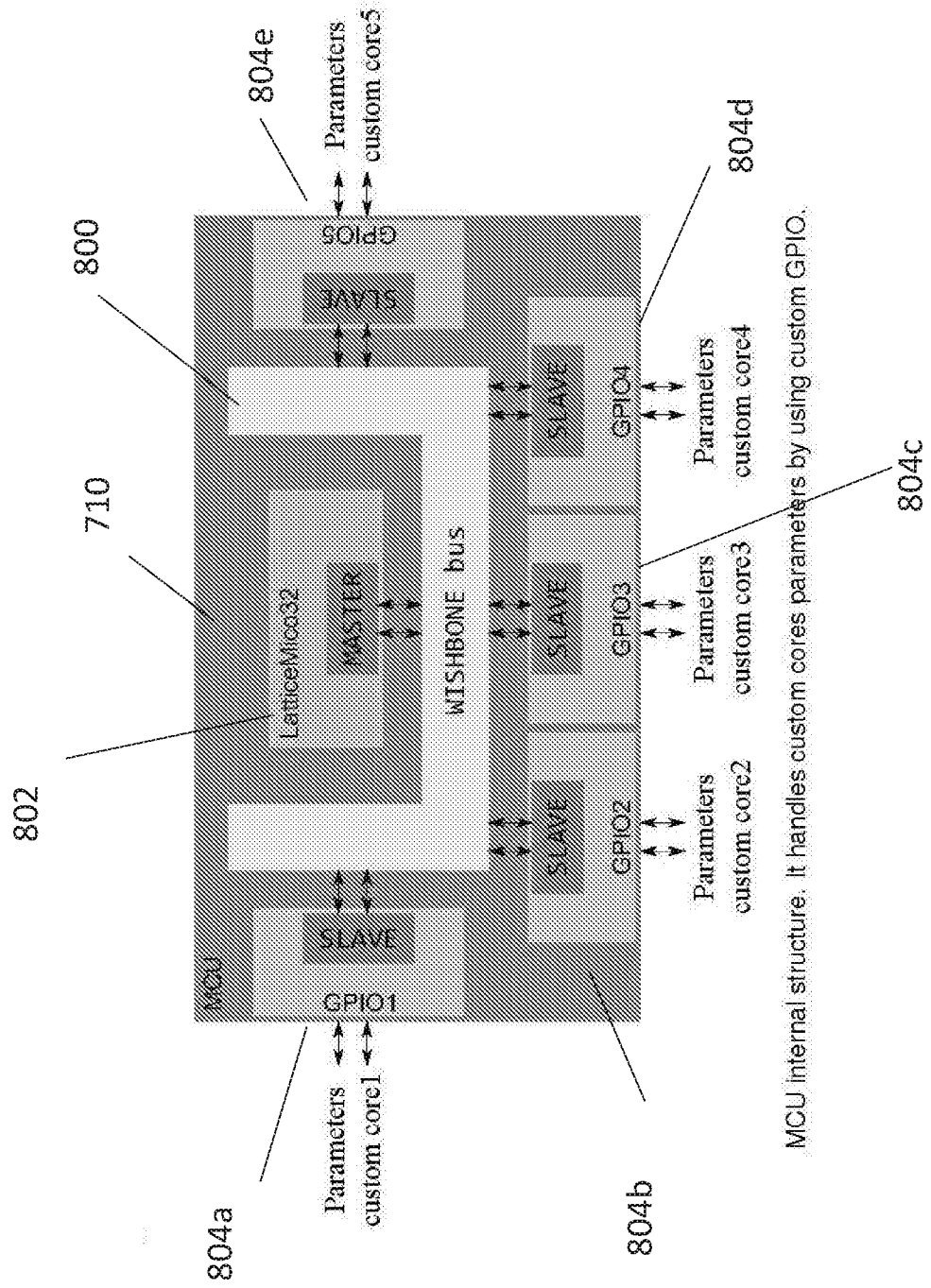
Figure 6E





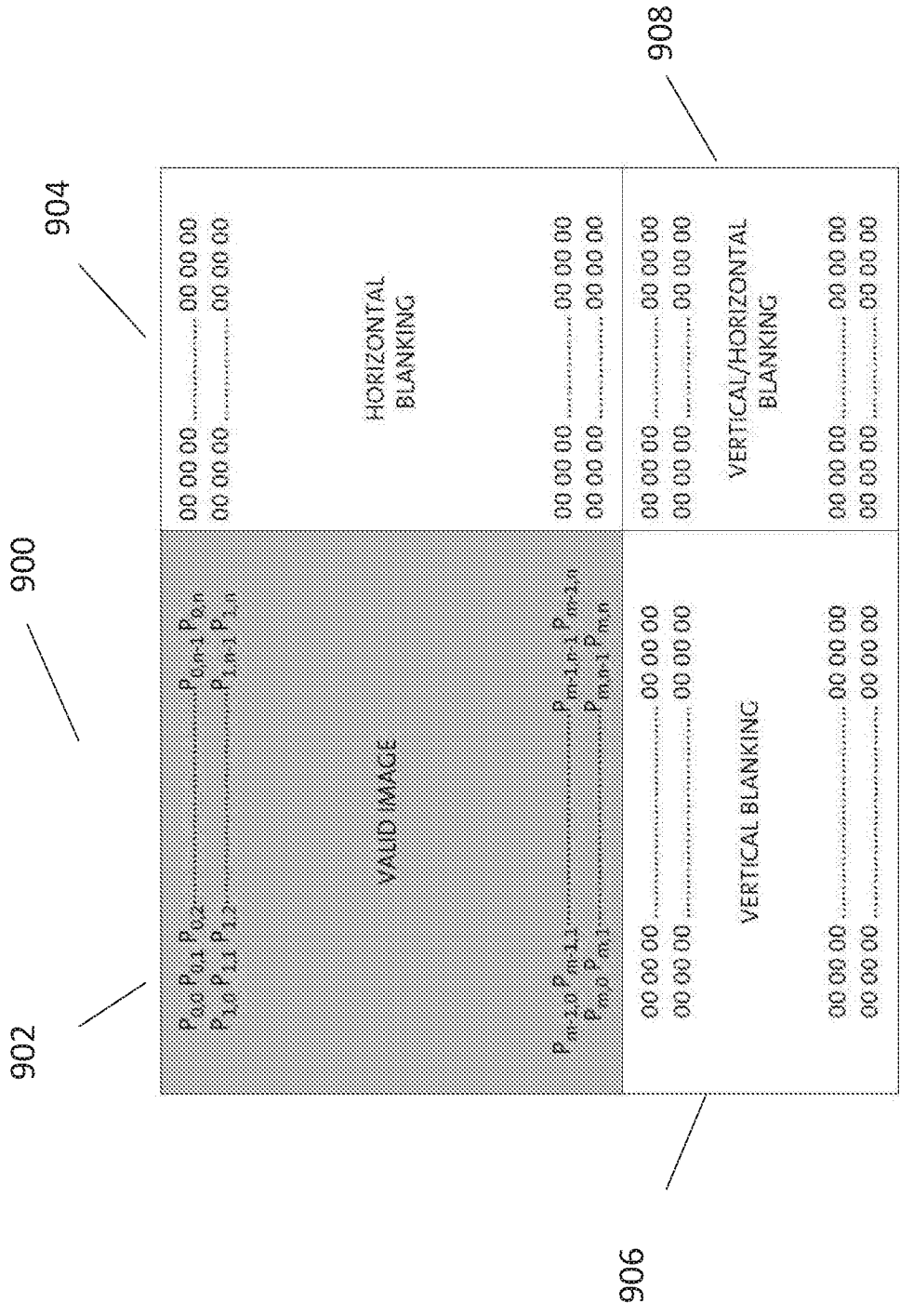
Scheme of the complete stereo video system.

Figure 8 – MCU



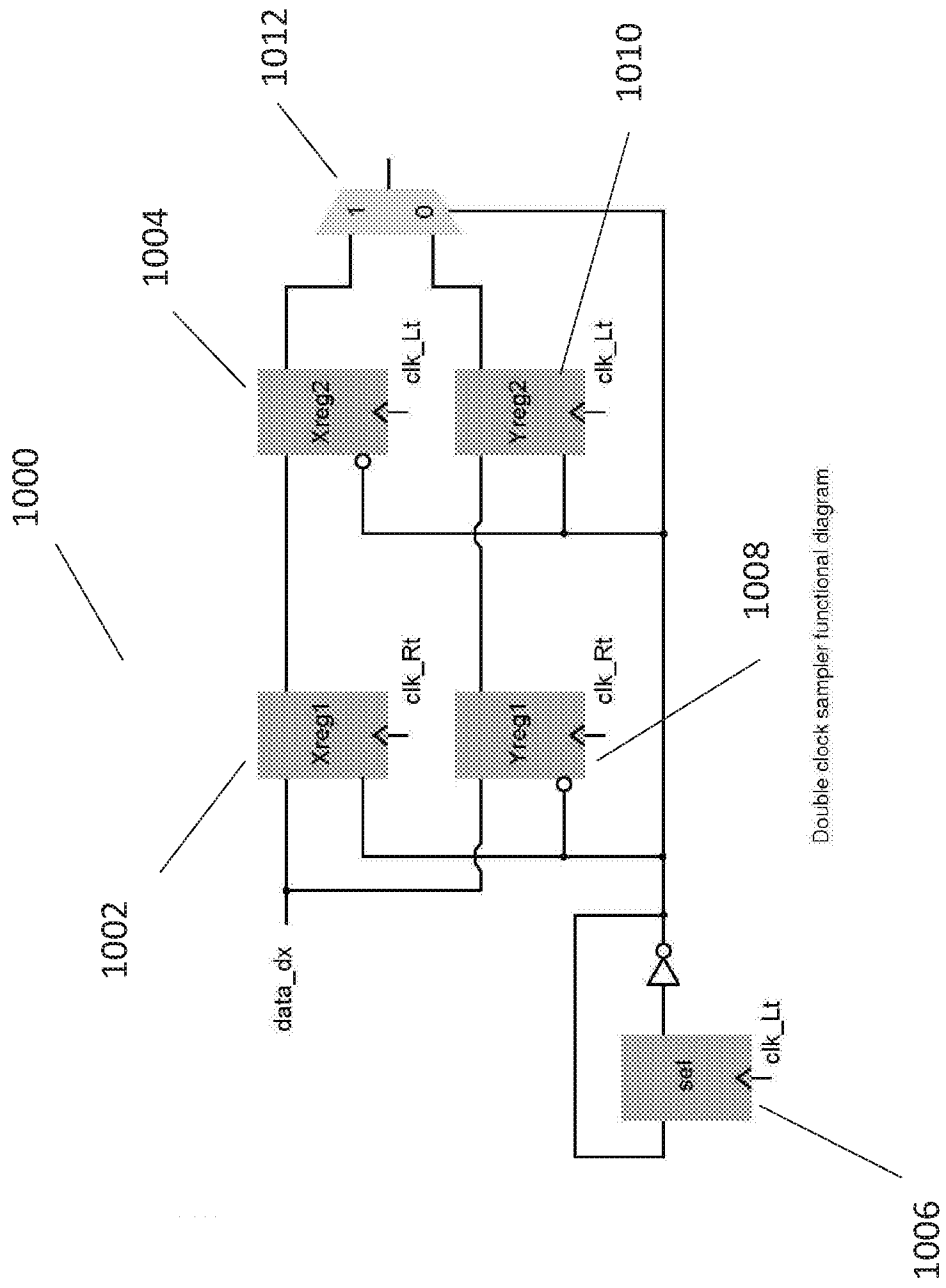
MCU internal structure. It handles custom cores parameters by using custom GPIO.

Figure 9



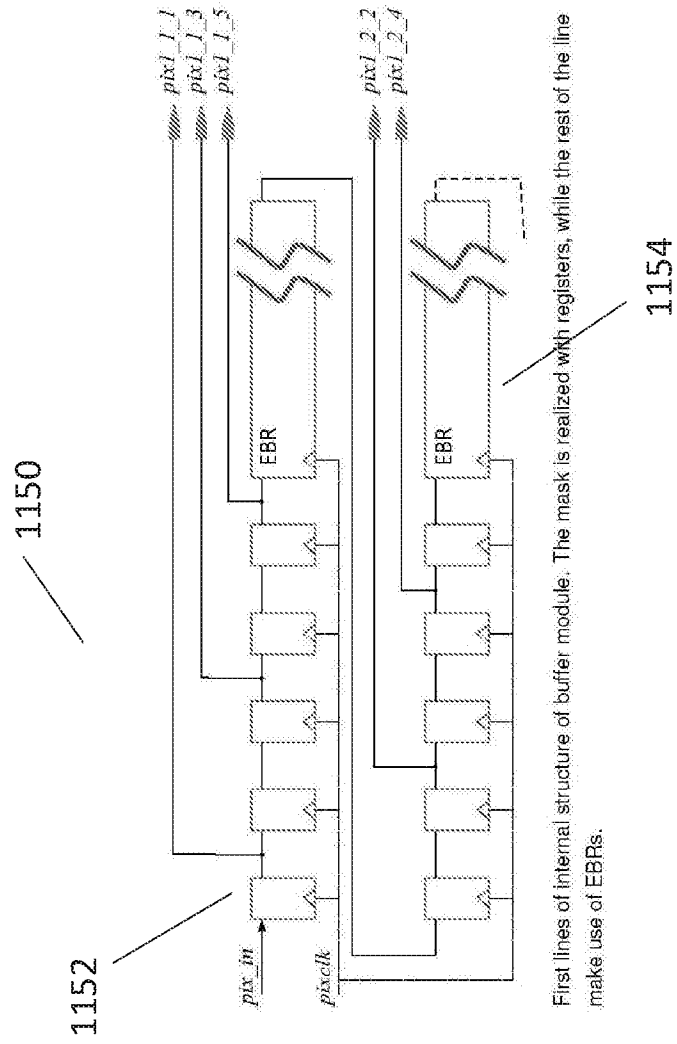
Camera readout scheme. In gray is highlighted the frame active area

Figure 10



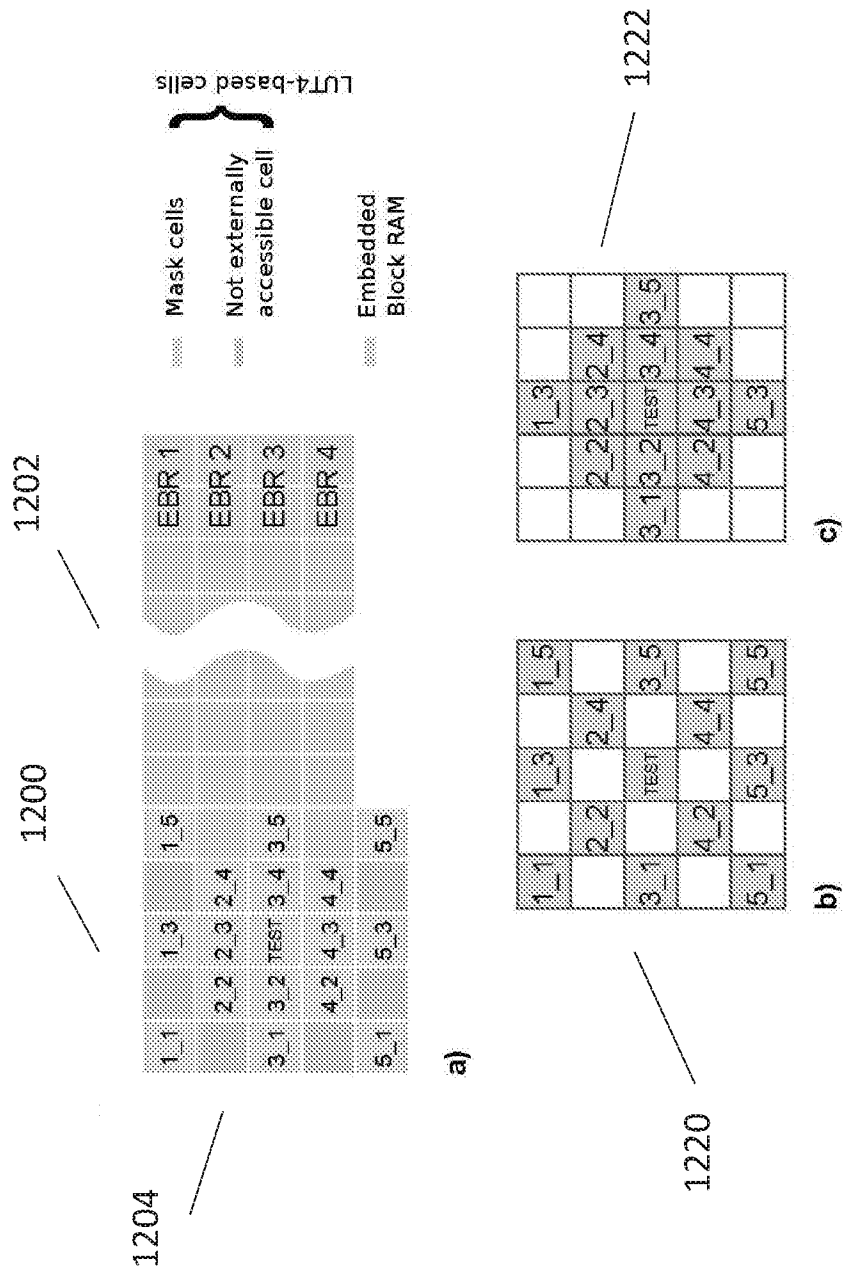
Double clock sampler functional diagram

Figure 11B



First lines of internal structure of buffer module. The mask is realized with registers, while the rest of the line make use of EBRs.

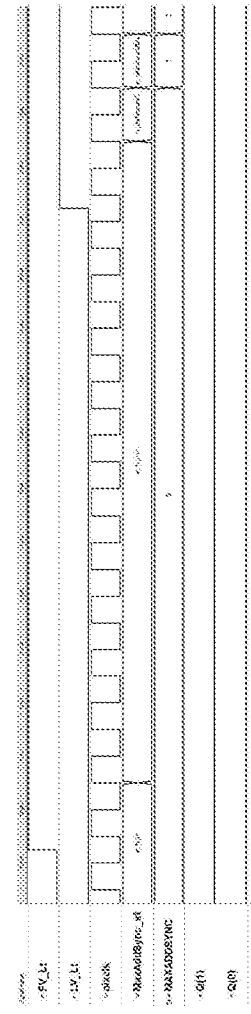
Figure 12A



Internal buffer cells arrangement: **a)** global structure, **b)** mask_1 for defective pixel detection, **c)** mask_2 for demosaic task.

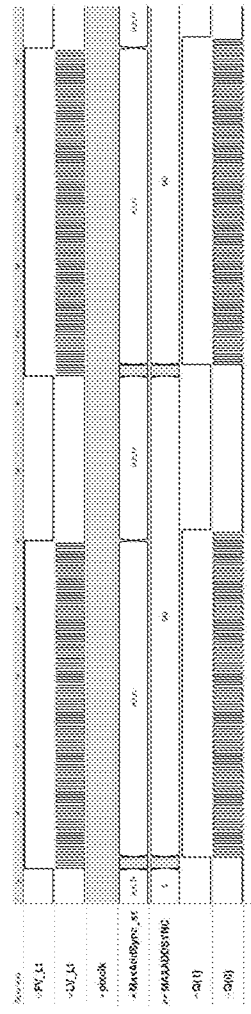
Figure 12D

1250



(a) States of state-machine handling sync signals delay.

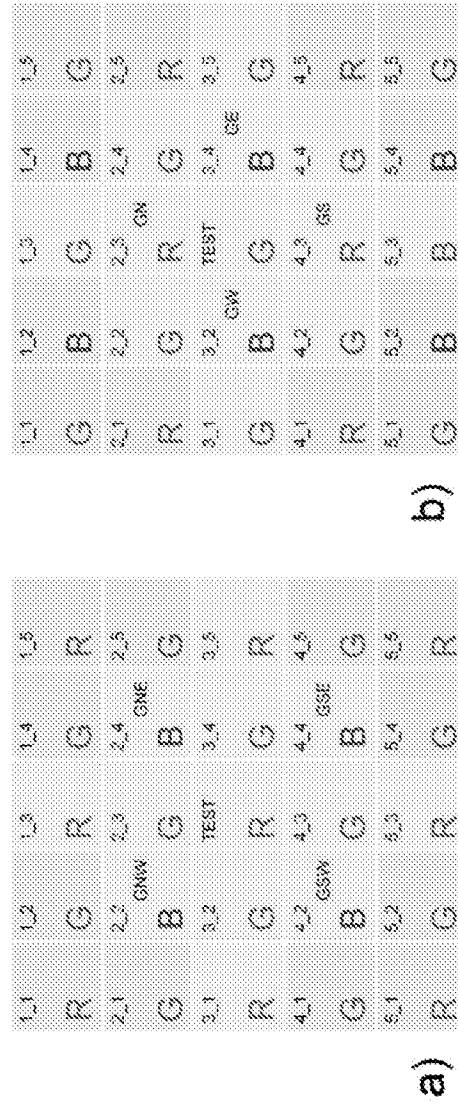
1252



(b) FV and LV signals at the output of time shift EBF.

Buffer time diagrams.

