



- (51) International Patent Classification:
G06K 9/62 (2006.01) *G06F 15/173* (2006.01)
- (21) International Application Number:
PCT/US2016/041159
- (22) International Filing Date:
6 July 2016 (06.07.2016)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
14/792,596 6 July 2015 (06.07.2015) US
- (63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 14/792,596 (CON)
Filed on 6 July 2015 (06.07.2015)
- (71) Applicant: TEXAS INSTRUMENTS INCORPORATED [US/US]; P.O. Box 655474, Mail Station 3999, Dallas, TX 75265-5474 (US).
- (71) Applicant (for JP only): TEXAS INSTRUMENTS JAPAN LIMITED [JP/JP]; 24-1, Nishi-shinjuku 6-chome, Shinjuku-ku Tokyo, 160-8366 (JP).
- (72) Inventors: JAGANNATHAN, Shyam; 1852 A, 6th A Main, D Block, 2nd Stage Rajajinagar, Bangalore 560010 (IN). SWAMI, Pramod, Kumar; No. 12A, 1st Cross, Royal Placid Phase 1, Harlur Road, HSR Layout Post, Bangalore 560102 (IN).
- (74) Agents: DAVIS, Michael, A. Jr. et al.; Texas Instruments Incorporated, P.O. Box 655474, Mail Station 3999, Dallas, TX 75265-5474 (US).
- (81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM,

[Continued on next page]

(54) Title: EFFICIENT DECISION TREE TRAVERSALS IN AN ADAPTIVE BOOSTING (ADABOOST) CLASSIFIER

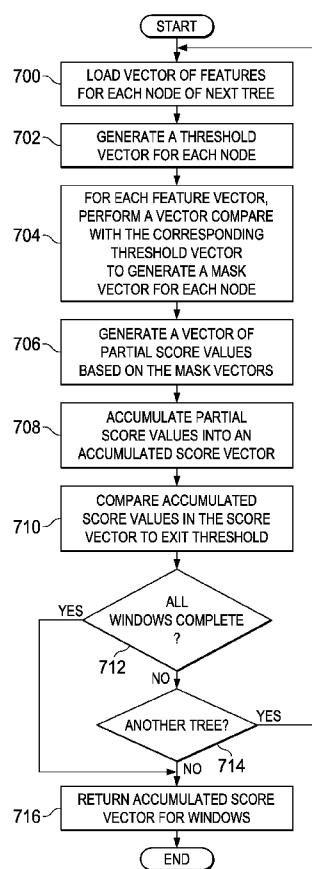


FIG. 7

(57) Abstract: In described examples of a method for object classification in a decision tree based adaptive boosting (AdaBoost) classifier implemented on a single-instruction multiple-data (SIMD) processor, the method includes receiving (700) feature vectors extracted from N consecutive window positions in an image in a memory coupled to the SIMD processor and evaluating (708) the N consecutive window positions concurrently by the AdaBoost classifier using the feature vectors and vector instructions of the SIMD processor, in which the AdaBoost classifier concurrently traverses (714) decision trees for the N consecutive window positions until classification is complete (712) for the N consecutive window positions.



AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, 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).

Declarations under Rule 4.17:

- *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*
- *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

EFFICIENT DECISION TREE TRAVERSALS IN
AN ADAPTIVE BOOSTING (ADABOOST) CLASSIFIER

[0001] This relates generally to adaptive boosting (AdaBoost) classification, and more particularly to efficient decision tree traversals in an AdaBoost classifier.

BACKGROUND

[0002] AdaBoost, short for "adaptive boosting", is an algorithm for constructing a strong classifier as a linear combination of weak classifiers, such as decision trees. In an AdaBoost classifier, the output of the weak classifiers is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive by tweaking subsequent weak learners in favor of instances misclassified by previous classifiers. AdaBoost (in which decision trees are used as the weak learners) is often referred to as the best out-of-the-box classifier and is a popular classifier for vision and data analytics. For example, a detailed description of AdaBoost is located in Y. Freund and R. Schapire, "A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting," Journal of Computer and System Sciences, Vol. 55, Issue 1, August 1997, pp.119-139.

SUMMARY

[0003] In described examples of a method for object classification in a decision tree based adaptive boosting (AdaBoost) classifier implemented on a single-instruction multiple-data (SIMD) processor, the method includes: receiving feature vectors extracted from N consecutive window positions in an image in a memory coupled to the SIMD processor, in which N is a vector width of the SIMD processor divided by a bit size of a feature, and in which a feature vector includes N feature values, one feature value for each of the N consecutive window positions; and evaluating the N consecutive window positions concurrently by the AdaBoost classifier using the feature vectors and vector instructions of the SIMD processor, in which the AdaBoost classifier concurrently traverses decision trees for the N consecutive window positions until classification is complete for the N consecutive window positions, in which a decision tree includes nodes, a threshold value for each node, and leaves, each leaf including a partial score.

[0004] In other described examples, a digital system includes a single-instruction multiple-data

(SIMD) processor, a memory component coupled to the SIMD processor, the memory component configured to store features extracted from an image, decision trees stored in the memory component, in which each decision tree includes nodes, a threshold value for each node, and leaves, each leaf including a partial score, and a decision tree based adaptive boosting (AdaBoost) classifier trained for object classification stored in the memory component, the AdaBoost classifier executable on the SIMD processor, in which the AdaBoost classifier uses the decision trees for object classification, the AdaBoost classifier configured to evaluate N consecutive window positions concurrently using the features and vector instructions of the SIMD processor, in which the AdaBoost classifier concurrently traverses decision trees for the N consecutive window positions until classification is complete for the N consecutive window positions and in which N is a vector width of the SIMD processor divided by a bit size of a feature.

[0005] In further described examples, a non-transitory computer readable medium stores software instructions. The software instructions, when executed on a single-instruction multiple-data (SIMD) processor, cause a method for object classification in a decision tree based adaptive boosting (AdaBoost) classifier to be executed. The method includes receiving feature vectors extracted from N consecutive window positions in an image in a memory coupled to the SIMD processor, in which N is a vector width of the SIMD processor divided by a bit size of a feature, and in which a feature vector includes N feature values, one feature value for each of the N consecutive window positions, and evaluating the N consecutive window positions concurrently by the AdaBoost classifier using the feature vectors and vector instructions of the SIMD processor, in which the AdaBoost classifier concurrently traverses decision trees for the N consecutive window positions until classification is complete for the N consecutive window positions, in which a decision tree includes nodes, a threshold value for each node, and leaves, each leaf including a partial score.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] FIG. 1 is an example illustrating the sliding window approach for pedestrian detection in the scene of an image.

[0007] FIG. 2 is an example illustrating the feature extraction for an image and the arrangement of the resulting features in memory for object detection.

[0008] FIG. 3 is an example binary decision tree for an AdaBoost classifier.

[0009] FIG. 4 is an example illustrating the general concept of a feature tuple in an AdaBoost classifier.

[0010] FIG. 5 is an example illustrating the general concept of partial scores in the leaves of each decision tree of an AdaBoost classifier.

[0011] FIG. 6 is an example illustrating feature vectors.

[0012] FIG. 7 is a flow diagram of a method for executing an AdaBoost classifier on a single-instruction multiple-data (SIMD) processor.

[0013] FIGS. 8-18 are examples.

[0014] FIG. 19 is a simplified block diagram of an example digital system configured to execute an embodiment of the method of FIG. 7.

[0015] FIG. 20 is a block diagram of an example SIMD digital signal processor.

DETAILED DESCRIPTION OF EXAMPLE EMBODIMENTS

[0016] Like elements in the various figures are denoted by like reference numerals for consistency.

[0017] As mentioned hereinabove, an AdaBoost classifier may be constructed as a linear combination of weak classifiers such as decision trees. Embodiments of the disclosure are directed to decision tree based AdaBoost classifiers. For example, embodiments are directed to implementing decision tree based AdaBoost classification on wide single-instruction multiple-data (SIMD) processors, also known as vector processors. Vector processors implement instructions that process multiple data points, *i.e.*, vectors of data points, simultaneously. More specifically, multiple data points can be packed into one data word and a vector instruction can perform an operation on each data point simultaneously. For example, in a 128-bit vector processor, eight 16 bit pixels of an image can be packed into one 128-bit word and the eight pixels can be processed simultaneously.

[0018] For ease of explanation, embodiments of the disclosure are described herein using an example AdaBoost classifier trained to detect objects (such as pedestrians) in an image. A sliding window approach is used to detect pedestrians in an image. FIG. 1 is an example illustrating the sliding window approach for pedestrian detection in the scene of an image. Generally, in a conventional approach, a window (also referred to as a box or an object model or an object patch) is moved through the image at overlapping horizontal and vertical positions, and features (computed based on the pixel values in the window at each position) are analyzed by the

classifier. The size of the window is based on the expected size of a pedestrian and is assumed to be 36x68. A window is examined at every fourth pixel vertically and horizontally. To identify different sizes of pedestrians, *i.e.*, pedestrians at differing distances in a scene, the pedestrian detection is performed on multiple different scales of the image.

[0019] The classification is performed based on features computed for an image at multiple different scales. For object detection, example features may include gradient magnitude, gradient orientation, block sums, intensity, and color information. A conventional classifier may be used. Further, computation of features for an image at multiple scales may be performed in a conventional manner. The example assumes the use of ten features computed using a cell size of 8x8: the components of the color space, *i.e.*, Y (luma component), Cb (blue difference chroma component) and Cr (red difference chroma component), the gradient magnitude, and a histogram of gradients (HOG) for six bins between 0-180 degrees. Thus, assuming a window size of 36x38, $8*16*10 = 1280$ features exist per window. Further, one window exists for each 4x4 block in each scale of an image.

[0020] FIG. 2 is an example illustrating the feature extraction for an image and the arrangement of the resulting features in memory for object detection. As mentioned hereinabove, features for pedestrian detection are computed for the original image (base resolution) and for multiple scales of the image. A sliding window approach (as described hereinabove) is used to compute the features, resulting in set of ten feature channels for each window for each scale. A feature channel contains the computed values of a particular type of feature for a window. For this example, feature channels 0-5 are the bins of the HOG, channel 6 is the gradient magnitudes, and channels 7-9 are the respective color components.

[0021] The feature channels of two sequential windows in a row of the image overlap in all but one position. For example, consider a window A at position x,y and the subsequent window B at position x+4, y. Each feature channel of window B is offset by one from the corresponding feature channel of window A. For example, if feature channel 0 of window A contains eight values, v1, v2, ..., v8, then feature channel 0 of window B contains eight values v2, ..., v8, v9. Similarly, feature channel 0 of window C at position x+8, y contains 8 values v3, ..., v8, v9, v10.

