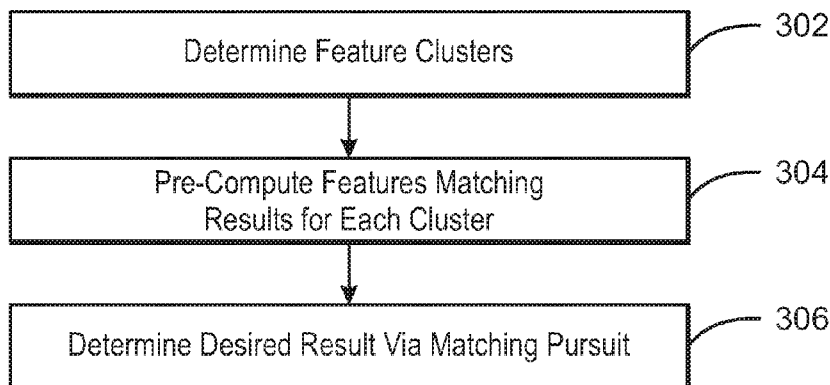




- (51) **International Patent Classification:**
G06N 99/00 (2010.01) *G06F 17/30* (2006.01)
- (21) **International Application Number:**
PCT/US2016/059366
- (22) **International Filing Date:**
28 October 2016 (28.10.2016)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
14/998,235 26 December 2015 (26.12.2015) US
- (71) **Applicant:** INTEL CORPORATION [US/US]; 2200 Mission College Blvd., Santa Clara, California 95054 (US).
- (72) **Inventors:** PAUL, Arnab; Hawthorn Farm 3, 5200 NE Elam Young Pkwy, Hillsboro, Oregon 97124 (US). JAIN, Nilesh; 4377 NW Diamondback Dr, Beaverton, Oregon 97006 (US). KUNG, Hsiang-Tsung; 2200 Mission College Blvd, MS SC12-303, Santa Clara, California 95054 (US).
- (74) **Agents:** HOLOMON, Jamilla N. et al.; International IP Law Group, P.L.L.C., c/o CPA Global, 900 Second Avenue South, Suite 600, Minneapolis, Minnesota 55402 (US).
- (81) **Designated States** (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.
- (84) **Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) **Title:** RECONSTRUCTION OF SIGNALS USING A GRAMIAN MATRIX



(57) **Abstract:** An apparatus is described herein. The apparatus includes a clustering mechanism that is to partition a dictionary into a plurality of clusters. The apparatus also includes a feature-matching mechanism that is to pre-compute feature matching results for each cluster of the plurality of clusters. Moreover, the apparatus includes a selector that is to locate a best representative feature from the dictionary in response to an input vector.

300
FIG. 3

**Declarations under Rule 4.17:**

- *as to the identity of the inventor (Rule 4.17(i))*
- *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

- *with international search report (Art. 21(3))*

RECONSTRUCTION OF SIGNALS USING A GRAMIAN MATRIX

Cross Reference to Related Application

[0001] The present application claims the benefit of the filing date of United States Patent Application Serial No. 14/998,235, by Paul et al., entitled "Reconstruction of Signals using a Gramian Matrix," filed December 26, 2015, and is incorporated herein by reference.

Background Art

[0002] Machine learning includes algorithms that can learn from and make predictions on data. In machine learning, the tasks of learning and recognition are often preceded by a feature-matching operation. An array of features is usually stored as a matrix, and can be referred to as a dictionary. Inputs may be in the form of vectors, and the matching operation is performed by multiplying the input-vector with the feature matrix. An algorithm called Matching Pursuit (MP), and its many variants, such as the Orthogonal Matching Pursuit (OMP) can be used to determine the best combination of feature vectors to describe a given input signal.

Brief Description of the Drawings

[0003] Fig. 1 is a block diagram of an electronic device that enables the reconstruction of signals using a Gramian matrix;

[0004] Fig. 2 is process flow diagram of a method for machine learning;

[0005] Fig. 3 is a process flow diagram of a method for feature matching;

[0006] Fig. 4 is a conceptual view of a high dimensional sphere of dictionary atoms;

[0007] Fig. 5 is an illustration of a set of clusters where a matching pursuit algorithm is applied; and

[0008] Fig. 6 is a block diagram showing a medium that contains logic for reconstruction of image signals via GMT.

[0009] The same numbers are used throughout the disclosure and the figures to reference like components and features. Numbers in the 100 series refer to features

originally found in Fig. 1; numbers in the 200 series refer to features originally found in Fig. 2; and so on.

Description of the Embodiments

[0010] Feature-matching may be used in sparse coding in order to provide a representation of input vectors as a weight linear combination of basis vectors. The basis vectors may be used to capture patterns in the input data. The basis vectors may be found in a dictionary and organized as a feature matrix, and the matching operation can be performed by multiplying the input-vector with the feature matrix. When the input vectors are very high-dimensional, and the feature matrix is very large, the multiplication step may consume a relatively large amount of time, which prevents real-time operations. Moreover, these vectors, and matrices all involve floating point operations that further worsens the situation by increasing the time it takes to complete the necessary computations. As a result, reconstruction of signals encoded using the sparse-coding techniques can take a large amount of time. In particular, the amount of time used for various calculations can prevent real-time implementation in hardware or software.

[0011] Embodiments described herein enable the reconstruction of features via a Gramian matrix. In embodiments, a clustering mechanism may partition a dictionary into a plurality of clusters. The feature matching results may be pre-computed for each cluster of the plurality of clusters. A best representative feature from the dictionary may be located via a matching pursuit algorithm in response to an input vector. In this manner, the number of multiplication operations is reduced significantly, enabling faster execution of feature matching, and ultimately faster execution of sparse coding techniques. The present techniques apply the clustering that occurs in the dictionary with an algebraic step in order to enable real-time operations, which results in an increase in speed of over two times compared to traditional feature-matching techniques.

[0012] Fig. 1 is a block diagram of an electronic device that enables the reconstruction of signals using a Gramian matrix. The electronic device 100 may be, for example, a laptop computer, tablet computer, mobile phone, smart phone, or a wearable device, among others. The electronic device 100 may include a central

processing unit (CPU) 102 that is configured to execute stored instructions, as well as a memory device 104 that stores instructions that are executable by the CPU 102. The CPU may be coupled to the memory device 104 by a bus 106. Additionally, the CPU 102 can be a single core processor, a multi-core processor, a computing cluster, or any number of other configurations. Furthermore, the electronic device 100 may include more than one CPU 102. The memory device 104 can include random access memory (RAM), read only memory (ROM), flash memory, or any other suitable memory systems. For example, the memory device 104 may include dynamic random access memory (DRAM).

[0013] The electronic device 100 also includes a graphics processing unit (GPU) 108. As shown, the CPU 102 can be coupled through the bus 106 to the GPU 108. The GPU 108 can be configured to perform any number of graphics operations within the electronic device 100. For example, the GPU 108 can be configured to render or manipulate graphics images, graphics frames, videos, or the like, to be displayed to a user of the electronic device 100. In some embodiments, the GPU 108 includes a number of graphics engines, wherein each graphics engine is configured to perform specific graphics tasks, or to execute specific types of workloads.

