US 20170046868A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2017/0046868 A1

CHERNOV et al. (43) Pub. Date: Feb. 16, 2017

(54) METHOD AND APPARATUS FOR CONSTRUCTING THREE DIMENSIONAL MODEL OF OBJECT

(71) Applicant: **Samsung Electronics Co., Ltd.**, Suwon-si (KR)

(72) Inventors: **Vitaly Vladimirovich CHERNOV**, Moscow (RU); **Artem Gennadievich SHAMSUAROV**, Moscow (RU); **Oleg Fanilevich MURATOV**, Moscow (RU); **Yury Vyacheslavovich SLYNKO**, Moscow Region (RU); **Maria Mikhailovna LYUBIMTSEVA**, Moscow (RU); **Victor Valentinovich BUCHA**, Moscow (RU)

(21) Appl. No.: **15/206,918**

(22) Filed: **Jul. 11, 2016**

(30) **Foreign Application Priority Data**

Aug. 14, 2015 (RU) 2015134345
May 13, 2016 (KR) 10-2016-0058779

Publication Classification

(51) **Int. Cl.**

G06T 15/04	(2006.01)
G06T 17/20	(2006.01)
G06T 7/00	(2006.01)
H04N 13/02	(2006.01)

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

CPC ***G06T 15/04*** (2013.01); ***H04N 13/0221*** (2013.01); ***G06T 17/205*** (2013.01); ***G06T 7/0042*** (2013.01); ***G06T 7/0071*** (2013.01); *H04N 2013/0081* (2013.01)

(57) **ABSTRACT**

A method and an apparatus for constructing a three-dimensional (3D) model of an object are provided. The method includes capturing images of the object by scanning the object along a trajectory around the object, estimating positions of a scanner, which respectively correspond to the captured images, refining the estimated positions of the scanner based on at least two locations on the trajectory, estimating depth maps corresponding to the refined positions of the scanner, generating a surface mesh of the object by fusing the depth maps, and constructing and displaying the 3D model of the object by mapping textures onto the generated surface mesh.

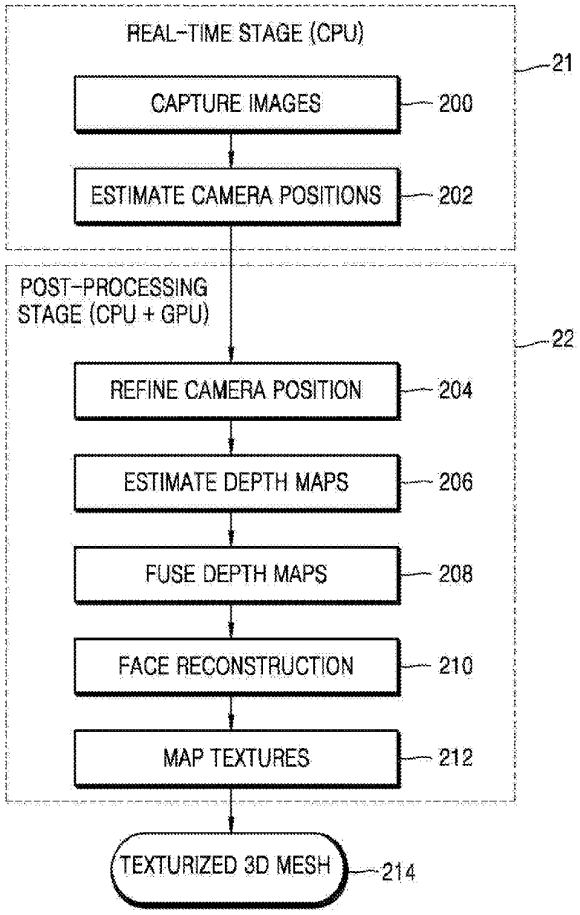


FIG. 1A

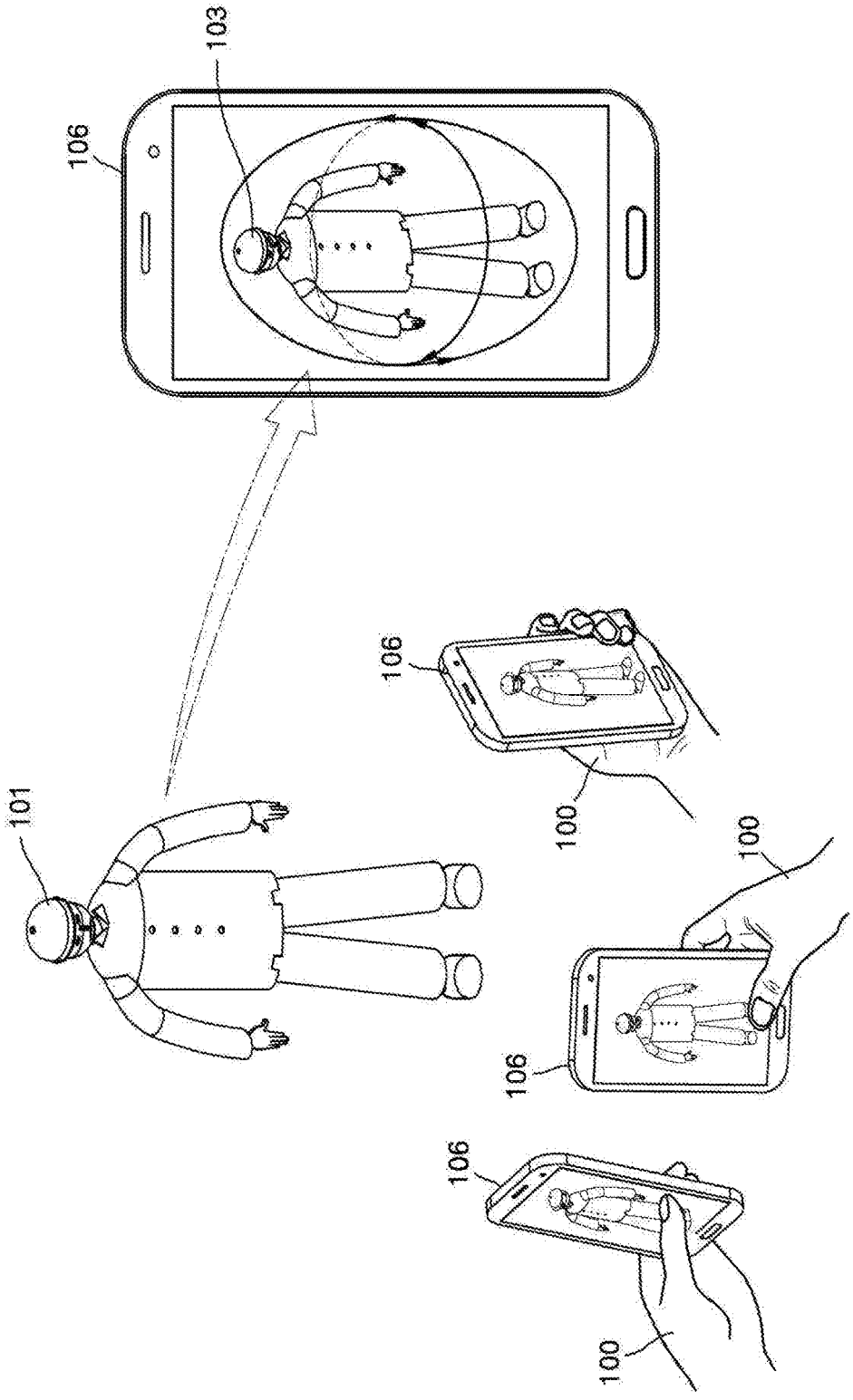


FIG. 1B

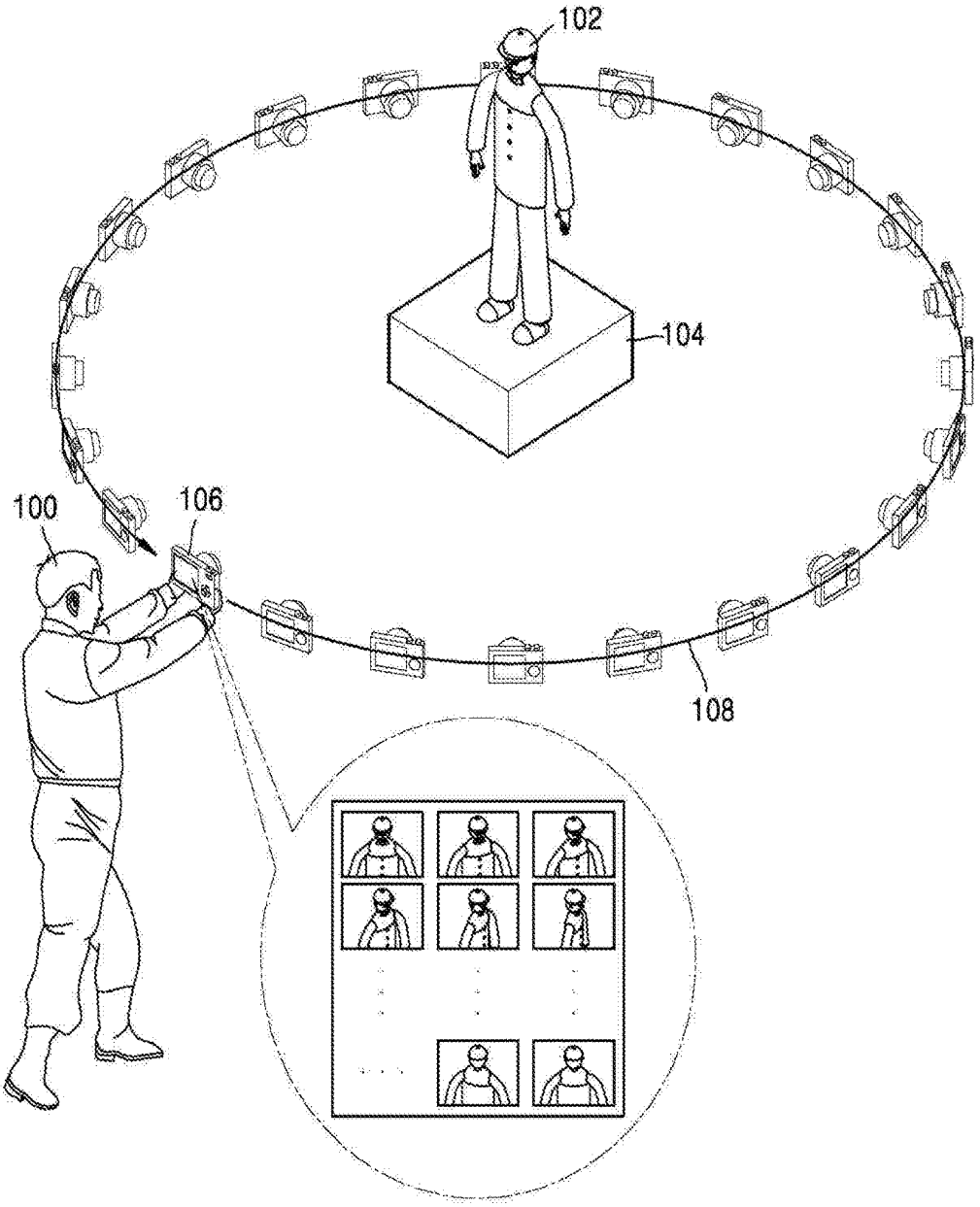


FIG. 2A

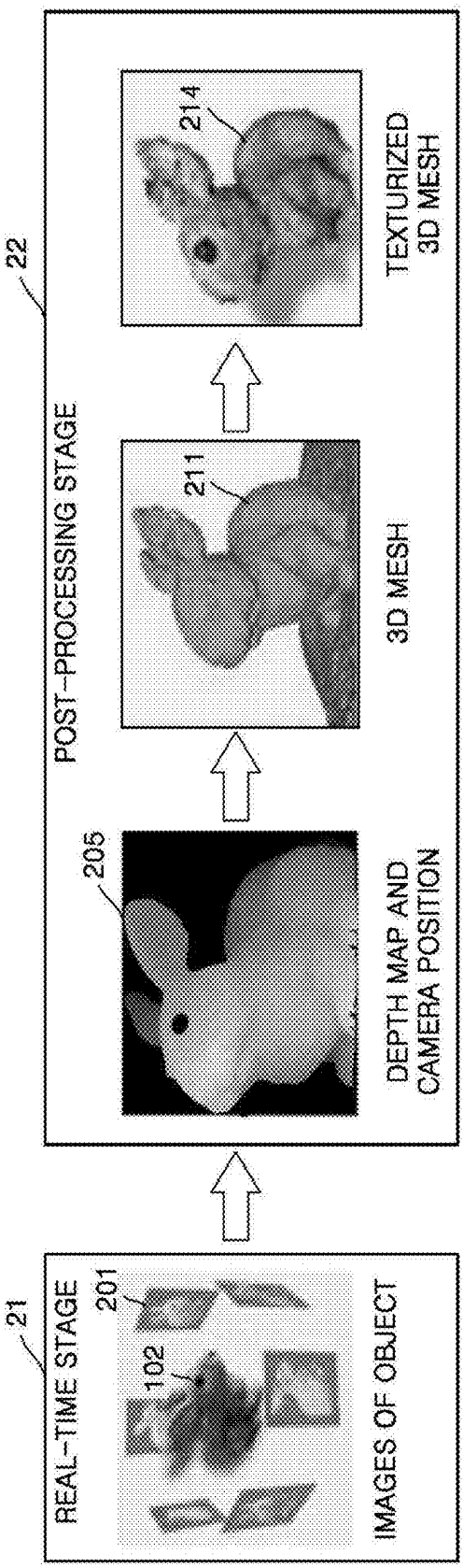

21
REAL-TIME STAGE
201
102
IMAGES OF OBJECT
22
POST-PROCESSING STAGE
205
DEPTH MAP AND CAMERA POSITION
211
3D MESH
214
TEXTURIZED 3D MESH

FIG. 2B

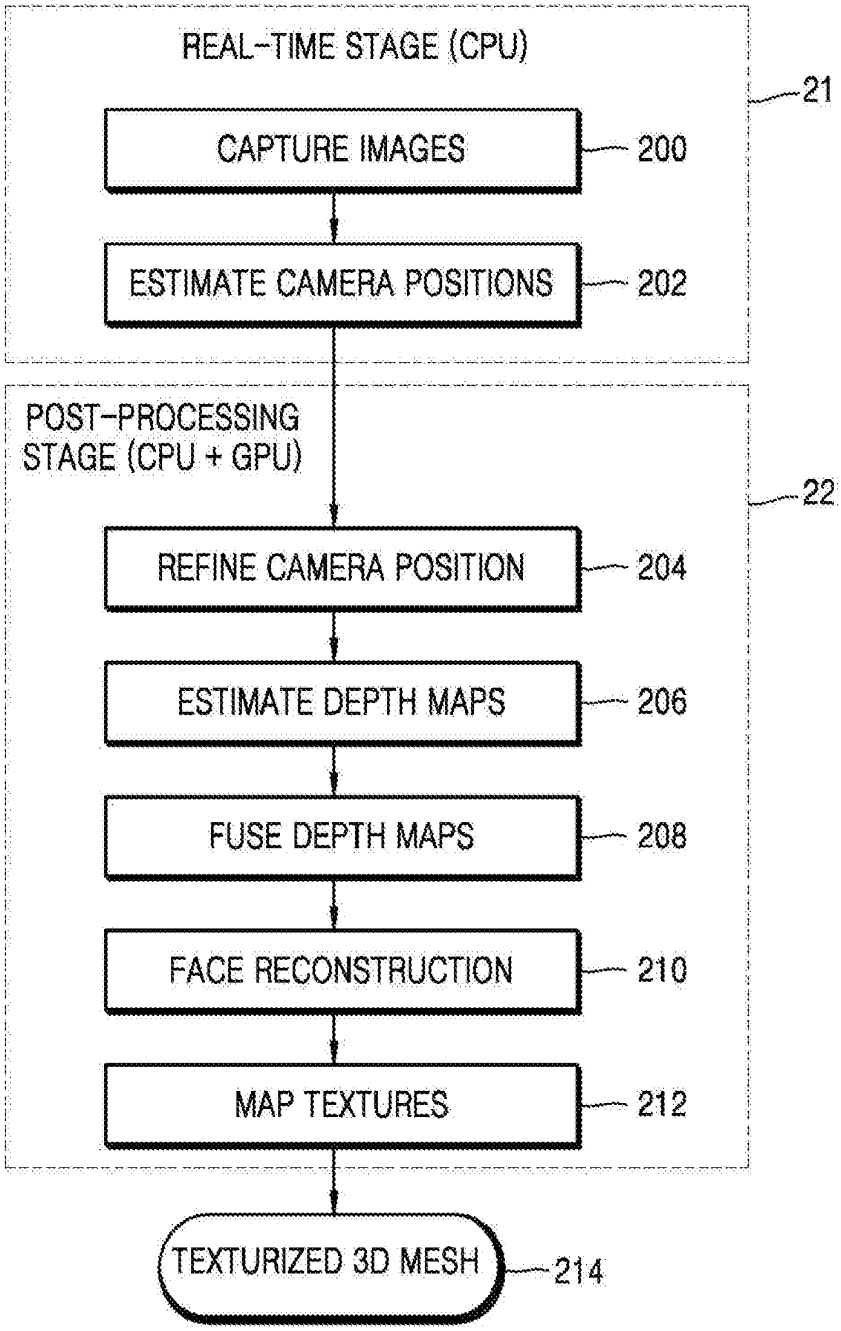

REAL-TIME STAGE (CPU)
21
CAPTURE IMAGES
200
ESTIMATE CAMERA POSITIONS
202
POST-PROCESSING STAGE (CPU + GPU)
22
REFINE CAMERA POSITION
204
ESTIMATE DEPTH MAPS
206
FUSE DEPTH MAPS
208
FACE RECONSTRUCTION
210
MAP TEXTURES
212
TEXTURIZED 3D MESH
214

