



- (51) **International Patent Classification:**
H04N 5/225 (2006.01) *H04N 5/228* (2006.01)
- (21) **International Application Number:**
PCT/US2012/031958
- (22) **International Filing Date:**
3 April 2012 (03.04.2012)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
13/155,882 8 June 2011 (08.06.2011) US
- (71) **Applicant (for all designated States except US):** **LYTRO, INC.** [US/US]; 1300 Terra Bella Avenue, Mountain View, California 94043 (US).
- (72) **Inventors; and**
- (75) **Inventors/Applicants (for US only):** **AKELEY, Kurt Barton** [US/US]; 21048 Bank Mill Road, Saratoga, California 95070 (US). **NG, Yi-Ren** [AU/US]; 1038 Haven Avenue, Redwood City, California 94063 (US). **WA-**

TERS, Kenneth Wayne [US/US]; 1742 Cheney Drive, San Jose, California 95128 (US). **FATAHALIAN, Kayvon** [US/US]; 1645 15th Street, San Francisco, California 94103 (US). **KNIGHT, Timothy James** [AU/US]; 4161D El Camino Way, Palo Alto, California 94036 (US). **ROMANENKO, Yuriy Aleksandrovich** [US/US]; 501 W. Hacienda Avenue, #D-23, Campbell, California 95008 (US).

(74) **Agent:** **RAUBVOGEL, Amir H.**; Raubvogel Law Office, 820 Lakeview Way, Redwood City, California 94062 (US).

(81) **Designated States** (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, 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.

[Continued on next page]

(54) **Title:** STORAGE AND TRANSMISSION OF PICTURES INCLUDING MULTIPLE FRAMES

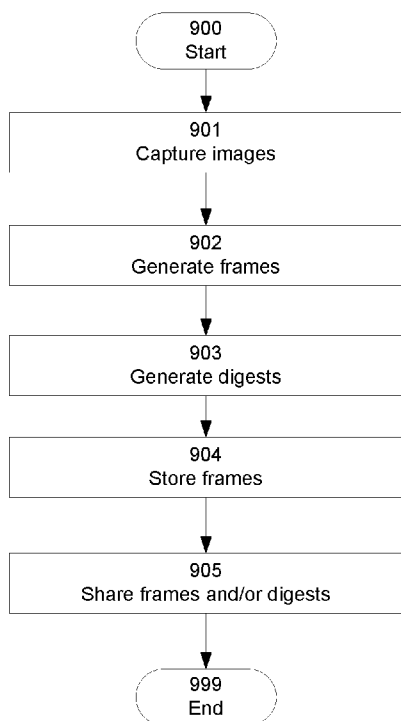


FIG. 9

(57) **Abstract:** According to various embodiments, multiple frames, each having image data and metadata, can be aggregated into pictures. The frames may come from different image capture devices, enabling aggregation of image data from multiple sources. Aggregation can be automatic, or it can be performed in response to user input specifying particular combinations of frames to be aggregated. In various embodiments, pictures are mutable, whereas immutability of the constituent frames is enforced. In various embodiments, certain metadata elements that are not essential to rendering can be selectively removed from frames, so as to address privacy concerns. In various embodiments, frames can be authenticated by the use of digests generated by a hash function.



(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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

- with international search report (*Art. 21(3)*)
- before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (*Rule 48.2(h)*)

STORAGE AND TRANSMISSION OF PICTURES INCLUDING MULTIPLE FRAMES

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application claims priority from U.S. Utility Application Serial No. 13/155,882 for "Storage and Transmission of Pictures Including Multiple Frames," (Atty. Docket No. LYT009), filed June 8, 2011, the disclosure of which is incorporated herein by reference.

FIELD OF THE INVENTION

[0002] The present invention relates to storage, manipulation, and/or transmission of image data and related data.

BACKGROUND

[0003] Light field photography captures information about the direction of light as it arrives at a sensor within a data acquisition device such as a light field camera. Such light field data can be used, along with captured image data, to create representations of scenes that can be manipulated by a user. Subsequent to image capture, light field processing can be used to generate images using the captured image data along with the light field data. Various types of light field processing can be performed, including for example refocusing, aberration correction, changing the viewing perspective, and the like. These and other techniques are described in the related U.S. Utility Application referenced above.

[0004] Conventionally, images may be represented as digital data that can be stored electronically. Many such image formats are known in the art, such as for example JPG, EXIF, BMP, PNG, PDF, TIFF and/or HD Photo data formats.

Such image formats can be used for storing, manipulating, displaying, and/or transmitting image data.

[0005] It is well known that such image data can be edited. Many software applications exist for editing digital images, such as for example Photoshop available from Adobe, Inc. of San Jose, California. Generally, such software applications make changes to image data directly, so that editing causes the original form of the image data to be lost.

[0006] It is also well known to provide the ability to share images, for example via a website, social networking site, email architecture, or the like. Generally, sharing of images requires that image data be transmitted and/or stored in multiple locations; these operations can consume bandwidth and can be inefficient when similar data is repeatedly transmitted and/or stored. Furthermore, repeated transmission, storage, and modification of image data can lead to concerns about privacy and authenticity.

SUMMARY

[0007] According to various embodiments of the invention, a system and method are provided for storing, manipulating, and/or transmitting image data, such as light field photographs and the like, in a unified manner that preserves relationships among constituent data elements, enforces immutability of source images, reduces bandwidth and storage requirements, and addresses privacy concerns. The system and method of the present invention also enable deferred combination of images captured from multiple sources and/or at different times.

[0008] According to various embodiments of the invention, a container file representing a scene (referred to herein as a “picture” or “picture file”) can include or be associated with any number of component image elements (referred to herein as “frames”). Frames may come from different image capture devices, enabling aggregation of image data from multiple sources. Frames can include image data as well as additional data describing the scene, its particular characteristics, image capture equipment, and/or the conditions under which the

frames were captured. Such additional data are referred to herein as metadata, which may be universal or application-specific. Metadata may include, for example, tags, edit lists, and/or any other information that may affect the way images derived from the picture look. Metadata may further include any other state information that is or may be associated with a frame or picture and is visible to an application. Picture files may also include instructions for combining frames and performing other operations on frames when rendering a final image.

[0009] In one embodiment, frames and pictures are implemented as data structures that can be manipulated by software running on a digital camera, computing device, or other electronic device. Frames and/or pictures can be stored on any suitable electronic storage device, using a suitable file format, and can be transmitted among electronic devices using any suitable protocol and/or network communication scheme.

[0010] Using the data structures and other characteristics defined herein, pictures having any number of frames can be stored, manipulated and/or transmitted. Each frame can represent an image captured by a sensor in a digital camera, along with appropriate state information. Each image can be a light field image or a conventional two-dimensional image. A frame having a two-dimensional image is referred to as a two-dimensional frame; a frame having a light field image is referred to as a light field frame. Pictures can include two-dimensional frames, light field frames, or any combination thereof. A picture can also be reduced to, or converted to, any standard image format as desired.

[0011] In one embodiment, each frame includes data representing an image detected by the sensor of the camera (image data), and may also include data describing other relevant camera parameters (metadata). Examples of metadata include camera settings such as zoom and exposure time, the geometry of a microlens array used in capturing a light field frame, and the like. The metadata may be provided in text format, XML, and/or in any other suitable

format. The image data contained in each frame may be provided in any suitable format, such as for example a raw image or a lossy compression of the raw image. If the image is a raw image, the frame is referred to as a raw frame.

[0012] In one embodiment, pictures can contain any number of frames, and can also contain additional metadata, if appropriate. In addition to one or more frames and/or metadata, pictures can also include data derived from the frames and metadata.

[0013] In some embodiments, constituent frames of a picture may have been captured at different times and even by different data acquisition devices. For example, several cameras can be used for capturing images to be aggregated into a picture. In such a scenario, each individual camera may not possess the complete picture, as each is contributing only a subset of the frames for the picture. According to one embodiment, the present invention enables deferred aggregation, wherein cameras are able to output pictures (either 2-D or light field), and their various components (such as frames), even if additional frames and/or data is/are expected at a later time. Additional frames and/or data can be aggregated with such pictures at a later time, either by the camera itself or by software running at a computing device.

[0014] In one embodiment, the system of the present invention provides mechanisms for ensuring that original frame data is preserved, even when users and/or other entities make changes to pictures. Specifically, in one embodiment, the system of the present invention allows editing of pictures (aggregations of frames, as well as other data), without allowing changes to the original frames themselves. Thus, frames are treated as immutable, so that any editing or changes made to images are made to pictures rather than their constituent frames. More particularly, special effects, distortions, color effects, refocusing, and/or other operations that can be performed on images are stored as instructions within the picture data structure, and are applied when rendering the final image, so that the original image data is not overwritten. Thus, the

instructions specified within a picture are mutable, even though the included frames remain immutable.

[0015] In one embodiment, the immutability of frames is enforced by the use of a digest, or hash, which is a unique identifier that may be recomputed (invariantly) from the frame's contents at any time. Unauthorized changes to a frame can thus be detected by comparing the frame's digest with a previously computed digest for that frame. In one embodiment, each picture file can include a digest for each frame, even when the frame itself is included in the picture, so that any unauthorized changes to frame data can be detected by performing a new hash operation on the frame data and comparing the digest with the result of the hash operation.

[0016] In one embodiment, the system of the present invention includes mechanisms for displaying a final image at an output device, based on stored picture data. The picture data may include any number of frames, as well as descriptions of operations that are to be performed on the frames. In one embodiment, aggregation and other operations are performed "on the fly" when an image is to be rendered for display or transmission. In another embodiment, a picture can be reduced or converted into an image file having conventional image data format; this image file can then be displayed, rendered, or transmitted by conventional means.

[0017] The present invention also provides a mechanism for improving the efficiency of image sharing. For example, when changes are made to a picture whose frames are already stored at two or more locations, only the changes need be transmitted from one location to another. Each location can use its existing copy of the frame data, without the need for transmitting such frame data from one location to another. Only the changes in operations to be performed on the frame data need be transmitted. The present invention thus avoids the needless and redundant transmission of frame data.

[0018] Accordingly, in various embodiments, the system of the present invention provides a mechanism by which multiple frames can be aggregated into pictures. This simplifies the process by which users can manage, view, and manipulate multiple images that have some logical relation to one another. Multi-frame pictures can also be reliably aggregated, with assurances of image integrity, even long after the individual images are captured. Such aggregation can be automatic, or it can be performed in response to user input specifying particular combinations of frames to be aggregated.

[0019] In addition, the system of the present invention provides a mechanism for allowing a picture to be modified, while enforcing the immutability of the constituent frames.

[0020] The present invention also provides additional advantages, as will be made apparent in the description provided herein. In one embodiment, certain metadata elements that are not essential to rendering can be selectively removed from frames. For example, in circumstances where an originator of an image wishes to remain anonymous, metadata elements that may identify of the originator can be obscured or deleted using the techniques of the present invention.

[0021] One skilled in the art will recognize that the technique for storing, manipulating, and transmitting light field picture described herein can be applied to other scenarios and conditions, and is not limited to the specific examples discussed herein. For example, the system of the present invention is not limited to light field pictures, but can also be applied to pictures taken by conventional cameras and other imaging devices, whether or not such pictures contain light field data.

BRIEF DESCRIPTION OF THE DRAWINGS

[0022] The accompanying drawings illustrate several embodiments of the invention and, together with the description, serve to explain the principles of the invention according to the embodiments. One skilled in the art will recognize

that the particular embodiments illustrated in the drawings are merely exemplary, and are not intended to limit the scope of the present invention.

[0023] Fig. 1A depicts an architecture for implementing the present invention in a camera, according to one embodiment.

[0024] Fig. 1B depicts an architecture for implementing the present invention wherein a camera collects image data from other cameras, according to one embodiment.

[0025] Fig. 1C depicts an architecture for implementing the present invention wherein a computing device collects image data from cameras, according to one embodiment.

[0026] Fig. 1D depicts an architecture for implementing the present invention in a client/server environment, according to one embodiment.

[0027] Fig. 1E depicts an architecture for implementing the present invention in a standalone computing device, according to one embodiment.

[0028] Fig. 2 depicts an example of a relationship between pictures and frames, according to one embodiment.

[0029] Fig. 3 depicts an example of a data structure for a frame, according to one embodiment.

[0030] Fig. 4A depicts a conceptual representation of digest generation, according to one embodiment.

[0031] Fig. 4B depicts an example of a relationship between pictures, frames, and digests, according to one embodiment.

[0032] Fig. 5 depicts an example of generation of a final image from components of a picture, according to one embodiment.

[0033] Fig. 6 depicts an example of the use of derived frames in generating a final image from components of a picture, according to one embodiment.

[0034] Fig. 7 depicts an example of generation of a high-dynamic range final image from frames having different exposure lengths, according to one embodiment.

[0035] Fig. 8 depicts an example of a mechanism for BLOB storage according to one embodiment.

[0036] Fig. 9 is a flowchart depicting a method of generating and sharing frames and/or digests, according to one embodiment.

[0037] Fig. 10 is a flowchart depicting a method of generating and storing a picture file specifying a combination of frames, according to one embodiment.

[0038] Fig. 11 is a flowchart depicting a method of rendering and outputting a final image based on a stored picture file, according to one embodiment.

DETAILED DESCRIPTION OF THE EMBODIMENTS

Terminology

[0039] The following terms are defined for purposes of the description provided herein:

- **Capture event:** the capture of a set of frames that are related. An example of a capture event on a single camera is sequential-flash relighting. Capture events can also involve multiple cameras, each capturing one or more frames.
- **Frame:** a data entity (stored, for example, in a file) containing a description of the state corresponding to a single captured sensor exposure in a camera. This state includes the sensor image, and other relevant camera parameters, specified as metadata. The sensor image may be either a raw image or a compressed representation of the raw image.
- **Digest:** a result of a hash function applied to a frame. In one embodiment, each digest is a 2-tuple: a first entry identifies a hash function, and a second entry provides the result of the identified hash function applied to a frame. As described below, digests provide a mechanism for authenticating frames.

- **Framed picture:** a picture that includes frame data for all of its frames. (No frame in a framed picture is represented by its digest.)
- **Immutable:** in one embodiment, the term “immutable” is used herein to refer to data that is not easily changeable by an ordinary user of the device without such a change being detected. It is possible, however, that a system administrator or other authorized individual may be able to make changes to such data, under certain circumstances.
- **Light field:** a collection of rays. A ray's direction specifies a path taken by light, and its color specifies the radiance of light following that path.
- **Light field frame:** a frame whose sensor image is a light field image.
- **Light field image:** a two-dimensional image that spatially encodes a four-dimensional light field. The sensor image from a light field camera is a light field image.
- **Light field picture:** a picture with one or more light field frames. (A picture with a mix of two-dimensional and light field frames is a light field picture.)
- **Microlens:** a small lens.
- **Picture:** a data entity (stored, for example, in a file) containing one or more frames, metadata, and/or data derived from the frames and/or metadata. Metadata can include tags, edit lists, and/or any other descriptive information or state associated with a picture or frame.
- **Pixel:** an n -tuple of intensity values, with an implied meaning for each value. A typical 3-tuple pixel format is RGB, wherein the first value is red intensity, the second green intensity, and the third blue intensity. Also refers to an individual sensor element for capturing data for a pixel.
- **Raw image:** an image (for example in Bayer format) that has not been subjected to any lossy compression since it was captured at a sensor.
- **Raw frame:** a frame in which the sensor image is a raw image.

- **Sensor image:** any representation of a raw image.
- **Two dimensional (2-D) frames:** a frame whose sensor image is a two-dimensional (2-D) image.
- **Two dimensional (2-D) image (or image):** a two-dimensional (2-D) array of pixels. The pixels are typically arranged in a square or rectangular Cartesian pattern, but other patterns are possible.
- **Two dimensional (2-D) picture:** a picture that includes only 2-D frames.

[0040] In addition, for ease of nomenclature, the term “camera” is used herein to refer to an image capture device or other data acquisition device. Such a data acquisition device can be any device or system for acquiring, recording, measuring, estimating, determining and/or computing data representative of a scene, including but not limited to two-dimensional image data, three-dimensional image data, and/or light field data. Such a data acquisition device may include optics, sensors, and image processing electronics for acquiring data representative of a scene, using techniques that are well known in the art. One skilled in the art will recognize that many types of data acquisition devices can be used in connection with the present invention, and that the invention is not limited to cameras. Thus, the use of the term “camera” herein is intended to be illustrative and exemplary, but should not be considered to limit the scope of the invention. Specifically, any use of such term herein should be considered to refer to any suitable data acquisition device.

System Architecture

[0041] Referring now to Fig. 1A, there is shown an architecture for implementing the present invention in a light field camera 100A according to one embodiment, although one skilled in the art will recognize that a 2-D camera can also be used. Camera 100A includes optics 111 (such as one or more lenses) for receiving and focusing an image; image sensor(s) 112, located in the optical image path for capturing the image as electronic data; and image processing

circuitry 113 for processing the image data to generate data file(s) representing picture(s) 201 and/or frame(s) 202. In the embodiment of Fig. 1A, picture(s) 201 and/or frame(s) 202 are stored in local data storage 101 at camera 100A; however, in other embodiments, such data can be transmitted, output, and/or stored elsewhere. Data storage 101 may be implemented as any magnetic, optical, and/or electrical storage device for storage of data in digital form, such as flash memory, magnetic hard drive, CD-ROM, and/or the like.

[0042] In one embodiment, optics 111, image sensor(s) 112, and/or image processing circuitry 113 operate as described in related U.S. Utility Application Serial No. 12/703,367 for “Light Field Camera Image, File and Configuration Data, and Method of Using, Storing and Communicating Same.”