[0014] The CPU 102 can be linked through the bus 106 to a display interface 110 configured to connect the electronic device 100 to a display device 122. The display device 122 can include a display screen that is a built-in component of the electronic device 100. The display device 122 can also include a computer monitor, television, or projector, among others, that is externally connected to the electronic device 100.

[0015] The CPU 102 can also be connected through the bus 106 to an input/output (I/O) device interface 114 configured to connect the electronic device 100 to one or more I/O devices 116. The I/O devices 116 can include, for example, a keyboard and a pointing device, wherein the pointing device can include a touchpad or a touchscreen, among others. The I/O devices 116 can be built-in components of the electronic device 100, or can be devices that are externally connected to the electronic device 100.

[0016] Accordingly, the electronic device 100 also includes a sparse coding mechanism 118 to enable the representation of data in a concise manner. Sparse coding can be used to find high level features in a set of data. In some embodiments, data, such as an input vector, can be expressed as a combination of a feature matrix multiplied by a sparse representation of the input vector. In compressed sensing, the sparse representation of the input vector can be recovered, assuming that the feature matrix is known. The calculations associated with compressed sensing and sparse coding can be time consuming, especially considering the input vectors are typically high dimensional data. In embodiments, high dimensional data are data characterized by few dozen to many thousands of dimensions. For example, data associated with the JPEG, MPEG, and MP3 standards can be considered high dimensional data. In order to reduce the time spent waiting for these calculations to complete, some input data calculations can be pre-computed. The sparse representations that are pre-computed are discovered by virtue of the tendency of input data to form clusters, as described below.

[0017] The computing device also includes a storage device 122 that is a physical memory such as a hard drive, an optical drive, a flash drive, an array of drives, or any combinations thereof. The storage device 122 can store user data, such as audio files, video files, audio/video files, and picture files, among others. The storage device 122 can also store programming code such as device drivers, software applications, operating systems, and the like. The programming code stored to the storage device 122 may be executed by the CPU 102, GPU 108, or any other processors that may be included in the electronic device 100.

[0018] The CPU 102 may be linked through the bus 106 to cellular hardware 124. The cellular hardware 124 may be any cellular technology, for example, the 4G standard (International Mobile Telecommunications-Advanced (IMT-Advanced) Standard promulgated by the International Telecommunications Union – Radio communication Sector (ITU-R)). In this manner, the PC 100 may access any network 126 without being tethered or paired to another device, where the network 130 is a cellular network.

[0019] The CPU 102 may also be linked through the bus 106 to WiFi hardware 126. The WiFi hardware is hardware according to WiFi standards

(standards promulgated as Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standards). The WiFi hardware 126 enables the wearable electronic device 100 to connect to the Internet using the Transmission Control Protocol and the Internet Protocol (TCP/IP), where the network 130 is the Internet. Accordingly, the wearable electronic device 100 can enable end-to-end connectivity with the Internet by addressing, routing, transmitting, and receiving data according to the TCP/IP protocol without the use of another device. Additionally, a Bluetooth Interface 128 may be coupled to the CPU 102 through the bus 106. The Bluetooth Interface 128 is an interface according to Bluetooth networks (based on the Bluetooth standard promulgated by the Bluetooth Special Interest Group). The Bluetooth Interface 128 enables the wearable electronic device 100 to be paired with other Bluetooth enabled devices through a personal area network (PAN). Accordingly, the network 130 may be a PAN. Examples of Bluetooth enabled devices include a laptop computer, desktop computer, ultrabook, tablet computer, mobile device, or server, among others.

[0020] The block diagram of Fig. 1 is not intended to indicate that the electronic device 100 is to include all of the components shown in Fig. 1. Rather, the computing system 100 can include fewer or additional components not illustrated in Fig. 1 (e.g., sensors, power management integrated circuits, additional network interfaces, etc.). The electronic device 100 may include any number of additional components not shown in Fig. 1, depending on the details of the specific implementation. Furthermore, any of the functionalities of the CPU 102 may be partially, or entirely, implemented in hardware and/or in a processor. For example, the functionality may be implemented with an application specific integrated circuit, in logic implemented in a processor, in logic implemented in a specialized graphics processing unit, or in any other device.

[0021] In embodiments, compressive sensing may be used to sample signals. In compressive sensing, a signal may be sampled and compressed at a low sampling rate. Signals may be reconstructed with prior knowledge that the signal is sparse or compressible. Compressive sensing can be used to compress data related to a machine learning problem in a particular domain, such that solving the problem on the measurements will yield the solution of original problem. In other

words, compressive sensing finds a compressed representation of the high dimensional data with minimal distortion of the signal. By enabling machine learning with a compressed domain, the number of samples taken can be scaled according to resource costs.

[0022] Inputs to a compressive sensing system are vectors, and the feature matching operation is performed by multiplying the input-vector with the feature matrix or dictionary. A signal is called sparse if it can be represented by a few components that are non-zero and the remaining components are zero. Such a signal can be sparsely approximated. Input vectors may be very large, and/or high dimensional. For example, a high dimensional input vector may include a pixel of an image. In such a scenario, the size of the image may cause each pixel to be large, resulting in a high dimensional input vector. Here, pixels are used for ease of description only. The present techniques may apply to any sort of signal processing.

[0023] Fig. 2 is process flow diagram of a method 200 for machine learning. At block 202, feature matching is performed. In machine learning, feature matching is often performed prior to a learning and recognition task. Feature matching compares detected features and notes their similarities. At block 204, after feature matching, the machine can then proceed to learning such that given an input data set regarding a particular domain, the machine can provide accurate information regarding the domain. At block 206, recognition occurs. As the machine proceeds with learning, the machine is able to recognize members of the domain.

[0024] Feature matching enables a machine to be trained using features where the resulting match of the feature is known. A component of a subspace can be represented by a potentially infinite number of features. While there is no universal definition of a feature, a feature may be determined based on the particular domain, problem, or the type of application applied to the machine learning. For example, in image processing, a feature may be defined as an “interesting” part of an image, and features are used as a starting point for many computer vision algorithms. In audio applications, a feature might be a particular frequency range of the audio signals. While particular features are described here, these features are for exemplary purposes only and the features described herein are not limited to the particular examples.

[0025] The feature matching operation is frequently performed by multiplying each input vector with the dictionary or feature matrix. However, when the input vectors are very high-dimensional, and the feature matrix is very large, the multiplication step takes too long, preventing real-time operations. Moreover, these vectors, and matrices all involve floating point operations that can add to the time needed to complete the calculations. However, the present techniques enable real time operations by pre-computing the calculations involved in feature matching.

[0026] Fig. 3 is a process flow diagram of a method 300 for feature matching. At block 302, the feature clusters are determined. Typically, the features in a dictionary are not uniformly spread out over a high dimensional sphere. Instead, the features tend to cluster in areas along the sphere. Thus, the dictionary can be partitioned into clusters. In examples, the clustering may be performed by in an unsupervised fashion using k-means clustering. Additionally, dictionary clustering may be performed via hierarchical clustering, in which the features are clustered for a hierarchical traversal.