FIG. 3

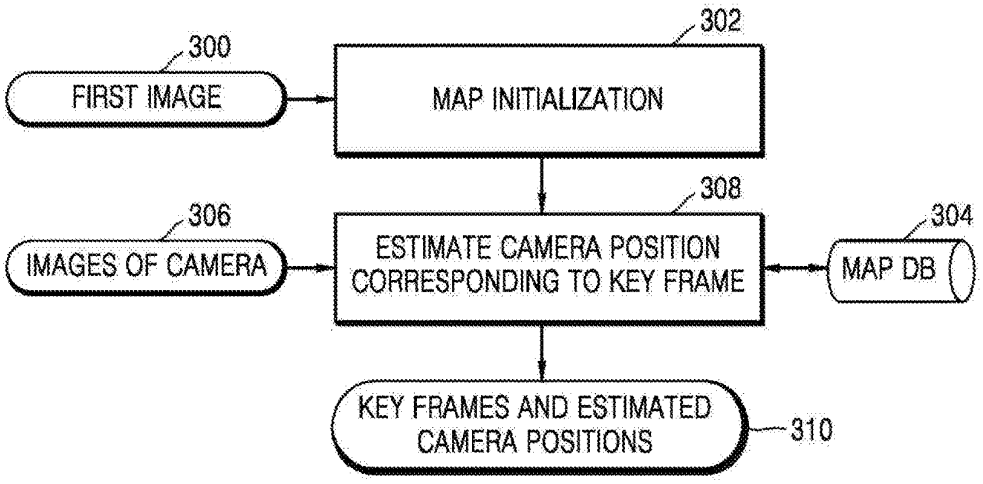

300
FIRST IMAGE
302
MAP INITIALIZATION
306
IMAGES OF CAMERA
308
ESTIMATE CAMERA POSITION CORRESPONDING TO KEY FRAME
304
MAP DB
KEY FRAMES AND ESTIMATED CAMERA POSITIONS
310

FIG. 4

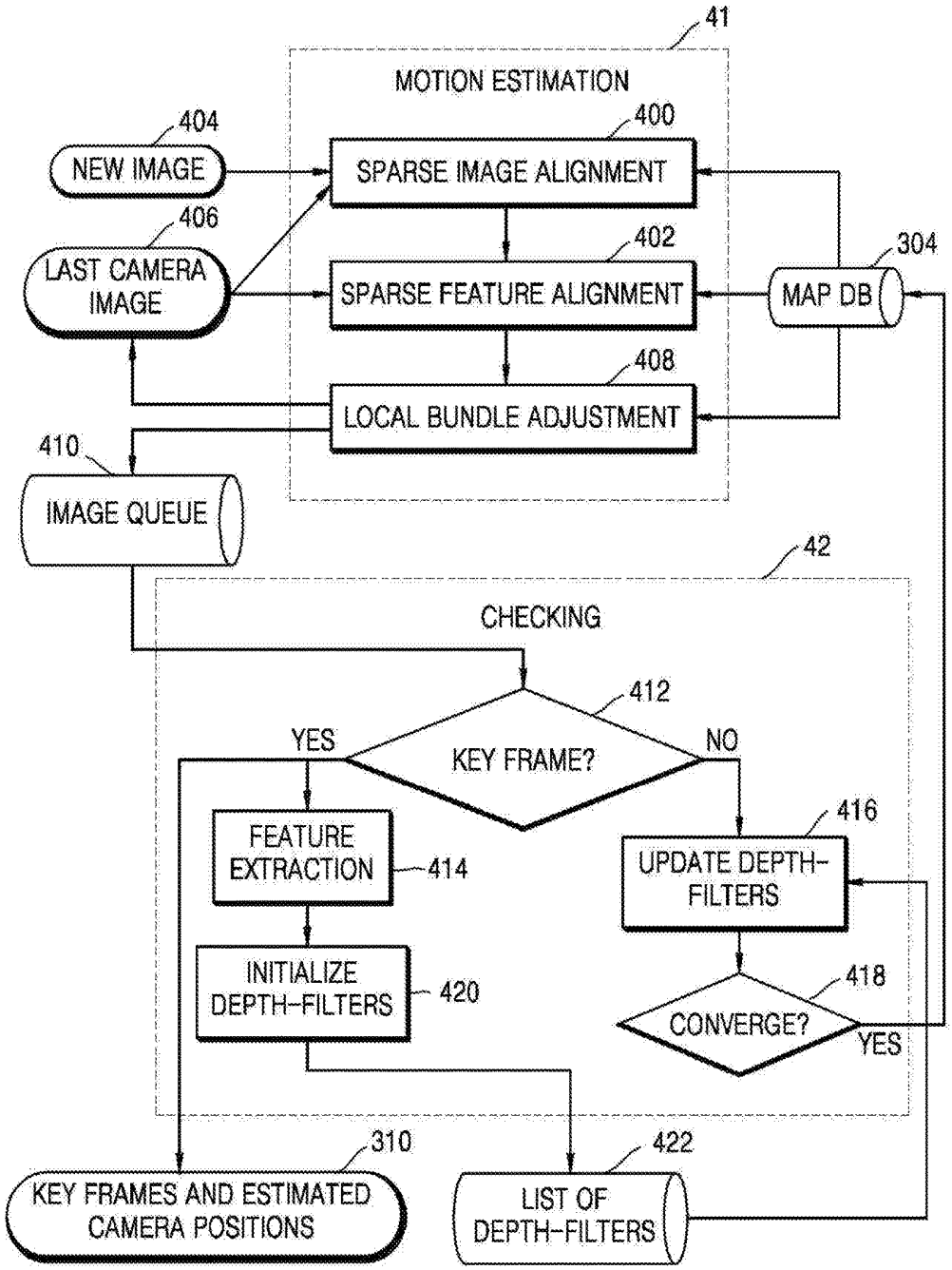

41
MOTION ESTIMATION
404
NEW IMAGE
400
SPARSE IMAGE ALIGNMENT
406
LAST CAMERA IMAGE
402
SPARSE FEATURE ALIGNMENT
304
MAP DB
408
LOCAL BUNDLE ADJUSTMENT
410
IMAGE QUEUE
42
CHECKING
412
YES
KEY FRAME?
NO
FEATURE EXTRACTION
414
416
UPDATE DEPTH-FILTERS
INITIALIZE DEPTH-FILTERS
420
418
CONVERGE?
YES
310
KEY FRAMES AND ESTIMATED CAMERA POSITIONS
422
LIST OF DEPTH-FILTERS

FIG. 5

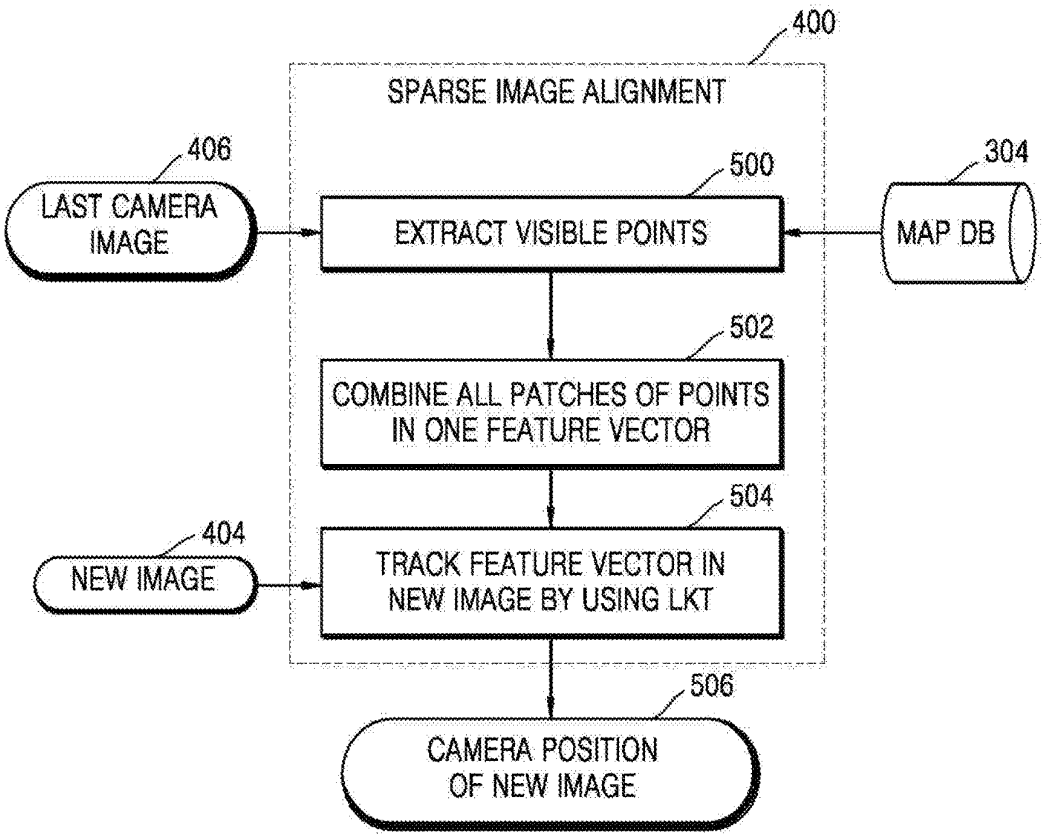

400
SPARSE IMAGE ALIGNMENT
406
LAST CAMERA IMAGE
500
EXTRACT VISIBLE POINTS
304
MAP DB
502
COMBINE ALL PATCHES OF POINTS IN ONE FEATURE VECTOR
504
404
NEW IMAGE
TRACK FEATURE VECTOR IN NEW IMAGE BY USING LKT
506
CAMERA POSITION OF NEW IMAGE

FIG. 6

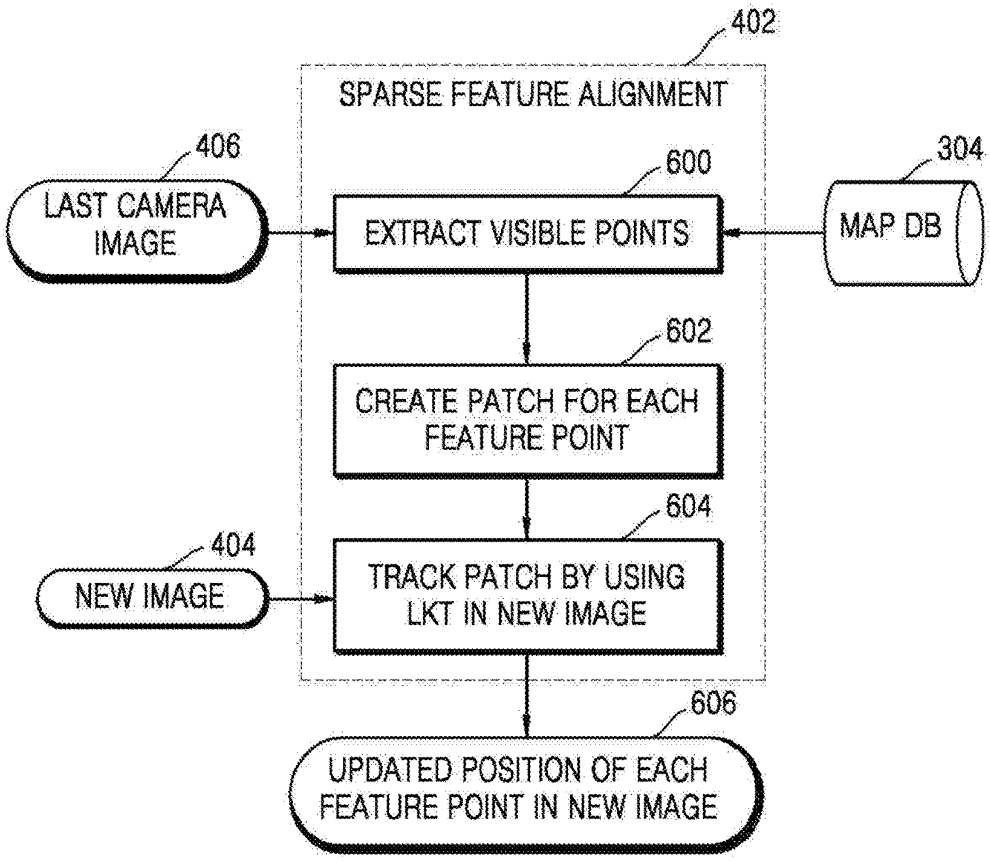

402
SPARSE FEATURE ALIGNMENT
406
LAST CAMERA IMAGE
600
EXTRACT VISIBLE POINTS
304
MAP DB
602
CREATE PATCH FOR EACH FEATURE POINT
604
404
NEW IMAGE
TRACK PATCH BY USING LKT IN NEW IMAGE
606
UPDATED POSITION OF EACH FEATURE POINT IN NEW IMAGE

FIG. 7

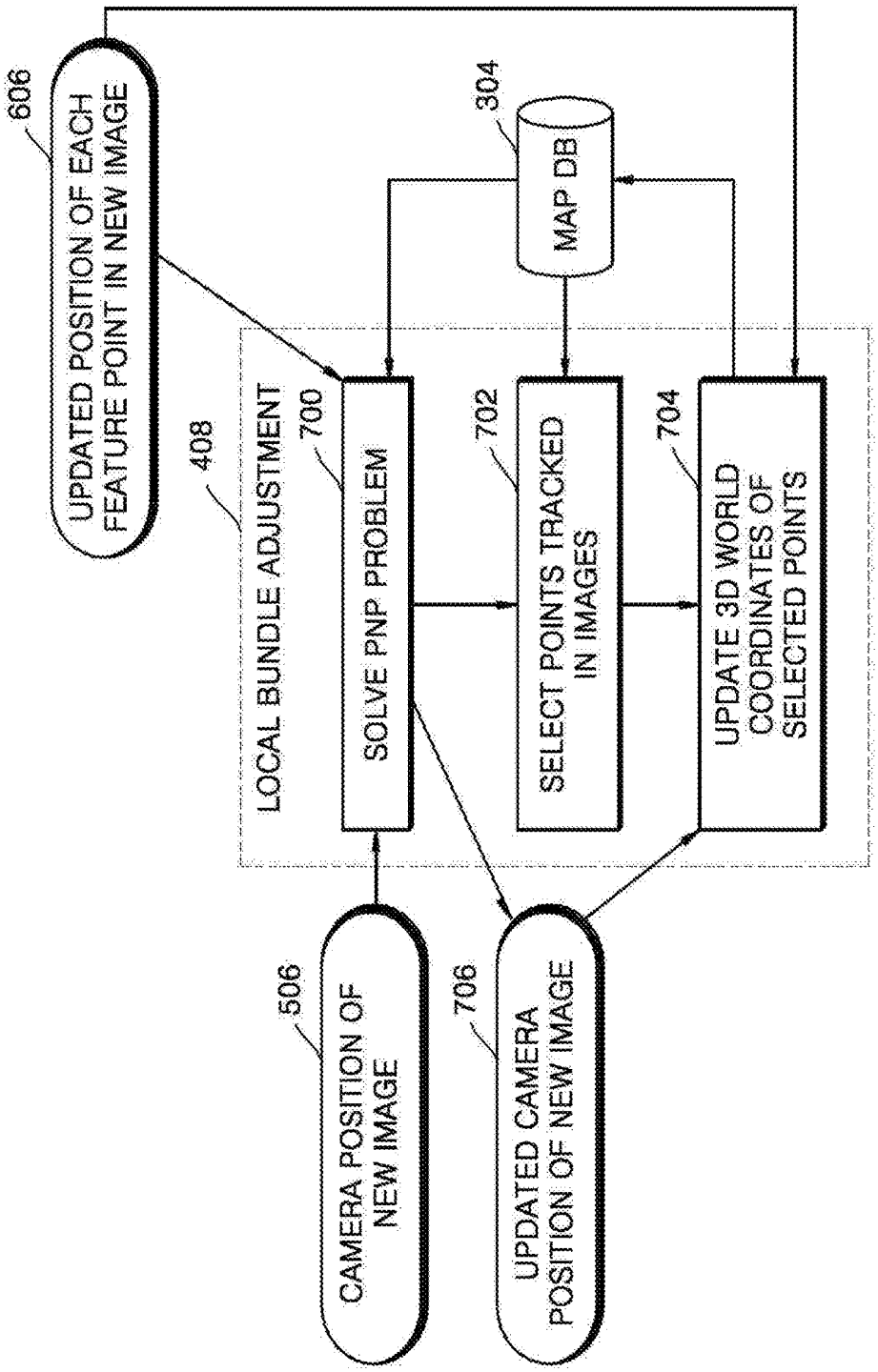

606
UPDATED POSITION OF EACH FEATURE POINT IN NEW IMAGE
304
MAP DB
408
LOCAL BUNDLE ADJUSTMENT
700
SOLVE PNP PROBLEM
702
SELECT POINTS TRACKED IN IMAGES
704
UPDATE 3D WORLD COORDINATES OF SELECTED POINTS
506
CAMERA POSITION OF NEW IMAGE
706
UPDATED CAMERA POSITION OF NEW IMAGE

