US 20180285697A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2018/0285697 A1

Shotton et al. (43) Pub. Date: Oct. 4, 2018

(54) **CAMERA/OBJECT POSE FROM PREDICTED COORDINATES**

(71) Applicant: **Microsoft Technology Licensing, LLC**, Redmond, WA (US)

(72) Inventors: **Jamie Daniel Joseph Shotton**, Cambridge (GB); **Benjamin Michael Glocker**, Cambridge (GB); **Christopher Zach**, Cambridge (GB); **Shahram Izadi**, Cambridge (GB); **Antonio Criminisi**, Cambridge (GB); **Andrew William Fitzgibbon**, Cambridge (GB)

(21) Appl. No.: **15/895,990**

(22) Filed: **Feb. 13, 2018**

Related U.S. Application Data

(63) Continuation of application No. 13/774,145, filed on Feb. 22, 2013, now Pat. No. 9,940,553.

Publication Classification

(51) **Int. Cl.**

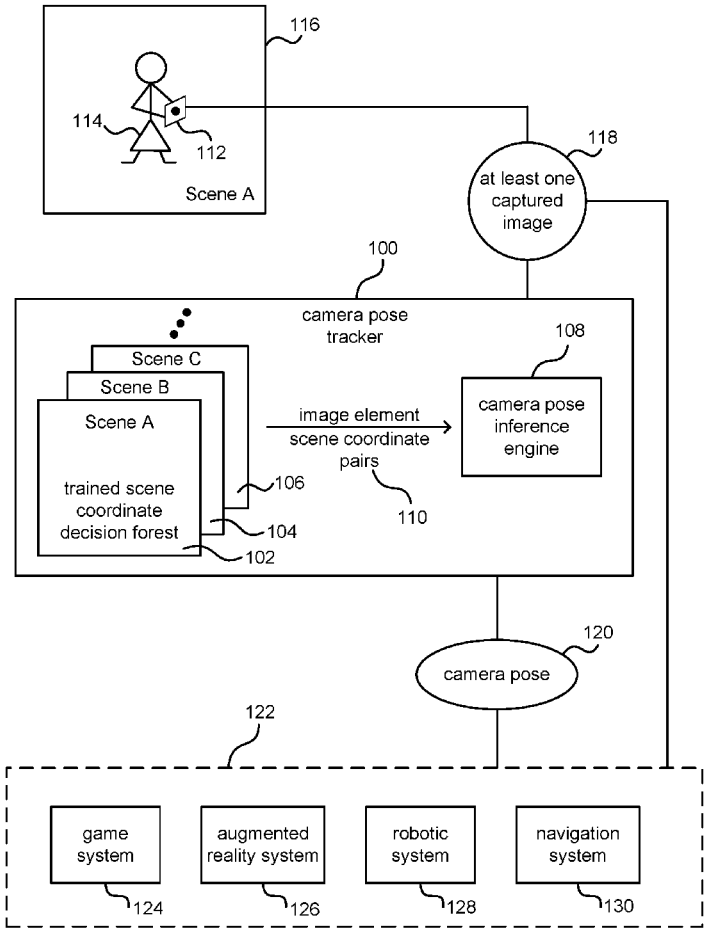
G06K 9/66	(2006.01)
G06K 9/62	(2006.01)
G06K 9/00	(2006.01)

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

CPC ***G06K 9/66*** (2013.01); ***G06K 9/6282*** (2013.01); ***G06K 9/00671*** (2013.01); ***G06K 9/6219*** (2013.01); ***G06K 9/6256*** (2013.01)

(57) **ABSTRACT**

Camera or object pose calculation is described, for example, to relocalize a mobile camera (such as on a smart phone) in a known environment or to compute the pose of an object moving relative to a fixed camera. The pose information is useful for robotics, augmented reality, navigation and other applications. In various embodiments where camera pose is calculated, a trained machine learning system associates image elements from an image of a scene, with points in the scene's 3D world coordinate frame. In examples where the camera is fixed and the pose of an object is to be calculated, the trained machine learning system associates image elements from an image of the object with points in an object coordinate frame. In examples, the image elements may be noisy and incomplete and a pose inference engine calculates an accurate estimate of the pose.



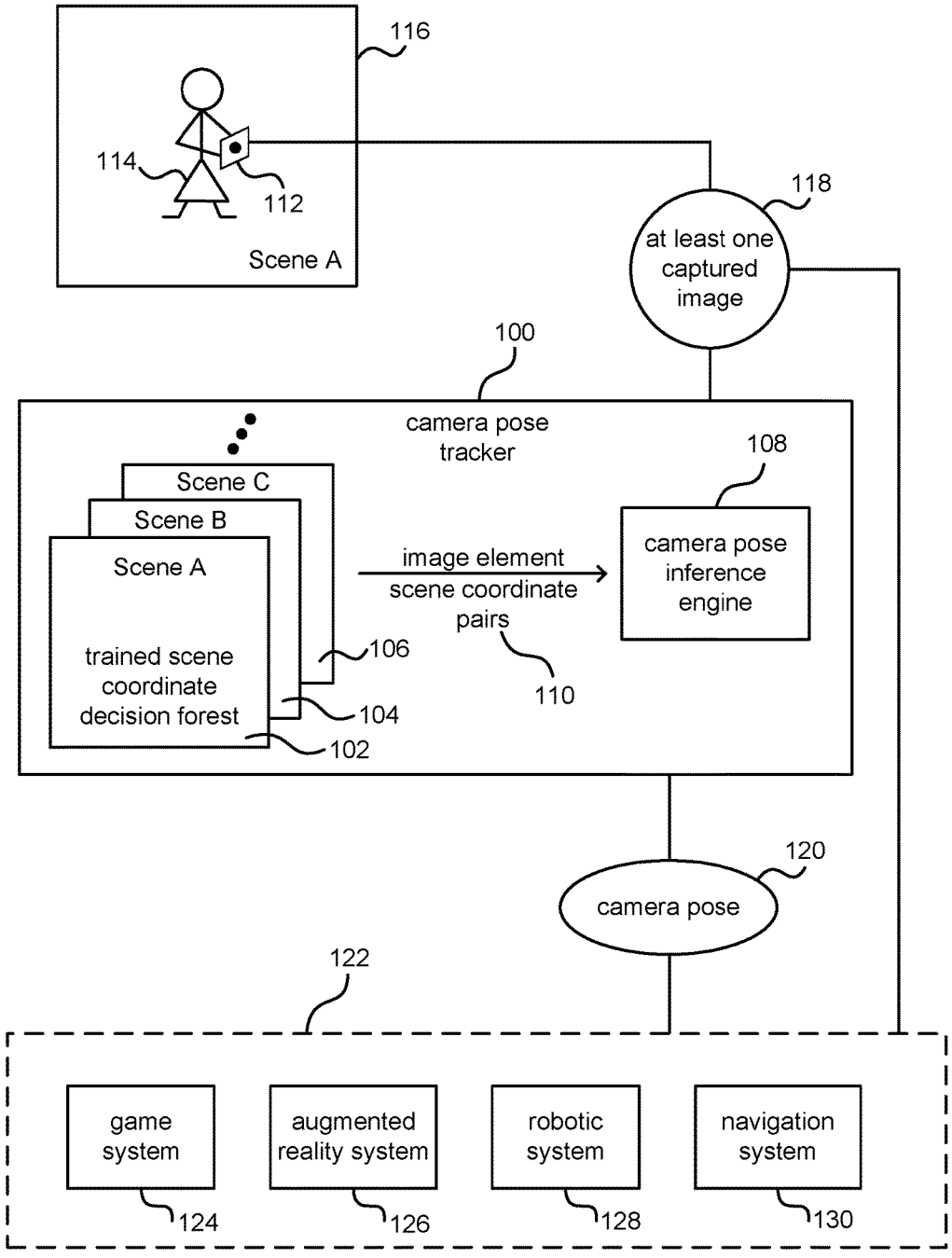

116
114
112
Scene A
118
at least one captured image
100
camera pose tracker
Scene C
Scene B
Scene A
trained scene coordinate decision forest
106
104
102
image element scene coordinate pairs
110
108
camera pose inference engine
120
camera pose
122
game system
124
augmented reality system
126
robotic system
128
navigation system
130

FIG. 1

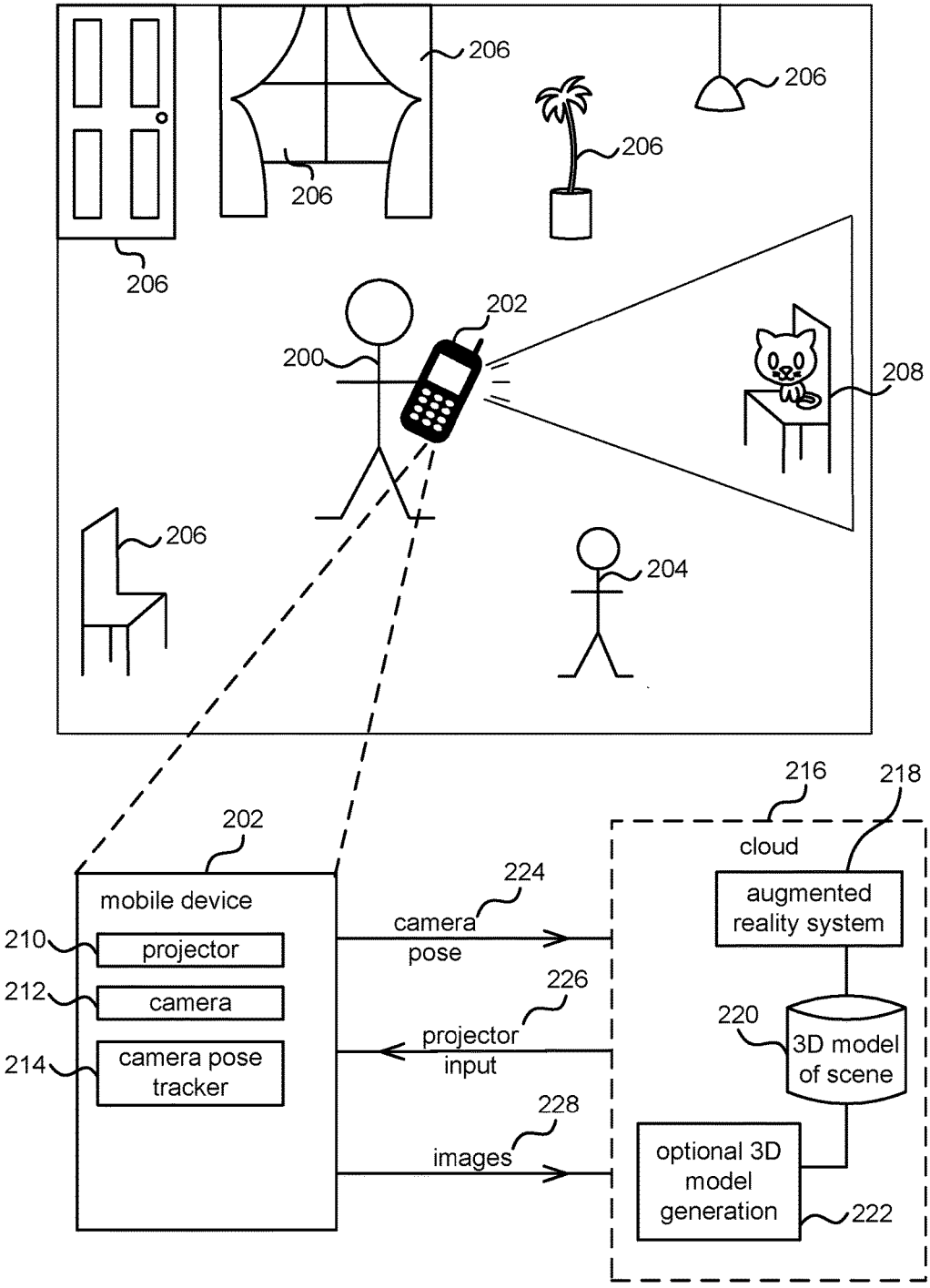

206
206
206
206
206
202
200
208
206
204
216
218
202
cloud
224
mobile device
augmented reality system
camera pose
210
projector
212
camera
226
220
3D model of scene
projector input
214
camera pose tracker
228
images
optional 3D model generation
222