[0043] Input device 108 receives input from user 110; such input may include commands for editing, deleting, transmitting, combining, and/or otherwise manipulating pictures(s) 201 and/or frame(s) 202. As described in more detail below, in one embodiment the system of the present invention does not permit user 110 to modify individual frames 202, but does allow user 110 to modify pictures 201, so as to allow image editing while preserving the immutability of component frames 202. In general, input device 108 can receive input provided via physical buttons, touchscreen, rocker switch, dial, knob, graphical user interface, or any other suitable mechanism, or any combination thereof. In one embodiment, user 110 operations on pictures 201 are stored as instructions, as described in more detail.

[0044] Image renderer 502 renders final image(s) 107 from picture(s) 201 and/or frame(s) 202, using instructions based on user input. Such rendering can be immediate or deferred. Output device 106 present final image(s) 107 to user 110. Output device 106 may be an integrated display screen on camera 100A, a separate display screen (not shown), and/or any other suitable device. Final image(s) 107 may include a representation of picture(s) 201.

[0045] In one embodiment, in response to commands presented by user 110 via input device 108, camera 100A combines any number of frames 202 to generate one or more picture(s) 201, according to the techniques described herein. In one embodiment, such combination is performed by generating instructions specifying various types of operations to be performed on frames 202, including for example combining frames 202 with one another.

[0046] In one embodiment, both frames 202 and pictures 201 are implemented as container files. As described below, such instructions can be stored for later application by image renderer 502, thus providing a mechanism for deferred aggregation.

[0047] Frames 202 may represent image data and metadata captured by camera 100A and/or by other data acquisition devices. Frames 202 may represent image data captured during a single capture event, or they may represent image data captured at different capture events. In this manner, the system of the present invention provides a mechanism for taking data representing multiple images taken at (potentially) multiple points in time at (potentially) multiple cameras or other devices, performing any desired operations on the data (for example to transform images), and/or combining the data into a picture 201 that is viewable by user 110.

[0048] In some embodiments, camera 100A may include components (not shown) for transmitting picture(s) 201, frame(s) 202, and/or final image(s) 107 in any suitable data format and according to any suitable data transmission protocol. For example, such data can be transmitted via the Internet, or via a wired or wireless communications network, personal area network such as BlueTooth, cellular telephone network, pager network, or the like. For illustrative purposes, however, Fig. 1A depicts an architecture for implementing the present invention in a stand-alone arrangement, without reference to other components.

[0049] In one embodiment, hash module 120 may transform frame(s) 202 into digest(s) 402 by applying a hash function, according to techniques described herein. Such transformation can save storage space and bandwidth in transmitting image data from one component to another, and can provide an authentication mechanism as described herein. Hash module 120 may be used for authenticating previously generated digest(s) 402 by comparing hash results with previously stored digest(s) 402. In various embodiments, hash module 120 may be implemented as software running on a processor, or as a part of image renderer 502, or as a separate component, or in any other suitable fashion. In other embodiments, hash module 120 may be omitted.

[0050] Referring now to Fig. 1B, there is shown an alternative architecture in which camera 100A collects image data from other cameras 100B, 100C, according to one embodiment. For illustrative purposes, camera 100B is shown as a two-dimensional camera, while cameras 100A, 100C are shown as light field cameras; however, any combination of two-dimensional and/or light field cameras can be used. For clarity, many components of cameras 100B, 100C are omitted from Fig. 1B.

[0051] Each camera 100B, 100C contains local data storage 101, which may include image data 301 and/or metadata 302 for images captured at that camera 100B, 100C. In the architecture depicted in Fig. 1B, cameras 100B, 100C transfer their image data 301 and/or metadata 302 to camera 100A, which collects and stores such data as frame(s) 202 in local data storage 101. Transfer for data from cameras 100B, 100C to camera 100A can take place according to any suitable method. For example, in one embodiment, cameras 100B, 100C, and/or 100A can be connected to one another via USB cable or other wired connection; in another embodiment such communication can take place via Bluetooth or another personal area network; in yet another embodiment, such communication can take place via any other wired or wireless communications network.

[0052] In one embodiment, camera 100A performs operations on data from frame(s) 202, including combining data from frame(s) 202 into one or more picture(s) 201, as described above and as set forth in more detail herein. As discussed above, such combination can take place in response to commands provided by user 110 via input device 108. Image renderer 502 renders final image(s) 107; final image(s) can be presented via output device 106, or otherwise transmitted, stored, and/or manipulated. Other components of camera 100A may operate in the same manner as described above in connection with Fig. 1A. Again, optional hash module 120 is also shown, for generating digest(s) 402 from frame(s) 202, and for authenticating previously generated digest(s) 402.

[0053] Referring now to Fig. 1C, there is shown an alternative architecture in which image data 301 and/or metadata 302 are collected by computing device 105. User 110 interacts with computing device 105 via input device 108, which may include a mouse, trackpad, trackball, keyboard, and/or any of the other input components mentioned above. Computing device 105 may be any electronic device, including for example and without limitation, a desktop computer, laptop computer, personal digital assistant (PDA), cellular telephone, smartphone, music player, handheld computer, tablet computer, kiosk, game system, enterprise computing system, server computer, or the like. In one embodiment, computing device 105 runs an operating system such as for example: Linux; Microsoft Windows, available from Microsoft Corporation of Redmond, Washington; Mac OS X, available from Apple Inc. of Cupertino, California; iOS, available from Apple Inc. of Cupertino, California; and/or any other operating system that is adapted for use on such devices.

[0054] Computing device 105 stores picture(s) 201 and/or frame(s) 202 in data storage 104, which may be located locally or remotely with respect to computing device 105. Data storage 104 may be implemented as any magnetic, optical, and/or electrical storage device for storage of data in digital form, such as flash memory, magnetic hard drive, CD-ROM, and/or the like. Data storage

104 can also be implemented remotely, for example at a server (not shown in Fig. 1C). Here, hash module 120 is shown as part of computing device 105; however, hash module 120 can instead be implemented in one or more of cameras 100 or in any other suitable component.

[0055] In one embodiment, computing device 105 includes a number of hardware components as are well known to those skilled in the art. In addition to data storage 104, input device 108 and output device 106, computing device 105 may include, for example, one or more processors 111 (which can be a conventional microprocessor for performing operations on data under the direction of software, according to well-known techniques) and memory 112 (such as random-access memory having a structure and architecture as are known in the art, for use by the one or more processors in the course of running software). Such components are well known in the art of computing architecture.

[0056] Referring now to Fig. 1D, there is shown an alternative architecture for implementing the present invention in a client/server environment, according to one embodiment. Image data 301 and/or metadata 302 are collected by server 109. Image data 301 and/or metadata 302 may be transmitted to server 109 via any suitable mechanism; one example is communications network 103 such as the Internet. In such an embodiment, image data 301 and/or metadata 302 can be transmitted using HTTP and/or any other suitable data transfer protocol. Client computing device 105 is communicatively coupled with server 109 via communications network 103.

[0057] User 110 interacts with computing device 105 via input device 108, which may include a mouse, trackpad, trackball, keyboard, and/or any of the other input components mentioned above. Under the direction of input device 108, computing device 105 transmits corresponding request to cause data to be transmitted, received, and/or manipulated by server 109. For example, server 109 can receive image data 301 and/or metadata 302 from cameras 100A, 100B, and store such data as frame(s) 202 in centralized data storage 104. In response to

requests received from client computing device 105, server 109 can perform operations on data, including combining frames 202 to generate picture(s) 201; such picture(s) 201 can then be transmitted to client computing device 105 for viewing on output device 106 as final image(s) 107. As described above, client computing device 105 can also store, transmit, or otherwise manipulate final image(s) 107.

[0058] In various embodiments, client computing device 105 may be any electronic device, including for example and without limitation, a desktop computer, laptop computer, personal digital assistant (PDA), cellular telephone, smartphone, music player, handheld computer, tablet computer, kiosk, game system, enterprise computing system, server computer, or the like. In one embodiment, computing device 105 runs an operating system such as for example: Linux; Microsoft Windows, available from Microsoft Corporation of Redmond, Washington; Mac OS X, available from Apple Inc. of Cupertino, California; iOS, available from Apple Inc. of Cupertino, California; and/or any other operating system that is adapted for use on such devices. In one embodiment, client computing device 105 includes a network interface (not shown) for enabling communication via network 103, and may also include browser software (not shown) for transmitting requests to server 109 and receiving responses therefrom. In this manner, user 110 can edit, delete, modify, manage, and otherwise manipulate data stored at server 109.

[0059] Centralized data storage 104 is shown as being located at server 109, although it can also be implemented locally at computing device 105 or at any suitable location. Providing centralized data storage 104 at server 109 allows pictures(s) 201 and frame(s) 202 to be accessed by a number of computing devices 105 in different locations, using a client/server architecture. However, such architecture is not necessary to practice the present invention. In one embodiment, computing device 105 communicates with server 109 via communications network 103 according to well known protocols.

[0060] In one embodiment, each of a number of cameras 100 generates digests 402 for its frames 202, and immediately shares these digests 402 with other cameras 100 participating in the capture event. Thus, each camera 100 has a “complete” version of a picture 201 representing the capture event, in the sense that each frame 202 for the capture event is represented either by its digest 402 or by the actual frame data 202. Subsequently, (such as, for example, when a suitable transmission path becomes available), each camera 100 transfers actual frame data 202 to central data server 109. Frame data 202 can then be retrieved from server 109 when any of the pictures 201 are to be viewed. In one embodiment, frame data 202 can be retrieved ahead of time, so as to improve response time in viewing a picture 201.

[0061] Referring now to Fig. 1E, there is shown an alternative architecture for implementing the present invention in a standalone computing device 105, according to one embodiment. Here, picture(s) 201 and frame(s) 202 may represent images previously collected from camera(s) and/or other data acquisition devices, and stored at data storage 104 of computing device 105. Fig. 1E illustrates that the techniques of the present invention can be practiced on image data without requiring the source of the image data to be present. Thus, for example, user 110 can manipulate frame(s) 202 to generate picture(s) 201 by interacting with computing device 105 via input device 108, causing processor 111 to perform operations on frame(s) 202 and generate picture(s) 201. As described above, individual frame(s) 202 can be taken from different cameras and/or at different times.

[0062] In one embodiment, computing device 105 includes the same components as described above in connection with Fig. 1C.

Data Structures

[0063] Referring now to Fig. 2, there is shown an example of a relationship between pictures 201 and frames 202, according to one embodiment. Frames 202 can be generated by cameras 100 and/or other visual data acquisition devices;

each frame 202 includes data related to an individual image element such as an image captured by a camera 100 or other visual data acquisition device. Any number of frames 202 can be combined to form a picture 201. For example a picture 201 may include frames 202 captured by different cameras 100 either simultaneously or in succession, and/or may include frames 202 captured by a single camera 100 in succession. Frames 202 may be captured as part of a single capture event or as part of multiple capture events. Pictures 201 may include any type of frames 202, in any combination including for example two-dimensional frames 202, light field frames 202, and the like. A picture 201 with one or more light field frames 202 is referred to as a light field picture.

[0064] In one embodiment, each frame 202 includes data representing an image detected by the sensor of the camera (image data), and may also include data describing other relevant camera parameters (metadata), such as for example, camera settings such as zoom and exposure time, the geometry of a microlens array used in capturing a light field frame, and the like. The image data contained in each frame 202 may be provided in any suitable format, such as for example a raw image or a lossy compression of the raw image, such as for example, a file in JPG, EXIF, BMP, PNG, PDF, TIFF and/or HD Photo format. The metadata may be provided in text format, XML, or in any other suitable format.

[0065] For illustrative purposes, in Fig. 2, frames 202 are shown as being enclosed by pictures 201. However, one skilled in the art will recognize that such a representation is conceptual only. In fact, in one embodiment, pictures 201 are related to their constituent frame(s) 202 by virtue of pointers in database records. In one embodiment, any particular frame 202 can be a constituent of any number of pictures 201, depending on how many pictures 201 contain a pointer to that frame 202. Similarly, any particular picture 201 can contain any number of frames 202, depending on how many frames 202 are identified as its constituents in its database record. In another embodiment, picture 201 may be a container

file that actually contains frame(s) 202. In general, references herein to a picture 201 “containing” one or more frames 202 mean that those frames 202 are associated with picture 201.

[0066] In one embodiment, if a frame 202 appears in more than one picture 201, it need only be stored once. Pointers are stored to establish relationships between the frame 202 and the various pictures 201 it corresponds to. Furthermore, if frame 202 data is not available, frame 202 can be represented by its corresponding digest 402, as described herein.

[0067] Accordingly, the system of the present invention provides a mechanism for storing and manipulating pictures 201 having an arbitrary number of frames 202. A collection of frames 202 may be designated as constituents of a particular picture 201 based on any appropriate criteria. In one embodiment, the mode of the capture event may define the relationship between a set of frames 202 and thus cause such frames 202 to be constituents of a common picture 201. For example, a photographer may capture a series of images using exposure bracketing. As another example, a set of images having different lighting characteristics can be captured, so that a final image can be generated from the captured images using interpolation or some other form of combination; such a process is referred to as “relighting”. In one embodiment, an indication that bracketing or relighting is being used may cause the frames 202 captured in a sequence to be associated with a common picture 201. Such an indication may be provided, for example, in metadata for frame(s) 202. However, one skilled in the art will recognize that any assignment of frames to pictures is allowed, whether specified automatically or manually, and that such assignment can be taken at the time of image capture (i.e., immediately after a capture event) or at any subsequent time. Thus, the system of the present invention allows pictures 201 having arbitrary sets of frames 202 to be created. In one embodiment, post-capture editing tools, such as image manipulation

software, can be used for manually specifying such relationships between pictures 201 and frames 202.

[0068] Referring now to Fig. 3, there is shown an example of a data structure for a frame 202, according to one embodiment. In one embodiment, frame 202 includes image data 301 and/or metadata 302, although some frames 202 may omit one or the other. In various embodiments, frames 202 can include image data 301 for two-dimensional and/or light field sensor images. In other embodiments, other types of image data 301 can be included in frames 202, such as three-dimensional image data and the like. In one embodiment, a depth map of the scene is extracted from the light field, so that three-dimensional scene data can be obtained and used. In another embodiment, a camera can capture a two-dimensional image, and use a range finder to capture a depth map; such captured information can be stored as frame data, so that the two-dimensional image and the depth map together form a three-dimensional image.

[0069] In one embodiment, metadata 302 includes fields for various parameters associated with image data 301, such as for example camera settings such as zoom and exposure time, the geometry of a microlens array used in capturing a light field frame, and the like.

[0070] In one embodiment, frames 202 are stored in frame files. Frame files may contain additional data, for example to enable operating systems and/or applications to work with frames 202. Examples of such additional data include: a file format identifier, to allow applications to recognize frame files; and thumbnail image, in a format that allows an operating system to display a thumbnail as a representation of the file. For ease of nomenclature, the term “frame” is used herein to refer both to frames and to their stored representations.

[0071] In one embodiment, metadata 302 may include identifying data, such as a serial number of the camera or other device used to capture the image, an identifier of the individual photographer operating the camera, the location where the image was captured, and/or the like. In some situations, user 110 or

some other individual may not want such information to be readily available to anyone interacting with frame 202. As described below, in one embodiment, such identifying data, as well as any other data that is not critical to the interpretation of image data 301, may be provided in a removable section of metadata 302. Thus, user 110, or an administrator or other individual, can specify that such data be removed or suppressed, so as to preserve privacy.

[0072] Metadata 302 can be provided in any appropriate format, such as for example a human-readable text file including name-value pairs. In one embodiment, metadata 302 is represented using name-value pairs in JavaScript Object Notation (JSON). In one embodiment, metadata 302 is editable by user 110 or any other individual having access to frame 202. In one embodiment, metadata 302 is provided in XML or text format, so that any text editor can be used for such editing.

Digests

[0073] In one embodiment, frames 202 can be represented as digests 402. Referring now to Fig. 4A, there is shown a conceptual representation of generating a digest 402 from a frame 202. A hash function 401 is defined, for generating a unique digest 402 for each frame 202. Hash function 401 can be any mathematical function that operates on the digital data that represents frame 202, to generate output that is unique for that frame 202, or output that is extremely unlikely to be identical for any two frames 202. In one embodiment, digests 402 are small relative to their corresponding frames 202, so that transmission, storage, and manipulation of such digests 402 is faster and more efficient than such operations would be on the frames 202 themselves. For example, in one embodiment, each digest is 256 bytes in length, although one skilled in the art will recognize that they may be of any length. A digest can also be referred to as a "hash".

[0074] Digests 402 are substantially unique with respect to one another, so that any attempt to modify data of a frame 202 can be detected by performing

hash function 401 on frame 202 data, since the resulting digest 402 would fail to match any previously stored digest 402 for that frame 202. In this manner, the system of the present invention is able to enforce immutability of digests 402 in an efficient manner. Computation of digests 402 by hash function 401 can be performed at camera 100 or any data acquisition device, or at computing device 105, or at server 109. Computation of digests 402 can take place at the time of image capture or at any subsequent time. Verification of image integrity can be performed at any suitable time by recalculating digest 402 from frame 202 data and comparing the result with a previously stored version of digest 402; any difference indicates that the underlying frame 202 has been altered.

[0075] In one embodiment, hash module 120 performs hash function 401 on frame 202 including image data and metadata 302. Hash module 120 may be implemented in software on camera 100, computing device 105, and/or server 109, and may be part of image renderer 502 or any other component, or may be a separate component. By performing hash function 501 on image data and metadata 302, the system of the present invention ensures that changes to either image data and/or metadata 302 can be detected. In this manner, hash function 401 is used to enforce immutability of both image data and metadata 302.