FIG. 8

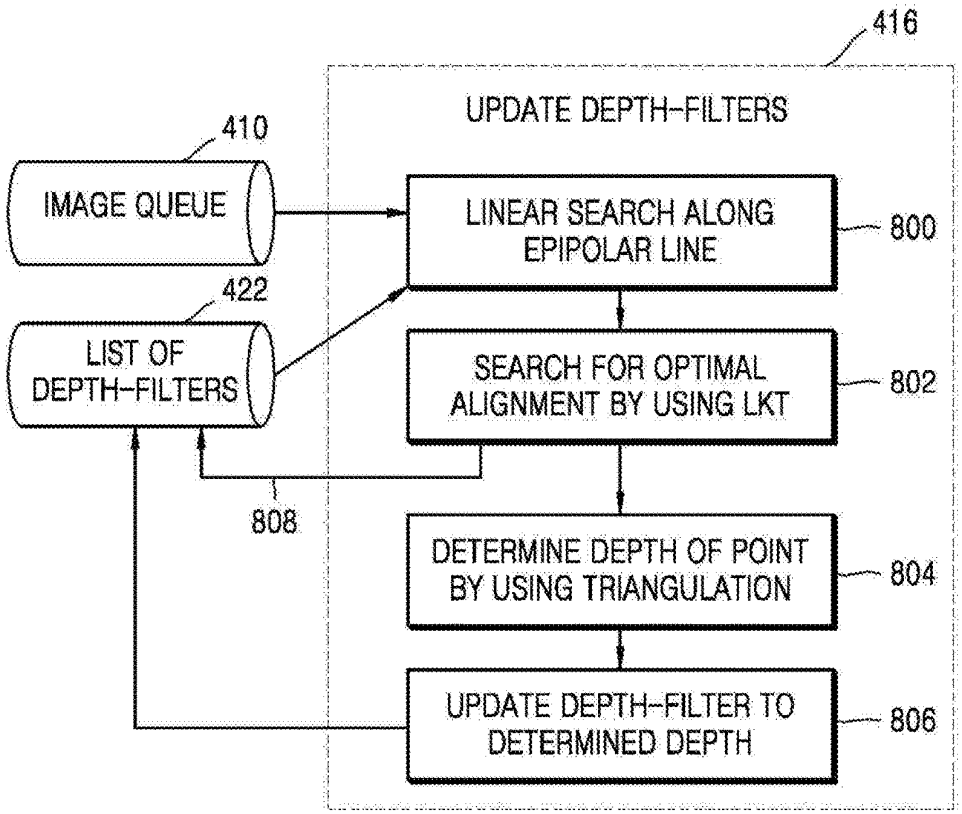

416
UPDATE DEPTH-FILTERS
410
IMAGE QUEUE
422
LIST OF DEPTH-FILTERS
LINEAR SEARCH ALONG EPIPOLAR LINE
800
SEARCH FOR OPTIMAL ALIGNMENT BY USING LKT
802
808
DETERMINE DEPTH OF POINT BY USING TRIANGULATION
804
UPDATE DEPTH-FILTER TO DETERMINED DEPTH
806

FIG. 9

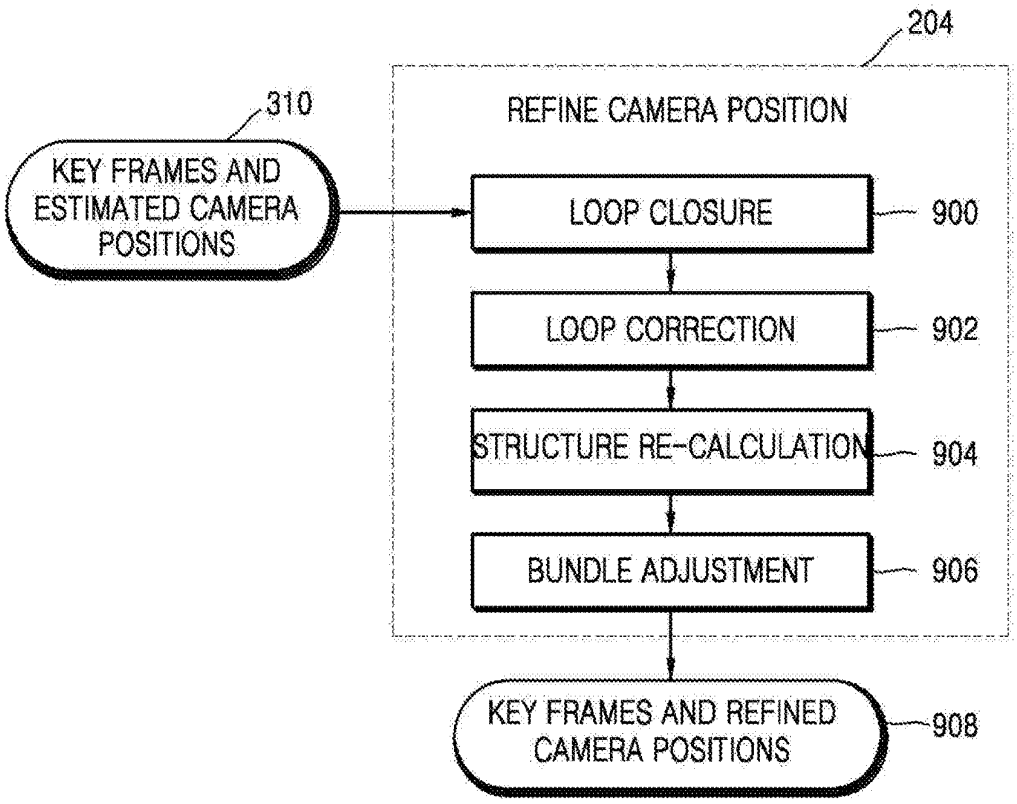

204
REFINE CAMERA POSITION
310
KEY FRAMES AND ESTIMATED CAMERA POSITIONS
LOOP CLOSURE
900
LOOP CORRECTION
902
STRUCTURE RE-CALCULATION
904
BUNDLE ADJUSTMENT
906
KEY FRAMES AND REFINED CAMERA POSITIONS
908

FIG. 10

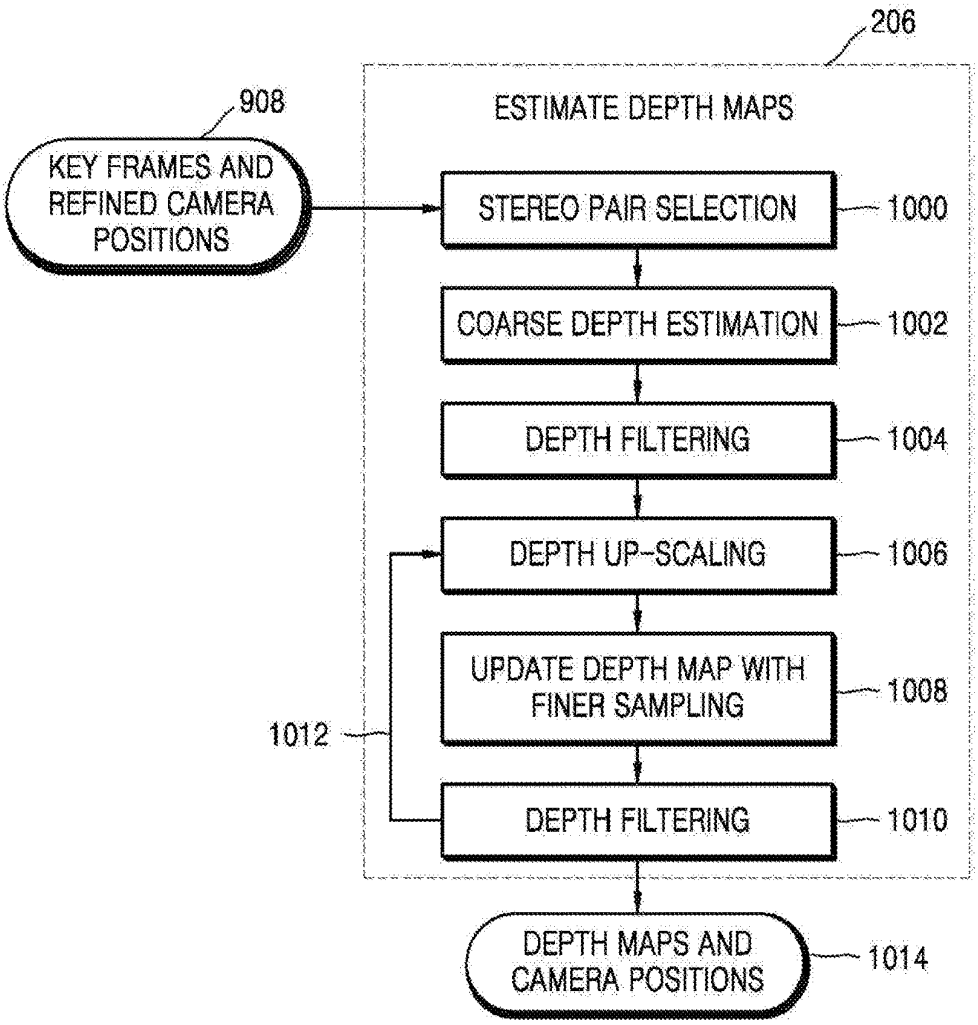

206
ESTIMATE DEPTH MAPS
908
KEY FRAMES AND REFINED CAMERA POSITIONS
STEREO PAIR SELECTION
1000
COARSE DEPTH ESTIMATION
1002
DEPTH FILTERING
1004
DEPTH UP-SCALING
1006
UPDATE DEPTH MAP WITH FINER SAMPLING
1008
1012
DEPTH FILTERING
1010
DEPTH MAPS AND CAMERA POSITIONS
1014

FIG. 11

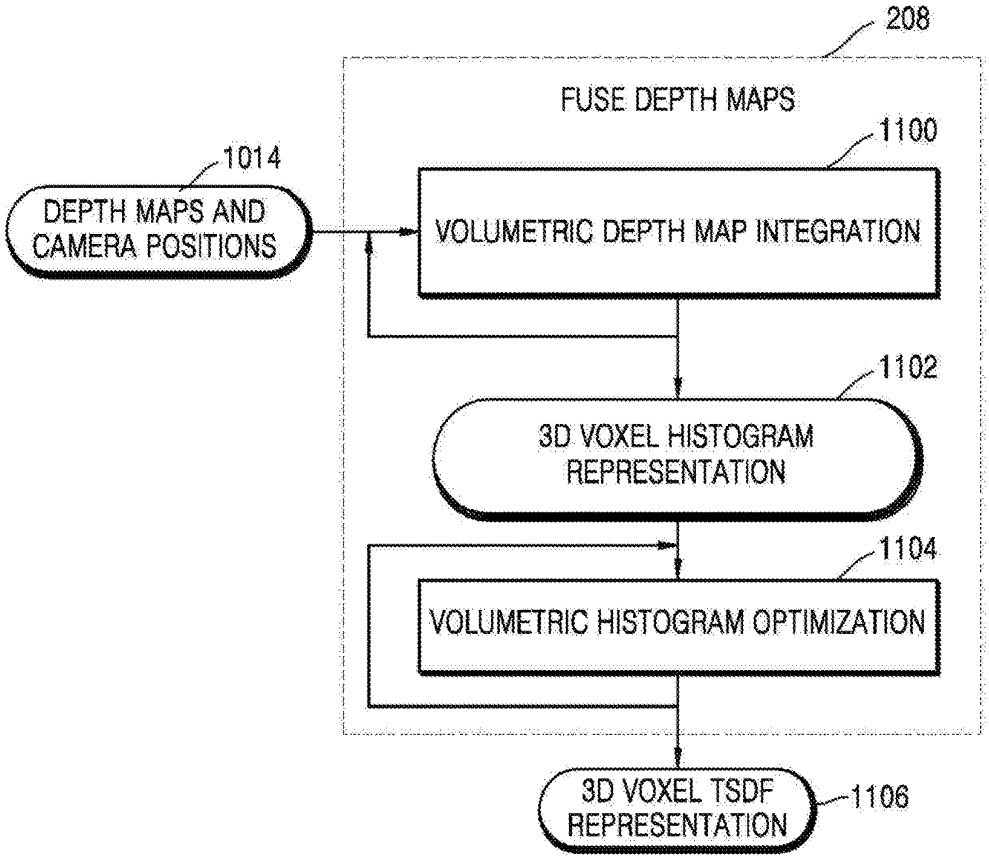

208
FUSE DEPTH MAPS
1100
1014
DEPTH MAPS AND CAMERA POSITIONS
VOLUMETRIC DEPTH MAP INTEGRATION
1102
3D VOXEL HISTOGRAM REPRESENTATION
1104
VOLUMETRIC HISTOGRAM OPTIMIZATION
3D VOXEL TSDF REPRESENTATION
1106

FIG. 12

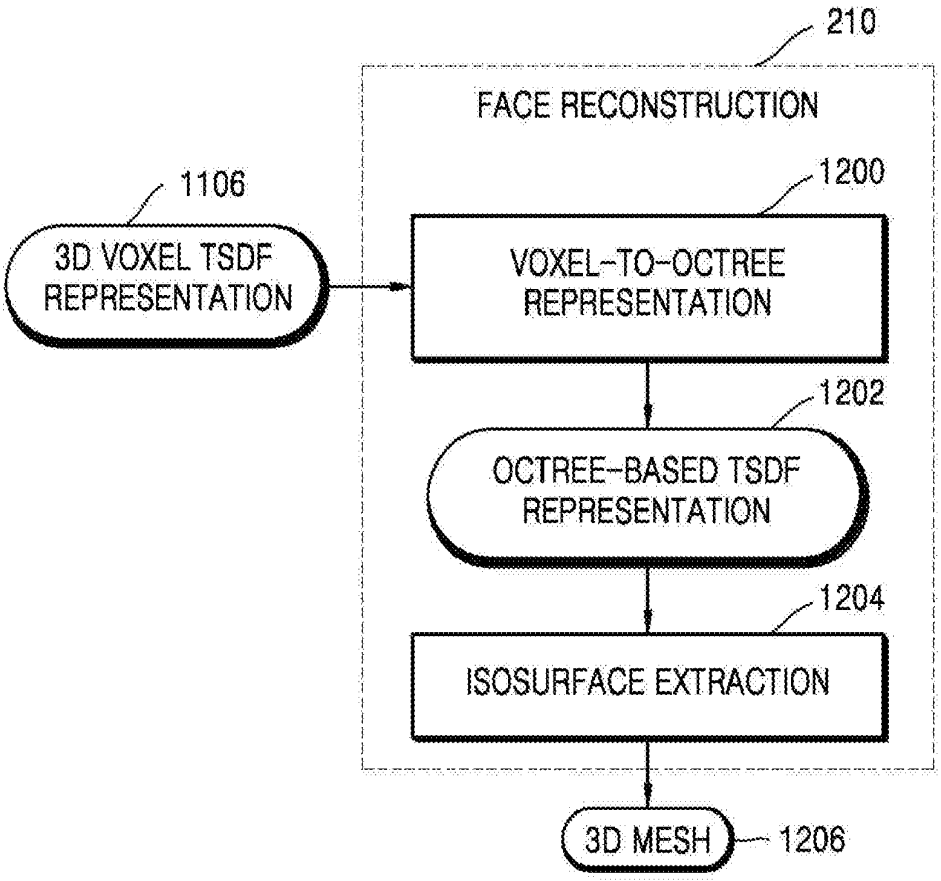

210
FACE RECONSTRUCTION
1106
3D VOXEL TSDF REPRESENTATION
1200
VOXEL-TO-OCTREE REPRESENTATION
1202
OCTREE-BASED TSDF REPRESENTATION
1204
ISOSURFACE EXTRACTION
3D MESH
1206