Figure 13



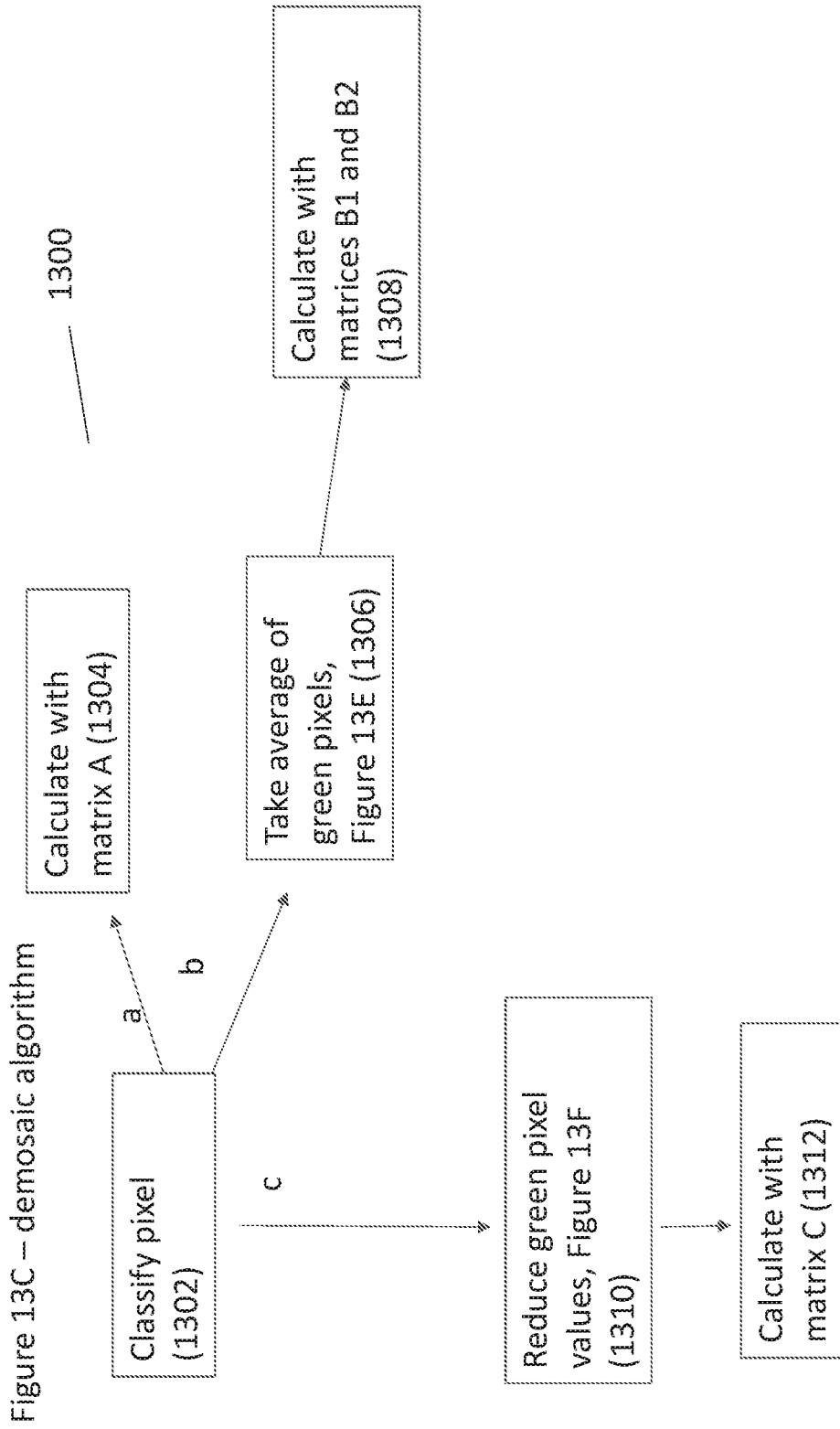


Figure 13D

$$\begin{bmatrix}
 0 & 0 & -0.125 & 0 & 0 & 0 \\
 0 & 0 & 0.25 & 0 & 0 & 0 \\
 -0.125 & 0.25 & 0.5 & 0.25 & -0.125 & 0 \\
 0 & 0 & 0.25 & 0 & 0 & 0 \\
 0 & 0 & -0.125 & 0 & 0 & 0
 \end{bmatrix}
 \begin{bmatrix}
 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & -0.125 & 0 & -0.125 & 0 & 0 \\
 -0.125 & 0.5 & 0.75 & 0.5 & -0.125 & 0 \\
 0 & -0.125 & 0 & -0.125 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0
 \end{bmatrix}$$

Matrix A

Matrix B1

Matrix B2

$$\begin{bmatrix}
 0 & 0 & 0 & 0 & 0 \\
 0 & 0.25 & -0.25 & 0.25 & 0 \\
 0 & -0.25 & 0 & -0.25 & 0 \\
 0 & 0.25 & -0.25 & 0.25 & 0 \\
 0 & 0 & 0 & 0 & 0
 \end{bmatrix}$$

Matrix C

Figure 13E

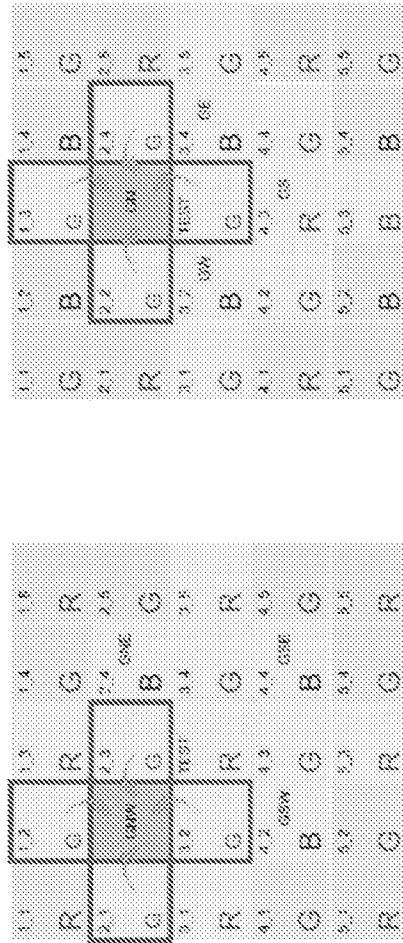


Figure 13F

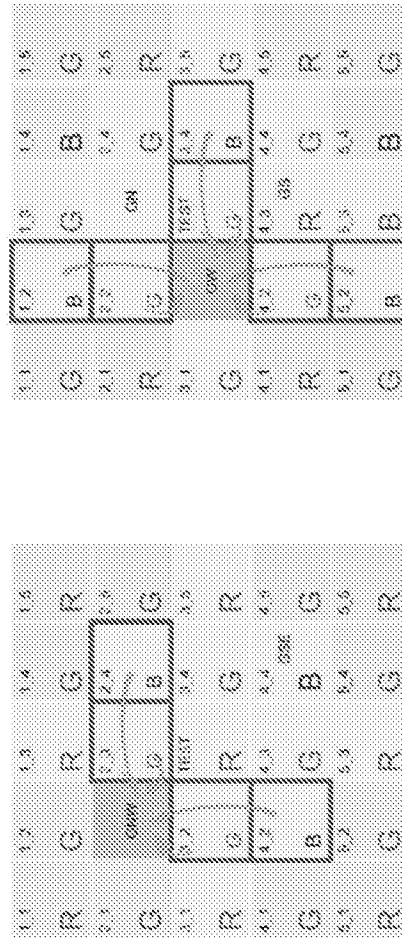


Figure 13G

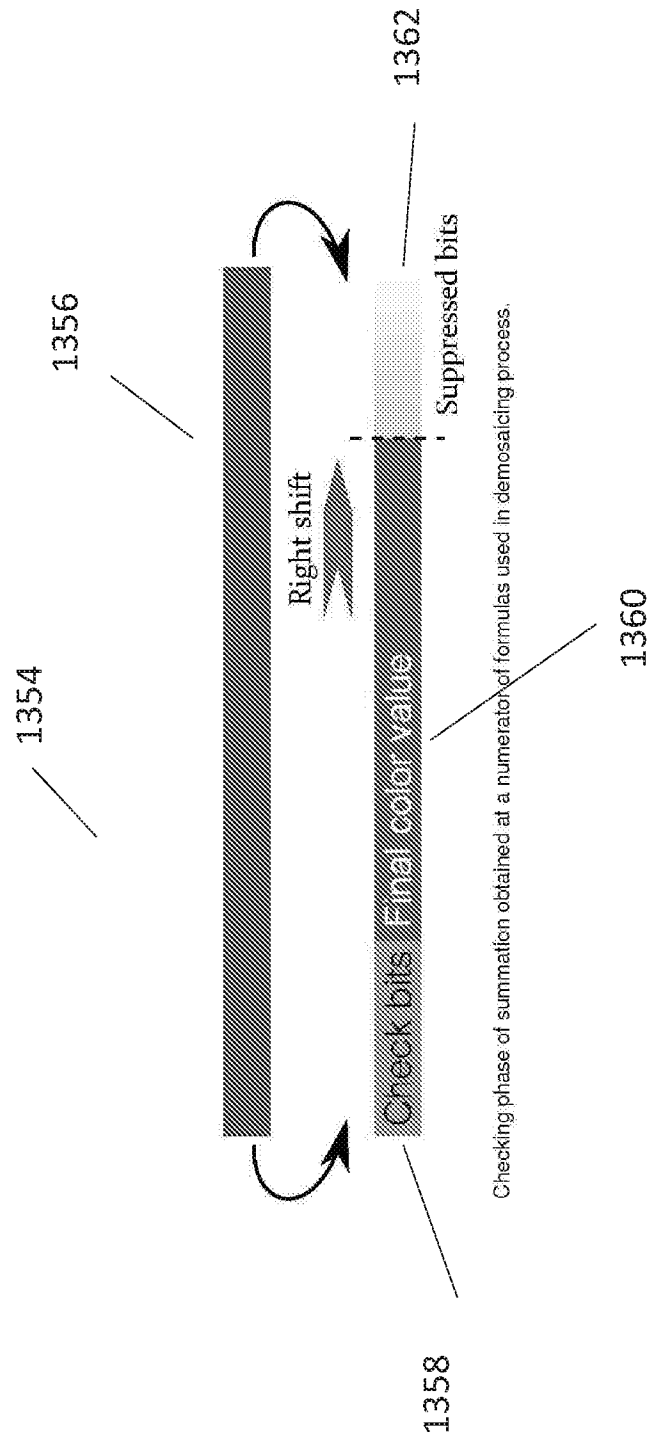


Figure 13H

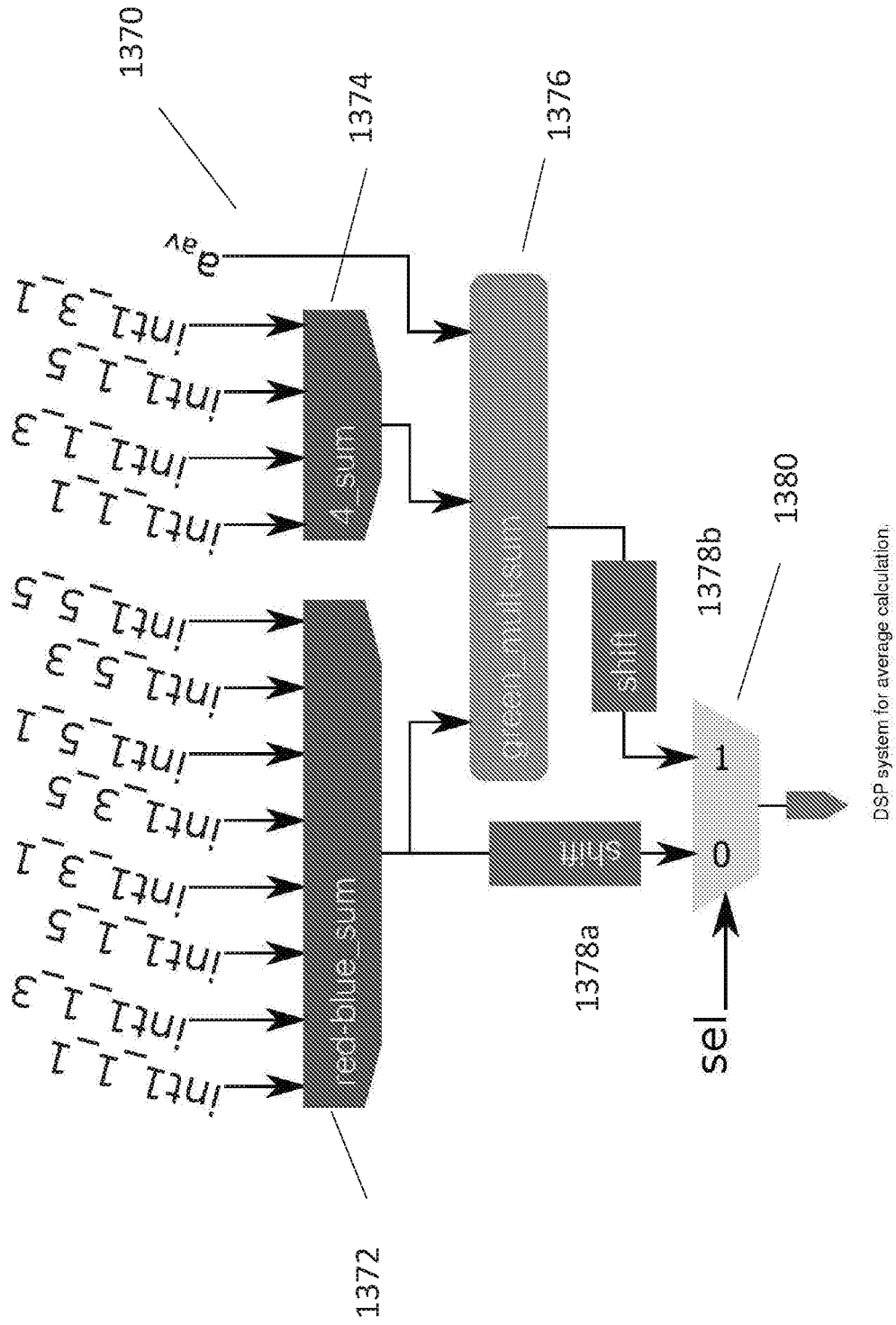


Figure 15 - histogram

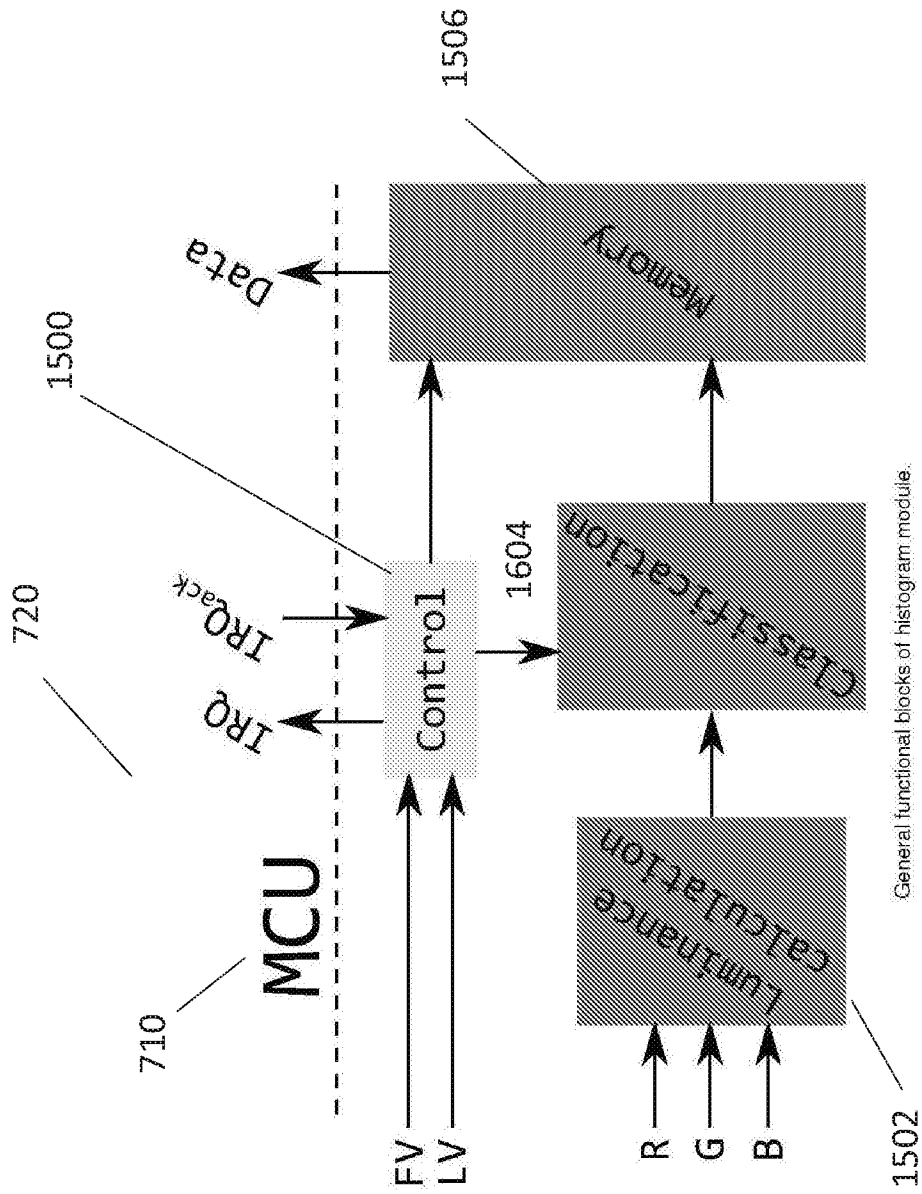
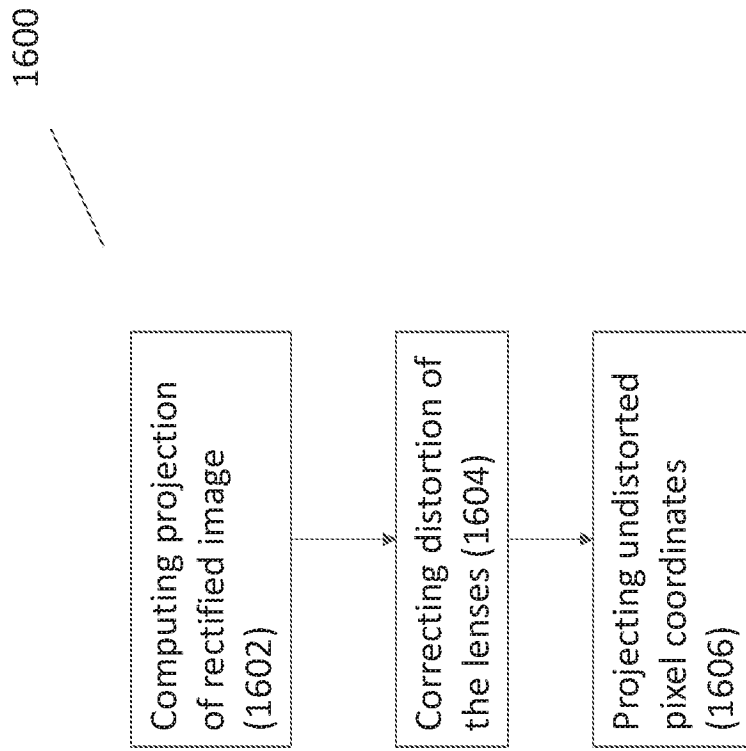