[0022] The feature data for each scale of the image is arranged in memory as shown in FIG. 2. For every fourth row of the image beginning with row 0, ten rows of feature data corresponding to the ten feature channels are “stacked up” in memory such that contiguous values in a row of

feature data corresponding to a row of a scale can be loaded by a vector load instruction. Thus, ten rows of feature data for row 0 of a scale are stored, followed by ten rows of feature data for row 4, followed by ten rows of feature data for row 8, etc.

[0023] The classifier is made up of 1280 binary two-level decision trees, each tree evaluating a feature at each node. A binary decision tree includes nodes and leaves with a binary split at each node as shown in the example of FIG. 3. The nodes are decision points in the tree and the leaves hold partial scores. A collective decision is formed based on partial scores resulting from traversals of multiple decision trees in a window. At each node, a feature value is compared against a threshold. The result of the comparison determines whether the left branch or the right branch out of a node is selected. The feature value input to the decision tree at each node will result in the selection of one of the four leaves.

[0024] FIG. 4 is an example illustrating the general concept of a feature tuple. A feature tuple is a set of features mapped to a decision tree in the classifier. This mapping is formed during the training of the AdaBoost classifier. More specifically, the location in a window of each feature in a tuple is established. Different decision trees may have different tuples as inputs. The number of features in a tuple is equal to the number of nodes in a decision tree.

[0025] FIG. 5 is an example illustrating the general concept of partial scores in the leaves of each decision tree. When a feature tuple is traversed through its respective decision tree, one of the leaves is selected, which contains a partial score for the particular feature tuple. The partial scores are summed and compared to a threshold, also referred to as a minima, exit threshold, or soft cascade threshold. If the minima is observed, the classification process may be terminated (exited) at that point or traversal of the remaining trees may continue to observe additional local minima, if any. The decision regarding exiting at any minima may be application specific. In the example classifier, tree traversal in a given window is terminated when the threshold is crossed during evaluation of the window.

[0026] As mentioned hereinabove, the particular features included in each feature tuple and the mapping of the tuples to decision trees is decided during the training of the AdaBoost classifier. For each node in each tree, the location in a window of the feature to be considered at that node is determined during training. Also, as mentioned hereinabove, the memory storage offsets of corresponding features between two sequential object patches is one. Thus, vectors of features for each feature tuple are available in memory. These feature vectors can be exploited in a

vector processor to perform classification on multiple object patches concurrently. As explained in reference to the method of FIG. 7, N candidate object patches can be evaluated concurrently, where N is the vector width divided by the feature size. For example, if the vector width is 512 and the feature size is 16 bits, then $N = 32$.

[0027] FIG. 6 is an example illustrating feature vectors assuming 32 sequential candidate object patches. In this example, the first ten rows (labeled 0-9) correspond respectively to the ten feature channels described hereinabove. For both decision tree 600 and decision tree 602, the feature values for each node for each object patch are contiguous in memory, which enables a vector load of the 32 values for each node.

[0028] FIG. 7 is a flow diagram of a method for executing a decision tree based AdaBoost classifier on a SIMD processor. For example, the method evaluates N consecutive windows (object patches) concurrently, where N is the SIMD processor vector width divided by the feature size. For purposes of explanation, the feature size is assumed to be 16 bits and N is assumed to be 32. Further, the classifier is assumed to be trained for pedestrian classification. For clarity of explanation, the method is described in reference to examples in FIGS. 8-18.

[0029] The AdaBoost classifier is assumed to be constructed as a linear combination of two level binary decision trees. As described in reference to FIG. 3, and as shown in the example of FIG. 8, a two level binary decision tree has three nodes and four leaves. As shown in the example of FIG. 9, each node logically includes a threshold value T , offset value O , and a feature F . The value of a feature F is fetched from memory at an offset O to be compared against the threshold T . The threshold value, offset, and particular feature for each node of each tree are identified when the classifier is trained, as are the leaf values L for each tree. In some embodiments, the tree parameters, *i.e.*, the threshold values, the offsets, and the leaf values, of each tree of the AdaBoost classifier are stored linearly in memory as illustrated in FIG. 10. The extraction of features of an image that the AdaBoost classifier is to evaluate for the presence of pedestrians and how these features are stored in memory is described hereinabove.

[0030] Referring to the example of FIG. 11, an image is searched for pedestrians using a sliding window approach in which the window size is based on the expected size of a pedestrian. The window positions searched are at an offset of one both vertically and horizontally. Further, the window positions are searched in raster scan order.

[0031] Referring again to the method of FIG. 7, to search 32 consecutive window positions, a

vector of features is loaded 600 for each node of the first decision tree of the classifier. Accordingly, three vectors of features are loaded, one for each node of the tree. The memory address of the vector for each feature is determined by the offset O for the feature in the tree. Due of the way the features are striped in memory, the offset of one between window positions, and the vector load capability of the SIMD processor, features for the 32 consecutive window positions are automatically loaded. FIG. 12 is an example illustrating this step.

[0032] A threshold vector is then generated 702 for each node of the tree, *i.e.*, three threshold vectors are generated. The threshold vector for a node is generated by replicating the threshold value for the node 32 times. Replication in a SIMD processor is an operation of reading one scalar value from memory and populating all elements of a vector with this scalar value.

[0033] Vector compares are then performed 704 between each of the loaded feature vectors and the respective corresponding threshold vectors to generate three mask vectors each of which indicates the result of a respective comparison. Each mask vector contains an indication of whether the comparison was true for each feature value. FIG. 13 is an example illustrating the threshold vectors and the comparison. The naming convention used in this and other figures is: F<node> <position>, where node is the node number of the tree and position is the relative window position, *e.g.*, F23 is the feature evaluated at node 2 of the tree for window position 3. FIG. 14 is an example illustrating the mask vectors resulting from the vector compare operations. In this example, the mask value corresponding to the comparison of each feature to the corresponding threshold is 0x0000 if the comparison is false and 0xFFFF if the comparison is true. Also, other mask values are useful to indicate the results of the compare operations.

[0034] The three mask vectors are then used to select a partial score value, *i.e.*, leaf value, for the tree traversal for each of the 32 object patches and to generate 706 a vector of partial scores in which each entry is the resulting partial score (leaf value) from the tree traversal for a corresponding object patch. In some embodiments, the generation of the vector of partial score values is performed as illustrated in the examples of FIGS. 15-17. As shown in FIG. 15, the three mask vectors M1, M2, and M3, are used to generate four leaf selection mask vectors K1, K2, K3, K4, one for each of the four leaf values of the tree. These leaf selection mask vectors are then used to select leaf values from four leaf vectors, LV1, LV2, LV3, LV4. The leaf vectors for the leaves of the tree, which are shown in FIG. 16, are generated by replicating each leaf value L1, L2, L3, and L4 in the corresponding vector 32 times.

[0035] The mask vectors M1, M2, M3 are logically combined as illustrated in FIG. 15 to generate the leaf selection mask vectors K1, K2, K3, K4. The logic is as follows: 1) when corresponding locations in M1 and M2 are true, then set the corresponding location in leaf selection mask vector K1 to select the corresponding location in leaf vector LV1; 2) when a location in M1 is true and the corresponding location in M2 is false, then set the corresponding location in leaf selection mask vector K2 to select the corresponding location in leaf vector LV2; 3) when a location in M1 is false and the corresponding location in M3 is false, then set the corresponding location in leaf selection mask vector K3 to select the corresponding location in leaf vector LV3; and 4) when corresponding locations in M1 and M3 are false, then set the corresponding location in leaf selection mask vector K4 to select the corresponding location in leaf vector LV4. Any locations in the leaf selection mask vectors not set to select a leaf value by the logical mask combinations are set to not select a leaf value. For this example, a selection value in a leaf selection mask vector is 0x0000 if the corresponding leaf value in the corresponding leaf vector is not to be selected and to 0xFFFF if the corresponding leaf value is to be selected. Also, other mask values are useful. A leaf selection mask vector is a logical combination of the mask vectors for the nodes in the traversal path of the decision tree that reaches the corresponding leaf.

[0036] The resulting leaf selection mask vectors K1, K2, K3, K4 are illustrated in FIG. 16. A logical and operation is performed between each leaf vector and the corresponding leaf selection mask vector to select leaf values (partial scores) from each leaf vector. The result of these four logical and operations is four vectors of leaf values as illustrated in FIG. 17. The four vectors are combined with logical or operations to generate the partial score vector in which each entry is the result of evaluating the decision tree for corresponding ones of the 32 window positions.

[0037] Referring again to FIG. 7, the partial score vector is accumulated 708 into an accumulated score vector having one entry for each of the 32 windows. For example, the partial score vector may be added to the accumulated score vector, which stores the sum of any previous partial scores from previous tree traversals. As explained in more detail hereinbelow, in some embodiments, an exit mask vector may be applied to the partial score vector before accumulating the partial scores to mask out partial scores for window positions that have met the criteria to terminate classification.

[0038] The accumulated score vector is compared 710 to an exit threshold vector and any

accumulated partial score values below the exit threshold are saved as final scores. If an accumulated partial score value for a given window position is below the exit threshold, then tree evaluation, *i.e.*, classification, for that window is complete. If the classification process for all 32 windows is complete 712, *i.e.*, all accumulated scores are below the exit threshold, or all trees have been traversed 714, then the classification process for the 32 windows is terminated and the final accumulated score vector is returned 716. Otherwise, the classification continues 700 with the next tree in the classifier. The order in which the trees are traversed is determined during the training process.

[0039] In some embodiments, because classification may not be complete for all of the 32 window positions, an exit mask vector is maintained that indicates which of the window positions has completed the classification process and which have not. The generation and use of the exit mask vector is illustrated in the example of FIG. 18. The vector comparison operation of the accumulated score vector to the exit threshold vector results in an exit mask vector that indicates which of the accumulated scores meets the exit criteria and which do not. For the next iteration of classification, a logical and operation of the inverse of the exit mask vector (“~” is bit invert) and the partial score vector is performed to mask out partial scores for any window positions that have previously exited. The resulting partial score vector is then added to the accumulated score vector, the comparison to the exit threshold vector is performed, and an updated exit mask is generated.

[0040] FIG. 19 is a simplified block diagram of an example digital system 1900 configured to execute an embodiment of the method of FIG. 7. In some embodiments, the digital system may be an integrated circuit, *i.e.*, a system-on-a-chip. For simplicity of explanation, pedestrian classification as used in the description of other figures is assumed. The digital system 1900 includes a master processor 1902, a camera 1904, an image signal processor (ISP) 1906, a feature extraction component 1908, a SIMD instruction set digital signal processor (DSP) 1910, and a shared memory 1912. The master processor 1902 controls the operation of the other components to perform operations needed for pedestrian classification in scenes captured by the camera 1904. The master processor 1902 may be any suitable processor, such as central processing units available from ARM Ltd.