[0076] In one embodiment, hash function 401 is designed so that non-substantive changes to metadata 302 do not cause changes to the resultant digest 402. For example, addition or deletion of whitespace, addition or deletion of comments, reordering of parameters, and the like, do not affect the value of digest 402. In this manner, frame immutability is enforced without hindering the ability to make non-substantive changes that may improve readability but do not change the operation of metadata 302. In one embodiment, hash function 401 ignores any removable sections that are not critical to image rendering, so that such sections can be removed or included without affecting digest 402.

[0077] Referring now to Fig. 4B, there is shown an example of a relationship between pictures 201, frames 202, and digests 402, according to one

embodiment. Here, one frame 202 is represented by the frame data itself, while two other frames are represented by digests 402. Any combination of frames 202 and/or digests 402 can be included in any particular picture 201. A picture 201 that includes frames 202 but no digests 402 is referred to as a framed picture. Thus, once actual frame data is available and stored within a picture 201, the picture 201 becomes a framed picture which can be rendered without reference to additional data. Hash module 120 can be used to generate digests 402, as well as to authenticate digests 402.

[0078] In one embodiment, data acquisition devices such as cameras 100 can share digests 402 rather than frames 202, while retaining their own frames 202 locally, so as to save bandwidth and/or storage space. Once digests 402 for a picture 201 are shared among cameras 100, each camera 100 has a complete version of picture 201; specifically, each frame 202 in picture 201 is represented either by frame data itself (as captured on that camera 100) or as a digest 402 of the frame 202 (which was computed, for example, by the camera 100 that captured that frame 202).

[0079] Subsequently, after pictures 201 from various cameras 100 have themselves been communicated to a central repository such as server 109, or are otherwise able to share data, final images 107 can be generated from pictures 201, using actual frame 202 data rather than digests 402.

[0080] In one embodiment, digests 402 help to ensure the integrity of frame 202 data. Since digests 402 can be recomputed from frames 202 at any time, corrupted and/or modified frames 202 can easily be detected.

[0081] As described above, in one embodiment, user 110 can manipulate pictures 201 by interacting with computing device 105 and/or camera 100. Input device(s) 108 may be provided for receiving input from user 110. In one embodiment, a user interface can be provided to enable user 110 to manipulate pictures 201 in various ways. For example, user 110 can specify various operations to be performed on constituent frames 202 of pictures 201, including

brightening, darkening, scaling, cropping, rotating, blurring, changing color balance, and the like. Users 110 can also specify that frames 202 are to be combined various ways, and specify relationships between individual frames 202.

[0082] However, in one embodiment, such manipulations are not stored as changed frame 202 data itself. Rather, they are stored as representations of operations to be performed on frames 202, so that the operations can be re-executed whenever picture 201 is to be rendered. Thus, instead of storing the result of an operation, in one embodiment the system of the present invention stores a representation of the operation itself. Underlying frames 202 can thus be left untouched.

[0083] In one embodiment, such operations are stored within the picture file 201 itself. Pictures 201 thus have the ability to store a recipe (instructions) for creating a particular final image, the ingredients (frames 202) for the recipe are maintained in their original form. Such an approach preserves the integrity of the original captured images and associated metadata (within frames 202), since such original frames 202 can always be retrieved no matter what operations are specified to be performed on such frames 202 in connection with a particular picture 201.

[0084] Referring now to Fig. 5, there is shown a block diagram depicting generation of a final image 107 from components of a picture 201, according to one embodiment. Frame 202 represents image data. Digests 402 are generated from frames 202, such as frames 202 captured at other cameras 100. Instructions 501 are representations of modifications and other operations user 110 has performed on frames 202 of picture 201.

[0085] Image renderer 502 (which may be implemented on computing device 105, camera 100, or any other component), generates final image 107 for display on output device 106, or for storage or transmission. When generating image 107, renderer 502 uses data stored in picture 201, including frames 202 and

instructions 501. Specifically, renderer 502 performs operations on frames 202 based on instructions 501. Prior to rendering final image 107, original frame(s) 202 can be obtained for any stored digest(s) 402, and the authenticity of such frame(s) 202 can be confirmed by applying hash function 401 and comparing the result to stored digest(s) 402. In one embodiment, image renderer 502 is implemented as a software application.

[0086] In one embodiment, final image 107 does not contain information describing the frames 202 and instructions 501 that were used in its creation. Rather, final image 107 is generated in any conventional image format, including for example RAW, TIFF, JPG, GIF, or the like. In other embodiments, however, some or all of the data describing frames 202 and instructions 501 may be included as metadata or in another format.

[0087] For example, instructions 501 may include a list of user-specified edits to displayed images, such as cropping, rotating, changing contrast, refocusing, and the like. Instructions 501 may indicate operations on a single frame 202 or on any combination of frames 202. In some cases, a series of instructions 501 may be applied in a specified order, so that the result of one instruction 501 is passed to the next instruction 501. In the example of Fig. 5, instructions 501A, 501B, 501C are performed on various frames 202, and then instructions 501D operate to combine the respective results of instructions 501A, 501B, 501C. When generating image 107, renderer 502 applies such instructions 501 to image data stored in frame 202, in the specified order, to generate final image(s) 107. In one embodiment, such steps are performed each time the image is to be displayed or rendered. In one embodiment, such operations include post-processing as described in related U.S. Utility Application Serial No. 12/703,367 for "Light Field Camera Image, File and Configuration Data, and Method of Using, Storing and Communicating Same," (Atty. Docket No. LYT3003), filed February 10, 2010, the disclosure of which is incorporated herein by reference.

[0088] In one embodiment, a user interface can be provided, so that user 110 can interactively specify parameters for operations to be performed; for example, a slider can be provided for controlling refocusing operations.

[0089] In one embodiment, rendering performance is improved by including acceleration structures within pictures. Such acceleration structures may include metadata that can be applied to frames 202. Such metadata can be derived from frames 202 and/or user-supplied metadata.

[0090] For example, in one embodiment, the system can store frames that are derived from application of specified operations. Referring now to Fig. 6, there is shown an example of the use of such an acceleration structure, referred to as derived frames 601. Derived frames 601 can be generated by applying instructions 501 to frames 202 and storing the result or a digest 402 of the result. Storage of derived frames 601 can improve efficiency, since image renderer 502 need not regenerate the frame each time final image 107 is to be rendered. Rather, derived frames 601 from a previous rendering operation can be retrieved. Derived frames 601 can thus be used as an intermediate storage mechanism so as to avoid redundant recalculation.

[0091] In one embodiment, derived frames 601 are generated by image renderer 502 and stored as components of picture 201. Digests 402 of derived frames 601 can be generated, stored, shared, and/or transmitted in a manner similar to that described above for digests 402 of frames 202. In one embodiment, derived frames 601 and/or their digests 402 can be stored in a cache (not shown) for rapid access.

[0092] In one embodiment, derived frames 601 can be regenerated when instructions 501 change, so that set of available derived frames 601 is kept current.

[0093] In one embodiment, generation of derived frames 601 is triggered manually in response to user 110 commands. In another embodiment, the system of the present invention automatically generates derived frames 601 when certain

defined circumstances arise; for example, when it is determined that generation of a derived frame 601 is likely to be useful and storage space is available. A determination that generation of a derived frame 601 is likely to be useful may depend, for example, on detection that certain instructions 501 are being repeatedly applied to the same frame(s) 202. In one embodiment, a user 110 or other entity can specify whether or not acceleration structures such as derived frames 601 should be used for a particular picture 201 or frame 202. In one embodiment, the specification as to whether an acceleration structure should be used may depend on the characteristics of the destination renderer 502; for example, a renderer 502 with superior processing power may not need an acceleration structure, whereas one which has inferior processing power (and/or is burdened by other concurrent tasks) may use acceleration structures. As another example, the available of cache storage space may determine whether acceleration structures such as derived frames 601 should be used.

[0094] In the example of Fig. 6, two derived frames 601 are shown, for illustrative purposes. Derived frame 601B is generated from frame 202A and is stored within picture 201. Derived frame 601A was generated based on some instructions not part of picture 201; for example, derived frame 601A may have been provided by some other device. Thus, picture 201 includes digest 402A of derived frame rather than derived frame 601A itself.

[0095] In this example, when picture 201 is to be rendered, the system of the present invention retrieves derived frame 601A and checks it against digest 402A. Then, image renderer 502 applies instructions as indicated: instructions 501B are applied to derived frame 601A and instructions 501C are applied to digest 402B. Instructions 501A need not be performed, since derived frame 601B is already available and contains the results of application of instructions 501A to frame 202A. Instructions 501D are applied to the results of instructions 501B and 501C in combination with derived frame 601B, to generate final image 107 for output.

[0096] Note that the example of Fig. 6 illustrates the application of instructions 502 to frames 202 as well as derived frames 601. In general, any number of instructions can be applied in succession to one or more frame(s) 502, derived frame(s) 601, and/or combination(s) thereof.

[0097] Derived frames 601 may include metadata, if appropriate, in the same manner as frames 602. In one embodiment, each derived frame 601 includes a pointer 602 or other identifier indicating the frame 202 from which it was derived. This allows a mechanism by which the original frame 202 to be retrieved. This pointer 602 can be provided as metadata within derived frame 601. Thus, for example, if user 110 wishes to perform an operation on the original frame 202, and only a derived frame 601 is available, the system of the present invention can refer to pointer 602 within derived frame 601 to locate and retrieve the corresponding original frame 202 so that it can be edited, modified, and more manipulated as appropriate.

[0098] Additional forms of acceleration structures can be used. For example, if the frame data are light field images, a stack of output images can be computed, each with a different focus distance. These images can be stored, so that a picture 201 can later be “refocused” by selecting one of the pre-computed images. If user 110 selects an apparent focal distance for which no image was pre-computed, such as a distance between two stored distances, interpolation and/or extrapolation can be used to generate an appropriate resultant image in an efficient manner.

[0099] The use of digests improves the efficiency with which frames 202 can be aggregated into pictures 201, particularly when such aggregation is deferred (i.e., takes place substantially later than image capture). For example, in situations where frames 202 from different cameras 100 or other data acquisition devices are to be aggregated with one another, embodiments of the present invention make such transfer of information describing such frames 202 more efficient. Digests 402 can be shared among the various devices, so that actual

image data need not be repeatedly transferred. A device can refer to frame 202 data by its corresponding digest 402, with the assurance that when actual frame 202 data arrives, it is authentic. The authenticity of frame 202 data can be assured by recomputing digest 402 from frame 202 data, and comparing the result with stored digest 402. Any discrepancy indicates that frame 202 has been modified or corrupted.

[0100] An example of a data structure employing digests is as follows:

```
"picture" : {  
  frame: {  
    image: blobref,  
    metadata: blobref,  
    privateMetadata: blobref  
  }  
}
```

[0101] Here, a picture 201 is defined as having a single frame 202. Frame 202 image data, metadata, and private metadata, each indicated as a "blobref". In one embodiment, each "blobref" includes a hash type (i.e., an identification of the hash function) and hash data, forming the digest 402. For example, a blobref might be "SHA1-2fd4e1c67a2d28fced849ee1bb76e7391b93eb12"; "sha1" indicates the hash type, and the remainder of the data is the hash data itself. The hash data can be of any suitable length, such as for example 40 hexadecimal digits.

[0102] The blobref refers to a binary large object (BLOB), according to well known techniques; this reference can be de-referenced by a blob server (described below), which returns the BLOB that the blobref represents.

[0103] In this example, frame 201 separates private metadata from other metadata. For example, metadata that might identify the photographer, location, and/or equipment can be omitted by deleting the privateMetadata blobref from frame 202.

Archival Support

[0104] In one embodiment, the user interface for enabling user 110 to manipulate pictures 201 allows various operations to be performed on pictures

201; however, such operations do not change the underlying frame 202 data itself. In one embodiment, the user interface does not provide a way for user 110 to delete frames 202; in another embodiment, such an operation is possible but only for certain authenticated users 110 and/or only under special conditions. Even if a picture 201 is deleted, in one embodiment its component frames 202 remain stored.

[0105] By making deletion of frames 202 difficult or impossible, the system of the present invention encourages persistence of underlying data and thereby provides support for archival functions. In addition, the techniques of using digests 402, as described above, ensure authenticity of frames 202, by providing a way for modifications or corruption of frames 202 to be detected. Such assurances of authenticity also help to support archival functions.

[0106] Furthermore, embodiments of the present invention allow frames 202 to be recombined in different ways to form different pictures 201, without affecting frames 202 themselves. In this manner, users 110 are free to perform such recombination while archival functions are supported.

Example: HDR imaging

[0107] Referring now to Fig. 7, there is shown an example of generation of a high-dynamic range final image 107 from frames 202 having different exposure lengths, according to one embodiment. High-dynamic range (HDR) imaging is a technique that combines two or more images having different exposure lengths so as to generate an image having a higher dynamic range than do the source images. HDR imaging thus provides a way to overcome the limited dynamic range available in some image sensors.

[0108] In Fig. 7, the techniques of the present invention are applied to generate an HDR final image 107. Optics 111, image sensor(s) 112, and image processing circuitry 113 of camera 110 generate two or more frames 202 having different exposure lengths. In the example, two frames 202 are shown: short-exposure frame 202B and long-exposure frame 202C. Frames 202B, 202C are

shown as being captured by a single camera 100; alternatively, they may come from different cameras or other data acquisition devices.

[0109] According to the techniques of the present invention, frames 202B, 202C are represented in picture 201. Metadata (not shown) may also be included in frames 202B, 202C, for example to identify the exposure lengths and other information about the images represented therein. Instructions 501A specify how frames 202B, 202C are to be combined to generate HDR final image 107, for example using HDR processing techniques to extract various portions of images depending on contrast levels.

[0110] In the example, HDR final image 107 is generated by image renderer 502 running at computing device 105. However, in other examples, image renderer 502 could be located at camera 100 or at any other location. In the example, renderer 502 combines frames 202B, 202C based on instructions 501A. In one embodiment, a user interface can be presented to allow the user to specify parameters for the rendering operation; for example a slider control (not shown) can be presented to allow user 110 to specify the overall contrast level of HDR final image 107.

[0111] In other examples, the representations of frames 202B, 202C within picture 201 may be replaced by digests (not shown in Fig. 7) according to the techniques described above.

JavaScript Object Notation (JSON)

[0112] In one embodiment, metadata for the present invention is stored using a JavaScript Object Notation (JSON) format, which provides a standardized text notation for objects. JSON is sufficiently robust to provide representations according to the techniques described herein, including objects, arrays, and hierarchies. JSON further provides a mechanism which is easy for humans to read, write, and understand.

[0113] One example of a generalized format for a JSON representation of an object is as follows:

```

object    ::= {} | { members }
members  ::= pair | pair , members
pair     ::= string : value
array    ::= [] | [ elements ]
elements ::= value | value , elements
value    ::= string | number | object | array |
           true | false | null

string   ::= "" | "chars"
number  ::= int | intFrac | intExp | intFracExp

```

[0114] Thus, the JSON representation can be used to store frame metadata in a key-value pair structure.

[0115] As described above, frame metadata may contain information describing the camera that captured an image. An example of a portion of such a representation in JSON is as follows:

```

"camera" :
{
  "make" : "Canon",
  "model" : "EOS 50D"
  "firmware" : "2.0.34 beta"
}

```

[0116] Data stored in the JSON representation may include integers, floating point values, strings, Boolean values, and any other suitable forms of data, and/or any combination thereof.

[0117] Given such a structure, a software application (such as image renderer 502) can access data in a frame 202 by performing a key lookup, and/or by traversing or iterating over the data structure, using known techniques. In this manner, image renderer 502 can use the data found within frames 202 and pictures 201 when generating final image(s) 107 and/or when combining frames 202 into pictures 201.

[0118] The JSON representation may also include structures; for example a value may itself contain a list of values, forming a hierarchy of nested key-value pair mappings. For example:

```

"key1" :
{
  "key2" : {
    "key3": [2.12891, 1.0, 1.29492]
  }
}

```

}

[0119] In one embodiment, binary data is stored in the JSON structure via a base64-encoding scheme.

[0120] Privacy concerns are addressed as described above. Identifying data, as well as any other data that is not critical to the interpretation of image data, may be provided in a removable section of metadata, for example in a separate section of the JSON representation. This section can be deleted without affecting image rendering operations, since the data contained therein is not used for such operations. An example of such a section is as follows:

```
"removable" : {
  "serial" : "520323552",
  "gps" : { ... },
  ...
}
```

[0121] Data to be used in rendering images may be included in any number of separate sections. These may include any or all of the following:

- a description section, providing a general description of the equipment used (without specific identifying information)
- an image section, containing image data;
- a devices section, specifying settings and parameters for the equipment used;
- a light field section, containing light field data (if the frame contains a light field image);

[0122] One skilled in the art will recognize that these are merely exemplary, and that any number of such sections can be provided.

[0123] **Description section** can contain any information generally describing the equipment used to capture the image. An example of a description section is as follows:

```
"camera" :
{
  "make" : "Canon",
  "model" : "EOS 50D"
  "firmware" : "2.0.34 beta"
}
```

[0124] **Image section** contains image data. Image section can contain color-related fields for converting raw images to RGB format. Image section can contain a "format" value indicating whether the format of the image is "raw" or "rgb". In addition, various other fields can be provided to indicate what corrections and/or other operations were performed on the captured image.

