

(19) **DANMARK**



Patent- og  
Varemærkestyrelsen

(12)

## Oversættelse af europæisk patentskrift

(10) **DK/EP 3066591 T3**

- 
- (51) Int.Cl.: **G 06 F 17/30 (2006.01)** **G 06 K 9/00 (2006.01)** **G 06 K 9/62 (2006.01)**  
**G 06 T 11/60 (2006.01)**
- (45) Oversættelsen bekendtgjort den: **2019-10-21**
- (80) Dato for Den Europæiske Patentmyndigheds bekendtgørelse om meddelelse af patentet: **2019-07-10**
- (86) Europæisk ansøgning nr.: **15731669.6**
- (86) Europæisk indleveringsdag: **2015-02-10**
- (87) Den europæiske ansøgnings publiceringsdag: **2016-09-14**
- (86) International ansøgning nr.: **IB2015000908**
- (87) Internationalt publikationsnr.: **WO2015125025**
- (30) Prioritet: **2014-02-10 US 201461937859 P**
- (84) Designerede stater: **AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR**
- (73) Patenthaver: **GEENEE GmbH, Obentrautstrasse 72, 10963 Berlin, Tyskland**
- (72) Opfinder: **HOLZSCHNEIDER, Philipp, Schlangenbaderstrasse 28c, 14179 Berlin, Tyskland**  
**MAGERS, Thorsten Otto Wolfgang, Schoenleinstrasse 8, 10967 Berlin, Tyskland**  
**PRIEHN, Hannes, Schoenleinstrasse 8, 10967 Berlin, Tyskland**
- (74) Fuldmægtig i Danmark: **NORDIC PATENT SERVICE A/S, Bredgade 30, 1260 København K, Danmark**
- (54) Benævnelse: **SYSTEMER OG FREMGANGSMÅDER TIL BILLEDKARAKTERISTIKBASERET GENKENDELSE**
- (56) Fremdragne publikationer:  
**US-A1- 2005 162 523**  
**US-A1- 2010 318 515**  
**XIN XIN ET AL: "Large Visual Repository Search with Hash Collision Design Optimization", IEEE MULTIMEDIA, IEEE SERVICE CENTER, NEW YORK, NY, US, vol. 20, no. 2, 1 April 2013 (2013-04-01), pages 62-71, XP011515223, ISSN: 1070-986X, DOI: 10.1109/MMUL.2013.22**  
**MANI MALEK ESMAEILI ET AL: "A Fast Approximate Nearest Neighbor Search Algorithm in the Hamming Space", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE COMPUTER SOCIETY, USA, vol. 34, no. 12, 1 December 2012 (2012-12-01), pages 2481-2488, XP011490765, ISSN: 0162-8828, DOI: 10.1109/TPAMI.2012.170**  
**FENG HAO ET AL: "A Fast Search Algorithm for a Large Fuzzy Database", IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE, PISCATAWAY, NJ, US, vol. 3, no. 2, 1 June 2008 (2008-06-01), pages 203-212, XP011207109, ISSN: 1556-6013**  
**Yan Ke ET AL: "Efficient Near-duplicate Detection and Sub-image Retrieval", Proceedings of ACM International Conference on Multimedia (MM), 1 January 2004 (2004-01-01), XP055094747, Retrieved from the Internet: URL:<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.5010&rep=rep1&type=pdf> [retrieved on 2014-01-03]**

Fortsættes ...



# DESCRIPTION

## RELATED APPLICATIONS

**[0001]** This application claims priority to U.S. Provisional Patent Application No. 61/937,859, filed February 10, 2014.

## FIELD

**[0002]** The present disclosure relates to visual object recognition, and more specifically, to methods and systems for recognition of content and objects in digital images, and methods and systems for rapid delivery of additional content and/or information related to the recognized content objects. Methods and systems disclosed herein may also be applicable to search services and content-based image retrieval and recognition processes.

## BACKGROUND

**[0003]** In today's world, nearly everyone has a camera-enabled mobile device with them at all times. Mobile device users have grown used to being able to use the mobile device to obtain additional information about the world around them using their mobile device. Mobile device users today will use the Internet to obtain additional information, research prices, view products before buying, and even purchase items or content. Users today want such access as quickly and efficiently as possible, with the fewest number of steps as possible.

**[0004]** A desire for more information can be triggered by, for example, viewing an object or scene in the world around us, something seen in print, such as a billboard, poster, or movie, or by something seen on a screen, such as a movie, TV show, website, or other digital content.

**[0005]** There are existing techniques for facilitating the delivery of additional information about items or content to a user's mobile device. For example, a marker, such as a one-dimensional barcode or two-dimensional QR-code, can be attached physically to a product, or can be printed next to an image of the product on a printed page, poster, or billboard. In some cases, artificially-generated patterns may be added to images. By scanning the code or artificially-generated pattern, either with a special purpose scanning device or using the camera function and an app on the user's mobile device, more information can be delivered to the user. The additional information can either be directly coded into the marker, or the information coded into the marker can be used to retrieve additional information from a database.

**[0006]** Technically more sophisticated is the recognition of images without the insertion of

artificial markers, since not every object has, or can have, such a marker. For these cases, techniques have been developed to recognize images using a combination of object recognition algorithms and pattern matching. The Viola-Jones method, for example, as described in "The Rapid Object Detection Using a Boosted Cascade of Simple Features," by Paul Viola and Michael Jones, performs a cascade of predefined scan operations in order to assess the probability of the presence of a certain shape in the image, and a classification algorithm, such as AdaBoost, to identify the object. Using an "integral image" as a random pixel access data structure, the Viola-Jones method selects a small number of critical visual features from a larger generic set to yield a subset of classifiers, then combines progressively more complex classifiers in a "cascade" (AdaBoost) to allow background regions to be discarded and more computations on promising object-like regions to be performed. Classification methods typically have two phases, a training phase and a test phase, during which the training data may be used to identify objects. Classification methods are computationally intensive and rely on the careful preselection of training images. These types of object recognition methods are typically used in the detection of objects with the same shape and characteristics, such as faces, traffic signs, brand logos, and the like.

**[0007]** Other approaches utilize distinct attributes or features present in images for image-based detection and recognition. In these systems, characteristics are extracted from a set of training images, and then the system detects whether there are corresponding characteristics among either a set of snapshots, or between a snapshot and a training set of images. Applications of image-based detection and recognition span from panorama and image stitching, sparse 3D Reconstruction, augmented reality (e.g., Microsoft® Photosynth™, VisualSFM™, Qualcomm® Vuforia™), to image search and recognition services provided by Google® Goggles™ and Kooaba™ / Vuforia™ Cloud Recognition. These image-based recognition techniques are used only to recognize objects and do not extract extra information deposited within the image. Further, existing technologies typically require the transmission of data-dense media files (such as the image itself or video and/or audio data) from a capturing device (e.g., a smartphone) to a processing server over a network, which further delays the recognition of the object. Existing methods also require that all additional information associated with the object be transmitted from a server back to the mobile device, thereby taking time and being unusable in situations where there is no network connection to the server.

**[0008]** There exist some methods and systems for extracting certain features from video and generating a "fingerprint" for transmitting to a content identification server for use in identifying the content. U.S. Patent No. 8,793,274 to Yu, for example, extracts VDNA (Video DNA) fingerprints from captured contents. The '274 patent, however, is specifically concerned with extracting fingerprints from video (that is capturing a sequence of images), including from the accompanying audio.

**[0009]** There also have been some attempts to provide some methods and systems for automatic recognition of media content, but all have to date been unsatisfactory. Any existing methods and systems that work for stationary media do not easily adapt well to use with video.

Systems that generate VDNA fingerprints are computationally intensive and use large media files that are difficult to transmit quickly over a network. Moreover, any system that uses VDNA fingerprints must be concerned with spikes in usage, because multiple accesses over the network at the same time will further exacerbate the bandwidth problem. Systems that attempt to recognize video media and objects therein must solve the problem of capturing a good usable image that contains the items of interest. Such systems must also account for the time delay that inevitably occurs between the moment when the scene that may have interested a user was displayed and the moment when the user initiates capture.

**[0010]** Hence, there is a need for systems and methods that require only a minimal amount of media data (such as a single image, instead of a VDNA-style fingerprint generation which requires using a series of images together with audio data) to detect or recognize an object or content, and which does not require transmission of media data over a network. Moreover, there is also a need for a system that is scalable to handle large volume of training data from the video frames, to overcome limitations that are typically associated with image recognitions using video frames as training set. Such limitations may include comparing vast amount of single images or video frames to classical image recognition domains, and huge redundancy in the training data set generated from video frames. Furthermore, there is also a need to speed up and improve the accuracy of the recognition process, especially when the image recognition is performed over a video. Lastly, there is also a need to filter out redundant queries to handle spikes in queries, to reserve precious network and computation resources.

**[0011]** XIN XIN ET AL: "Large Visual Repository Search with Hash Collision Design Optimization", 1 April 2013, XP011515223, is directed towards a means of performing a visual search over a large image repository. MANI MALEK ESMAEILE ET AL: "A Fast Approximate Nearest Neighbor Search Algorithm in the Hamming Space", 1 December 2012, XP011490765 is directed towards a nearest neighbor search algorithm for a Hamming space.

### **BRIEF DESCRIPTION OF THE DRAWINGS**

**[0012]** The accompanying drawings, which are incorporated in and constitute a part of this disclosure, illustrate several embodiments and, together with the description, serve to explain the disclosed principles. In the drawings:

**FIG. 1A** is a diagram illustrating an exemplary system for providing users with additional information related to an item of interest in displayed digital content, consistent with embodiments of the present disclosure.

**FIG. 1B** illustrates use of exemplary methods and systems described herein for obtaining more information about objects or things in displayed digital content.

**FIG. 1C** illustrates use of exemplary methods and systems described herein for providing more information to users about objects or things in a digital media.

**FIG. 2A** illustrates the prior art process of generating from the salient points in an image a collection of feature descriptor vectors to be assembled as image fingerprints.

**FIG. 2B** are diagrams illustrating prior art methods for finding a fitting transform that matches the corresponding feature locations of the query image and the reference image.

**FIG. 3** is a block diagram illustrating an exemplary system for obtaining additional information on an object of interest in an image snapshot, consistent with embodiments of the present disclosure.

**FIG. 4A** is a flow chart illustrating an exemplary method for querying an Image Feature Index database to process a live camera video feed as input, consistent with embodiments of the present disclosure.

**FIG. 4B** is a flow chart illustrating an exemplary method for processing the result of querying the Image Feature Index database to find matching image, consistent with embodiments of the present disclosure.

**FIG. 4C** is a flow chart illustrating an exemplary method for querying a cached image set before querying the image feature index database to find matching image, consistent with embodiments of the present disclosure.

**FIGs. 5A-5H** are diagrams illustrating prior art methods for querying, organizing and accessing the image feature index database.

**FIGs. 6A-6E** are diagrams illustrating other exemplary methods for querying, organizing and accessing the Image Feature Index database, consistent with embodiments of the present disclosure.

**FIG. 7** is a block diagram illustrating an exemplary system for processing image fingerprints, consistent with embodiments of the present disclosure.

**FIG. 8** is a block diagram illustrating another exemplary system for processing image fingerprints, consistent with embodiments of the present disclosure.

**FIG. 9** is a block diagram illustrating an exemplary system on which embodiments of the present disclosure can be implemented.

#### **DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS**

**[0013]** Reference will now be made in detail to the embodiments, the examples of which are illustrated in the accompanying drawings. Whenever possible, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

**[0014]** In accordance with aspects of the invention there are provided a server for identifying and transmitting to a mobile device additional information relating to a query image obtained by a user with the mobile device, a computer-method for identifying and displaying additional information related to the first query image obtained by a user with a mobile device, and a computer readable medium as set out in claims 1, 2, and 5 respectively.

**[0015]** Methods and systems are described herein that provide users with an information rich viewing experience, by providing users with additional information about people, places, and things that they see in digital content or the world around them. The methods and systems described herein offer several advantages over prior art systems. First, methods and systems herein operate on a single image frame, whether captured from video, print, or the world around the user. The methods and systems herein may be practiced by an application operated on a user's mobile device connected to a web-based server, provided the mobile device has image capture functionality, and the server provides an image recognition service. In fact, one of the advantages of the inventions disclosed herein is that these methods and systems may be implemented using photos made with handheld consumer cameras as found in smart phones and tablets.

**[0016]** In methods and systems described herein, a "fingerprint" of the snapshot is extracted and used to identify additional information. Methods and systems are described herein for identifying and describing image features and generating a unique fingerprint of the single image. In addition, the original image need not be stored on the user's mobile device, as the fingerprint may be used for displaying back to the user with the additional information, if desired.

**[0017]** Using systems and methods described herein, retrieving additional information by identifying the media or content related to the initial snapshot is performed using the fingerprint as a query. Image Feature-Based Recognition, as described herein, uses a feature index to identify a smaller set of candidate matches from a larger database of images. Use of the feature index enables faster searches over large image sets. Existing solutions using feature indexing fail in scenarios with huge sets (hundreds of millions) image features with a great number of similarities, as typically seen in collections of single frames extracted from video material. Instead, the methods and systems described herein overcome the problems with the prior art using a distance metric and a radical hash table design exploiting probabilistic effects. The methods and systems described herein automatically sort out distinct image features over redundant ones, allowing only the more distinctive data points to remain resident within the index, yielding a lean index that can be quickly used in the identification process. In contrast, traditional Approximate Nearest-Neighbor (ANN) methods store all data points unfiltered which greatly deteriorates their matching performance.

**[0018]** The methods and systems described herein provide means for handling large amounts of image data while maintaining image resolution and distinctiveness. The methods and systems described herein will also handle spikes in request traffic due to high periods of use.

[0019] Further, in some embodiments described herein, additional information may be identified without performing full-scale image recognition on the mobile device. Methods and systems described herein introduce use of a stand-alone replicable layer that operates between the web-communication layer and a search system on a server to intercept and handle many of the requests. By selectively caching and identifying similar or equivalent requests at the mobile device, fewer requests need to be forwarded to the server.

[0020] The multiple inventions described herein may be implemented alone or in various combinations with one another. Each of the various inventions will be described in further detail below.

[0021] **FIG. 1A** is a diagram showing at a high level a system 100 for connecting content providers 101 to users 102 through visual media 103, using some or all of the inventions described herein. Content providers 101 may be, for example, producers, retailers, advertisers, media content owners, and other entities that have product or content. Visual media 103 may be, for example, a television broadcast, movie, print, or billboard. The visual media can be delivered, via distribution network 104, to a video media broadcasting platform 105 or a printed media 106.

[0022] In the system shown in **FIG. 1A**, a user 102 has a mobile device, such as tablet 121 or smartphone 122. The mobile device may also be, in some embodiments, a camera with a processor, memory, and wireless communication capability. In some embodiments, the mobile device comprises an accelerometer, gyroscope, or other device for detecting the orientation of the mobile device. In embodiments described herein, the mobile device is Internet-enabled. Embodiments of the mobile device will be described in more detail below with respect to **FIGS. 3, 8, and 9**.

[0023] In some embodiments, user 102 can consume content 123 as displayed or broadcasted on platforms 105 and 106 directly. Content 123 may also be displayed directly in the user's mobile device.

[0024] In a first step, user identifies something of interest about which he or she would like to learn more information. The thing of interest may be, for example, an item displayed in content 123 or 124, such as a product, prop, clothing, or other objects. The thing of interest could also be the location of the scene itself, or the topic depicted in the scene. In some embodiments, the thing of interest is the particular content itself, for example, the movie, video, or program that is being displayed.

[0025] Using an end-user application and the camera functionality on the user's mobile device, the user takes a snapshot of the screen depicting the thing of interest. The snapshot may be a close-up of the thing itself, or can include the surrounding area of the thing of interest. The snapshot may also comprise more than just the display device that is displaying the content and the content being displayed on the screen. In some embodiments, such as when content 121 or 122 is being displayed on the user's mobile device, the user may take a snapshot of the



screen depicting the thing of interest by taking a screen capture.

**[0026] FIG. 1B** illustrates use of exemplary methods and systems described herein for obtaining more information about objects or things in displayed digital content. As shown in **FIG. 1B**, the user may hold mobile devices 121 or 122 such that a large area of view (including the thing of interest) is shown on the screen. In some embodiments, the entire area of view is captured. In some embodiments, the system may automatically determine that only a certain portion of the area of view (such as the display screen showing content) is material, and only the display screen is captured. In some embodiments, the user may indicate, via interface 150, a subset of the image shown in the display by, for example, touching or circling the thing of interest with the user's finger or a stylus, as indicated by portion 151. The end-user app may then capture only portion 151 of the image in the display of the mobile device. If a portion is indicated, portion 151 of the image is then used to generate a query to be processed by an image database. The image database can then return an image 152, which is determined to be matching with at least part of portion 151, for displaying at the mobile devices.

**[0027]** Additional information 153 may be selected based on the captured image, and displayed to the user. The additional information 153 displayed will be information related to or associated with the media content in the snapshot. The additional information 153 may be selected based on the entire snapshot or any identified portion 151..

**[0028]** In some embodiments, image 152 may be displayed along with additional information 153, either above, below, or beside additional information 153. In some embodiments, image 152 may include tags 154 (also called annotations) that correlate to the additional information 154. In at least some embodiments, the tags 154 appear next to some or all of the items as depicted in image 152, for which additional information 153 has been identified. In some embodiments, tags 154 may be used by the user to navigate to parts of additional information 153 associated with the item marked by the tag. By clicking on a tag, for example, the user may be taken to the associated information on the screen which may be search results or a link to additional information. In some embodiments, the tag is itself represents a hyperlink to additional information.

**[0029]** In the embodiment shown in **FIG. 1C**, image 163 displayed in the center has already been annotated. The diagram on the left shows a directory 162 which stores, in a hierarchy, additional information 164. Additional information 164 can be associated with a particular tag 166 pointing to an object (e.g., a sofa as illustrated in **FIG. 1C**), and can include information related to the object. The information can include images, text description, pricing, links to further information, etc., that are pertinent to the object, and the information is stored in a hierarchical structure under directory 162. As illustrated in **FIG. 1C**, after the user activates tag 166, at least part of additional information 164 is displayed. For example, image 167 of the sofa, textual description 168 of the sofa, and other information, can be displayed.

**[0030]** Returning now to **FIG. 1A**, after the user takes a snapshot, the end-user app generates a fingerprint of the snapshot or the indicated portion of the snapshot. There exist many known

methods for generating a fingerprint of an image. Generally, as a first step, these methods detect salient points in the image using a salient point detection algorithm, such as Features from Accelerated Segment Test (FAST), Harris, Maximally stable extremal regions (MSER), etc., or the equivalent. The salient points can then be used, as shown in **FIG. 2A**, to generate feature descriptor vectors. As shown in **FIG. 2A**, salient points 219a and 219b of an image 218 can be represented respectively by vectors 220a and 220b. Scale-invariant feature transform (SIFT), Speeded Up Robust Features (SURF), Binary Robust Invariant Scalable Keypoint (BRISK), Fast Retina Keypoint (FREAK), are examples of known methods for generating feature descriptor vectors. Feature descriptor vectors can be a fixed-size vector of floating point numbers or bits that characterize the pixel patch (e.g., a 64-dimensional or 128-dimensional floating point vector, or a 512-dimension bit vector.) The vector can be generated by sampling the pixel patches, arranged within the image according to pixel grid 222, within a descriptor sampling grid 224, and the vector values can be chosen such that a distance between two vectors representing two pixel patches correlates with a degree of similarity (e.g., in luminance/brightness) between the two pixel patches.