FIG. 2

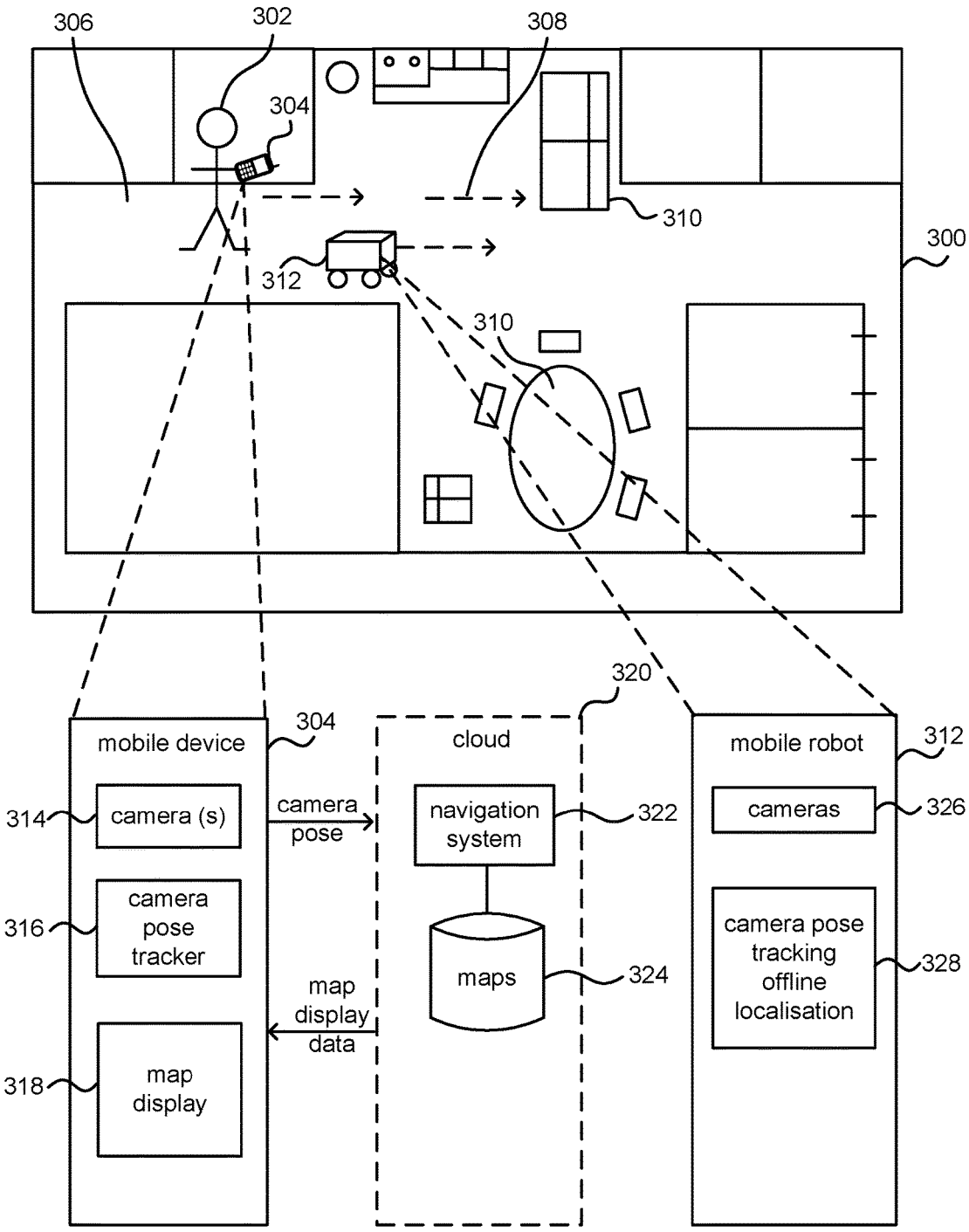

306
302
308
304
310
300
312
310
320
304
mobile device
cloud
mobile robot
312
314
camera (s)
camera pose
navigation system
322
cameras
326
316
camera pose tracker
maps
324
camera pose tracking offline localisation
328
map display data
318
map display