Figure 16



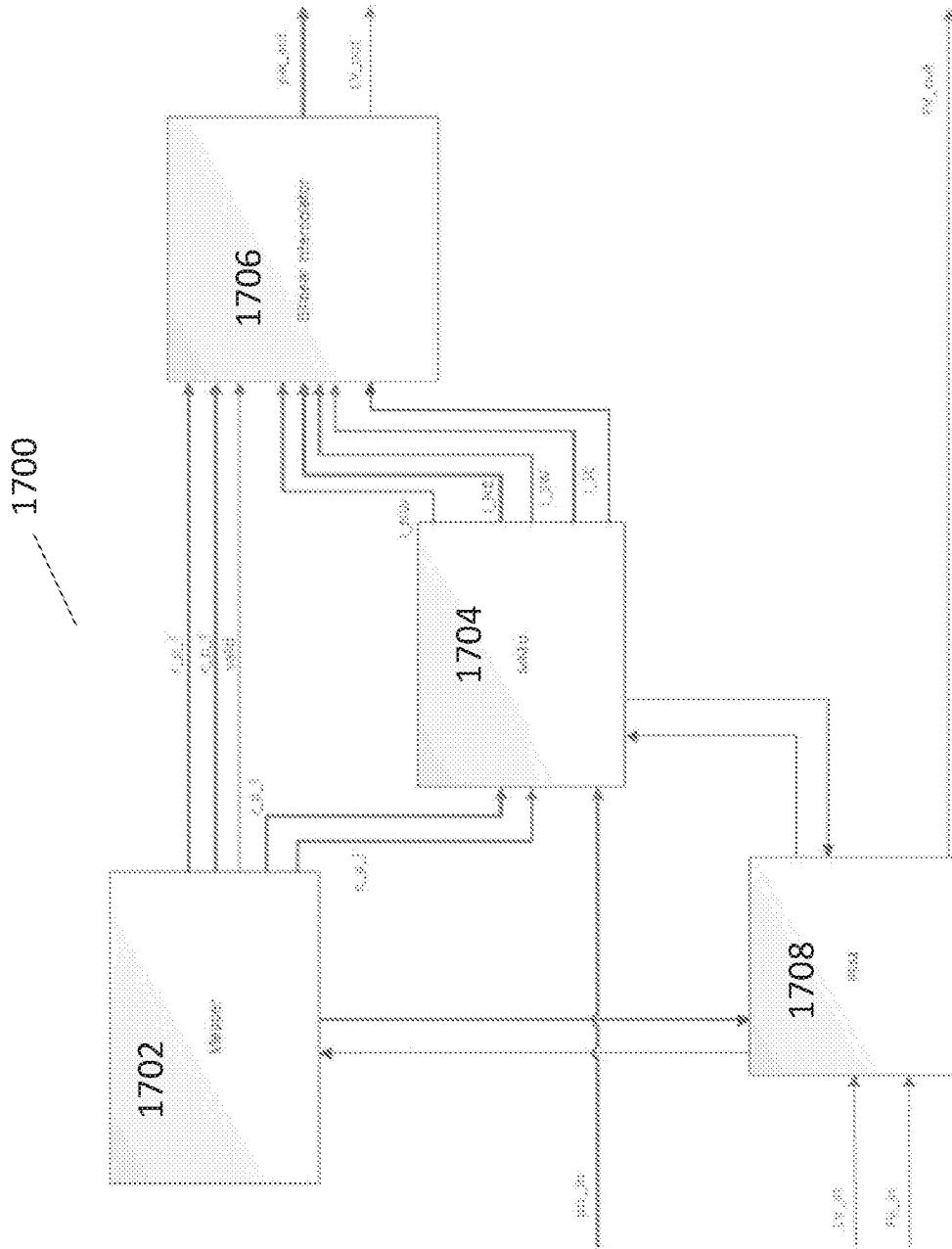


Figure 17A

1702

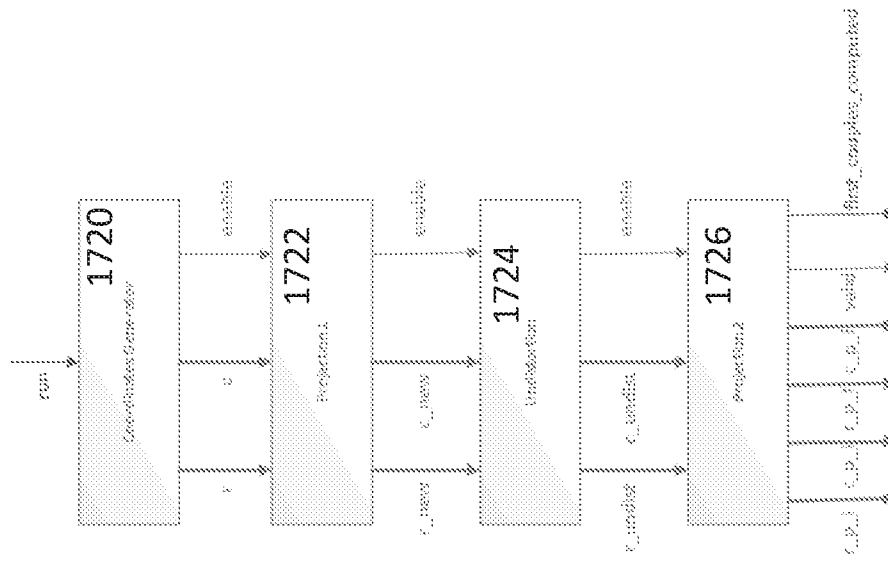


Figure 17B

Figure 17D

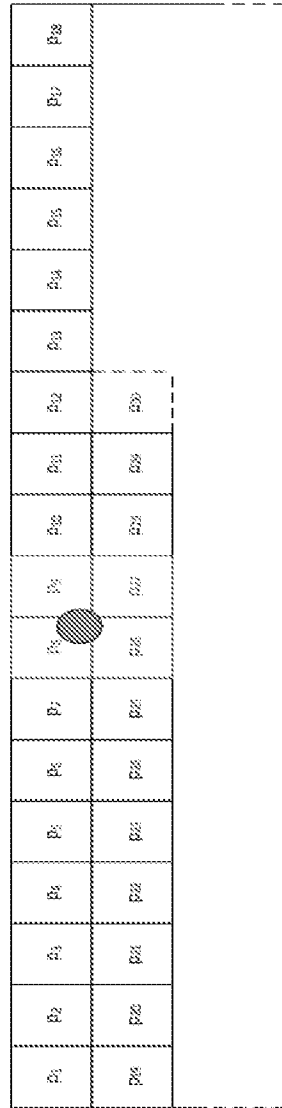


Figure 17E

00	00	00	00
p1	p2	p3	p4
p5	p6	p7	p8
p9	p10	p11	p12
p13	p14	p15	p16
p17	p18	p19	p20
p21	p22	p23	p24
p25	p26	p27	p28
p29	p30		

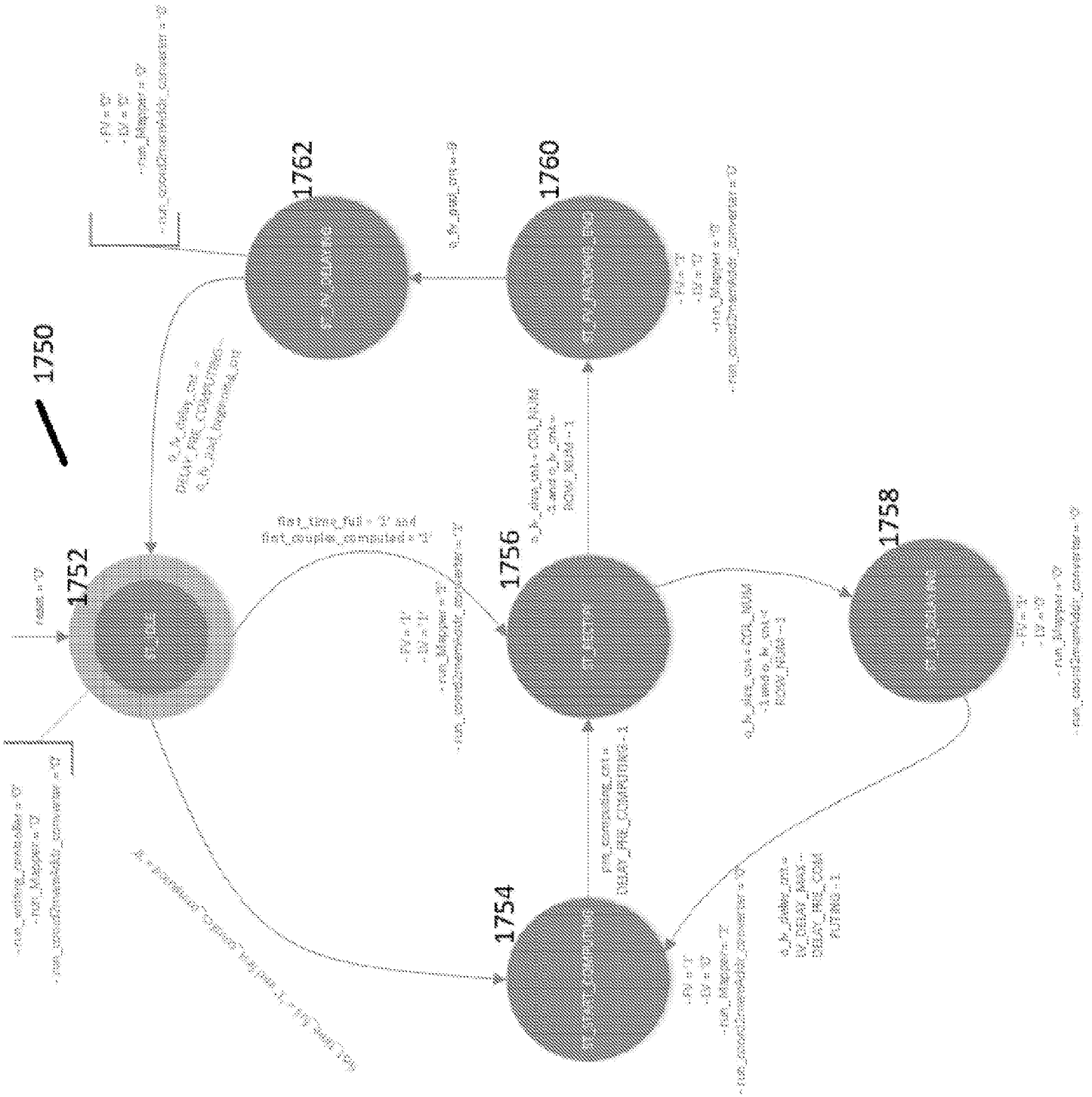


Figure 17F

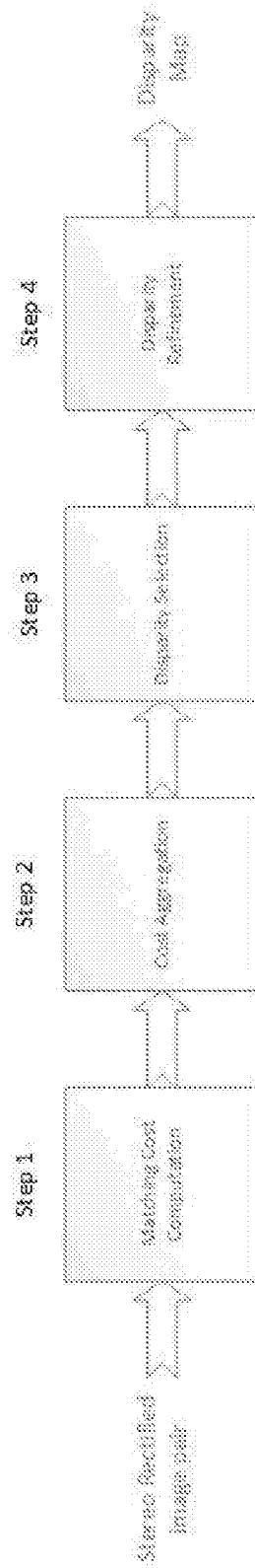
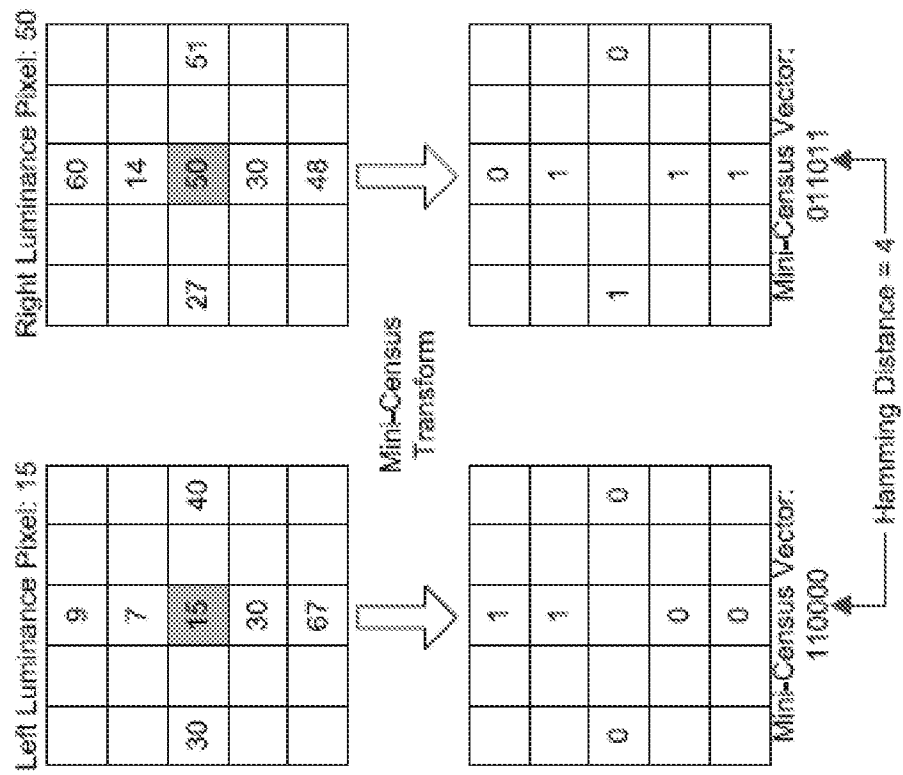