[0027] At block 304, future results of feature matching may be pre-computed. As noted above, the feature matching operation is frequently performed by multiplying each input vectors with the dictionary or feature matrix. In embodiments, future results of feature matching may be pre-computed using a Gramian matrix (Gram-matrix). Specifically, what is known a Gram-matrix trick (GMT), can be used to effectively pre-compute future results, thereby reducing repeated matrix multiplication of the cluster-matrices.

[0028] The use of the Gram matrix results in a less computationally intensive operation that can pre-compute the feature matching for a plurality of input vectors. The GMT reduces the number of multiplication operations to obtain the right cluster, resulting in saving a significant number of floating-point multiplication operations. In general, pre-computing based on the GMT of the cluster-centroid matrix may yield a sequence of clusters (for explaining the successive residuals) that may not be in sync with the sequence of vectors yielded by the GM pre-computation of the actual dictionary matrix. In other words, the sequence of atoms so obtained in the second case may not belong to the corresponding clusters yielded by the GMT-based matching pursuit on the cluster matrix.

[0029] However, when the clusters are physically relatively large, the GMT can be applied on the cluster-centroid matrix itself, and the resulting sequence will stay in sync with the actual sequence that would have been obtained had the original dictionary been used for feature-matching. Since the size of the cluster-centroid matrix is much smaller, the number of multiplications performed can be reduced. Thus, combining the GMT with large clusters can result in much improved performance of a matching-pursuit algorithm.

[0030] At block 306, a desired result can be determined via matching pursuit. When an input vector is provided, matching pursuit enables the pre-computed result to be located. In embodiments, matching pursuit is a sparse approximation that locates the best matching projection of the high dimensional data onto an over-complete dictionary. Matching pursuit locates an atom of the dictionary that has the largest inner product with the input vector, subtracts the contribution due to that atom, and the repeats this process until the input vector is satisfactorily decomposed.

[0031] Fig. 4 is a conceptual view of a high dimensional sphere 400 of dictionary atoms. As used herein, a dictionary atom is a feature of the domain. The sphere is centered on an x-axis 402, a y-axis 404, and a z-axis 406. Each axis is normalized such that the sphere 400 is a unit sphere. Typically, the dictionary atoms occur in clusters. Thus, the sphere 400 is illustrated with clusters 410, 412, 414, and 416. Each cluster has a centroid 420, 422, 424, and 426, respectively. Each cluster represents a region where the feature-vectors of the dictionary generally cluster over the unit sphere. The centroid vectors 430, 432, 434, and 436 are collected into a new matrix C. A matching pursuit (MP) algorithm can be executed over this cluster-centroid matrix C.

[0032] Fig. 5 is an illustration of a set of clusters 500 where a matching pursuit algorithm is applied. The set of clusters 500 includes a cluster 510 and a cluster 512. The cluster 510 includes a centroid C_i 520, and the cluster 512 includes a centroid C_i 522. In embodiments, the columns of the C_i of the new matrix, represents a cluster of feature, and the matching pursuit is run over this to select the best candidate cluster. Thus, when an input vector is obtained, the matching pursuit can be used to select the best candidate cluster. For each input vector at every

iteration(i), the best representative feature (one cluster centroid) is selected, and then the real best feature is retrieved from the dictionary, where the search is located only on the selected cluster.

[0033] Consider the centroid matrix C as a dictionary. A sequence of matching pursuit iterations may be performed over the columns of the matrix C . At each iteration, the actual cluster residual f_i will determine the next cluster selected. If each cluster spans a sufficiently large area of the unit sphere, then there is a high chance that the actual residual f_{i+1} and the cluster residual a_{i+1} will both end up pointing to the same cluster. In this manner, the product of the centroid matrix and the input vectors is computed a single time, and the GMT is subsequently used to compute the next cluster residuals a_{i+1} . In embodiments, the residuals may represent the computation results located in each cluster.

[0034] For example, consider a unit sphere with a center 502 and clusters 510 and 512. The vector 530 represents a dictionary atom of the cluster 510, and the vector 532 represents a dictionary atom of the cluster 512. An input vector 534, is input into feature matching as described herein. The multiplication $C^T X$ has been computed a single time, and represents the input vector X multiplied by the feature matrix C^T . Accordingly, when the input vector/residual 534 is input, the cluster residual 538 (which becomes 542 after normalization) indicates the next cluster to be selected according to a matching pursuit algorithm. The actual residual 536 represents the direction of the next feature vector to be selected based on the input vector, while the residual based on the cluster 538 represents the residual from the currently selected cluster. According to the present techniques, the actual residual 536 and the residual based on the cluster 538 can be used to guide the matching pursuit algorithm to the same cluster C_{i+1} in the next iteration. Since the actual residual normalized 540 and the cluster residual normalized 542 belong to the same cluster 522, the matching pursuit algorithm has succeeded in finding the correct result of $C^T X$ for the input vector 534 within the cluster 522. It is important to note that the vectors 540 and 542 are likely to belong to the same cluster 522 (as long as the cluster-size is reasonably big, this is expected). This property of the algorithm keeps the GMT-based matching pursuit on the cluster-matrix in sync with the GMT-based matching pursuit on the original dictionary (much larger size).

[0035] Fig. 6 is a block diagram showing a medium 600 that contains logic for reconstruction of image signals via GMT. The medium 600 may be a computer-readable medium, including a non-transitory medium that stores code that can be accessed by a processor 602 over a computer bus 604. For example, the computer-readable medium 600 can be volatile or non-volatile data storage device. The medium 600 can also be a logic unit, such as an Application Specific Integrated Circuit (ASIC), a Field Programmable Gate Array (FPGA), or an arrangement of logic gates implemented in one or more integrated circuits, for example.

[0036] The medium 600 may include modules 606-612 configured to perform the techniques described herein. For example, a clustering module 606 may be configured to partition the dictionary into clusters. A GMT module 608 may be configured to pre-compute results of feature-matching. A matching pursuit module 610 may be configured to locate the correct pre-computed result given an input vector. In some embodiments, the modules 607-612 may be modules of computer code configured to direct the operations of the processor 602.

[0037] The block diagram of Fig. 6 is not intended to indicate that the medium 600 is to include all of the components shown in Fig. 6. Further, the medium 600 may include any number of additional components not shown in Fig. 6, depending on the details of the specific implementation.

[0038] Example 1 is an apparatus. The apparatus includes a clustering mechanism that is to partition a dictionary into a plurality of clusters; a feature-matching mechanism that is to pre-compute feature matching results for each cluster of the plurality of clusters; and a selector that is to locate a best representative feature from the dictionary in response to an input vector.

[0039] Example 2 includes the apparatus of example 1, including or excluding optional features. In this example, the feature-matching mechanism is to pre-compute feature matching results using a Gram-matrix.

[0040] Example 3 includes the apparatus of any one of examples 1 to 2, including or excluding optional features. In this example, the feature-matching mechanism is to pre-compute feature matching results using a Gram-matrix trick.