FIG. 13

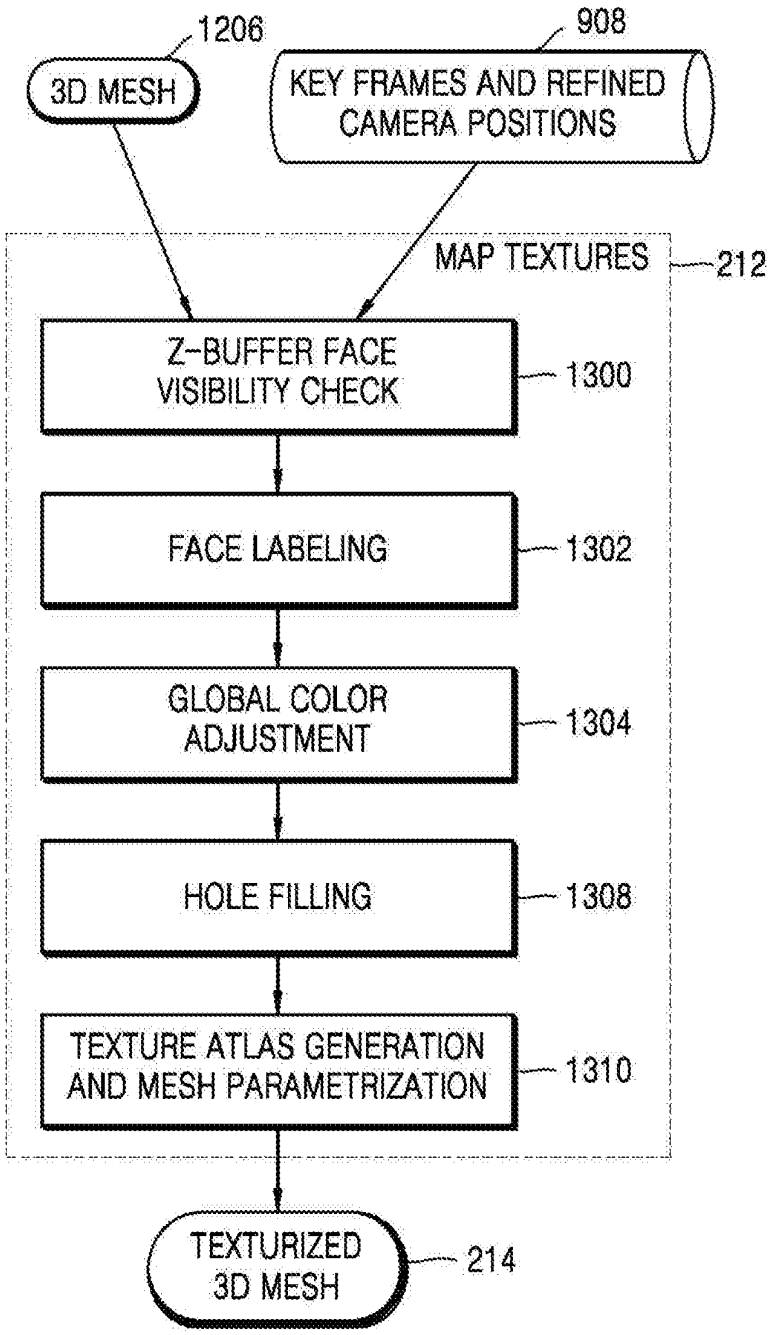

1206
3D MESH
908
KEY FRAMES AND REFINED CAMERA POSITIONS
MAP TEXTURES
212
Z-BUFFER FACE VISIBILITY CHECK
1300
FACE LABELING
1302
GLOBAL COLOR ADJUSTMENT
1304
HOLE FILLING
1308
TEXTURE ATLAS GENERATION AND MESH PARAMETRIZATION
1310
TEXTURIZED 3D MESH
214

FIG. 14

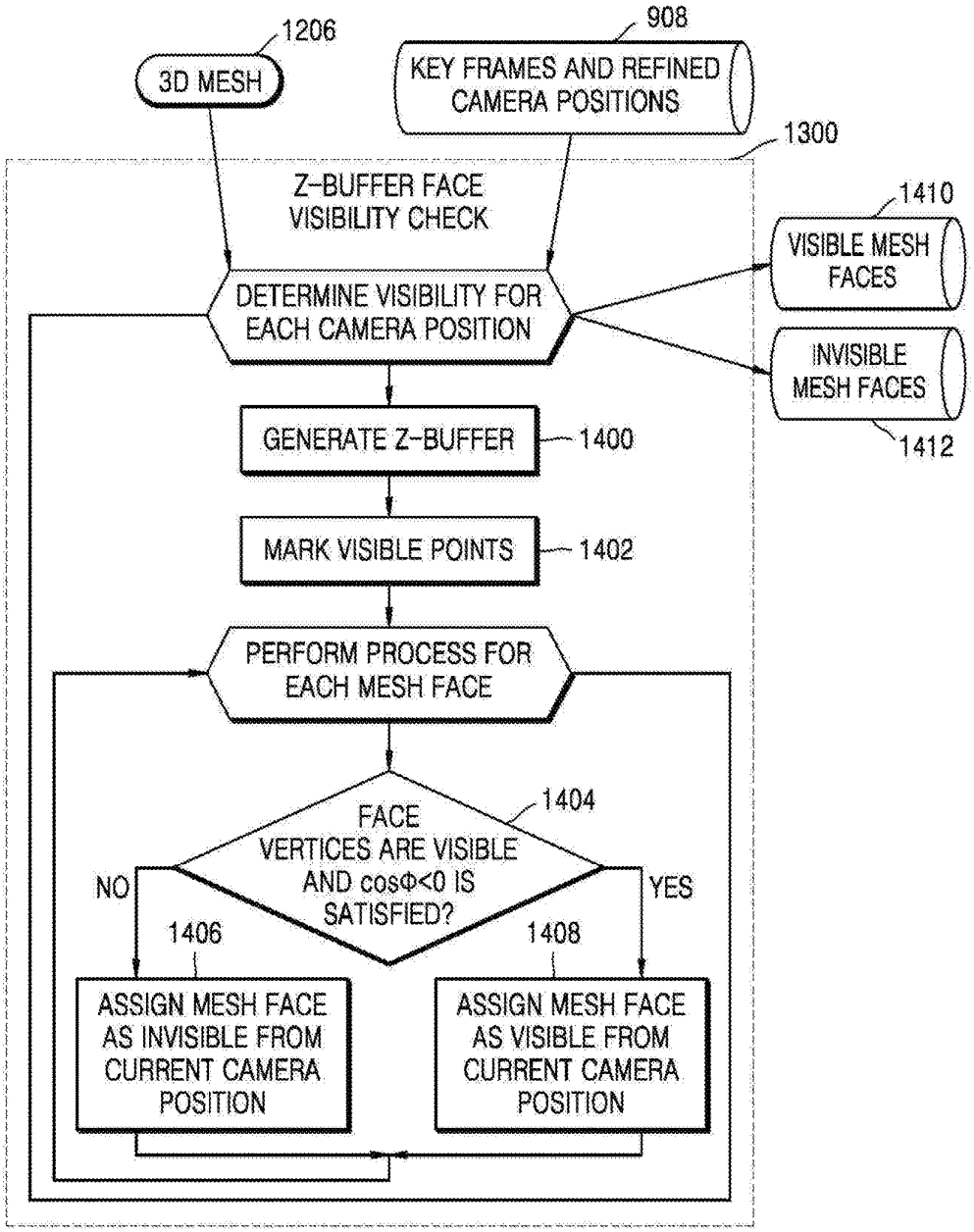

1206
3D MESH
908
KEY FRAMES AND REFINED CAMERA POSITIONS
1300
Z-BUFFER FACE VISIBILITY CHECK
1410
VISIBLE MESH FACES
DETERMINE VISIBILITY FOR EACH CAMERA POSITION
INVISIBLE MESH FACES
1412
GENERATE Z-BUFFER
1400
MARK VISIBLE POINTS
1402
PERFORM PROCESS FOR EACH MESH FACE
1404
FACE VERTICES ARE VISIBLE AND cosΦ<0 IS SATISFIED?
NO
YES
1406
1408
ASSIGN MESH FACE AS INVISIBLE FROM CURRENT CAMERA POSITION
ASSIGN MESH FACE AS VISIBLE FROM CURRENT CAMERA POSITION

FIG. 15

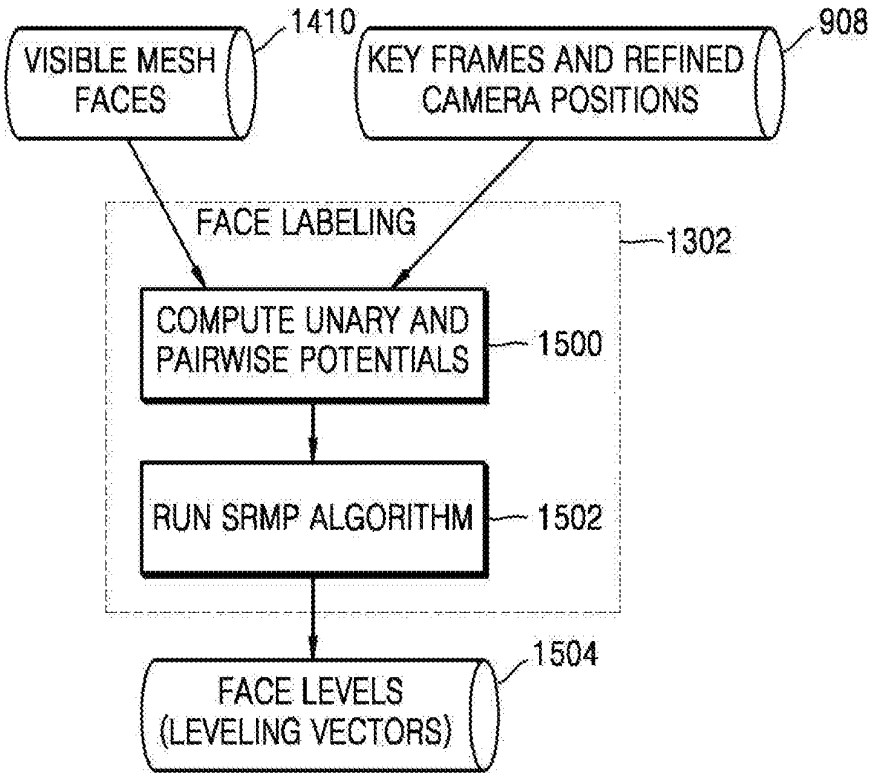

VISIBLE MESH FACES
1410
KEY FRAMES AND REFINED CAMERA POSITIONS
908
FACE LABELING
1302
COMPUTE UNARY AND PAIRWISE POTENTIALS
1500
RUN SRMP ALGORITHM
1502
FACE LEVELS (LEVELING VECTORS)
1504

FIG. 16

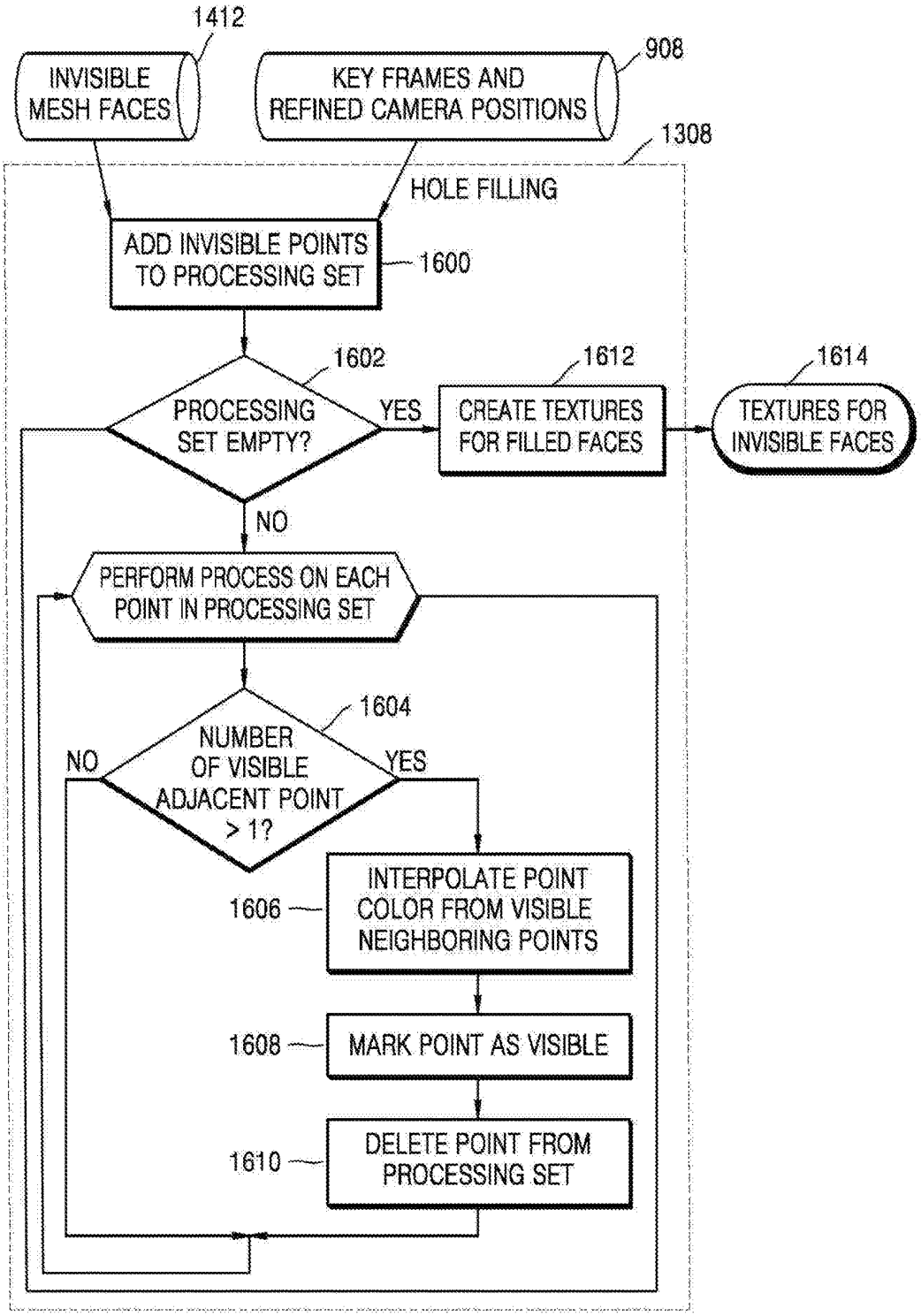

1412
INVISIBLE MESH FACES
908
KEY FRAMES AND REFINED CAMERA POSITIONS
1308
HOLE FILLING
ADD INVISIBLE POINTS TO PROCESSING SET
1600
1602
PROCESSING SET EMPTY?
YES
1612
CREATE TEXTURES FOR FILLED FACES
1614
TEXTURES FOR INVISIBLE FACES
NO
PERFORM PROCESS ON EACH POINT IN PROCESSING SET
1604
NUMBER OF VISIBLE ADJACENT POINT > 1?
NO
YES
1606
INTERPOLATE POINT COLOR FROM VISIBLE NEIGHBORING POINTS
1608
MARK POINT AS VISIBLE
1610
DELETE POINT FROM PROCESSING SET