Figure 18A

Figure 18B



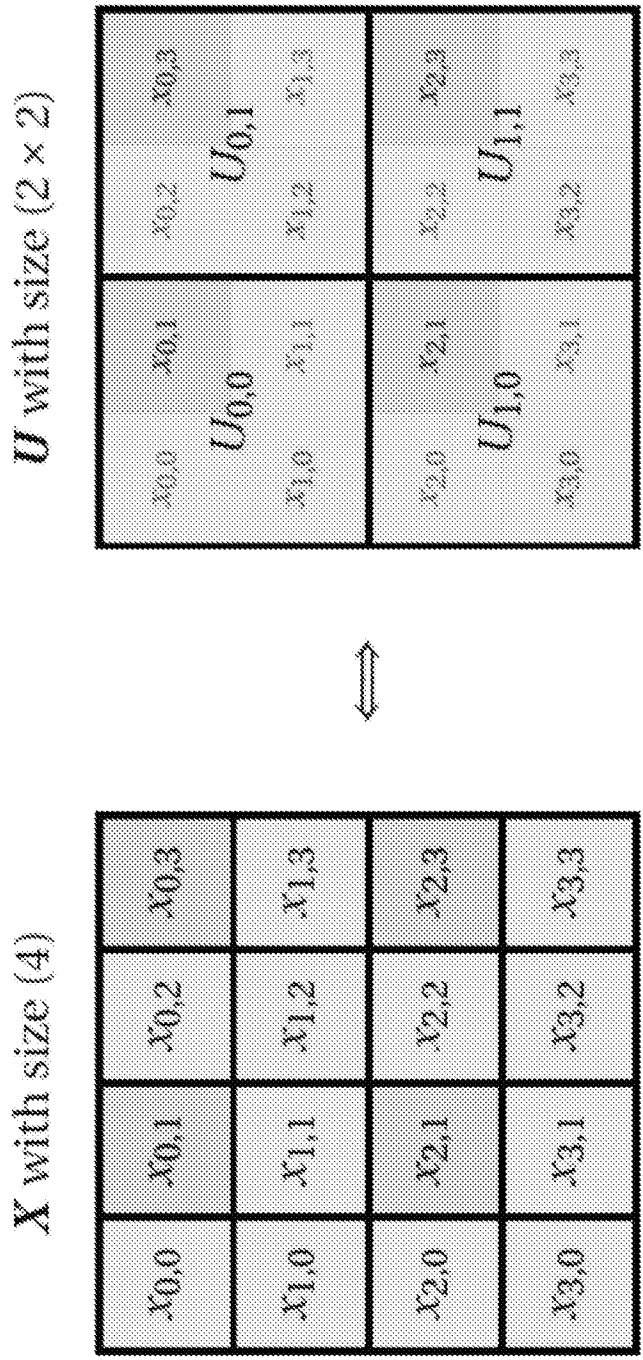


Figure 19A

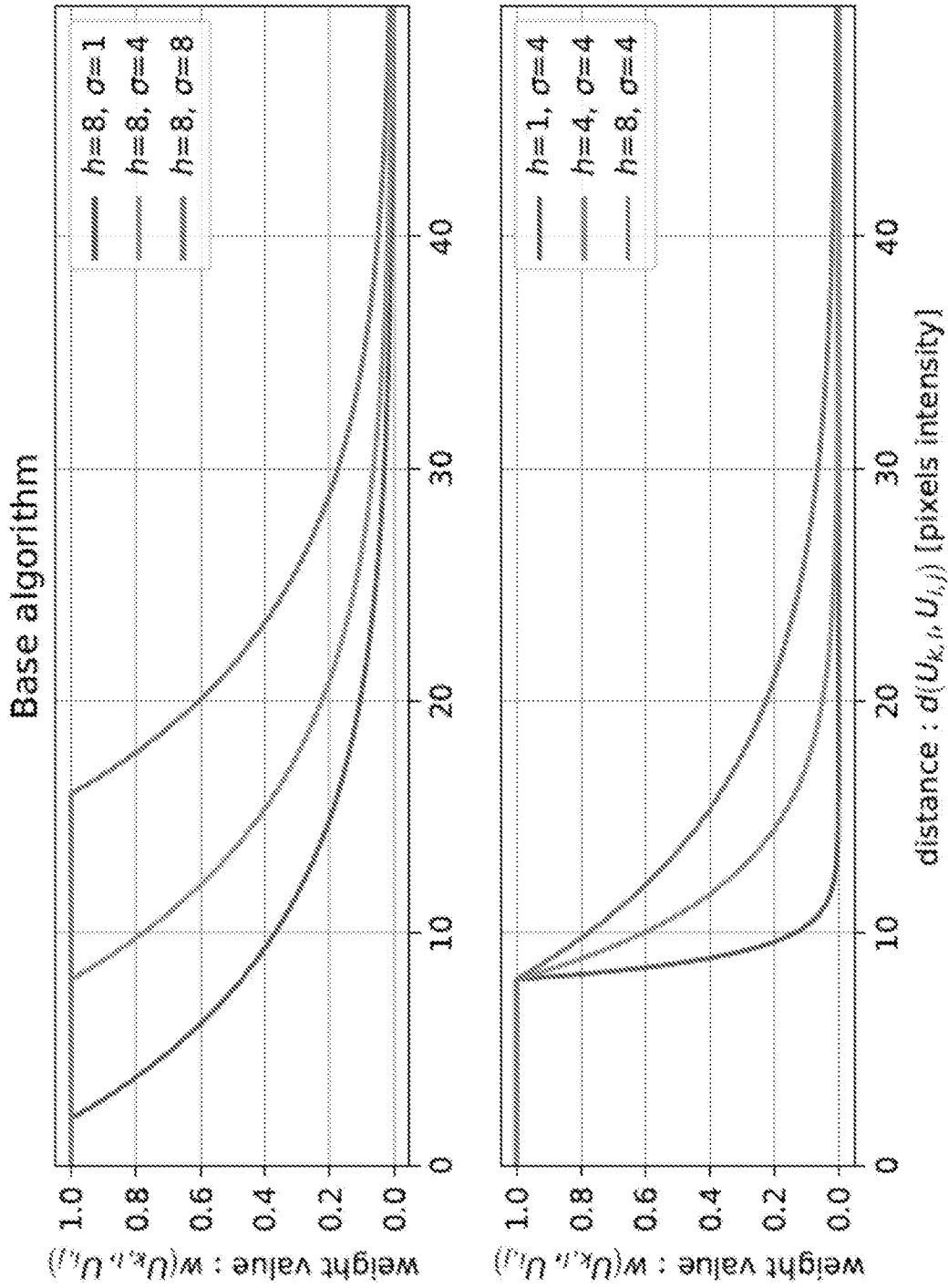


Figure 19B

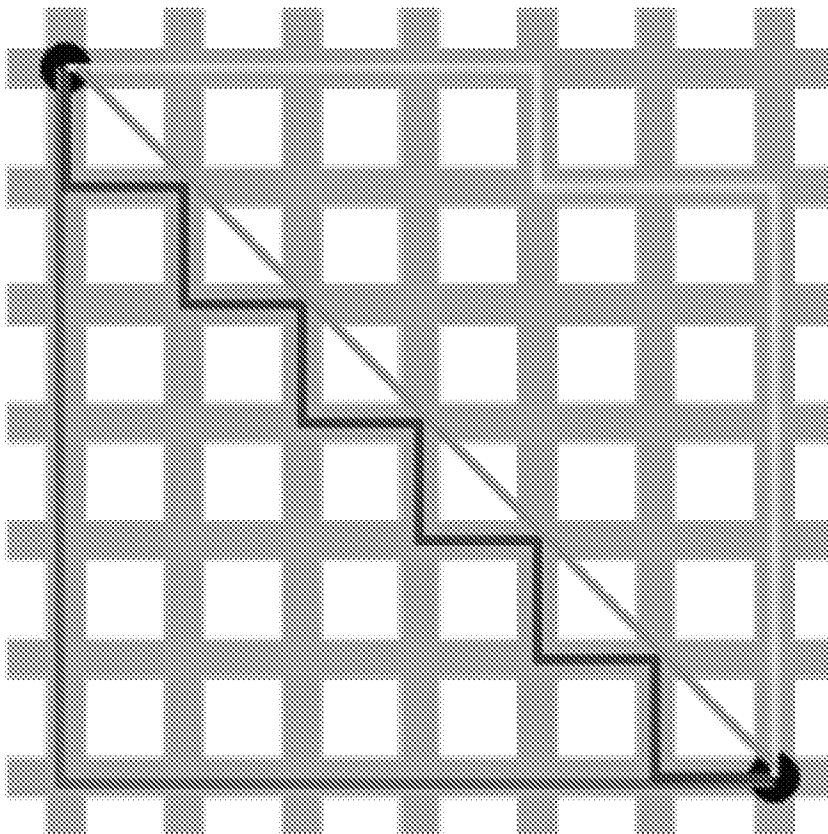


Figure 19C

Figure 19D

Algorithm 1 "W-means" algorithmic description

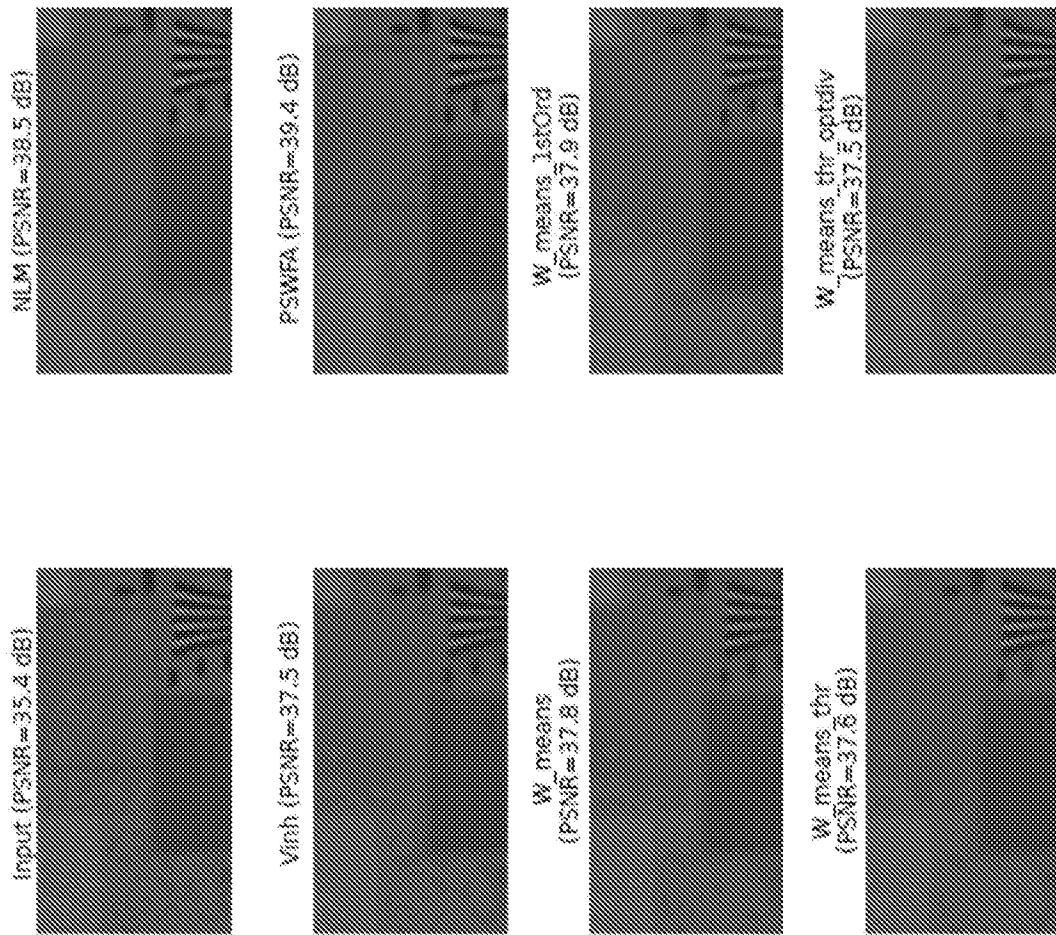
```

1: function WMEANS_UNIT(inputImage)
2:   for  $i = [3, 5, \dots, \text{height}(\text{inputImage}) - 3]$  do
3:     for  $j = [3, 5, \dots, \text{width}(\text{inputImage}) - 3]$  do
4:        $U_{i,j} \leftarrow \text{inputImage}[i : i + 2, j : j + 2]$ 
5:        $V_{i,j} \leftarrow 0$ 
6:        $C_{i,j} \leftarrow 0$ 
7:       for  $k = i + [-2 * f, -2 * f + 2, \dots, 2 * f]$  do
8:         for  $l = j + [-2 * f, -2 * f + 2, \dots, 2 * f]$  do
9:            $U_{k,l} \leftarrow \text{inputImage}[k : k + 2, l : l + 2]$ 
10:           $d \leftarrow \text{sqrt}(\text{sum}((U_{i,j} - U_{k,l})^2) / 4)$ 
11:           $w \leftarrow \exp(-(\text{max}([d - 2\sigma, 0])) / h)$ 
12:           $V_{i,j} \leftarrow V_{i,j} + U_{k,l} * w$ 
13:           $C_{i,j} \leftarrow C_{i,j} + w$ 
14:        end for
15:      end for
16:       $\text{outputImage}[i : i + 2, j : j + 2] \leftarrow V_{i,j} / C_{i,j}$ 
17:    end for
18:  end for
19:  return outputImage
20: end function

```

▷ For all 4-color pixels
 ▷ excluding borders
 ▷ Extract the 4-color center pixel
 ▷ Init output pixel
 ▷ Init sum of weights
 ▷ For all 4-color neighbors
 ▷ Extract the 4-color neighbor
 ▷ Compute the color Euclidean distance
 ▷ Compute weight
 ▷ Weighted accumulation for output pixel
 ▷ Accumulate weights
 ▷ Output the 4-color pixel

Figure 20



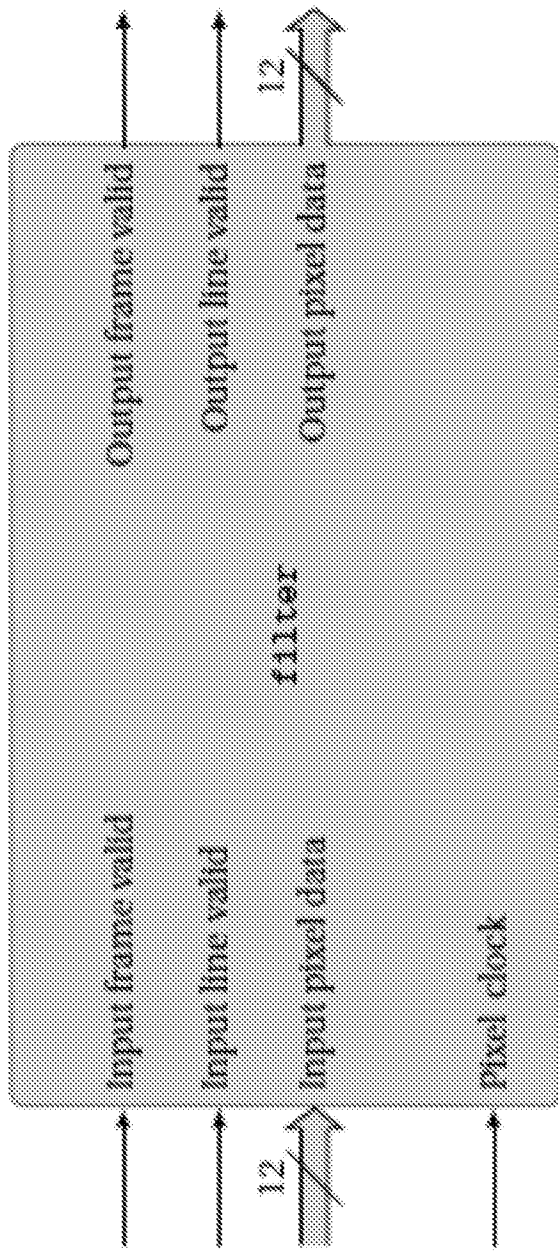
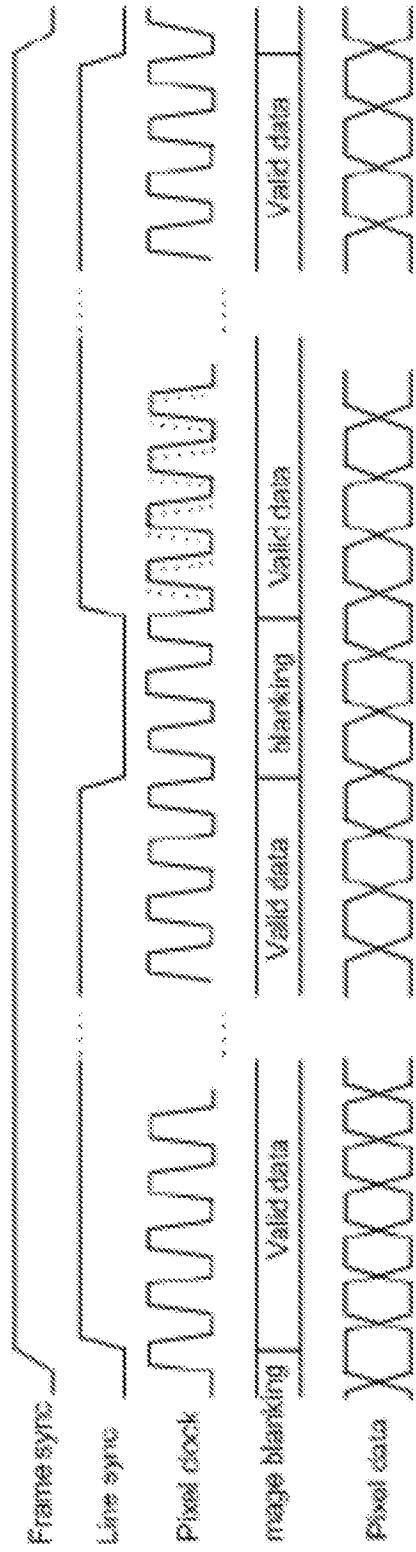


Figure 21A

Figure 21B



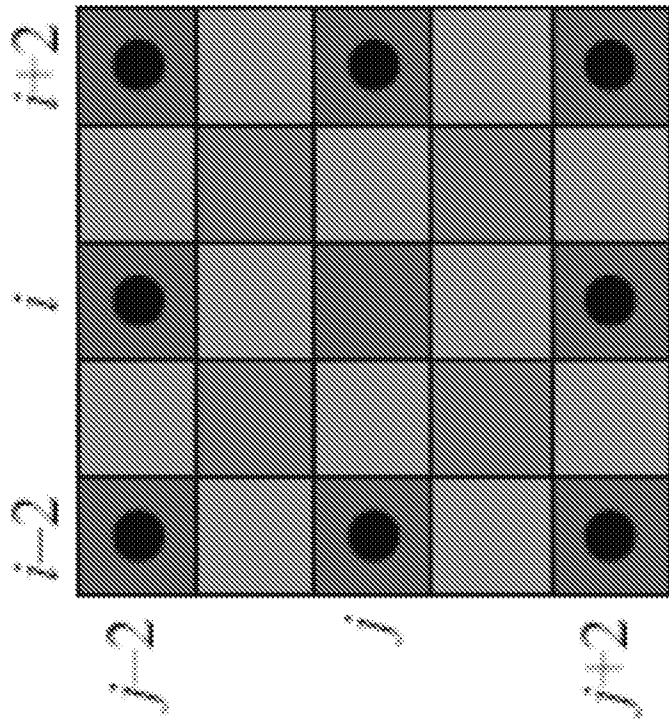
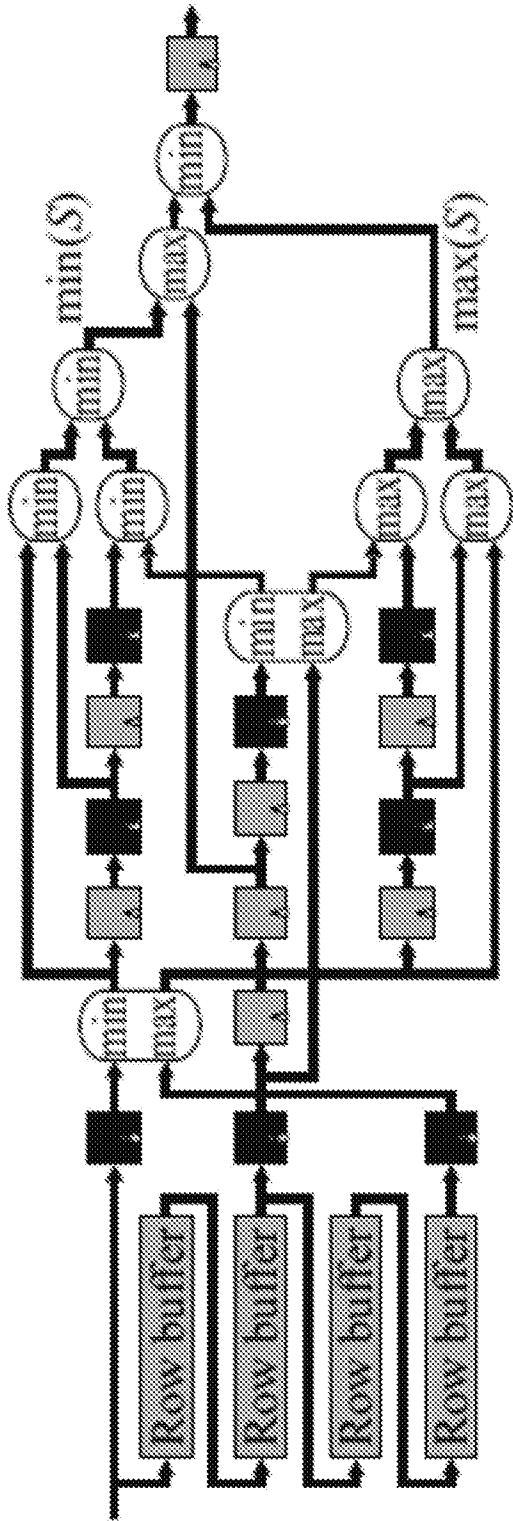


Figure 22A

Figure 22B



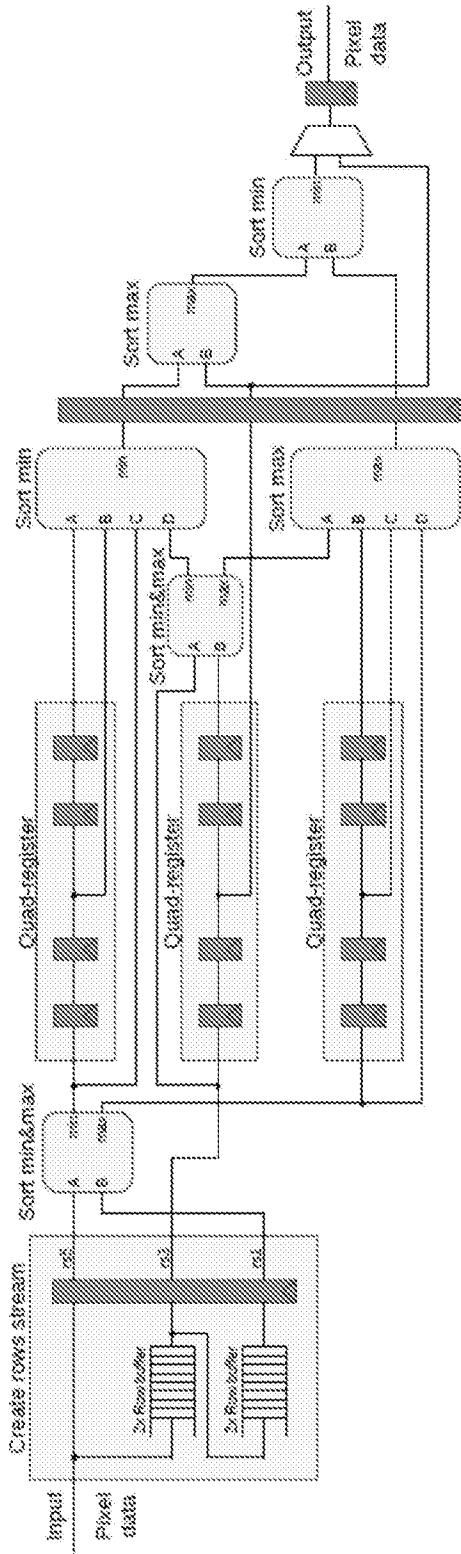
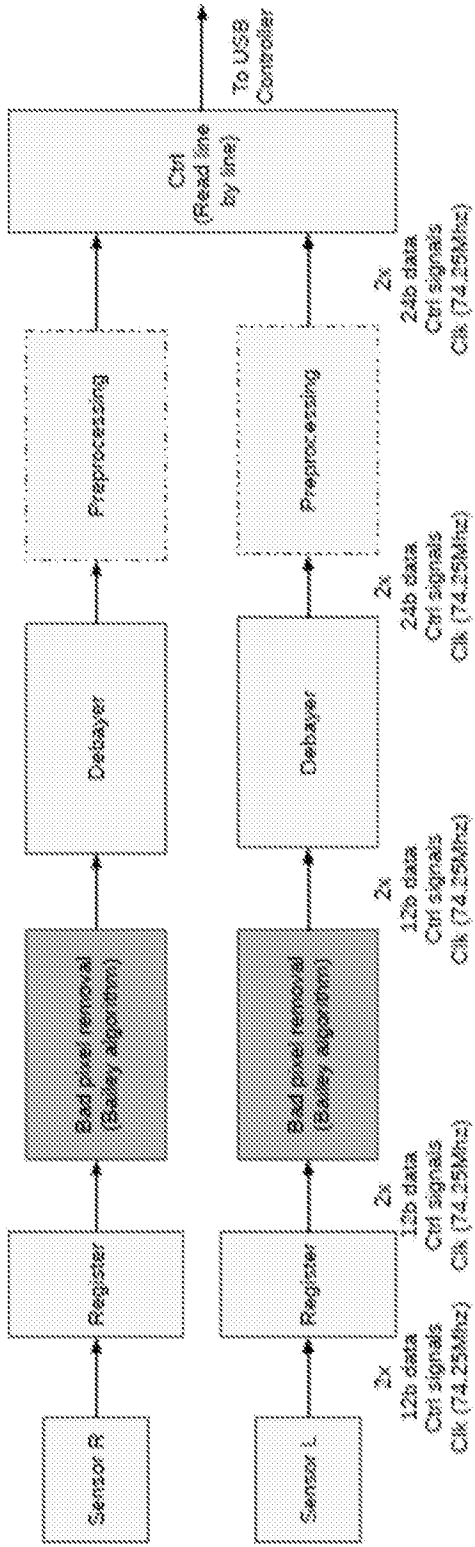