[0125] An example of an image section is as follows:

```
"image" : {
  "timeStamp" : "2009:07:04 03:00:46 GMT",
  "orientation" : 1,
  "width" : 4752,
  "height" : 3168,
  "format" : "raw",
  "raw" : {
    "mosaic" : {
      "type" : "r,g;g,b",
      "firstPixelMosaicIndex" : { "x" : 0, "y" : 1 } },
    "pixelRange" : { "black" : 1024, "white" : 15763 },
    "pixelFormat" : { "bpp" : 16, "endian" : "little", "shift"
: 4 } },
    "whiteBalanceMultipliers" : [2.12891, 1, 1.29492],
    "ccmRgbToSrgb" : [2.26064, -1.48416, 0.223518, -0.100973,
1.59904,
                                -0.498071, 0.0106269, -
0.58439, 1.57376] },
    "gamma" : [0, 1, 2, 4, 6, 9, ..., 4050, 4070, 4092]
  }
}
```

[0126] **Devices section** specifies camera hardware and/or settings; for example, lens manufacturer and model, exposure settings, and the like. In one embodiment, this section is used to break out information for component parts of the camera that may be considered to be individual devices. An example is as follows:

```
"devices" : {
  "lens" : {
    "make" : "Tamron",
    "macro" : true,
    "model" : "xxx",
    "focalLength"
: 50,
    "fNumber" : 4,
    "motorPosition"
: { "zoom" : 200, "focus" : 120 } },
    "flash" : {
      "make" : "Canon",
      "firmware" : "beta",
      "model" : "xxx",
      "brightness" :
2.3,
      "duration" : 0.1 },
    "ndfilter" : { "stops" : 3.0 },
  }
}
```

```

        "sensor" : {
            "exposureDuration" : 0.1,          "iso" : 400,
            "analogGain" : 34.0,                "digitalGain" :
1.0 }
        "accelerometer" : { "samples" : [ ... ] }
    }

```

[0127] **Light field** section provides data relating to light fields, image refocusing, and the like. Such data is relevant if the image is a light field image. An example is as follows:

```

"lightfield" : {
    "index" : 1,
    "mla" : {
        "type" : "hexRowMajor",
    "pitch" : 51.12,
        "scale" : { "x" : 1, "y" : 1 },
    "rotation" : 0.002319,
        "sensorOffset" : { "x" : -15.275, "y" : -44.65, "z" : 200
    },
        "defects" : [ { "x" : 1, "y" : 3}, { "x" : 28, "y" : 35} ]
    },
    "sensor" : { "pixelPitch" : 4.7 },
    "lens" : { "exitPupilOffset" : { "x" : 0.0, "y" : 0.0, "z" :
57.5 } },
    "intensityFunction" : {
        "dimensions" : { "x" : 44, "y" : 32, "u" : 13, "v" : 13 },
        "jpegDataStream" : "/9j/4AAQSkZJRgABAQAAQABAAD/DBAQE
..... mJFZ//Z" }
    }
}

```

[0128] In one embodiment, the “defects” key refers to a set of (x,y) tuples indicating defective pixels. Such information can be useful in generating images, as defective pixels can be ignored, recomputed from adjacent pixels, down-weighted, or otherwise processed. One skilled in the art will recognize that various techniques for dealing with such defects can be used. If a concern exists that the specific locations of defects can uniquely identify a camera, raising privacy issues, the “defects” values can be omitted or can be kept hidden so that they are not exposed to unauthorized users.

[0129] Frame digests are supported by the JSON data structure. As described above, a digest can be stored as both a hash type and hash data. The following is an example of a digest within the removable section of a JSON data structure:

```

"removable" : {

```

```

    "serial" : "520323552",
    "gps" : { ... },
    "digest" : {
        "type" : "sha1",
        "hash" : " 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12" }
    }

```

[0130] In various embodiments, metadata (such as JSON data structures) can be included in a file separate from the image itself. Thus, one file contains the image data (for example, img_0021.jpg, img_0021.dng, img_0021.raw, or the like), and another file in the same directory contains the JSON metadata (for example, img_0021.txt). In one embodiment, the files can be related to one another by a common filename (other than the extension) and/or by being located in the same directory.

[0131] Alternatively, the image data and the metadata can be stored in a single file. For example, the JSON data structure can be included in an ancillary tag according to the exchangeable image file format (EXIF), or it can be appended to the end of the image file. Alternatively, a file format can be defined to include both image data and metadata.

Example of JSON Specification

[0132] The following is an example of a JSON specification for picture files according to one embodiment. One skilled in the art will recognize that this example is intended to be illustrative only, and that many other variables, formats, arrangements, and syntaxes can be used without departing from the essential characteristics of the present invention, as defined in the claims.

```

MASTER_OBJ ::=
{
    "version" :
    {
        "major" : INT,                //
initially 1, incremented for incompatible versions
        "minor" : INT                //
initially 0, incremented for forward-compatible versions, reset when
major is incremented
        "provisionalDate" : STRING,    // to be
included only if the JSON is formatted to a provisional version
                                         // string
format is iso8601: "yyyy-mm-dd"
    },

```

```

        "thumbnailArray" : [ THUMBNAI_OBJ ],           // may be
empty.
        "picture" :                                     //
optional.
        {
            "derivationArray" : [ PICTURE_ID ],         //
required. e.g., great-grandfather, grandfather, father, current
            "frameArray" : [ FRAME_OBJ ],               // may be
empty.
            "viewArray" : [ VIEW_OBJ ],                 // may be
empty.
            "accelerationArray" : [ ACCELERATION_OBJ ]   // may be
empty.
        }
    }

```

```

THUMBNAI_OBJ ::=
{
    "imageRef" : BLOBREF,                               // the
blobref, the http, or the inline image must be present
    "imageUrl" : URL,
    "image" : INLINE_IMAGE_OBJ,
    "representation" : IMAGE_REPRESENTATION_ENUM,
    "width" : PIXEL,
    "height" : PIXEL
}

```

```

INLINE_IMAGE_OBJ ::= BASE64_IMAGE_OBJ | ARRAY_IMAGE_OBJ

```

```

BASE64_IMAGE_OBJ ::=
{
    // base-64 encoding of a standard format, such as JSON
}

```

```

ARRAY_IMAGE_OBJ ::=
{
    "pixelFormat" : "rgb" | "depth",
    "values" : [ ARRAY_IMAGE_PIXEL ]
}

```

```

ARRAY_IMAGE_PIXEL ::= NUMBER | [ NUMBER ]             // either
a single NUMBER, or an array of two or more NUMBERS

```

```

IMAGE_REPRESENTATION_ENUM ::=
    "jpeg" |
    "array"

```

```

PICTURE_ID ::= STRING                                 // a hash
generated from camera sn, picture index, and nonce

```

```

FRAME_OBJ ::=
{
    "frame" :
    {
        "imageRef" : BLOBREF,                //
required.    "image" : INLINE_IMAGE_OBJ,    //
optional, but blobref must be specified in either case.
        "metadataRef" : BLOBREF,            //
required.    "metadata" : FRAME_METADATA_OBJ, //
optional, but blobref must be specified in either case.
        "privateMetadataRef" : BLOBREF,     //
optional.    "privateMetadata" : PRIVATE_FRAME_METADATA_OBJ, //
optional, but cannot be present if blobref is not present.
    },
    "parameters" : FRAME_PARAMETER_OBJ,
    "accelerationArray" : [ ACCELERATION_OBJ ] // may be
empty.
}

```

```

FRAME_METADATA_OBJ ::=
{
    "type" : "2d" | "lightField",
    "camera" :
    {
        "make" : STRING,                    // e.g.,
"Lytro"
        "model" : STRING,                  // e.g.,
"Firefly 1.0"
        "firmware" : STRING                // e.g.,
"2.0.34"
    }
    "modes" :
    {
        "autoRotate" : BOOLEAN,            //
optional.    "defaultCameraTurns" : TURN, //
optional.    "selectedCameraTurns" : TURN, // camera
turns, CCW when viewing the display
    }
    "image" :
    {
        "width" : PIXEL,                  // e.g.,
3280
        "height" : PIXEL                  // e.g.,
3280
        "representation" : FRAME_IMAGE_REPRESENTATION_ENUM,
        "rawDetails" :                     //
required for representations "jpegRaw" and "raw", not present otherwise
    {
        "mosaic" :                         //
describes the pattern of pixels in the raw image
    }
}

```

```

        "tile" : SENSOR_TILE_ENUM, // name of
tile pattern
        "upperLeftPixel" : SENSOR_TILE_PIXEL_ENUM, // name of
pixel in tile pattern that is upper left in the image
    }
    "cropFromSensor" : //
optional
    {
        "left" : INT, // number
of columns of pixels removed from the left side of the sensor image
        "right" : INT, // number
of columns of pixels removed from the right side of the sensor image
        "top" : INT, // number
of rows of pixels removed from the top of the sensor image
        "bottom" : INT // number
of rows of pixels removed from the bottom of the sensor image
    },
    "pixelPacking" : // how
individual pixels are packed into the raw data (see discussion below)
    {
        "endianness" : "little" | "big", // data
are a byte stream. Little endian means LSBs come before MSBs. Big
means the opposite.
        "bitsPerPixel" : BIT // bits
per pixel in the raw stream. Greater than or equal to sensor
bitsPerPixel.
    },
    "pixelFormat" : // how to
interpret the value of an individual pixel (after it is unpacked from
the raw data)
    {
        "rightShift" : BIT, // shift
required to right-justify the valid bitsPerPixel (may not be negative)
        "black" : INT, // post-
shift black value (values below this are negative values, due to noise)
        "white" : INT, // post-
shift white value (values above this are special reserved values)
        "defect" : INT //
optional. Post-shift value for defective pixels
    }
},
"color" :
{
    "mode" : "bakedIntoFrame" | "recommended",
    "whiteBalanceGain" : //
optional if "recommended"
    {
        "r" : NUMBER,
        "g" : NUMBER,
        "b" : NUMBER
    },
    "ccmRgbToSrgbArray" : [ NUMBER ], //
optional if "recommended", 3x3 array, row major, premultiplies color
vector (r,g,b)T
    "cctKelvin" : NUMBER, //
optional (never baked in), can look-up in XYZ, then convert to sRGB if
required

```

```

        "gamma" : NUMBER //
optional if "recommended"
    }
    "devices"
    {
        "accelerometer" : //
optional.
        {
            "make" : STRING, //
optional.
            "model" : STRING, //
optional.
            "firmware" : STRING, //
optional.
            "sampleArray" : [ ACCELEROMETER_SAMPLE_OBJ ] // must
not be empty.
        },
        "clock" :
        {
            "make" : STRING, //
optional.
            "model" : STRING, //
optional.
            "firmware" : STRING, //
optional.
            "zuluTime" : STRING // ISO
8601, e.g., "2011-03-30T18:07:25.134Z", fraction to millisecond, Zulu
time (no local offset)
            "localTime" : STRING //
optional, e.g., "2011-03-30T10:07:25.134", fraction to millisecond, no
trailing "Z"
        },
        "gps" : //
optional.
        {
            "make" : STRING, //
optional.
            "model" : STRING, //
optional.
            "firmware" : STRING //
optional.
        },
        "flashArray" : [ FLASH_OBJ ], // may be
empty.
        "lens" :
        {
            "make" : STRING, //
optional.
            "model" : STRING, //
optional.
            "firmware" : STRING, //
optional.
            "temperature" : CELSIUS, //
optional.
            "focalLength" : METER, // focal
length of the lens (taking zoom into account)

```

```

        "focusDistance" : METER,                                // object-
coordinate focus distance of the lens to the MLA
        "macro" : BOOLEAN,                                        // true if
in macro mode (focusDistance is less than its usual function of
focalLength)
        "infinityLambda" : LAMBDA,                              //
distance ahead of the MLA that is in focus at infinity (along optical
axis)
        "zoomStep" : INT,                                        //
optional, zoom stepper-motor position
        "focusStep" : INT,                                      //
optional, focus stepper-motor position
        "entranceFNumber" : NUMBER,                             // ratio
of focal length to the diameter of the entrance pupil (traditional f-
number)
        "exitFNumber" : NUMBER,                                  // ratio
of focal length to the diameter of the exit pupil
        "exitPupilOffset" :
        {
            "x" : METER,
            "y" : METER,
            "z" : METER
        }
    },
    "mla" :
    {
        "make" : STRING,                                         //
optional.
        "model" : STRING,                                         //
optional.
        "tiling" : MLA_PACKING_ENUM,
        "rotation" : RADIAN,                                     //
positive values indicate CW rotation, viewed from the optical axis in
front of the camera
        "lensPitch" : METER,
        "focalLength" : METER,
        "scaleFactor" :                                         //
optional.
        {
            "x" : NUMBER,
            "y" : NUMBER
        },
        "sensorOffset" :
        {
            "x" : METER,
            "y" : METER,
            "z" : METER
        },
        "defectArray" : [ MLA_DEFECT_OBJ ]                       // may be
empty.
    }
    "ndfilter" :
    {
        "make" : STRING,                                         //
optional.
        "model" : STRING,                                         //
optional.

```

```

        "stops" : NUMBER //
effective light blockage for this frame
    },
    "sensor" :
    {
        "make" : STRING, //
optional.
        "model" : STRING, //
optional.
        "firmware" : STRING, //
optional.
        "iso" : INT,
        "analogGain" : //
optional.
        {
            "r" : NUMBER,
            "gr" : NUMBER,
            "gb" : NUMBER,
            "b" : NUMBER
        },
        "digitalGain" : //
optional.
        {
            "r" : NUMBER,
            "gr" : NUMBER,
            "gb" : NUMBER,
            "b" : NUMBER
        },
        "mosaic" :
        {
            "tile" : SENSOR_TILE_ENUM, // name of
tile pattern
            "upperLeftPixel" : SENSOR_TILE_PIXEL_ENUM, // name of
pixel in tile pattern that is upper left
        },
        "bitsPerPixel" : BIT,
        "pixelPitch" : METER
    },
    "shutter" :
    {
        "make" : STRING, //
optional.
        "model" : STRING, //
optional.
        "mechanism" : SHUTTER_MECHANISM_ENUM,
        "pixelExposureDuration" : SECOND, //
approximate exposure time of an individual pixel
        "frameExposureDuration" : SECOND //
approximate exposure time from start of first-pixel exposure to end of
last-pixel exposure
    }
    "soc" :
    {
        "make" : STRING, //
optional.
        "model" : STRING, //
optional.

```

```

        "firmware" : STRING,                                //
optional.        "temperature" : CELSIUS                    //
optional.        }
                }
            }

FRAME_IMAGE_REPRESENTATION_ENUM ::=
    "jpeg" |
    "jpegRaw" |
    "dng" |
    "raw"

FLASH_OBJ ::=
{
    "make" : STRING,                                        //
optional.        "model" : STRING,                          //
optional.        "firmware" : STRING,                        //
optional.        "brightness" : NUMBER,
                "duration" : SECOND
    }

ACCELEROMETER_SAMPLE_OBJ ::=
{
    "x" : G,                                                //
    acceleration in the camera's positive x direction (right)
    "y" : G,                                                //
    acceleration in the camera's positive y direction (up)
    "z" : G,                                                //
    acceleration in the camera's positive z direction (backward)
    "time" : SECOND                                         //
    measured from clock value (negative before, positive after)
    }

SHUTTER_MECHANISM_ENUM ::=
    "sensorRolling" |                                     // perhaps
required for a very short shutter duration
    "sensorOpenApertureClose" |                           // most
common
    "apertureOpenClose"                                   // would
this ever be used?

SENSOR_TILE_ENUM ::=
    "r,gr:gb,b"                                           //
standard Bayer tile

SENSOR_TILE_PIXEL_ENUM ::=
    "r" |
    "gr" |
    "gb" |

```

```

    "b"

    MLA_PACKING_ENUM ::=
        "squareUniform" |
        "hexUniformRowMajor" |
        "hexUniformColumnMajor"

    MLA_DEFECT_OBJ ::=
    {
        "x" : MLA_COORD, //
increases left-to-right
        "y" : MLA_COORD, //
increases top-to-bottom in square grid, 60 degrees down and to the right
in hex grid
        "type" : MLA_DEFECT_TYPE_ENUM,
    }

    PRIVATE_FRAME_METADATA_OBJ ::=
    {
        "derivationArray" : [ FRAME_REF_OBJ ], // may be
empty, e.g., great-grandfather, grandfather, father (no current, it's
available elsewhere)
        "camera" :
        {
            "serialNumber" : STRING
        },
        "devices" :
        {
            "gps" : //
optional.
        {
            "latitudeDeg" : DEGREES, // decimal
fraction, not minutes and seconds
            "longitudeDeg" : DEGREES // decimal
fraction, not minutes and seconds
        }
        }
    }

    FRAME_REF_OBJ ::= //
blobrefs only, no actual data
    {
        "imageRef" : BLOBREF, // must be
present.
        "metadataRef" : BLOBREF, // must be
present.
        "privateMetadataRef" : BLOBREF, // present
unless private data have been stripped.
        "description" : STRING //
optional.
    }

    FRAME_PARAMETER_OBJ ::=

```

```

    {
      "vendorContent" : VENDOR_FRAME_PARAMETER_OBJ
    }

VIEW_OBJ ::=
{
  "vendorContent" : VENDOR_VIEW_OBJ
}

ACCELERATION_OBJ ::=
{
  "type" : VENDOR_STRING, //
  Uniquely identifies the type of this acceleration object
  "generator" : VENDOR_STRING, //
  optional, software system used to compute acceleration data (e.g.,
  images)
  "keep" : BOOLEAN, //
  optional, presumed false if not present
  "vendorContent" : VENDOR_ACCELERATION_OBJ
}

```

[0133] In various embodiments, any number of extensions can be made to the JSON specification for picture files; these may be provided, for example, for certain types of equipment or vendors according to one embodiment.

Binary Large Object (BLOB) Storage

[0134] In one embodiment, frame and/or picture data is stored as binary large objects (BLOBs). "Blobrefs" can be used as wrappers for such BLOBs; each blobref holds or refers to a BLOB. As described above, blobrefs can contain hash type and hash data, so as to facilitate authentication of data stored in BLOBs.