[0041] The camera 1904 captures images of a scene and provides those images to the ISP 1906. The ISP 1906 performs image processing on each image to prepare the image for feature

extraction. For example, the ISP 1906 may perform operations on the images, such as white balancing, black level adjustment, noise filtering, conversion from RGB to YCbCr, and edge enhancement.

[0042] The feature extraction component 1908 performs feature extraction on images from the ISP 1906. Feature extraction is described hereinabove. The extracted features are stored in shared memory 1912 for use in the method.

[0043] The shared memory component 1912 may be on-chip memory, external memory, or a combination thereof. Any suitable memory design may be used. For example, the memory component 1912 may include static random access memory (SRAM), dynamic random access memory (DRAM), synchronous DRAM (SDRAM), read-only memory (ROM), flash memory, or a combination thereof.

[0044] Further, the memory component 1912 stores software instructions for the AdaBoost classifier 1916 that include software instructions to perform an embodiment of the method of FIG. 6. The memory component also stores the features 1914 computed by the feature extraction component 1908, and the decision trees 1918 used by the classifier 1916. Some or all of the software instructions and decisions trees may be initially stored in a computer-readable medium such as a compact disc (CD), a diskette, a tape, a file, memory, or any other computer readable storage device and loaded and stored on the digital system 1900. In some cases, the software instructions may also be sold in a computer program product, which includes the computer-readable medium and packaging materials for the computer-readable medium. In some cases, the software instructions may be distributed to the digital system 1900 via removable computer readable media (e.g., floppy disk, optical disk, flash memory, USB key), via a transmission path from computer readable media on another computer system (e.g., a server).

[0045] The DSP 1910 executes the software instructions of the classifier 1916 to perform pedestrian classification using the extracted features 1914. The DSP implements a SIMD instruction set providing at least vector load operations, vector compare operations, vector addition and subtraction operations, vector logical operations, and replication operations. Any suitable DSP with an appropriate SIMD instruction set may be used. One such DSP is described in reference to FIG. 20. The results of the classification are communicated to the master processor 1902 for further processing, such as pedestrian detection based on the classification results and decision making based on the results of the pedestrian detection.

[0046] FIG. 20 is a high level block diagram of an example SIMD digital signal processor (DSP) 2000 suitable for executing one or more embodiments of the method of FIG. 7. The illustrated DSP is the TMS32C66x DSP available from Texas Instruments, Inc. The C66x DSP 2000 includes eight functional units, two register files, and two data paths. The two general-purpose register files each contain thirty-two 32-bit registers for a total of 64 registers. The general-purpose registers are useful for data or can be data address pointers. The data types supported include packed 8-bit data, packed 16-bit data, 32-bit data, 40-bit data, and 64-bit data. The C66x DSP 2000 supports up to 4 way SIMD operations for 16 bit data and up to 8 way SIMD operations for 8 bit data. Thus, the SIMD width for each data path is 64 bits, other than for some multiply operations which can handle up to 128 bits of packed data. A detailed description of the C66x and instruction set is located in “TMS320C66x DSP CPU and Instruction Set Reference Guide,” SPRUGH7, November 2010, which is incorporated by reference herein.

[0047] As mentioned hereinabove, the method of FIG. 7 evaluates N windows (object patches) concurrently, where N is the SIMD width divided by the feature size. Thus, for the DSP 2000, if the feature size in a method embodiment is 8 bits, $N = 8$, and if the feature size in a method embodiment is 16 bits, $N = 4$.

Other Embodiments

[0048] For example, embodiments have been described in which the decision trees are assumed to be two level binary decision trees. Also, the decision trees may include more than two levels and/or are not required to be binary.

[0049] In another example, embodiments have been described in which the tree traversal for an object patch is terminated when the accumulated partial score for the object patch falls below an exit threshold. Also, rather than terminating tree traversal for such an object patch, traversal of the remaining trees may be continued to observe additional local minima, if any.

[0050] In another example, embodiments have been described assuming that the classifier is performing pedestrian classification. Other embodiments are possible for other types of object classification in an image, such as traffic signs, vehicles, cyclists and animals.

[0051] One or more of the steps shown in the figures and described herein may be performed concurrently, may be combined, and/or may be performed in a different order than the order shown in the figures and/or described herein. Accordingly, embodiments are not limited to the

specific ordering of steps shown in the figures and/or described herein.

[0052] Components may be referred to by different names and/or may be combined in ways not shown herein without departing from the described functionality. Also, for example, if a first device couples to a second device, that connection may be through a direct electrical connection, through an indirect electrical connection via other devices and connections, through an optical electrical connection, and/or through a wireless electrical connection.

[0053] Modifications are possible in the described embodiments, and other embodiments are possible, within the scope of the claims.

CLAIMS

What is claimed is:

1. A method for object classification in a decision tree based adaptive boosting (AdaBoost) classifier implemented on a single-instruction multiple-data (SIMD) processor, the method comprising:

receiving feature vectors extracted from N consecutive window positions in an image in a memory coupled to the SIMD processor, wherein N is a vector width of the SIMD processor divided by a bit size of a feature, and wherein a feature vector includes N feature values, one feature value for each of the N consecutive window positions; and

evaluating the N consecutive window positions concurrently by the AdaBoost classifier using the feature vectors and vector instructions of the SIMD processor, wherein the AdaBoost classifier concurrently traverses decision trees for the N consecutive window positions until classification is complete for the N consecutive window positions, wherein a decision tree includes a plurality of nodes, a threshold value for each node, and a plurality of leaves, each leaf including a partial score.

2. The method of claim 1, wherein evaluating the N consecutive window positions includes:

loading a plurality of the feature vectors using a vector load instruction of the SIMD processor, wherein one feature vector is loaded for each node of a single decision tree of the AdaBoost classifier;

comparing each feature vector to a corresponding threshold vector using a vector compare instruction of the SIMD processor to generate a mask vector for each node, wherein the corresponding threshold vector includes N copies of the threshold value for the node corresponding to the feature vector, and wherein the mask vector includes N comparison results, one for each of the N features of the feature vector;

generating a partial score vector based on the mask vectors and the partial score values of the leaves of the decision tree, the partial score vector including N partial score values, one for each of the N consecutive window positions;

accumulating the partial scores into an accumulated score vector, the accumulated score vector including N accumulated score values, one for each of the N consecutive window positions; and

comparing the accumulated score vector to an exit threshold vector using a vector

compare instruction of the SIMD processor to determine whether object classification can be terminated for one or more of the N consecutive window positions.

3. The method of claim 2, wherein generating a partial score vector includes:

generating a leaf selection mask vector for each of the leaves of the decision tree based on the mask vectors, wherein a leaf selection mask vector is a logical combination of mask vectors for nodes in a traversal path of the single decision tree that reaches the leaf corresponding to the leaf selection mask vector; and

performing a logical and operation of each leaf selection mask vector with a corresponding leaf vector to select partial scores for each of the N window positions from the leaf vectors, wherein a corresponding leaf vector includes N copies of a partial score of the leaf.

4. The method of claim 1, wherein the decision trees are two-level binary decision trees.

5. The method of claim 1, wherein the AdaBoost classifier is trained for pedestrian classification.

6. The method of claim 1, wherein the SIMD processor is a digital signal processor.

7. A digital system comprising:

a single-instruction multiple-data (SIMD) processor;

a memory component coupled to the SIMD processor, the memory component configured to store features extracted from an image;

a plurality of decision trees stored in the memory component, wherein each decision tree includes a plurality of nodes, a threshold value for each node, and a plurality of leaves, each leaf including a partial score; and

a decision tree based adaptive boosting (AdaBoost) classifier trained for object classification stored in the memory component, the AdaBoost classifier executable on the SIMD processor, wherein the AdaBoost classifier uses the plurality of decision trees for object classification, the AdaBoost classifier configured to evaluate N consecutive window positions concurrently using the features and vector instructions of the SIMD processor, wherein the AdaBoost classifier concurrently traverses decision trees for the N consecutive window positions until classification is complete for the N consecutive window positions, and wherein N is a vector width of the SIMD processor divided by a bit size of a feature.

8. The digital system of claim 7, including a feature extraction component coupled to the memory component and configured to extract the features from the N consecutive window

positions in an image.

9. The digital system of claim 8, including a camera coupled to the feature extraction component to provide the image.

10. The digital system of claim 7, wherein the AdaBoost classifier is configured to evaluate the N consecutive window positions by

loading a plurality of feature vectors from the memory component using a vector load instruction of the SIMD processor, wherein one feature vector is loaded for each node of a single decision tree of the plurality of decision trees, and wherein a feature vector includes N feature values, one feature value for each of the N consecutive window positions;

comparing each feature vector to a corresponding threshold vector using a vector compare instruction of the SIMD processor to generate a mask vector for each node, wherein the corresponding threshold vector includes N copies of the threshold value for the node corresponding to the feature vector, and wherein the mask vector includes N comparison results, one for each of the N features of the feature vector;

generating a partial score vector based on the mask vectors and the partial score values of the leaves of the decision tree, the partial score vector including N partial score values, one for each of the N consecutive window positions;

accumulating the partial scores into an accumulated score vector, the accumulated score vector including N accumulated score values, one for each of the N consecutive window positions; and

comparing the accumulated score vector to an exit threshold vector using a vector compare instruction of the SIMD processor to determine whether object classification can be terminated for one or more of the N consecutive window positions.

11. The digital system of claim 10, wherein generating a partial score vector includes:

generating a leaf selection mask vector for each of the leaves of the decision tree based on the mask vectors, wherein a leaf selection mask vector is a logical combination of mask vectors for nodes in a traversal path of the single decision tree that reaches the leaf corresponding to the leaf selection mask vector; and

performing a logical and operation of each leaf selection mask vector with a corresponding leaf vector to select partial scores for each of the N window positions from the leaf vectors, wherein a corresponding leaf vector includes N copies of a partial score of the leaf.

12. The digital system of claim 7, wherein the decision trees are two level binary decision trees.

13. The digital system of claim 7, wherein the AdaBoost classifier is trained for pedestrian classification.

14. The digital system of claim 7, wherein the SIMD processor is a digital signal processor.

15. A non-transitory computer readable medium storing software instructions that, when executed on a single-instruction multiple-data (SIMD) processor, cause a method for object classification in a decision tree based adaptive boosting (AdaBoost) classifier to be executed, the method comprising:

receiving feature vectors extracted from N consecutive window positions in an image in a memory coupled to the SIMD processor, wherein N is a vector width of the SIMD processor divided by a bit size of a feature, and wherein a feature vector includes N feature values, one feature value for each of the N consecutive window positions; and

evaluating the N consecutive window positions concurrently by the AdaBoost classifier using the feature vectors and vector instructions of the SIMD processor, wherein the AdaBoost classifier concurrently traverses decision trees for the N consecutive window positions until classification is complete for the N consecutive window positions, wherein a decision tree includes a plurality of nodes, a threshold value for each node, and a plurality of leaves, each leaf including a partial score.