[0041] Example 4 includes the apparatus of any one of examples 1 to 3, including or excluding optional features. In this example, the selector is to locate a

best representative feature from the dictionary in response to an input vector via a matching pursuit algorithm.

[0042] Example 5 includes the apparatus of any one of examples 1 to 4, including or excluding optional features. In this example, the clustering mechanism is to partition the dictionary into a plurality of clusters for a hierarchical traversal.

[0043] Example 6 includes the apparatus of any one of examples 1 to 5, including or excluding optional features. In this example, each cluster of the plurality of clusters is substantially large such that a resulting feature matching vector remains synchronized with GMT-based matching pursuit on an original dictionary.

[0044] Example 7 includes the apparatus of any one of examples 1 to 6, including or excluding optional features. In this example, the selector that is to locate the best representative feature from the dictionary in an iterative fashion by using cluster residuals to determine the next cluster to be selected until the best representative feature is found. Optionally, the cluster residuals are computed using a Gram-matrix based pre-computation.

[0045] Example 8 includes the apparatus of any one of examples 1 to 7, including or excluding optional features. In this example, the best representative feature is a feature vector.

[0046] Example 9 includes the apparatus of any one of examples 1 to 8, including or excluding optional features. In this example, pre-computing feature matching results gives an increase of at least two times when compare to traditional feature matching.

[0047] Example 10 is a method. The method includes partitioning a dictionary into a plurality of clusters; pre-computing feature matching results for each cluster of the plurality of clusters; and locating a best representative feature from the dictionary in response to an input vector.

[0048] Example 11 includes the method of example 10, including or excluding optional features. In this example, pre-computing feature matching results for each cluster of the plurality of clusters is performed using a Gram-matrix.

[0049] Example 12 includes the method of any one of examples 10 to 11, including or excluding optional features. In this example, pre-computing feature

matching results for each cluster of the plurality of clusters is performed using a Gram-matrix trick.

[0050] Example 13 includes the method of any one of examples 10 to 12, including or excluding optional features. In this example, locating a best representative feature from the dictionary in response to an input vector is performed using a matching pursuit algorithm.

[0051] Example 14 includes the method of any one of examples 10 to 13, including or excluding optional features. In this example, the dictionary is partitioned into a plurality of clusters for a hierarchical traversal.

[0052] Example 15 includes the method of any one of examples 10 to 14, including or excluding optional features. In this example, each cluster of the plurality of clusters is substantially large such that a resulting feature matching vector remains synchronized with GMT-based matching pursuit on an original dictionary.

[0053] Example 16 includes the method of any one of examples 10 to 15, including or excluding optional features. In this example, the best representative feature from the dictionary is located in an iterative fashion by using cluster residuals to determine the next cluster to be selected until the best representative feature is found. Optionally, the cluster residuals are computed using a Gram-matrix based pre-computation.

[0054] Example 17 includes the method of any one of examples 10 to 16, including or excluding optional features. In this example, the best representative feature is a feature vector.

[0055] Example 18 includes the method of any one of examples 10 to 17, including or excluding optional features. In this example, pre-computing feature matching results gives an increase of at least two times when compare to traditional feature matching.

[0056] Example 19 is a tangible, non-transitory, computer-readable medium. The computer-readable medium includes instructions that direct the processor to partition a dictionary into a plurality of clusters; pre-compute feature matching results for each cluster of the plurality of clusters; and locate a best representative feature from the dictionary in response to an input vector.

[0057] Example 20 includes the computer-readable medium of example 19, including or excluding optional features. In this example, pre-computing feature matching results for each cluster of the plurality of clusters is performed using a Gram-matrix.

[0058] Example 21 includes the computer-readable medium of any one of examples 19 to 20, including or excluding optional features. In this example, pre-computing feature matching results for each cluster of the plurality of clusters is performed using a Gram-matrix trick.

[0059] Example 22 includes the computer-readable medium of any one of examples 19 to 21, including or excluding optional features. In this example, locating a best representative feature from the dictionary in response to an input vector is performed using a matching pursuit algorithm.

[0060] Example 23 includes the computer-readable medium of any one of examples 19 to 22, including or excluding optional features. In this example, the dictionary is partitioned into a plurality of clusters for a hierarchical traversal.

[0061] Example 24 includes the computer-readable medium of any one of examples 19 to 23, including or excluding optional features. In this example, each cluster of the plurality of clusters is substantially large such that a resulting feature matching vector remains synchronized.

[0062] Example 25 includes the computer-readable medium of any one of examples 19 to 24, including or excluding optional features. In this example, the best representative feature from the dictionary is located in an iterative fashion by using cluster residuals to determine the next cluster to be selected until the best representative feature is found. Optionally, the cluster residuals are computed using a Gram-matrix trick.

[0063] Example 26 includes the computer-readable medium of any one of examples 19 to 25, including or excluding optional features. In this example, the best representative feature is a feature vector.

[0064] Example 27 includes the computer-readable medium of any one of examples 19 to 26, including or excluding optional features. In this example, pre-computing feature matching results gives an increase of at least two times when compare to traditional feature matching.

[0065] Example 28 is a system. The system includes instructions that direct the processor to a display; an image capture mechanism; a memory that is to store instructions and that is communicatively coupled to the image capture mechanism and the display; and a processor communicatively coupled to the image capture mechanism, the display, and the memory, wherein when the processor is to execute the instructions, the processor is to: partition a dictionary into a plurality of clusters; pre-compute feature matching results for each cluster of the plurality of clusters; and locate a best representative feature from the dictionary in response to an input vector.

[0066] Example 29 includes the system of example 28, including or excluding optional features. In this example, the feature-matching mechanism that is to pre-compute feature matching results using a Gram-matrix.

[0067] Example 30 includes the system of any one of examples 28 to 29, including or excluding optional features. In this example, the feature-matching mechanism that is to pre-compute feature matching results using a Gram-matrix trick.

[0068] Example 31 includes the system of any one of examples 28 to 30, including or excluding optional features. In this example, the selector is to locate a best representative feature from the dictionary in response to an input vector via a matching pursuit algorithm.

[0069] Example 32 includes the system of any one of examples 28 to 31, including or excluding optional features. In this example, the clustering mechanism is to partition the dictionary into a plurality of clusters for a hierarchical traversal.

[0070] Example 33 includes the system of any one of examples 28 to 32, including or excluding optional features. In this example, each cluster of the plurality of clusters is substantially large such that a resulting feature matching vector remains synchronized with GMT-based matching pursuit on an original dictionary.

[0071] Example 34 includes the system of any one of examples 28 to 33, including or excluding optional features. In this example, the selector that is to locate the best representative feature from the dictionary in an iterative fashion by using cluster residuals to determine the next cluster to be selected until the best representative feature is found.

[0072] Example 35 includes the system of any one of examples 28 to 34, including or excluding optional features. In this example, the cluster residuals are computed using a Gram-matrix based pre-computation.

[0073] Example 36 includes the system of any one of examples 28 to 35, including or excluding optional features. In this example, the best representative feature is a feature vector.