FIG. 17

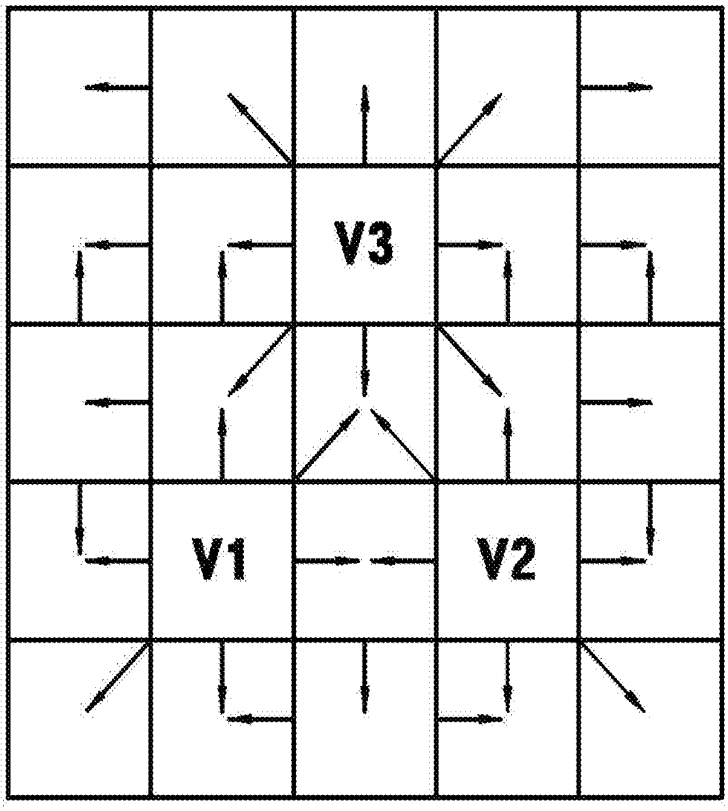


FIG. 18

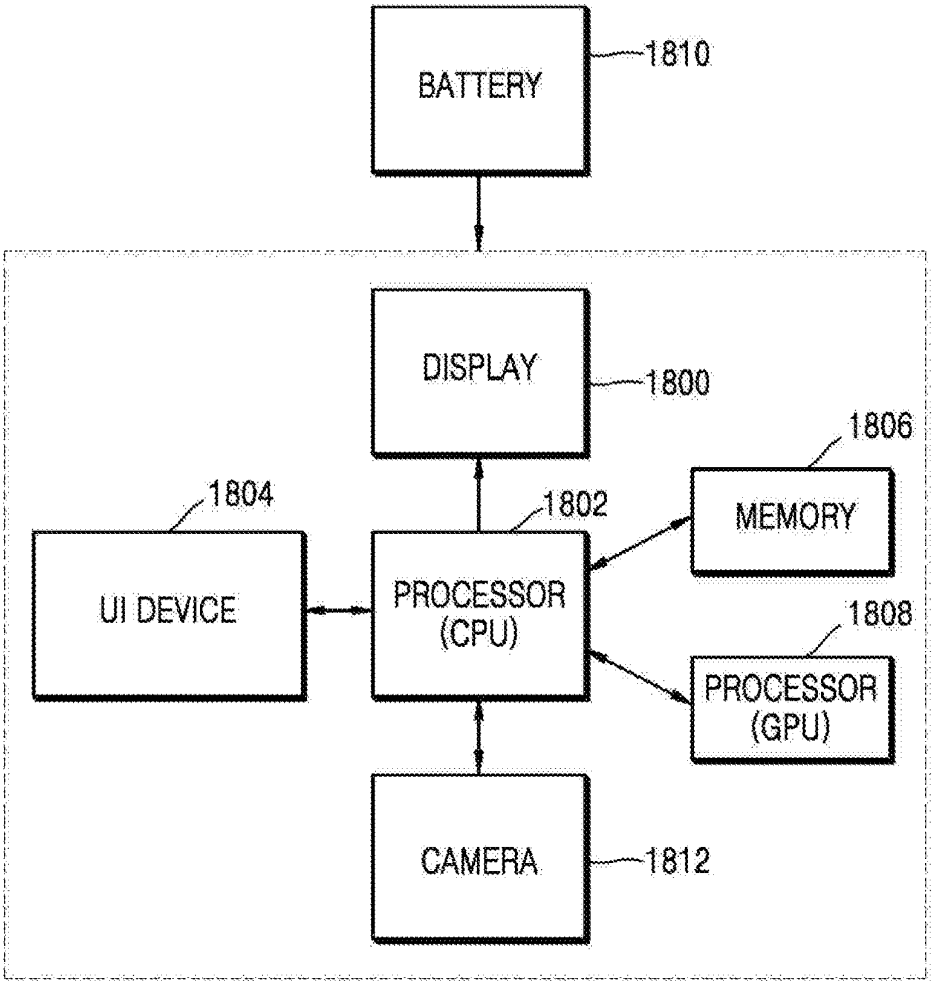
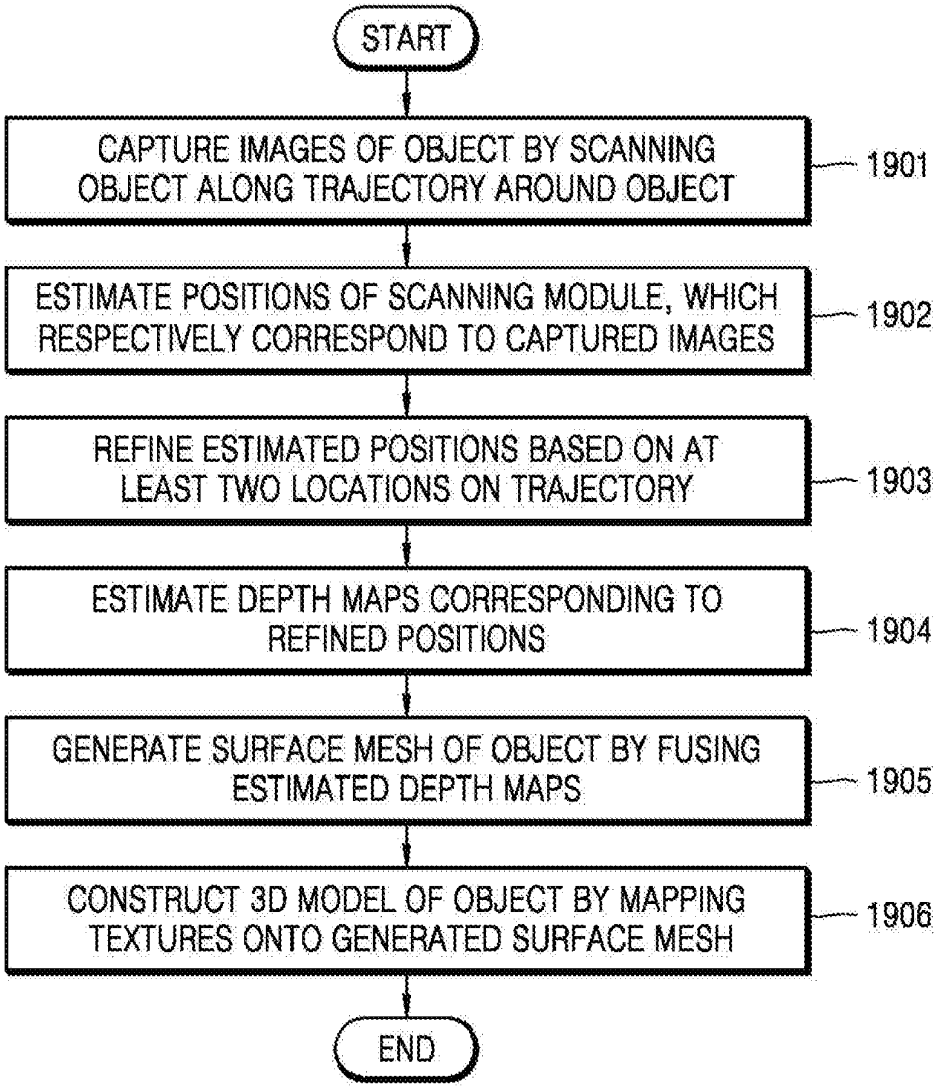

BATTERY
1810
DISPLAY
1800
1806
MEMORY
1804
1802
UI DEVICE
PROCESSOR (CPU)
1808
PROCESSOR (GPU)
CAMERA
1812

FIG. 19


START
CAPTURE IMAGES OF OBJECT BY SCANNING OBJECT ALONG TRAJECTORY AROUND OBJECT
1901
ESTIMATE POSITIONS OF SCANNING MODULE, WHICH RESPECTIVELY CORRESPOND TO CAPTURED IMAGES
1902
REFINE ESTIMATED POSITIONS BASED ON AT LEAST TWO LOCATIONS ON TRAJECTORY
1903
ESTIMATE DEPTH MAPS CORRESPONDING TO REFINED POSITIONS
1904
GENERATE SURFACE MESH OF OBJECT BY FUSING ESTIMATED DEPTH MAPS
1905
CONSTRUCT 3D MODEL OF OBJECT BY MAPPING TEXTURES ONTO GENERATED SURFACE MESH
1906
END

METHOD AND APPARATUS FOR CONSTRUCTING THREE DIMENSIONAL MODEL OF OBJECT

CROSS-REFERENCE TO RELATED APPLICATION(S)

[0001] This application claims the benefit under 35 U.S.C. §119(a) of a Russian patent application filed on Aug. 14, 2015 in the Federal Service for Intellectual Property (Rospatent) and assigned Serial number 2015134345, and of a Korean patent application filed on May 13, 2016 in the Korean Intellectual Property Office and assigned Serial number 10-2016-0058779, the entire disclosure of each of which is hereby incorporated by reference.

TECHNICAL FIELD

[0002] The present disclosure relates to image processing. More particularly, the present disclosure relates to a method and apparatus for constructing a three-dimensional (3D) model of an object by using images generated by scanning the object in 360°.

BACKGROUND

[0003] Three-dimensional (3D) reconstruction is a process of computing mathematical representation of a 3D surface of an object from two-dimensional (2D) projections of the object obtained from different points of view.

[0004] Generally, the 3D reconstruction is based on various optical systems used for image capturing. The optical system may be configured either as a depth camera (for example, an active depth sensor or stereo camera), or as a monocular camera.

[0005] The depth camera may capture depth maps of an object or environment in real time. Here, the depth maps may be captured immediately in case of the active depth sensor or after processing a pair of rectified stereo images in case of the stereo camera. Each pixel in the depth maps corresponds to a distance from one point of the object to the depth camera. However, when the depth camera is included in a device to capture the depth maps, many hardware resources related to a depth sensor is required, and thus hardware configuration of the device may be complex.

[0006] A silhouette-based 3D reconstruction approach may use monocular camera-based approaches. In the silhouette-based 3D reconstruction approach, silhouettes of an image are extracted and then a volume structure of the object is formed. According to the silhouette-based 3D reconstruction approach, a polygonal mesh with color texture is generated. However, in the silhouette-based 3D reconstruction approach, complicated computations related to extraction of silhouettes and fusion of volume data need to be performed in a device having low processing capability, such as a mobile device. Also, in the silhouette-based 3D reconstruction approach, concavities that are not seen in silhouette images are unable to be recovered.

[0007] An example of a method using monocular camera-based hardware includes a structure-from-motion (SfM)-based 3D reconstruction process. In the SfM-based 3D reconstruction process, trajectories of extracted feature points are used to reconstruct camera motion and 3D positions of an object. However, in the SfM-based 3D reconstruction process, since a 3D point cloud or polygonal mesh is reconstructed without texture, visual appearance of a reconstructed 3D model is not sufficient. Also, in the SfM-based 3D reconstruction process, since a large number of feature points need to be processed in order to increase accuracy, computational workloads may increase.

[0008] Since the 3D reconstruction approaches described above require many complex computations, it may be infeasible to implement the 3D reconstruction approaches on mobile platforms having limited process performances, such as a smart phones or tablets. Cloud-based data processing may be used to overcome the limited process performances of the mobile platforms, but in this case, the mobile platforms need to be network accessible all the time and may be time consuming compared to the 3D reconstruction approaches described above.

[0009] Thus, there is a need for techniques of 3D reconstruction of an object, which may be performed by using less time and suitable computational workloads, and also without use of additional hardware resources.

[0010] The above information is presented as background information only to assist with an understanding of the present disclosure. No determination has been made, and no assertion is made, as to whether any of the above might be applicable as prior art with regard to the present disclosure.

SUMMARY

[0011] Aspects of the present disclosure are to address at least the above-mentioned problems and/or disadvantages and to provide at least the advantages described below. Accordingly, an aspect of the present disclosure is to provide methods and apparatuses for constructing a three-dimensional (3D) model of an object.

[0012] Another aspect of the present disclosure is to provide non-transitory computer-readable recording media having recorded thereon programs, which when executed by a computer, perform the methods.

[0013] In accordance with an aspect of the present disclosure, a method of constructing a 3D model of an object is provided. The method includes capturing images of the object by scanning the object along a trajectory around the object, estimating positions of a scanner, which respectively correspond to the captured images, refining the estimated positions of the scanner based on at least two locations on the trajectory, estimating depth maps corresponding to the refined positions of the scanner, generating a surface mesh of the object by fusing the estimated depth maps, and constructing and displaying the 3D model of the object by mapping textures onto the generated surface mesh.

[0014] In accordance with another aspect of the present disclosure, an apparatus for configuring a 3D model of an object is provided. The apparatus includes a scanner configured to capture images of the object by scanning the object along a trajectory around the object, at least one processor configured to estimate positions of the scanner, which respectively correspond to the captured images, refine the estimated positions of the scanner based on at least two locations on the trajectory, estimate depth maps corresponding to the refined locations of the scanner, generate a surface mesh of the object by fusing the estimated depth maps, and construct the 3D model of the object by mapping textures onto the generated surface mesh, and a display configured to display the 3D mode of the object.

[0015] Other aspects, advantages, and salient features of the disclosure will become apparent to those skilled in the art from the following detailed description, which, taken in

conjunction with the annexed drawings, discloses various embodiments of the present disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The patent or application file contains at least one drawing executed in color. Copies of this patent or patent application publication with color drawing(s) will be provided by the Office upon request and payment of the necessary fee.

[0017] The above and other aspects, features, and advantages of certain embodiments of the present disclosure will be more apparent from the following description taken in conjunction with the accompanying drawings, in which:

[0018] FIG. **1**A is a diagram for describing a three-dimensional (3D) model of an object displayed on a device according to an embodiment of the present disclosure;

[0019] FIG. **1**B is a diagram for describing scanning of an object by using a device for 3D reconstruction according to an embodiment of the present disclosure;

[0020] FIG. **2**A is a diagram for describing a 3D reconstruction process according to an embodiment of the present disclosure;

[0021] FIG. **2**B is a flowchart of a 3D reconstruction process according to an embodiment of the present disclosure;

[0022] FIG. **3** is a diagram for describing estimating of a camera position according to an embodiment of the present disclosure;

[0023] FIG. **4** is a diagram for describing estimating of a camera position by processing a new image according to an embodiment of the present disclosure;

[0024] FIG. **5** is a diagram for describing sparse image alignment according to an embodiment of the present disclosure;

[0025] FIG. **6** is a diagram for describing sparse feature alignment according to an embodiment of the present disclosure;

[0026] FIG. **7** is a diagram for describing local bundle adjustment according to an embodiment of the present disclosure;

[0027] FIG. **8** is a diagram for describing updating of depth-filters according to an embodiment of the present disclosure;

[0028] FIG. **9** is a diagram for describing camera position refinement according to an embodiment of the present disclosure;

[0029] FIG. **10** is a diagram for describing depth map estimation according to an embodiment of the present disclosure,

[0030] FIG. **11** is a diagram for describing depth map fusion according to an embodiment of the present disclosure;

[0031] FIG. **12** is a diagram for describing surface reconstruction according to an embodiment of the present disclosure;

[0032] FIG. **13** is a diagram for describing texture mapping according to an embodiment of the present disclosure;

[0033] FIG. **14** is a diagram for describing a z-buffer face visibility check according to an embodiment of the present disclosure;

[0034] FIG. **15** is a diagram for describing face labeling according to an embodiment of the present disclosure;

[0035] FIG. **16** is a diagram for describing hole filling according to an embodiment of the present disclosure;

[0036] FIG. **17** is a diagram of a structure of a block according to an embodiment of the present disclosure;

[0037] FIG. **18** is a diagram of a hardware structure of a device according to an embodiment of the present disclosure; and

[0038] FIG. **19** is a flowchart of a method of reconstructing a 3D model of an object according to an embodiment of the present disclosure.

[0039] Throughout the drawings, like reference numerals will be understood to refer to like parts, components, and structures.

DETAILED DESCRIPTION

[0040] The following description with reference to the accompanying drawings is provided to assist in a comprehensive understanding of various embodiments of the present disclosure as defined by the claims and their equivalents. It includes various specific details to assist in that understanding but these are to be regarded as merely exemplary. Accordingly, those of ordinary skill in the art will recognize that various changes and modifications of the various embodiments described herein can be made without departing from the scope and spirit of the present disclosure. In addition, descriptions of well-known functions and constructions may be omitted for clarity and conciseness.