FIG. 3

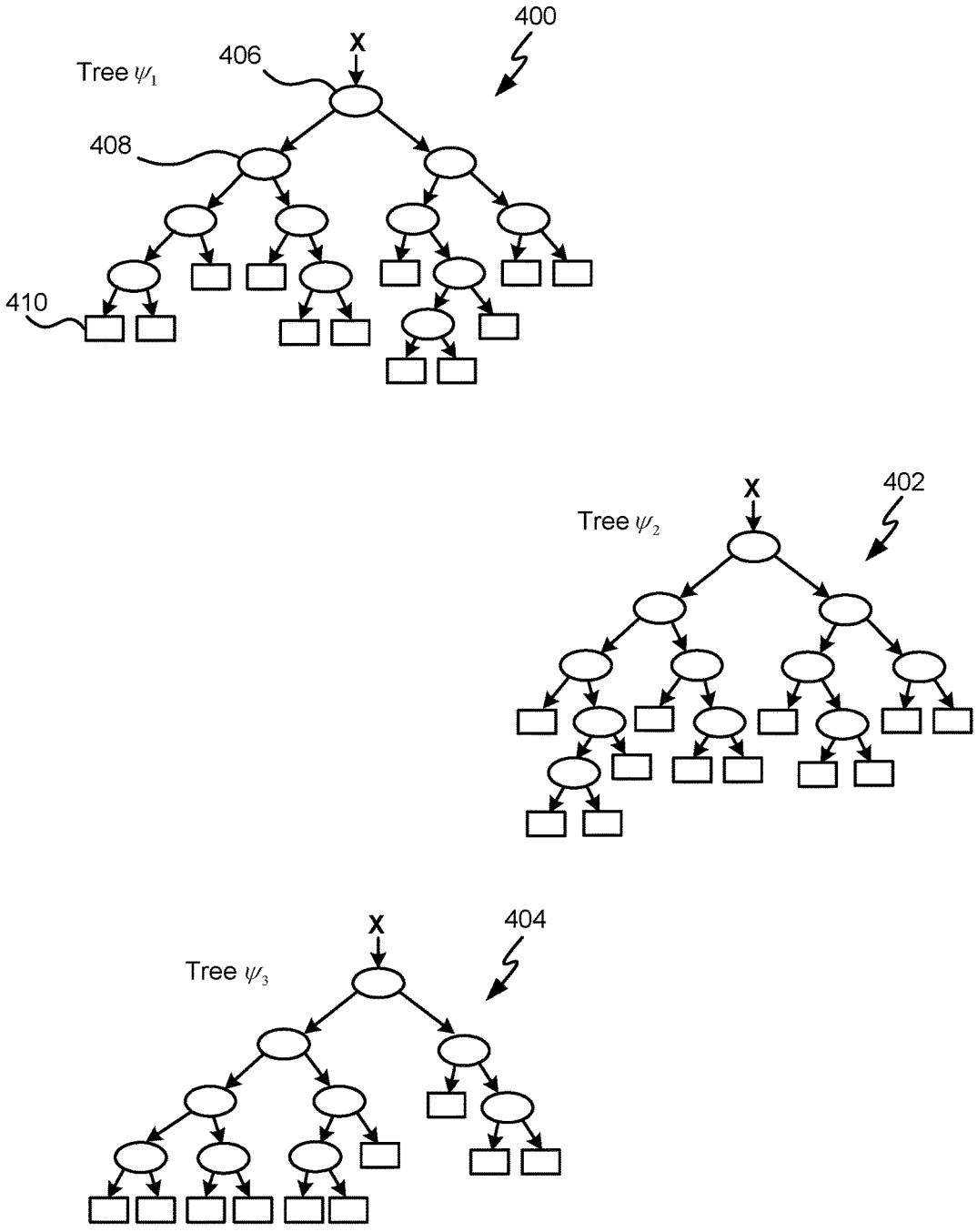


FIG. 4

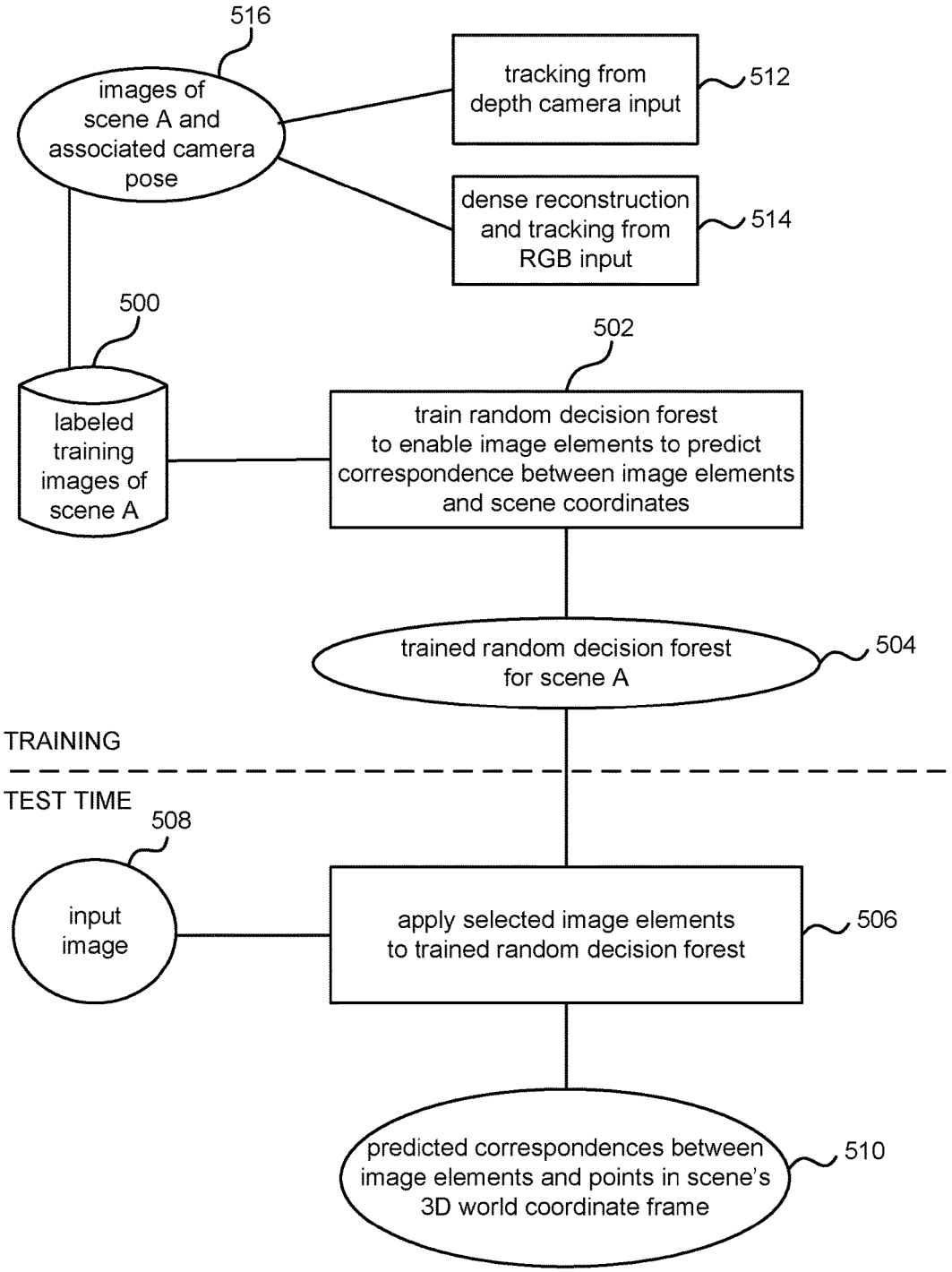

516
images of scene A and associated camera pose
tracking from depth camera input
512
dense reconstruction and tracking from RGB input
514
500
labeled training images of scene A
502
train random decision forest to enable image elements to predict correspondence between image elements and scene coordinates
trained random decision forest for scene A
504
TRAINING
TEST TIME
508
input image
apply selected image elements to trained random decision forest
506
predicted correspondences between image elements and points in scene's 3D world coordinate frame
510

FIG. 5

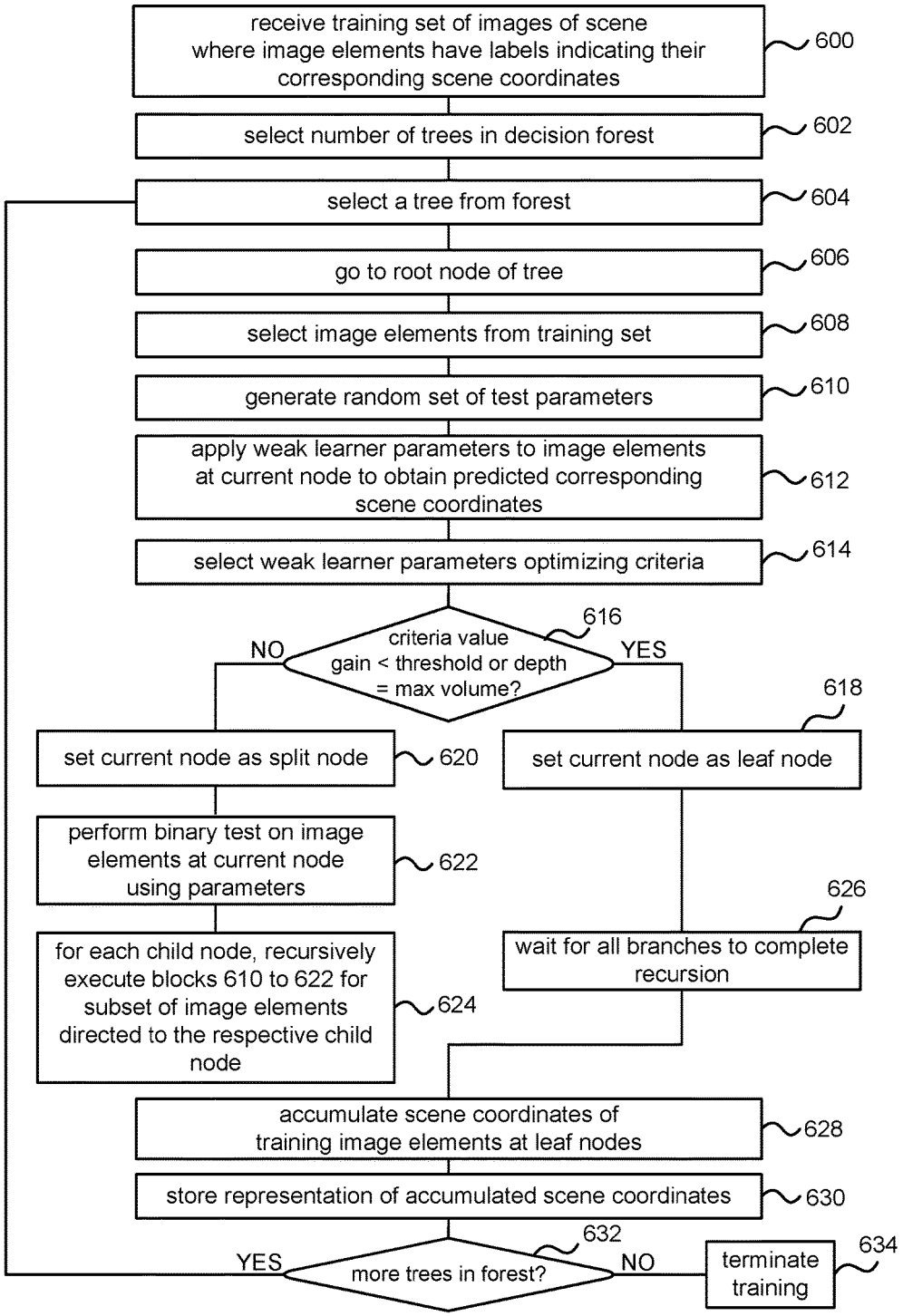

receive training set of images of scene where image elements have labels indicating their corresponding scene coordinates
600
select number of trees in decision forest
602
select a tree from forest
604
go to root node of tree
606
select image elements from training set
608
generate random set of test parameters
610
apply weak learner parameters to image elements at current node to obtain predicted corresponding scene coordinates
612
select weak learner parameters optimizing criteria
614
criteria value gain < threshold or depth = max volume?
616
NO
YES
set current node as split node
620
set current node as leaf node
618
perform binary test on image elements at current node using parameters
622
for each child node, recursively execute blocks 610 to 622 for subset of image elements directed to the respective child node
624
wait for all branches to complete recursion
626
accumulate scene coordinates of training image elements at leaf nodes
628
store representation of accumulated scene coordinates
630
more trees in forest?
632
YES
NO
terminate training
634

FIG. 6

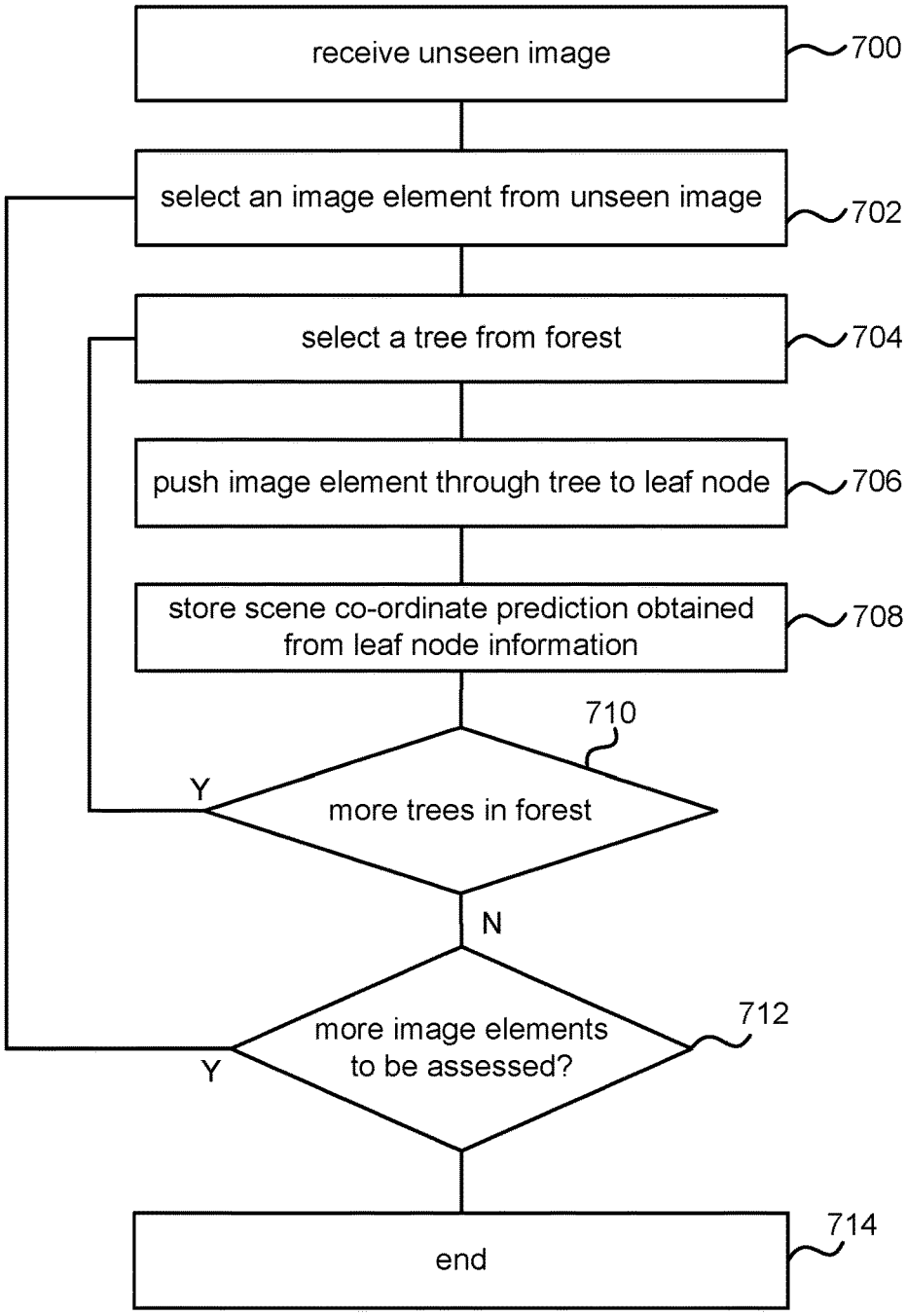

receive unseen image
700
select an image element from unseen image
702
select a tree from forest
704
push image element through tree to leaf node
706
store scene co-ordinate prediction obtained from leaf node information
708
710
more trees in forest
Y
N
more image elements to be assessed?
712
Y
end
714