Figure 23

Figure 24



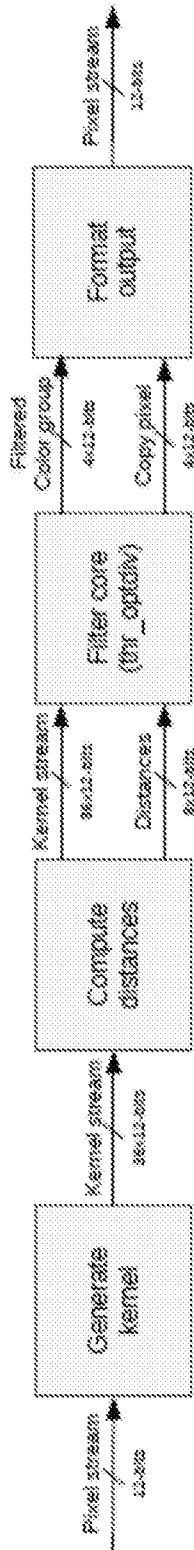


Figure 25

Figure 26

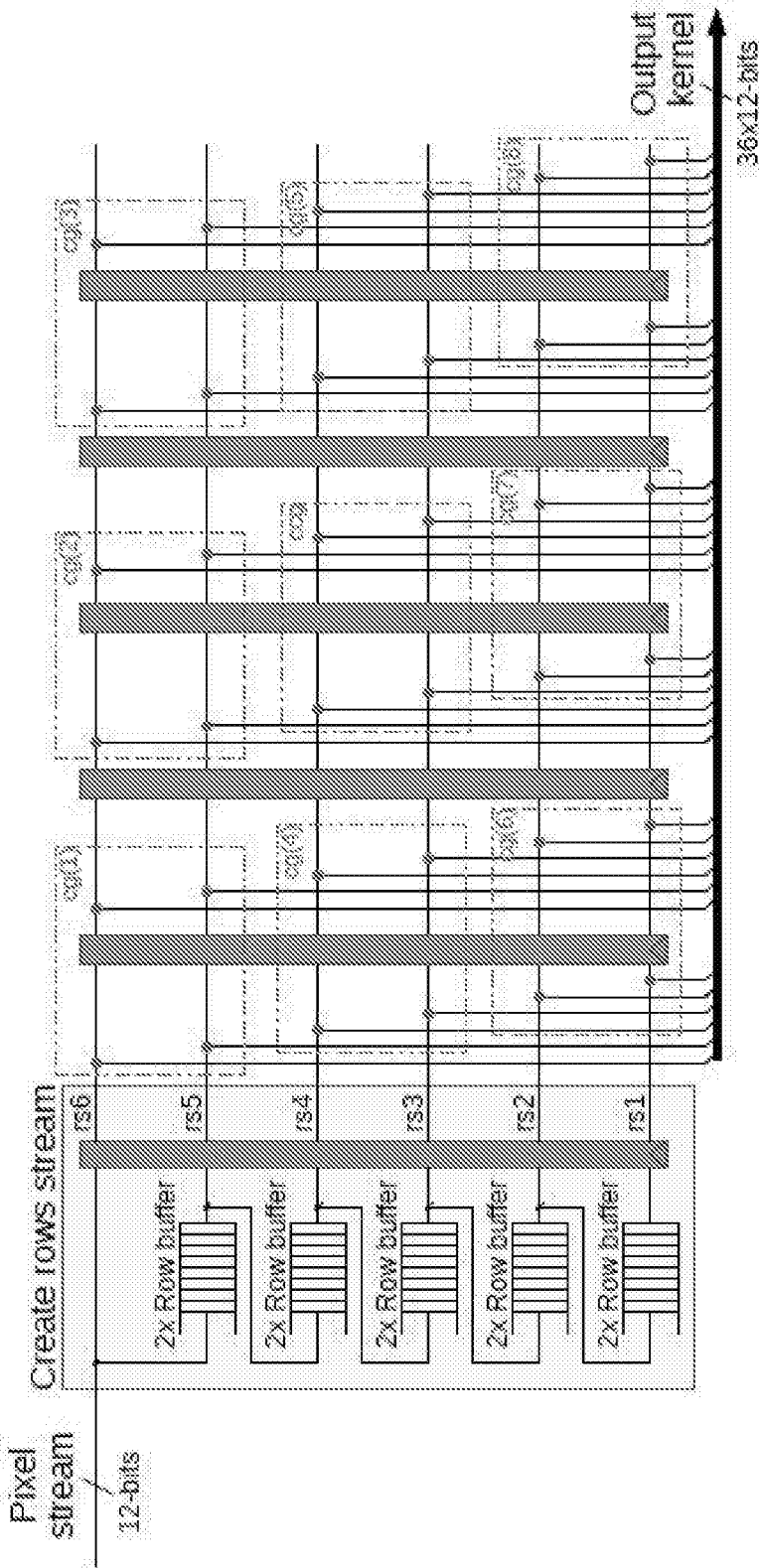
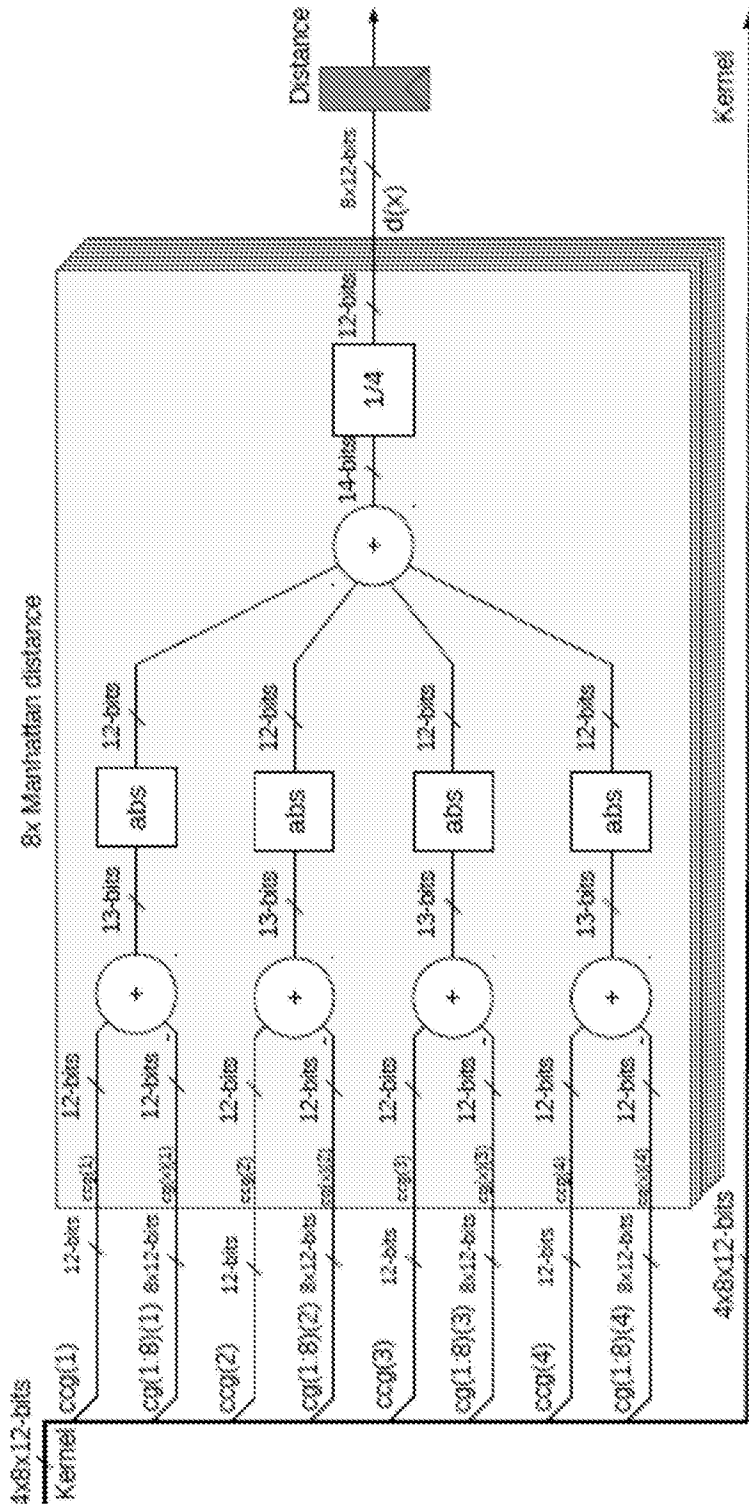


Figure 27



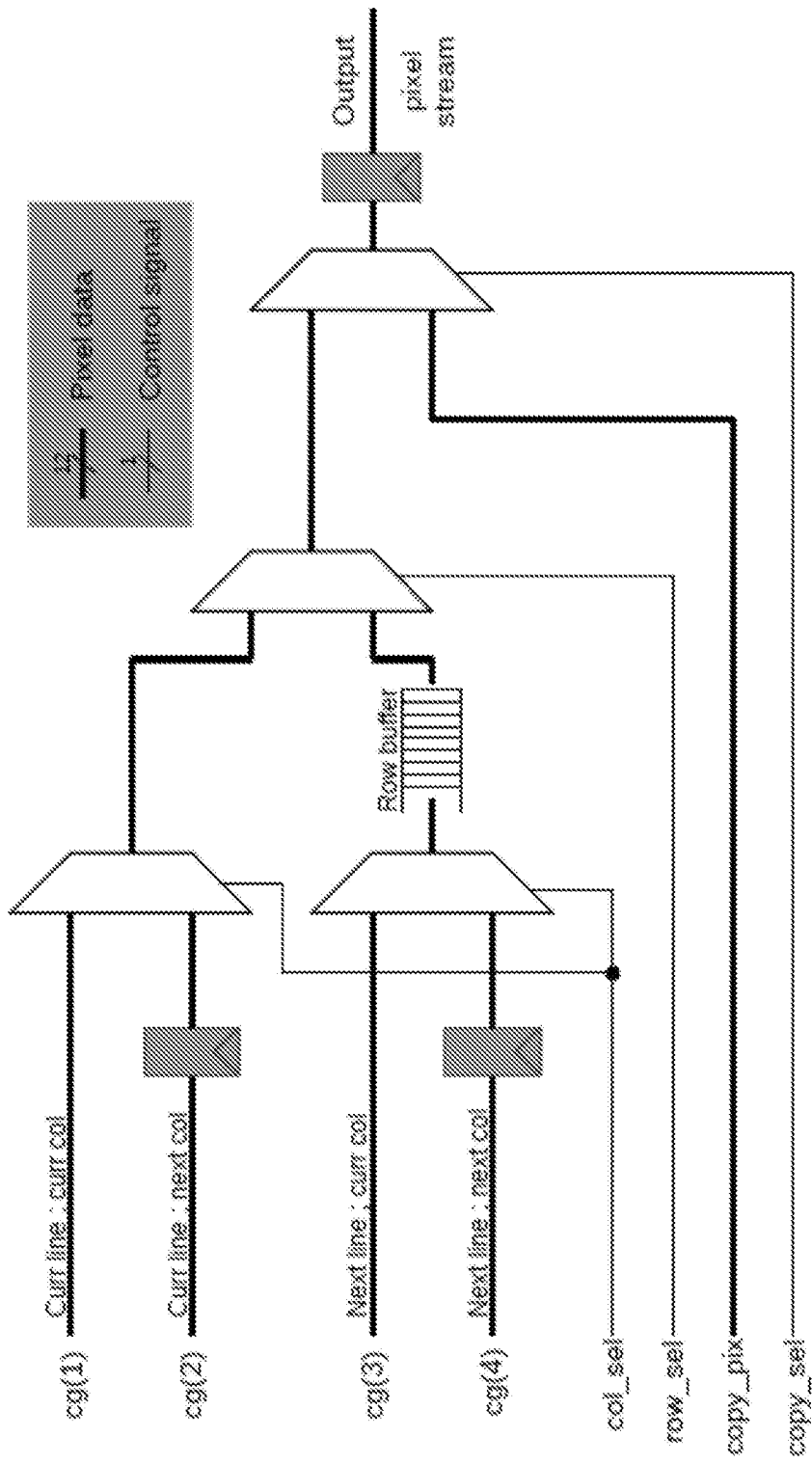


Figure 29A

Figure 29B

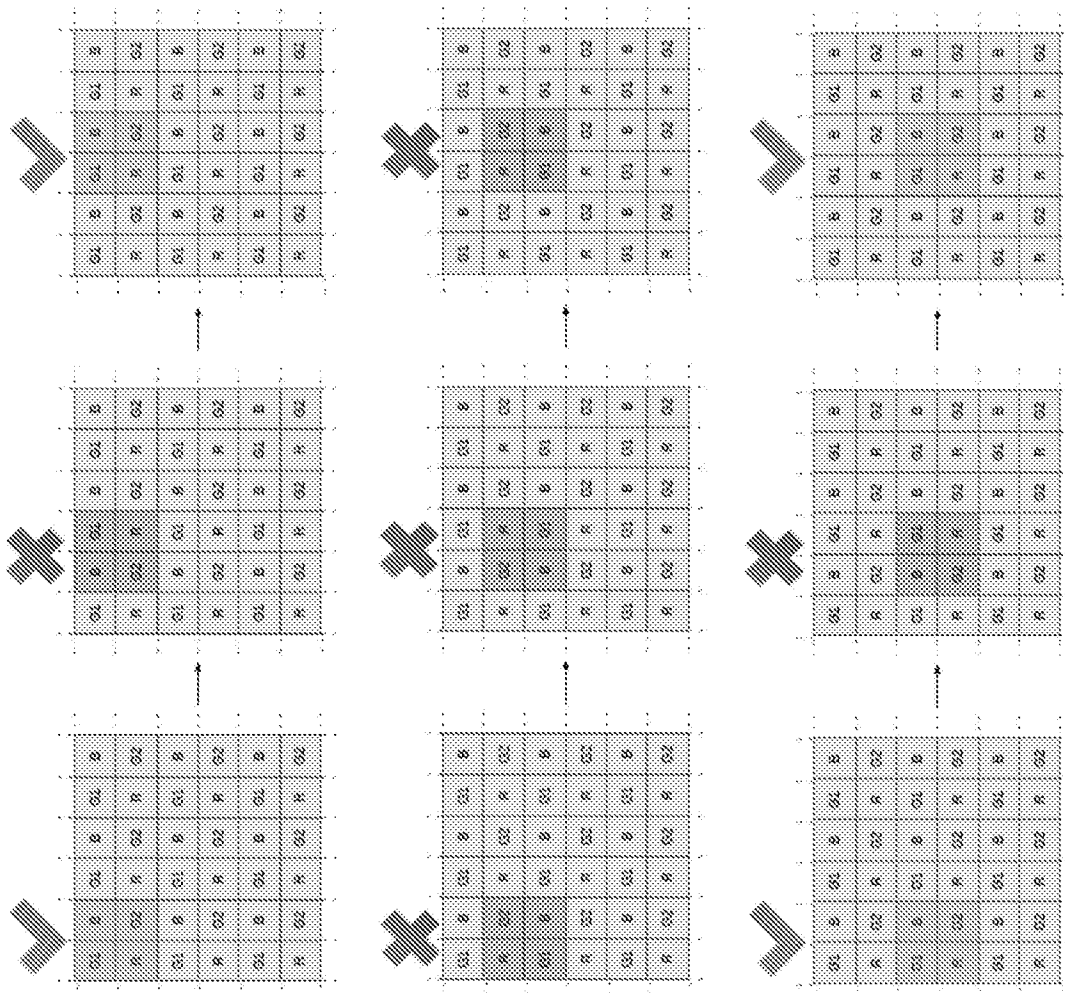


Figure 30

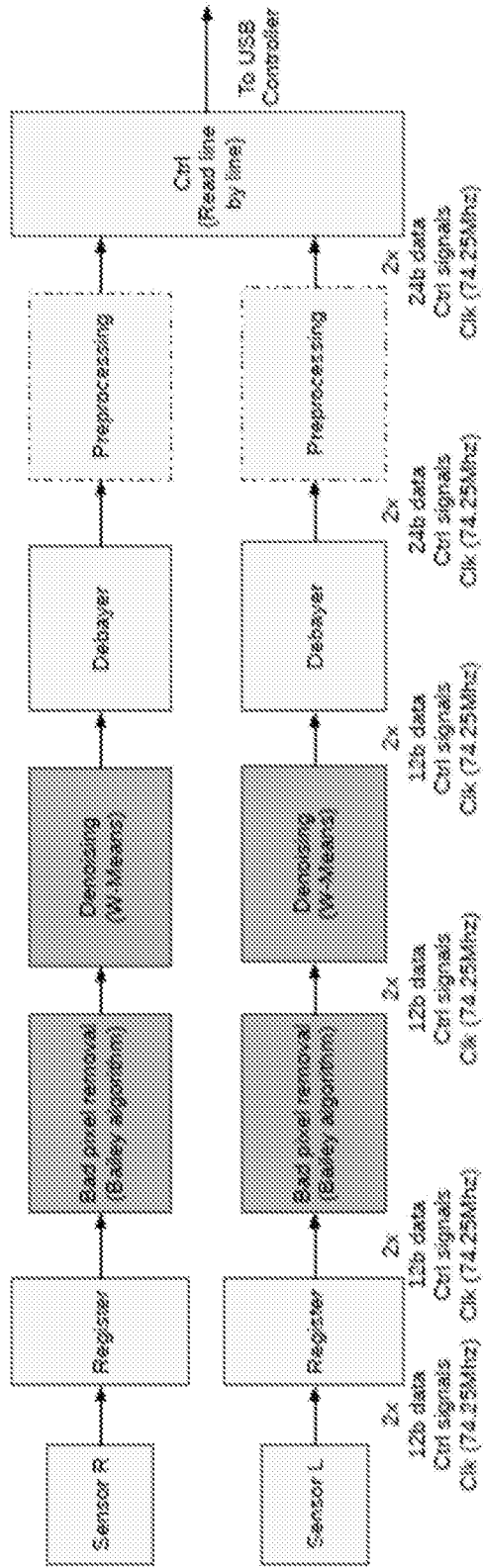


Figure 31B

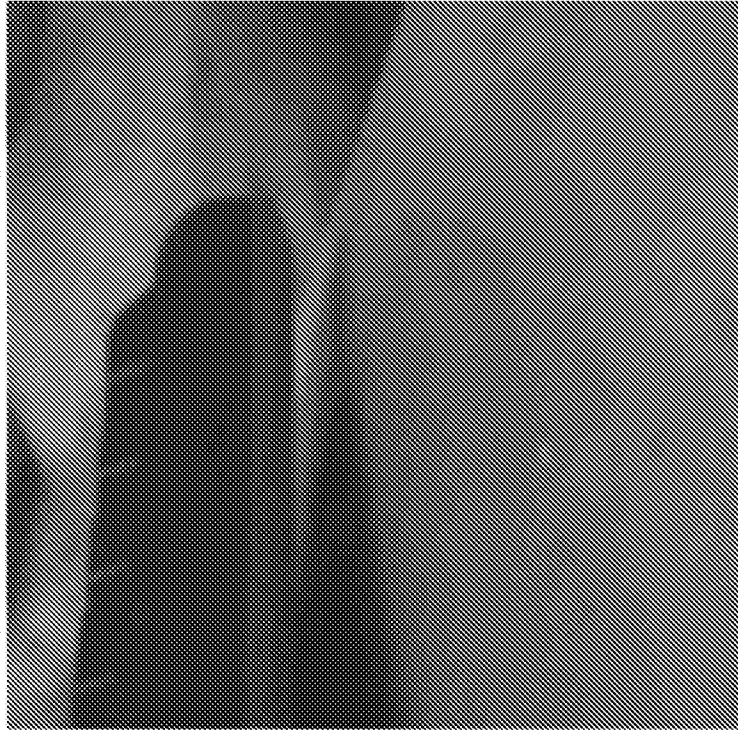


Figure 31A

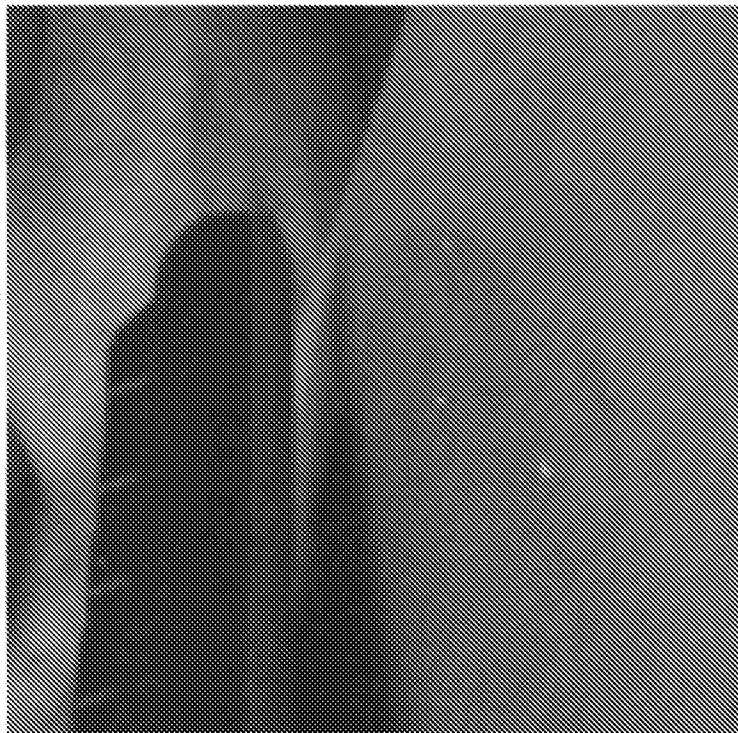
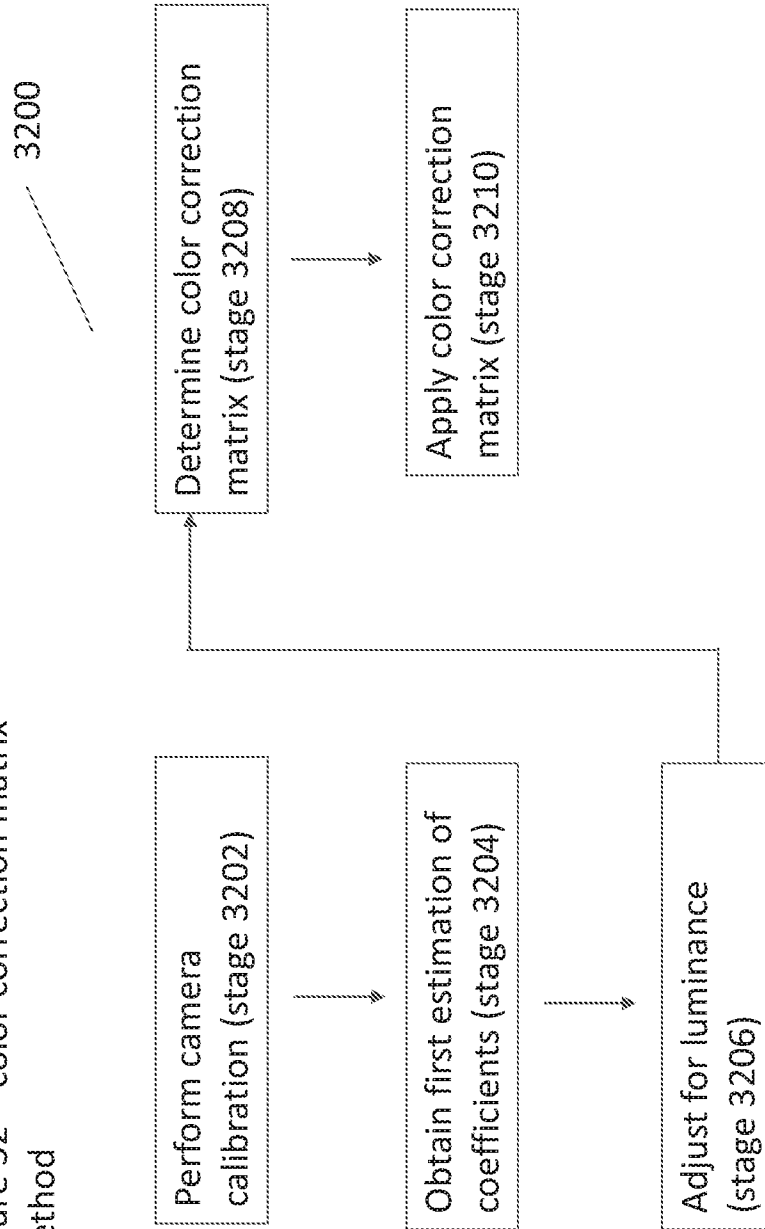