[0135] Referring now to Fig. 8, there is shown an example of a mechanism for BLOB storage according to one embodiment. Front end 802 receives commands 801 such as put(blobref, blob) and get(blobref) (which returns a BLOB). Front end 802 may be an application programming interface (API), for example, for receiving commands 801 from a browser, input device, or other source. Upon receiving a request to retrieve a BLOB, front end 802 may pass a search request to search server 803, which obtains index data from index storage 804 so as to identify the location of the requested BLOB. Front end 802 makes requests of the appropriate blob server(s) 505, based on the index data, to obtain

the requested BLOB. Blob servers 805 retrieve requested BLOBs from blob storage 806.

[0136] In one embodiment, Blob servers 805 communicate with one another to keep their data in sync, so as to avoid discrepancies in stored BLOBs. In addition, search server 803 may periodically communicate with one or more Blob servers 805 in order to update its index 804.

Method

[0137] Referring now to Fig. 9, there is shown a flowchart depicting a method of generating and sharing frames 202 and/or digests 402, according to one embodiment. Camera 100 or other data acquisition device captures 901 images and generates 902 frames 202 from the captured images. As described above frames 202 can contain metadata describing the equipment used in capturing the image, as well as removable identifying information and the like. Digests 402 are generated 903 by applying a hash function to frame data, as described above. Frames 202 are stored 903, either at camera 100 or at some central location such as server 109, another camera 100, computing device 105, or the like. Digests 402 and/or frames 202 are shared 905 among devices, if appropriate, to allow images captured in step 901 to be combined with images captured at other devices. As described above, in some embodiments digests 402 are shared so as to conserve bandwidth. Sharing 905 of frames 202 and/or digests 402 can include transmission over a computing network, as described above.

[0138] Referring now to Fig. 10, there is shown a flowchart depicting a method of generating and storing a picture 201 file specifying a combination of frames 202, according to one embodiment. In one embodiment, such an operation is performed in response to receiving input 1001, for example a command entered via input device 108, to combine images. Alternatively, such an operation can be performed automatically in response to a determination that two or more images are related (for example if it is determined that they are of

the same subject at approximately the same time). Picture 201 file is generated 1002, which can include any combination of frames 202 and/or digests 402. As specified above, picture 201 may include instructions for combining frames 202. Picture 201 file is stored 1003, for example in data storage 104 at a camera 100, computing device 105, server 109, or any other suitable location.

[0139] Once a picture 201 file has been generated and stored, a final image 107 can be generated from picture 201, using the image data and instructions contained therein. Such rendering can happen at any time, even long after picture 201 file has been generated.

[0140] Referring now to Fig. 11, there is shown a flowchart depicting a method of rendering and outputting a final image based on a stored picture 201 file, according to one embodiment. In one embodiment, such an operation is performed in response to receiving 1101 input, for example via input device 108, to render a final image 107. In one embodiment, the steps of Fig. 10 are performed by image renderer 502 in response to such input.

[0141] Picture file 1102 is retrieved. If picture 201 includes frames 202, the frames are retrieved for rendering. If picture 201 includes digests 402, the corresponding frames 202 are retrieved 1103, and can be authenticated by independent application 1104 of the hash function to frame 202 data, as described above. The hash result from such independent application is compared 1105 against stored digest 402 for authentication. If the retrieved frames 202 are authenticated, final image 107 is rendered 1106 based on data in picture file 201, including image data and/or instructions. Once final image 107 has been rendered, it is output 1104, for example on output device 106 such as a display screen.

[0142] In one embodiment, user input is received for controlling the rendering operation 1106. For example, user 110 may be able to control various parameters of the image combination, such as size, contrast, distortion, and the

like. Any type of suitable user interface may be provided for enabling such user control.

[0143] The present invention has been described in particular detail with respect to possible embodiments. Those of skill in the art will appreciate that the invention may be practiced in other embodiments. First, the particular naming of the components, capitalization of terms, the attributes, data structures, or any other programming or structural aspect is not mandatory or significant, and the mechanisms that implement the invention or its features may have different names, formats, or protocols. Further, the system may be implemented via a combination of hardware and software, as described, or entirely in hardware elements, or entirely in software elements. Also, the particular division of functionality between the various system components described herein is merely exemplary, and not mandatory; functions performed by a single system component may instead be performed by multiple components, and functions performed by multiple components may instead be performed by a single component.

[0144] In various embodiments, the present invention can be implemented as a system or a method for performing the above-described techniques, either singly or in any combination. In another embodiment, the present invention can be implemented as a computer program product comprising a nontransitory computer-readable storage medium and computer program code, encoded on the medium, for causing a processor in a computing device or other electronic device to perform the above-described techniques.

[0145] Reference in the specification to “one embodiment” or to “an embodiment” means that a particular feature, structure, or characteristic described in connection with the embodiments is included in at least one embodiment of the invention. The appearances of the phrase “in one

embodiment" in various places in the specification are not necessarily all referring to the same embodiment.

[0146] Some portions of the above are presented in terms of algorithms and symbolic representations of operations on data bits within a memory of a computing device. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps (instructions) leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic or optical signals capable of being stored, transferred, combined, compared and otherwise manipulated. It is convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. Furthermore, it is also convenient at times, to refer to certain arrangements of steps requiring physical manipulations of physical quantities as modules or code devices, without loss of generality.

[0147] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "displaying" or "determining" or the like, refer to the action and processes of a computer system, or similar electronic computing module and/or device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0148] Certain aspects of the present invention include process steps and instructions described herein in the form of an algorithm. It should be noted that

the process steps and instructions of the present invention can be embodied in software, firmware and/or hardware, and when embodied in software, can be downloaded to reside on and be operated from different platforms used by a variety of operating systems.

[0149] The present invention also relates to an apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computing device selectively activated or reconfigured by a computer program stored in the computing device. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, flash memory, solid state drives, magnetic or optical cards, application specific integrated circuits (ASICs), or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus. Further, the computing devices referred to herein may include a single processor or may be architectures employing multiple processor designs for increased computing capability.

[0150] The algorithms and displays presented herein are not inherently related to any particular computing device, virtualized system, or other apparatus. Various general-purpose systems may also be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will be apparent from the description provided herein. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the present invention as described herein, and any references above to specific

languages are provided for disclosure of enablement and best mode of the present invention.

[0151] Accordingly, in various embodiments, the present invention can be implemented as software, hardware, and/or other elements for controlling a computer system, computing device, or other electronic device, or any combination or plurality thereof. Such an electronic device can include, for example, a processor, an input device (such as a keyboard, mouse, touchpad, trackpad, joystick, trackball, microphone, and/or any combination thereof), an output device (such as a screen, speaker, and/or the like), memory, long-term storage (such as magnetic storage, optical storage, and/or the like), and/or network connectivity, according to techniques that are well known in the art. Such an electronic device may be portable or nonportable. Examples of electronic devices that may be used for implementing the invention include: a mobile phone, personal digital assistant, smartphone, kiosk, server computer, enterprise computing device, desktop computer, laptop computer, tablet computer, consumer electronic device, television, set-top box, or the like. An electronic device for implementing the present invention may use any operating system such as, for example: Linux; Microsoft Windows, available from Microsoft Corporation of Redmond, Washington; Mac OS X, available from Apple Inc. of Cupertino, California; iOS, available from Apple Inc. of Cupertino, California; and/or any other operating system that is adapted for use on the device.

[0152] While the invention has been described with respect to a limited number of embodiments, those skilled in the art, having benefit of the above description, will appreciate that other embodiments may be devised which do not depart from the scope of the present invention as described herein. In addition, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter.

Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the claims.

CLAIMS

What is claimed is:

- 1 1. A system for generating an image from component elements,
2 comprising:
3 a storage device, configured to store a container file associated with a
4 plurality of component elements, each component element
5 comprising:
6 image data representing a captured image; and
7 metadata for the image data;
8 wherein the container file is further associated with instructions for
9 combining the component elements with one another;
10 an image renderer, communicatively coupled to the storage device,
11 configured to render a viewable image by retrieving the stored
12 container file and performing at least one operation on at least a
13 subset of the component elements according to the instructions;
14 and
15 an output device, communicatively coupled to the image renderer, for
16 outputting the viewable image.
- 1 2. The system of claim 1, wherein the container file comprises a picture
2 file, and wherein each component element comprises a frame.
- 1 3. The system of claim 1, wherein the image renderer performs the at least
2 one operation by combining at least two of the component elements according to
3 the instructions.
- 1 4. The system of claim 1, wherein the image data represents an image
2 captured at a camera.

1 5. The system of claim 1, wherein the image data represents at least one
2 light field image.

1 6. The system of claim 1, further comprising:
2 at least one camera, configured to capture image data;

3 wherein at least one of the stored component elements comprises image
4 data captured by the camera.

1 7. The system of claim 1, wherein the instructions further specify
2 operations to be performed on at least one of the component elements to render
3 the viewable image.

1 8. The system of claim 1, wherein:
2 the component elements associated with the container file are immutable;
3 and
4 the instructions associated with the container file are mutable.

1 9. The system of claim 1, further comprising:
2 an input device, communicatively coupled to the storage device, for
3 receiving user input for modifying the viewable image;

4 and wherein, responsive to the user input:
5 the storage device is configured to store new instructions specifying
6 operations to be performed on at least one of the component
7 elements to render a modified viewable image;
8 the image renderer is configured to render the modified viewable image
9 by combining the component elements according to the new
10 instructions; and
11 the output device is configured to output the modified viewable image.

1 10. The system of claim 1, wherein the image data comprises light field
2 data.

1 11. The system of claim 1, wherein the metadata comprises a plurality of
2 name-value pairs.

1 12. The system of claim 1, wherein the metadata is represented using
2 JavaScript Object Notation.

1 13. The system of claim 1, wherein the metadata comprises at least one
2 selected from the group consisting of:

3 a description of equipment used to capture the captured image;
4 a description of settings used when capturing the captured image;
5 a location associated with the captured image;
6 at least one parameter associated with the image data; and
7 light field data associated with the image data.

1 14. The system of claim 1, wherein the metadata comprises at least one
2 removable metadata section.

1 15. The system of claim 1, further comprising:
2 a hash module, configured to apply a hash function to generate at least
3 one digest from at least one component element;

4 wherein:
5 the storage device is configured to store the at least one digest.

1 16. The system of claim 15, further comprising:
2 an authenticator, configured to authenticate a component element by
3 comparing a digest generated by the hash module from the

4 component element with at least one previously stored digest
5 associated with the component element.

1 17. The system of claim 1, further comprising:
2 a transmission device, communicatively coupled to the image renderer,
3 configured to transmit the viewable image.

1 18. The system of claim 1, wherein:
2 the image renderer is further configured to generate a derived component
3 element from at least one component element;
4 the storage device is further configured to store the derived component
5 element; and
6 the image renderer is further configured to render the viewable image by
7 performing at least one operation on at least a subset of the
8 derived component elements according to the instructions.

1 19. The system of claim 18, wherein the image renderer performs the at
2 least one operation by combining at least one derived component element with at
3 least one component element according to the instructions.

1 20. The system of claim 1, wherein the storage device, the image renderer,
2 and the output device are components of a camera, and wherein the camera
3 further comprises an image capture mechanism configured to capture the image
4 data.

1 21. The system of claim 20, wherein the camera further comprises a
2 communications interface configured to receive at least one component element
3 from a second camera.

1 22. The system of claim 1, wherein the storage device, the image renderer,
2 and the output device are components of a computing device.

1 23. The system of claim 22, wherein the computing device further
2 comprises a communications interface configured to receive at least one
3 component element from a camera.

1 24. The system of claim 22, wherein the computing device further
2 comprises a communications interface configured to receive at least one
3 component element via an electronic communications network.

1 25. The system of claim 1, wherein the storage device is implemented at a
2 server, and wherein the image renderer, and the output device are components of
3 a client computing device communicatively coupled to receive data from the
4 server.

1 26. The system of claim 1, wherein the plurality of component elements
2 comprises component elements captured from a plurality of image capture
3 devices.

1 27. The system of claim 1, wherein the plurality of component elements
2 comprises component elements shared between a plurality of devices
3 communicatively coupled with one another via an electronic communications
4 network.

1 28. The system of claim 1, further comprising:
2 an input device, communicatively coupled to the storage device,
3 configured to receive user input for modifying parameters for
4 rendering the viewable image; and
5 wherein the image renderer is configured to render the viewable image
6 according to the received user input.

1 29. A system for generating an image from a plurality of component
2 elements, comprising:
3 at least one image capture device, each image capture device comprising:
4 an image capture mechanism, configured to capture at least one
5 image;
6 a hash module, configured to apply a hash function to generate at
7 least one digest from at least one captured image;
8 a storage device, communicatively coupled to the image capture
9 mechanism and to the hash module, configured to store
10 the at least one captured image and the at least one
11 digest; and
12 a communications interface, configured to transmit the at least one
13 digest to at least one other device;
14 an image renderer, configured to render a viewable image from a plurality
15 of component elements by:
16 receiving a plurality of captured images;
17 authenticating at least one of the received images by applying a
18 hash module to the received image and comparing the
19 result with a previously stored digest; and
20 combining the plurality of captured images to generate a viewable
21 image; and
22 an output device, communicatively coupled to the image renderer, for
23 outputting the viewable image.

1 30. The system of claim 29, wherein the image renderer is associated with
2 a server, and wherein receiving the plurality of captured images comprises
3 receiving the captured images at the server.

1 31. The system of claim 29, wherein the image renderer is associated with
2 an image capture device, and wherein receiving the plurality of captured images
3 comprises receiving the captured images at the image capture device associated
4 with the image renderer.

1 32. The system of claim 29, wherein the image capture device is
2 configured to transmit the captured images to the server responsive to a trigger
3 event.

1 33. The system of claim 32, wherein the trigger event comprises
2 availability of sufficient bandwidth to transmit the captured images.

1 34. The system of claim 29, wherein each image capture device comprises
2 a camera.

1 35. The system of claim 29, wherein:
2 the storage device of each image capture device is further configured to
3 store metadata describing the at least one captured image;
4 the hash module generates the at least one digest from the captured image
5 and the metadata; and
6 the image renderer combines the plurality of captured images using at
7 least a subset of the metadata.

1 36. The system of claim 29, wherein the communications interface is
2 configured to transmit the at least one digest to at least one other device via an
3 electronic network.

1 37. The system of claim 29, wherein the image renderer is configured to
2 receive the plurality of captured images via an electronic network.

1 38. The system of claim 29, wherein the storage device is configured to
2 store a container file associated with a plurality of component elements, each
3 component element comprising:
4 image data representing a captured image; and
5 metadata for the image data;

6 wherein the container file is further associated with instructions for
7 combining the component elements with one another;

8 and wherein the image renderer is configured to retrieve the stored
9 container file and perform at least one operation on at least a subset of the
10 component elements according to the instructions.

1 39. A method for generating an image from component elements,
2 comprising:
3 storing a container file associated with a plurality of component elements,
4 each component element comprising:
5 image data representing a captured image; and
6 metadata for the image data;
7 wherein the container file is further associated with instructions for
8 combining the component elements with one another;
9 retrieving the stored container file;
10 performing at least one operation on at least a subset of the component
11 elements according to the instructions, to generate a viewable
12 image; and
13 outputting the viewable image an output device.

1 40. The method of claim 39, wherein the container file comprises a picture
2 file, and wherein each component element comprises a frame.

41. The method of claim 39, wherein performing at least one operation on
at least a subset of the component elements comprises combining at least two of
the component elements according to the instructions.

1 42. The method of claim 39, wherein the instructions further specify
2 operations to be performed on at least one of the component elements to render
3 the viewable image.

1 43. The method of claim 39, wherein:
2 the component elements associated with the container file are immutable;
3 and
4 the instructions associated with the container file are mutable.

1 44. The method of claim 39, further comprising:
2 receiving user input for modifying the viewable image;
3 and, responsive to the user input:
4 storing new instructions specifying operations to be performed on
5 at least one of the component elements to render a
6 modified viewable image;
7 rendering the modified viewable image by combining the
8 component elements according to the new instructions;
9 and
10 outputting the modified viewable image.

45. The method of claim 39, further comprising:
applying a hash function to generate at least one digest from at least one
component element; and
storing the at least one digest.

1 46. The method of claim 45, further comprising authenticating a
2 component element by comparing a digest generated by the hash module from
3 the component element with at least one previously stored digest associated with
4 the component element.

1 47. The method of claim 39, further comprising transmitting the viewable
2 image.

1 48. The method of claim 39, further comprising:
2 generating a derived component element from at least one component
3 element;
4 storing the derived component element; and
5 rendering the viewable image by performing at least one operation on at
6 least a subset of the derived component elements according to
7 the instructions.

1 49. The method of claim 39, further comprising:
2 receiving user input for modifying parameters for rendering the viewable
3 image; and
4 rendering the viewable image according to the received user input.

1 50. A method for generating an image from a plurality of component
2 elements, comprising:
3 at a first device, capturing at least one image;
4 storing the at least one captured image;
5 receiving at least one digest from at least one second device, each digest
6 having been generated by application of a hash function to a
7 remotely-captured image;
8 subsequently obtaining the at least one remotely-captured image;

9 authenticating the obtained remotely-captured image by applying a hash
10 module to the obtained remotely-captured image and
11 comparing the result with a previously received digest;
12 combining the obtained remotely-captured image with at least one image
13 captured at the first device, to generate a viewable image; and
14 outputting the viewable image.

1 51. The method of claim 50, wherein receiving at least one digest from at
2 least one second device comprises receiving at least one digest from a server.

1 52. The method of claim 50, further comprising:
2 applying a hash function to generate at least one digest from at least one
3 captured image; and
4 transmitting the generated digest to at least a second device.

1 53. The method of claim 52, further comprising transmitting at least one of
2 the captured images to a server.

1 54. The method of claim 53, wherein transmitting at least one of the
2 captured images to a server comprises transmitting at least one of the captured
3 image to the server responsive to a trigger event.