FIG. 7

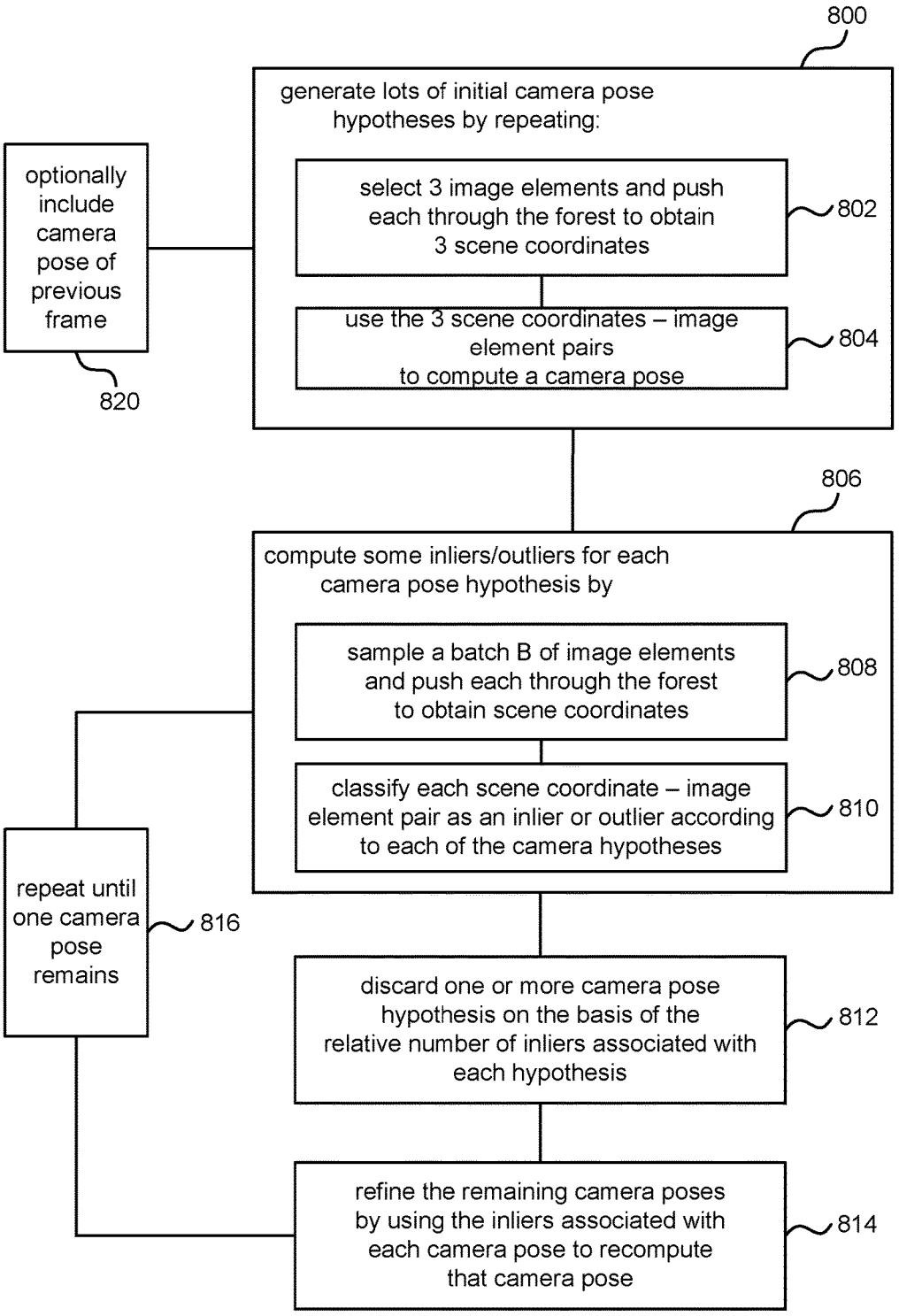

800
generate lots of initial camera pose hypotheses by repeating:
select 3 image elements and push each through the forest to obtain 3 scene coordinates
802
use the 3 scene coordinates – image element pairs to compute a camera pose
804
optionally include camera pose of previous frame
820
806
compute some inliers/outliers for each camera pose hypothesis by
sample a batch B of image elements and push each through the forest to obtain scene coordinates
808
classify each scene coordinate – image element pair as an inlier or outlier according to each of the camera hypotheses
810
repeat until one camera pose remains
816
discard one or more camera pose hypothesis on the basis of the relative number of inliers associated with each hypothesis
812
refine the remaining camera poses by using the inliers associated with each camera pose to recompute that camera pose
814

FIG. 8

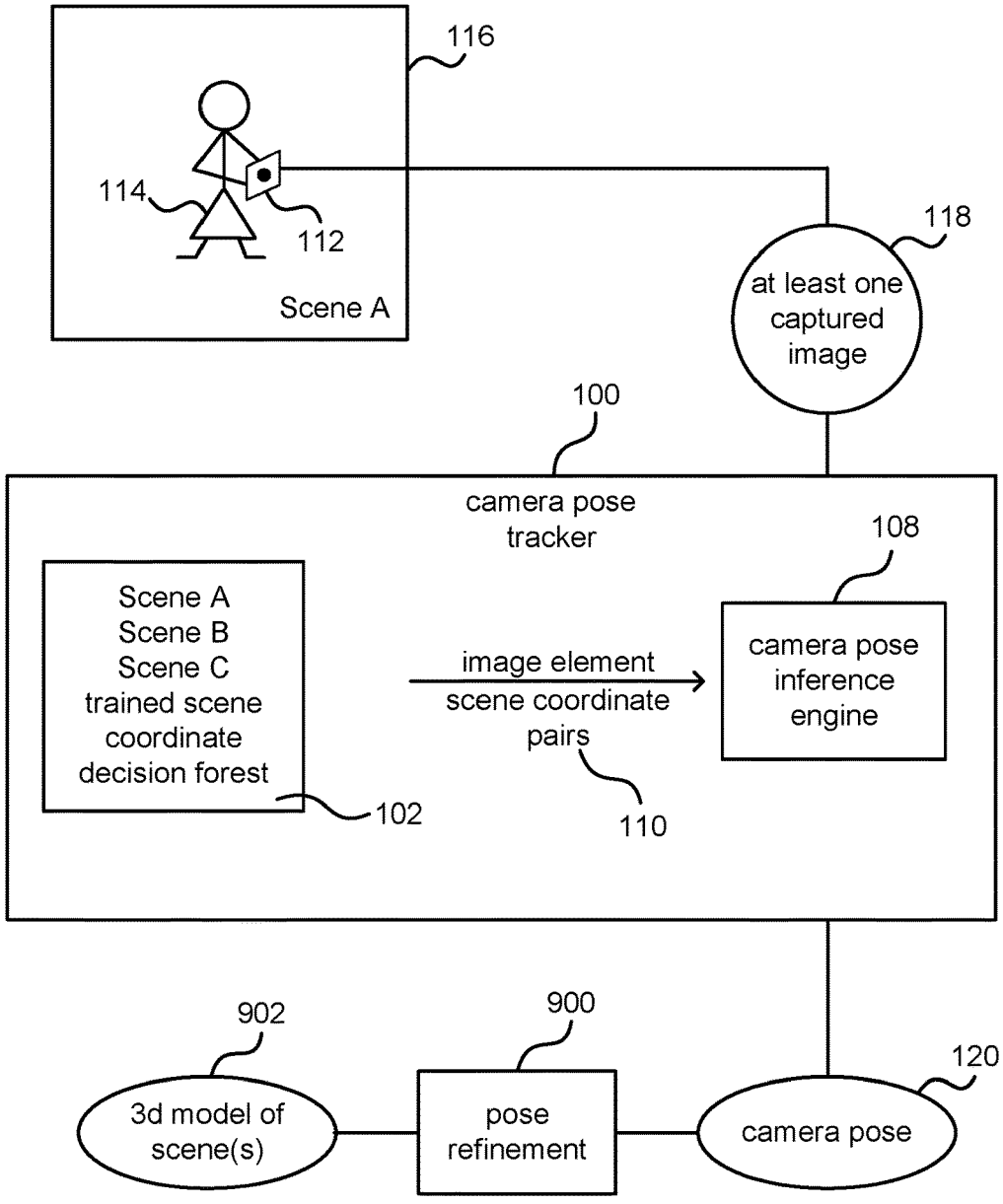

116
114
112
Scene A
118
at least one captured image
100
camera pose tracker
108
Scene A
Scene B
Scene C
trained scene coordinate decision forest
102
image element scene coordinate pairs
110
camera pose inference engine
902
900
120
3d model of scene(s)
pose refinement
camera pose

FIG. 9

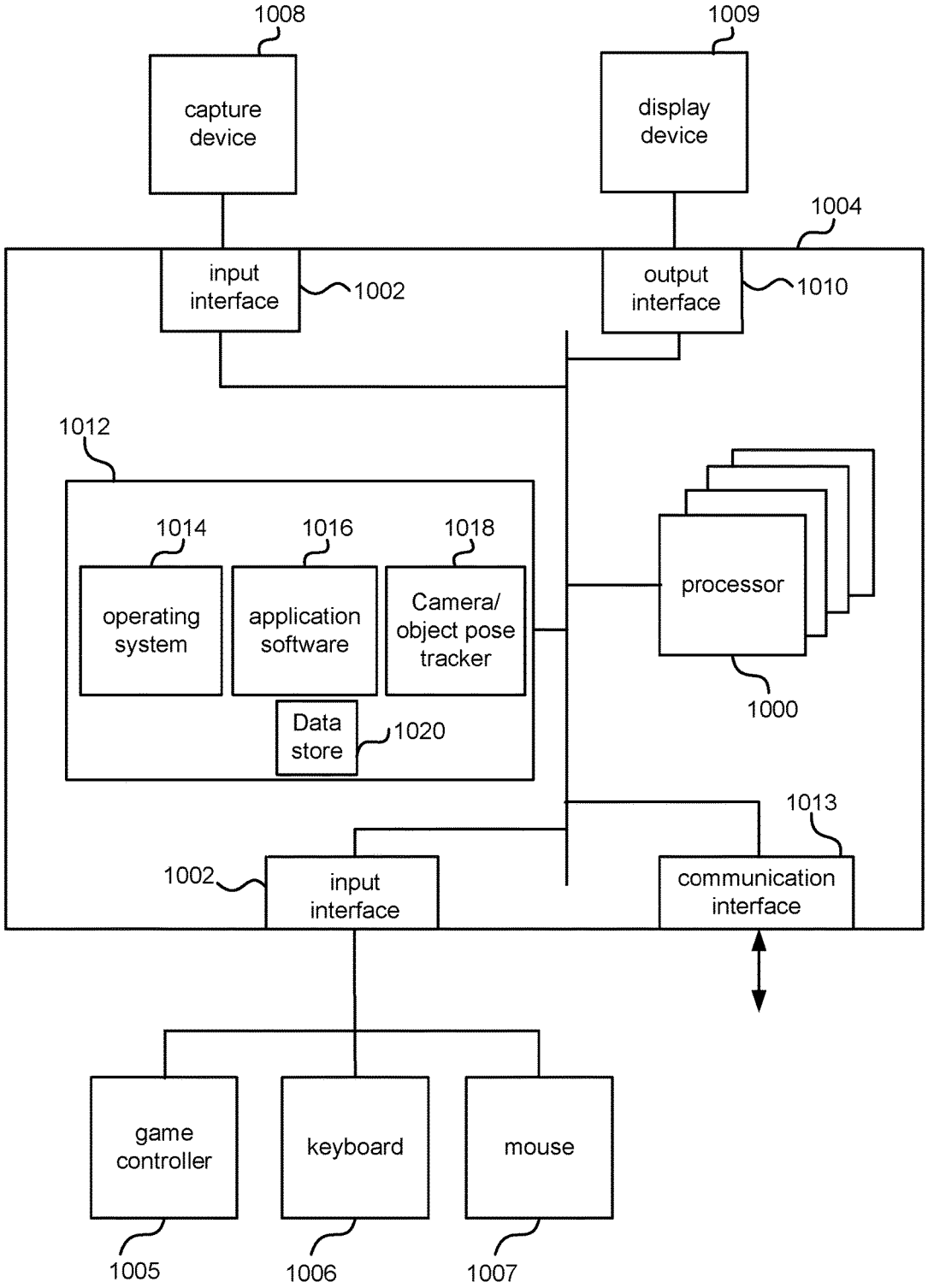

1008
capture device
1009
display device
1004
input interface
1002
output interface
1010
1012
1014
operating system
1016
application software
1018
Camera/ object pose tracker
Data store
1020
processor
1000
1013
1002
input interface
communication interface
game controller
keyboard
mouse
1005
1006
1007