Figure 32 – color correction matrix method



3300

Figure 33B – FPGA top

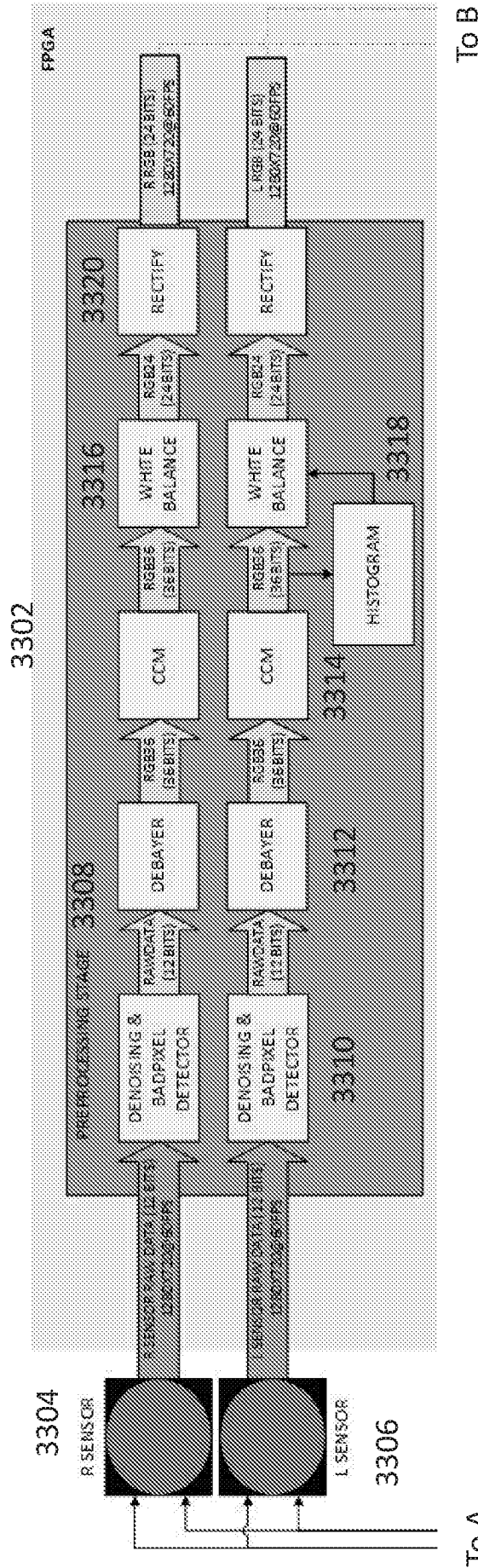
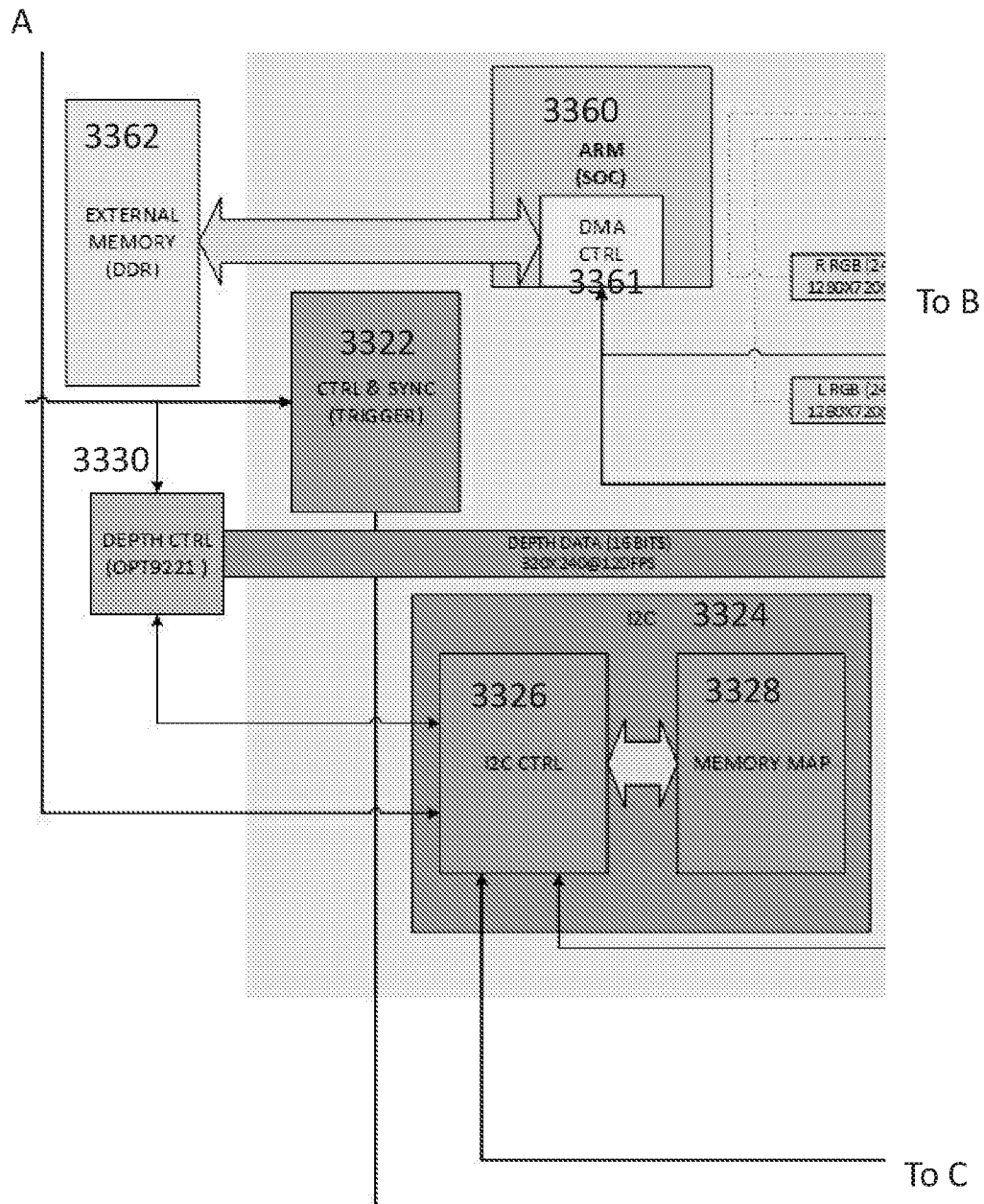