**[0031]** As shown in **FIG. 2B**, the image content can be differently oriented as compared to corresponding content in the reference image, and matches can still be found. For example, query features 242 of query image 240 may be found to match with query features 252 of reference image 250. The relative pose (or orientation) of the image content 240 can then be derived from a set of relationships between the locations of each of query features 242 within query image 240, and the locations of the corresponding reference features 252 within reference image 250. The set of relationships can be represented as a transformation matrix. In some embodiments, the transformation matrix can be calculated iteratively to transform the locations of the features of reference image 250, until the transformed locations of the features of the reference image fit with the locations of the at least some of the features of query image 240. For each iteration of transformation matrix calculation, the validity of the transformation matrix can be checked by determining whether there is a fit for a consensus set of corresponding features between the reference image (for which the feature locations has been transform with the transformation matrix) and the query image (for which the feature locations are not transformed). In some embodiments, the transformation matrix can be calculated using least squares approximation. As discussed later with respect to the inventive methods described in **FIGS. 4A** and **4B**, the determination of pose can be performed at either the mobile device or at the server.

**[0032]** A combination of feature descriptors can then be used to represent one or more visual features included in a query image, which are graphically rendered in a combination of pixel patches corresponding to the feature descriptors. The combination of feature descriptors can also be used to identify, with a certain degree of certainty, a reference image including the features descriptors. The identification of the reference image can be based on a relationship between the feature descriptors associated with the reference image and the feature descriptors extracted from (and associated with) the query image and included in the fingerprint. The relationship between the feature descriptors can determine the likelihood of success of the identification.

**[0033]** The fingerprint is a digital representation of certain features extracted from the snapshot. The fingerprint can include (or be generated based on) a set of feature descriptors tailored for the purpose of the query. The feature descriptors to be included in the generation of the image fingerprint can be selected by the user. The end-user application can then detect salient points within the image portion 151, and generate the feature descriptors accordingly. When the user does not make a selection, the end-user application may detect salient points from the entirety of the snapshot, and generate fingerprints associated with the detected salient points accordingly. In some embodiments, the user can further select a particular object in an image via a tapping action on interface 150, and the end-user application can prioritize the transmission of fingerprints associated the selected object.

**[0034]** Referring back to **FIG. 1A**, after a fingerprint of the snapshot image is obtained or generated, the fingerprint is used to obtain additional information. In at least one embodiment, the fingerprint is transmitted to a server 140 via communications link 130. The fingerprint may be transmitted wirelessly to the server via a wireless link, or via another communications link.

**[0035]** Server 140 may comprise, for example, one or more replicable query caches 141, an image matching module 142, and an image Feature Index database 143. These components, within the dashed circle, may also be referred to as the Geenee search system.

**[0036]** Replicable query caches 141 can function as an entry point to the Geenee Search System. Replicable query caches 141 can receive query fingerprints directly from mobile devices. Before forwarding an incoming query fingerprint to the image matching module 142, the incoming query fingerprint is compared to the fingerprints previously received. If a match is found, a prior response to the previously-received fingerprints (e.g., a set of candidate images, or a matching image, or no matching images, etc.) can be returned. If the incoming query fingerprint does not match with the previously-received fingerprints, the incoming query fingerprint can then be forwarded to image matching module 142.

**[0037]** In some embodiments, image matching module 142 receives the incoming query fingerprint directly. Image matching module 142 forwards the query fingerprint to Image Feature Index database 143, and further processed. From the further processing, a set of candidate images (or fingerprint references to those images) is identified and returned to the image matching module 142. Annotations database 144 stores a set of reference images, from which the candidate images are to be identified, and a mapping between the set of reference images and the feature descriptors representing features included in the reference images. For reference images that are annotated, annotation database 144 also stores a mapping between the annotation and additional information associated with the annotation.

**[0038]** After receiving the candidate image fingerprints from Image Feature Index database 143, the image matching module 142 then compares the fingerprints of the candidate images to the query fingerprint (or portion thereof), and determines if there is a match. If there is a match, it then retrieves the matching image along with annotations attached to it from the

annotations database 144, and returns this as a result.

**[0039]** As will be described in more detail below, Image Feature Index database 143 provides a searchable index for feature descriptors extracted from reference images provided previously to the annotations database. Various embodiments of Image-Feature-Index-based search consistent with the present disclosure are described below with respect to, for example, **FIGs. 6A-6E**.

**[0040]** Server 140 may also comprise a content provider interface 145. The content provider interface 145 may be implemented as a web service. Content providers may use the content provider interface to provide media material (and additional annotations to be attached thereon) intended to be included to Geenee Search. At least part of the media material, and the additional annotations, can then be stored at annotations database 144.

**[0041]** It should be understood that server 140 could be one server or a collection of servers. In some embodiments, server 140 can include a plurality of servers coupled together to form a cloud 160. Further, although Image Feature Index 143 and annotations database 144 are shown as residing on server 140, they can also reside in other storage areas accessible to server 140. In some embodiments, content provider interface 145 is a cloud-based application.

**[0042]** As an illustrative example, referring back to **FIG. 1B**, the user chooses a particular scene of a TV show, being broadcasted on a TV, and takes a snapshot of it, to include the particular scene in the query, which is then transmitted to server 140. Within server 140, the query can yield a set of candidate images including a reference image of the particular scene, if feature descriptors associated with the screenshot are also stored in Image Feature Index database 143. The Image Matching module 140 detects a matching image among the candidate images, and can return the matching image, or an indication of no matching image, as a response to the user's query. Additional information associated with the matching candidate image can also be retrieved from annotation database 144. In some embodiments, the user can further select an object in the snapshot via tapping to limit the scope of query to the feature descriptors associated with the object. The user can also select an object in the matching image, returned by server 140 in response to a query, to acquire other additional information.

#### **Image Feature-Based Recognition on the Mobile Device**

**[0043]** **FIG. 3** is a block diagram illustrating an exemplary system for obtaining additional information on an object of interest in an image snapshot, consistent with embodiments of the present disclosure. System 300 may be, for example, a mobile device comprising at least one processor, memory, communication hardware and software, and a camera. In some embodiments, system 300 is a client device 910 as shown in **FIG. 9**. As system 300 may be operated by a user attempting to obtain additional information on an object of interest, system 300 may also be referred to as a user's mobile device. **FIG. 3** illustrates various modules,

which may be implemented in hardware or software, that may be present on system 300, consistent with methods and systems described herein. As shown in **FIG. 3**, system 300 includes an image acquisition module 310, a feature descriptor generation module 320, a fingerprint generation module 330, and an additional data acquisition module 340.

**[0044]** Image acquisition module 310 acquires data representing a first image or "snapshot". The snapshot may be captured by, for example, a camera, or camera functionality of a smartphone or tablet. In some embodiments, a snapshot of the entire scene visible to the camera is taken and stored. In some embodiments, a user may indicate one or more portions of the scene visible on the display, and only the indicated portions are stored. Image acquisition module 310 can also acquire image data from other sources (e.g., from a local or remote data storage device). The first image data is transmitted to feature descriptor generation module 320 for further processing.

**[0045]** Feature descriptor generation module 320 can extract one or more features based on the pixel values represented by the first image data, and can generate numerical representation of the extracted features. In some embodiments, feature descriptor generation module 320 detects and locates salient points within the selected one or more portions of the first image, and generates feature descriptors to represent the salient points. Methods and systems for detecting and locating salient points, and generating feature descriptor vectors therefrom, are described above.

**[0046]** Fingerprint generation module 330 generates an image fingerprint based on the feature descriptor vectors generated by feature descriptor generation module 220. The fingerprint can include (or be generated based on) a set of feature descriptors based on salient points detected in a portion of the captured image (or a specific depicted object) selected by the user, or salient points detected in the entirety of the captured image.

**[0047]** After the fingerprint is generated, the fingerprint is used to acquire additional information by additional data acquisition module 340. In some embodiments, the fingerprint is transmitted to another system (e.g., system 700 as to be described in **FIG. 7** or server 940 of **FIG. 9**) over a network to be used in identifying additional information. The process of searching for and identifying additional information based on the fingerprint will be described in more detail below with respect to, for example, **FIG. 4A**

**[0048]** In some embodiments, the processing of the query includes determining, with a certain degree of certainty, that the feature descriptors of the query image match with the feature descriptors of a candidate image. Based on this determination, the processor of the query can then identify the media content (e.g. an image depicting a scene of a TV show, an image of a published advertisement, etc.) being depicted in the query image. Details about the processing of the query will be described later in, for example, **FIG. 6A-6E**.

**[0049]** After the querying with the fingerprint is processed, additional data acquisition module 340 can acquire additional data based on the result of the query. For example, based on the

query, a second image including the object represented by the fingerprint in the query is located, with a certain degree of certainty, and additional data acquisition module 340 can acquire the second image for displaying. The second image can include additional information associated with the object that is not depicted in the first image (e.g., a different viewpoint, a different background, additional items now depicted in the first image, etc.). Moreover, additional data related to the object can also be acquired by additional data acquisition module 340. Referring back to the illustrative example described earlier in **FIGs 1B-C**, after a query including the feature descriptors of the sofa is processed, a sofa of a specific brand and model is identified from the query, and the brand and model information can be acquired by additional data acquisition module 340 and displayed to the user. Moreover, as a result of the identification, other information, such as the price, the vendors who are selling the sofa, etc., can also be acquired by additional data acquisition module 340 and displayed to the user.

### **Image-Feature-Based Search**

**[0050]** For a given query image, Image-Feature-Based Search finds matches within a large collection of images (training set) by comparing the pre-extracted image features of the training set with those extracted from the query image. In order to keep these costly per-image comparisons at a minimum, an additional search data structure is used to preselect a coarse candidate set efficiently. This search data structure, called the image Feature Index, allows a similarity search (e.g., approximate nearest neighbor (ANN) lookup) to be performed, where a given query image feature data point can be mapped to its closest corresponding data points of a training set of images. The images associated with these data points will be retrieved as a set of candidate images. As will be described later, the Image Feature Index provides a mapping table capable of indexing / addressing feature data points. Such a mapping table can be a hash table, and the relationship between the address and the feature data points can be defined according to a hash function.

**[0051]** Reference is now made to **FIG. 4A**, which is a flow chart illustrating an exemplary method for performing image feature-based recognition from video, without the user taking a snapshot, consistent with embodiments of the present disclosure. In step 402, a stable camera image (i.e., objects appear to be stationary) is captured by the user's mobile device. In some embodiments, the mobile device may determine that an image is stable when it detects little or no movement using, for example, an accelerometer sensor of the mobile phone or other device that detects movement. After detecting little or no movement, the mobile device may capture one or more images. In some embodiments, where multiple images are captured, the mobile device may determine which of the multiple images is the most stable image, and use the most stable image as the query image.

**[0052]** In some embodiments, particularly those in which a user is taking a snapshot of video, there may be a delay between what is displayed on the screen and when the snapshot is taken after the user initiates the taking of the snapshot. Often, because time has elapsed, the captured snapshot no longer contains the items or scene of interest. In some embodiments

described herein, an image taken some amount of time (e.g. one second) prior to the snapshot being taken is used as the query image.

**[0053]** In step 404, the query image is used to find a matching image within the reference image set. Feature descriptors are extracted from the salient points detected in the query image, and a fingerprint is generated from the extracted feature descriptors. The fingerprint is then transmitted to a server, where image feature-based recognition is performed to find a set of candidate images, from which a matching image can be found.

**[0054]** If a matching image cannot be found, in step 406, meaning that no further analysis can be carried out for that particular camera image, step 402 is reexecuted to acquire a different stable image.

**[0055]** If a matching image is found in step 406, step 408 can then be executed, at the mobile device. In step 408, the pose of the query image can be estimated. The estimation of pose and fit can also be part of the determination about whether a query image and a reference image match, as to be described in **FIG. 4B**. After determining that the two images fit, the pose information may be transmitted to the mobile device in step 410. In some embodiments, the pose information may be displayed at the mobile device.

**[0056]** In step 412, the system determines if tracking is lost. Tracking can be determined based on, for example, whether a substantial change in pose is detected. In some embodiments, steps 402 through 412 can be used to first train the system to identify and track things of interest (as indicated by the user or through other means). A tracking can be lost when, for example, the person who operates the mobile device shifts the shooting direction substantially, or a new object that has not been identified before appears in the video.

**[0057]** Once the system is capable of tracking the thing of interest, step 414 can be executed to start processing live camera images, from which salient points can be detected and located in step 416, and feature descriptors are generated in step 418. The feature descriptors can then be used to estimate the pose again in step 408.

**[0058]** With such the arrangement as shown in **FIG. 4A**, content that has already be identified before may not need to be queried again. Therefore, the number of accesses to the image feature database (either located remotely or local to the device that generates the query) can be reduced. This is especially useful when a mobile device is used to generate query from a live video, when it is expected that images of the same object with identical features are captured repeatedly. Performance may be improved by not having to query the Image Feature database repeatedly for identical features. In this case, the mobile device can act as a gatekeeper to determine whether a tracking is lost (e.g., via steps 406 through 412), and access the database only when the tracking is lost (e.g., the person who operates the camcorder shifts the shooting direction substantially, or a new object that has not be identified before appears in the video).

[0059] Reference is now made to **FIG. 4B**, which illustrates an exemplary method for determining whether a candidate image matches with a query image, as described herein. In step 432, a query including an image fingerprint is received. In step 436, one or more feature descriptor vectors for the salient points of the query image are obtained from the fingerprint. In step 438, the Image Feature Index database is queried, and a set of candidate image's fingerprints are obtained based on the feature descriptors in the query. In some embodiments, the candidate images are stored in a database (e.g. annotations database 144 of **FIG. 1A**). The reference images may be stored along with corresponding fingerprints and annotations. In some embodiments, the reference images and annotations are provided by the content providers.

[0060] Steps 440-448 illustrate a loop that compares all candidates with the query image by using the direct matching comparison technique as shown in, for example, **FIG. 2B**, and can be part of step 404 of **FIG. 4A**. In step 440, a reference image is obtained from the set of candidate images, and may then be removed from the set. In step 442, feature distances between the query image and the reference image are calculated. These feature distances may be used to identify the corresponding features between the query image and the reference image. The feature distances may be expressed, for example, using Euclidean distance in the feature space. In such an example, for  $M$  features in a query image and  $N$  features in a reference image, the result may be an  $M \times N$  Matrix of Euclidean distances. Typically only a certain percentage (e.g., 20%) of the closest features is accepted as a "correspondence". In step 444, based on the identified corresponding features, a pose (e.g., orientation) of the reference image relative to the query image can be estimated, in a similar manner as described in **FIG. 2B**. The query image and the reference image are then compared in step 446, taking into account any difference in the poses of the query image content and the reference image content, to determine whether the query image matches the reference image. In some embodiments, step 446 is executed in a similar manner as described in **FIG. 2B**, where a homographic transformation is applied to the reference image, and then a determination is made whether at least part of the query image, including the consensus set of features, fit into the transformed reference image, from which it is determined whether the two images match. If it is determined that the two images do not match, step 448 will be executed to determine whether there are any images left in the set of candidate images. If there are remaining candidate images, steps 440, 442, 444, 446, and 448 will be repeated for the next reference image, until no more images are left in the candidate set. If none of the reference images in the candidate set matches with the query image, it will be concluded that there is no match, and an appropriate indication will be returned to the mobile device in step 450.

[0061] **FIG. 4C** is a flow chart illustrating exemplary method for querying a cached image set before querying the image feature index database to find matching image, consistent with embodiments of the present disclosure. In some embodiments, reference images that have been found to match a query can be deposited in a cached image set database. In some embodiments, the method of **FIG. 4C** can be implemented by replicable query caches 141 of **FIG. 1A**. A two-tier search can be performed by first searching the cached image set database.



As the cached image set database is typically much smaller than the full reference image database, the search requires less time and fewer computational resources. In some embodiments, the cached image set database can also be organized in the same way as the reference image database (e.g. employing image feature index), but on a smaller scale. In some embodiments, the cached image set database can also store a mapping between a previously-received query fingerprint and its corresponding matching reference image. The cached query can also be associated with multiple users. When multiple identical queries (e.g., queries including the same fingerprint) are received from different users, only one of the query is processed (either by performing a search at the cached image set database, or at the reference image database), and other users who submit the same query can be forwarded with the search result.

**[0062]** As shown in **FIG. 4C**, a query fingerprint is retrieved in step 432, and feature descriptors are selected from the fingerprint in step 436. Steps 432 and 436 are the same as **FIG. 4B** and their description is not repeated. The selected feature descriptors can be then used to search the cached image set database in step 463. The search can be done based on performing a search for the current query feature descriptors within the cached image set, or by comparing the current query fingerprints with previous-received query fingerprint.

**[0063]** In some embodiments, if a cached query image is found in step 464, the user (or the mobile device associated with the user) can be registered and associated with the cache hit, in step 465. As described before, associating the user with a history of cache hits can be useful in refining the cache access policy. And then step 466 can be executed, where the retrieved reference image is compared with the query image to determine if they are matching, similar to steps 440 through 448 of **FIG. 4B**. The search result can be reported in step 467, following which additional information can be provided to the user based on the search result.

**[0064]** On the other hand, if no cached query image is found in step 464, the query fingerprint will then be submitted to query the reference image database in step 468. In step 469, if a match is found, the query fingerprint can be added to the cached image set database and be associated with the matching reference image found in step 468. In some embodiments, the search result can also be broadcasted to other users, in step 470. The broadcasting can be useful in discouraging other users from submitting the same query to the system.

### **Image Feature Index**

**[0065]** A key feature of the novel image feature-based recognition methods and systems described herein is the image feature index. The methods and systems described trade in compensable loss in video frame accuracy for lossy probabilistic data management, enabling drastically lowered computational costs and storage requirements in conjunction with automatic pruning of less distinctive image content. Such methods and systems are particularly suitable for managing reference image data generated from videos.

**[0066]** To understand the new methods and systems described herein, the following is a description of prior art systems. **FIG. 5A-5H** illustrate the problems associated with known methods for using hash tables. In **FIG. 5A**, hash table 510 is used to organize feature descriptors associated with a set of training images, from which candidate images may be selected. Hash table 510 include a bucket array 512, where each bucket address is associated, via hash functions, with one or more segments of the feature space. Each bucket can then store references to features (e.g. features 514) that reside geometrically within the bucket- associated space segment. Each hash bucket can have an unlimited number of slots, and can contain an unlimited number of feature references. Hash buckets are associated with feature descriptors (and vector components thereof) based on a relationship between the feature descriptors, determined from the location of the feature vector component within the feature space, the hash address associated with the feature space segment, and the hash bucket associated with that address. In some cases, one or more hash functions are applied to the feature descriptor vector, and the output of the hash functions can be used to determine which addresses (and which hash buckets) the feature descriptor should be associated with.

**[0067]** In some cases, hash functions are used to partition a feature space 515, which is the name given to a collection of features which can be represented by a set of feature descriptors. When the feature descriptors are expressed as vectors, a feature descriptor vector will have a component in each of the dimension. The hash functions can then partition the vector space (representing the feature space) into one or more areas (e.g., feature space area 518), where each area can have a collection of feature vector components representing dataset features 514, and each area can be associated with a hash bucket address. Each feature vector is then mapped by the hash functions into the hash buckets, and vectors meeting certain criteria (e.g., representing similar features) may be mapped into the same area (and hence mapped to the same hash bucket).