[0074] Example 37 includes the system of any one of examples 28 to 36, including or excluding optional features. In this example, pre-computing feature matching results gives an increase of at least two times when compare to traditional feature matching.

[0075] Example 38 is an apparatus. The apparatus includes instructions that direct the processor to a clustering mechanism that is to partition a dictionary into a plurality of clusters; a means to pre-compute feature matching results for each cluster of the plurality of clusters; and a selector that is to locate a best representative feature from the dictionary in response to an input vector.

[0076] Example 39 includes the apparatus of example 38, including or excluding optional features. In this example, the means to pre-compute feature matching results is to pre-compute feature matching results using a Gram-matrix.

[0077] Example 40 includes the apparatus of any one of examples 38 to 39, including or excluding optional features. In this example, the means to pre-compute feature matching results is to pre-compute feature matching results using a Gram-matrix trick.

[0078] Example 41 includes the apparatus of any one of examples 38 to 40, including or excluding optional features. In this example, the selector is to locate a best representative feature from the dictionary in response to an input vector via a matching pursuit algorithm.

[0079] Example 42 includes the apparatus of any one of examples 38 to 41, including or excluding optional features. In this example, the clustering mechanism is to partition the dictionary into a plurality of clusters for a hierarchical traversal.

[0080] Example 43 includes the apparatus of any one of examples 38 to 42, including or excluding optional features. In this example, each cluster of the plurality

of clusters is substantially large such that a resulting feature matching vector remains synchronized with GMT-based matching pursuit on an original dictionary.

[0081] Example 44 includes the apparatus of any one of examples 38 to 43, including or excluding optional features. In this example, the selector that is to locate the best representative feature from the dictionary in an iterative fashion by using cluster residuals to determine the next cluster to be selected until the best representative feature is found. Optionally, the cluster residuals are computed using a Gram-matrix based pre-computation.

[0082] Example 45 includes the apparatus of any one of examples 38 to 44, including or excluding optional features. In this example, the best representative feature is a feature vector.

[0083] Example 46 includes the apparatus of any one of examples 38 to 45, including or excluding optional features. In this example, the means to pre-compute feature matching results gives an increase of at least two times when compare to traditional feature matching.

[0084] Some embodiments may be implemented in one or a combination of hardware, firmware, and software. Some embodiments may also be implemented as instructions stored on the tangible, non-transitory, machine-readable medium, which may be read and executed by a computing platform to perform the operations described. In addition, a machine-readable medium may include any mechanism for storing or transmitting information in a form readable by a machine, e.g., a computer. For example, a machine-readable medium may include read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; or electrical, optical, acoustical or other form of propagated signals, e.g., carrier waves, infrared signals, digital signals, or the interfaces that transmit and/or receive signals, among others.

[0085] An embodiment is an implementation or example. Reference in the specification to “an embodiment,” “one embodiment,” “some embodiments,” “various embodiments,” or “other embodiments” means that a particular feature, structure, or characteristic described in connection with the embodiments is included in at least some embodiments, but not necessarily all embodiments, of the present techniques.

The various appearances of “an embodiment,” “one embodiment,” or “some embodiments” are not necessarily all referring to the same embodiments.

[0086] Not all components, features, structures, characteristics, etc. described and illustrated herein need be included in a particular embodiment or embodiments. If the specification states a component, feature, structure, or characteristic “may”, “might”, “can” or “could” be included, for example, that particular component, feature, structure, or characteristic is not required to be included. If the specification or claim refers to “a” or “an” element, that does not mean there is only one of the element. If the specification or claims refer to “an additional” element, that does not preclude there being more than one of the additional element.

[0087] It is to be noted that, although some embodiments have been described in reference to particular implementations, other implementations are possible according to some embodiments. Additionally, the arrangement and/or order of circuit elements or other features illustrated in the drawings and/or described herein need not be arranged in the particular way illustrated and described. Many other arrangements are possible according to some embodiments.

[0088] In each system shown in a figure, the elements in some cases may each have a same reference number or a different reference number to suggest that the elements represented could be different and/or similar. However, an element may be flexible enough to have different implementations and work with some or all of the systems shown or described herein. The various elements shown in the figures may be the same or different. Which one is referred to as a first element and which is called a second element is arbitrary.

[0089] It is to be understood that specifics in the aforementioned examples may be used anywhere in one or more embodiments. For instance, all optional features of the computing device described above may also be implemented with respect to either of the methods or the computer-readable medium described herein. Furthermore, although flow diagrams and/or state diagrams may have been used herein to describe embodiments, the techniques are not limited to those diagrams or to corresponding descriptions herein. For example, flow need not move through each illustrated box or state or in exactly the same order as illustrated and described herein.

[0090] The present techniques are not restricted to the particular details listed herein. Indeed, those skilled in the art having the benefit of this disclosure will appreciate that many other variations from the foregoing description and drawings may be made within the scope of the present techniques. Accordingly, it is the following claims including any amendments thereto that define the scope of the present techniques.

Claims

What is claimed is:

1. An apparatus, comprising:
a clustering mechanism that is to partition a dictionary into a plurality of clusters;
a feature-matching mechanism that is to pre-compute feature matching results for each cluster of the plurality of clusters; and
a selector that is to locate a best representative feature from the dictionary in response to an input vector.
2. The apparatus of claim 1, wherein the feature-matching mechanism is to pre-compute feature matching results using a Gram-matrix.
3. The apparatus of claim 1, wherein the feature-matching mechanism is to pre-compute feature matching results using a Gram-matrix trick.
4. The apparatus of claim 1, wherein the selector is to locate a best representative feature from the dictionary in response to an input vector via a matching pursuit algorithm.
5. The apparatus of claim 1, wherein the clustering mechanism is to partition the dictionary into a plurality of clusters for a hierarchical traversal.
6. A method, comprising:
partitioning a dictionary into a plurality of clusters;
pre-computing feature matching results for each cluster of the plurality of clusters; and
locating a best representative feature from the dictionary in response to an input vector.

7. The method of claim 6, wherein each cluster of the plurality of clusters is substantially large such that a resulting feature matching vector remains synchronized with GMT-based matching pursuit on an original dictionary.

8. The method of claim 6, wherein the best representative feature from the dictionary is located in an iterative fashion by using cluster residuals to determine the next cluster to be selected until the best representative feature is found.

9. The method of claim 8, wherein the cluster residuals are computed using a Gram-matrix based pre-computation.

10. The method of claim 6, wherein the best representative feature is a feature vector.

11. The method of claim 6, wherein pre-computing feature matching results gives an increase of at least two times when compare to traditional feature matching.

12. A tangible, non-transitory, computer-readable medium comprising instructions that, when executed by a processor, direct the processor to:
partition a dictionary into a plurality of clusters;
pre-compute feature matching results for each cluster of the plurality of clusters; and
locate a best representative feature from the dictionary in response to an input vector.

13. The computer readable medium of claim 12, wherein pre-computing feature matching results for each cluster of the plurality of clusters is performed using a Gram-matrix.

14. The computer readable medium of claim 12, wherein pre-computing feature matching results for each cluster of the plurality of clusters is performed using a Gram-matrix trick.