FIG. 10

CAMERA/OBJECT POSE FROM PREDICTED COORDINATES

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application is a continuation of U.S. Non-Provisional application Ser. No. 13/774,145, filed on Feb. 22, 2013, and entitled "CAMERA/OBJECT POSE FROM PREDICTED COORDINATES" and is hereby incorporated by reference herein for all intents and purposes.

BACKGROUND

[0002] For many applications, such as robotics, vehicle navigation, computer game applications, medical applications and other problem domains, it is valuable to be able to find orientation and position of a camera as it moves in a known environment. Orientation and position of a camera is known as camera pose and may comprise six degrees of freedom (three of translation and three of rotation). Where a camera is fixed and an object moves relative to the camera it is also useful to be able to compute the pose of the object.

[0003] A previous approach uses keyframe matching where a whole test image is matched against exemplar training images (keyframes). K matching keyframes are found, and the poses (keyposes) of those keyframes are interpolated to generate an output camera pose. Keyframe matching tends to be very approximate in the pose result.

[0004] Another previous approach uses keypoint matching where a sparse set of interest points are detected in a test image and matched using keypoint descriptors to a known database of descriptors. Given a putative set of matches, a robust optimization is run to find the camera pose for which the largest number of those matches are consistent geometrically. Keypoint matching struggles in situations where too few keypoints are detected.

[0005] Existing approaches are limited in accuracy, robustness and speed.

[0006] The embodiments described below are not limited to implementations which solve any or all of the disadvantages of known systems for finding camera or object pose.

SUMMARY

[0007] The following presents a simplified summary of the disclosure in order to provide a basic understanding to the reader. This summary is not an extensive overview of the disclosure and it does not identify key/critical elements or delineate the scope of the specification. Its sole purpose is to present a selection of concepts disclosed herein in a simplified form as a prelude to the more detailed description that is presented later.

[0008] Camera or object pose calculation is described, for example, to relocalize a mobile camera (such as on a smart phone) in a known environment or to compute the pose of an object moving relative to a fixed camera. The pose information is useful for robotics, augmented reality, navigation and other applications. In various embodiments where camera pose is calculated, a trained machine learning system associates image elements from an image of a scene, with points in the scene's 3D world coordinate frame. In examples where the camera is fixed and the pose of an object is to be calculated, the trained machine learning system associates image elements from an image of the object with points in an object coordinate frame. In examples, the image elements may be noisy and incomplete and a pose inference engine calculates an accurate estimate of the pose.

[0009] Many of the attendant features will be more readily appreciated as the same becomes better understood by reference to the following detailed description considered in connection with the accompanying drawings.

DESCRIPTION OF THE DRAWINGS

[0010] The present description will be better understood from the following detailed description read in light of the accompanying drawings, wherein:

[0011] FIG. **1** is a schematic diagram of a camera pose tracker for relocalizing a mobile camera (such as in a smart phone) in scene A;

[0012] FIG. **2** is a schematic diagram of a person holding a mobile device with a camera and a camera pose tracker and which communicates with an augmented reality system to enable an image of a cat to be projected into the scene in a realistic manner;

[0013] FIG. **3** is a schematic diagram of a person and a robot each with a camera and a camera pose tracker;

[0014] FIG. **4** is a schematic diagram of three random decision trees forming at least part of a random decision forest;

[0015] FIG. **5** is a flow diagram of a method of training a random decision forest to predict correspondences between image elements and scene coordinates; and using the trained random decision forest;

[0016] FIG. **6** is a flow diagram of a method of training a random decision forest using images of a scene where image elements have labels indicating their corresponding scene coordinates;

[0017] FIG. **7** is a flow diagram of a method of using a trained random decision forest to obtain scene coordinate—image element pairs;

[0018] FIG. **8** is a flow diagram of a method at a camera pose inference engine of using scene-coordinate-image element pairs to infer camera pose;

[0019] FIG. **9** is a schematic diagram of the camera pose tracker of FIG. **1** where a 3D model of the scene is available;

[0020] FIG. **10** illustrates an exemplary computing-based device in which embodiments of a camera or object pose tracker may be implemented.

[0021] Like reference numerals are used to designate like parts in the accompanying drawings.

DETAILED DESCRIPTION

[0022] The detailed description provided below in connection with the appended drawings is intended as a description of the present examples and is not intended to represent the only forms in which the present example may be constructed or utilized. The description sets forth the functions of the example and the sequence of steps for constructing and operating the example. However, the same or equivalent functions and sequences may be accomplished by different examples.

[0023] Although the present examples are described and illustrated herein as being implemented using a random decision forest, the system described is provided as an example and not a limitation. As those skilled in the art will appreciate, the present examples may be implemented using a variety of different types of machine learning systems

including but not limited to support vector machines, Gaussian process regression systems.

[0024] FIG. **1** is a schematic diagram of a camera pose tracker for relocalizing a mobile camera (such as in a smart phone) in scene A. In this example a person **114** is holding the mobile camera **112** which is integral with a communications device such as a smart phone. The person **114** uses the mobile camera **112** to capture at least one image **118** of scene A **116**, such as a living room, office or other environment. The image may be a depth image, a color image (referred to as an RGB image) or may comprise both a depth image and a color image. In some examples a stream of images is captured by the mobile camera.

[0025] A camera pose tracker **100** is either integral with the smart phone or is provided at another entity in communication with the smart phone. The camera pose tracker **100** is implemented using software and/or hardware as described in more detail below with reference to FIG. **10**. The camera pose tracker **100** comprises a plurality of trained scene coordinate decision forests **102**, **104**, **106** one for each of a plurality of scenes. The trained scene coordinate decision forests may be stored at the camera pose tracker or may be located at another entity which is in communication with the camera pose tracker. Each scene coordinate decision forest is a type of machine learning system which takes image elements (from images of its associated scene) as input and produces estimates of scene coordinates (in world space) of points in a scene which the image elements depict. Image elements may be pixels, groups of pixels, voxels, groups of voxels, blobs, patches or other components of an image. Other types of machine learning system may be used in place of the scene coordinate decision forest. For example, support vector machine regression systems, Gaussian process regression systems.

[0026] A decision forest comprises one or more decision trees each having a root node, a plurality of split nodes and a plurality of leaf nodes. Image elements of an image may be pushed through trees of a decision forest from the root to a leaf node in a process whereby a decision is made at each split node. The decision is made according to characteristics of the image element and characteristics of test image elements displaced therefrom by spatial offsets specified by the parameters at the split node. At a split node the image element proceeds to the next level of the tree down a branch chosen according to the results of the decision. The random decision forest may use regression or classification as described in more detail below. During training, parameter values (also referred to as features) are learnt for use at the split nodes and data is accumulated at the leaf nodes. For example, distributions of scene coordinates are accumulated at the leaf nodes.

[0027] Storing all the scene coordinates at the leaf nodes during training may be very memory intensive since large amounts of training data are typically used for practical applications. The scene coordinates may be aggregated in order that they may be stored in a compact manner. Various different aggregation processes may be used. An example in which modes of the distribution of scene coordinates are store is described in more detail below.

[0028] In the example of FIG. **1** there is a plurality of trained scene coordinate decision forests; one for each of a plurality of scenes. However, it is also possible to have a single trained scene coordinate decision forest which operates for a plurality of scenes. This is explained below with reference to FIG. **9**.

[0029] The scene coordinate decision forest(s) provide image element-scene coordinate pair estimates **110** for input to a camera pose inference engine **108** in the camera pose tracker **100**. Information about the certainty of the image element-scene coordinate estimates may also be available. The camera pose inference engine **108** may use an energy optimization approach to find a camera pose which is a good fit to a plurality of image element—scene coordinate pairs predicted by the scene coordinate decision forest. This is described in more detail below with reference to FIG. **8**. In some examples scene coordinates for each available image element may be computed and used in the energy optimization. However, to achieve performance improvements whilst retaining accuracy, a subsample of image elements may be used to compute predicted scene coordinates.

[0030] The camera pose inference engine **108** uses many image element-scene coordinate pairs **110** to infer the pose of the mobile camera **112** using an energy optimization approach as mentioned above. Many more than three pairs (the minimum needed) may be used to improve accuracy. For example, the at least one captured image **118** may be noisy and may have missing image elements, especially where the captured image **118** is a depth image. On the other hand, to obtain a scene coordinate prediction for each image element in an image is computationally expensive and time consuming because each image element needs to be pushed through the forest as described with reference to FIG. **7**. Therefore, in some examples, the camera pose inference engine may use an iterative process which gives the benefit that a subsample of image elements are used to compute scene coordinate predictions whilst taking accuracy into account.

[0031] The camera pose **120** output by the camera pose tracker may be in the form of a set of parameters with six degrees of freedom, three indicating the rotation of the camera and three indicating the position of the camera. For example, the output of the camera pose tracker is a set of registration parameters of a transform from camera space to world space. In some examples these registration parameters are provided as a six degree of freedom (6DOF) pose estimate in the form of an SE_3 matrix describing the rotation and translation of the camera relative to real-world coordinates.

[0032] The camera pose **120** output by the camera pose tracker **100** may be input to a downstream system **122** together with the captured image(s) **118**. The downstream system may be a game system **124**, an augmented reality system **126**, a robotic system **128**, a navigation system **130** or other system. An example where the downstream system **122** is an augmented reality system is described with reference to FIG. **2**.

[0033] The examples described show how camera pose may be calculated. These examples may be modified in a straightforward manner to enable pose of an object to be calculated where the camera is fixed. In this case the machine learning system is trained using training images of an object where image elements are labeled with object coordinates. An object pose tracker is then provided which uses the methods described herein adapted to the situation where the camera is fixed and pose of an object is to be calculated.