**[0068]** That way, Locality sensitive hashing (LSH) can be used to perform the similarity search, in which hash table 510 is associated with a family of hash functions which can associate, with high probability, similar data points (e.g., vectors constituting similar feature descriptors) with the same hash buckets. Locality Sensitivity refers to the fact that the probability of association between data points increases steadily for increasing similarity between the data points. As shown in **FIG. 5A**, for a given query feature 519, the feature descriptor corresponding to the feature 519 can include a set of vector components 520 (denoted by "x") along each vector dimension. For each vector component 520, an approximate k-nearest neighbor search (ANN) can be performed by selecting data points that share the same hash bucket membership with the query vector component, and that are also within a similarity distance limit 521 of the query vector component. Those data points can then be ranked according to, for example, a distance between a feature descriptor vector including those data points as vector components, and the feature descriptor representing the query features. A matching determination can be made based on the ranking (e.g., shortest distance).

**[0069]** **FIG. 5B** illustrates a prior art method 530 for performing the query process to an image feature index database as described before. In step 531, feature descriptors included in a

query fingerprint are extracted and stored in a data structure (e.g., a list), for looping through the individual features. Steps 534 through 537 are executed to perform a first round of feature descriptor retrieval from the image feature index database in response to the query. As described before, a family of hash functions is used to partition the feature sample space. Therefore, each hash function in the family is retrieved in step 535 (after checking there is unused hash function left in step 534), and then the retrieve function is used to generate an address to a bucket based on a particular query feature descriptor in step 535. The reference feature descriptor data points stored in the addressed bucket is then collected in step 536. Steps 534 through 537 are repeated until all hash functions in the family has been used, as determined in step 534,

**[0070]** After the reference feature descriptor data points are collected, they will be filtered via steps 538 through 542, where a distance is computed between each collected reference feature descriptor data point and the query feature descriptor data point in step 540 and compared with a certain distance  $R$  in step 541, and only those reference feature descriptors that within distance  $R$  will be accepted in step 542. Steps 538 through 542 are repeated until all reference data points collected from steps 534 through 537 are analyzed.

**[0071]** After all the query features have been analyzed, reference images containing the accepted features will be collected as candidate images in step 543. The candidate images are then sorted according to, for example, a summed distance (from the query feature descriptors) of the all the accepted reference feature descriptors, in step 544, and the sorted list is reported in step 545. Based on the sorted list, it can then be determined which candidate image to use for further processing.

**[0072] FIG. 5C** illustrates a prior art method 550 for inserting a reference image into an image feature index database as described before. In step 551, reference features are extracted from a reference image. In some embodiments, the reference features (represented by feature descriptors) are identified directly from a fingerprint associated with the reference image. The extracted features will then be selected, one at a time, in step 553. The selected feature descriptor will then be input to a family of hash functions, where each hash function of the family will be picked, in step 555, to calculate a bucket address from the selected feature descriptor, in step 556. The reference feature descriptor will then be added to the bucket, by appending the feature descriptor data point to an entry list of the bucket, in step 557. After all the hash functions have been used to calculate the address for the selected feature descriptor, in step 554, the next extracted feature descriptor will be selected for processing in step 553, until all of the extracted feature descriptors are processed, as determined in step 552. Feature vectors will also be stored as hash keys associated with the reference image in step 558.

**[0073]** One potential drawback with the method described in **FIGs 5A-5B** is that the selected bucket may not provide good coverage of feature data points that similar to the query data points. For example, as shown in **FIG. 5D**, query feature vector component 570 is mapped to feature space area 572. Within the similarity distance limit 571, only area portion 573, which is within the same feature space area as query feature vector component 570, will be searched.

Data point 574, being also within area portion 573, is found to be the apparent nearest neighbor to query feature vector component 570. However, data point 575 is the one that has the shortest distance from query feature vector component 570, but is not selected because it belongs to a different feature space area. Data point 575 will be found if the search extends to adjacent buckets (and adjacent feature space area), but such an arrangement can be very computation intensive since the number of adjacent buckets grows exponentially for increasing dimensionality. Therefore, other than extending the search to adjacent buckets, it is desirable to further improve the likelihood that similar data points are associated with the same bucket, thus improving accuracy of the search

**[0074]** One known technique to further improve the likelihood that similar data points are associated with the same bucket is, as shown in **FIG. 5E**, to use multiple hash tables 530 associated with different hash functions. In such a method, multiple hash functions (e.g., hash function families 582 and 584) can partition the same feature spaces in different yet overlapping manners (as denoted by the solid-line grid and dotted-line grid), such that the feature descriptors included in a query can lead to accessing of multiple buckets simultaneously covering a set of overlapping feature space areas simultaneously in different tables, and the search can be confined to the set of areas. Although the overlapping buckets allow better coverage of a partitioned feature space, it is possible that reference features that are not closest to the query features can be included in a bucket addressed by the query features, and additional filtering may need to be performed (e.g., by computing a distance between the reference features in the bucket and the query feature) to determine whether the reference features are to be included in the candidate image set.

**[0075]** Another **known** technique for improving LSH, as shown in **FIG. 5F**, is using an entropy-based LSH to generate random data points near the query data point. These random data points can also cause the hash functions to map feature space areas 590, 592, and 594 into the same hash bucket. The similarity distance limit 596 for ANN search can also be expanded (relative to similarity distance limit 521 of **FIG. 5A**) to include the random data points. Moreover, as shown in **FIG. 5G**, multi-probe LSH can derive additional query points by employing a query derived probing sequence of allegedly neighboring buckets.

**[0076]** While the aforementioned LSH techniques can improve bucket coverage effect inherent to parameter space partitioning, there can be huge computational cost and storage requirement when the candidate images, whose feature descriptors are to be extracted and organized under hash table 510, are generated from a video. Because of the huge volume of data generated from a video, it can be very time-consuming and computationally intensive to examine each data point stored in any given bucket while performing the ANN search, which involves calculation of the actual distance between each data point in a hash bucket and the query data point. Moreover, as shown in **FIG. 5H**, video typically includes a large amount of repeated image content and, as a result, the vast majority of extracted image features may not contribute to any distinctive information, or can cause a minority of distinctive image features to be lost in the majority of less distinctive features, diluting any image match probability ranking that depends on feature presence. Besides, over-occupied hash buckets can also contribute

disproportionally many data points to candidate feature descriptor sets and thwart performance gains for certain recurring image features, while consistent feature motions and transforms can diminish the ability of homographic alignment testing of feature correspondences to reliably identify matches.

### Enhanced LSH

**[0077]** Methods and systems describing a modified technique of LSH will be provided in the following. The modified technique can be used to exploit the statistical coverage properties of the superposition of multiple well-chosen non-congruent hash functions as a coarse replacement distance metric. These methods and systems require fewer computations and less storage.

**[0078]** The described method eliminates the need of performing distance computations using the vector components of hash table stored data-points as in the aforementioned prior art implementations of LSH based ANN. The positional information hidden in multiple super-imposed non-congruent hash functions can be tapped directly, by interpreting the overlapping hash buckets as an irregular super-sampling of the continuous parameter space. Given the property of locality sensitive hashing, the probability of a hash bucket collision between samples  $p_i$  and query sample  $q$  increases the closer they are. Interpreted reversely for a given set of  $L$  different locality sensitive hash functions, this implies that the  $D$  dimensional feature space area coverage of overlapping hash buckets within a similarity distance limit around a query point  $q$  increases for smaller radii ( $R_1 < R_2$ ):

$$\frac{1}{R_1^D} \cdot \sum_{i=1}^L \int \int \dots \int_{|p-q| < R_1} \delta(h_i(p), h_i(q)) \geq \frac{1}{R_2^D} \cdot \sum_{i=1}^L \int \int \dots \int_{|p-q| < R_2} \delta(h_i(p), h_i(q))$$

**[0079]** In other words, for a query point  $q$  the number of simultaneously overlapping hash buckets addressed by  $h(q)$  relates coarsely with the distance to the query point. This observation can be used directly as a quantized distance ranking metric:

$$R(p, q) = \sum_{i=1}^L \delta(h_i(q), h_i(p)) \sim \|p - q\|$$

**[0080]** The closer a sample point  $p_i$  comes to  $q$ , the higher is the probability that it's covered by some of the overlapping hash buckets selected by  $q$  and as a consequence the more frequently it shows up as a duplicate when collecting all points attached to the buckets. In a way, all spatial data point locations can be seen as lossy encoded or super-sampled within the bucket membership pattern of the hash tables itself. This encoding serves as a replacement for the data point locations.

**[0081]** The quality and resolution of this encoding is defined by the hash functions. First, the density of the overlapping buckets around a query point converges to a normal distribution (i.e., the more non-congruent hash functions are used, the finer the resolution gets.):

$$\lim_{L \rightarrow \infty} \frac{1}{L} \sum_{i=1}^L \delta(h_i(\vec{q} + \vec{d} \cdot x), h_i(\vec{q})) = e^{-0.5 \frac{x^2}{\sigma_d^2}} \quad \vec{d} \in R^D \wedge \sigma \sim |\vec{d}|$$

**[0082]** Second, the absolute distance rank increases between closer samples and the rest (including noise). Only the data points  $p$  that lie within a distance corresponding to the largest bucket size to a query point  $q$  is subject to the aforementioned relationship between bucket coverage and distance:

$$\sum_{i=1}^L \delta(h_i(p), h_i(q)) * \sum_{i=1}^L \delta(h_i(r), h_i(q)) = \sum_{i=1}^L P(h_i(p) = h_i(q)) * L * P_{\text{collision}}$$

for data points  $p, r : |p - q| < R < |r - q|$

**[0083]** Those samples lying outside that large bucket size distance do not share the same bucket covered area, but may share buckets due to hash address collisions / bucket aliasing. For given ideal hash functions, these collisions are statistically distributed uniformly over the parameter space, and all samples that are not subject to the distance-rank-relation have the same fixed chance  $P_{\text{collision}}$  to show up as false positives in the candidate set. The signal-to-noise gap can be adjusted either by increasing the number of hash functions, or by increasing the table size, effectively reducing the probability  $P_{\text{collision}}$ .

**[0084]** The effect of increasing the number of hash functions is illustrated in **FIG. 6A**. On the left, query feature data point 693 is addressed to a bucket space 694. A number of hash functions are used to partition bucket space 694, where each of hash-function addressed space (e.g. space 695) can overlap with data point 693. Data points within portion of space 695 that does not overlap with 694 (e.g. space 696) may still be collected as candidate data points (due to its association with a hash function whose addressed space 695 overlaps with the query data point). The density of overlapping buckets around the query point can be represented by a distribution 690. As shown on the right, as the number of hash functions increases, distribution 690 converges to become a Gaussian distribution. Due to the nature of a hash function, that is mapping an infinite parameter space to a finite set of buckets, data-points that lie outside all segments 694, 695, 696, addressed by query data point 693, may still fall into the same hash bucket. Assuming ideal hash functions, these so called bucket collisions appear as uniform noise, since they are not subject to the locality sensitivity property.

**[0085]** **FIG. 6B-C** shows exemplary methods 602, 604, and 606 for organizing the search data structure that further exploits the probabilistic nature of the aforementioned distance rank metric. In some embodiments, as shown in **FIG. 6B**, multiple concurrent hash function address spaces can be superimposed in one unified hash table 607, where the hash indexes from the hash families are mapped to one set of hash buckets, allowing multiple hash functions to address the same hash bucket, with each hash bucket storing a list of elements. Moreover,

hash tables 608 and 609 of **FIG. 6C** illustrate a novel-storage approach of using a modified open-addressing hash table layout, where a bucket has always either zero or one elements. To resolve hash collisions (i.e., multiple hash functions pointing to the same bucket), an open-addressing method can be employed. As shown by method 604 of **FIG. 6C**, when a feature data point is to be written to a bucket addressed at 3, which already stores data. The hash table can be probed for empty bucket according to a linear probing sequence. In this illustrative example, the hash table will be traversed in a linear fashion (e.g., traversing buckets that are consecutively addressed). When a hash table address is calculated for a query feature data point, all elements starting from the calculated hash table address until an empty bucket is reached will be included in the candidate data points. Candidate feature data points collected along the probe sequence can be filtered using distance comparison, where only those feature data points that are within a predefined similarity distance to the query feature data points are retained.

**[0086]** In some embodiments, as shown in method 606 of **FIG. 6C**, instead of linear probing, a probabilistic local probing can be used for resolving hash collision by randomly scattering inserted data points in buckets within a predefined range around the bucket originally addressed by the data points, and also no probing sequence is tracked. Moreover, no filtering using distance comparison is performed, and the feature's vector components, which are used for hash address calculation, do not need to be stored.

**[0087]** Since the aforementioned probabilistic distance metric uses occurrence of feature data points as distance indicator, candidate data points that are collected within a bucket but originate from different hash addresses can appear as noise that is statistically distributed over the feature space, when later retrieved as a result of query. As described below, these data points can be differentiated from the correctly collected data points based on a number of occurrences of these data points within a block of bucket entries. For example, images that have been referenced equal or less than the hash table intrinsic noise-level may be discarded (see, for example, step 616 of **FIG. 6D**). The noise can be mitigated (precisely steered) by, for example, increasing the size of hash tables, increasing the number of hash functions, etc,

**[0088]** Moreover, as shown in **FIG. 6C**, under method 606, an entry can be overwritten by more recent data. For example, bucket addressed at 89 used to store data points 620, but as a result of probabilistic local probing, it is overwritten by data points 630. Since typically data points that are associated with over-occupied buckets come from voluminous yet less distinctive features, the overwriting of these data allow these features to be blurred out, and their influences to determining the matching features can be reduced. This overwriting policy can also be adjusted to handle the overflowing situation (i.e., when all the entries in the hash table are stored with data).

**[0089]** With a hash table arranged according to, for example, hash tables 607-609 of **FIG. 6B-C**, adjacent areas in the hash table are roughly the equivalent of a bucket list. Accordingly, for each hash address calculated from a feature data point, a fixed-size block read of the hash table around the hash address can be performed using, for example, paging functionality

commonly employed in memory-mapped file accessing. Each block being read in this manner can represent a bucket list, which in turn represents an overlapped bucket oversampling feature space area partitioned by the hash function. As explained above, the number of simultaneously overlapping hash buckets addressed by a query data point (i.e., the hash buckets that are associated address calculated by applying the hash functions to the query data point) relates coarsely with the distance of the query data point, such that the closer a particular data point in the bucket comes to the query data point, the higher is the probability of that particular data point being covered by at least some of the overlapping hash buckets addressed by the query data point. As a result, that particular data point can show up more frequently as a duplicate within the overlapping hash buckets (i.e., hash buckets mapped to overlapped feature space area), and the number of occurrence of that particular data point within the overlapping hash buckets (represented by the block read) can be used to estimate its distance from the query data point. Those data points that occur most frequently within the block can then be determined to be the matching features in response to the query, and included in the candidate data points. From the candidate data points, the images associated with the candidate data points can be acquired as candidate images.

**[0090] FIG. 6D** illustrates an exemplary method for performing the query process to an Image Feature Index database, with the database organized using, for example, hash table 609 of **FIG. 6C**. Steps 532-536 are the same as those in **FIG. 5B** and their descriptions are not repeated here. After the bucket address is determined in step 536, a block read of the hash table around that bucket, as described above with reference to **FIG. 6C**, is performed in step 612. Each duplicate reference feature data point collected is then counted by, for example, keeping track of a reference counter associated with each reference feature data point in step 613. After all the query data points are analyzed, the candidate images associated with the collected reference feature data points are selected as matching candidates in step 614. Each candidate image can be associated with a combination of reference counters associated with each duplicate reference feature data points as determined in step 613. The combination of the reference counters can then be used to rank and sort the candidate images in step 615. In step 616, candidate images with some reference counters that are above a certain threshold can be discarded. Such a threshold can be determined based on a predetermined noise level.

**[0091] FIG. 6E** illustrates a method for inserting a reference image into an image feature index database as described before. Steps 551-556 are the same as similarly-numbered steps of **FIG. 5C**, and their description is not repeated here. In step 621, after a first bucket address is determined for the reference feature quantity, a random function is used to calculate a second bucket address. The random function can be used to randomly scatter inserted data points in buckets within a predefined range around the bucket originally addressed by the data points, similar to method 602 of **FIG. 6B**. The reference data point will then be stored at a bucket addressed by the second address, in step 622. If there are other older data being stored in that bucket, that data will be discarded in favor of the reference data point, in step 623.

**[0092] FIG. 7** is a block diagram illustrating an exemplary system 700 with which embodiments described herein can be implemented, consistent with embodiments of the present disclosure.



In some embodiments, system 700 can be part of server 140 of **FIG. 1A**. In some embodiments, system 700 can be used to process a query sent by, for example, system 300 of **FIG. 3**. In at least one exemplary embodiment, system 700 comprises image matching module 752, reference images database 720, hash table 730, hash table management module 740, candidate feature descriptors retrieval module 746, candidate image retrieval module 750, and hash functions 770.

**[0093]** In at least some exemplary embodiments, image matching module 752 is image matching module 142 as described above with respect to **FIG. 1A**.

**[0094]** Reference images database 720 can store a set of candidate images. Each candidate image can be an image of visual media (e.g., a screenshot of a scene during a TV show), and can include a set of pre-determined visual features. These visual features can be pre-determined to be highly likely to be included in a query to be processed by system 700. These features can be represented by the aforementioned feature descriptors. Each reference image can then be associated with one or more feature descriptors corresponding to at least some of the set of pre-determined visual features included in the image. When a query including a set of feature descriptors is received, one or more candidate images can be retrieved from reference images database 720 in response to a query based on a determination that the feature descriptors included the retrieved candidate images match with at least some of feature descriptors included in the fingerprint of the query, with a certain degree of certainty. In some embodiments, those retrieved candidate images can then be compared with the query image (i.e., the image from which the query is generated) to find the best matching candidate image.

**[0095]** Although a database of reference images is presented here, it should be understood that there are other ways of grouping visual features in a database (e.g., associating a combination of visual features or feature descriptors with a particular object, instead of a candidate image), which are fully applicable to the techniques disclosed herein and are within the scope of the present disclosure.

**[0096]** Hash table 730 is associated with a set of hash functions 770. In some embodiments, hash table management module 740 further includes a collision resolving module 742 and an overflow handling module 744.

**[0097]** In some embodiments, hash table management module 740, candidate feature descriptors retrieval module 746, and candidate images retrieval module 750 can be configured to implement the processing of query as disclosed in **FIG. -6D-E**. Hash table management module 740 provides access to hash table 730 in response to a query. Hash table management module 340 provides a mapping table that maps the hash indexes (i.e. the addresses) of all the hash tables included in hash table 730 to one set of hash buckets, similar to unified hash tables 602,604 and 606 of **FIG. 6B-C**. The addresses are calculated based on the feature descriptor data points using hash functions 770.

**[0098]** Collision resolving module 742, which is part of hash table management module 740, handles hash collision (i.e., when two different data points are addressed to the same bucket) when associating a new data point (e.g., a set of feature descriptors) with hash table 730. In some embodiments, collision resolving module 742 implements the open-addressing and probing methods as described before in reference to **FIGs. 6C** and **6E**.