16. The computer readable of claim 15, wherein evaluating the N consecutive window positions includes:

loading a plurality of the feature vectors using a vector load instruction of the SIMD processor, wherein one feature vector is loaded for each node of a single decision tree of the AdaBoost classifier;

comparing each feature vector to a corresponding threshold vector using a vector compare instruction of the SIMD processor to generate a mask vector for each node, wherein the corresponding threshold vector includes N copies of the threshold value for the node corresponding to the feature vector, and wherein the mask vector includes N comparison results, one for each of the N features of the feature vector;

generating a partial score vector based on the mask vectors and the partial score values of the leaves of the decision tree, the partial score vector including N partial score values, one for

each of the N consecutive window positions;

accumulating the partial scores into an accumulated score vector, the accumulated score vector including N accumulated score values, one for each of the N consecutive window positions; and

comparing the accumulated score vector to an exit threshold vector using a vector compare instruction of the SIMD processor to determine whether object classification can be terminated for one or more of the N consecutive window positions.

17. The computer readable medium of claim 16, wherein generating a partial score vector includes:

generating a leaf selection mask vector for each of the leaves of the decision tree based on the mask vectors, wherein a leaf selection mask vector is a logical combination of mask vectors for nodes in a traversal path of the single decision tree that reaches the leaf corresponding to the leaf selection mask vector; and

performing a logical and operation of each leaf selection mask vector with a corresponding leaf vector to select partial scores for each of the N window positions from the leaf vectors, wherein a corresponding leaf vector includes N copies of a partial score of the leaf.