Figure 33C – FPGA bottom left



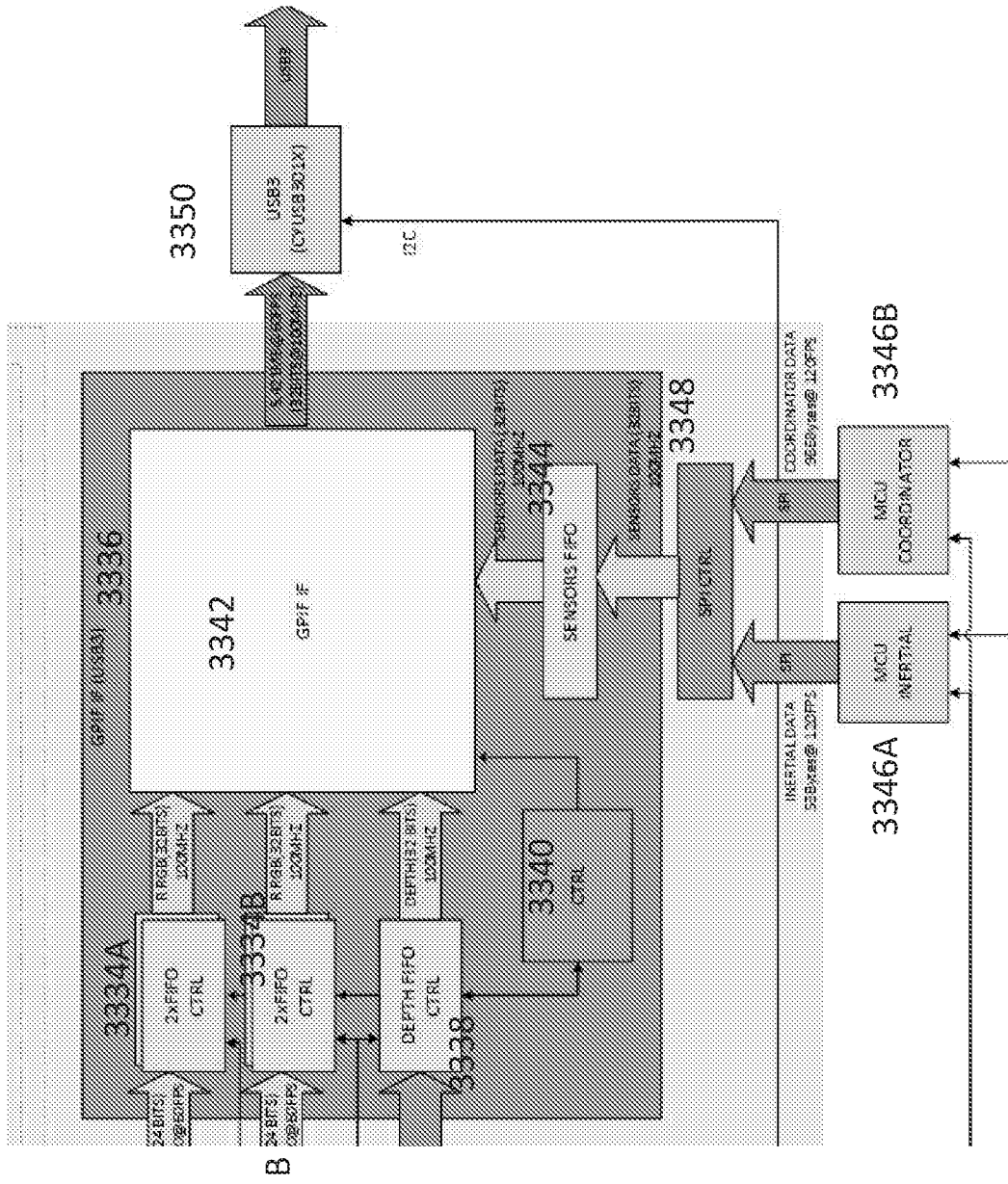


Figure 33D- FPGA right bottom

Figure 34

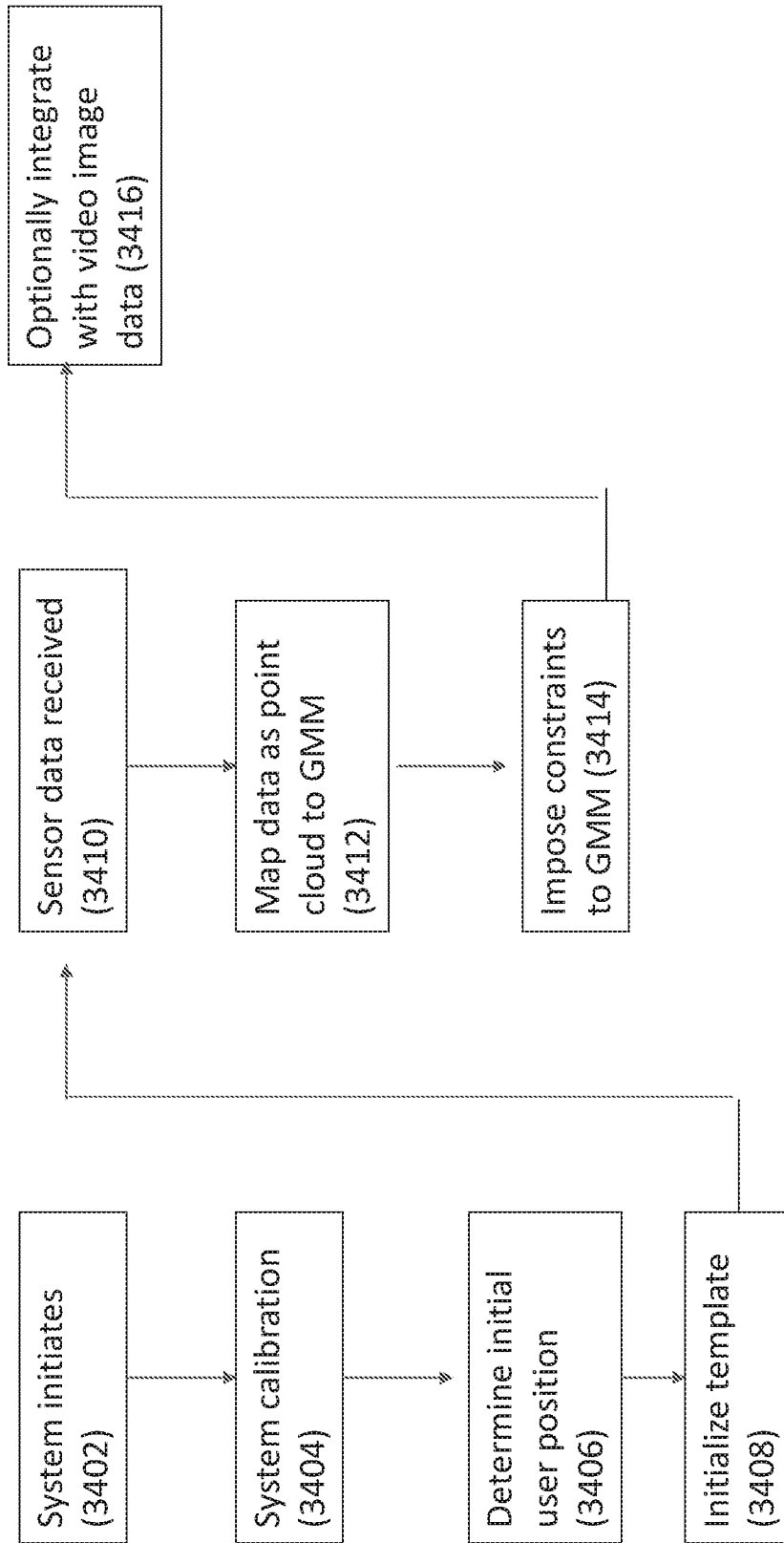


Figure 35

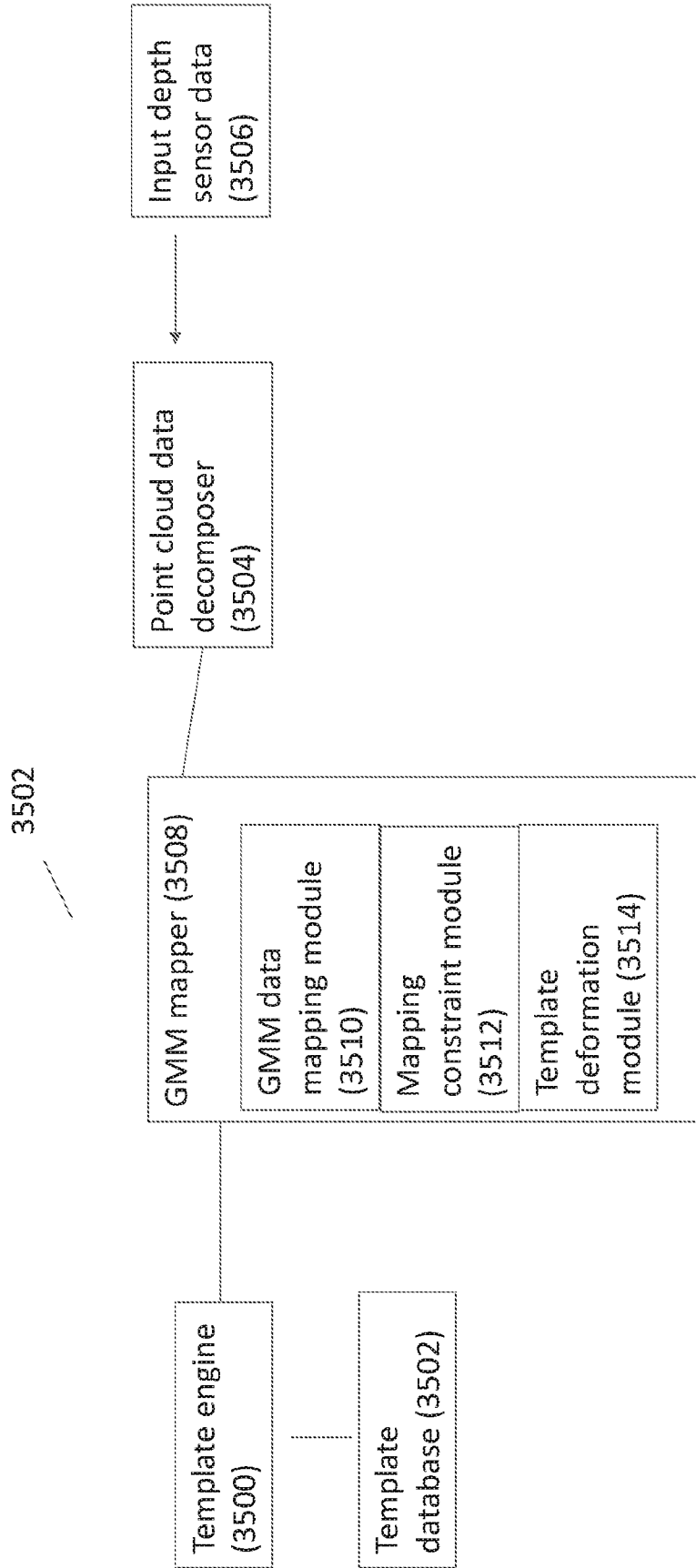


Figure 36

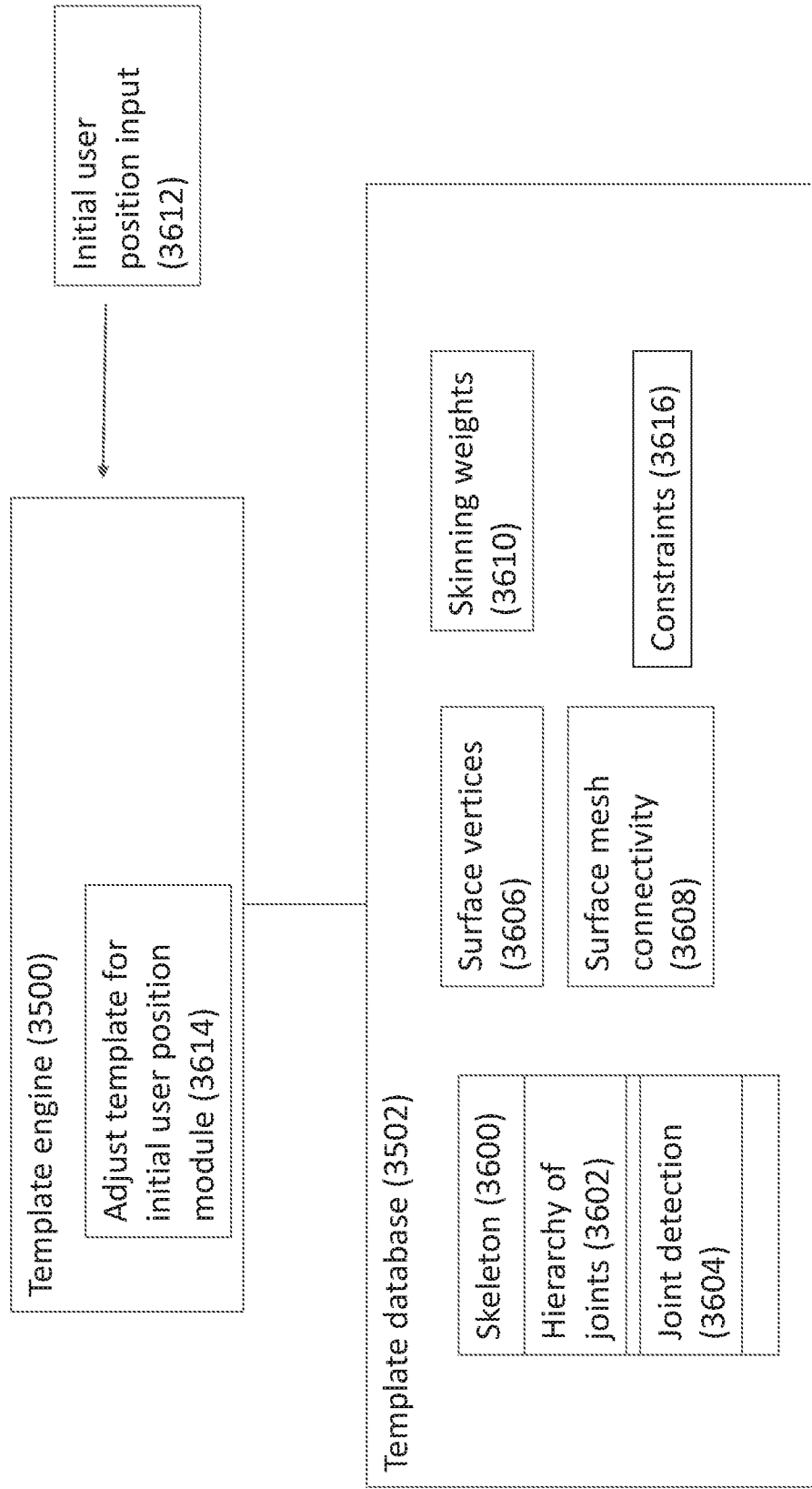


Figure 37

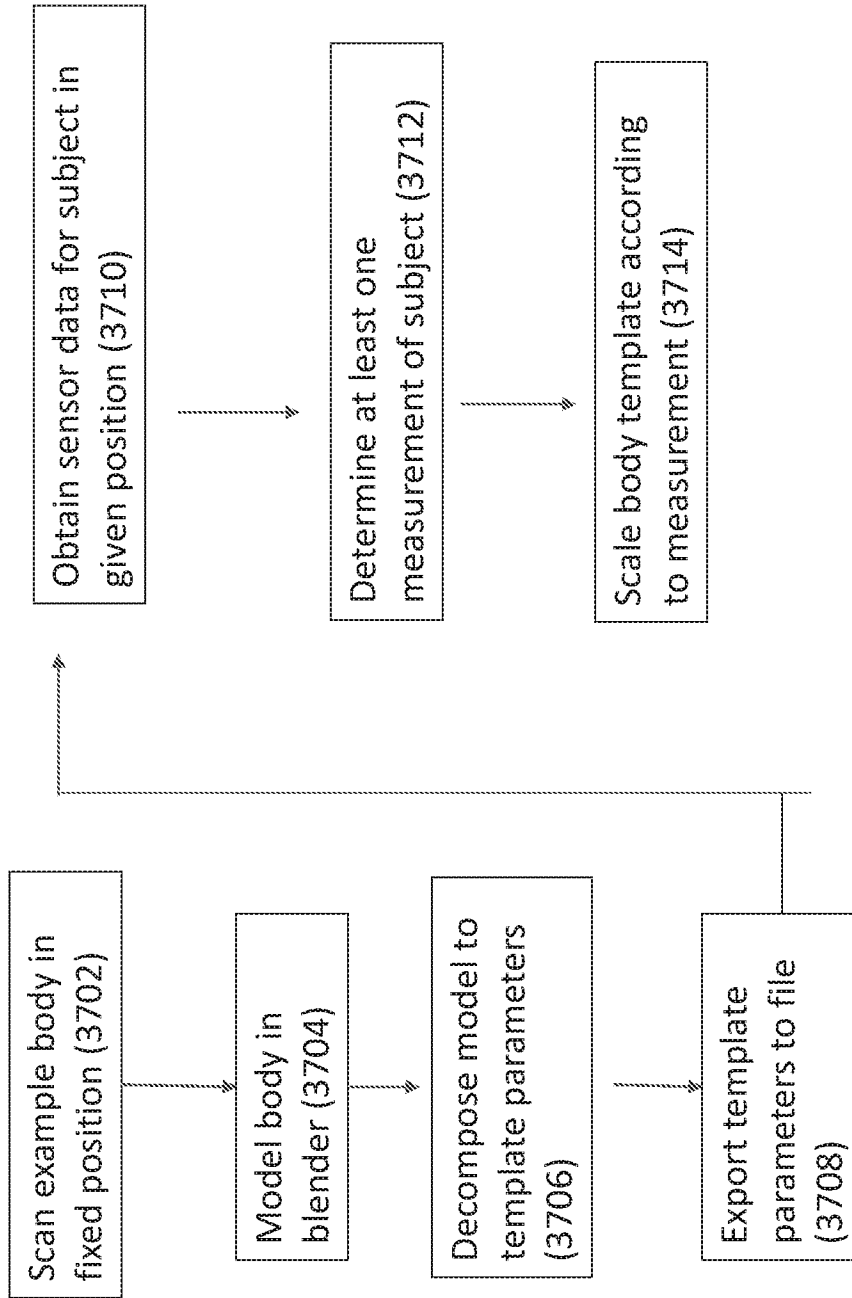


Figure 38A

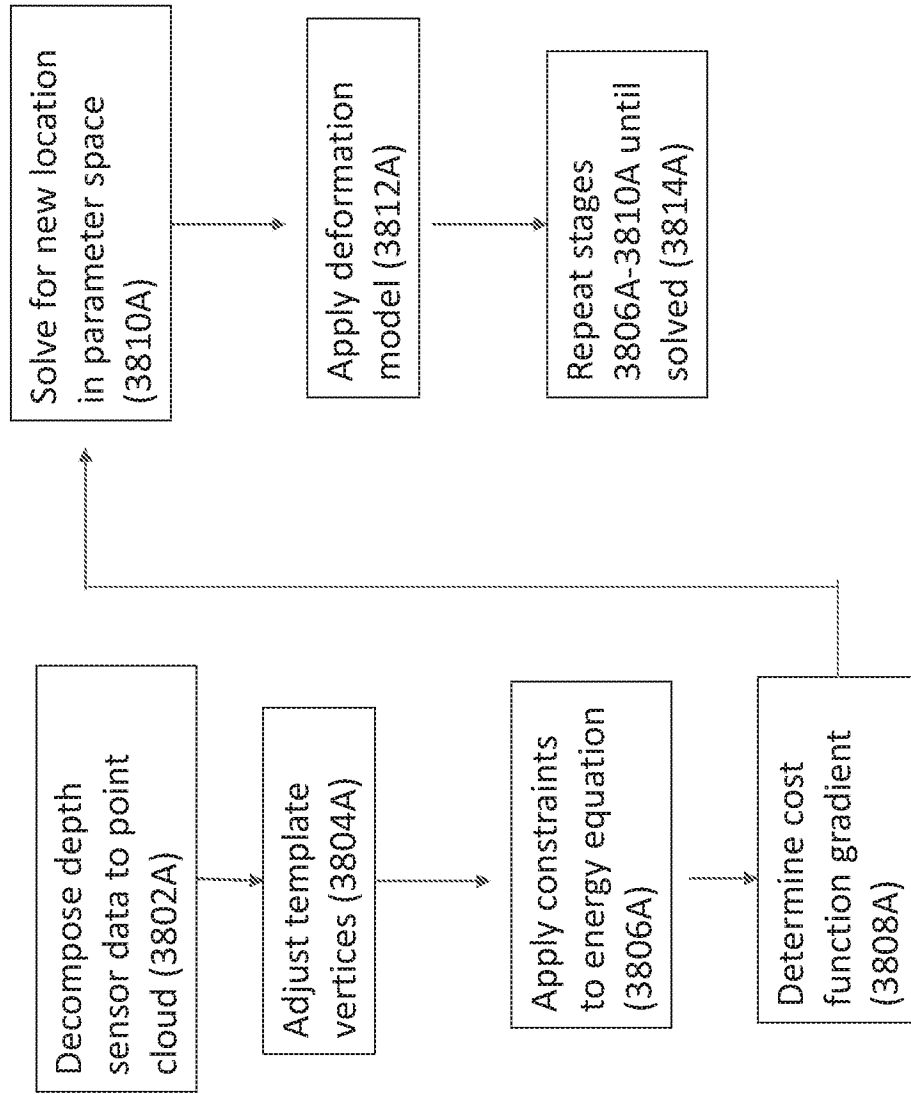


Figure 38B

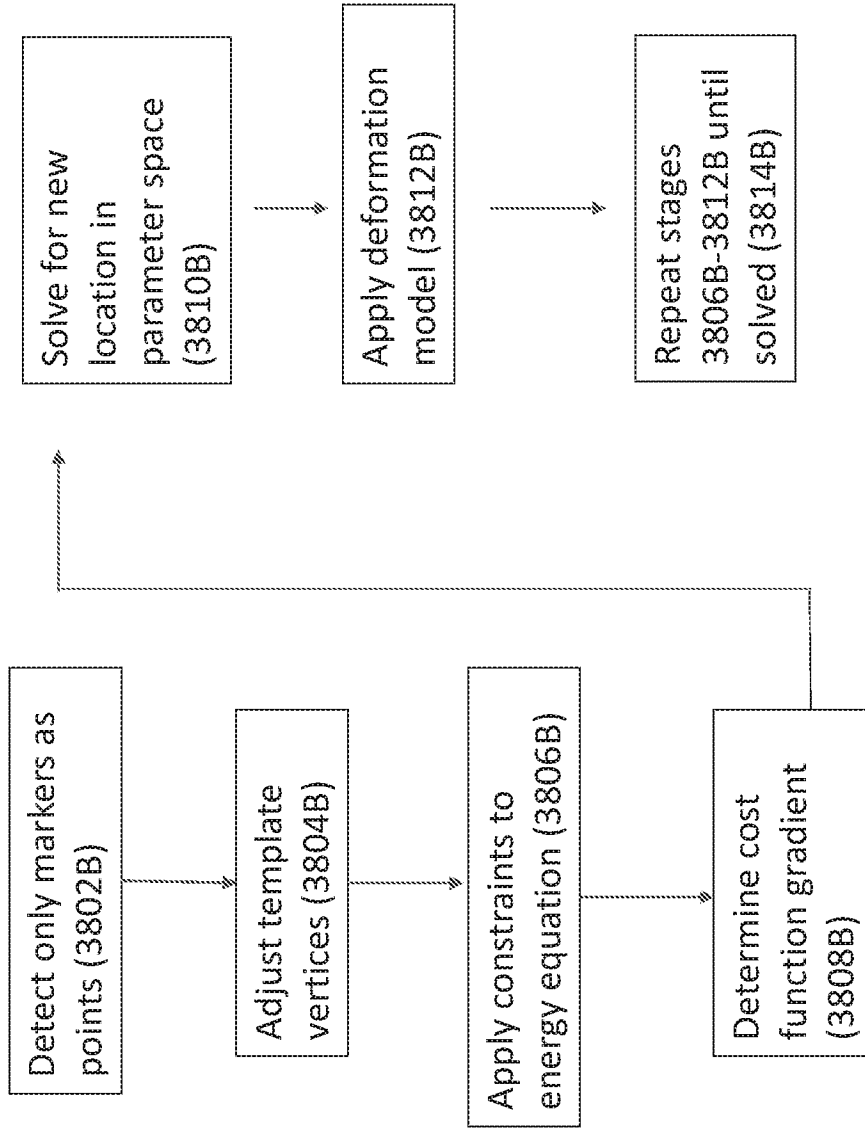


Figure 38C

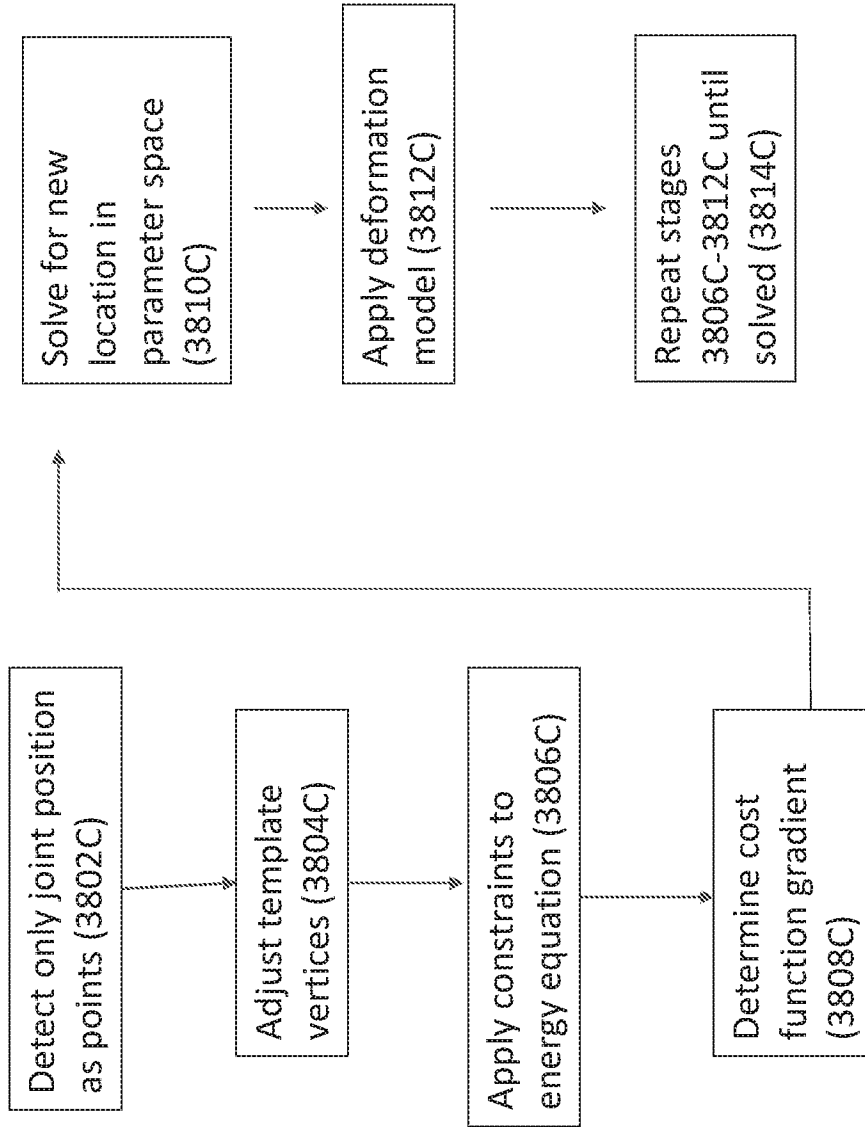