**[0099]** Overflow handling module 744, which is part of hash table management module 740, handles the association of a new data point (e.g., a set of feature descriptors) with hash table 730 in the case that no empty slot can be found in the hash table. In some embodiments, overflow handling module 744 implements the overwriting policy as described before in reference to **FIGs. 6C** and **6E**.

**[0100]** Candidate feature descriptors retrieval module 746 accesses hash table 730 to acquire a set of candidate feature descriptors data points. For each feature descriptor included in a query, candidate feature descriptors retrieval module 746 applies the hash functions to calculate the addresses, uses the addresses to locate the buckets, and then retrieves all the data points associated with the buckets. In the case where a probing sequence is not tracked, all the data points associated with other buckets along the probing sequence will be retrieved as well. In the case where probabilistic local probing is used, all the data points within the predefined range around the bucket addressed will also be retrieved. After the candidate feature descriptors data points are retrieved, the data points are then sent to candidate images retrieval module 750, which can then determine which set of the feature descriptors data points repeats the most frequently within the set of candidate data points. These feature descriptors will be determined to be the most similar to the feature descriptors included in the query, and can then be used to identify a set of candidate images or other additional information in response to the query. In some embodiments, candidate feature descriptors retrieval module 746 and candidate images retrieval module 750 implement at least part of methods of **FIGs. 6D** and **6E**.

**[0101]** After receiving the candidate image, image matching module 752 then compares the fingerprints of the candidate images to the query fingerprint (or portion thereof), and determines if there is a match. If there is a match, it then system 700 retrieves the matching image along with annotations attached to it from the annotations database 780, and returns the additional information and annotations to the mobile device.

**[0102]** In some embodiments, as shown in **FIG. 8**, an exemplary client device 800 can include, in addition to system 300 of **FIG. 3**, a query cache 840 and query cache management module 850, to form a two-tier system for query processing. Query cache 840 can store a mapping between previously-received queries and the feature descriptors (and/or candidate image(s)) retrieved from server 140 in response to those queries, employ a method similar to as described in **FIG. 4C**. When a new query is received, query cache management module 850 can compare the content of the new query with the content of the queries stored in query cache 800. If a match is found in query cache 840, the associated feature descriptors (and/or candidate image(s)) can be provided as a response without sending the query to server 130. If

a match cannot be found in query cache 840 and/or the candidate images database 820, the query will then be transmitted to system 700.

**[0103]** In some embodiments, such a two-tier system can also be implemented between two different servers, or as part of system 700, to handle queries originated from different client devices but include similar content. After a first client device submits a first query which causes server 130 to retrieve a matching image, another client device may submit a second query which includes similar content. In that case, the second query can be handled by searching through query cache 840, and the result to the first query can be reused. Such an arrangement can be useful to handle spikes in queries with similar content. For example, as described before, when a video is being broadcasted to millions of viewers, many of them may submit a query by taking a snapshot of an item of interest that appears simultaneously on their display devices, and these queries can include very similar content. After the first query is processed by accessing the much larger reference images database 720 and the result is added to query cache 840, the other queries including the same item of interest can then be handled by accessing the query cache 840 instead, which can improve efficiency enormously.

**[0104] FIG. 9** is a block diagram of an exemplary system 900 with which embodiments described herein can be implemented, consistent with embodiments of the present disclosure. As shown in **FIG. 9**, system 900 includes a client device 910, a network 930, and a server 940. Client device 910 includes one or more processors 912, a memory device 914, a storage device 916, a display 917, a network interface 918, a camera 119 (or other image generation device), and an accelerometer 922 (or other orientation determination device), all of which can communicate with each other via a bus 920. In some embodiments, display 917 is preferably a touchscreen. The I/O devices can include a microphone and any other devices that can acquire and/or output a signal. Through network 930, client device 910 can exchange data with a server 940. Server 940 also includes one or more processors 942, a memory device 944, a storage device 946, and a network interface 948, all of which can communicate with each other via a bus 950.

**[0105]** Both memories 914 and 944 can be a random access memory (RAM) or other volatile storage devices for storing information and instructions to be executed by, respectively, processors 912 and 942. Memories 914 and 944 also can be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processors 912 and 942. Such instructions, after being stored in non-transitory storage media accessible to processors 912 and 914 (e.g., storage devices 916 and 946), render computer systems 910 and 940 into special-purpose machines that are customized to perform the operations specified in the instructions. The instructions can be organized into different software modules, which can include, by way of example, components, such as software components, object-oriented software components, class components and task components, processes, functions, fields, procedures, subroutines, segments of program code, drivers, firmware, microcode, circuitry, data, databases, data structures, tables, arrays, and variables.

**[0106]** In general, the word "module," as used herein, refers to logic embodied in hardware or

firmware, or to a collection of software instructions, possibly having entry and exit points, written in a programming language, such as, for example, Java, Lua, C or C++. A software module can be compiled and linked into an executable program, installed in a dynamic link library, or written in an interpreted programming language such as, for example, BASIC, Perl, or Python. It will be appreciated that software modules can be callable from other modules or from themselves, and/or can be invoked in response to detected events or interrupts. Software modules configured for execution on computing devices can be provided on a computer readable medium, such as a compact disc, digital video disc, flash drive, magnetic disc, or any other tangible medium, or as a digital download (and can be originally stored in a compressed or installable format that requires installation, decompression, or decryption prior to execution). Such software code can be stored, partially or fully, on a memory device of the executing computing device, for execution by the computing device. Software instructions can be embedded in firmware, such as an EPROM. It will be further appreciated that hardware modules can be comprised of connected logic units, such as gates and flip-flops, and/or can be comprised of programmable units, such as programmable gate arrays or processors. The modules or computing device functionality described herein are preferably implemented as software modules, but can be represented in hardware or firmware. Generally, the modules described herein refer to logical modules that can be combined with other modules or divided into sub-modules despite their physical organization or storage.

**[0107]** Client device 910 and server 940 can implement the techniques described herein using customized hard-wired logic, one or more ASICs or FPGAs, firmware and/or program logic which in combination with the computer system causes or programs client device 910 and server 940 to be a special-purpose machine. According to some embodiments, the operations, functionalities, and techniques and other features described herein are performed by client device 940 and server 940 in response to processors 912 and 942 executing one or more sequences of one or more instructions contained in, respectively, memories 914 and 944. Such instructions can be read into memories 914 and 944 from another storage medium, such as storage devices 916 and 946. Execution of the sequences of instructions contained in memories 914 and 944 cause respectively processors 912 and 942 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry can be used in place of or in combination with software instructions.

**[0108]** The term "non-transitory media" as used herein refers to any non-transitory media for storing data and/or instructions that cause a machine to operate in a specific fashion. Such non-transitory media can comprise non-volatile media and/or volatile media. Non-volatile media can include, for example, optical or magnetic devices, such as storage devices 916 and 946. Volatile media can include dynamic memory, such as memories 914 and 944. Common forms of non-transitory media include, for example, a floppy disk, a flexible disk, hard disk, solid state drive, magnetic tape, or any other magnetic data storage medium, a CD-ROM, any other optical data storage medium, any physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, NVRAM, any other memory chip or cartridge, and networked versions of the same.

**[0109]** For example, devices 121 and 122 of **FIG. 1A** can be implemented as client device 910, where system 300 of **FIG. 3** and methods 400, 430, and 460 of **FIGs. 4A-4B** can be implemented as software instructions stored in storage 916 and memory 914. In some embodiments, server 140 of **FIG. 1A** can be implemented as server 940, where system 700 of **FIG. 7** and methods 600, 602, 610 and 620 of **FIGs. 6B-6D** can be implemented as software instructions stored in storage 946 and memory 944.

**[0110]** Network interfaces 918 and 948 can provide a two-way data communication coupling to network 930. For example, network interfaces 918 and 948 can be an integrated services digital network (ISDN) card, cable modem, satellite modem, or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, network interfaces 918 and 948 can be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links can also be implemented. In any such implementation, network interfaces 918 and 948 can send and receive electrical, electromagnetic or optical signals that carry digital data streams representing various types of information, and provide the data stream to storage devices 916 and 946. Processors 912 and 942 can then convert the data into a different form (e.g., by executing software instructions to compress or decompress the data), and then store the converted data into the storage devices (e.g., storage devices 916 and 946), and/or transmit the converted data via network interfaces 918 and 948 over network 930.

**[0111]** According to some embodiments, the operations, techniques, and/or components described herein can be implemented by an electronic device, which can include one or more special-purpose computing devices. The special-purpose computing devices can be hard-wired to perform the operations, techniques, and/or components described herein, or can include digital electronic devices such as one or more application-specific integrated circuits (ASICs) or field programmable gate arrays (FPGAs) that are persistently programmed to perform the operations, techniques and/or components described herein, or can include one or more hardware processors programmed to perform such features of the present disclosure pursuant to program instructions in firmware, memory, other storage, or a combination. Such special-purpose computing devices can also combine custom hard-wired logic, ASICs, or FPGAs with custom programming to accomplish the technique and other features of the present disclosure. The special-purpose computing devices can be desktop computer systems, portable computer systems, handheld devices, networking devices, or any other device that incorporates hard-wired and/or program logic to implement the techniques and other features of the present disclosure.

**[0112]** The one or more special-purpose computing devices can be generally controlled and coordinated by operating system software, such as iOS, Android, Blackberry, Chrome OS, Windows XP, Windows Vista, Windows 7, Windows 8, Windows Server, Windows CE, Unix, Linux, SunOS, Solaris, VxWorks, or other compatible operating systems. In other embodiments, the computing device can be controlled by a proprietary operating system. Operating systems control and schedule computer processes for execution, perform memory management, provide file system, networking, I/O services, and provide a user interface

functionality, such as a graphical user interface ("GUI"), among other things.

**[0113]** The aforementioned systems described herein have many uses and may be used in a variety of scenarios. For example, the methods and systems described herein may be used in systems for delivering advertising to a user of a mobile device. For example, a user can identify an item, topic, scenery, or location of interest while visiting a place, reading a publication, or watching a movie. Using the inventive systems and methods herein, the user can take a snapshot of the item, the scenery, or anything of interest, and generate an image. The image can be captured including surrounding area and at any view angle. The system then extracts features from the image, generates a query including feature descriptors corresponding to at least some of the images, and submits the query for a search based on the extracted features to a remote server. The additional information that may be displayed to the user can be provided for free to the user, or some or all of the information may be in the form of paid advertising. In some scenarios, one entity may provide the search service and charge vendors for sending the additional information about products or services that are sent to users. In some scenarios, mobile device users may be charged for the ability to retrieve information on products or services using the aforementioned methods and systems.

**[0114]** In the foregoing specification, embodiments have been described with reference to numerous specific details that can vary from implementation to implementation. Certain adaptations and modifications of the described embodiments can be made. Other embodiments can be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and examples be considered as exemplary only. It is also intended that the sequence of steps shown in figures are only for illustrative purposes and are not intended to be limited to any particular sequence of steps. As such, those skilled in the art can appreciate that these steps can be performed in a different order while implementing the same method.

## REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

### Patent documents cited in the description

- [US61937859A](#) **[0001]**
- [US8793274B](#) **[0008]**

**Non-patent literature cited in the description**

- **XIN XIN et al.** Large Visual Repository Search with Hash Collision Design Optimization, 2013, [\[0011\]](#)
- **MANI MALEK ESMAEILE et al.** A Fast Approximate Nearest Neighbor Search Algorithm in the Hamming Space, 2012, [\[0011\]](#)

**Patentkrav**

1. Server til identificering og transmission til en mobil enhed (121, 122) af supplerende informationer vedrørende et forespørgselsbillede opnået af en bruger med mobilenheden, hvilken server omfatter:

- 5            en referencebilleddatabase (720), hvori der er lagret en eller flere deskriptorer af referencekarakteristika, der er forbundet med ét eller flere referencebilleder, i en hash-tabel, hvor hash-tabellen omfatter en flerhed af kurve, hvor hver kurv er forbundet med en adresse og hver kurv omfatter enten intet eller ét element, hvor lagring af én eller flere deskriptorer af
- 10           referencekarakteristika omfatter:
- at beregne, for hver af den ene eller flere deskriptorer af referencekarakteristika, en flerhed af første hash-adresser ved anvendelse af en flerhed af hash-funktioner;
- at anvende en vilkårlig funktion til at beregne, for hver af flerheden af første
- 15           hash-adresser, en anden hash-adresse inden for et forudbestemt interval omkring en tilsvarende første hash-adresse;
- at lagre den tilsvarende deskriptor af referencekarakteristika i kurve, der er forbundet med flerheden af anden hash-adresser, og udskifte det ene element i kurven med deskriptoren af referencekarakteristika, når den tilsvarende kurv
- 20           er tom; og
- en processor til udførelse af moduler, der omfatter:
- et forespørgselsmodtagelsesmodul til modtagelse, fra mobilenheden (121, 122), af en første forespørgsel indbefattende én eller flere deskriptorer af forespurgte billedkarakteristika;
- 25           et modul til hentning af deskriptorer af kandidatkarakteristika (746) til:
- beregning, for hver af den ene eller flere deskriptorer af forespurgte billedkarakteristika, der er forbundet med forespørgselsbilledet, af en flerhed af tredje hash-adresser ved anvendelse af flerheden af hash-funktioner, og
- beregning af et antal forekomster af hver deskriptor af
- 30           referencebilledkarakteristika, der er lagret i referencebilleddatabasen (720) inden for et andet foruddefineret adresseinterval om hver af flerheden af tredje hash-adresser;



et modul til hentning af kandidatbilleder (750) til:

udvælgelse af et sæt af deskriptorer af referencebilledkarakteristika, der var beregnet til at forekomme hyppigst;

et billedmatchningsmodul (142; 710) til:

- 5 sammenligning af det udvalgte sæt af deskriptorer af referencebilledkarakteristika med den ene eller flere deskriptorer af forespurgte billedkarakteristika for at bestemme, om noget referencebillede matcher forespørgselsbilledet;
- 10 hvilken server endvidere er konfigureret til at udføre trinnene, hvis et matchende referencebillede er bestemt, med opnåelse af supplerende informationer, der er forbundet med det mindst en matchende referencebillede, fra en annotationsdatabase (144; 780), og transmission af de supplerende informationer til mobilenheden for visning som supplerende informationer vedrørende det mindst ene element af interesse for
- 15 brugeren.

**2.** Computerimplementeret fremgangsmåde til identificering og visning af supplerende informationer vedrørende et første forespørgselsbillede opnået af en

- 20 bruger med en mobil enhed (121, 122), hvilken fremgangsmåde omfatter:
- lagring, i en referencebilleddatabase (720), af én eller flere deskriptorer af referencekarakteristika i en hash-tabel, hvor hash-tabellen omfatter en flerhed af kurve, hvor hver kurv er forbundet med en adresse og hver kurv omfatter enten intet eller ét element, hvor lagring af én eller flere deskriptorer af
- 25 referencekarakteristika omfatter:
- beregning, for hver af den ene eller flere deskriptorer af referencekarakteristika, af en flerhed af første hash-adresser ved anvendelse af en flerhed af hash-funktioner;
- anvendelse af en vilkårlig funktion til at beregne, for hver af flerheden af første
- 30 hash-adresser, en anden hash-adresse inden for et forudbestemt interval omkring en tilsvarende første hash-adresse;
- lagring af den tilsvarende deskriptor af referencekarakteristika i kurve, der er forbundet med flerheden af anden hash-adresser, og udskiftning af det ene

element i kurven med deskriptoren af referencekarakteristika, når den tilsvarende kurv ikke er tom;

modtagelse, fra mobilenheden, af en første forespørgsel indbefattende én eller flere deskriptorer af forespurgte billedkarakteristika svarende til mindst ét element af interesse for brugeren, hvilket element er indbefattet i det første forespørgselsbillede;

som reaktion på modtagelse af den første forespørgsel, udførelse af en lighedssøgning, hvor lighedssøgningen omfatter:

for hver af deskriptorerne af forespurgte billedkarakteristika, der er forbundet med forespørgselsbilledet, beregning af en flerhed af tredje hash-adresser ved anvendelse af flerheden af hash-funktioner;

beregning af et antal forekomster af hver deskriptor af referencebilledkarakteristika, der er lagret i referencebilleddatabasen (720) inden for et andet foruddefineret adresseinterval om hver af flerheden af tredje hash-adresser; og

bestemmelse af ét eller flere kandidatbilleder baseret på deskriptorerne af referencebilledkarakteristika, der var beregnet til at forekomme hyppigst;

sammenligning af deskriptorerne af karakteristika for det ene eller flere kandidatbilleder med deskriptorerne af forespurgte billedkarakteristika for at bestemme, om der er et match; og,

hvis der er et match, hentning af supplerende informationer, der er forbundet med matchet; og

transmission, til mobilenheden, af de supplerende informationer, der er forbundet med matchet, for visning til brugeren.

### 3. Computerimplementeret fremgangsmåde ifølge krav 2,

hvor bestemmelsen af ét eller flere kandidatbilleder omfatter:

bestemmelse af, at antallet af forekomster af en deskriptor af referencebilledkarakteristika overstiger en forudbestemt tærskel; og

hentning af ét eller flere kandidatbilleder, der er forbundet med denne deskriptor af referencebilledkarakteristika.

4. Computerimplementeret fremgangsmåde ifølge krav 3, der endvidere omfatter:  
modtagelse, fra mobilenheden (121, 122), af en anden forespørgsel  
indbefattende én eller flere anden deskriptorer af forespurgt  
billedkarakteristika;
- 5 sammenligning af den ene eller flere første deskriptorer af forespurgt  
billedkarakteristika med den ene eller flere anden deskriptorer af forespurgt  
billedkarakteristika; og  
baseret på et resultat af sammenligningen:  
bestemmelse om ikke at udføre lighedssøgningen; og
- 10 hentning, baseret på en kortlægning mellem den første forespørgsel og det  
ene eller flere kandidatbilleder, der er lagret i en forespørgselscache (141), af  
det ene eller flere kandidatbilleder.
- 15 5. Computer-læsbart medium, der omfatter instruktioner, der, når de udføres af en  
computer, bevirker, at computeren udfører fremgangsmåden ifølge et hvilket som  
helst af krav 2 til 4.