18. The computer readable medium of claim 15, wherein the decision trees are two-level binary decision trees.

19. The computer readable medium of claim 15, wherein the AdaBoost classifier is trained for pedestrian classification.

20. The computer readable medium of claim 15, wherein the SIMD processor is a digital signal processor.

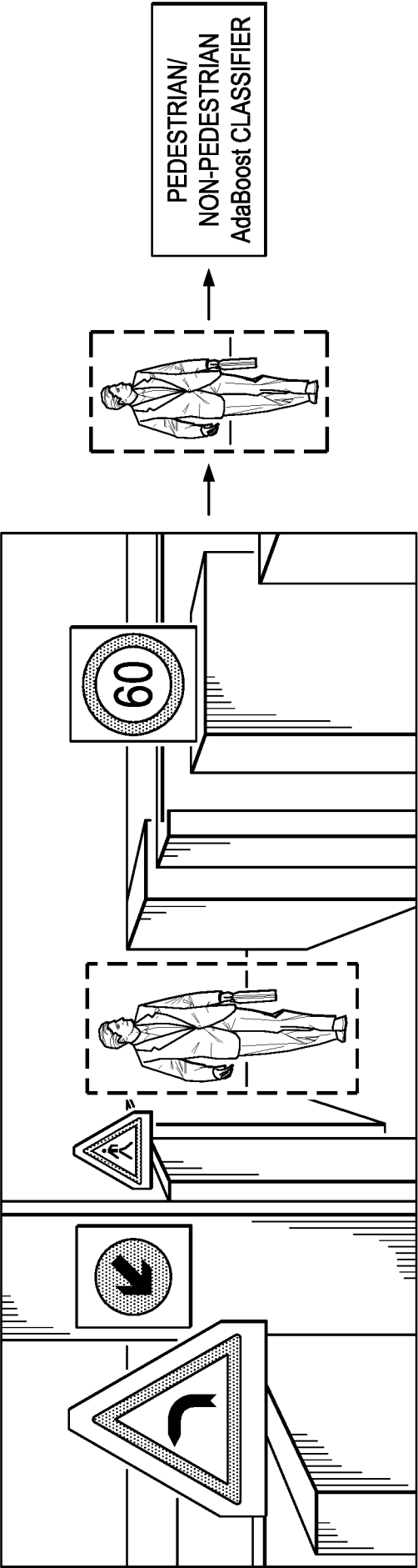


FIG. 1