15. The computer readable medium of claim 12, wherein locating a best representative feature from the dictionary in response to an input vector is performed using a matching pursuit algorithm.

16. A system, comprising:
a display;
an image capture mechanism;
a memory that is to store instructions and that is communicatively coupled to the image capture mechanism and the display; and
a processor communicatively coupled to the image capture mechanism, the display, and the memory, wherein when the processor is to execute the instructions, the processor is to:
partition a dictionary into a plurality of clusters;
pre-compute feature matching results for each cluster of the plurality of clusters; and
locate a best representative feature from the dictionary in response to an input vector.

17. The system of claim 16, wherein the feature-matching mechanism that is to pre-compute feature matching results using a Gram-matrix.

18. The system of claim 16, wherein the feature-matching mechanism that is to pre-compute feature matching results using a Gram-matrix trick.

19. The system of claim 16, wherein the selector is to locate a best representative feature from the dictionary in response to an input vector via a matching pursuit algorithm.

20. The system of claim 16, wherein the clustering mechanism is to partition the dictionary into a plurality of clusters for a hierarchical traversal.

21. An apparatus, comprising:
a clustering mechanism that is to partition a dictionary into a plurality of clusters;
a means to pre-compute feature matching results for each cluster of the plurality of clusters; and
a selector that is to locate a best representative feature from the dictionary in response to an input vector.
22. The apparatus of claim 21, wherein the means to pre-compute feature matching results is to pre-compute feature matching results using a Gram-matrix.
23. The apparatus of claim 21, wherein the means to pre-compute feature matching results is to pre-compute feature matching results using a Gram-matrix trick.
24. The apparatus of claim 21, wherein the selector is to locate a best representative feature from the dictionary in response to an input vector via a matching pursuit algorithm.
25. The apparatus of claim 21, wherein the clustering mechanism is to partition the dictionary into a plurality of clusters for a hierarchical traversal.

1/6

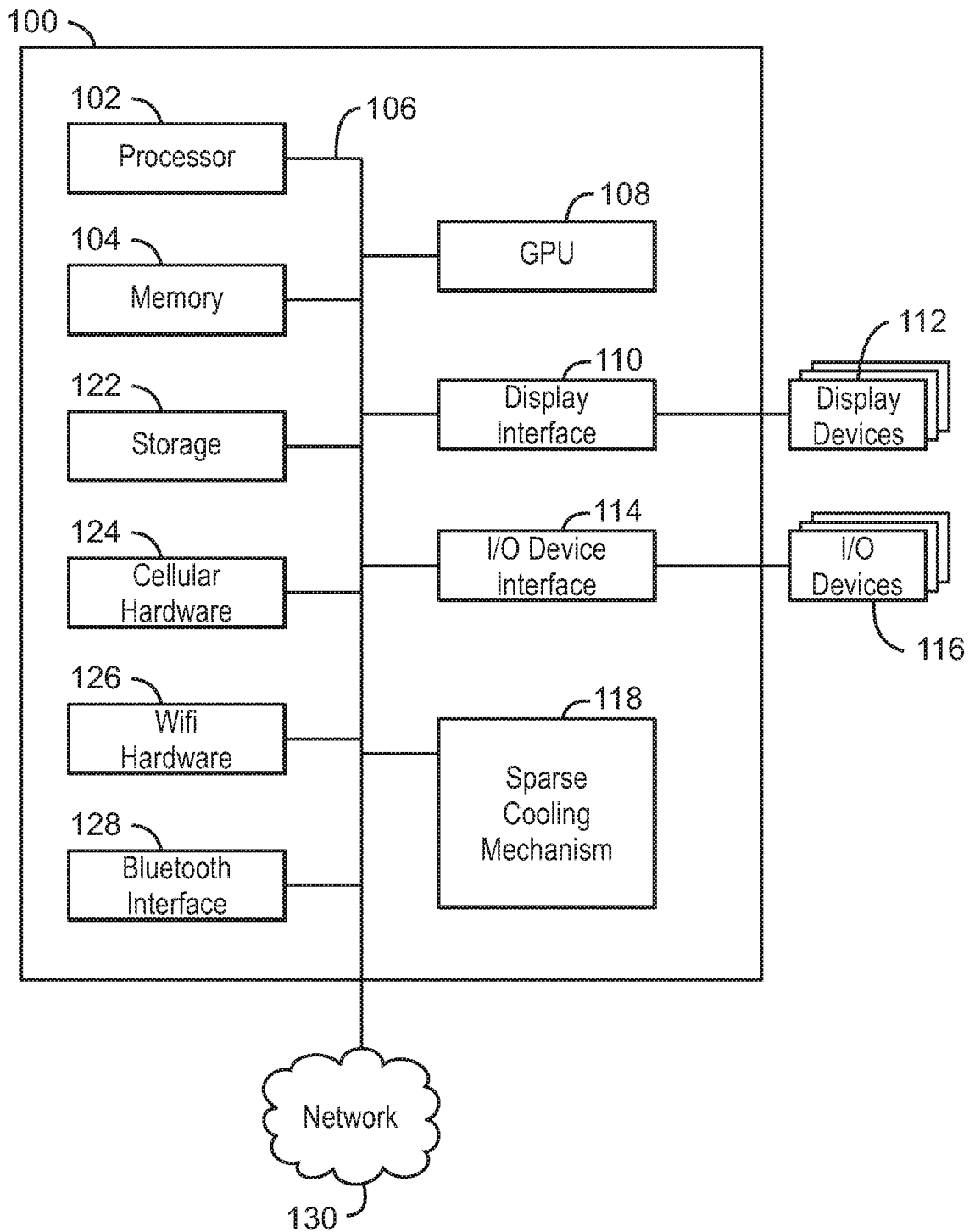
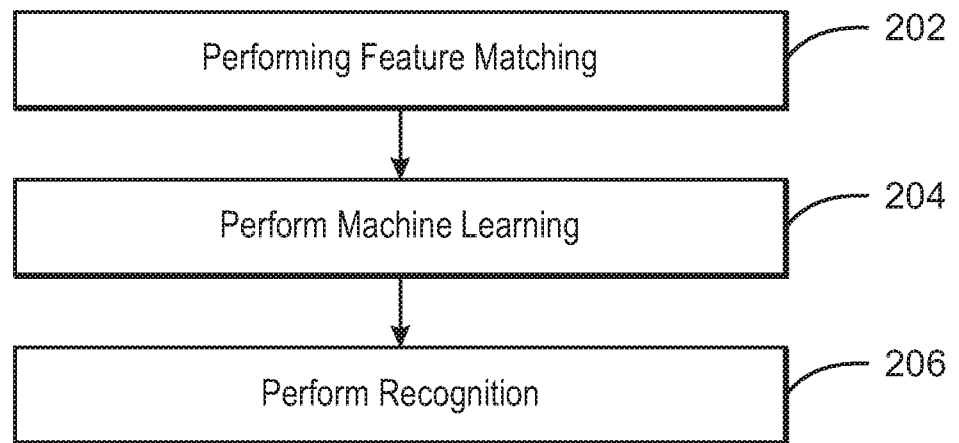