1 55. The method of claim 54, wherein the trigger event comprises
2 availability of sufficient bandwidth to transmit the at least one captured images.

1 56. The method of claim 50, further comprising:
2 storing metadata describing the at least one captured image; and
3 combining the obtained remotely-captured image with at least one image
4 captured at the first device comprises combining the obtained
5 remotely-captured image with at least one image captured at the
6 first device using at least a subset of the metadata.

1 57. A computer program product for generating an image from
2 component elements, comprising:
3 a nontransitory computer-readable storage medium; and
4 computer program code, encoded on the medium, configured to cause at
5 least one processor to perform the steps of:
6 storing a container file associated with a plurality of component
7 elements, each component element comprising:
8 image data representing a captured image; and
9 metadata for the image data;
10 wherein the container file is further associated with
11 instructions for combining the component
12 elements with one another;
13 retrieving the stored container file;
14 performing at least one operation on at least a subset of the
15 component elements according to the instructions, to
16 generate a viewable image; and
17 causing an output device to output the viewable image.

1 58. The computer program product of claim 57, wherein the container file
2 comprises a picture file, and wherein each component element comprises a
3 frame.

1 59. The computer program product of claim 57, wherein the computer
2 program code configured to cause at least one processor to perform at least one
3 operation on at least a subset of the component elements comprises computer
4 program code configured to cause at least one processor to combine at least two
5 of the component elements according to the instructions.

1 60. The computer program product of claim 57, wherein the instructions
2 further specify operations to be performed on at least one of the component
3 elements to render the viewable image.

1 61. The computer program product of claim 57, wherein:
2 the component elements associated with the container file are immutable;
3 and
4 the instructions associated with the container file are mutable.

1 62. The computer program product of claim 57, further comprising:
2 computer program code configured to cause at least one input device to
3 receive user input for modifying the viewable image;
4 computer program code configured to cause at least one processor to,
5 responsive to the user input, perform the steps of:
6 causing a storage device to store new instructions specifying
7 operations to be performed on at least one of the
8 component elements to render a modified viewable
9 image;
10 rendering the modified viewable image by combining the
11 component elements according to the new instructions;
12 and
13 causing an output device to output the modified viewable image.

1 63. The computer program product of claim 57, further comprising
2 computer program code configured to cause at least one processor to perform the
3 steps of:
4 applying a hash function to generate at least one digest from at least one
5 component element; and
6 storing the at least one digest.

1 64. The computer program product of claim 63, further comprising
2 computer program code configured to cause at least one processor to perform the
3 step of authenticating a component element by comparing a digest generated by
4 the hash module from the component element with at least one previously stored
5 digest associated with the component element.

1 65. The computer program product of claim 57, further comprising
2 computer program code configured to cause at least one processor to perform the
3 steps of:

4 generating a derived component element from at least one component
5 element;

6 causing a storage device to store the derived component element; and

7 rendering the viewable image by performing at least one operation on at
8 least a subset of the derived component elements according to
9 the instructions.

1 66. A computer program product for generating an image from a plurality
2 of component elements, comprising:

3 a nontransitory computer-readable storage medium; and

4 computer program code, encoded on the medium, configured to cause at
5 least one processor at a first device to perform the steps of:

6 capturing at least one image;

7 causing a storage device to store the at least one captured image;

8 receiving at least one digest from at least one second device, each

9 digest having been generated by application of a hash

10 function to a remotely-captured image;

11 subsequently obtaining the at least one remotely-captured image;

12 authenticating the obtained remotely-captured image by applying a

13 hash module to the obtained remotely-captured image

14 and comparing the result with a previously received
15 digest;
16 combining the obtained remotely-captured image with at least one
17 image captured at the first device, to generate a viewable
18 image; and
19 causing an output device to output the viewable image.

1 67. The computer program product of claim 66, wherein the computer
2 program code configured to cause at least one processor to receive at least one
3 digest from at least one second device comprises computer program code
4 configured to cause at least one processor to receive at least one digest from a
5 server.

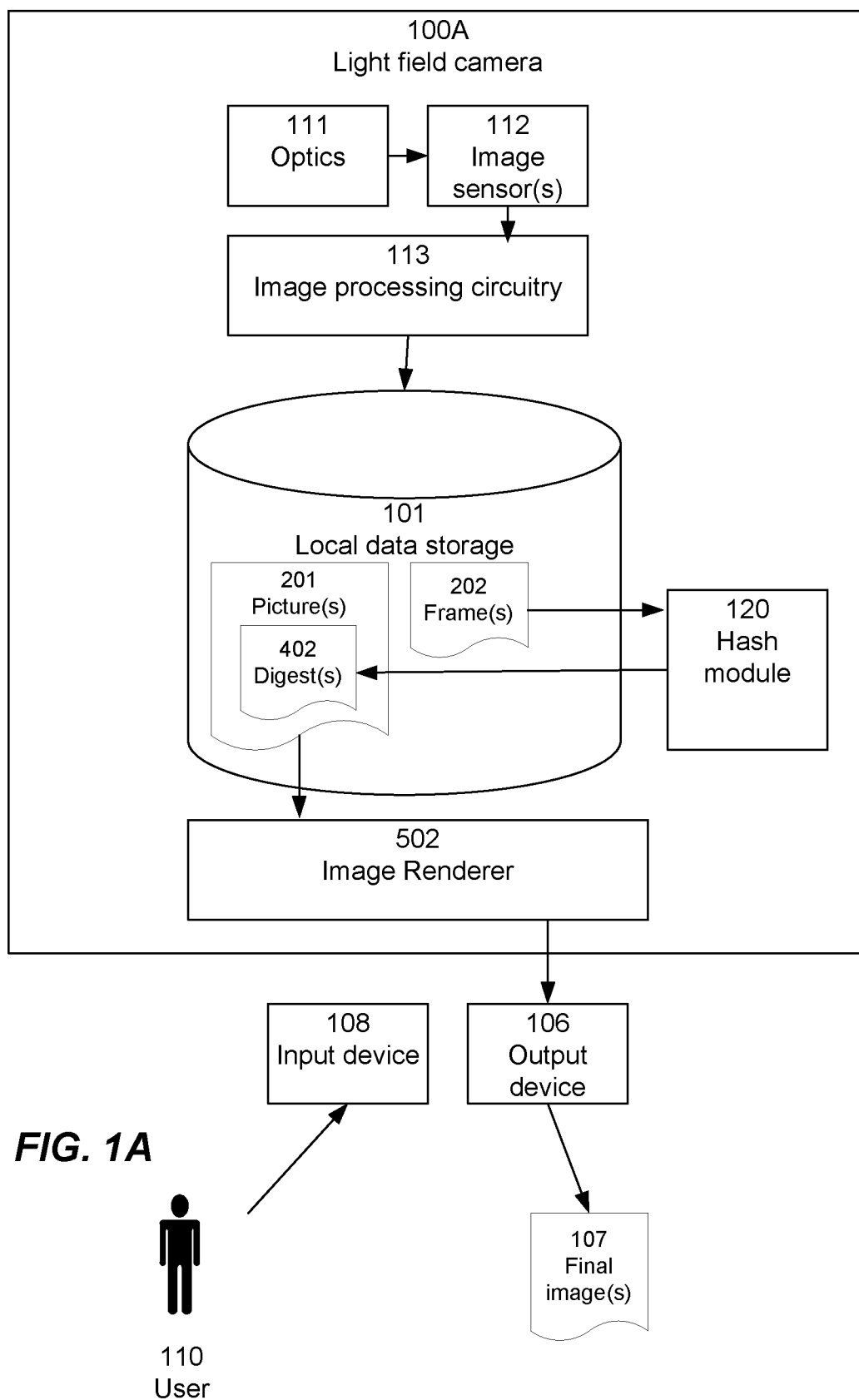
1 68. The computer program product of claim 66, further comprising
2 computer program code configured to cause at least one processor to perform the
3 steps of:
4 applying a hash function to generate at least one digest from at least one
5 captured image; and
6 transmitting the generated digest to at least a second device.

1 69. The computer program product of claim 68, further comprising
2 computer program code configured to cause at least one processor to transmit at
3 least one of the captured images to a server.

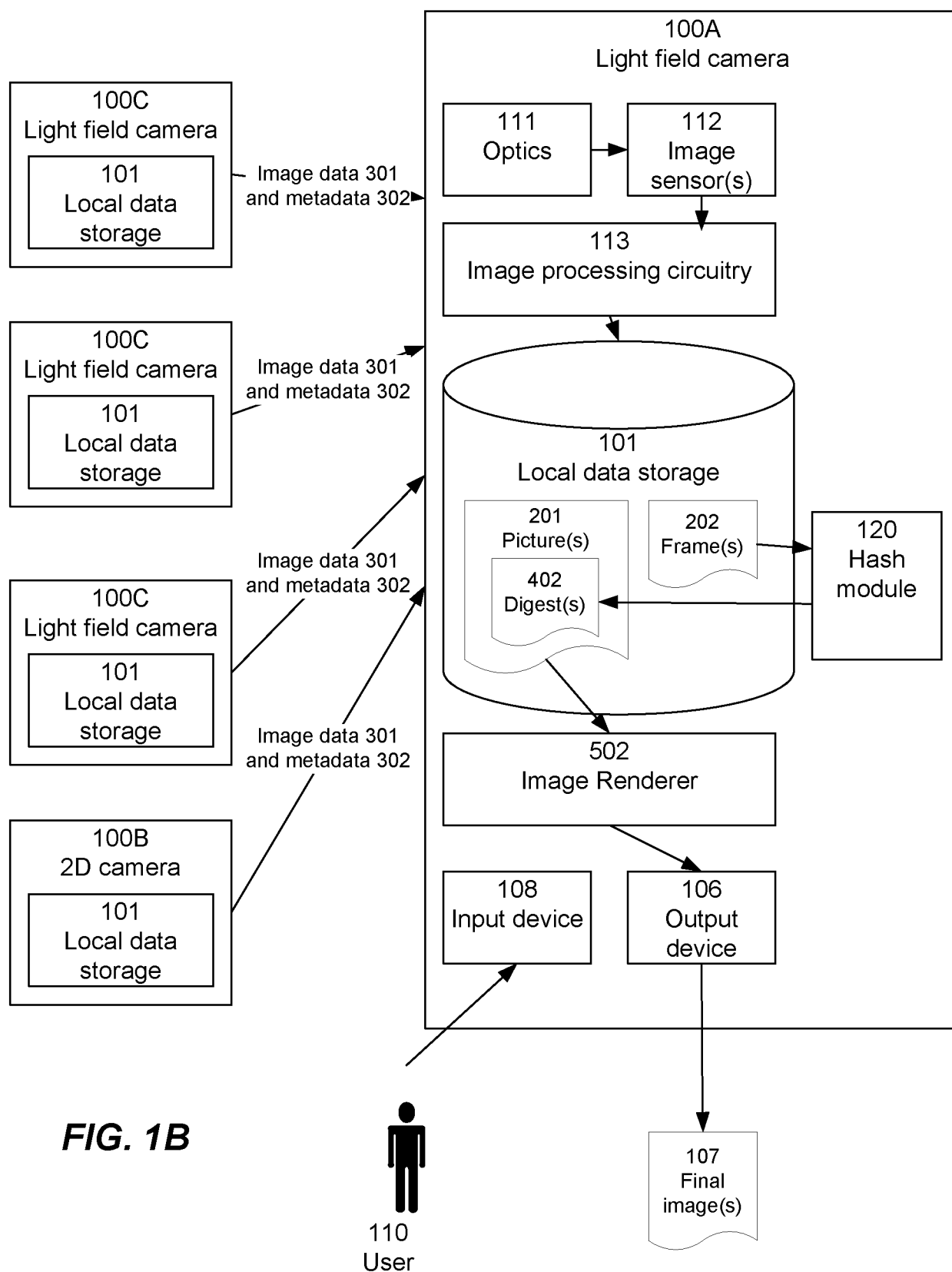
1 70. The computer program product of claim 66, further comprising
2 computer program code configured to cause at least one processor to perform the
3 steps of:
4 causing a storage device to store metadata describing the at least one
5 captured image; and

6 combining the obtained remotely-captured image with at least one image
7 captured at the first device comprises combining the obtained
8 remotely-captured image with at least one image captured at the
9 first device using at least a subset of the metadata.