## DRAWINGS

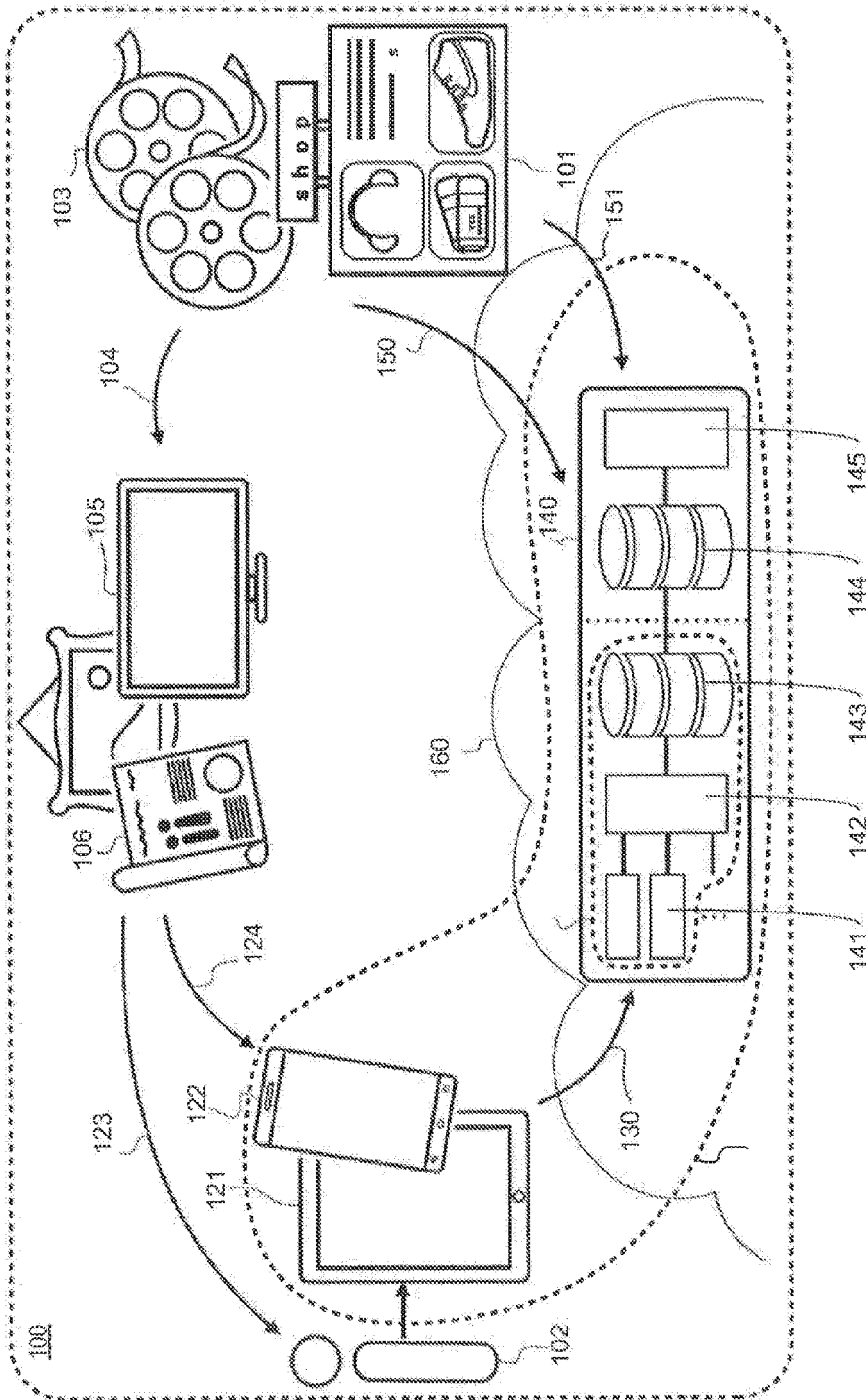


FIG. 1A

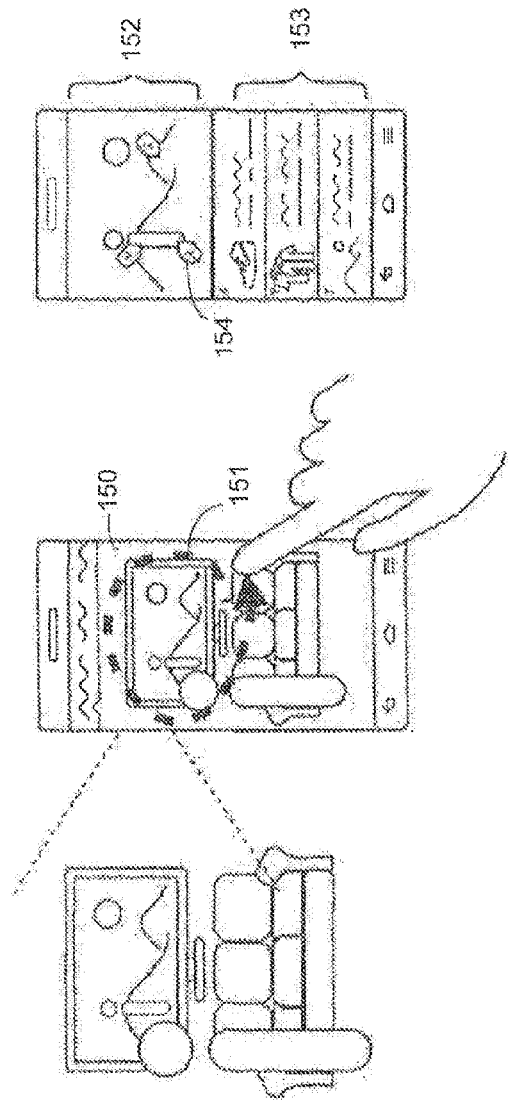


FIG. 1B

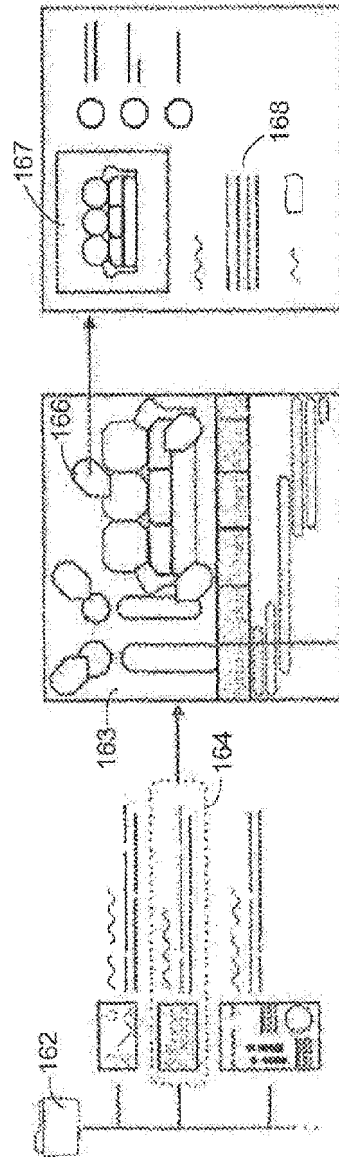


FIG. 1C

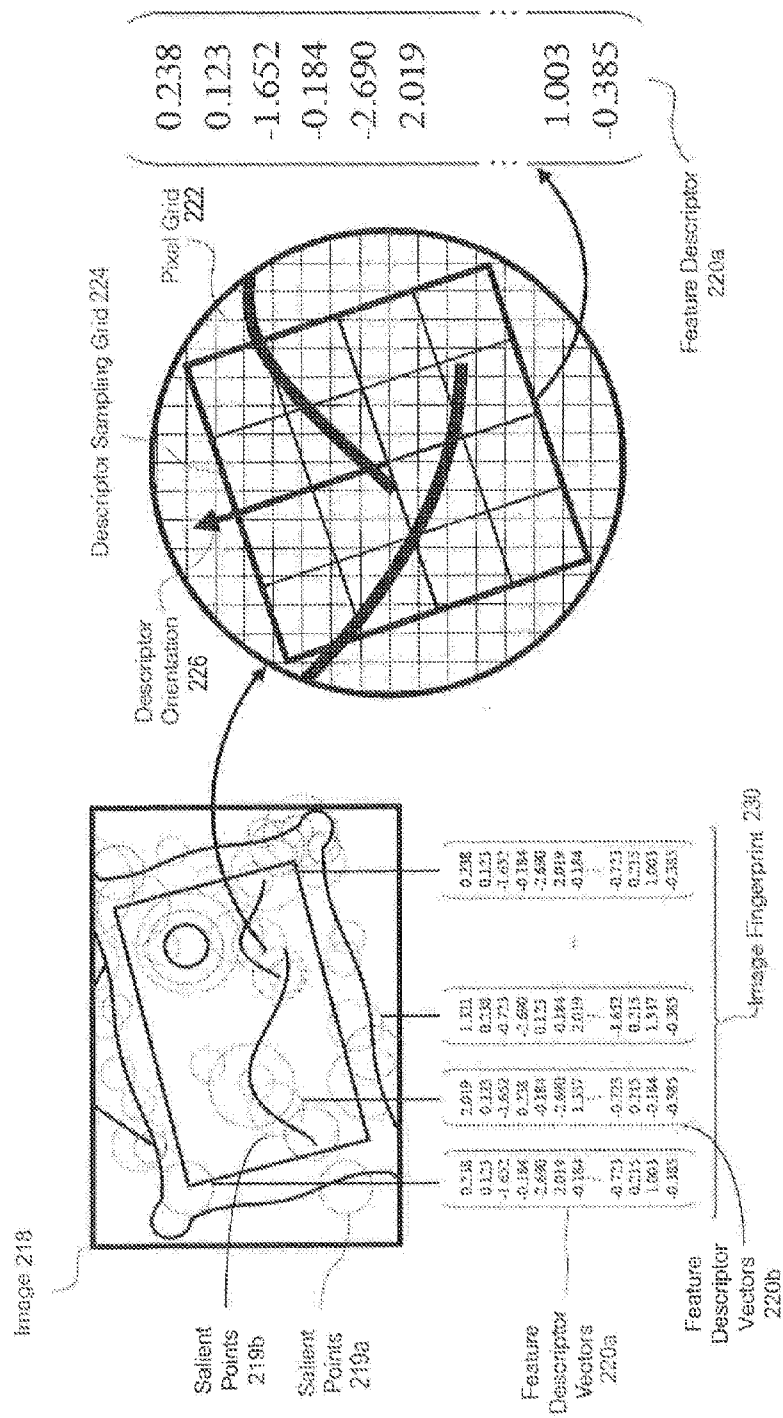
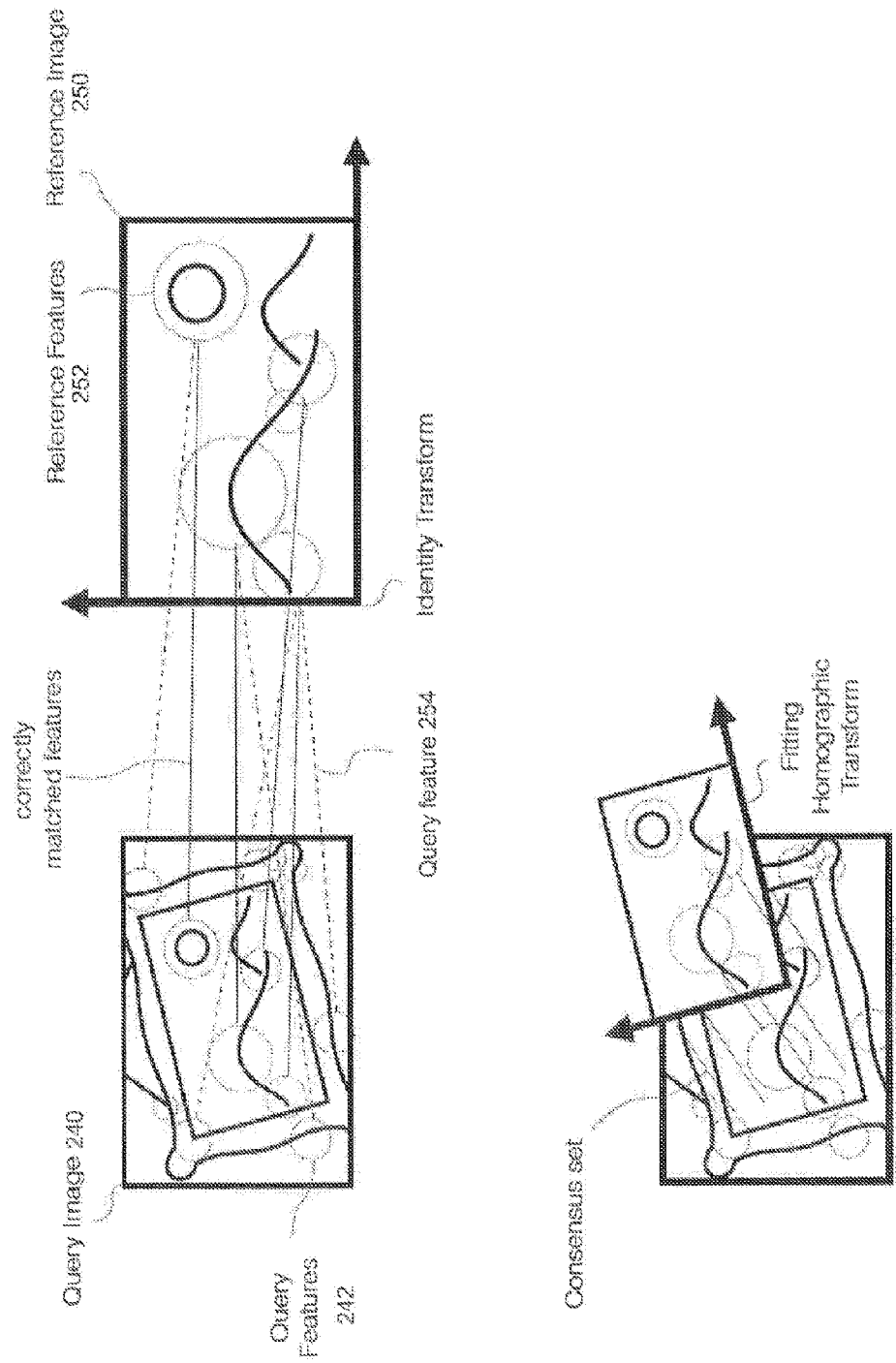


FIG. 2A  
(PRIOR ART)