[0034] Alternatively, or in addition, the camera pose tracker or object pose tracker described herein can be performed, at least in part, by one or more hardware logic components. For example, and without limitation, illustrative types of hardware logic components that can be used include Field-programmable Gate Arrays (FPGAs), Program-specific Integrated Circuits (ASICs), Program-specific Standard Products (ASSPs), System-on-a-chip systems (SOCs), Complex Programmable Logic Devices (CPLDs), Graphics Processing Units (GPUs).

[0035] FIG. **2** is a schematic diagram of a person **200** holding a mobile device **202** which has a camera **212**, a camera pose tracker **214** and a projector **210**. For example, the mobile device may be a smart phone. Other components of the mobile device to enable it to function as a smart phone such as a communications interface, display screen, power source and other components are not shown for clarity. A person **200** holding the mobile device **202** is able to capture images of the scene or environment in which the user is moving. In the example of FIG. **2** the scene or environment is a living room containing various objects **206** and another person **204**.

[0036] The mobile device is able to communicate with one or more entities provided in the cloud **216** such as an augmented reality system **218**, a 3D model of the scene **220** and an optional 3D model generation system **222**.

[0037] For example, the user **200** operates the mobile device **202** to capture images of the scene which are used by the camera pose tracker **214** to compute the pose (position and orientation) of the camera. At the consent of the user, the camera pose is sent **224** to the entities in the cloud **216** optionally with the images **228**. The augmented reality system **218** may have access to a 3D model of the scene **220** (for example, a 3D model of the living room) and may use the 3D model and the camera pose to calculate projector input **226**. The projector input **226** is sent to the mobile device **202** and may be projected by the projector **210** into the scene. For example, an image of a cat **208** may be projected into the scene in a realistic manner taking into account the 3D model of the scene and the camera pose. The 3D model of the scene could be a computer aided design (CAD) model, or could be a model of the surfaces in the scene built up from images captured of the scene using a 3D model generation system **222**. An example of a 3D model generation system which may be used is described in US patent application "Three-Dimensional Environment Reconstruction" Newcombe, Richard et al. published on Aug. 2, 2012 US20120194516. Other types of 3D model and 3D model generation systems may also be used.

[0038] An example where the downstream system **122** is a navigation system is now described with reference to FIG. **3**. FIG. **3** has a plan view of a floor of an office **300** with various objects **310**. A person **302** holding a mobile device **304** is walking along a corridor **306** in the direction of arrows **308**. The mobile device **304** has one or more cameras **314**, a camera pose tracker **316** and a map display **318**. The mobile device **304** may be a smart phone or other mobile communications device as described with reference to FIG. **2** and which is able to communicate with a navigation system **322** in the cloud **320**. The navigation system **322** receives the camera pose from the mobile device (where the user has consented to the disclosure of this information) and uses that information together with maps **324** of the floor of the office to calculate map display data to aid the person **302** in navigating the office floor. The map display data is sent to the mobile device and may be displayed at map display **318**.

[0039] An example where the downstream system **122** is a robotic system is now described with reference to FIG. **3**. A robot vehicle **312** moves along the corridor **306** and captures images using one or more cameras **326** on the robot vehicle. A camera pose tracker **328** at the robot vehicle is able to calculate pose of the camera(s) where the scene is already known to the robot vehicle.

[0040] FIG. **4** is a schematic diagram of an example decision forest comprising three decision trees: a first tree **400** (denoted tree Ψ_1); a second tree **402** (denoted tree Ψ_2); and a third tree **404** (denoted tree Ψ_3). Each decision tree comprises a root node (e.g. root node **406** of the first decision tree **700**), a plurality of internal nodes, called split nodes (e.g. split node **408** of the first decision tree **400**), and a plurality of leaf nodes (e.g. leaf node **410** of the first decision tree **400**).

[0041] In operation, each root and split node of each tree performs a binary test (or possibly an n-ary test) on the input data and based on the result directs the data to the left or right child node. The leaf nodes do not perform any action; they store accumulated scene coordinates (and optionally other information). For example, probability distributions may be stored representing the accumulated scene coordinates.

[0042] FIG. **5** is a flow diagram of a method of training a random decision forest to predict correspondences between image elements and scene coordinates. This is illustrated in the upper part of FIG. **5** above the dotted line in the region labeled "training". The lower part of FIG. **5** below the dotted line shows method steps at test time when the trained random decision forest is used to predict (or estimate) correspondences between image elements from an image of a scene and points in the scene's 3D world coordinate frame (scene coordinates).

[0043] A random decision forest is trained **502** to enable image elements to generate predictions of correspondences between themselves and scene coordinates. During training, labeled training images **500** of at least one scene, such as scene A, are used. For example, a labeled training image comprises, for each image element, a point in a scene's 3D world coordinate frame which the image element depicts. To obtain the labeled training images various different methods may be used to capture images **516** of scene A and record or calculate the pose of the camera for each captured image. Using this data a scene coordinate may be calculated indicating the world point depicted by an image element. To capture the images and record or calculate the associated camera pose, one approach is to carry out camera tracking from depth camera input **512**. For example as described in US patent application "Real-time camera tracking using depth maps" Newcombe, Richard et al. published on Aug. 2, 2012 US20120196679. Another approach is to carry out dense reconstruction and camera tracking from RGB camera input **514**. It is also possible to use a CAD model to generate synthetic training data. The training images themselves (i.e. not the label images) may be real or synthetic.

[0044] An example of the training process of box **502** is described below with reference to FIG. **6**. The result of training is a trained random decision forest **504** for scene A (in the case where the training images were of scene A).

[0045] At test time an input image **508** of scene A is received and a plurality of image elements are selected from

the input image. The image elements may be selected at random or in another manner (for example, by selecting such that spurious or noisy image elements are omitted). Each selected image element may be applied **506** to the trained decision forest to obtain predicted correspondences **510** between those image elements and points in the scene's 3D world coordinate frame.

[0046] FIG. **6** is a flow diagram of a method of training a random decision forest using images of a scene where image elements have labels indicating their corresponding scene coordinates. A training set of images of a scene is received **600** where the image elements have labels indicating the scene coordinate of the scene point they depict. A number of trees to be used in the decision forest is selected **602**, for example, between 3 and 20 trees.

[0047] A decision tree from the decision forest is selected **604** (e.g. the first decision tree **600**) and the root node **606** is selected **606**. At least a subset of the image elements from each of the training images are then selected **608**. For example, the image may be filtered to remove noisy or spurious image elements.

[0048] A random set of test parameters (also called weak learners) are then generated **610** for use by the binary test performed at the root node as candidate features. In one example, the binary test is of the form: $\xi > f(x;\theta) > \tau$, such that $f(x;\theta)$ is a function applied to image element x with parameters θ, and with the output of the function compared to threshold values ξ and τ. If the result of $f(x;\theta)$ is in the range between ξ and τ then the result of the binary test is true. Otherwise, the result of the binary test is false. In other examples, only one of the threshold values ξ and τ can be used, such that the result of the binary test is true if the result of $f(x;\theta)$ is greater than (or alternatively less than) a threshold value. In the example described here, the parameter θ defines a feature of the image.

[0049] A candidate function $f(x;\theta)$ makes use of image information which is available at test time. The parameter θ for the function $f(x;\theta)$ is randomly generated during training. The process for generating the parameter θ can comprise generating random spatial offset values in the form of a two or three dimensional displacement. The result of the function $f(x;\theta)$ is then computed by observing the depth (or intensity value in the case of an RGB image and depth image pair) value for one or more test image elements which are displaced from the image element of interest x in the image by spatial offsets. The spatial offsets are optionally made depth invariant by scaling by 1/depth of the image element of interest. Where RGB images are used without depth images the result of the function $f(x;\theta)$ may be computed by observing the intensity value in a specified one of the red, green or blue color channel for one or more test image elements which are displaced from the image element of interest x in the image by spatial offsets.

[0050] The result of the binary test performed at a root node or split node determines which child node an image element is passed to. For example, if the result of the binary test is true, the image element is passed to a first child node, whereas if the result is false, the image element is passed to a second child node.

[0051] The random set of test parameters generated comprise a plurality of random values for the function parameter θ and the threshold values ξ and τ. In order to inject randomness into the decision trees, the function parameters θ of each split node are optimized only over a randomly sampled subset Θ of all possible parameters. This is an effective and simple way of injecting randomness into the trees, and increases generalization.

[0052] Then, every combination of test parameter may be applied **612** to each image element in the set of training images. In other words, available values for θ (i.e. $\theta_i \in \Theta$) are tried one after the other, in combination with available values of ξ and τ for each image element in each training image. For each combination, criteria (also referred to as objectives) are calculated **614**. The combination of parameters that optimize the criteria is selected **614** and stored at the current node for future use.

[0053] In an example the objective is a reduction-in-variance objective expressed as follows:

$$Q(S_n\theta) = V(S_n) - \sum_{d\in\{L,R\}} \frac{|S_n^d(\theta)|}{|S_n|} V(S_n^d(\theta))$$

[0054] Which may be expressed in words as the reduction in variance of the training examples at split node n, with weak learner parameters θ equal to the variance of all the training examples which reach that split node minus the sum of the variances of the training examples which reach the left and right child nodes of the split node. The variance may be calculated as:

$$V(S) = \frac{1}{|S|} \sum_{(p,m)\in S} \|m - \bar{m}\|_2^2$$

[0055] Which may be expressed in words as, the variance of a set of training examples S equals the average of the differences between the scene coordinates m and the mean of the scene coordinates in S.

[0056] As an alternative to a reduction-in-variance objective, other criteria can be used, such as logarithm of the determinant, or the continuous information gain.

[0057] It is then determined **616** whether the value for the calculated criteria is less than (or greater than) a threshold. If the value for the calculated criteria is less than the threshold, then this indicates that further expansion of the tree does not provide significant benefit. This gives rise to asymmetrical trees which naturally stop growing when no further nodes are beneficial. In such cases, the current node is set **618** as a leaf node. Similarly, the current depth of the tree is determined (i.e. how many levels of nodes are between the root node and the current node). If this is greater than a predefined maximum value, then the current node is set **618** as a leaf node. Each leaf node has scene coordinate predictions which accumulate at that leaf node during the training process as described below.

[0058] It is also possible to use another stopping criterion in combination with those already mentioned. For example, to assess the number of example image elements that reach the leaf. If there are too few examples (compared with a threshold for example) then the process may be arranged to stop to avoid overfitting. However, it is not essential to use this stopping criterion.

[0059] If the value for the calculated criteria is greater than or equal to the threshold, and the tree depth is less than the maximum value, then the current node is set **620** as a split

node. As the current node is a split node, it has child nodes, and the process then moves to training these child nodes. Each child node is trained using a subset of the training image elements at the current node. The subset of image elements sent to a child node is determined using the parameters that optimized the criteria. These parameters are used in the binary test, and the binary test performed **622** on all image elements at the current node. The image elements that pass the binary test form a first subset sent to a first child node, and the image elements that fail the binary test form a second subset sent to a second child node.