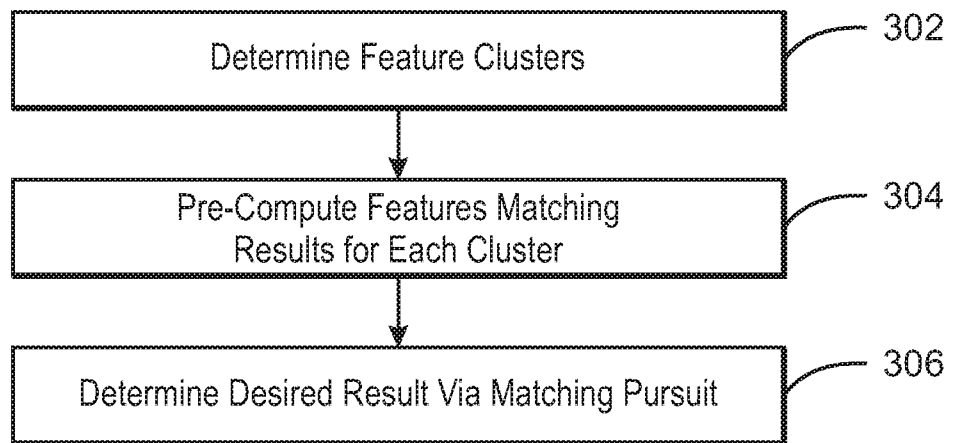
FIG. 1

2/6



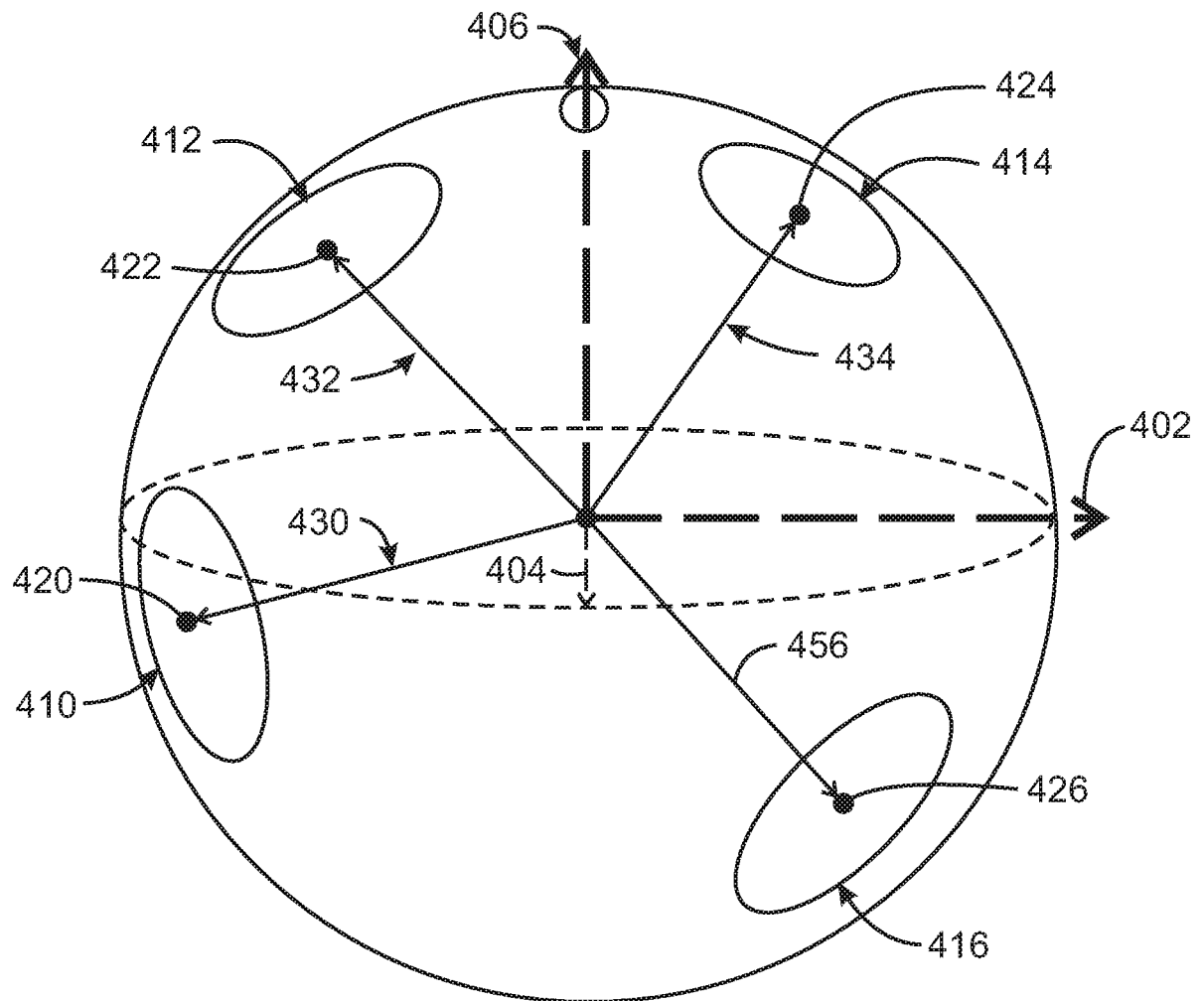
200
FIG. 2

3/6



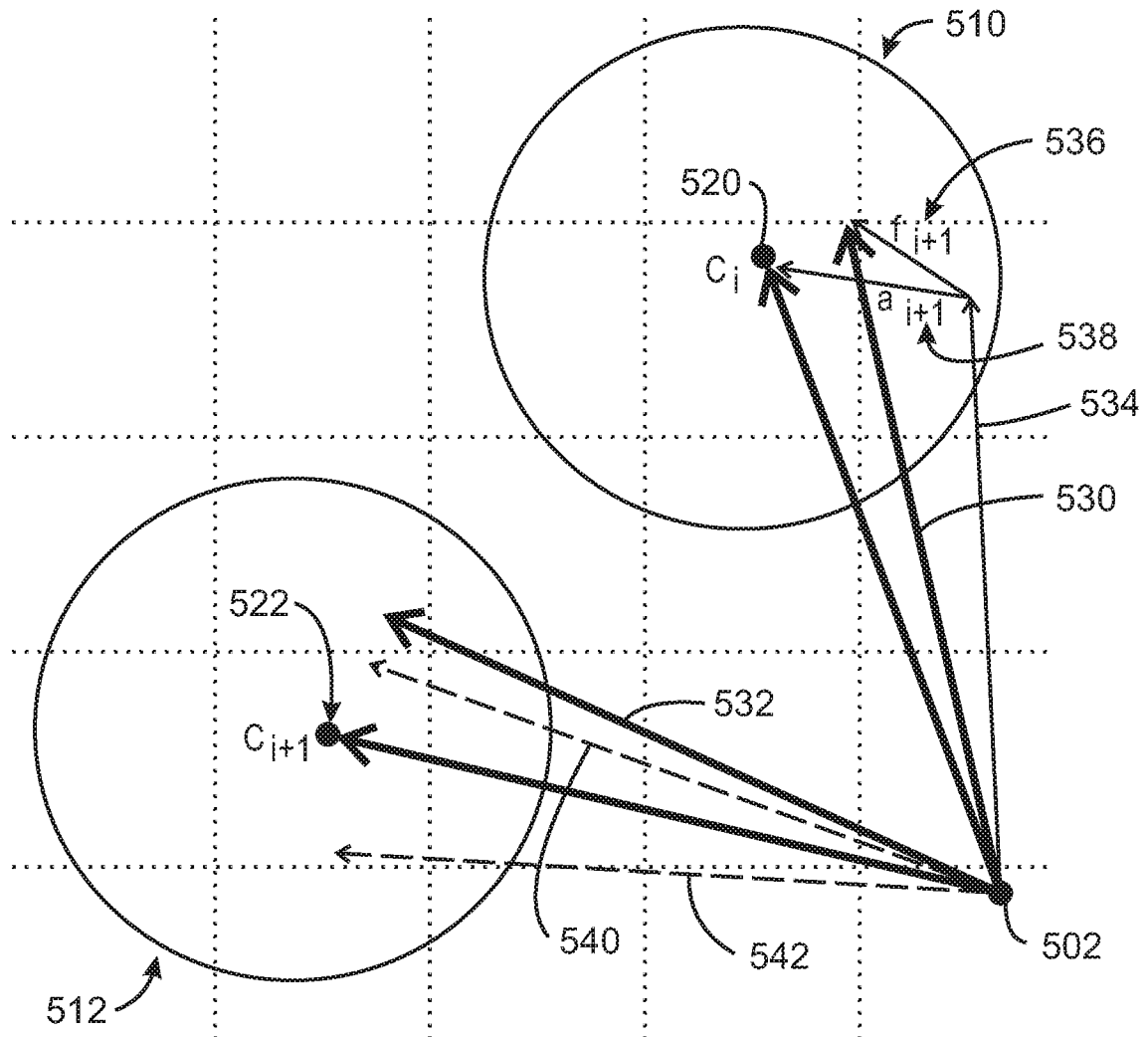
300
FIG. 3

4/6



400
FIG. 4

5/6



500
FIG. 5

6/6

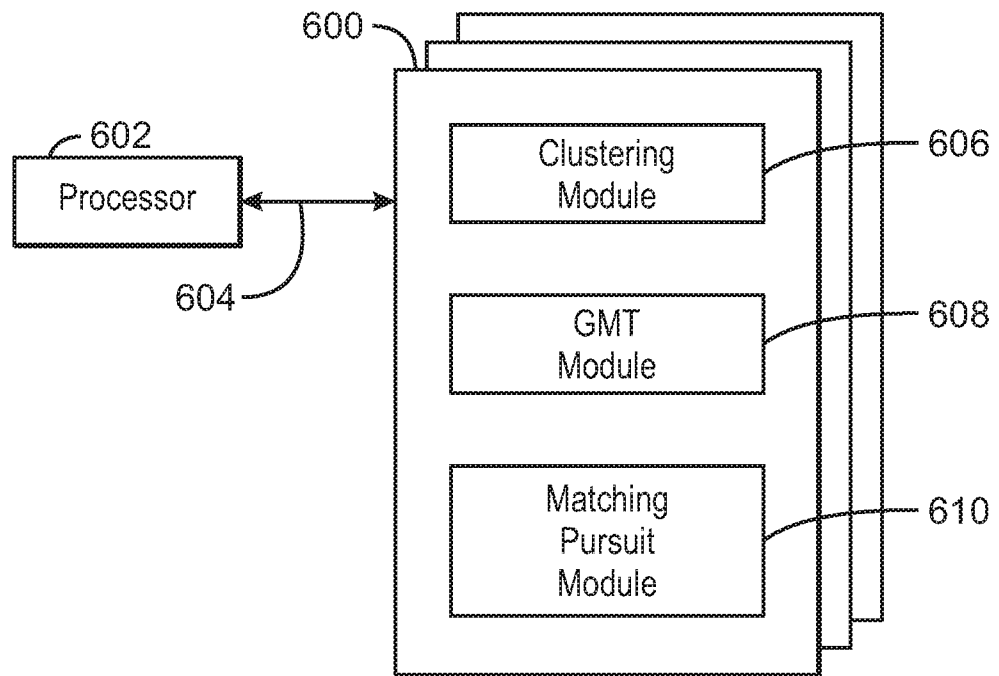


FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/059366**A. CLASSIFICATION OF SUBJECT MATTER****G06N 99/00(2010.01)i, G06F 17/30(2006.01)i**

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06N 99/00; G06F 17/30; G06K 9/62; G06N 5/00; G06F 15/18

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: clustering, dictionary, feature matching, pre-computing, input vector

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2014-0258295 A1 (MICROSOFT CORP.) 11 September 2014 See paragraphs [0007]-[0008], [0019], [0033]-[0047]; claim 1; and figure 3.	1-25
A	US 2013-0064460 A1 (TONG ZHANG et al.) 14 March 2013 See paragraphs [0024]-[0030]; claim 1; and figure 2A.	1-25
A	PABLO SPRECHMANN et al., 'Dictionary learning and sparse coding for unsupervised clustering', 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), 2010.03., pp. 2042-2045 <URL: http://ieeexplore.ieee.org/document/5494985 > See page 2043, section 2 - page 2044, section 4.1; and figure 1.	1-25