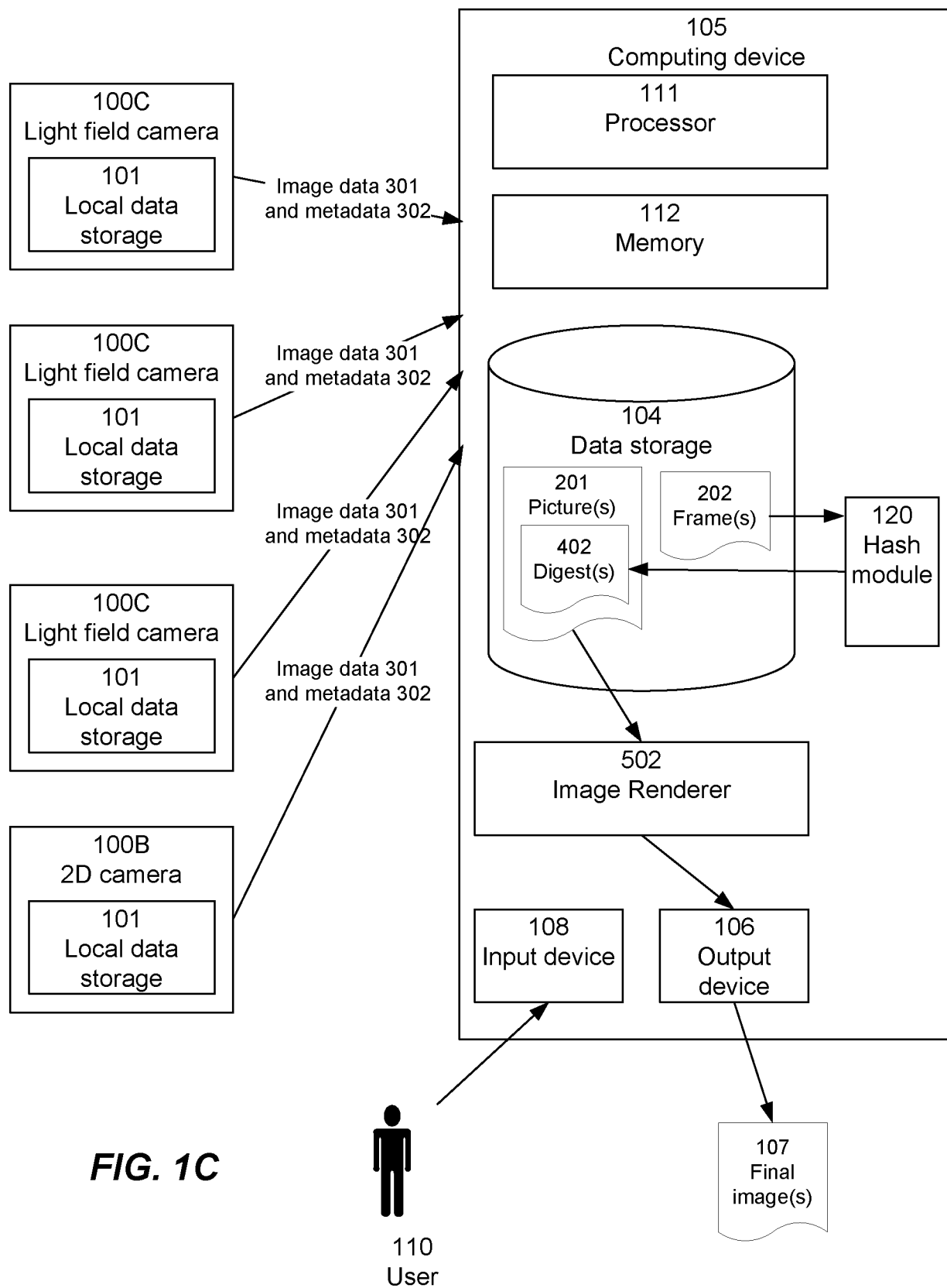
1/15



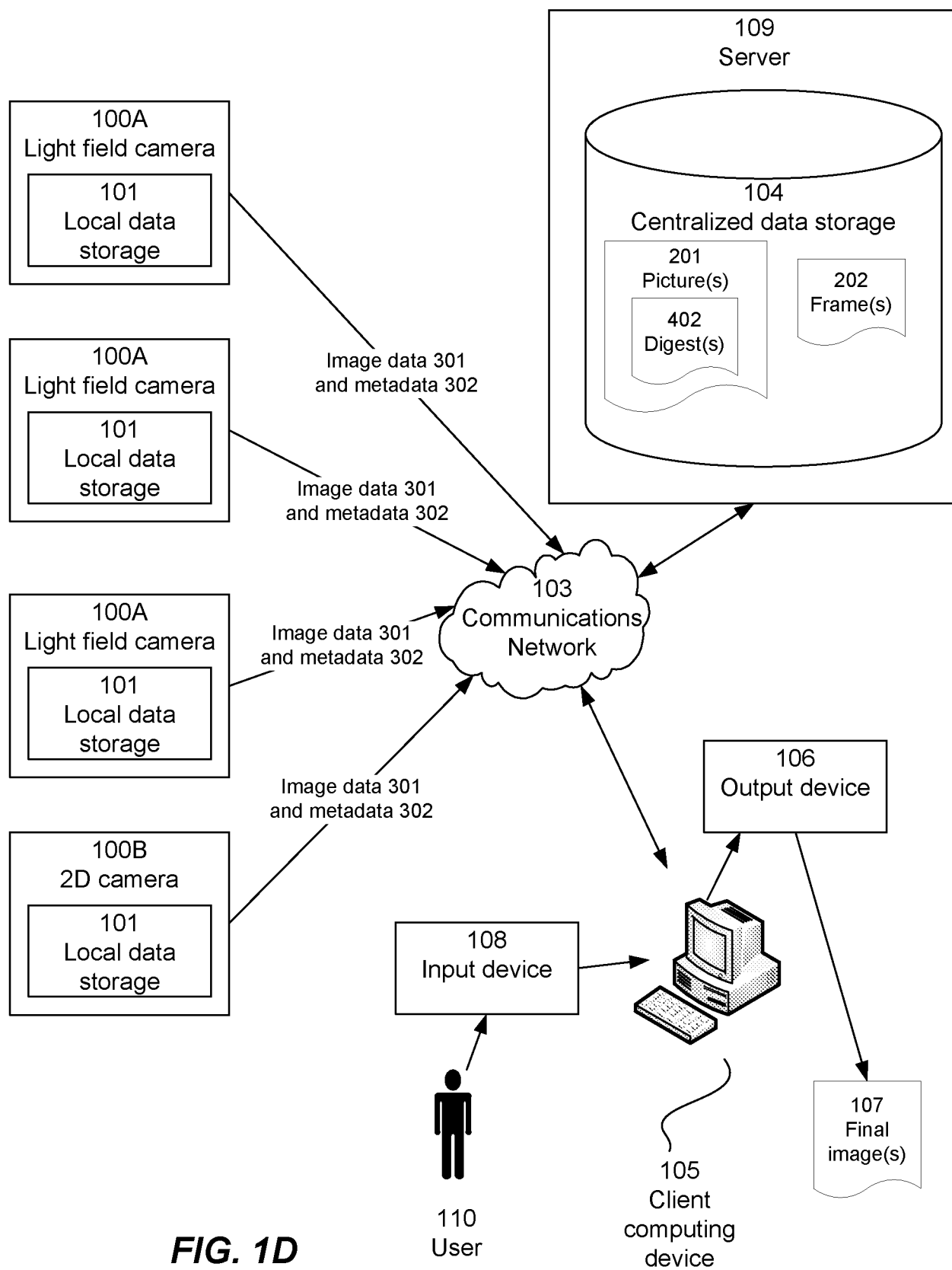
2/15

**FIG. 1B**

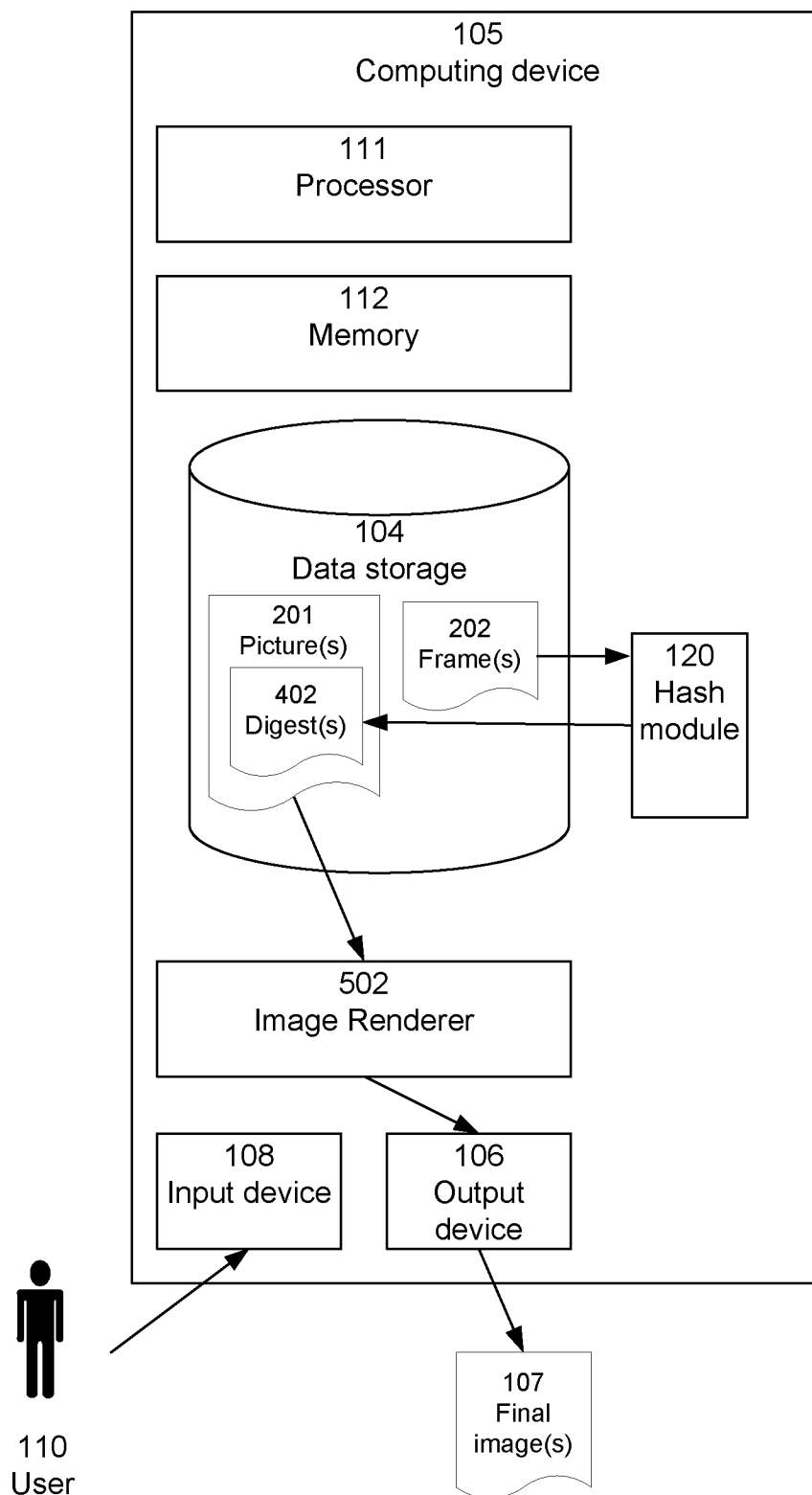
3/15



4/15

**FIG. 1D**

5/15

**FIG. 1E**

6/15

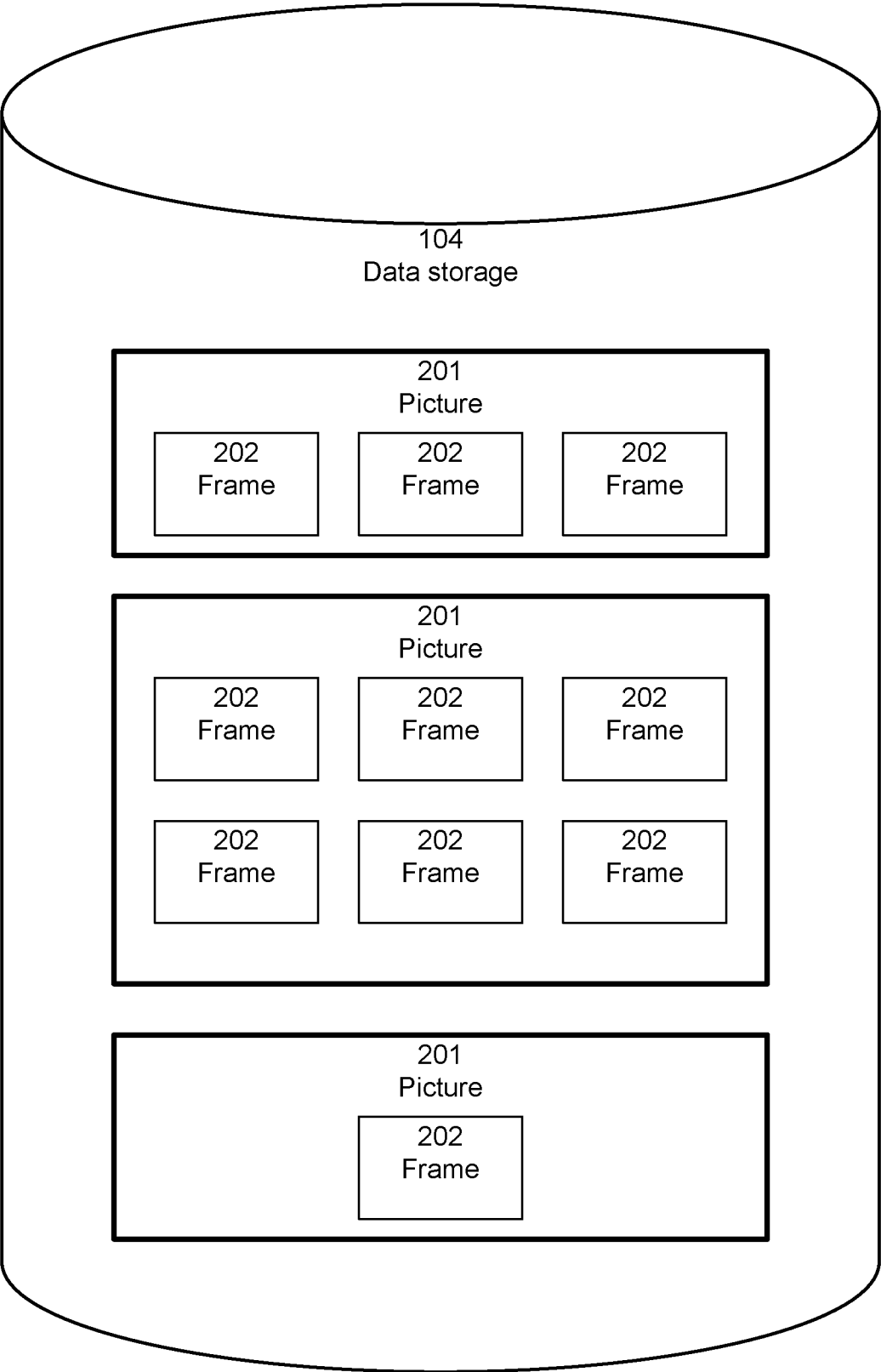


FIG. 2

7/15

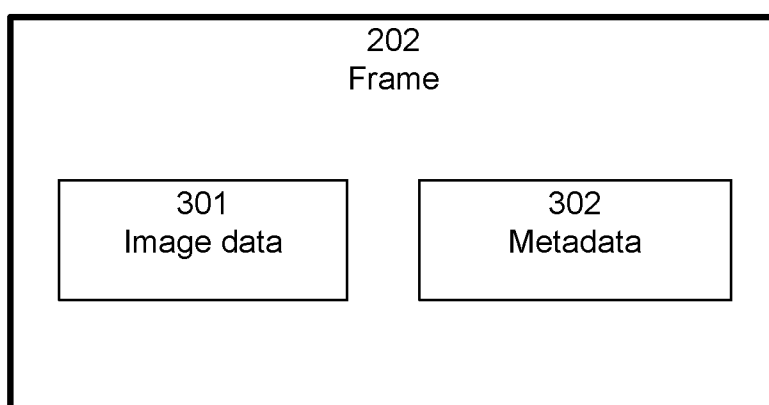
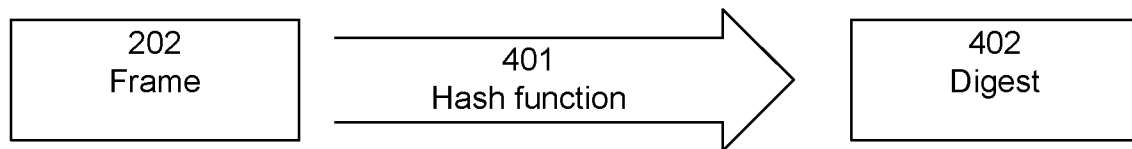
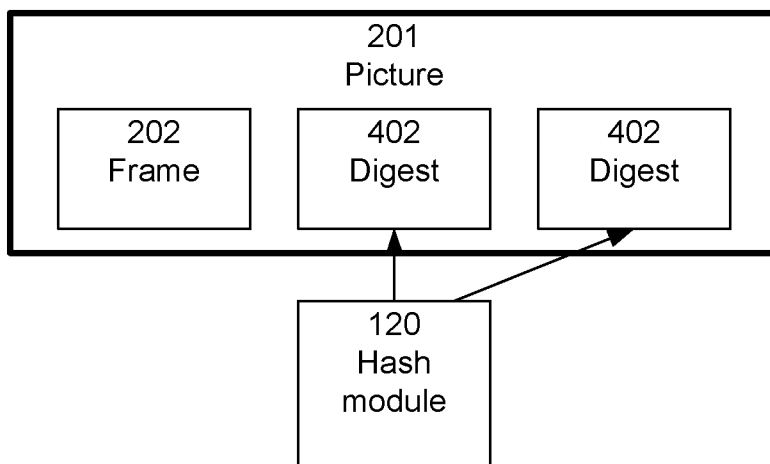
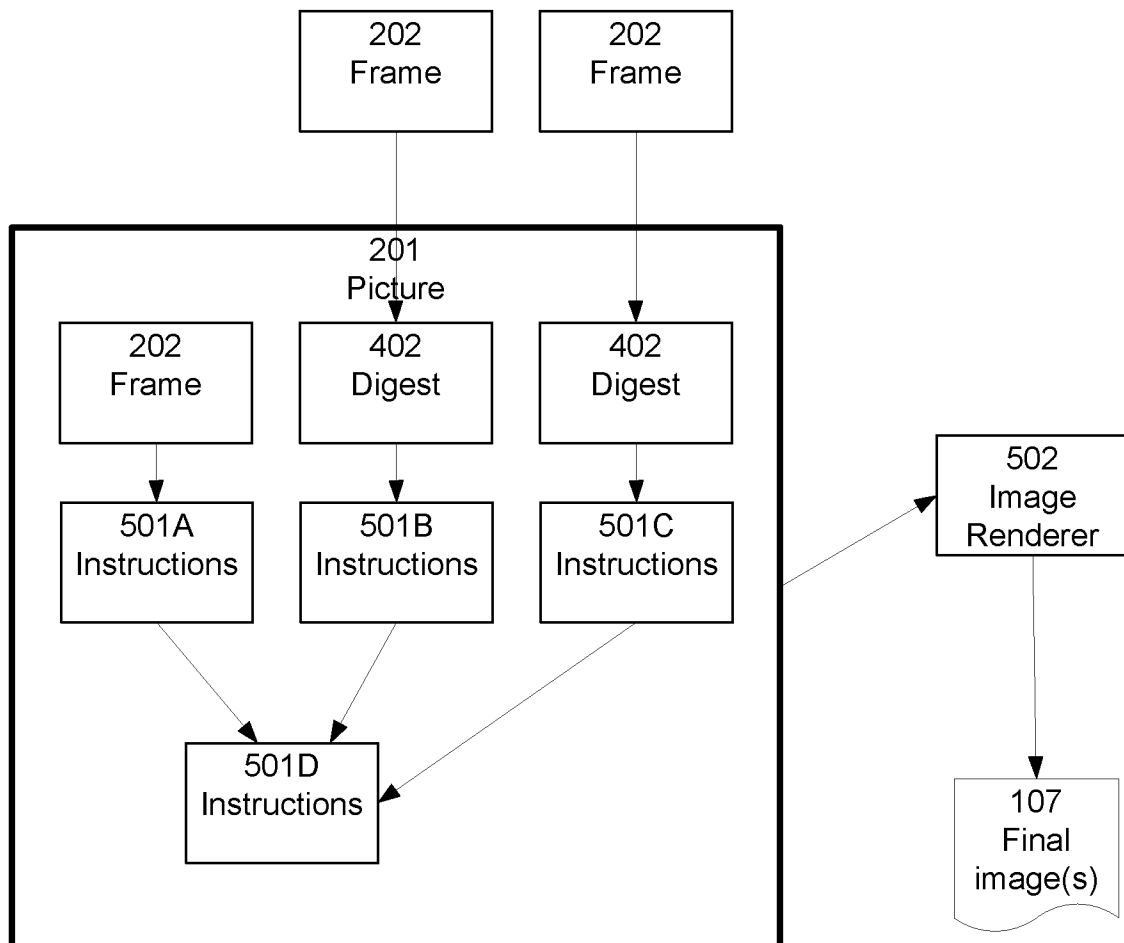