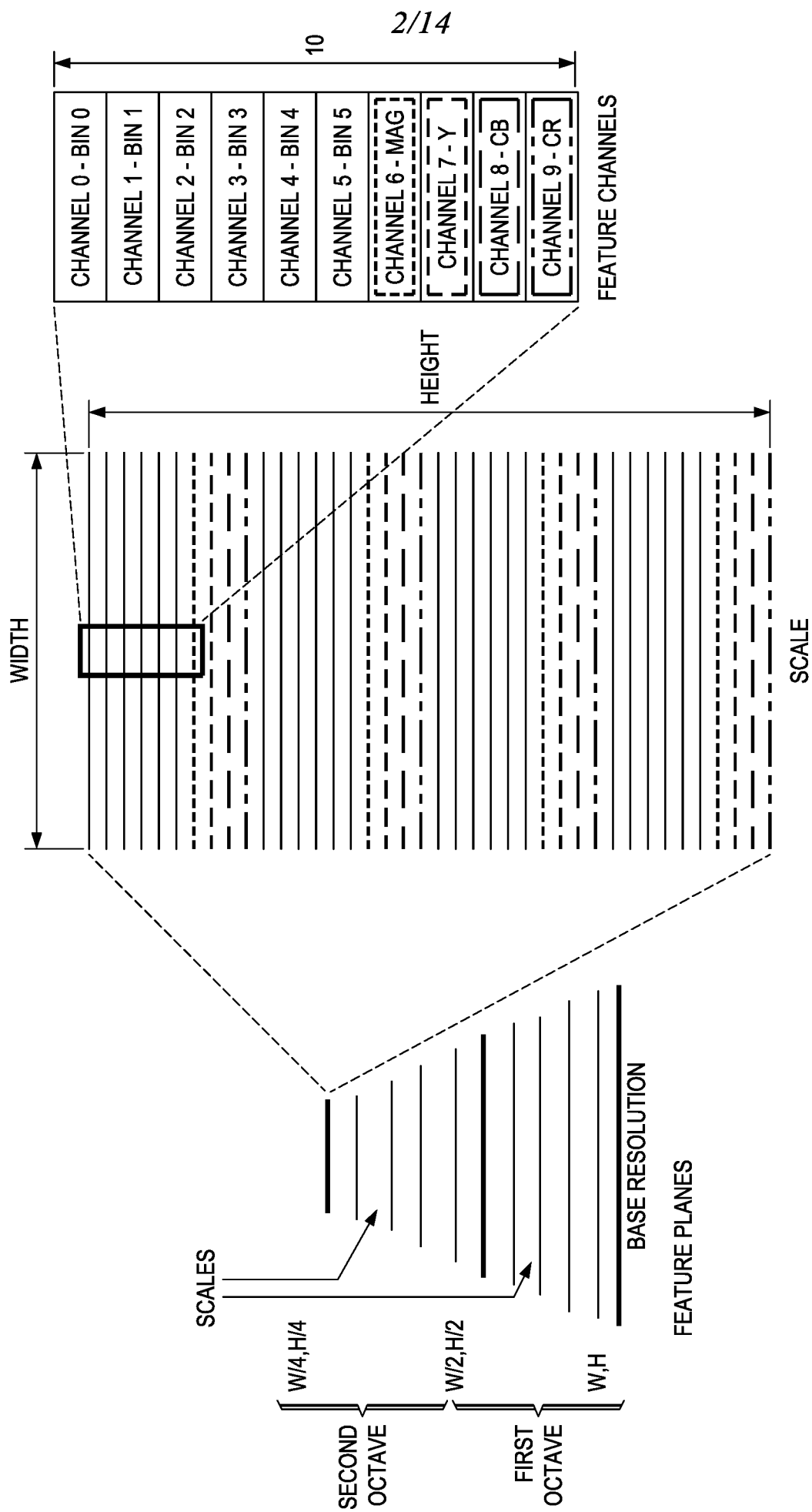


FIG. 2

3/14

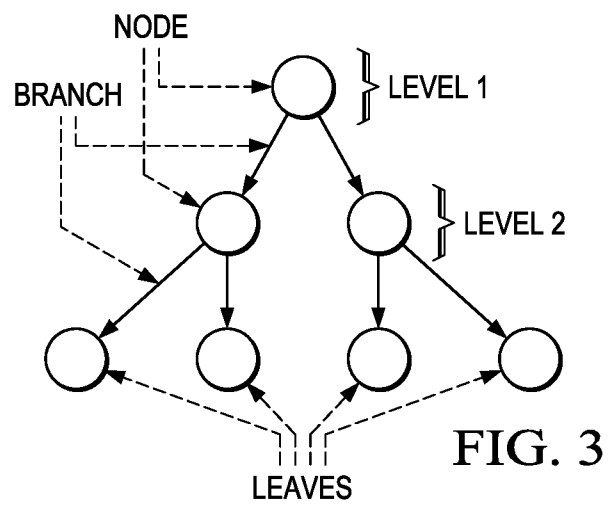


FIG. 3

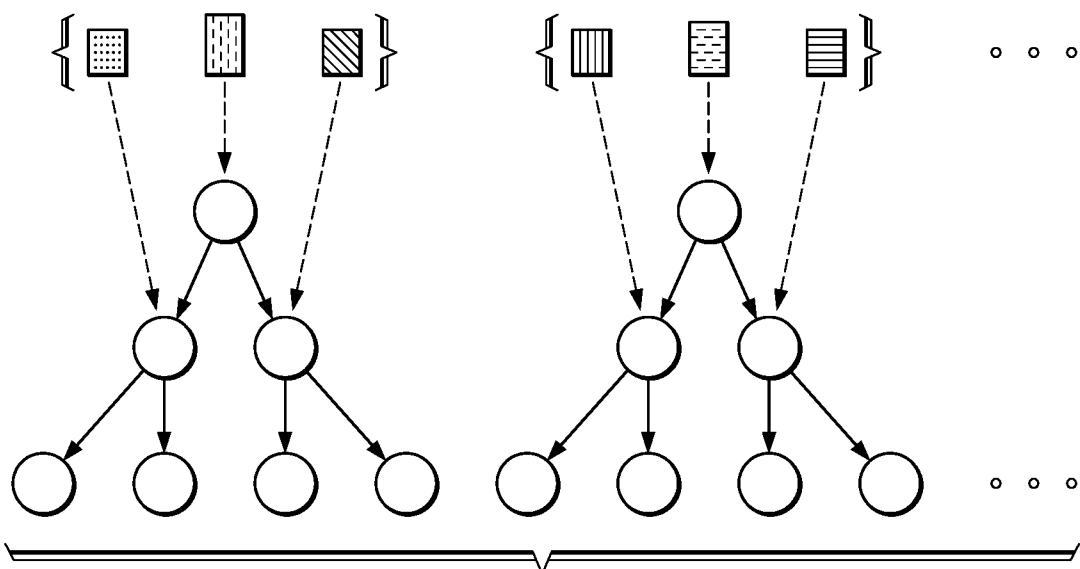


FIG. 4

4/14

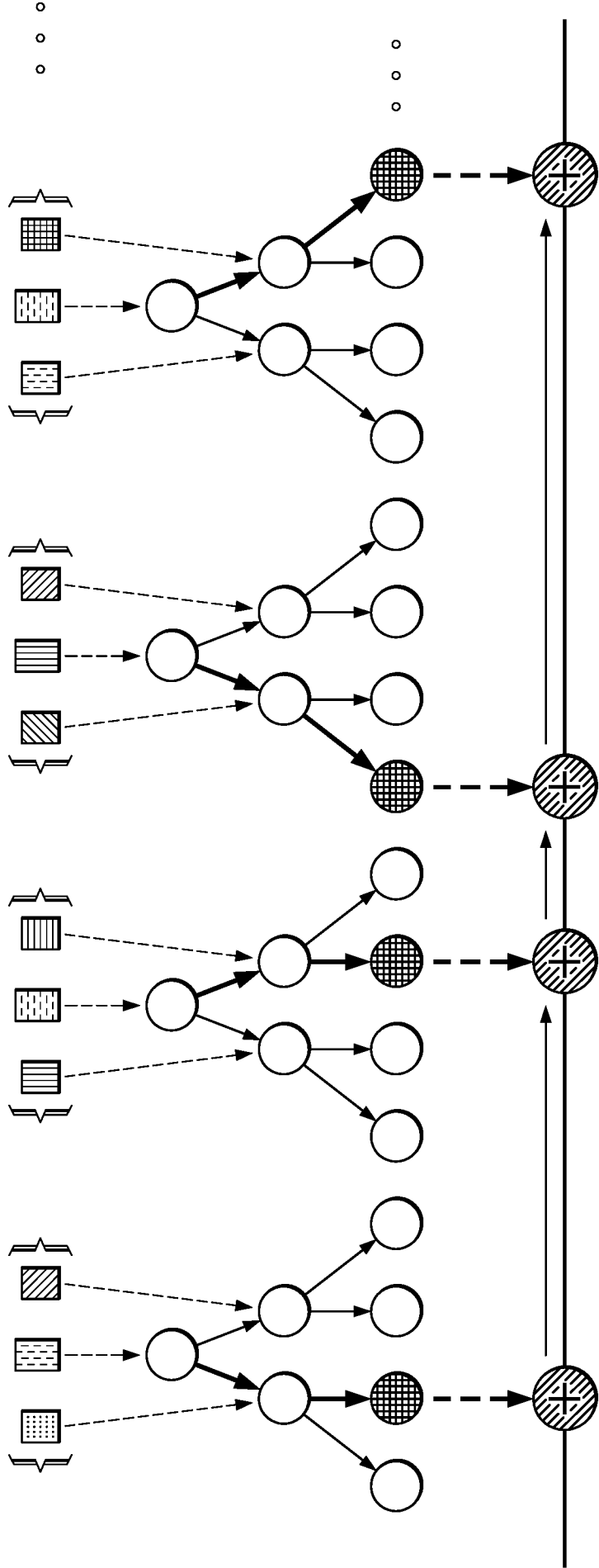


FIG. 5

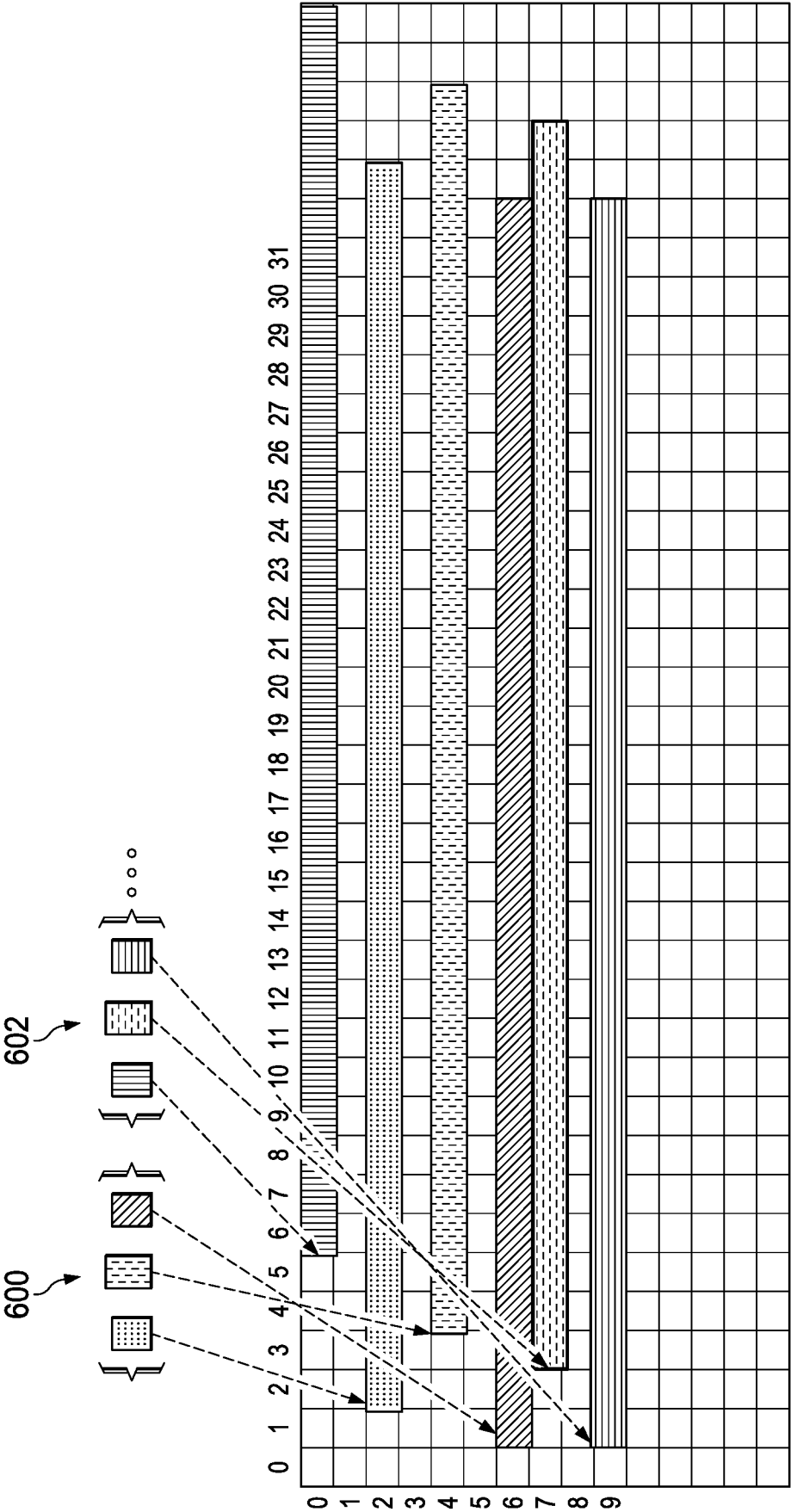


FIG. 6

6/14

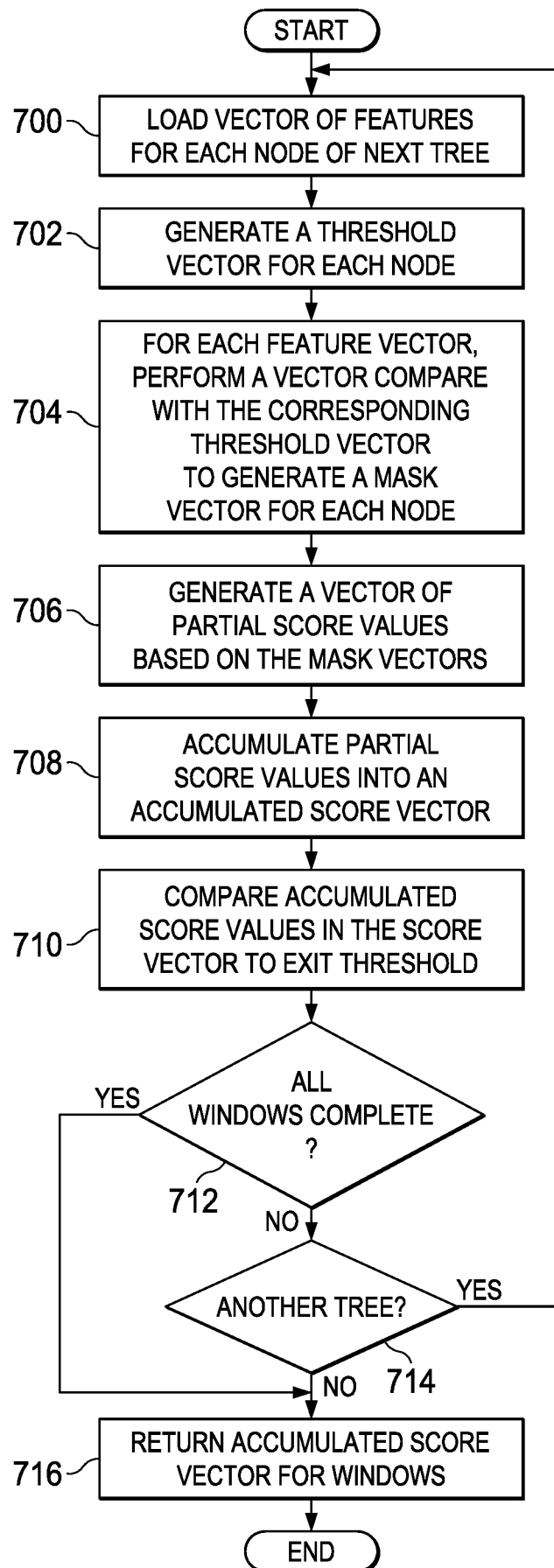


FIG. 7

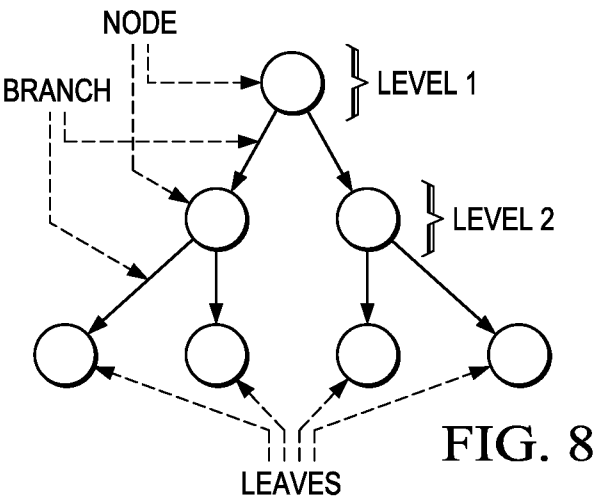


FIG. 8

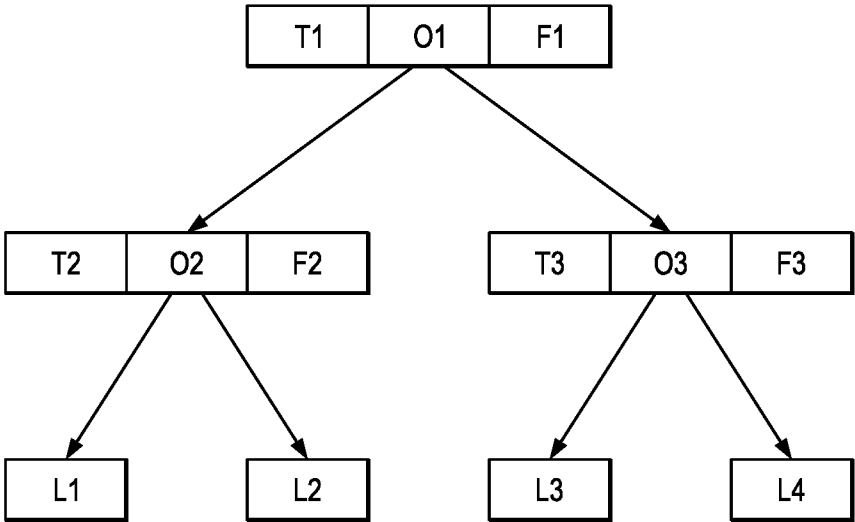


FIG. 9