[0060] For each of the child nodes, the process as outlined in blocks **610** to **622** of FIG. **6** are recursively executed **624** for the subset of image elements directed to the respective child node. In other words, for each child node, new random test parameters are generated **610**, applied **612** to the respective subset of image elements, parameters optimizing the criteria selected **614**, and the type of node (split or leaf) determined **616**. If it is a leaf node, then the current branch of recursion ceases. If it is a split node, binary tests are performed **622** to determine further subsets of image elements and another branch of recursion starts. Therefore, this process recursively moves through the tree, training each node until leaf nodes are reached at each branch. As leaf nodes are reached, the process waits **626** until the nodes in all branches have been trained. Note that, in other examples, the same functionality can be attained using alternative techniques to recursion.

[0061] Once all the nodes in the tree have been trained to determine the parameters for the binary test optimizing the criteria at each split node, and leaf nodes have been selected to terminate each branch, then scene coordinates may be accumulated **628** at the leaf nodes of the tree. This is the training stage and so particular image elements which reach a given leaf node have specified scene coordinates known from the ground truth training data. A representation of the scene coordinates may be stored **630** using various different methods. For example by aggregating the scene coordinates or storing statistics representing the distribution of scene coordinates.

[0062] In some embodiments a multi-modal distribution is fitted to the accumulated scene coordinates. Examples of fitting a multi-model distribution include using expectation maximization (such as fitting a Gaussian mixture model); using mean shift mode detection; using any suitable clustering process such as k-means clustering, agglomerative clustering or other clustering processes. Characteristics of the clusters or multi-modal distributions are then stored rather than storing the individual scene coordinates. In some examples a handful of the samples of the individual scene coordinates may be stored.

[0063] A weight may also be stored for each cluster or mode. For example, a mean shift mode detection algorithm is used and the number of scene coordinates that reached a particular mode may be used as a weight for that mode. Mean shift mode detection is an algorithm that efficiently detects the modes (peaks) in a distribution defined by a Parzen window density estimator. In another example, the density as defined by a Parzen window density estimator may be used as a weight. A Parzen window density estimator (also known as a kernel density estimator) is a non-parametric process for estimating a probability density function, in this case of the accumulated scene coordinates. A Parzen window density estimator takes a bandwidth parameter which can be thought of as controlling a degree of smoothing.

[0064] In an example a sub-sample of the training image elements that reach a leaf are taken and input to a mean shift mode detection process. This clusters the scene coordinates into a small set of modes. One or more of these modes may be stored for example, according to the number of examples assigned to each mode.

[0065] Once the accumulated scene coordinates have been stored it is determined **632** whether more trees are present in the decision forest. If so, then the next tree in the decision forest is selected, and the process repeats. If all the trees in the forest have been trained, and no others remain, then the training process is complete and the process terminates **634**.

[0066] Therefore, as a result of the training process, one or more decision trees are trained using empirical training images. Each tree comprises a plurality of split nodes storing optimized test parameters, and leaf nodes storing associated scene coordinates or representations of aggregated scene coordinates. Due to the random generation of parameters from a limited subset used at each node, and the possible subsampled set of training data used in each tree, the trees of the forest are distinct (i.e. different) from each other.

[0067] The training process may be performed in advance of using the trained prediction system to identify scene coordinates for image elements of depth or RGB images of one or more known scenes. The decision forest and the optimized test parameters may be stored on a storage device for use in identifying scene coordinates of image elements at a later time.

[0068] FIG. **7** illustrates a flowchart of a process for predicting scene coordinates in a previously unseen image (a depth image, an RGB image, or a pair of rectified depth and RGB images) using a decision forest that has been trained as described with reference to FIG. **6**. Firstly, an unseen image is received **700**. An image is referred to as 'unseen' to distinguish it from a training image which has the scene coordinates already specified.

[0069] An image element from the unseen image is selected **702**. A trained decision tree from the decision forest is also selected **704**. The selected image element is pushed **706** through the selected decision tree, such that it is tested against the trained parameters at a node, and then passed to the appropriate child in dependence on the outcome of the test, and the process repeated until the image element reaches a leaf node. Once the image element reaches a leaf node, the accumulated scene coordinates (from the training stage) associated with this leaf node are stored **708** for this image element. In an example where the leaf node stores one or more modes of a distribution of scene coordinates, one or more of those modes are stored for this image element.

[0070] If it is determined **710** that there are more decision trees in the forest, then a new decision tree is selected **704**, the image element pushed **706** through the tree and the accumulated scene coordinates stored **708**. This is repeated until it has been performed for all the decision trees in the forest. The final prediction of the forest for an image element may be an aggregate of the scene coordinates obtained from the leaf found at each tree. Where one or more modes of a distribution of scene coordinates are stored at the leaves, the final prediction of the forest may be a union of the modes from the leaf found at each tree. Note that the process for pushing an image element through the plurality of trees in

the decision forest can also be performed in parallel, instead of in sequence as shown in FIG. **7**.

[0071] It is then determined **712** whether further unanalyzed image elements are to be assessed, and if so another image element is selected and the process repeated. The camera pose inference engine may be arranged to determine whether further unanalyzed image elements are to be assessed as described below with reference to FIG. **8**.

[0072] FIG. **8** is a flow diagram of a method at a camera pose inference engine of using scene-coordinate-image element pairs to infer camera pose. As mentioned above the camera pose inference engine may use an energy optimization approach to find a camera pose which is a good fit to a plurality of image element—scene coordinate pairs predicted by the scene coordinate decision forest. In the case that depth images, or both depth and RGB images are used, an example energy function may be:

$$E(H) = \sum_{i \in I} \rho\left(\min_{m \in M_i} \|m - Hx_i\|_2 \right) = \sum_{i \in I} e_i(H)$$

[0073] Where $i \in I$ is an image element index; ρ is a robust error function; $m \in M_i$ represents the set of modes (3D locations in the scene's world space) predicted by the trees in the forest at image element p_i; and x_i are the 3D coordinates in camera space corresponding to pixel p_i which may be obtained by back-projecting the depth image elements. The energy function may be considered as counting the number of outliers for a given camera hypothesis H. The above notation uses homogeneous 3D coordinates.

[0074] In the case that RGB images are used without depth images the energy function may be modified by

$$E(H) = \sum_{i \in I} \rho(\min_{m \in M_i} \|\pi(KH^{-1}m - p_i)\|_2 = \sum_{i \in I} e_i(H)$$

[0075] where ρ is a robust error function, π projects from 3D to 2D image coordinates, K is a matrix that encodes the camera intrinsic parameters, and p_i is the 2D image element coordinate.

[0076] Note that E, ρ and e_i may be separated out with different superscripts such as rgb/depth in the above equations.

[0077] In order to optimize the energy function an iterative process may be used to search for good camera pose candidates amongst a set of possible camera pose candidates. Samples of image element—scene coordinate pairs are taken and used to assess the camera pose candidates. The camera pose candidates may be refined or updated using a subset of the image element-scene coordinate pairs. By using samples of image element-scene coordinate pairs rather than each image element-scene coordinate pair from an image computation time is reduced without loss of accuracy.

[0078] An example iterative process which may be used at the camera pose inference engine is now described with reference to FIG. **8**. A set of initial camera pose candidates or hypotheses is generated **800** by, for each camera pose candidate, selecting **802** three image elements from the input image (which may be a depth image, an RGB image or a pair of rectified depth and RGB images). The selection may be random or may take into account noise or missing values in the input image. It is also possible to pick pairs where the scene coordinate is more certain where certainty information is available from the forest. In some examples a minimum distance separation between the image elements may be enforced in order to improve accuracy. Each image element is pushed through the trained scene coordinate decision forest to obtain three scene coordinates. The three image element-scene coordinate pairs are used to compute **804** a camera pose using any suitable method such as the Kabsch algorithm also known as orthogonal Procrustes alignment which uses a singular value decomposition to compute the camera pose hypothesis. In some examples the set of initial camera pose candidates may include **820** one or more camera poses of previous frames where a stream of images is available. It may also include a camera pose predicted from knowledge of the camera's path.

[0079] For each camera pose hypothesis some inliers or outliers are computed **806**. Inliers and outliers are image element-scene coordinate pairs which are classified as either being consistent with a camera pose hypothesis or not. To compute inliers and outliers a batch B of image elements is sampled **808** from the input image and applied to the trained forest to obtain scene coordinates. The sampling may be random or may take into account noise or missing values in the input image. Each scene coordinate-image element pair may be classified **810** as an inlier or an outlier according to each of the camera pose hypotheses. For example, by comparing what the forest says the scene coordinate is for the image element and what the camera pose hypothesis says the scene coordinate is for the image element.

[0080] Optionally, one or more of the camera pose hypotheses may be discarded **812** on the basis of the relative number of inliers (or outliers) associated with each hypothesis, or on the basis of a rank ordering by outlier count with the other hypotheses. In various examples the ranking or selecting hypotheses may be achieved by counting how many outliers each camera pose hypothesis has. Camera pose hypotheses with fewer outliers have a higher energy according to the energy function above.

[0081] Optionally, the remaining camera pose hypotheses may be refined **814** by using the inliers associated with each camera pose to recompute that camera pose (using the Kabsch algorithm mentioned above). For efficiency the process may store and update the means and covariance matrices used by the singular value decomposition.

[0082] The process may repeat **816** by sampling another batch B of image elements and so on until one or a specified number of camera poses remains or according to other criteria (such as the number of iterations).

[0083] The camera pose inference engine is able to produce an accurate camera pose estimate at interactive rates. This is achieved without an explicit 3D model of the scene having to be computed. A 3D model of the scene can be thought of as implicitly encoded in the trained random decision forest. Because the forest has been trained to work at any valid image element it is possible to sample image elements at test time. The sampling avoids the need to compute interest points and the expense of densely evaluation the forest.

[0084] FIG. **9** is a schematic diagram of the camera pose tracker of FIG. **1** where a 3D model **902** of the scene is available. For example the 3D model may be a CAD model

or may be a dense reconstruction of the scene built up from depth images of the scene as described in US patent application "Three-dimensional environment reconstruction" Newcombe, Richard et al. published on Aug. 2, 2012 US20120194516. A pose refinement process **900** may be carried out to improve the accuracy of the camera pose **120**. The pose refinement process **900** may be an iterative closest point pose refinement as described in US patent application "Real-time camera tracking using depth maps" Newcombe, Richard et al. published on Aug. 2, 2012 US20120196679. In another example the pose refinement process **900** may seek to align depth observations from the mobile camera with surfaces of the 3D model of the scene in order to find an updated position and orientation of the camera which facilitates the alignment. This is described in U.S. patent application Ser. No. 13/749,497 filed on 24 Jan. 2013 entitled "Camera pose estimation for 3D reconstruction" Sharp et al.

[0085] The example shown in FIG. **9** has a camera pose tracker with one trained random decision forest rather than a plurality of trained random decision forests as in FIG. **1**. This is intended to illustrate that a single forest may encapsulate a plurality of scenes by training the single forest using training data from those scenes. The training data comprises scene coordinates for image elements and also labels for image elements which identify a particular scene. Each sub-scene may be given a 3D sub-region of the full 3D world coordinate space and the forest may then be trained as described above. The camera pose tracker output may comprise an estimated camera pose and a scene so that the camera pose tracker is also able to carry out scene recognition. This enables the camera pose tracker to send data to a downstream system identifying which of a plurality of possible scenes the camera is in.