[0041] The terms and words used in the following description and claims are not limited to the bibliographical meanings, but, are merely used by the inventor to enable a clear and consistent understanding of the present disclosure. Accordingly, it should be apparent to those skilled in the art that the following description of various embodiments of the present disclosure is provided for illustration purpose only and not for the purpose of limiting the present disclosure as defined by the appended claims and their equivalents.

[0042] It is to be understood that the singular forms "a," "an," and "the" include plural referents unless the context clearly dictates otherwise. Thus, for example, reference to "a component surface" includes reference to one or more of such surfaces.

[0043] Also, when a part "includes" or "comprises" an element, unless there is a particular description contrary thereto, the part may further include other elements, not excluding the other elements. In the following description, terms such as "unit" and "module" indicate a unit for processing at least one function or operation, wherein the unit and the block may be embodied as hardware or software or may be embodied by combining hardware and software.

[0044] The terms "includes" or "comprises" used herein should not be construed as necessarily including several components or several operations described in the specification, and some components or some operations may not be included or additional components or additional operations may be further included.

[0045] Reference will now be made in detail to embodiments, examples of which are illustrated in the accompanying drawings, wherein like reference numerals refer to like elements throughout. In this regard, the present embodiments may have different forms and should not be construed as being limited to the descriptions set forth herein. Accordingly, the embodiments are merely described below, by referring to the figures, to explain aspects.

[0046] FIG. **1**A is a diagram for describing a three-dimensional (3D) model **103** of an object **101** displayed on a device **106**, according to an embodiment of the present disclosure.

[0047] Referring to FIG. **1**A, a user **100** captures appearance images of the object **101** having a shape of a dummy by using the device **106**, so as to reconstruct the 3D model **103** of the object **101**. A scanning module (for example, a camera) of the device **106** obtains images of the object **101** while the user **100** moves around the object **101** and holds the scanning module (the camera) of the device **106** to face the object **101**.

[0048] The device **106** extracts object data from the images by using an internal processor, such as a central processing unit (CPU) or a graphics processing unit (GPU), and generates the 3D model **103** by performing a reconstruction process on the object data. Similarly to the object **101**, the 3D model **103** may be reconstructed to a representation of a texturized mesh. Meanwhile, in one or more embodiments, the term "reconstruction" means that an actual object is processed to a virtual 3D model or texturized mesh displayable on a display screen, and may be replaced by a similar term, such as "generation" or "construction" of a 3D model or texturized mesh.

[0049] The 3D model **103** is displayed on a display screen of the device **106**. The user **100** may turn the 3D model **103** around by touching the display screen to check the 3D model **103** of the object **101**.

[0050] FIG. **1**B is a diagram for describing scanning of an object **102** by using a device **106** for 3D reconstruction according to an embodiment of the present disclosure.

[0051] Referring to FIG. **1**B, it is assumed that the user **100** of the device **106** is reconstructing a 3D model of the object **102** having a shape of a dummy

[0052] Processes for 3D reconstruction begin by scanning the object **102** using the scanning module of the device **106**. The scanning module may correspond to a camera included in the device **106** that is hand-held, such as a mobile phone or a tablet. The scanning module may be a monocular camera, a multi-view stereo camera, a depth sensor, or a combination thereof, but a type of the scanning module is not limited thereto.

[0053] For convenience of scanning, the object **102** may be placed on a pedestal **104**. While scanning the object **102**, the scanning module of the device **106** may always face the object **102**. At this time, the user **100** moves along a trajectory **108** while holding the scanning module of the device **106** to face the object **102**. In FIG. **1**B, although the trajectory **108** is shown to be a circular trajectory in a counterclockwise direction, but the trajectory **108** is not limited thereto and may vary. For example, the trajectory **108** may have a closed loop or an opened loop. Alternatively, the trajectory **108** may have a circular shape or another arbitrary shape. While scanning the object **102**, the user **100** may form the trajectory **108** by moving clockwise, counterclockwise, or both clockwise and counterclockwise.

[0054] After scanning the object **102** is finished, the user **100** stops capturing images of the object **102**. In other words, the device **106** stops scanning the object **102** after the user **100** went around the trajectory **108**. Once the capturing of the images is completed, the device **106** executes software for processing the captured images to perform a post-processing stage. A result of performing the post-processing stage is a texturized mesh, and may be the 3D model of the object **102**. Operations performed during the post-processing stage will be described in detail later. Finally, the texturized mesh (i.e., a reconstructed 3D model) is stored in the device **106**.

[0055] FIG. **2**A is a diagram for describing a 3D reconstruction process according to an embodiment of the present disclosure.

[0056] Referring to FIG. **2**A, the 3D reconstruction process includes a real-time stage **21** and a post-processing stage **22**.

[0057] During the real-time stage **21**, the device **106** captures images **201** of the object **102** while being moved by a user along a trajectory around the object **102**. Each of the images **201** includes information about the object **102** and a background.

[0058] During the post-processing stage **22**, the device **106** first generates a depth map **205** regarding the object **102** included in the images **201**. Since the images **201** are captured in different camera orientations, the depth map **205** may include information about a depth of the object **102** in each camera orientation. Then, the device **106** reconstructs a representation of a 3D mesh **211** regarding the object **102** by fusing the depth maps **205** respectively corresponding to the camera orientations. Finally, the device **106** generates a texturized 3D mesh **214** by texturing the 3D mesh **211** based on texture information of the object **102** obtained from the images **201**. The texturized 3D mesh **214** may be a representation of a reconstructed 3D model of the object **102**.

[0059] FIG. **2**B is a flowchart of a 3D reconstruction process according to an embodiment of the present disclosure.

[0060] Referring to FIG. **2**B, the 3D reconstruction process may include two main stages, i.e., the real-time stage **21** and the post-processing stage **22**.

[0061] The real-time stage **21** includes a sub-stage **200** at which images of the object **102** are captured as described above with reference to FIG. **1**B, and a sub-stage **202** at which camera positions are estimated by using the captured images.

[0062] The estimated camera positions may be characterized by a set of different parameters. The set of parameters may include camera coordinates and camera orientations in a space. The camera orientation may be defined by yaw, pitch, and roll angles. The sub-stage **202** may be performed by using a simultaneous localization and mapping (SLAM) system. By using the SLAM system, it is possible to estimate not only a trajectory of a camera but also an orientation of the camera at each moment of time by using the camera coordinates and parameters of the camera orientations. By using the SLAM system, a robust camera position may be estimated under any capturing conditions, such as translation, rotation, and shaking movements. The real-time stage **21** may be performed by the CPU of the device **106**.

[0063] After the real-time stage **21** is completed, the post-processing stage **22** is performed. The post-processing stage **22** may be performed by both of the CPU and the GPU of the device **106**, and usually takes several seconds.

[0064] It is known that the SLAM system usually leads to a significant drift in a trajectory. Accordingly, a complete circular trajectory may be estimated as an incomplete arc. Thus, in order to increase accuracy of the estimated camera positions (both the trajectory and the orientation of the camera), the estimated camera positions are refined in a

sub-stage **204**. For the refining, all types of available information related to a start point and an end point of the trajectory may be used.

[0065] The camera positions may be estimated in two operations in a coarse-to-fine manner. Coarse estimation is performed by using the SLAM system while the object **102** is scanned. Then, fine estimation is performed by refining coarse-estimated camera positions based on the information related to the start and end points of the trajectory, for compensation for drift caused by configuration peculiarities of the SLAM system.

[0066] During a sub-stage **206**, depth maps are generated based on the refined camera positions and the captured images.

[0067] During a sub-stage **208**, the depth maps obtained from the different camera positions are fused.

[0068] During a sub-stage **210**, a surface mesh, i.e., a surface reconstruction, is generated by fusing the depth maps. The surface mesh is a 3D representation of the object **102**.

[0069] During a sub-stage **212**, textures are mapped onto the surface mesh. By mapping the textures onto the surface mesh, the texturized 3D mesh (or a 3D model) **214** that is a 3D reconstruction result of the object **102** is generated. The texturized 3D mesh **214** may be output through a display of the device **106**.

[0070] Fast 3D model reconstruction may be performed by extracting depth maps at a high speed and fusing the depth maps using the CPU and the GPU of the device **106**. Also, high quality 3D model reconstruction may be provided via accurate surface reconstruction and texture mapping.

[0071] Hereinafter, the sub-stages **200** through **212** in the 3D reconstruction process will be described in detail.

[0072] FIG. **3** is a diagram for describing estimating of a camera position according to an embodiment of the present disclosure. Camera position estimation shown in FIG. **3** may be related to the sub-stage **202** of FIG. **2**B.

[0073] Referring to FIG. **3**, the camera position estimation may begin by performing map initialization **302** using a first image **300**. The map initialization **302** denotes initialization of the camera position estimation. A world reference frame for the map initialization **302** is assigned to be the same as a reference frame of the first image **300**. A features from accelerated segment test (FAST) algorithm using corner detection may be used to extract key points from the first image **300**. The FAST algorithm is disclosed in, for example, Edward Rosten and Tom Drummond, "Fusing points and lines for high performance tracking", In Computer Vision, 2005, ICCV 2005, Tenth Institute of Electrical and Electronics Engineers (IEEE) International Conference, volume 2, pages 1508-1515, IEEE, 2005. A mean scene depth is assigned to each extracted key point. A map database (DB) **304** stores the key points extracted according to the FAST algorithm together with the assigned mean scene depth. The key points in the map DB **304** are stored in the word reference frame that is the same as the reference frame of the first image **300**.

[0074] All sequential images **306** captured by a camera (i.e., the scanning module) of the device **106** are used together with maps stored in the map DB **304** to determine which frame (or image) corresponds to a key frame for tracking and 3D reconstruction. Camera positions corresponding to the key frames may be estimated in operation **308** by updating the maps stored in the map DB **304**. The results of the sub-stage **202** of FIG. **2**B are the key frames and the estimated camera positions **310**.

[0075] FIG. **4** is a diagram for describing estimating of a camera position by processing a new image **404**, according to an embodiment of the present disclosure.

[0076] Referring to FIG. **4**, camera position estimation may include a motion estimation stage **41** and a checking stage **42**. The SLAM system described above with reference to FIG. **3** may also be used in the camera position estimation of FIG. **4**. By using the camera position estimation of FIG. **4**, not only a trajectory of the camera, but also a camera orientation at each moment of time may be estimated.

[0077] The motion estimation stage **41** includes a sub-stage **400** for sparse image alignment, a sub-stage **402** for sparse feature alignment, and a sub-stage **408** for local bundle adjustment.

[0078] During the sub-stage **400** for the sparse image alignment, a camera position corresponding to the new image **404** is estimated based on key points of a map stored in the map DB **304**. The key points of the map are projected onto the new image **404**, and the camera position corresponding to the new image **404** is estimated by using a Direct Image Alignment algorithm.

[0079] During the sub-stage **402** for the sparse feature alignment, the new image **404** and the camera position corresponding to the new image **404** are used. The key points of the map are projected onto the new image **404**, and a position of each key point in the new image **404** is refined by using a Lucas Kanade tracker (LKT or KLT). The LKT tracks a patch (rectangular area) around a projection of the key point by using gradient descent. The LKT is disclosed in, for example, Simon Baker and lain Matthews, "Lucas-kanade 20 years on: A unifying framework", International journal of computer vision, 56(3):221-255, 2004.

[0080] During the sub-stage **408** for the local bundle adjustment, the new image **404**, the camera position corresponding to the new image **404**, and a list of map points are used together with the refined positions of the key points of the new image **404**. The camera position is calculated based on initial approximation obtained during the sub-stage **400**, point coordinates stored in the map DB **304**, and the refined positions of the key points obtained in the sub-stage **402**, which is a classical Perspective-n-Point (PnP) problem. The calculated camera position may be assigned to the new image **404**, and used during another stage (or another sub-stage) without further changes. After refining the camera position, 3D world coordinates of the map points may also be refined.

[0081] The new image **404** after the sub-stage **408** may be set as a last camera image **406** to be additionally used during the sub-stages **400** and **402**.

[0082] The new image **404** and the camera position corresponding to the new image **404** are stored in an image queue **410**. Images in the image queue **410** may be used during the checking stage **42** later.

[0083] During a sub-stage **412** of the checking stage **42**, it is checked whether an image input from the image queue **410**, for example, the new image **404**, is a key frame.

[0084] When it is checked that the new image **404** is a key frame, the new image **404** is stored in a list of the key frames and estimated camera positions **310**, together with the camera position corresponding to the new image **404**. Then, during a sub-stage **414**, feature extraction is performed on the new image **404**.

[0085] During a sub-stage **420**, depth-filters with a depth expectation equal to a current mean scene depth is created for each feature point and stored in a list **422** of active depth-filters for depth-filtering. Examples of the depth-filtering are disclosed in "Video-based, real-time multi-view stereo" by George Vogiatzis and Carlos Hernandez, Image and Vision Computing, 29(7):434-441, 2011.

[0086] In order to satisfy 360° scanning conditions, the depth-filters are prolongated from previous key frames. Patches of the depth-filters that are not processed yet are aligned on a last key frame, and a new patch is bound to each depth-filter while a depth-filter state is unchanged.

[0087] When it is checked that the new image **404** is not a key frame, a sub-stage **416** for updating the depth-filters is performed. During the sub-stage **416**, a search along an epipolar line is performed to align a feature point from the list **422** with a point in the new image **404**. After the alignment, a point depth is calculated by using triangulation techniques. The point depth is registered in the depth-filter of the corresponding point. Then, during a sub-stage **418**, each depth-filter is checked for convergence. If a depth-filter is converged (for example, when a depth variation is sufficiently small), an associated point is added to the map DB **304** and excluded from the list **422** of active depth-filters.

[0088] The motion estimation stage **41** and the checking stage **42** are performed on each captured image. After all captured images are processed, a full list of the key frames **310** and the estimated camera positions **310** are used during the sub-stage **204** (camera position refinement) described with reference to FIG. **2**B.

[0089] FIG. **5** is a diagram for describing sparse image alignment **400** according to an embodiment of the present disclosure.

[0090] Referring to FIG. **5**, in operation **500**, visible points for the last camera image **406** are extracted from the map DB **304**. The visible points may be filtered to satisfy the 360° scanning conditions. Meanwhile, in order to satisfy the 360° scanning conditions, points captured under significantly different directions of view may be excluded because images including such points are largely different from an original patch and tracking is not reliable. Also, points that are occluded by other points may also be excluded. A z-buffer technique may be used to filter such points.

[0091] In operation **502**, patches of all filtered points are combined in one feature vector. In operation **504**, the feature vector is tracked by using the LKT in the new image **404**. Then, a camera position **506** of the new image **404** is obtained from the LKT.

[0092] FIG. **6** is a diagram for describing sparse feature alignment **402** according to an embodiment of the present disclosure.

[0093] Referring to FIG. **6**, in operation **600**, visible points for the last camera image **406** are extracted from the map DB **304**, taking into account the estimated camera position. In operation **602**, a patch is independently created for each feature point. In operation **604**, each patch is tracked by using the LKT in the new image **404**. An updated position of each feature point in the new image **404** is obtained in operation **606**.

[0094] FIG. **7** is a diagram for describing local bundle adjustment **408** according to an embodiment of the present disclosure.

[0095] Referring to FIG. **7**, in operation **700**, the camera position **506** of the new image **404** is updated with a PnP procedure based on the initial approximation obtained in the sub-stage **400** and the updated position **606** of each feature point in the new image **404** obtained in the sub-stage **402**. As such, an updated camera position **706** of the new image **404** is obtained. The updated camera position **706** is assigned to the new image **404**.

[0096] In operation **702**, points used several times during the sub-stage **402** are selected for structure optimization. In operation **704**, word 3D coordinates of the selected points are updated by using bundle adjustment techniques using the updated positions **606** of the feature points and the updated camera position **706**. The updated world 3D coordinates are stored in the map DB **304**.

[0097] FIG. **8** is a diagram for describing a sub-stage **416**, i.e., an updating of depth-filters according to an embodiment of the present disclosure.

[0098] Referring to FIG. **8**, the depth-filters are updated for each image that is not a key frame in the image queue **410**. An epipolar line in a new image is calculated for each depth-filter in the list **422**. In operation **800**, a linear search around a depth expectation along the epipliar line is performed. In operation **802**, the LKT searches for optimal alignment of points from the depth-filter around an epipliar search minimum. When the optimal alignment is not found, the depth-filter is updated with a fault flag **808**. When the optimal alignment is found, a depth is found by using a triangulation procedure in operation **804**. In operation **806**, the depth-filter is updated to the determined depth. The updating of the depth-filter is used to determine new mathematical expectation and variance of a depth value for the depth-filter.