Figure 38D

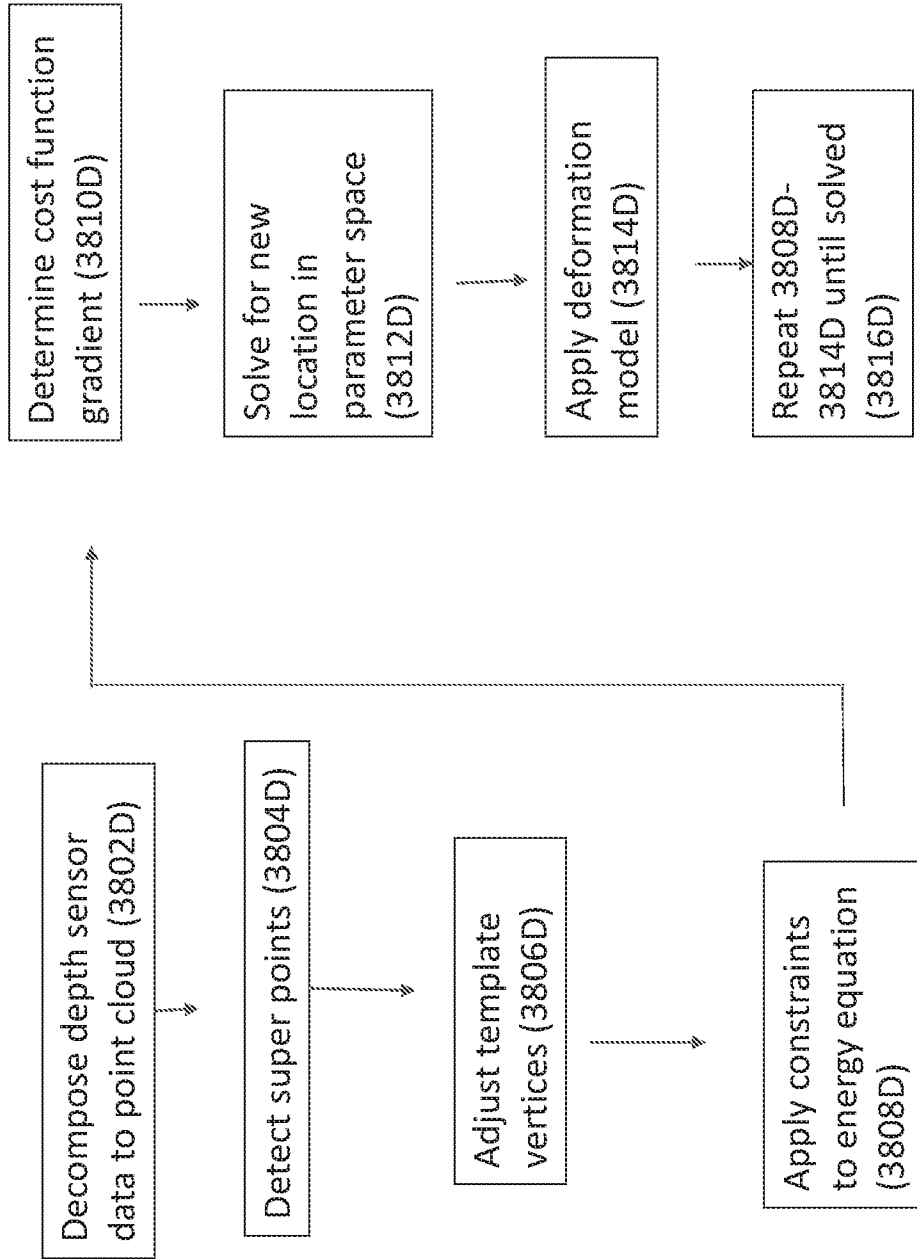


Figure 38E

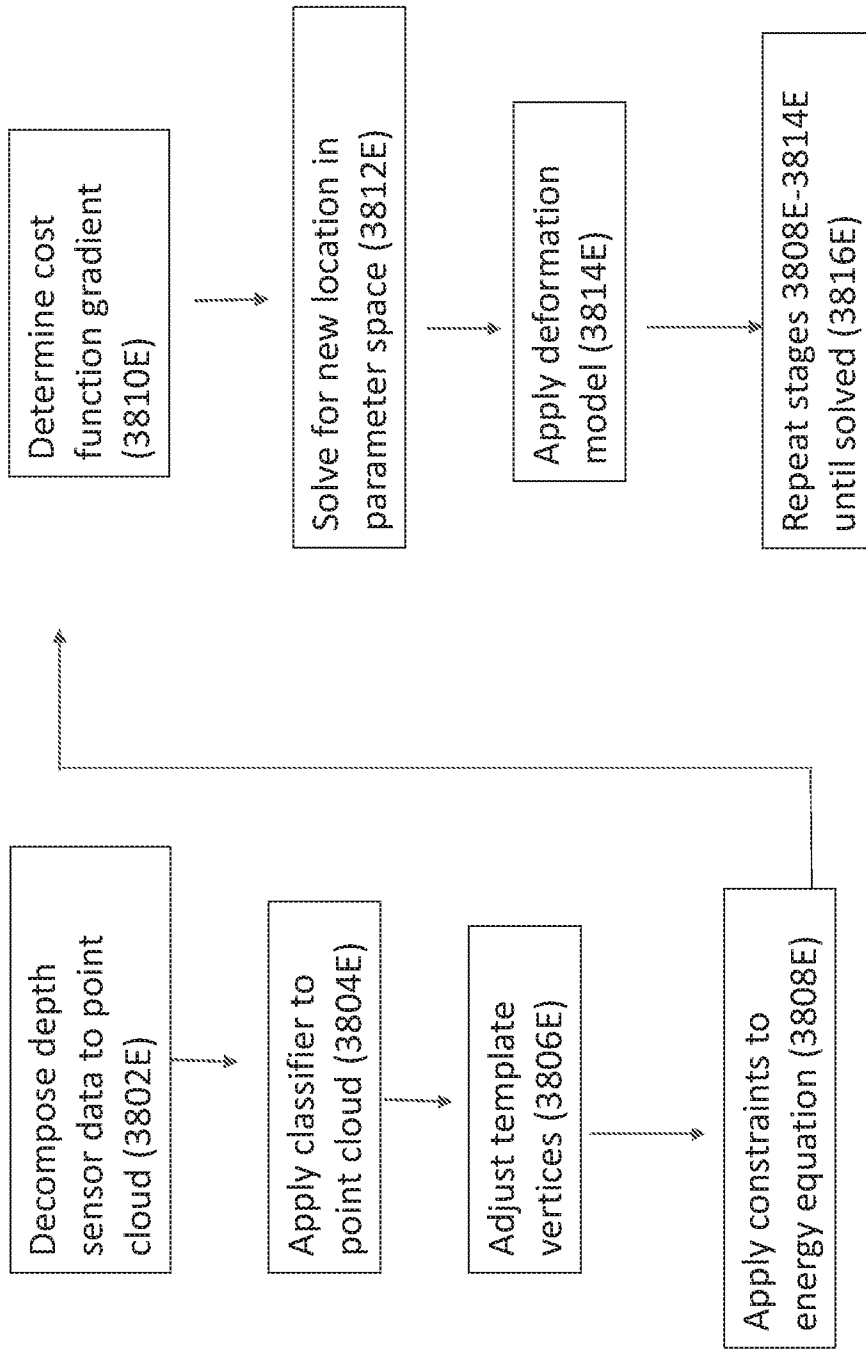


Figure 39

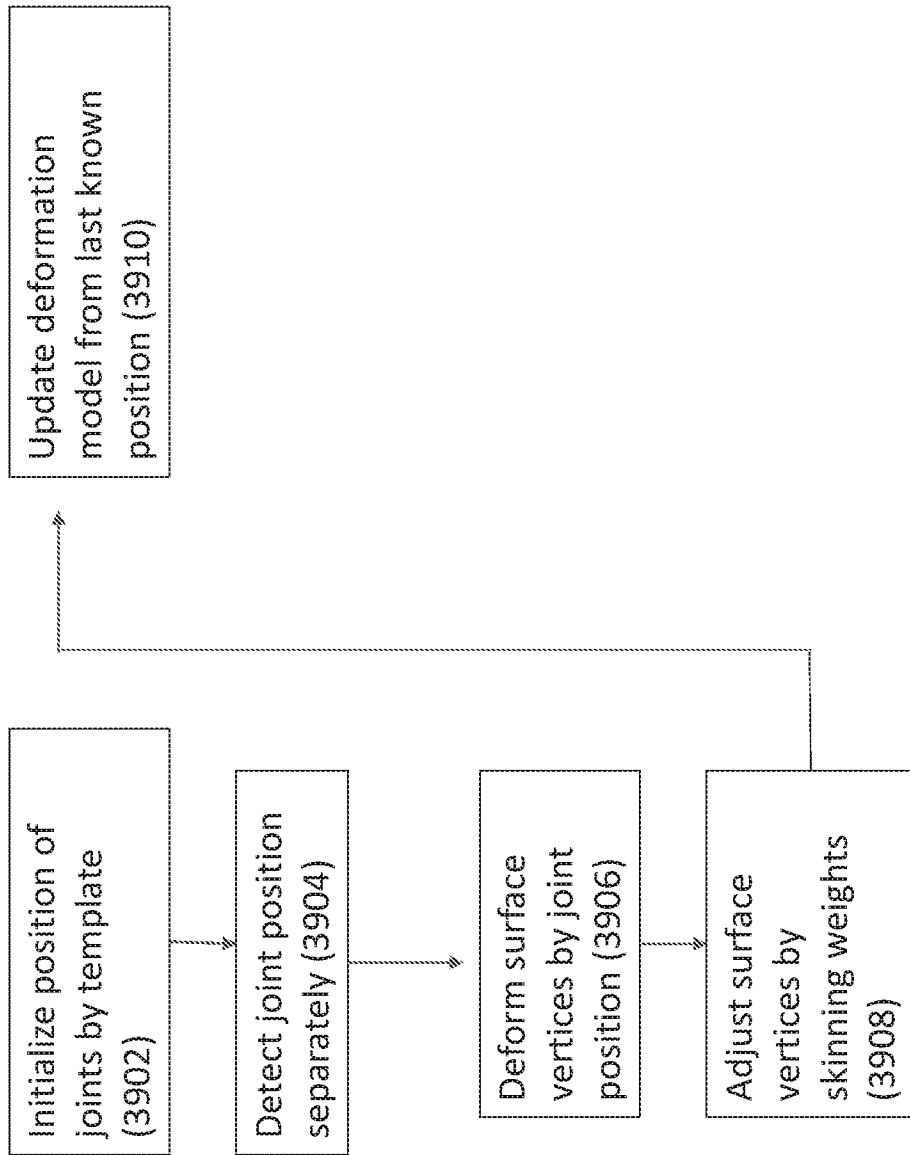


Figure 40

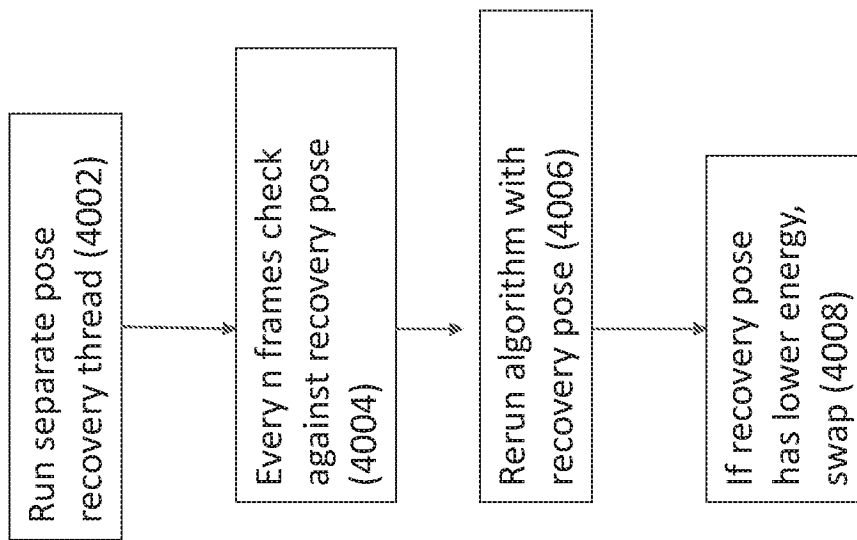


Figure 41

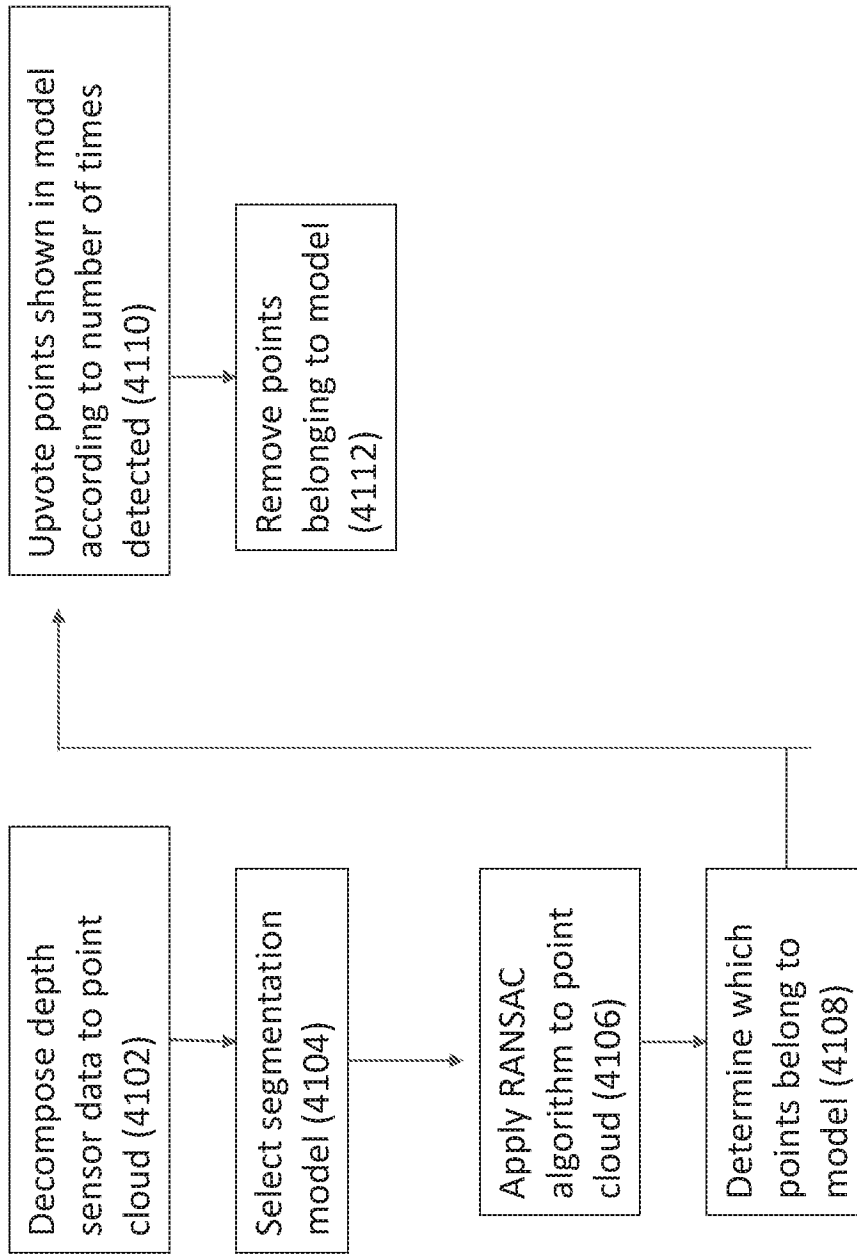


Figure 42

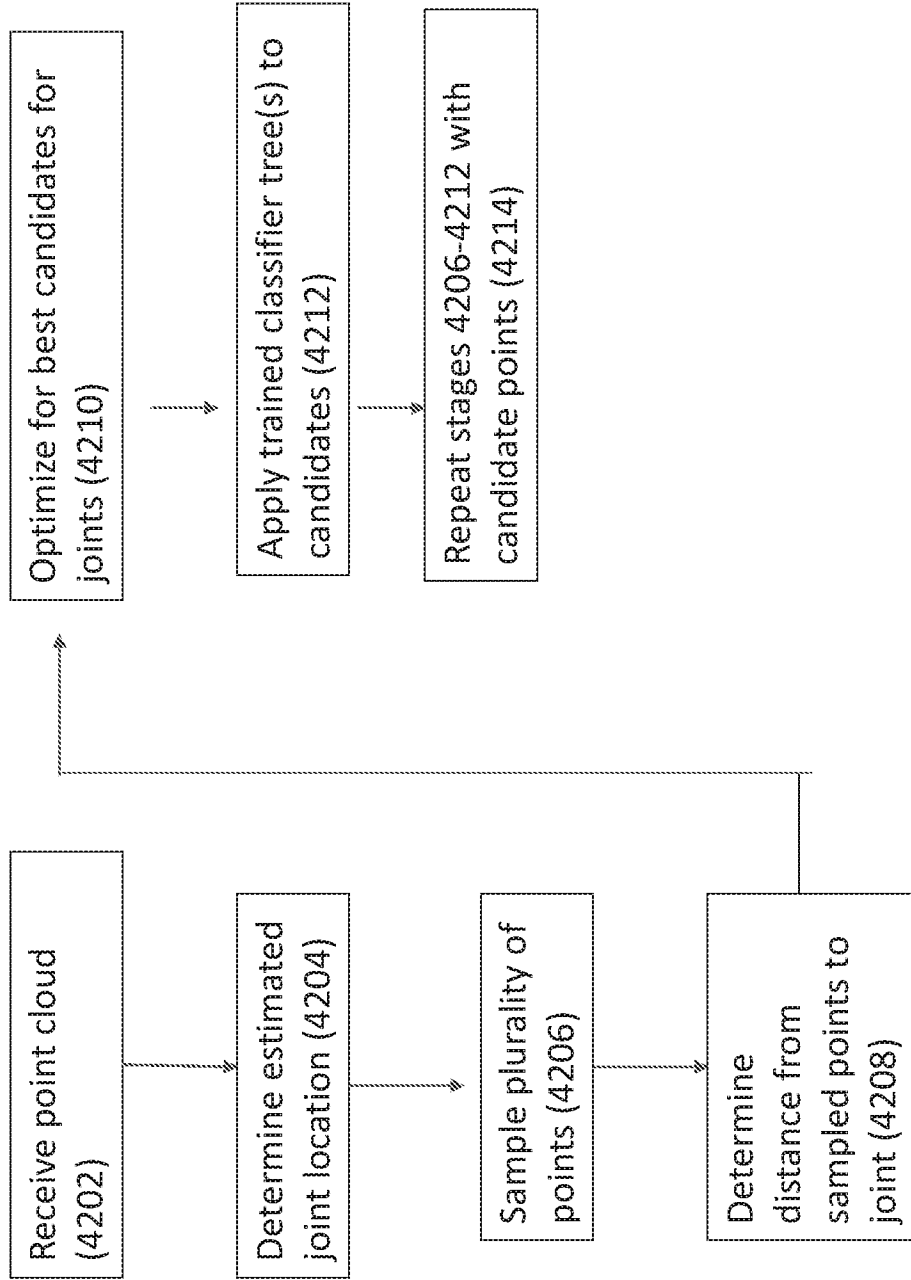


Figure 43

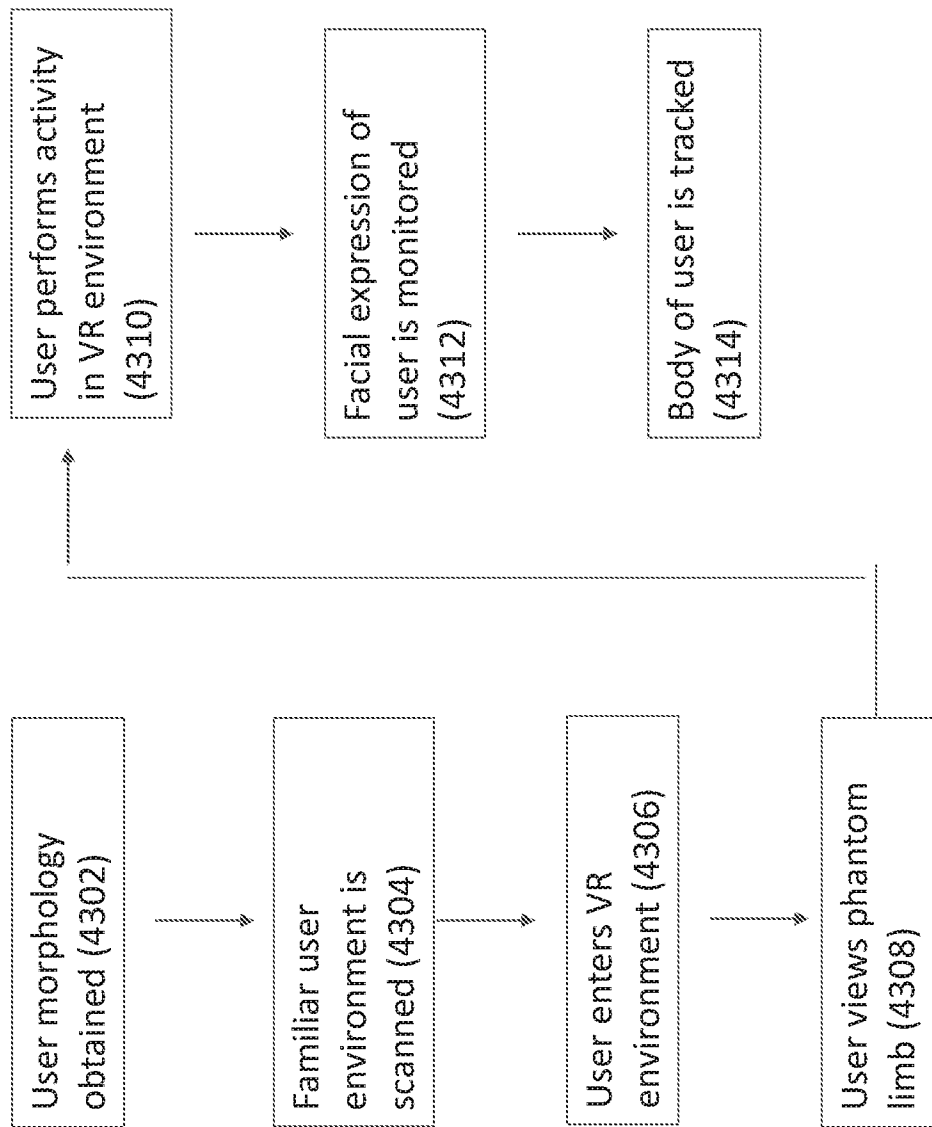


Figure 44

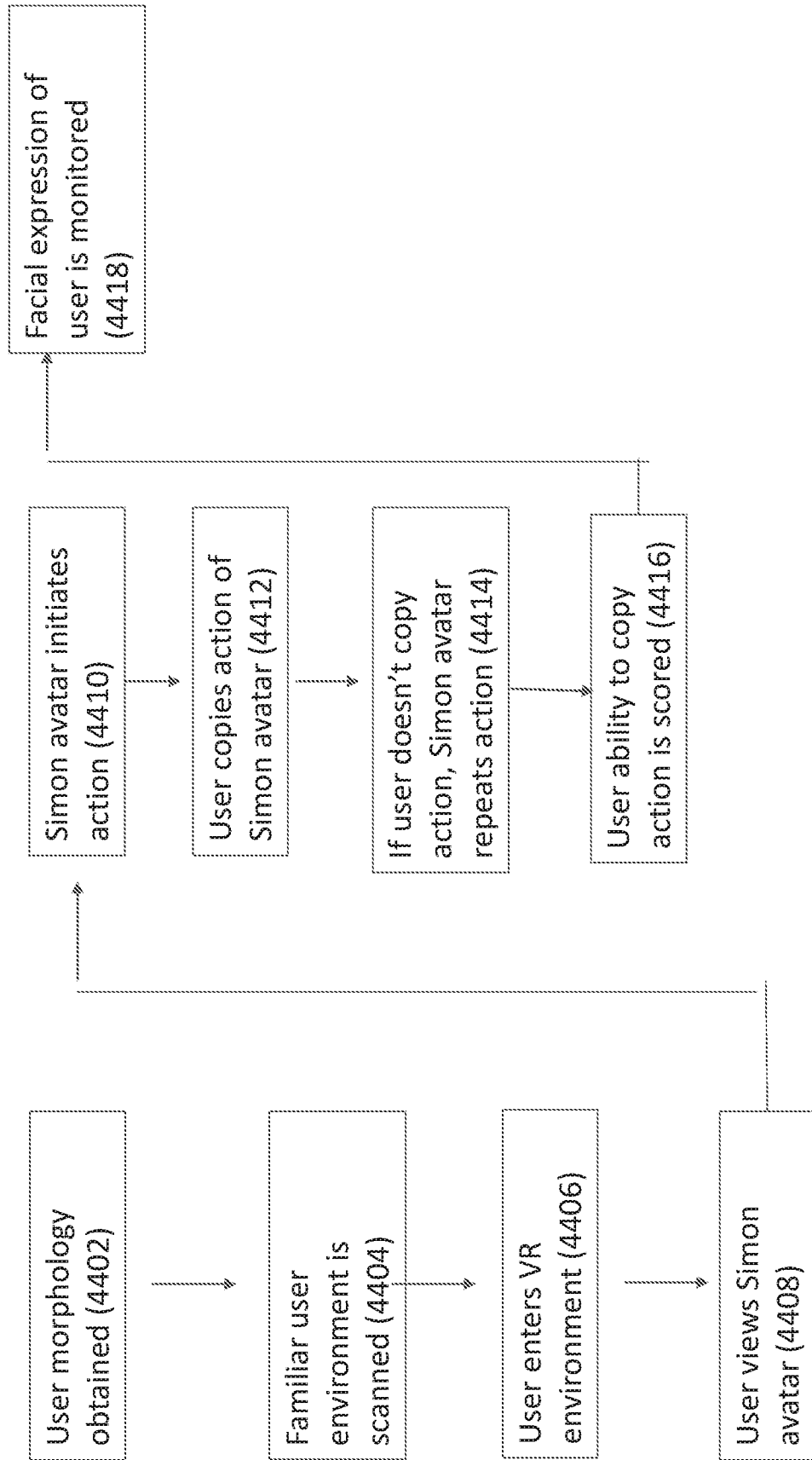


Figure 45

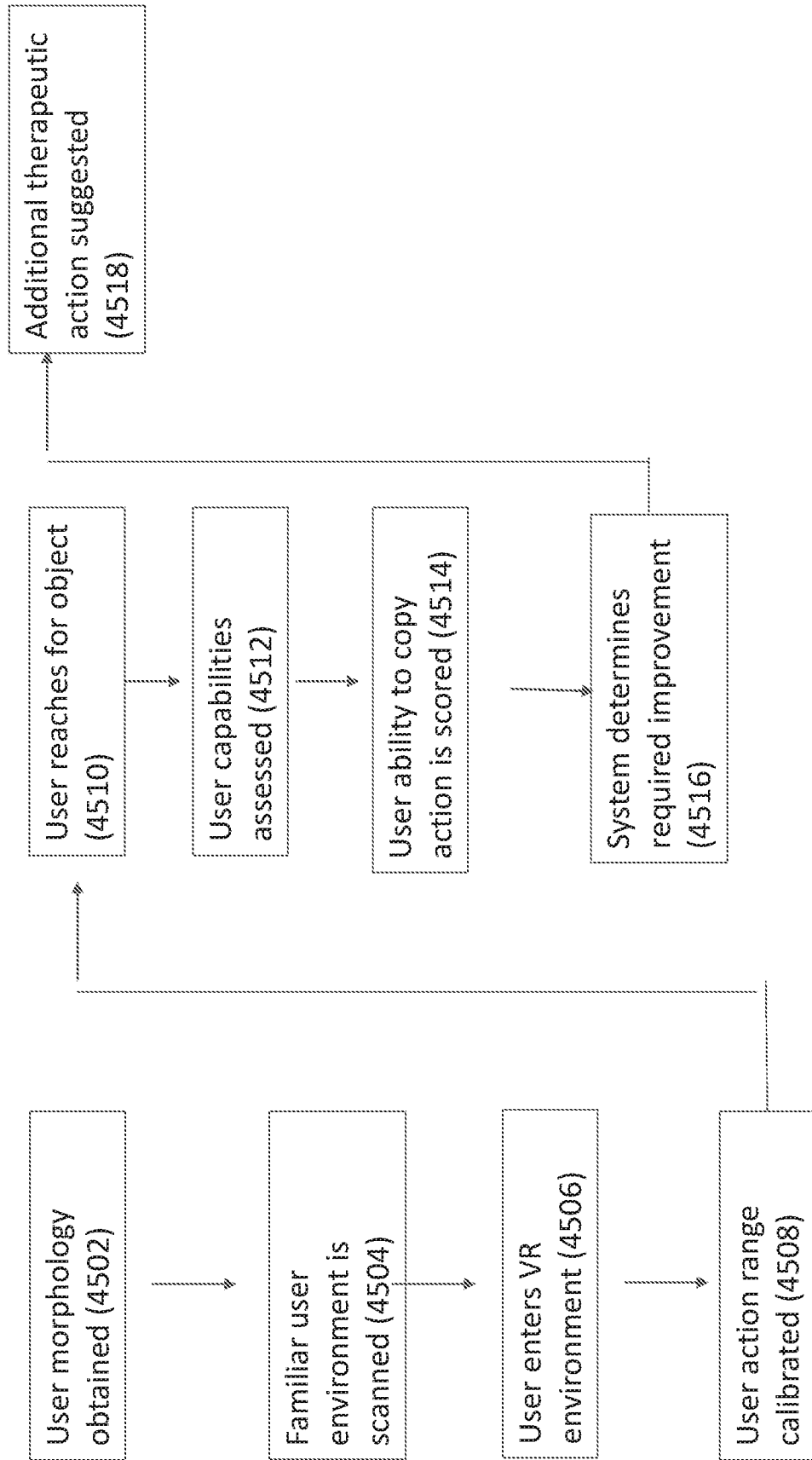


Figure 46