[0086] FIG. **10** illustrates various components of an exemplary computing-based device **1004** which may be implemented as any form of a computing and/or electronic device, and in which embodiments of a camera pose tracker or object pose tracker may be implemented.

[0087] The computing-based device **1004** comprises one or more input interfaces **1002** arranged to receive and process input from one or more devices, such as user input devices (e.g. capture device **1008**, a game controller **1005**, a keyboard **1006**, a mouse **1007**). This user input may be used to control software applications, camera pose tracking or object pose tracking. For example, capture device **1008** may be a mobile depth camera arranged to capture depth maps of a scene. It may also be a fixed depth camera arranged to capture depth maps of an object. In another example, capture device **1008** comprises both a depth camera and an RGB camera. The computing-based device **1004** may be arranged to provide camera or object pose tracking at interactive rates.

[0088] The computing-based device **1004** also comprises an output interface **1010** arranged to output display information to a display device **1009** which can be separate from or integral to the computing device **1004**. The display information may provide a graphical user interface. In an example, the display device **1009** may also act as the user input device if it is a touch sensitive display device. The output interface **1010** may also output date to devices other than the display device, e.g. a locally connected printing device.

[0089] In some examples the user input devices **1005**, **1007**, **1008**, **1009** may detect voice input, user gestures or other user actions and may provide a natural user interface (NUI). This user input may be used to control a game or other application. The output interface **1010** may also output data to devices other than the display device, e.g. a locally connected printing device.

[0090] The input interface **1002**, output interface **1010**, display device **1009** and optionally the user input devices **1005**, **1007**, **1008**, **1009** may comprise NUI technology which enables a user to interact with the computing-based device in a natural manner, free from artificial constraints imposed by input devices such as mice, keyboards, remote controls and the like. Examples of NUI technology that may be provided include but are not limited to those relying on voice and/or speech recognition, touch and/or stylus recognition (touch sensitive displays), gesture recognition both on screen and adjacent to the screen, air gestures, head and eye tracking, voice and speech, vision, touch, gestures, and machine intelligence. Other examples of NUI technology that may be used include intention and goal understanding systems, motion gesture detection systems using depth cameras (such as stereoscopic camera systems, infrared camera systems, rgb camera systems and combinations of these), motion gesture detection using accelerometers/gyroscopes, facial recognition, 3D displays, head, eye and gaze tracking, immersive augmented reality and virtual reality systems and technologies for sensing brain activity using electric field sensing electrodes (EEG and related methods).

[0091] Computer executable instructions may be provided using any computer-readable media that is accessible by computing based device **1004**. Computer-readable media may include, for example, computer storage media such as memory **1012** and communications media. Computer storage media, such as memory **1012**, includes volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EPROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other non-transmission medium that can be used to store information for access by a computing device.

[0092] In contrast, communication media may embody computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave, or other transport mechanism. As defined herein, computer storage media does not include communication media. Therefore, a computer storage medium should not be interpreted to be a propagating signal per se. Propagated signals may be present in a computer storage media, but propagated signals per se are not examples of computer storage media. Although the computer storage media (memory **1012**) is shown within the computing-based device **1004** it will be appreciated that the storage may be distributed or located remotely and accessed via a network or other communication link (e.g. using communication interface **1013**).

[0093] Computing-based device **1004** also comprises one or more processors **1000** which may be microprocessors, controllers or any other suitable type of processors for processing computing executable instructions to control the operation of the device in order to provide real-time camera

tracking. In some examples, for example where a system on a chip architecture is used, the processors **1000** may include one or more fixed function blocks (also referred to as accelerators) which implement a part of the method of real-time camera tracking in hardware (rather than software or firmware).

[0094] Platform software comprising an operating system **1014** or any other suitable platform software may be provided at the computing-based device to enable application software **1016** to be executed on the device. Other software than may be executed on the computing device **1004** comprises: camera/object pose tracker **1018** which comprises a pose inference engine. A trained support vector machine regression system may also be provided and/or a trained Gaussian process regression system. A data store **1020** is provided to store data such as previously received images, camera pose estimates, object pose estimates, trained random decision forests registration parameters, user configurable parameters, other parameters, 3D models of scenes, game state information, game metadata, map data and other data.

[0095] The term 'computer' or 'computing-based device' is used herein to refer to any device with processing capability such that it can execute instructions. Those skilled in the art will realize that such processing capabilities are incorporated into many different devices and therefore the terms 'computer' and 'computing-based device' each include PCs, servers, mobile telephones (including smart phones), tablet computers, set-top boxes, media players, games consoles, personal digital assistants and many other devices.

[0096] The methods described herein may be performed by software in machine readable form on a tangible storage medium e.g. in the form of a computer program comprising computer program code means adapted to perform all the steps of any of the methods described herein when the program is run on a computer and where the computer program may be embodied on a computer readable medium. Examples of tangible storage media include computer storage devices comprising computer-readable media such as disks, thumb drives, memory etc. and do not include propagated signals. Propagated signals may be present in a tangible storage media, but propagated signals per se are not examples of tangible storage media. The software can be suitable for execution on a parallel processor or a serial processor such that the method steps may be carried out in any suitable order, or simultaneously.

[0097] This acknowledges that software can be a valuable, separately tradable commodity. It is intended to encompass software, which runs on or controls "dumb" or standard hardware, to carry out the desired functions. It is also intended to encompass software which "describes" or defines the configuration of hardware, such as HDL (hardware description language) software, as is used for designing silicon chips, or for configuring universal programmable chips, to carry out desired functions.

[0098] Those skilled in the art will realize that storage devices utilized to store program instructions can be distributed across a network. For example, a remote computer may store an example of the process described as software. A local or terminal computer may access the remote computer and download a part or all of the software to run the program. Alternatively, the local computer may download pieces of the software as needed, or execute some software instructions at the local terminal and some at the remote computer (or computer network). Those skilled in the art will also realize that by utilizing conventional techniques known to those skilled in the art that all, or a portion of the software instructions may be carried out by a dedicated circuit, such as a DSP, programmable logic array, or the like.

[0099] Any range or device value given herein may be extended or altered without losing the effect sought, as will be apparent to the skilled person.

[0100] Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

[0101] It will be understood that the benefits and advantages described above may relate to one embodiment or may relate to several embodiments. The embodiments are not limited to those that solve any or all of the stated problems or those that have any or all of the stated benefits and advantages. It will further be understood that reference to 'an' item refers to one or more of those items.

[0102] The steps of the methods described herein may be carried out in any suitable order, or simultaneously where appropriate. Additionally, individual blocks may be deleted from any of the methods without departing from the spirit and scope of the subject matter described herein. Aspects of any of the examples described above may be combined with aspects of any of the other examples described to form further examples without losing the effect sought.

[0103] The term 'comprising' is used herein to mean including the method blocks or elements identified, but that such blocks or elements do not comprise an exclusive list and a method or apparatus may contain additional blocks or elements.

[0104] It will be understood that the above description is given by way of example only and that various modifications may be made by those skilled in the art. The above specification, examples and data provide a complete description of the structure and use of exemplary embodiments. Although various embodiments have been described above with a certain degree of particularity, or with reference to one or more individual embodiments, those skilled in the art could make numerous alterations to the disclosed embodiments without departing from the spirit or scope of this specification.

1. A method of calculating pose of an entity comprising:

receiving, at a processor, at least one image where the image is either: of the entity and captured by a fixed camera, or of a scene captured by a mobile camera in the scene;

applying image elements of the at least one image to a trained machine learning system to obtain a plurality of associations between image elements and points in either entity coordinates or scene coordinates; and

calculating the pose of the entity from the associations.

2. A method as claimed in claim **1** where the entity is a mobile camera and the pose of the camera is calculated.

3. A method as claimed in claim **1** where the entity is an object and the pose of the object is calculated using the at least one image captured by a fixed camera.

4. A method as claimed in claim **1** comprising calculating the pose of the entity as parameters having six degrees of

freedom, three indicating rotation of the entity and three indicating position of the entity.

5. A method as claimed in claim **1**, the machine learning system having been trained using images with image elements labeled either with scene coordinates or object coordinates.

6. A method as claimed in claim **1** where the machine learning system is a random decision forest.

7. A method as claimed in claim **1** where the machine learning system comprises a plurality of trained random forests and the method comprises applying the image elements of the at least one image to the plurality of trained random forests, each random forest having been trained using images from a different one of a plurality of scenes, and calculating which of the scenes the mobile camera was in when the at least one image was captured.

8. A method as claimed in claim **1** the machine learning system having been trained using images of a plurality of scenes with image elements labeled with scene identifiers and labeled with scene coordinates of points in the scene the image elements depict.

9. A method as claimed in claim **1** comprising applying only a subsample of the image elements of the at least one image to the trained machine learning system.

10. A method as claimed in claim **1** comprising calculating the pose by searching amongst a set of possible pose candidates and using samples of associations between image elements and points to assess the pose candidates.

11. A method as claimed in claim **1** comprising receiving at the processor, a stream of images, and calculating the pose by searching amongst a set of possible pose candidates which includes a pose calculated from another image in the stream.

12. A method as claimed in claim **1** at least partially carried out using hardware logic selected from any one or more of: a field-programmable gate array, a program-specific integrated circuit, a program-specific standard product, a system-on-a-chip, a complex programmable logic device, a graphics processing unit.

13. A method as claimed in claim **1** where the entity is a mobile camera and the pose of the camera is calculated, the method comprising accessing a 3D model of the scene and refining the camera pose using the accessed 3D model.

14. A method comprising:

receiving, at a processor, a plurality of images, each image having a plurality of image elements labeled with coordinates of points either in a scene the image elements depict or of an object the image elements depict;

training a machine learning system using the received plurality of images such that when an image element from another image is applied to the machine learning system, an estimate of a coordinate of a point the image element depicts is produced.

15. A method as claimed in claim **14** comprising receiving a plurality of images of sub-scenes, each having a plurality of image elements labeled with scene coordinates in a space in which the sub-scenes are embedded; and training the machine learning system using the received plurality of images of sub-scenes.

16. A pose tracker comprising:

a processor arranged to receive at least one image either of an object captured by a fixed camera, or of a scene captured by a mobile camera;

the processor arranged to apply image elements of the at least one image to a trained machine learning system to obtain a plurality of associations between image elements and points in either object coordinates or scene coordinates; and

a pose inference engine arranged to calculate a position and orientation of either the object or the mobile camera from the associations.

17. A pose tracker as claimed in claim **16** the processor arranged to apply only a subsample of the image elements of the at least one image to the trained machine learning system.

18. A pose tracker as claimed in claim **16** the pose inference engine arranged to calculate the pose by searching amongst a set of possible pose candidates and using samples of associations between image elements and points in either object coordinates or scene coordinates to assess the pose candidates.

19. A pose tracker as claimed in claim **16** the processor arranged to receive a stream of images, and comprising a pose inference engine arranged to calculate the pose by searching amongst a set of possible pose candidates which includes a pose calculated from another image in the stream.

20. A pose tracker as claimed in claim **16** at least partially implemented using hardware logic selected from any one or more of: a field-programmable gate array, a program-specific integrated circuit, a program-specific standard product, a system-on-a-chip, a complex programmable logic device, a graphics processing unit.

* * * * *