A	US 2006-0074821 A1 (NELLO CRISTIANINI) 06 April 2006 See paragraphs [0017]-[0021], [0098]-[0102]; claim 18; and figure 5.	1-25
A	JP 2008-097275 A (SHARP CORP.) 24 April 2008 See paragraphs [0043]-[0049], [0089]-[0103]; claim 1; and figures 11-12.	1-25



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search

03 February 2017 (03.02.2017)

Date of mailing of the international search report

03 February 2017 (03.02.2017)

Name and mailing address of the ISA/KR

International Application Division

Korean Intellectual Property Office

189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea

Facsimile No. +82-42-481-8578

Authorized officer

BYUN, Sung Cheal

Telephone No. +82-42-481-8262



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/059366

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014-0258295 A1	11/09/2014	None	
US 2013-0064460 A1	14/03/2013	US 8861873 B2 WO 2011-152843 A1	14/10/2014 08/12/2011
US 2006-0074821 A1	06/04/2006	EP 1384155 A1 EP 1384155 A4 US 2003-0041041 A1 US 6944602 B2 US 7299213 B2 WO 02-071243 A1	28/01/2004 28/02/2007 27/02/2003 13/09/2005 20/11/2007 12/09/2002
JP 2008-097275 A	24/04/2008	CN 101553825 A CN 101553825 B EP 2073146 A1 EP 2073146 A4 JP 4087421 B2 WO 2008-044380 A1	07/10/2009 23/05/2012 24/06/2009 21/10/2009 21/05/2008 17/04/2008