**FIG. 2B**  
(PRIOR ART)

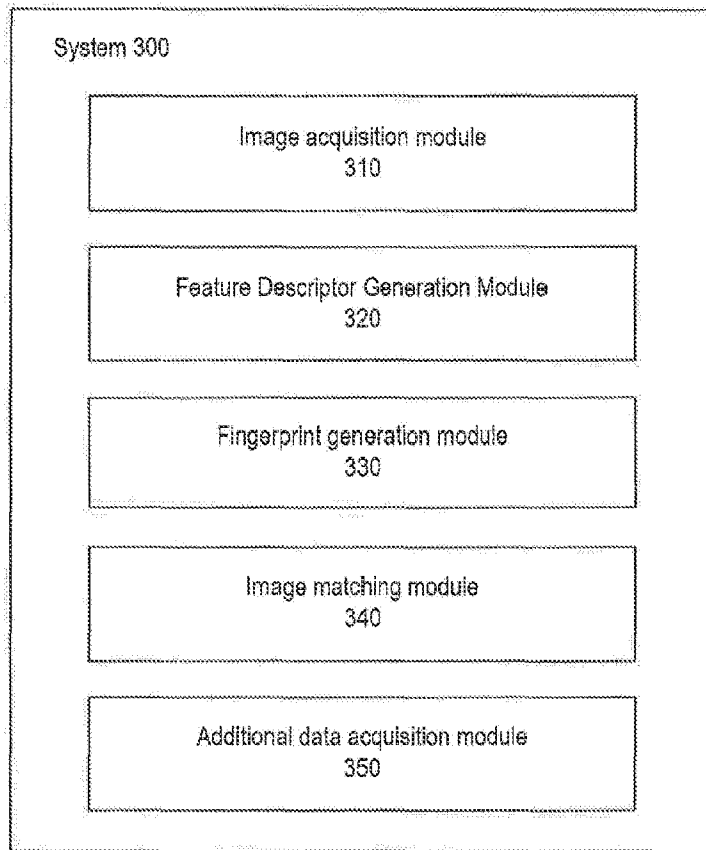


FIG. 3



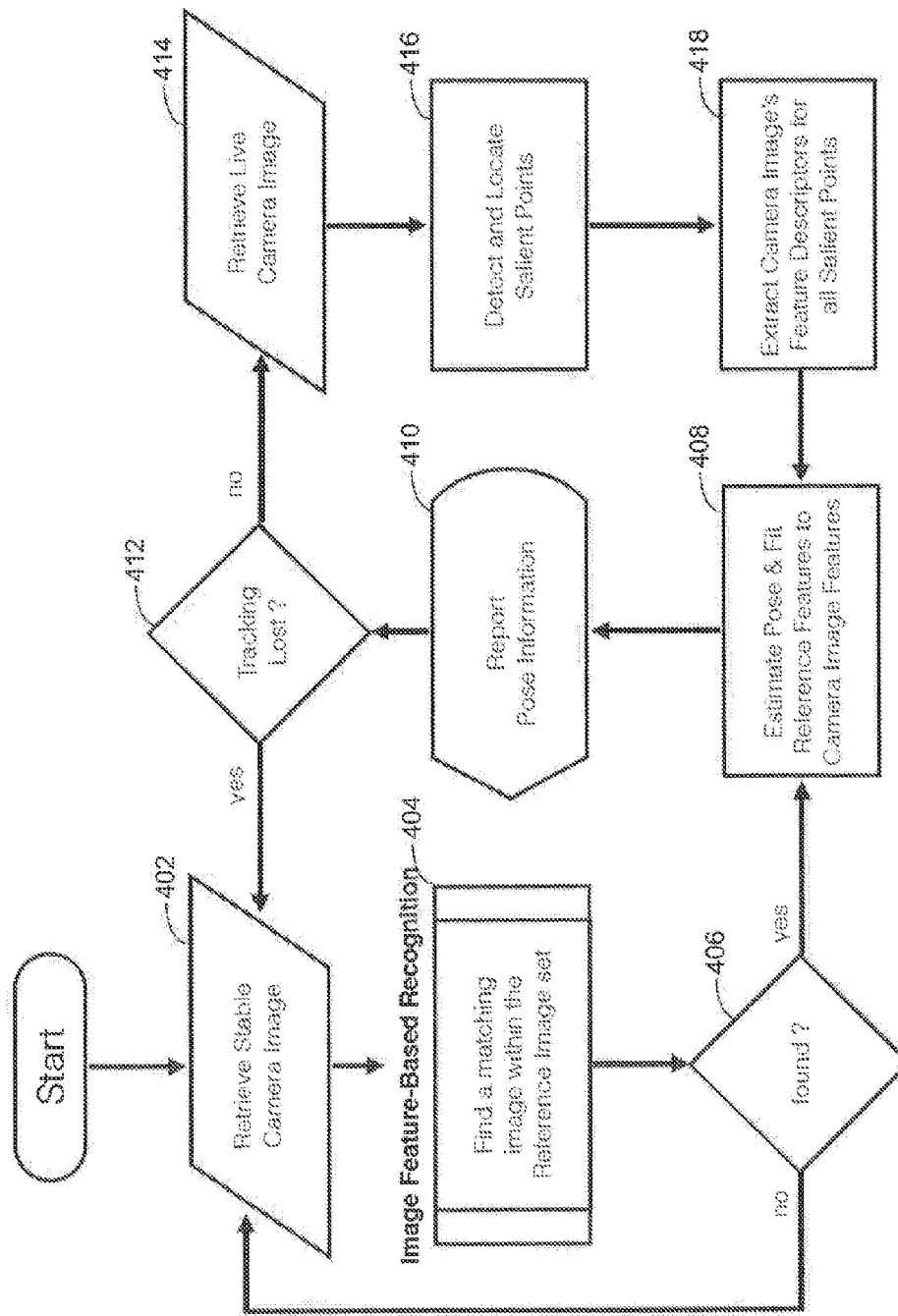


FIG. 4A

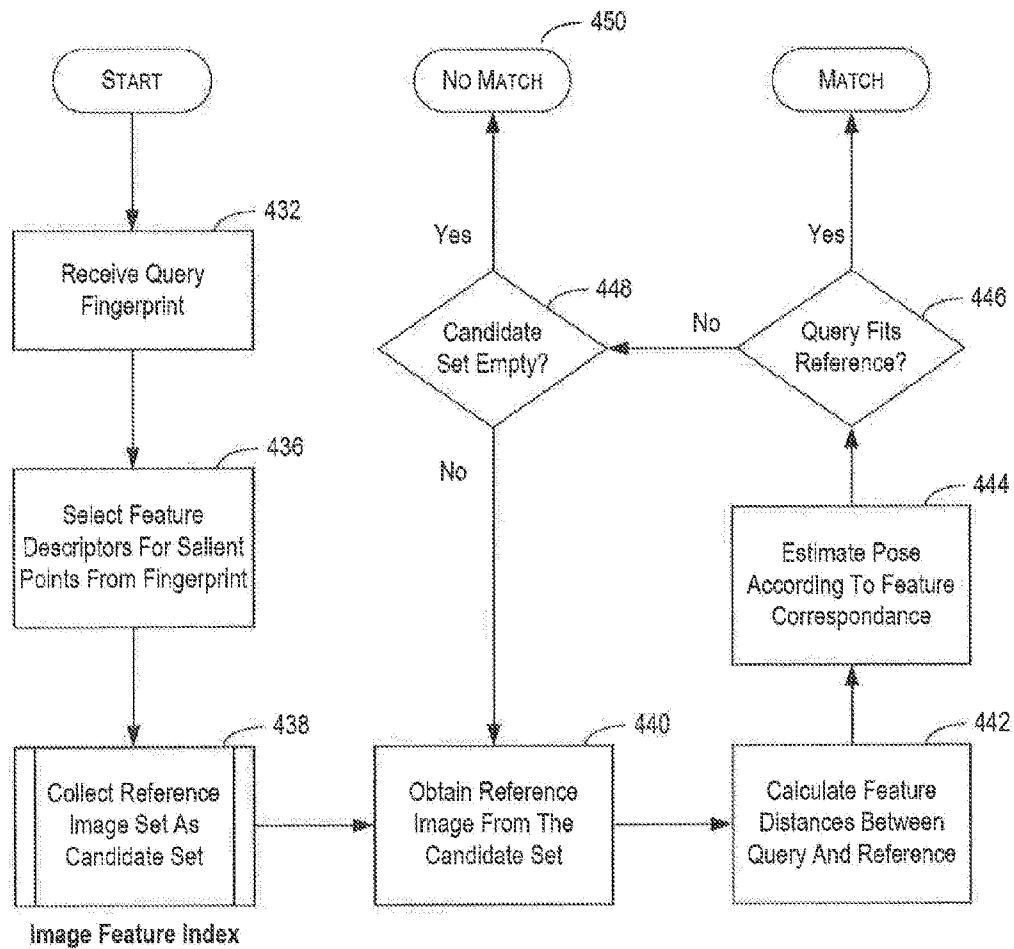


FIG. 4B

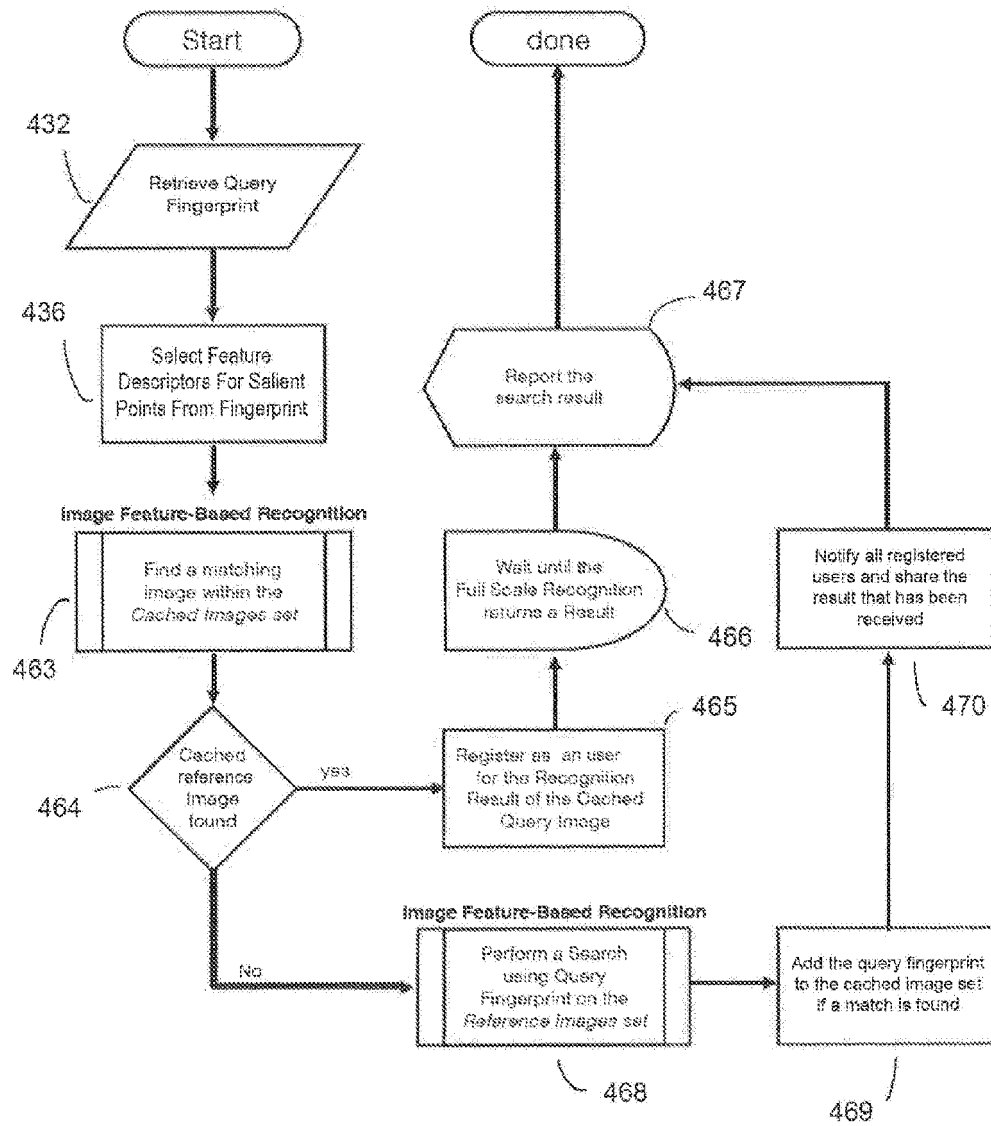


FIG. 4C

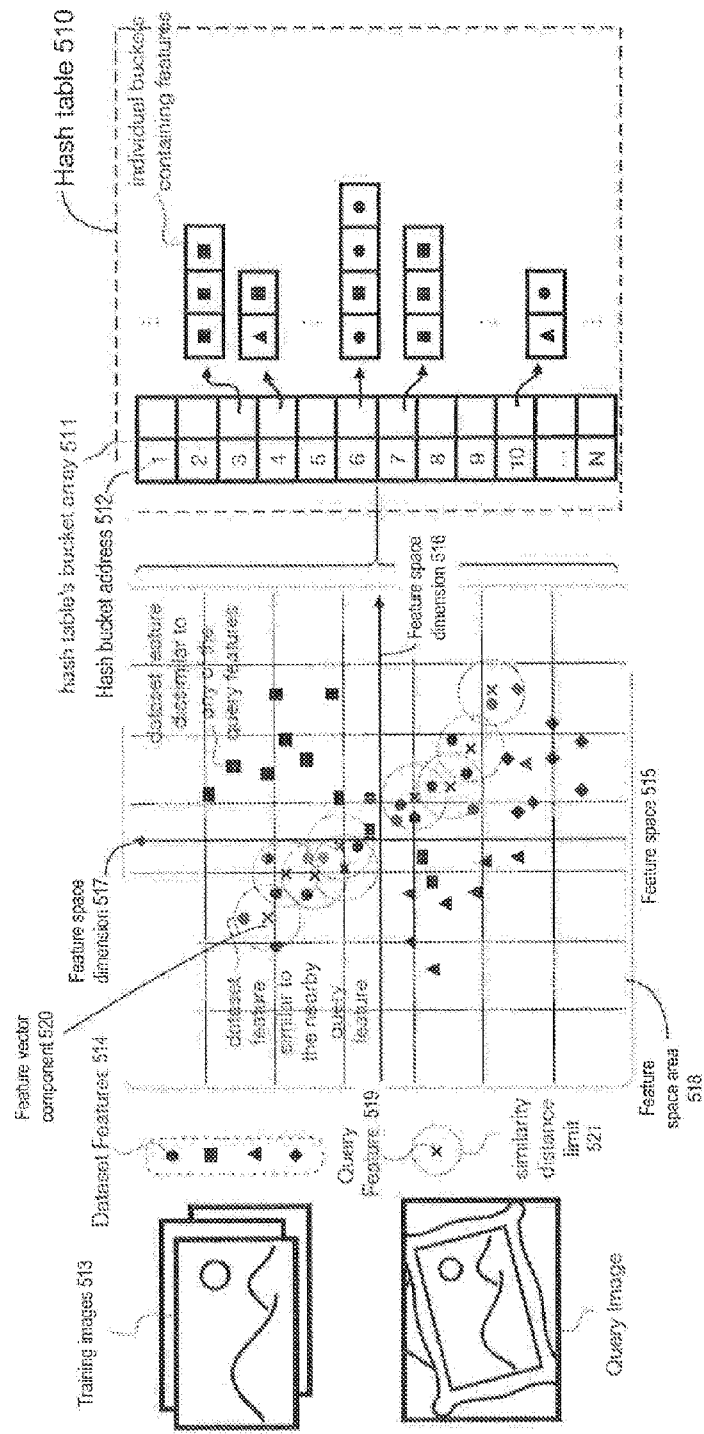
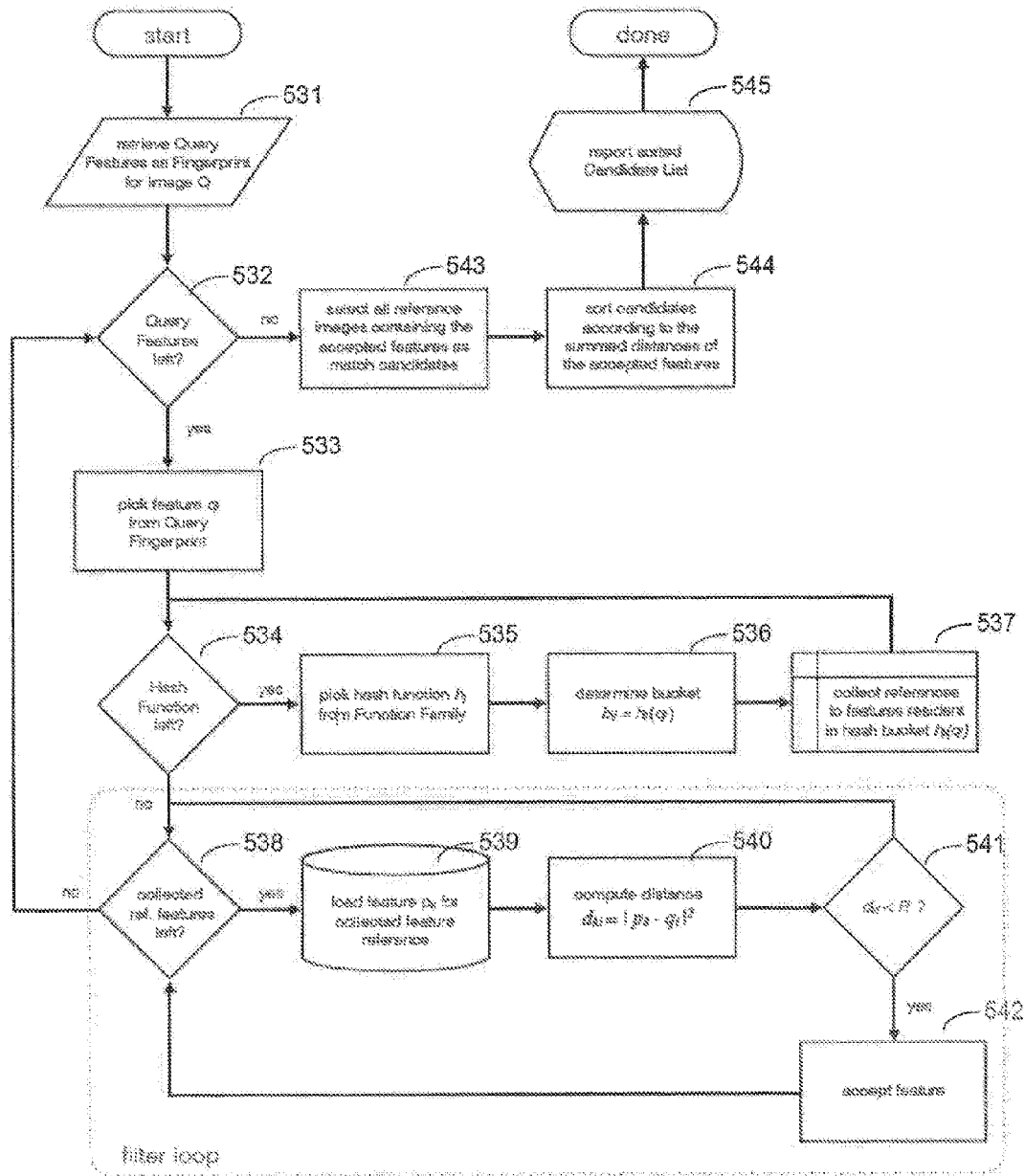


FIG. 5A



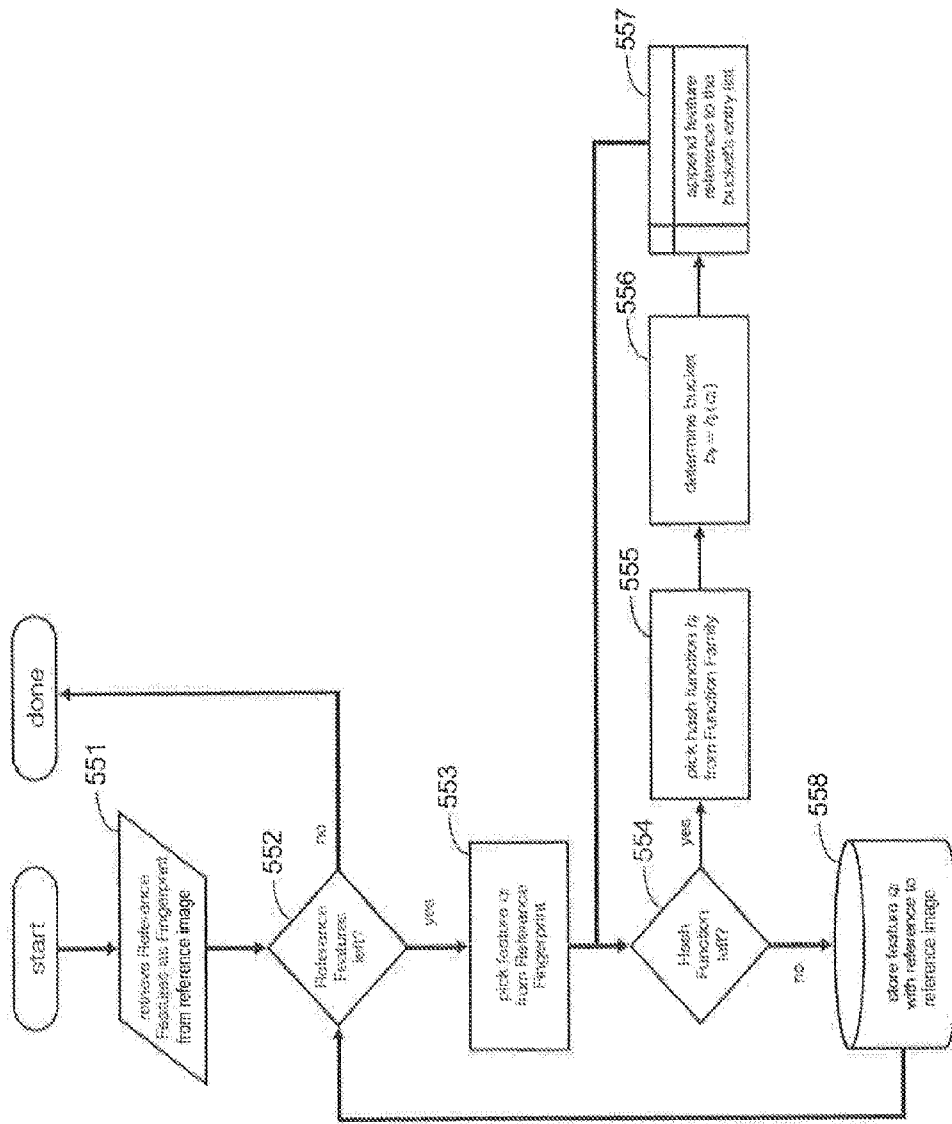
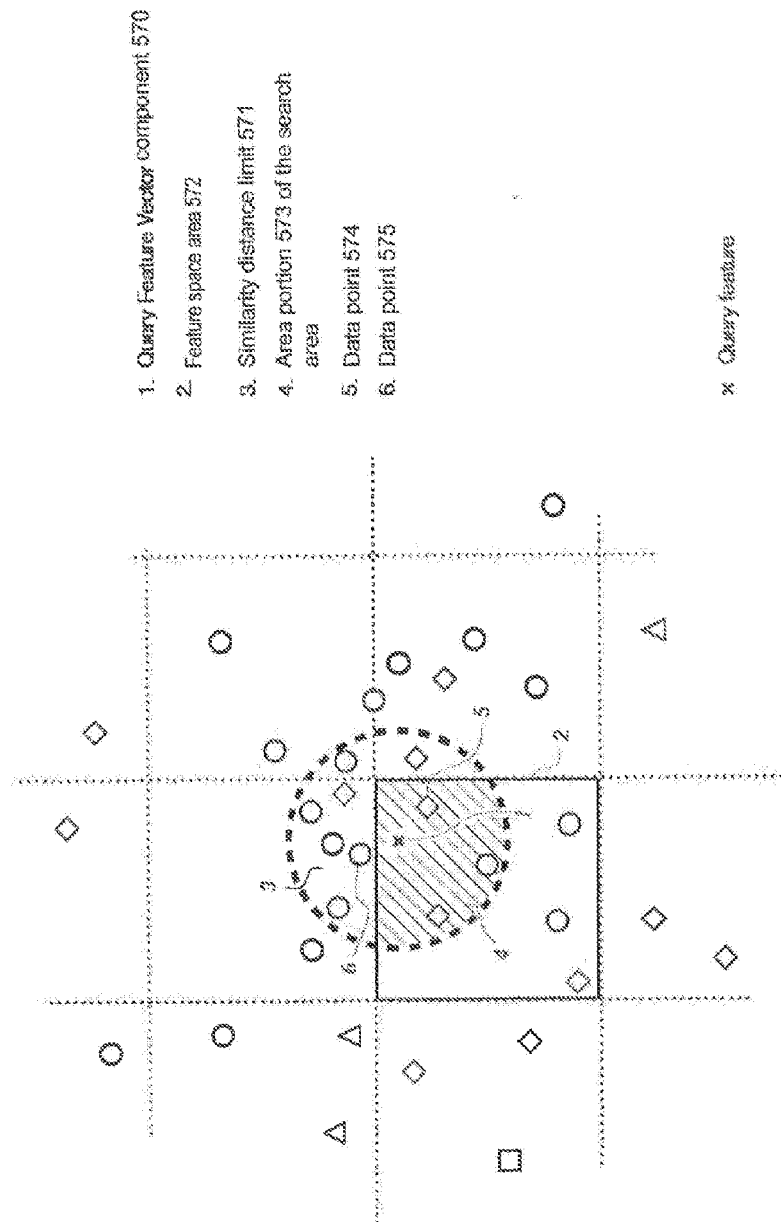