8/14

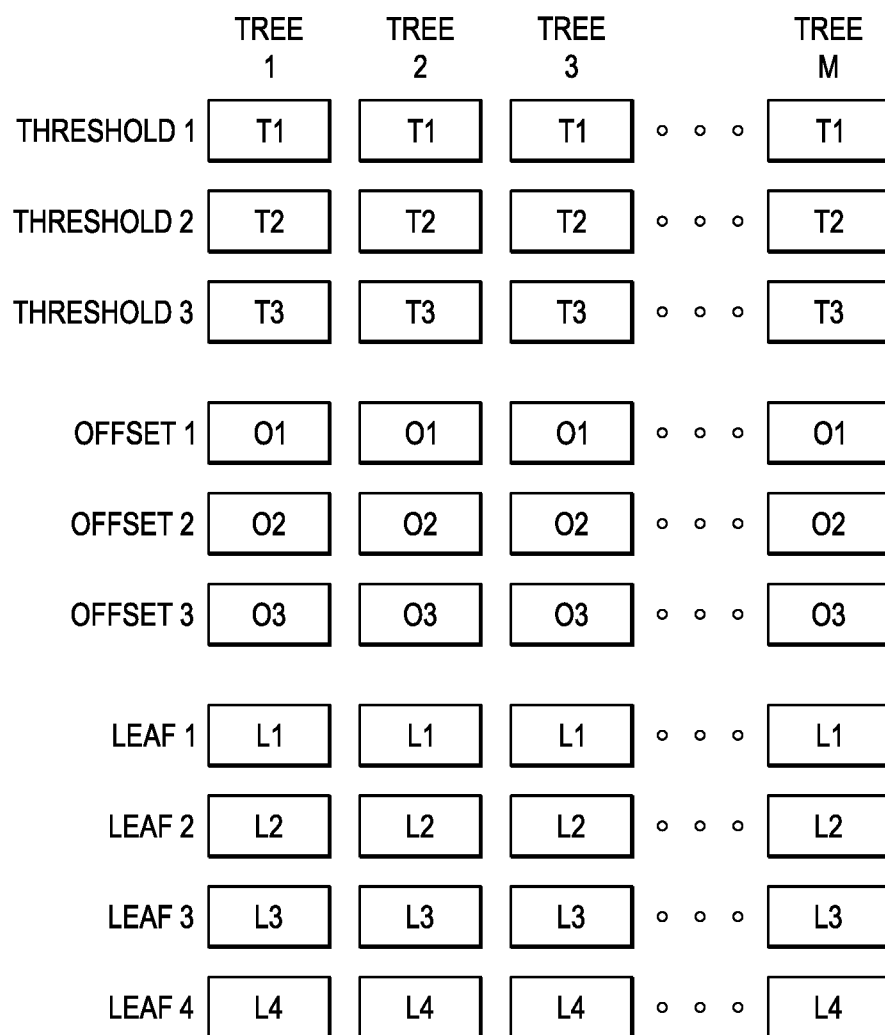


FIG. 10

9/14

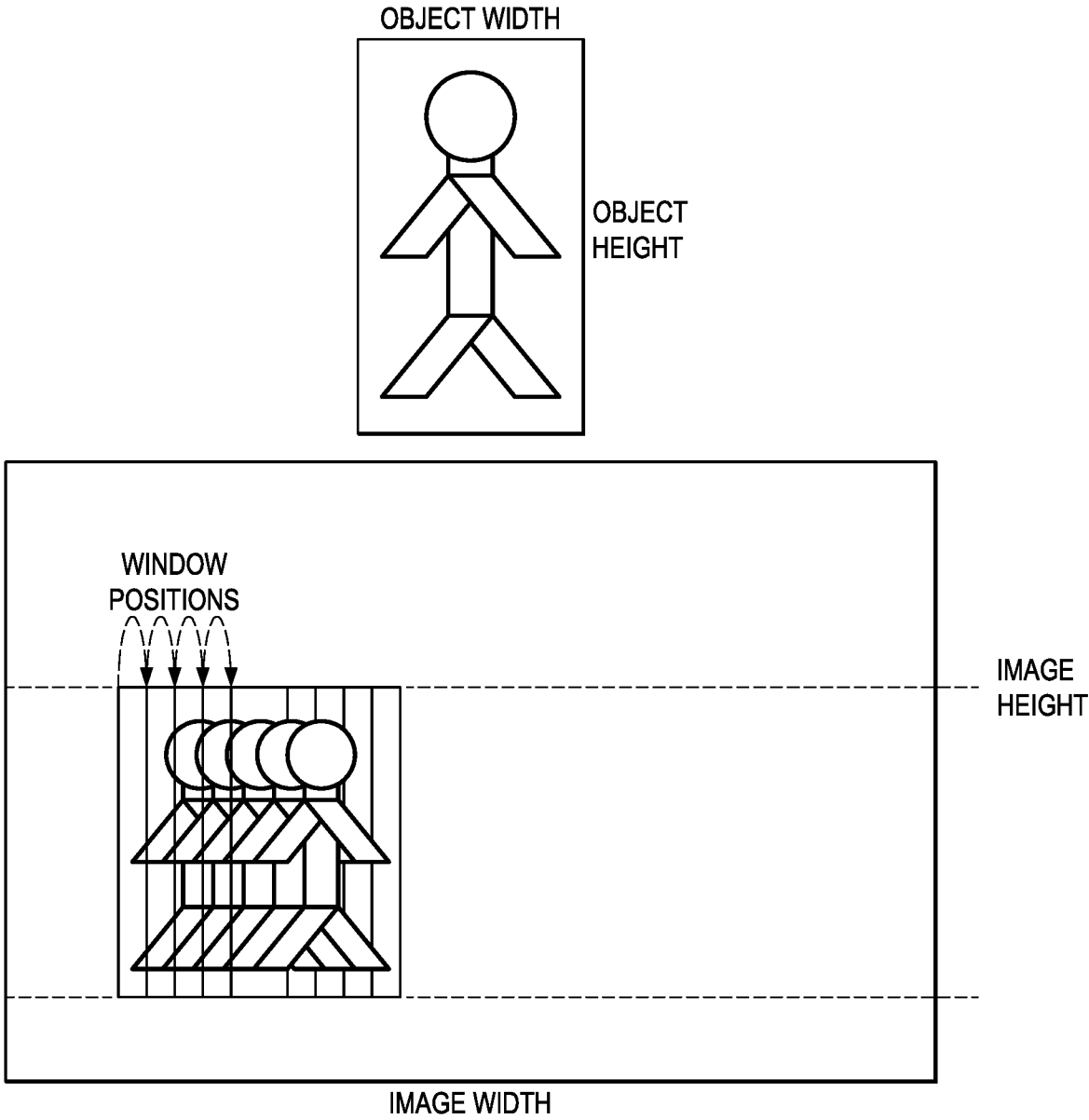


FIG. 11

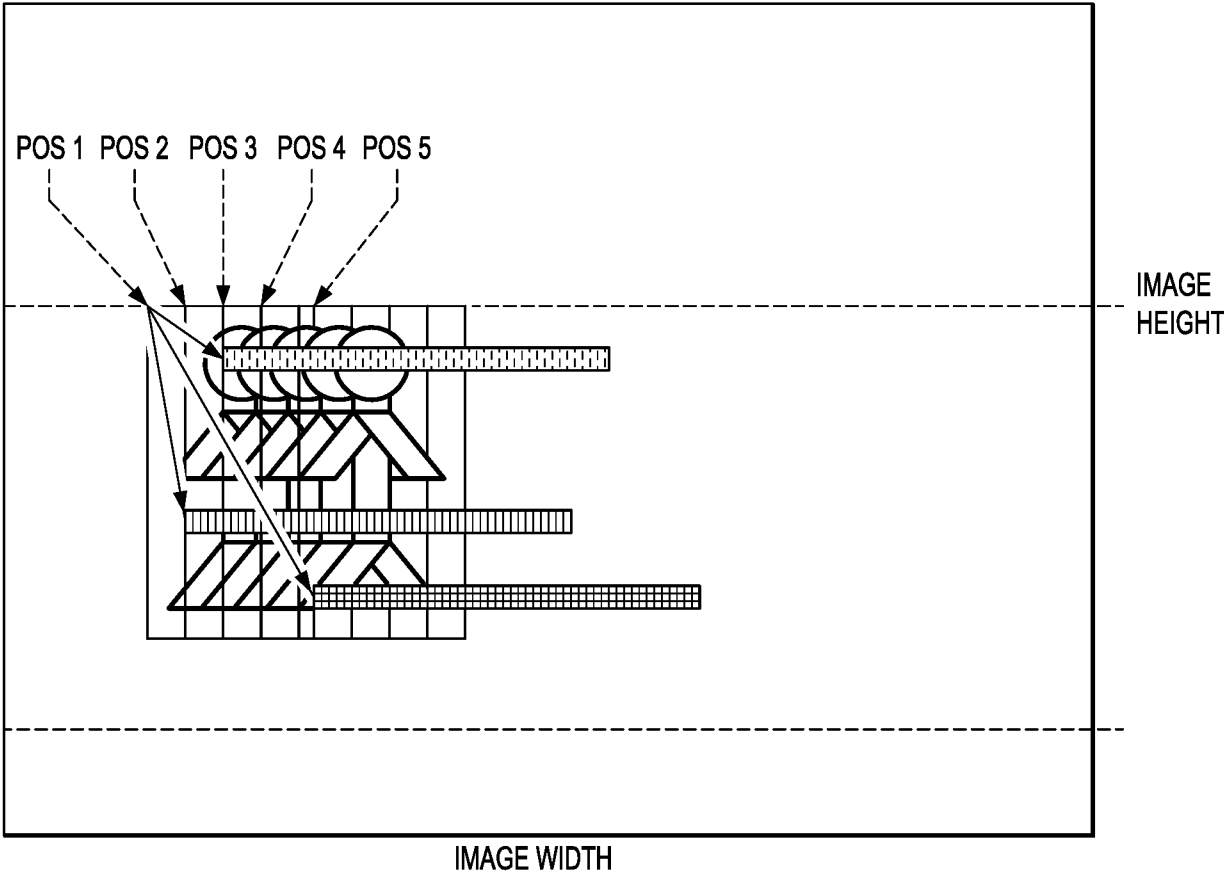
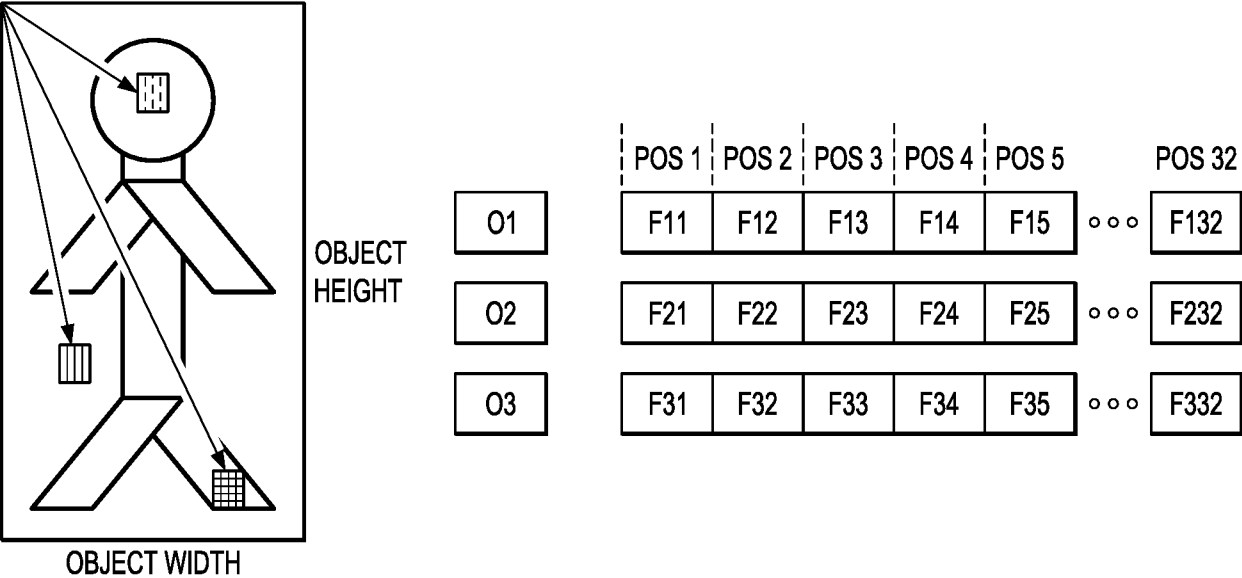
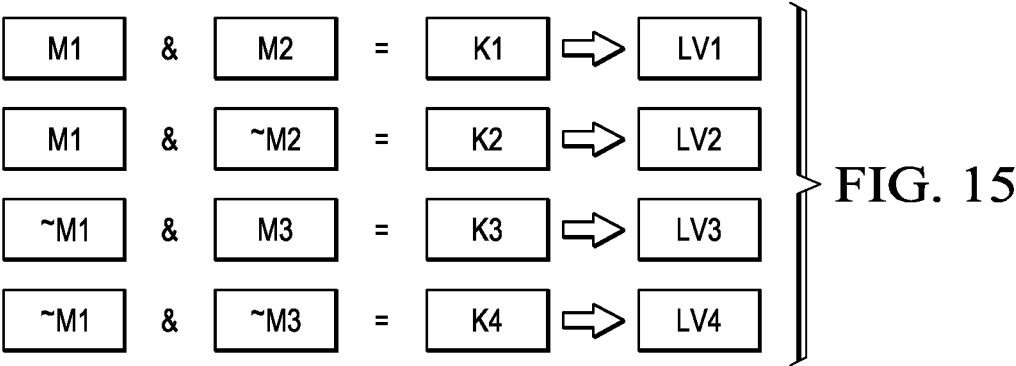
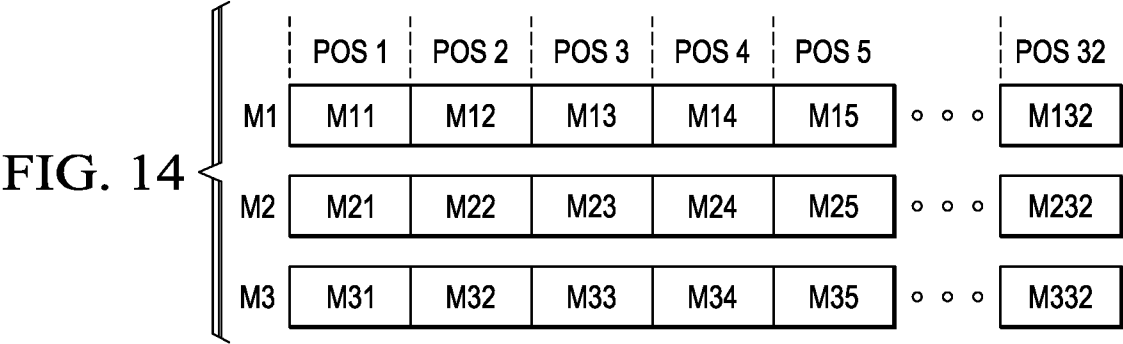
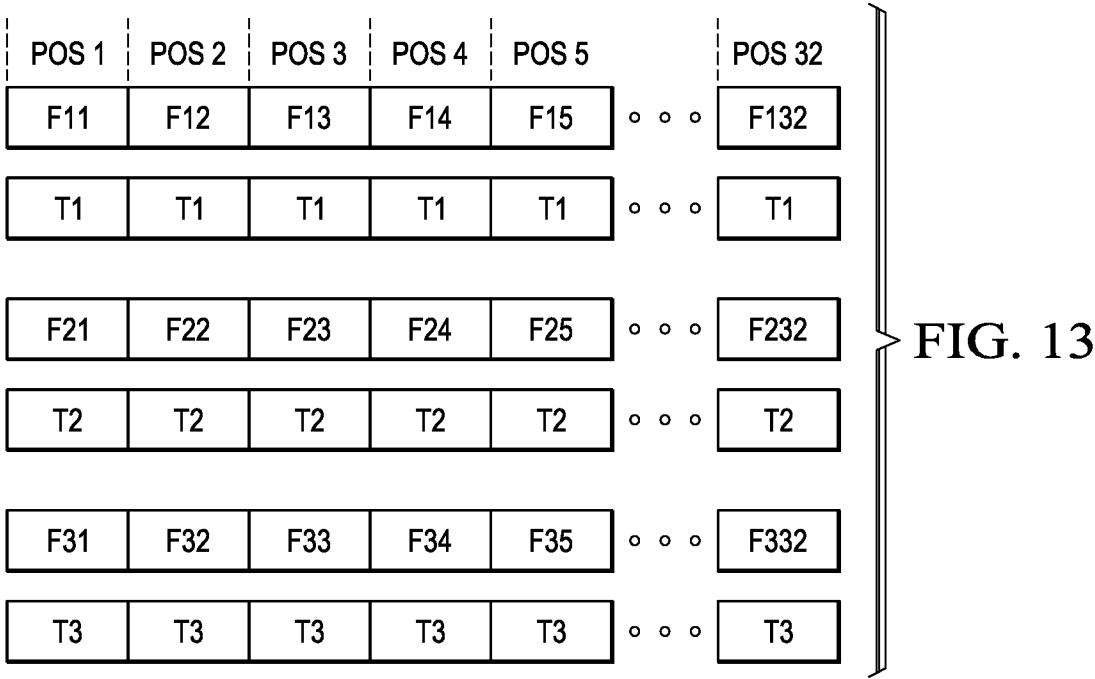
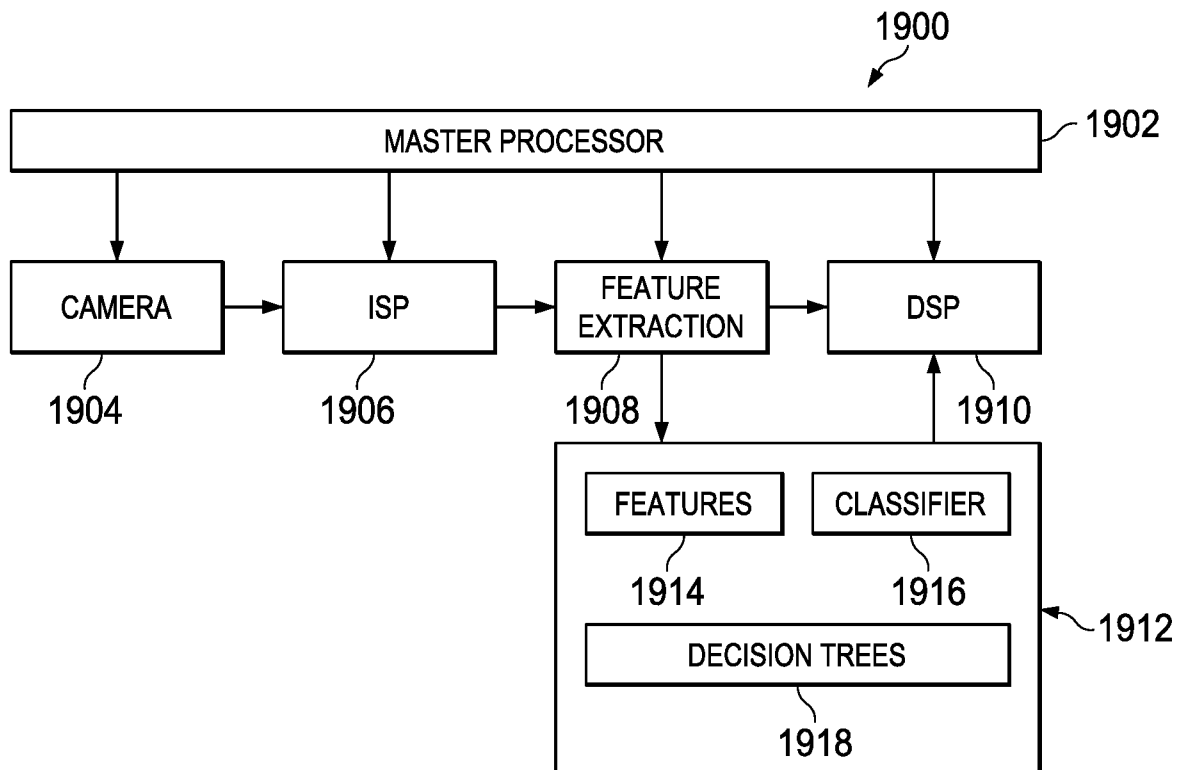
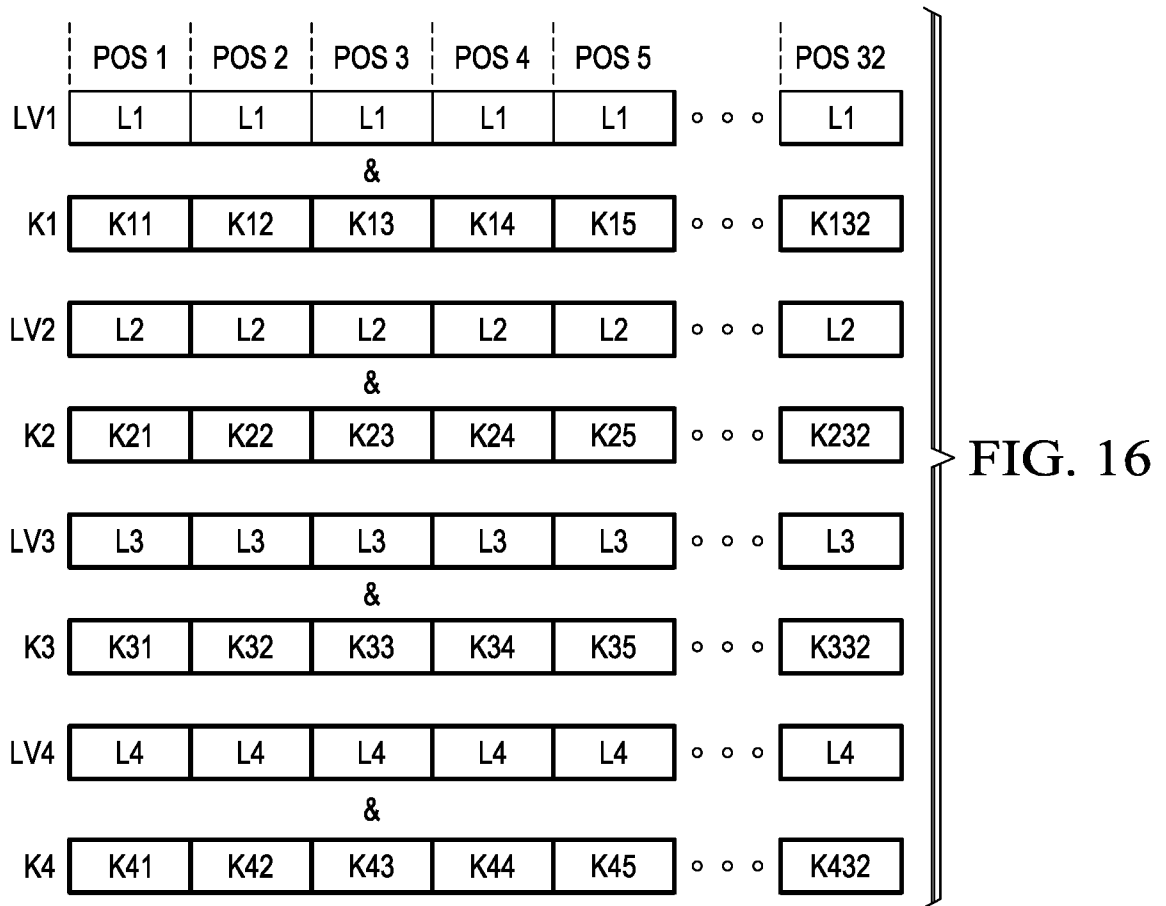