[0099] FIG. **9** is a diagram for describing a sub-stage **204**, i.e., camera position refinement according to an embodiment of the present disclosure.

[0100] When the real-time stage **21** including the sub-stages **200** and **202** are finished, the post-processing stage **22** of the 3D reconstruction process using the CPU and the GPU of the device **106** begins as shown in FIG. **2**B.

[0101] The sub-stage **204** may be executed by using the CPU of the device **106**. The camera positions obtained by the SLAM system may include a significant drift and may be needed to be refined before additional processing. Such a drift may be expressed as a systematic error in estimating of x, y, and z coordinates and yaw, pitch, and roll angles of the camera. The camera positions are refined to compensate for the drift, thereby increasing the overall accuracy of the camera positions. If the trajectory **108** is circular, the trajectory **108** may have natural loop closure points. Once a loop closure point is detected, it is possible to estimate and compensate for an accumulated drift of the camera positions. The camera position refinement includes four operations, i.e., loop closure in operation **900**, loop correction in operation **902**, structure re-calculation in operation **904**, and bundle adjustment in operation **906**.

[0102] Referring to FIG. **9**, in operation **900**, a loop point of a trajectory is sought, and for each received key frame, it is checked whether the loop point is a candidate for a loop closure point. For each key frame, a relative Euclidean distance in a SE3 space to other key frames is calculated. When the calculated relative Euclidean distance is equal to or smaller than a threshold value ξ_{sim}, it is checked how many common map points the key frames have. When to key frames KF_i and KF_k have more map points than n_{KFpt}, a relative pose constraint τ_{ik} is computed using the PnP

algorithm from common point projections. Two key frames that are spatially adjacent to each other have a few or no common points. Thus, these two key frames are considered as candidates for a loop closure point. For performing loop closure, it is required to compute a position constraint using the PnP algorithm, which requires feature correspondences. At locations of point projections of the key frame KF_i, a set of binary robust invariant scalable keypoints (BRISK) features ft_i invariant to scale and rotation are extracted, and from the key frame KF_i, a set of similar features ft_k are extracted at locations of FAST corners. BRISK is disclosed in Stefan Leutenegger, Margarita Chli, and Roland Yves Siegwart. Brisk: Binary robust invariant scalable keypoints. in computer vision (ICCV), 2011 IEEE International Conference on, pages 2548-2555. IEEE, 2011. The BRISK features have limited viewpoint change invariance. Since the trajectory is circular, viewpoint change when revising the same part of trajectory is very little.

[0103] From the two sets described above, feature correspondences are found. Since 3D coordinates for features in the set ft_i are known, the position of the key frame KF_k relative to KF_i may be computed using the PnP algorithm, resulting in a forward constraint τ_{ik}^{ft}. The same procedure is performed for KF_k, resulting in a backward constraint τ_{ki}^{ft}. Feature matching may give wrong correspondences that may result in a wrong PnP solution. In order to avoid the inclusion of wrong PnP solutions into the camera position refinement, a reciprocal check is performed: according to properties of a Lie group, if there are two camera positions containing no error, forward and backward constraints should agree: $\xi_{ik}^{ft}=(\xi_{ki}^{ft})^{-1}$. Thus a disagreement between the forward and backward constraints may be computed as $err_{rc}=\|\ln(\xi_{ik}^{ft}\cdot(\xi_{ki}^{ft})^{-1})\|$. If the disagreement err_{rc}, is equal to or smaller than a threshold value err_{rc}^{th}, the forward constraint ξ_{ik}^{ft} and backward constraint ξ_{ki}^{ft} are added into optimization; otherwise, the forward constraint ξ_{ik}^{ft} and backward constraint ξ_{ki}^{ft} are considered as invalid and are ignored.

[0104] The forward and backward constraints computed in operation **900** are used during operation **902**. Due to the drift for a first key frame and a loop closure key frame (an end key frame), the forward and backwards constraints disagree. In order to compensate for the drift, the camera position corresponding to each key frame is re-estimated such that the disagreement is distributed among frames. A SLAM approach provides a variance to each camera position. The variance indicates to what extend the camera position is corrupted by measurement noise and a camera drift. Key frames having a high position variance are provided with stronger position correction, while key frames having a low variance are provided with less position correction. As a result, the accuracy of the camera positions may be increased due to drift removal.

[0105] Optimized camera positions are then calculated in operation **904**, where 3D coordinates of points are updated due to changes in the camera positions. The optimized camera positions are calculated such that a depth (z coordinate) of a point with respect to the first key frame is the same as before loop correction.

[0106] In operation **906**, final refinement of the camera positions is performed to decrease the measurement noise. For each key frame KF_i, a projection uv_m^i of point m obtained through alignment in the SLAM approach is compared to a predicted projection computed according to $uvp_m^i=\pi(SE3_i*p_m)$, and a point re-projection error is calculated according to $err(m,i)=\|uv_m^i-uvp_m^i\|$. total re-projection error is computed according to $err_{ba}=\Sigma_{i=0,\ m=0}^{i=m,m=M}err(m,i)$. The total re-projection error accounts for the measurement noise presented in the camera positions. The bundle adjustment reduces the total re-projection error by jointly optimizing the camera positions and the 3D coordinates of the points. The bundle adjustment is done by performing Gaussian optimization through an iterative correction of the 3D coordinates of the points and the camera positions in a direction of decreasing the total re-projection error. After the optimization, a set of key frames with the refined camera positions is used in operation **908** for further processing.

[0107] FIG. **10** is a diagram for describing a sub-stage **206**, i.e., depth map estimation according to an embodiment of the present disclosure.

[0108] Referring to FIG. **10**, the key frames with the refined camera positions, which are obtained as results of performing the sub-stage **204**, are used as inputs of the sub-stage **206**. In operation **1000** for stereo pair selection, for each key frame, another frame is selected and a non-rectified stereo pair is formed. Second frame selection is based on the camera positions to ensure a sufficient, but not too large translational motion between the images of the stereo pair. The stereo pair and relative camera positions are used for depth map computation.

[0109] The depth map is computed with a coarse-to-fine pyramidal approach. In operation **1002** for coarse depth estimation, low resolution sampling is used with respect to the images from the stereo pair, which results in a low resolution depth map. The low resolution depth map may be further filtered in operation **1004** for depth filtering. Then, on a next pyramid level, finer resolution sampling is used and the low resolution result from the previous operation is up-scaled in operation **1006** and used to estimate a depth map of a current pyramid level. The estimating of the depth map is refined with a finer sampled image on the current pyramid level in operation **1008**, and may be further filtered in operation **1010**. Such an up-sampling procedure may be performed several times via a loop **1012**, until a depth map of desired resolution is obtained or until runtime limits are met.

[0110] On a first (coarsest) pyramid level, only a small number of pixels are processed, so a more complicated and more accurate depth map estimation method may be used. For example, in case of block-matching or cost-volume filtering methods, a larger window size may be used, which usually provides better accuracy at the cost of bigger runtime.

[0111] Operation **1002** for the coarse depth estimation may include sub-stages, i.e., estimation of a search range and sampling, computation of matching costs, aggregation of costs, disparity estimation based on cost minimization, and depth triangulation. The search range is estimated from given estimations of possible near and far depth values for a current pixel in a first image. Here, the current pixel may be obtained from a sparse point cloud of a reconstructed scene created during the SLAM approach. The depth values allow determining a search segment on an epipolar line corresponding to the current pixel. Then, pixel sampling may be used with respect to a search segment on a second image in order to determine possible locations of a second image point corresponding to the current pixel in the first image. The matching cost is computed from a pixel-wise

color difference or other distance metric on an image color or gradient. The cost aggregation is performed by a weighted accumulation of costs in some regions around the current pixel. The disparity is estimated by a winner-takes-all (WTA) cost minimization strategy along search segments on the epipolar line. The depth is triangulated from corresponding point locations on both images of the stereo pair using epipolar constraints.

[0112] Despite that the images in the stereo pair are not rectified, the current algorithm provides very efficient parallel implementation on the GPU of the device **106**. Memory consumption may be also reduced because there is no need to store full cost-volume in the memory. The image may be processed in small regions and for each of the regions, matching cost values may be stored in a GPU local memory.

[0113] Since the computed depth maps are used in depth map fusion, wrong depth map estimations may be filtered out as much as possible and only the depth values which are accurate with highest confidence may be left, because fusion methods may usually handle well missing data by interpolation or propagation from existing values, but are more sensitive to heavy outliers. Depth outliers may be efficiently filtered out by analyzing accumulated ratios of cost to a minimal value for current pixel during WTA cost minimization. Additional filtering may be performed with left-right consistency check. Outlier filtering may be also incorporated into the coarse-to-fine strategy, by performing it on each pyramid level and thus reducing computational effort on finer pyramid levels.

[0114] Operations **1004** and **1010** for depth filtering includes sub-stages, i.e., filtering by photometric ambiguity and left-right consistency check. The photometric ambiguity is estimated during WTA cost minimization and is calculated by analyzing accumulated ratios of cost to a minimal value for the current pixel. When texture is absent or ambiguous (periodic along the epipolar line), many costs will have similar values to the minimal cost, causing ratios to be around 1, which allows to filter these ambiguities by analyzing the ratios. The left-right consistency check is performed by analyzing the consistency of both depth maps for the left and right images in the stereo pair. The consistency is determined by checking re-projection errors for each pixel using depth values from both depth maps.

[0115] The depth filtering allows one to significantly reduce the number of pixels to be processed on a next pyramid level. Performing depth filtering on each level significantly reduces a runtime, because finer pyramid levels are typically much slower than coarse pyramid levels.

[0116] The depth map updating on the next pyramid level includes two operations, i.e., coarse depth map up-scaling in operation **1006** and depth map refinement in operation **1008**. A coarse depth map may be up-scaled to finer resolution using various filtering methods, such as nearest-neighbor, bilinear, box filtering, Gauss filtering, color image guided bilateral, or other edge-preserving filtering. In depth up-scaling, simple bilinear filtering with support of missing values is used for performance improvement. The up-scaled depth is used before the depth map refinement in operation **1008**. During each operation, a matching cost is re-evaluated with finer sampling corresponding to the current pyramid level. The corresponding point is found from the minimization of refined cost values among all operations, and the depth is triangulated again. The reduction of a support region for an estimated pixel on each finer pyramid level decreases an effect of smoothing and yields more accurate depth values.

[0117] The result of the sub-stage **206** is a set of depth maps with the camera positions **1014**, and the set of depth maps stores information about a depth of the captured object for every key frame.

[0118] FIG. **11** is a diagram for describing a sub-stage **208**, i.e., depth map fusion, according to an embodiment of the present disclosure.

[0119] Referring to FIG. **11**, in operation **1100** for volumetric depth map integration, the set of depth maps with the camera positions computed in the sub-stage **206** are processed to obtain volumetric depth map integration or fusion. The result of such fusion is used to generate a 3D voxel histogram representation in operation **1102**. For every voxel, an approximate signed distance to the surfaces induced by the depth maps is computed, clamped to a user-supplied threshold value, scaled to an interval (−1, 1), and stored as a histogram of respective frequencies n_j. Each voxel holds a histogram including N bins around evenly spaced bin center c_j. N−2 bins are used to represent distribution of values in the interior of the interval (−1, 1), i.e. voxels close to a supposed surface. Two separate bins are reserved for values indicating a larger distance to the surface, one bin counts (scaled) distances smaller than or equal to −1 (occluded voxel) and one bin represents values greater than or equal to 1 (empty voxel). The total number of histogram counts is limited to the number of the available depth maps.

[0120] In operation **1104** for volumetric histogram optimization, a variational approach is used. In operation **1106**, 3D voxel signed distance function representation (or 3D voxel truncated signed distance function (TSDF) representation) is obtained as an iterative solution of the minimization of an energy function. Here, as described in Christopher Zach, "Fast and high quality fusion of depth maps", In Proceedings of the international symposium on 3D data processing, visualization and transmission (3DPVT), volume 1, CiteSeer, 2008, the energy function may be calculated according to Equation 1 below.

$$E^{TV-L^1} = \int_{\Omega} \left\{ |\nabla u| + \lambda \sum_j n_j |u - c_j| \right\} d^p x$$

[0121] Equation 1

[0122] The result of sub-stage **208** is a TSDF representation of the captured object **102**, which is one of methods for representing an arbitrary 3D surface, widely used in computer graphics.

[0123] FIG. **12** is a diagram for describing a sub-stage **210**, i.e., surface reconstruction, according to an embodiment of the present disclosure.

[0124] Referring to FIG. **12**, operation **1200** for voxel-to-octree representation is performed. The TSDF representation computed in the sub-stage **208** is processed to obtain a voxel-to-octree representation for the object **102**. In operation **1202**, the voxel-to-octree representation is used to obtain an octree-based TSDF representation. A maximum octree depth controls complexity (the number of polygons) of the 3D mesh of the object **102**.

[0125] During operation **1204** for isosurface extraction, a 3D mesh **1206** is reconstructed as described in Michael

Kazhdan, Allison Klein, Ketan Dalal, and Hugues Hoppe, "Unconstrained isosurface extraction on arbitrary octrees", In Symposium on Geometry Processing, volume 7, 2007.

[0126] FIG. **13** is a diagram for describing a sub-stage **212**, i.e., texture mapping, according to an embodiment of the present disclosure.

[0127] Referring to FIG. **13**, the 3D mesh **1206** and the key frames and refined camera positions **908** are used as inputs during the texture mapping. In operation **1300** for z-buffer face visibility check, it is checked from which camera positions each face is visible. The camera is characterized by a camera image and a camera position—in other words, the key frames and the refined camera positions **908**, respectively.

[0128] Each visible mesh face has to be textured by projection to one of the camera images. In operation **1302** for face labeling, it is defined which of the camera images will be used for this purpose. There is a number of patches that are closed areas of mesh faces which are provided with textures from one image. Seams between adjacent patches may be very noticeable. In operation **1304** for global color adjustment, color discontinuities between adjacent patches are adjusted, making the seams less visible.

[0129] In operation **1308** for hole filling, textures for invisible mesh faces are created.

[0130] In operation **1310** for texture atlas generation and mesh parametrization, a texture atlas from the camera images and textures for invisible faces are created. Also, the mesh parameterization, i.e. defining texture coordinates on the texture atlas for each mesh face, is performed to obtain the texturized 3D mesh **214**.

[0131] FIG. **14** is a diagram for describing operation **1300**, i.e., the z-buffer face visibility check, according to an embodiment of the present disclosure.

[0132] The 3D mesh **1206** and the key frames and refined camera positions **908** are used as inputs for the z-buffer visibility check. For each camera position, it is determined what mesh faces are visible.

[0133] Referring to FIG. **14**, in operation **1400**, a z-buffer is generated, and an image having the same size as a camera image, each pixel of which contains a minimal depth of faces that are projected in this pixel, is generated. A depth of a mesh face is a maximal depth of vertices of the mesh face in current camera coordinates.

[0134] In operation **1402**, next points are separated into visible and invisible. A point p is marked as visible from a camera position r, if p may be projected in a pixel pxl on a camera r image and $z_p \le z_{buf}(\text{pxl})$ is satisfied. Here, z_p is a depth of the point p in camera r coordinates and $z_{buf}(\text{pxl})$ is a value of the z-buffer in pxl.

[0135] Each mesh face f_i is assigned as visible from the camera position r in operation **1408** if all f_i vertices are visible from the camera position r and $\cos\phi<0$ is satisfied in operation **1404**, and if not, the mesh face f_i is assigned as invisible from the camera position r in operation **1406**. Here, ϕ is an angle between the camera position r in a view direction and the mesh face f_i is an angle of face normal.

[0136] As a result, two lists are formed: a list **1410** of visible mesh faces with information from which camera positions the mesh faces are visible, and a list **1412** of invisible mesh faces.

[0137] FIG. **15** is a diagram for describing operation **1302**, i.e., the face labeling, according to an embodiment of the present disclosure.

[0138] In operation **1302**, a labeling vector l **1504** that defines, for each visible mesh face f_i from the list **1410**, textures from a camera image l_i from the key frames **908** is computed. The labeling vector l **1504** is found by Markov random field (MRF) energy minimization, and Equation 2 below may be used.

$$E(l) = \sum_{f_i \in Faces} w_i^{l_i} + \alpha \sum_{(f_i,f_j)\in Edges} w_{i,j}^{l_i,l_j} \to \min_{l} \qquad \text{Equation 2}$$

[0139] The first term defines the quality of texture fragments used for texturing. In this case, $w_i^{l_i}=\sin^2\phi$. Here, ϕ denotes an angle between the camera image l_i in a view direction and a face normal of the mesh face f_i.