FIG. 5C



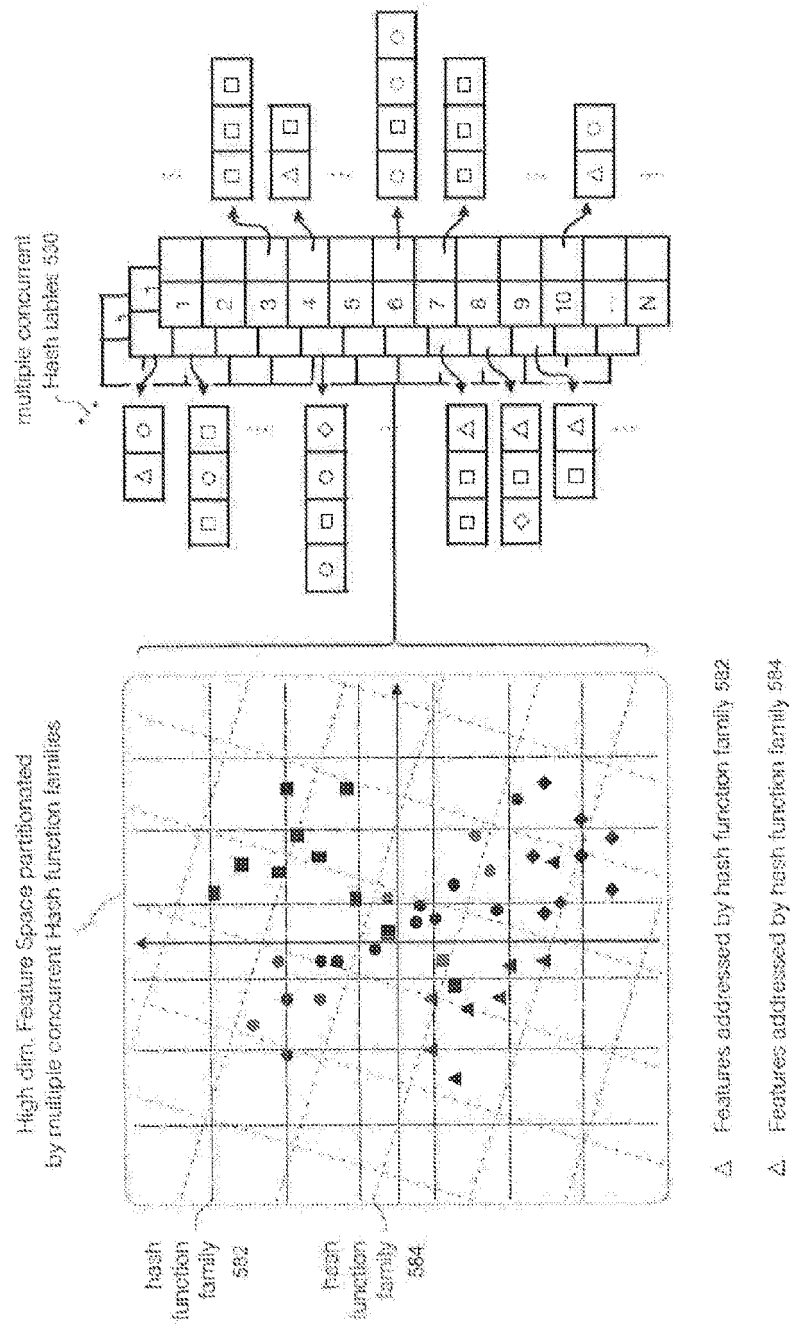


FIG. 5E



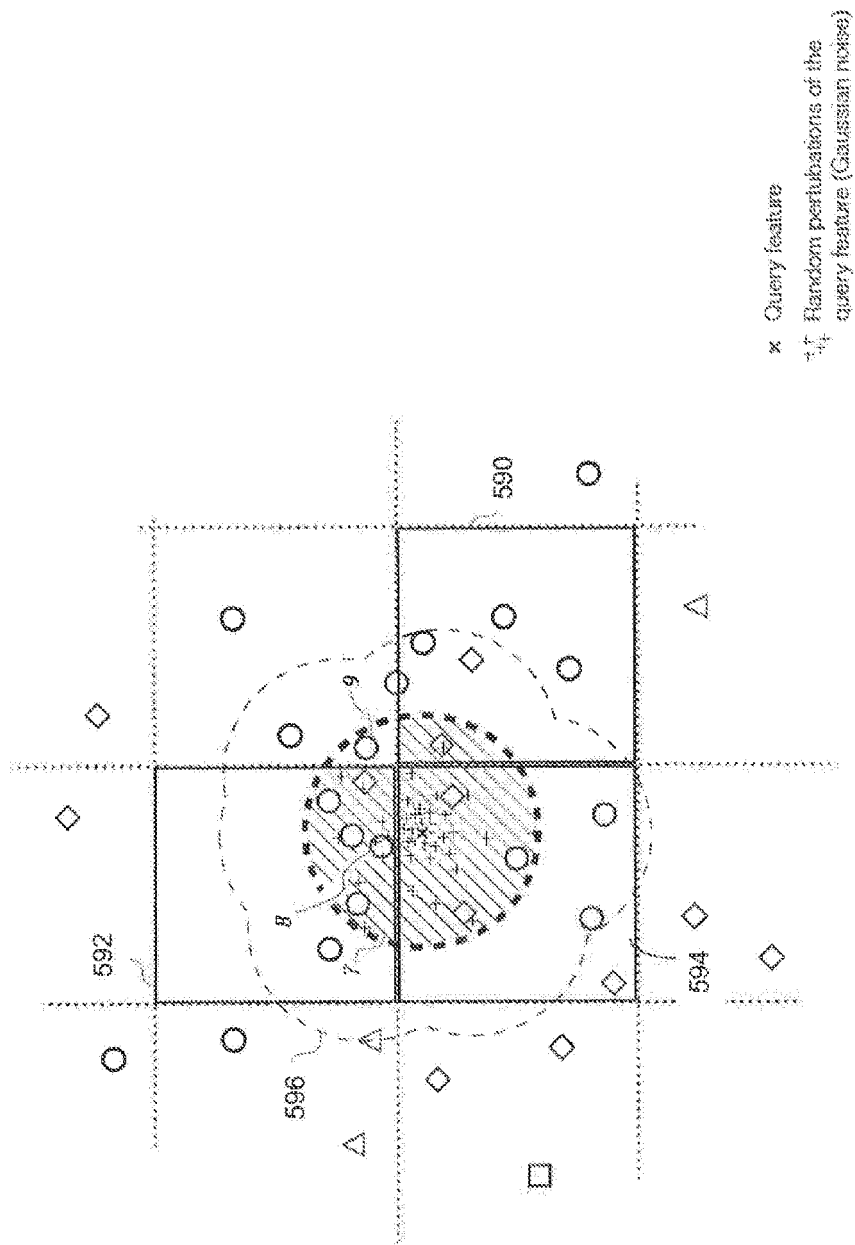


FIG. 5F

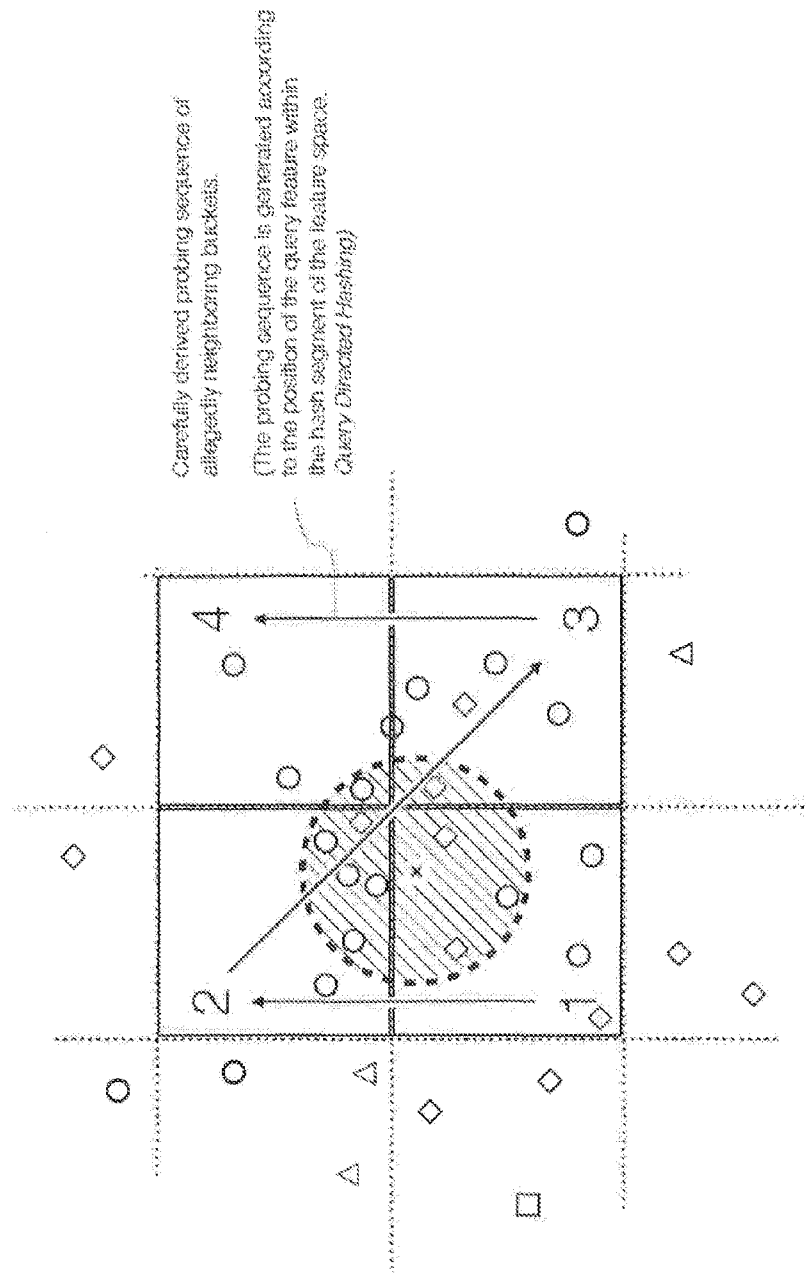


FIG. 5G

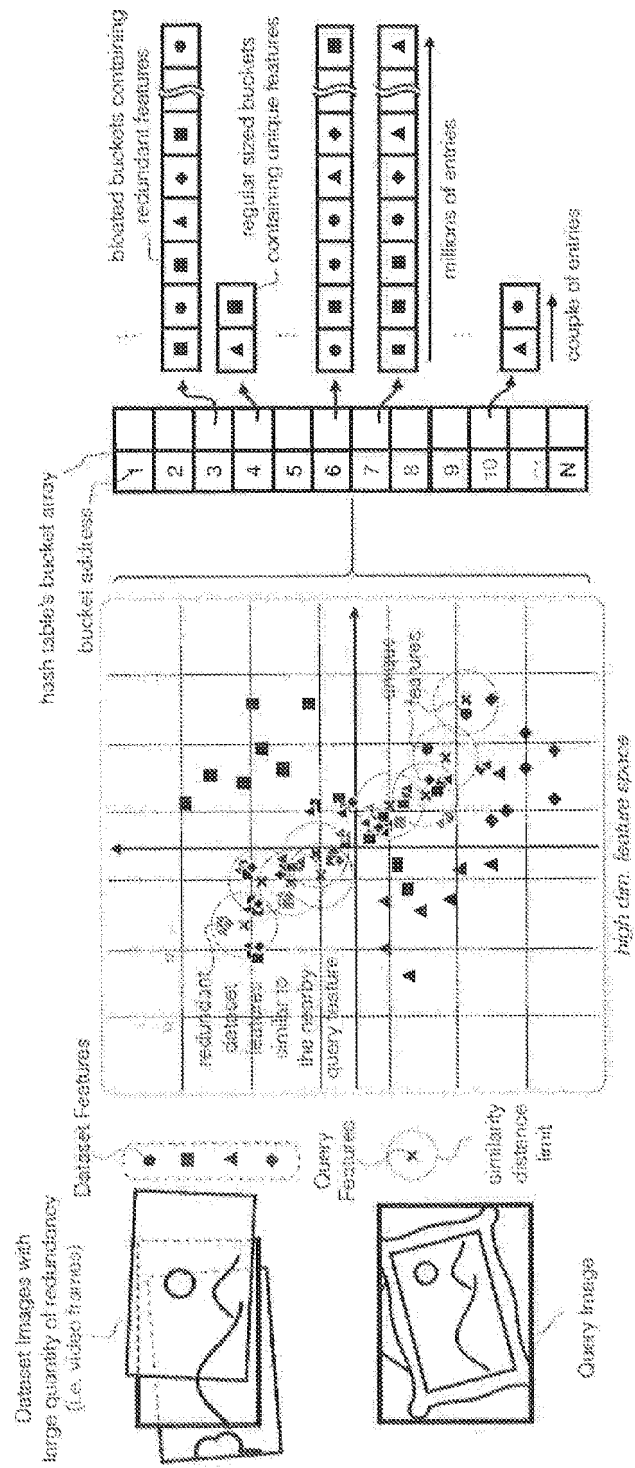


FIG. 5H

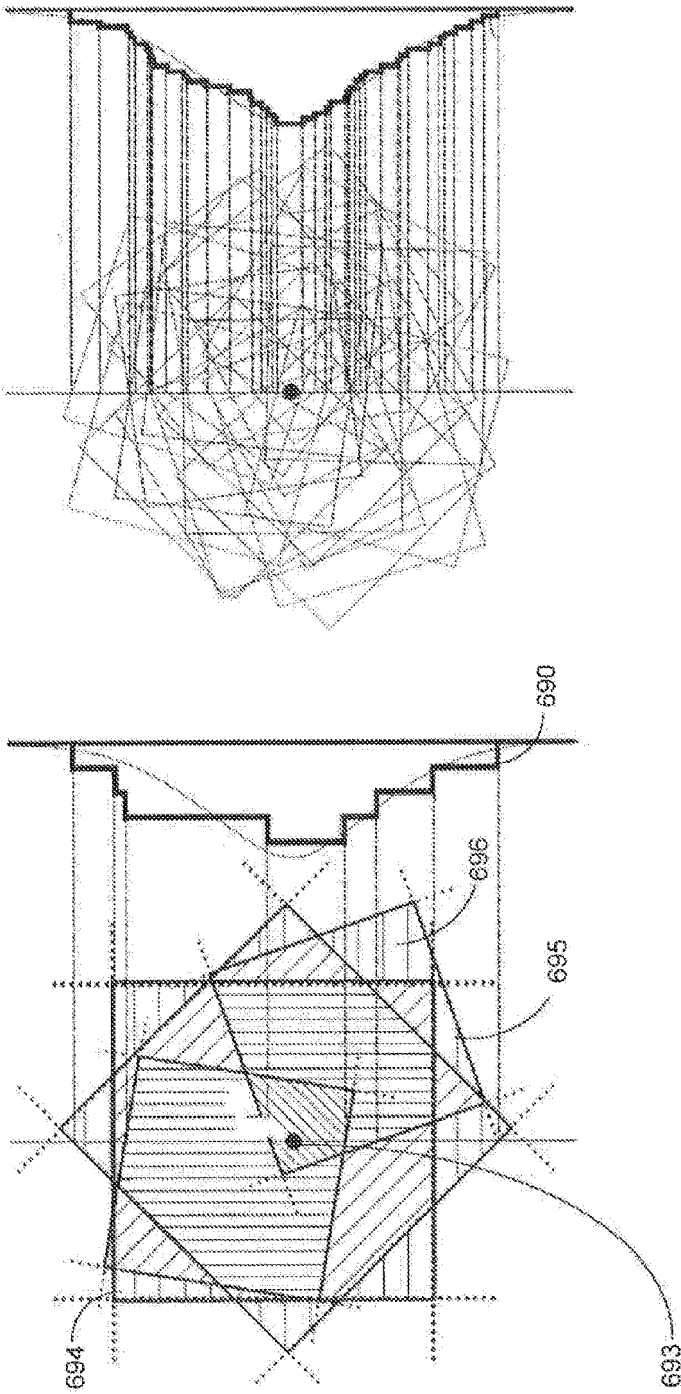
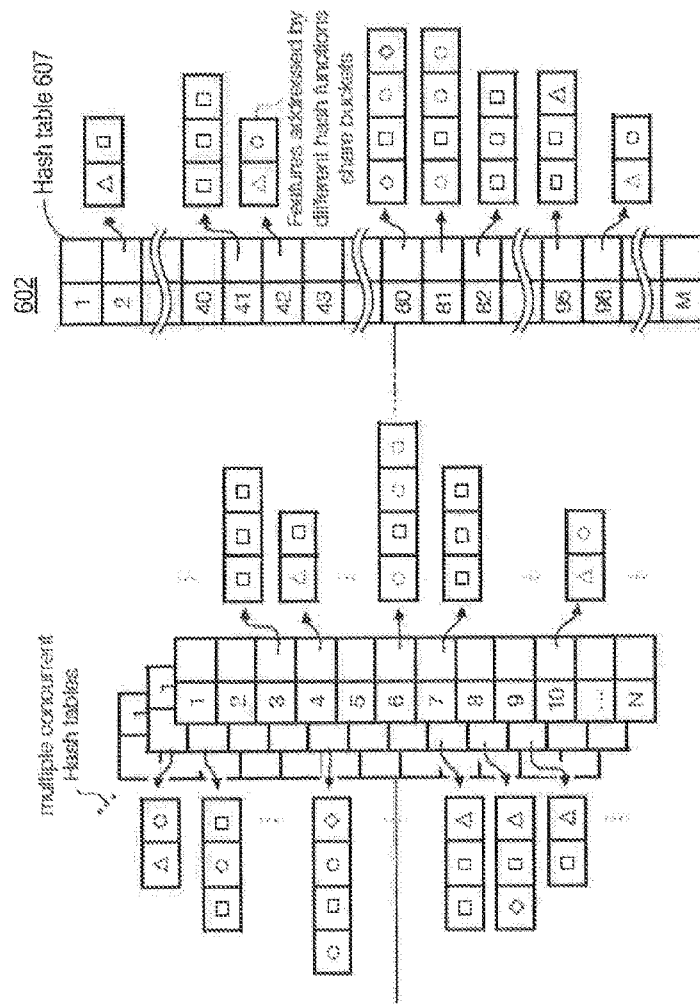


FIG. 6A



multiple concurrent hash function address spaces  
superimposed in one unified hash table