FIG. 3

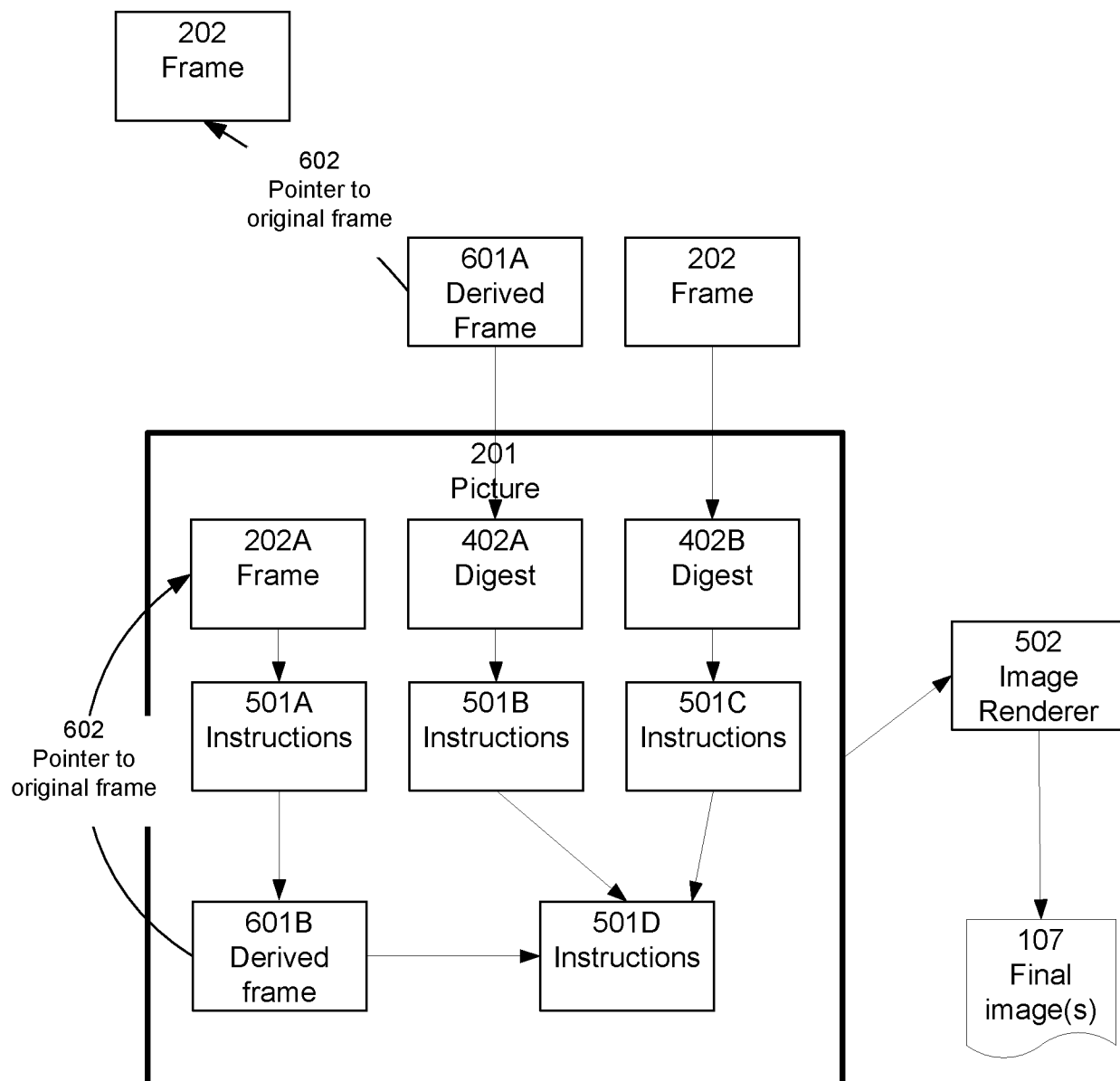
8/15

**FIG. 4A****FIG. 4B**

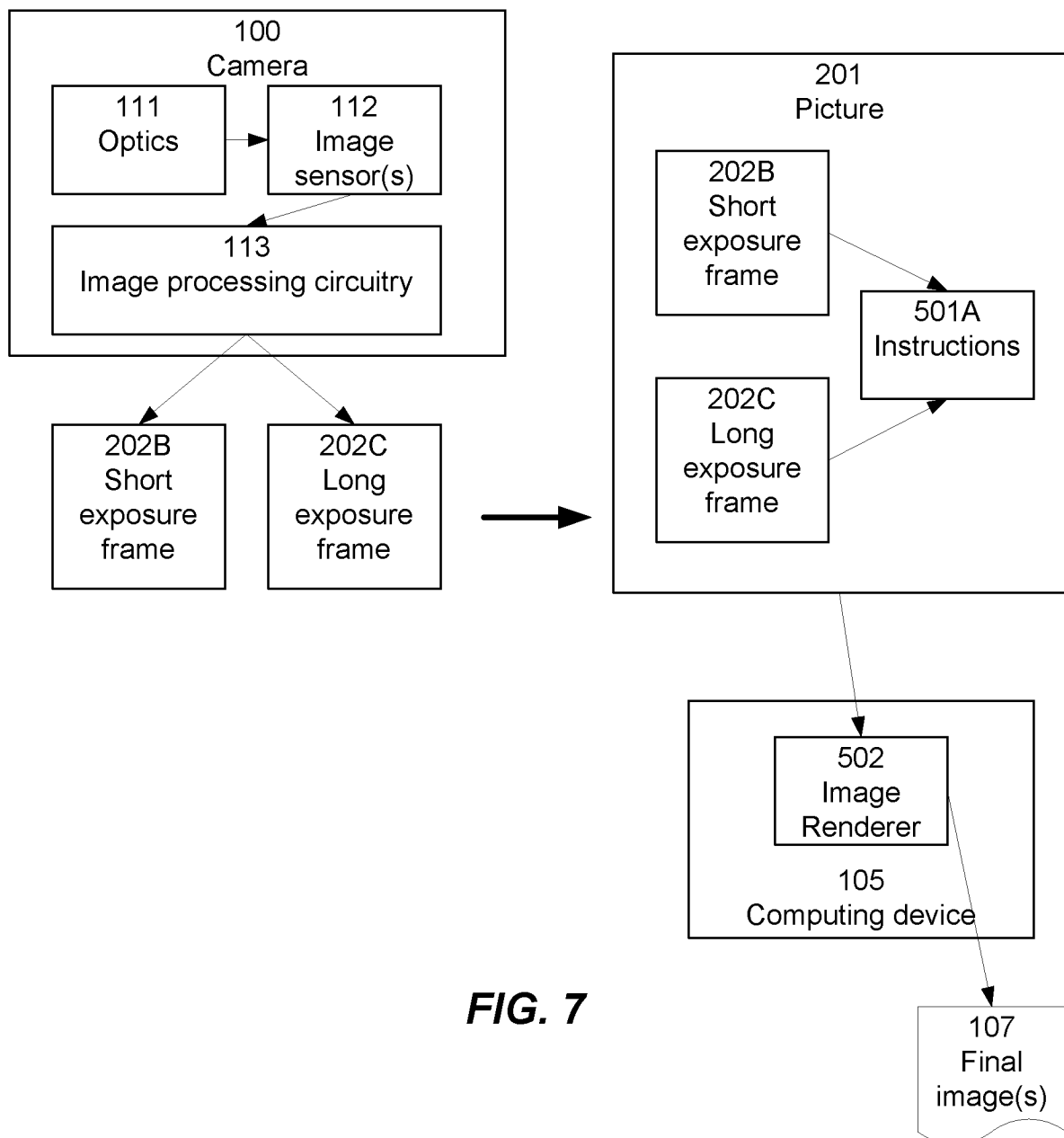
9/15

**FIG. 5**

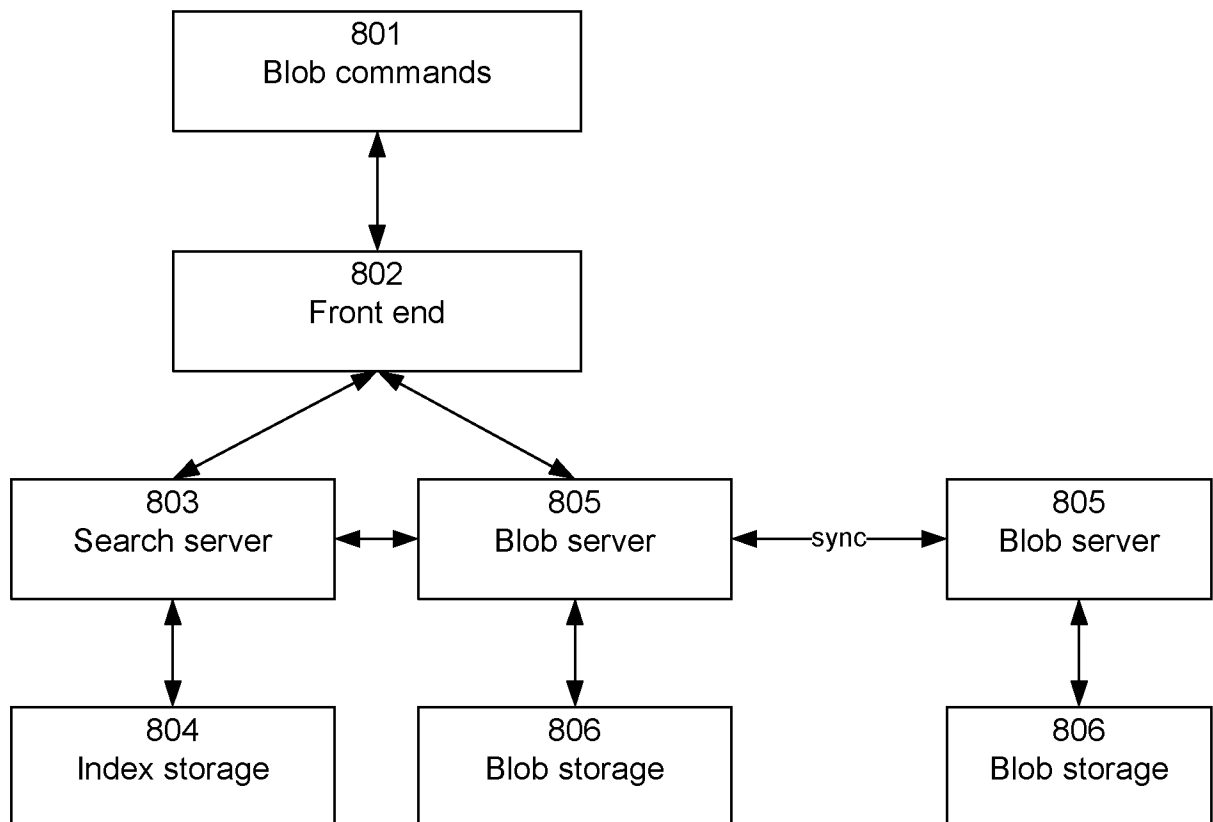
10/15

**FIG. 6**

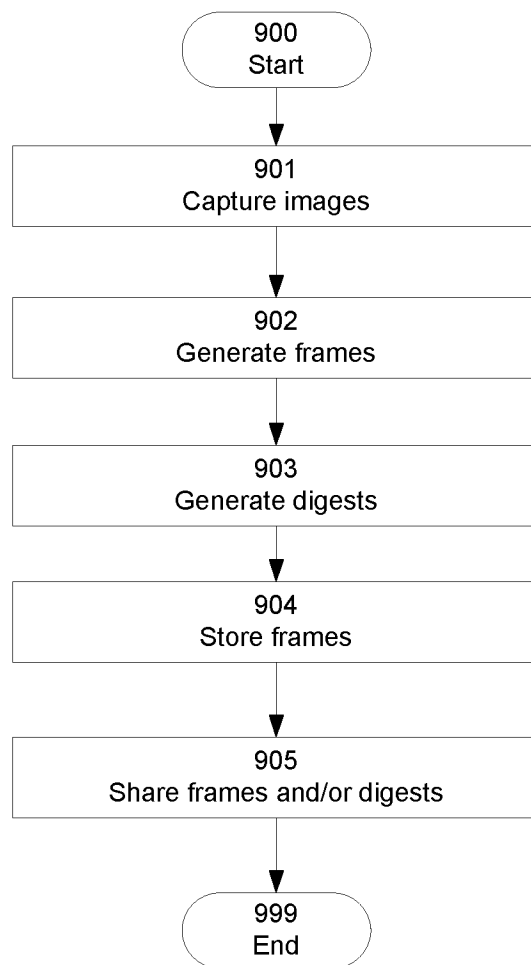
11/15

**FIG. 7**

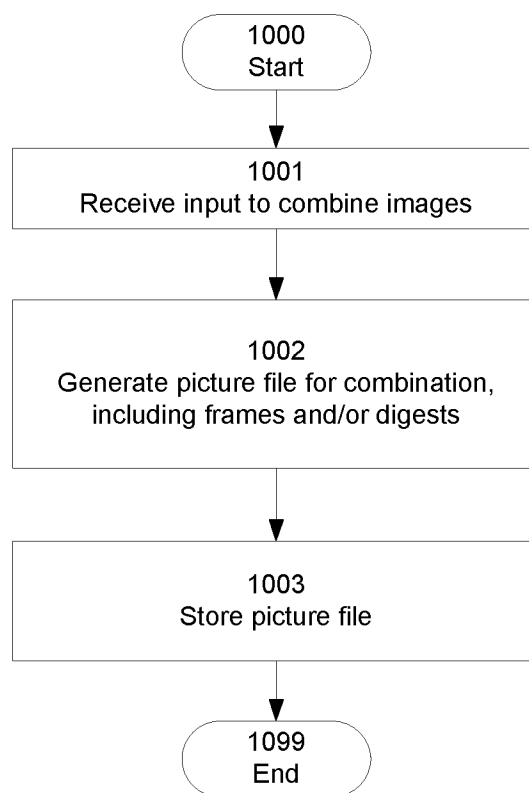
12/15

Blob Storage**FIG. 8**

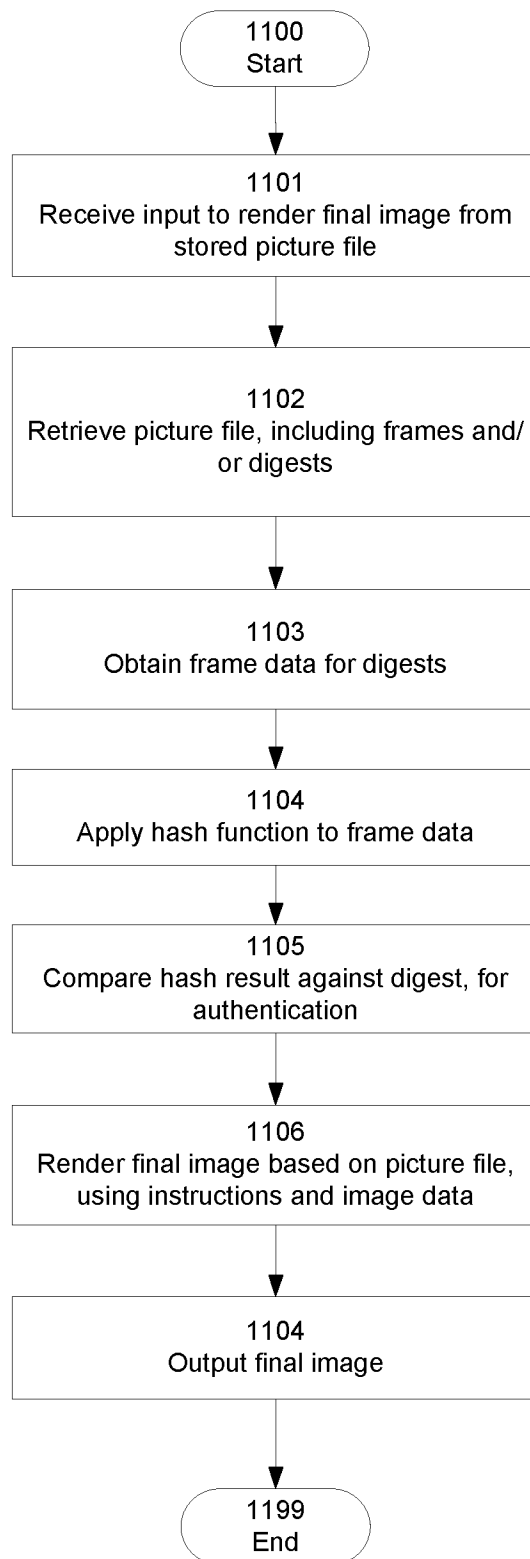
13/15

**FIG. 9**

14/15

**FIG. 10**

15/15

**FIG. 11**

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2012/031958**A. CLASSIFICATION OF SUBJECT MATTER****H04N 5/225(2006.01)i, H04N 5/228(2006.01)i**

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N 5/225; H04N 5/232; G06F 9/445; H04N 5/262; G06F 9/45

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords:image metadata,digest hash

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	KR 10-2011-0059247 A (삼성전자주식회사) 02 June 2011 See abstract;see paragraphs [0010]-[0012],[0024];claim 1 and figures 1,2.	1-70
A	KR 10-2009-0016453 A (어도비시스템즈인코오폐레이티드) 13 February 2009 See abstract;see paragraphs [0034]-[0041];claims 1,6,7,8 and figure 1.	1-70
A	US 2006-0130017 A1 (COHEN JASON, et al.) 15 June 2006 See abstract;see paragraphs [0029]-[0033];claims 1,4,7,14,17 and figure 2.	1-70



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

05 OCTOBER 2012 (05.10.2012)

Date of mailing of the international search report

23 OCTOBER 2012 (23.10.2012)

Name and mailing address of the ISA/KR


 Korean Intellectual Property Office
 189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan
 City, 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

KU, Dae Sung

Telephone No. 82-42-481-8192



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2012/031958

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
KR 10-2011-0059247 A	02.06.2011	US 2011-0129165 A1 WO 2011-065738 A2 WO 2011-065738 A3	02.06.2011 03.06.2011 03.06.2011
KR 10-2009-0016453 A	13.02.2009	CN 101455071 A EP 2008445 A1 JP 04-981124 B2 JP 2009-532993 A US 2007-0230944 A1 US 2010-0020187 A1 US 7620309 B2 US 8238738 B2 WO 2007-115281 A1	10.06.2009 31.12.2008 27.04.2012 10.09.2009 04.10.2007 28.01.2010 17.11.2009 07.08.2012 11.10.2007
US 2006-0130017 A1	15.06.2006	CN 100378648 C CN 100378648 C0 CN 1839413 A0 EP 1654708 A1 JP 2007-521528 A KR 10-2007-0048638 A US 2003-0231188 A1 US 2003-0233363 A1 US 2004-0006689 A1 US 2004-0015536 A1 US 2004-0034849 A1 US 2005-0240627 A1 US 2006-0259754 A1 US 6947954 B2 US 7017144 B2 US 7120786 B2 US 7461095 B2 US 7464176 B2 US 7664944 B2 WO 2005-020156 A1	02.04.2008 02.04.2008 27.09.2006 10.05.2006 02.08.2007 09.05.2007 18.12.2003 18.12.2003 08.01.2004 22.01.2004 19.02.2004 27.10.2005 16.11.2006 20.09.2005 21.03.2006 10.10.2006 02.12.2008 09.12.2008 16.02.2010 03.03.2005