[0140] The second term defines visibility of seams between adjacent texture fragments. In this case, if the mesh faces f_i and f_j have an adjacent edge E_{ij}, then the visibility of seams is measured with an integral difference of colors of the adjacent edge E_{ij} in the corresponding camera image l_i and texture fragments of the camera image l_j. Here, Equation 3 may be used.

$$w_{i,j}^{l_i,l_j} = \int_{E_{ij}} d_{RGB}(Pr^{l_i}(X), Pr^{l_j}(x))dX \qquad \text{Equation 3}$$

[0141] In Equation 3, Pr^{l_i} denotes a projection operator for a view direction of the camera image l_i, and $d_{RGB}(\bullet,\bullet)$ denotes an Euclidean distance in a red/green/blue (RGB) color space.

[0142] Referring to FIG. **15**, in operation **1500**, unary potential w_i and pairwise potential w_{ij} are computed for the list **1410** of all visible mesh faces and camera images from the key frames **908**. Normalized potentials are used, so α shows trade-off between two energy terms, which is experimentally set to 50.

[0143] Next, a sequential reweighted message passing (SRMP) algorithm is run in operation **1502** to solve an MRF problem. As a result, the labeling vector l **1504** is obtained, which characterizes face labels.

[0144] In operation **1304**, global color adjustment is performed. Here, the 3D mesh **1206**, the key frames and the refined camera positions **908**, and the labeling vector **1504** are used as inputs. The labeling vector **1504** defines the number of texture patches, and after operation **1304**, seams between the texture patches become less noticeable.

[0145] Let f be a texture intensity function corresponding to one of RGB components. f is considered to be continuous on each texture patch and has discontinuities on boundaries between the texture patches. A goal is to find a leveling function g so that f+g will be continuous on patches and boundaries.

[0146] Each seam vertex v is split into several copies: v_{l_1} is textured from l_1 and v_{l_2} is textured from l_2. One for each patch contains v_s. The leveling function g is found in each vertex of the mesh according to Equation 4 below.

$$\sum_{\substack{v \text{ lies on seam,} \\ \text{splits into } v_{l_1}, v_{l_2}}} \left(f'_{v_{l_1}} + g_{v_{l_1}} - \left(f'_{v_{l_2}} + g_{v_{l_2}}\right)\right)^2 + \qquad \text{Equation 4}$$

$$\frac{1}{\lambda} \sum_{\substack{vi,vj \text{ are adjacent,} \\ \text{from the same patch}}} \left(g_{v_i} - g_{v_j}\right)^2 \to \min_{g}$$

[0147] In Equation 4, $f_{v_{li}}'$ denotes the mean of a weighted sum of colors along seam edges containing the seam vertex v from the camera image I_i. In the weighted sum, the weight in the seam vertex v is 1 and linearly decreases along an edge.

[0148] In this case, an MRF problem is solved using an Eigen's conjugate gradient for each channel of RGB separately. After values of the leveling function g are computed for all mesh vertices, values between vertices are calculated using interpolation. The leveling function g is applied to corresponding fragments of camera images (key frames **908**).

[0149] FIG. **16** is a diagram for describing operation **1308**, i.e., the hole filling, according to an embodiment of the present disclosure.

[0150] Referring to FIG. **16**, a color is assigned to invisible points in operations **1600** through **1610** and textures for invisible mesh faces are written in operation **1612**.

[0151] Invisible point colors are computed using interpolation of known point colors. A color of a point p_{vis} that is visible is provided from the same camera image as for texturing a mesh face which contains p_{vis}. If there are several faces containing the points p_{vis} that are labeled differently, either of the several faces is chosen.

[0152] In operation **1600**, all invisible points (vertices of the invisible mesh faces from the list **1412**) are added into a processing set.

[0153] Then, in operation **1602**, an interpolation process is performed while the processing set does not become empty.

[0154] During the interpolation process, colors are computed in operation **1604** for each point p from the processing set that has more than one visible adjacent point. The color of the point p is obtained in operation **1606** by interpolating colors of all visible adjacent points. Each of RGB-channels is averaged separately. Then, the point p is marked as visible in operation **1608** and is deleted in operation **1610** from the processing set. The interpolation process is performed while the processing set does not become empty in operation **1602**.

[0155] Finally, textures **1614** for invisible faces are constructed in operation **1612** from blocks of size 5×5 pixels. Here, one block is for each face.

[0156] FIG. **17** is a diagram of a structure of a block according to an embodiment of the present disclosure. Cells with V_1, v_2, v_3 have colors of face vertices which were found during the interpolation process. The colors of other cells are computed as following: the color of each cell is a mean of colors of cells from which there are arrows to this cell.

[0157] The final operation of the texture mapping process (the sub-stage **212**) is operation **1310**, i.e. texture atlas generation and parametrization. First, for a minimum bounding box of camera images, areas that are used for texture mapping are chosen. Then, some of the areas are scaled and combined to create, by using textures for the invisible faces from the list **1614**, a texture atlas of certain size. Next, mesh parametrization is performed by calculating texture coordinates for each mesh face vertex.

[0158] FIG. **18** is a diagram of a hardware structure of a device **106**, according to an embodiment of the present disclosure.

[0159] Referring to FIG. **18**, the device **106** includes a display **1800**, a CPU **1802**, one or more user interface devices **1804**, a memory **1806**, a GPU **1808**, a battery **1810**, and a camera **1812**.

[0160] The CPU **1802** and the GPU **1808** execute the above-described 3D reconstruction process. The 3D reconstruction process may be executed by a low performance and power consumption processor. The 3D reconstruction process may be executed by using one processor or two or more processors.

[0161] The memory **1806** stores different software required to execute the 3D reconstruction process, which may be used by the CPU **1802** and the GPU **1808**. Furthermore, the memory **1806** may also store intermediate data and final results obtained during the execution of the 3D reconstruction process. The 3D reconstruction process may be implemented by using a low memory size.

[0162] The camera **1812** (or a scanning module) is used to scan the object **102** and capture an image of the object **102**.

[0163] The display **1800** is used to aid the user **100** to correctly points onto the object **102** and to display progress and final results, for example, a texturized mesh of the object **102**.

[0164] The UI devices **1804**, such as one or more buttons, microphones, speakers, etc., may be used to start and stop scanning and to interact with the texturized mesh of the object **102**.

[0165] The battery **1810** is used to supply power to the device **106**.

[0166] The hardware structure of the device **106** is not limited to that of FIG. **18**. Unlike FIG. **18**, the device **106** may include additional components or may not include some of the components shown in FIG. **18**. For example, the device **106** may not include the UI device **1804**. Also, the display **1800** may be embodied as a touch screen displaying a UI for controlling the scanning of the object **102**. Here, the UI may allow the user **100** to change a scanning angle or a scanning time, or may be provided with other functions necessary for the 3D reconstruction process.

[0167] The device **106** may be connected, wirelessly or via wires, to one or more printing units, or to one or more other hand-held devices. For example, the device **106** may be connected to a printing unit that 3D-prints the texturized mesh (3D model) of the object **102**. Meanwhile, the device **106** may be connected to other hand-held devices to transmit the 3D model of the object **102** to the other hand-held devices.

[0168] During the 3D reconstruction process, the user **100** may take the following actions:

[0169] 1. The user **100** executes software installed in the device **106** and starts the scanning of the object **102** by using the camera **1812**. Here, the user **100** may use one or more UI devices **1804**. For example, the scanning of the object **102** may be activated by a user voice signal.

[0170] 2. The user **100** moves along the trajectory **108** around the object **102**. The software may guide the user **100** about how to move the camera **1812** around the object for a proper 3D reconstruction process with messages displayed on the display **1800**.

[0171] 3. The user **100** stops the scanning of the object **102** and touches a "Start reconstruction" button. Post processing including the sub-stages **204**, **206**, **208**, **210**, and **212** is executed using the CPU **1802** and the GPU **1808**, while storing process results obtained during post processing in the memory **1806**. As a result, the user **100** sees the texturized 3D mesh **214** (3D model) of the object **102** through the display **1800**. The user **100** may rotate or scale the 3D model.

[0172] 4. The user **100** may save the 3D model, transmit the 3D model to computers, or perform additional operations, such as rendering or changing textures.

[0173] According to the current embodiment of the present disclosure, a fast and user-friendly object capturing method may be provided to the user **100** since key frames to be used for the 3D reconstruction process are automatically selected. In addition, images do not need to be manually stored or transmitted to cloud since a cloud computing service is not used.

[0174] Although in the above embodiments of the present disclosure, the post processing is executed only by the device **106**, but an embodiment is not limited thereto, and a cloud computing service may be used for the post processing.

[0175] Furthermore, the device **106** may transmit the images to a host PC in real-time, and camera position estimation and post processing may also be performed by the PC host.

[0176] The images captured by the camera **1812** are not only possible data used for the camera positions estimation (the sub-stage **202**). For example, the device **106** may include some additional devices, such as inertial sensors, and data from such additional devices may be used for the camera position estimation in addition to the captured images.

[0177] The textured mesh is obtained as a result of the 3D reconstruction process. There may be two different modes for the 3D reconstruction process. In other words, there may be a general object model mode and a face model mode. A main difference between the two modes is a scanning behavior. In the general object model mode, 360° scanning is performed, while in the face model mode, only a face of an object is scanned at a scanning angle of 180°. However, any other forms of 3D model representation are available. For example, a dense 3D point cloud may be used for 3D reconstruction. A 3D model without textures in the form of a polygonal mesh may be represented.

[0178] Any kind of mobile devices, which has the hardware structure of FIG. **18**, such as smart phones, tablets, and smart watches, may be used for the 3D reconstruction process.

[0179] 3D printing may be performed based on a 3D model generated according to the 3D reconstruction process according to the current embodiment of the present disclosure.

[0180] According to the embodiments, a 3D photo may be used or shared on the Internet by storing a reconstructed 3D model of a human face.

[0181] FIG. **19** is a flowchart of a method of reconstructing a 3D model of an object according to an embodiment of the present disclosure.

[0182] Referring to FIG. **19**, the method includes the operations described above, which are performed in time-series, and thus details described above may be applied to the method of FIG. **19** even if omitted.

[0183] In operation **1901**, the device **106** captures images of an object by scanning the object along a trajectory around the object.

[0184] In operation **1902**, the device **106** estimates positions of a scanning module, which respectively correspond to the captured images.

[0185] In operation **1903**, the device **106** refines the estimated positions based on at least two locations on the trajectory.

[0186] In operation **1904**, the device **106** estimates depth maps corresponding to the refined positions.

[0187] In operation **1905**, the device **106** generates a surface mesh of the object by fusing the estimated depth maps.

[0188] In operation **1906**, the device **106** constructs and displays a 3D model of the object by mapping textures onto the generated surface mesh.

[0189] The embodiments may be written as computer programs and may be implemented in general-use digital computers that execute the programs using a non-transitory computer-readable recording medium. Examples of the non-transitory computer-readable recording medium include magnetic storage media (e.g., read only memory (ROM), floppy disks, hard disks, etc.), optical recording media (e.g., compact disc ROMs (CD-ROMs), or digital versatile discs (DVDs)), etc.

[0190] It should be understood that embodiments described herein should be considered in a descriptive sense only and not for purposes of limitation. Descriptions of features or aspects within each embodiment should typically be considered as available for other similar features or aspects in other embodiments of the present disclosure.

[0191] While the present disclosure has been shown and described with reference to various embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the present disclosure as defined by the appended claims and their equivalents.

What is claimed is:

1. A method of constructing a three-dimensional (3D) model of an object, the method comprising:

capturing images of the object by scanning the object along a trajectory around the object;

estimating positions of a scanner, which respectively correspond to the captured images;

refining the estimated positions of the scanner based on at least two locations on the trajectory;

estimating depth maps corresponding to the refined positions of the scanner;

generating a surface mesh of the object by fusing the estimated depth maps; and

constructing and displaying the 3D model of the object by mapping textures onto the generated surface mesh.

2. The method of claim **1**,

wherein the trajectory is a closed loop or an opened loop, and

wherein the at least two locations on the trajectory comprise start and end points of the trajectory, which correspond to loop closure points for a loop closure of the trajectory.

3. The method of claim **1**, wherein the capturing of the images further comprises capturing the images by scanning the object in 360° by using a simultaneous localization and mapping (SLAM).

4. The method of claim **1**, wherein the estimating of the positions of the scanner further comprises:

performing map initialization by using a first image from among the captured images;

extracting key points from the first image by using a features from accelerated segment test (FAST) algorithm performing corner detection; and

estimating the positions of the scanner, which respectively correspond to the captured images by using the extracted key points.

5. The method of claim **4**, wherein the estimating of the positions of the scanner further comprises:

determining whether each of the captured images corresponds to a key frame; and

storing a captured image determined to be the key frame in a memory by matching the captured image determined to be the key frame to each of the estimated positions of the scanner.

6. The method of claim **5**, wherein the refining of the estimated positions further comprises:

obtaining loop points on the trajectory with respect to the key frame;

checking whether the loop points are loop closure points; and

when the loop points are the loop closure points, refining the estimated positions based on a result of performing a loop closure by using the loop closure points.

7. The method of claim **1**, wherein the refining of the estimated positions is performed to compensate for an effect of a drift generated when the object is scanned by using a simultaneous localization and mapping (SLAM).

8. The method of claim **1**, wherein the estimating of the depth maps further comprises:

selecting a stereo pair comprising adjacent key frames; and

estimating the depth maps from the selected stereo pair by using a pyramidal approach in which a result of low resolution sampling is up-sampled via sampling of finer resolution.

9. The method of claim **1**, wherein the generating of the surface mesh further comprises:

generating the surface mesh corresponding to a voxel-to-octree representation regarding the object by fusing the estimated depth maps by using 3D voxel truncated signed distance function (TSDF).

10. The method of claim **1**, wherein the constructing and displaying of the 3D model further comprises:

checking whether each face of the surface mesh is visible or invisible at each of the refined positions;

texturing a visible face of the surface mesh to a projection corresponding to one of the captured images based on a result of the checking; and

constructing and displaying the 3D model by generating the texturized surface mesh.

11. An apparatus for configuring a three-dimensional (3D) model of an object, the apparatus comprising:

a scanner configured to capture images of the object by scanning the object along a trajectory around the object;

at least one processor configured to:

estimate positions of the scanner, which respectively correspond to the captured images,

refine the estimated positions of the scanner based on at least two locations on the trajectory,

estimate depth maps corresponding to the refined locations of the scanner,

generate a surface mesh of the object by fusing the estimated depth maps, and

construct the 3D model of the object by mapping textures onto the generated surface mesh; and

a display configured to display the 3D mode of the object.

12. The apparatus of claim **11**,

wherein the trajectory is a closed loop or an opened loop, and

wherein the at least two locations on the trajectory comprise start and end points of the trajectory, which correspond to loop closure points for a loop closure of the trajectory.

13. The apparatus of claim **11**, wherein the scanner is further configured to capture the images by scanning the object in 360° by using a simultaneous localization and mapping (SLAM).

14. The apparatus of claim **11**, wherein the at least one processor is further configured to:

perform map initialization by using a first image from among the captured images,

extract key points from the first image by using a features from accelerated segment test (FAST) algorithm performing corner detection, and

estimate the positions of the scanner, which respectively correspond to the captured images by using the extracted key points.

15. The apparatus of claim **14**, wherein the at least one processor is further configured to:

determine whether each of the captured images corresponds to a key frame, and

store a captured image determined to be the key frame in a memory by matching the captured image determined to be the key frame to each of the estimated positions of the scanner.

16. The apparatus of claim **15**, wherein the at least one processor is further configured to:

obtain loop points on the trajectory with respect to the key frame,

check whether the loop points are loop closure points, and

when the loop points are the loop closure points, refine the estimated positions based on a result of performing loop closure by using the loop closure points.

17. The apparatus of claim **11**, wherein the at least one processor is further configured to refine the positions of the scanner to compensate for an effect of a drift generated when the object is scanned by using a simultaneous localization and mapping (SLAM).

18. The apparatus of claim **11**, wherein the at least one processor is further configured to:

select a stereo pair comprising adjacent key frames, and

estimate the depth maps from the selected stereo pair by using a pyramidal approach in which a result of low resolution sampling is up-sampled via sampling of finer resolution.

19. The apparatus of claim **11**, wherein the at least one processor is further configured to generate the surface mesh corresponding to a voxel-to-octree representation regarding the object by fusing the estimated depth maps by using 3D voxel truncated signed distance function (TSDF).

20. The apparatus of claim **11**, wherein the at least one processor is further configured to:

check whether each face of the surface mesh is visible or invisible at each of the refined positions,

texture a visible face of the surface mesh to a projection corresponding to one of the captured images based on a result of the checking, and

construct the 3D model by generating the texturized surface mesh.

* * * * *