FIG. 6B

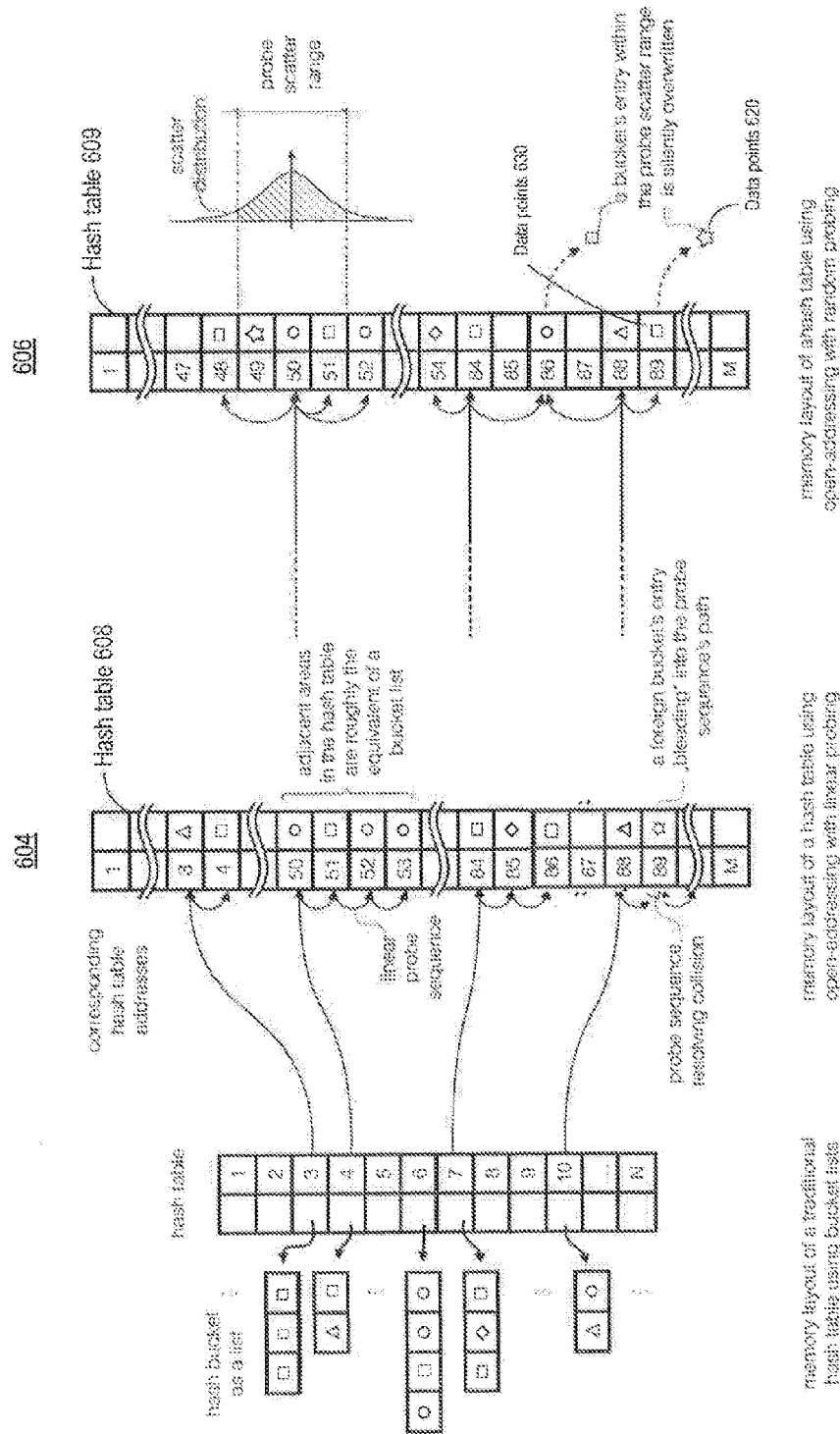


FIG. 6C

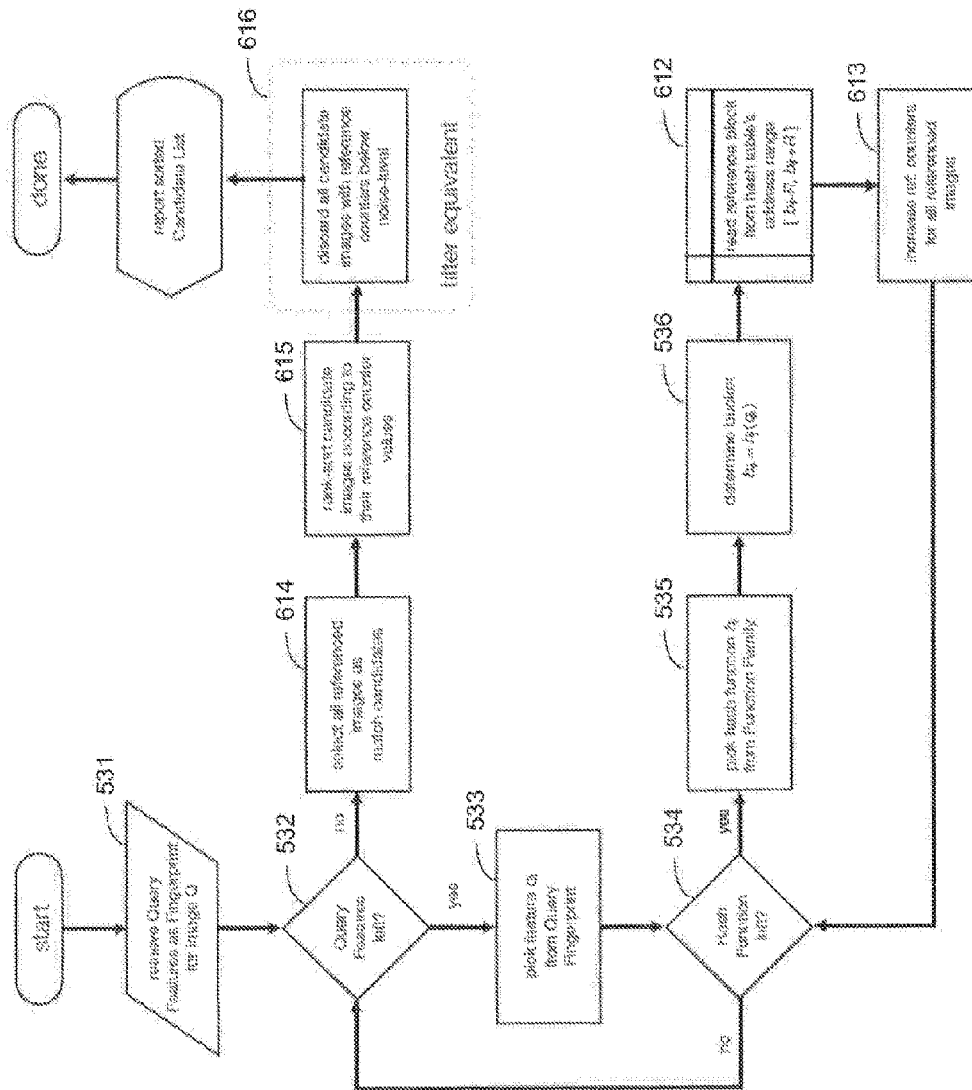


FIG. 6D

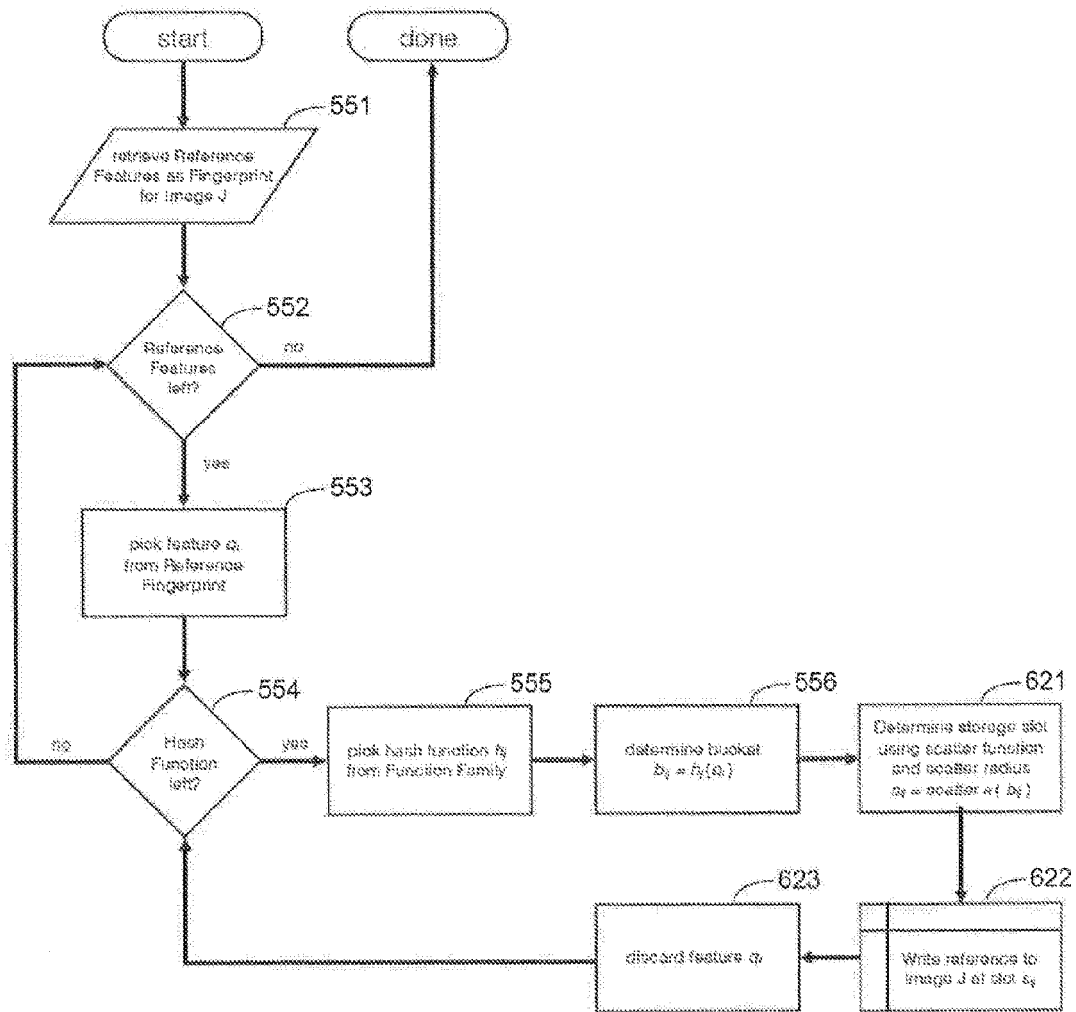


FIG. 6E



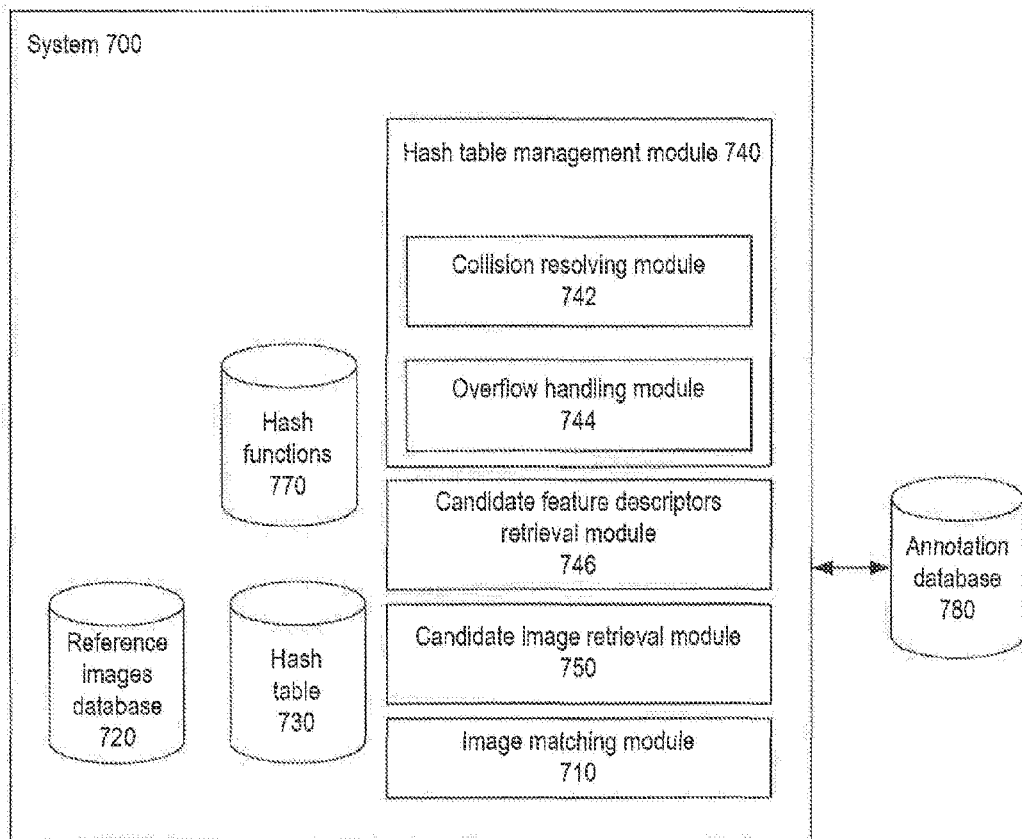


FIG. 7

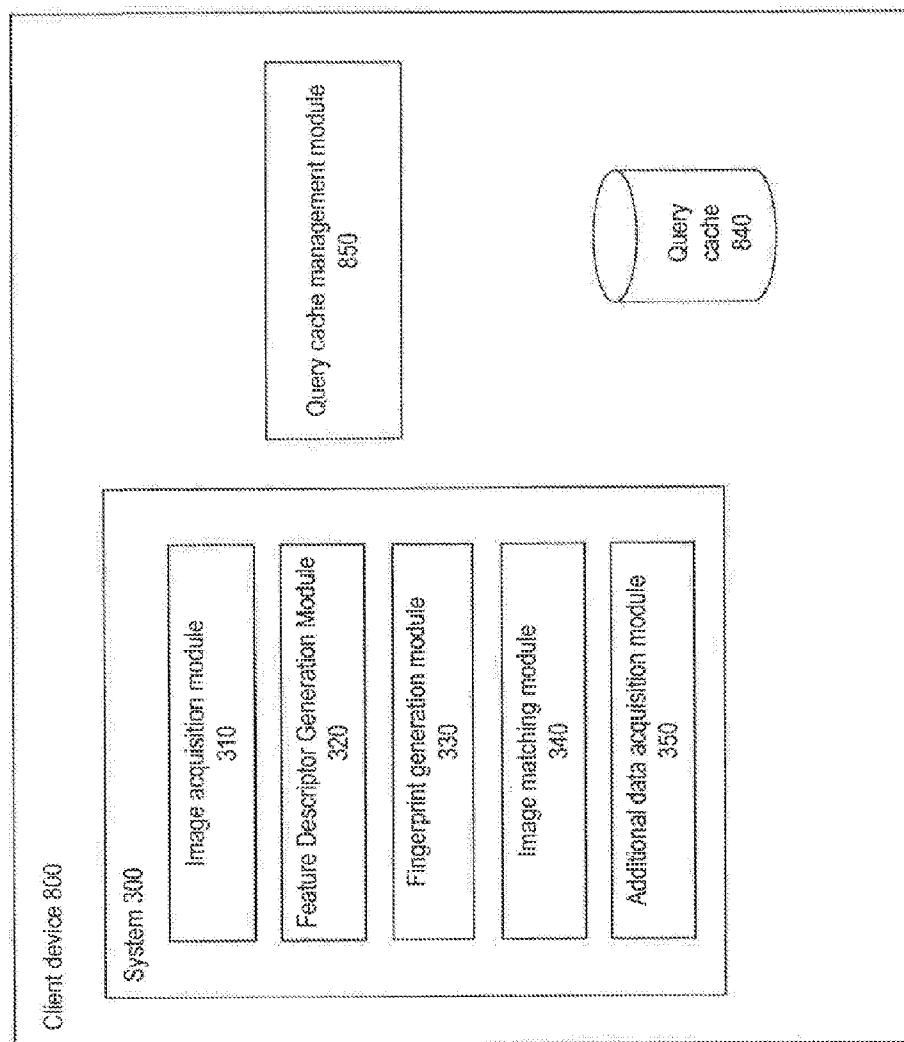


FIG. 8

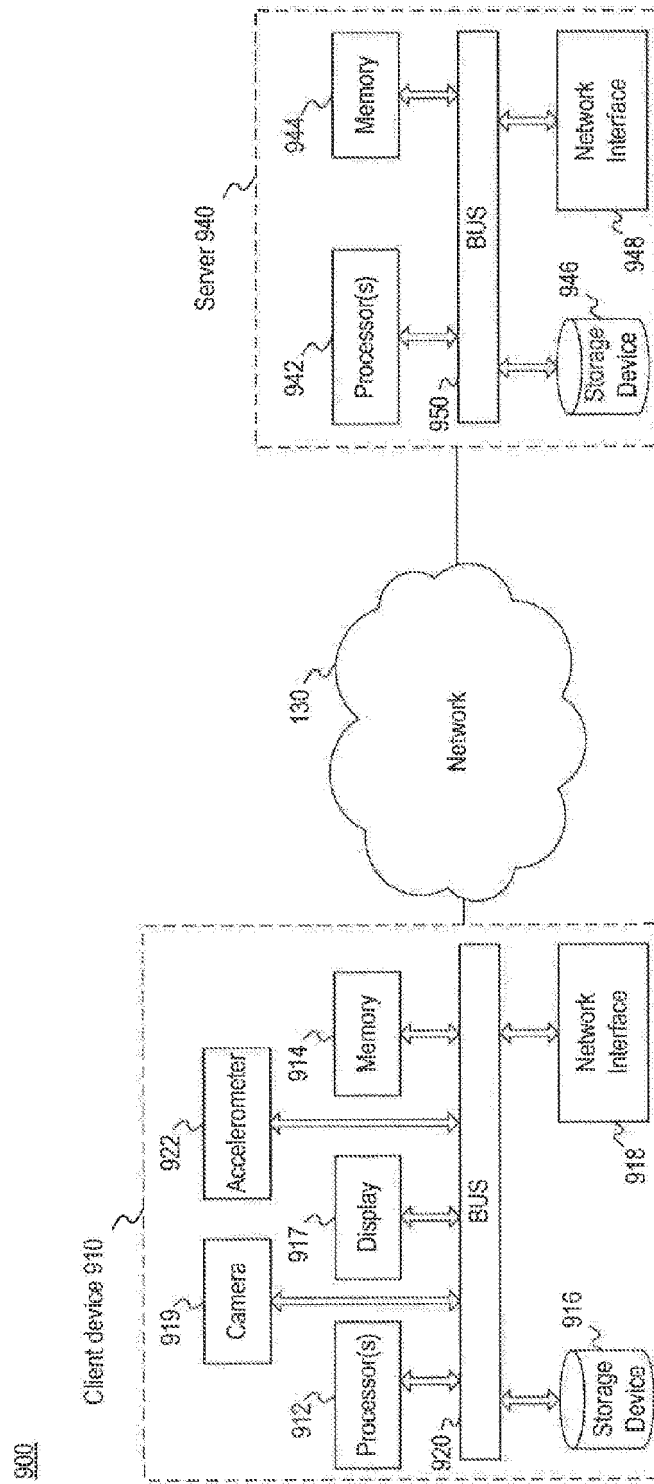


FIG. 9