FIG. 12



12/14



13/14

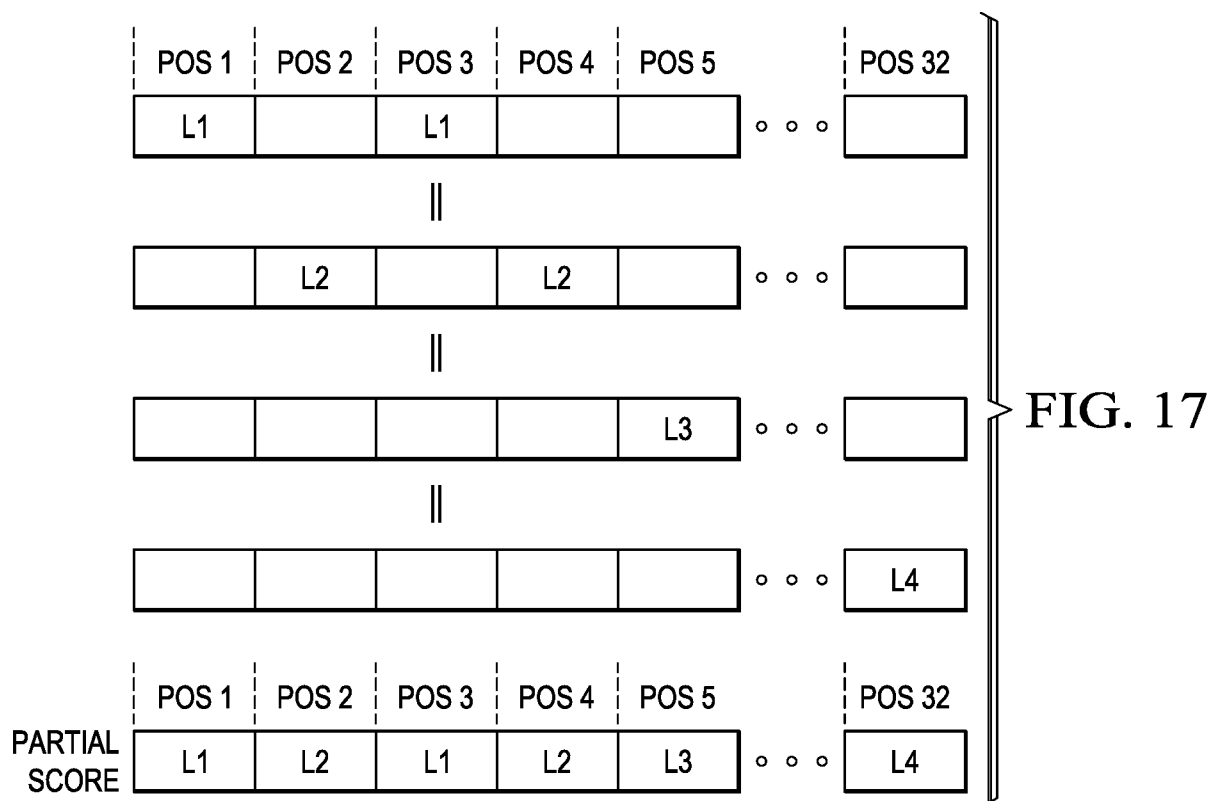


FIG. 17

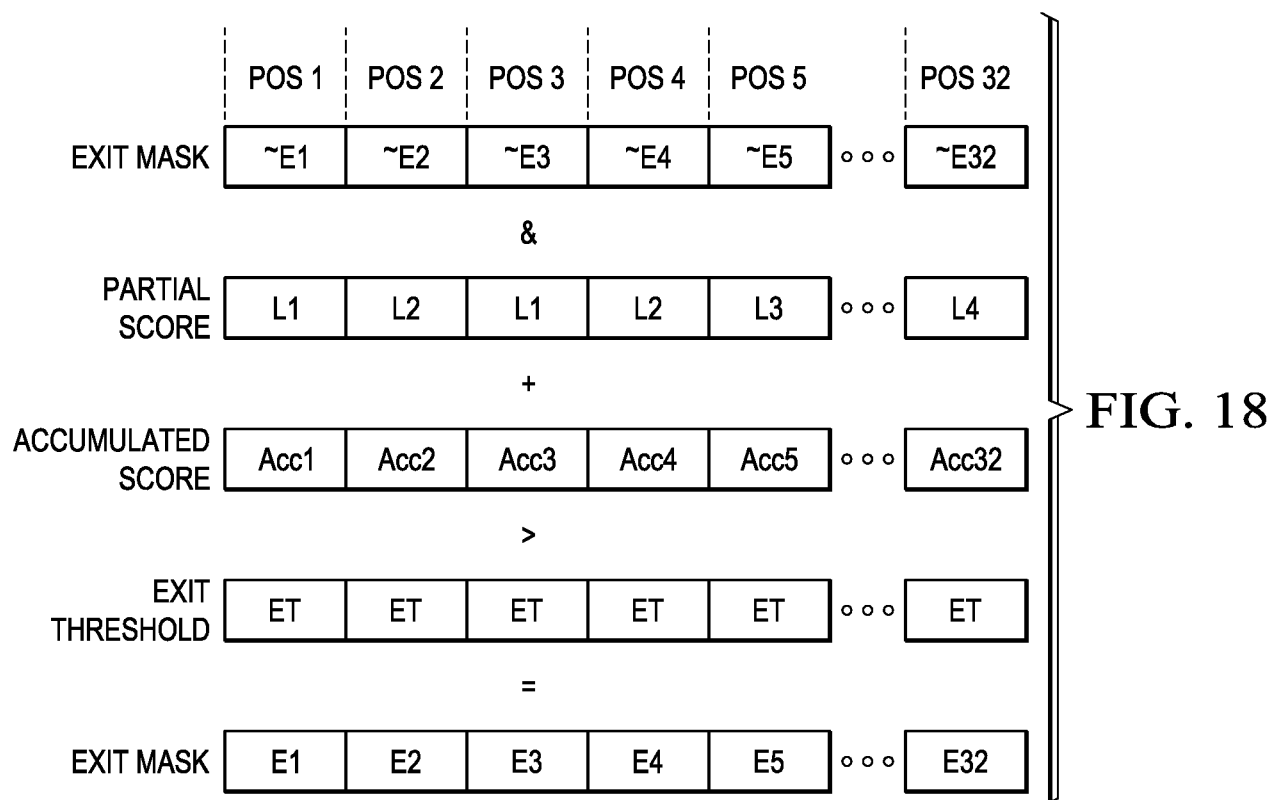


FIG. 18

14/14

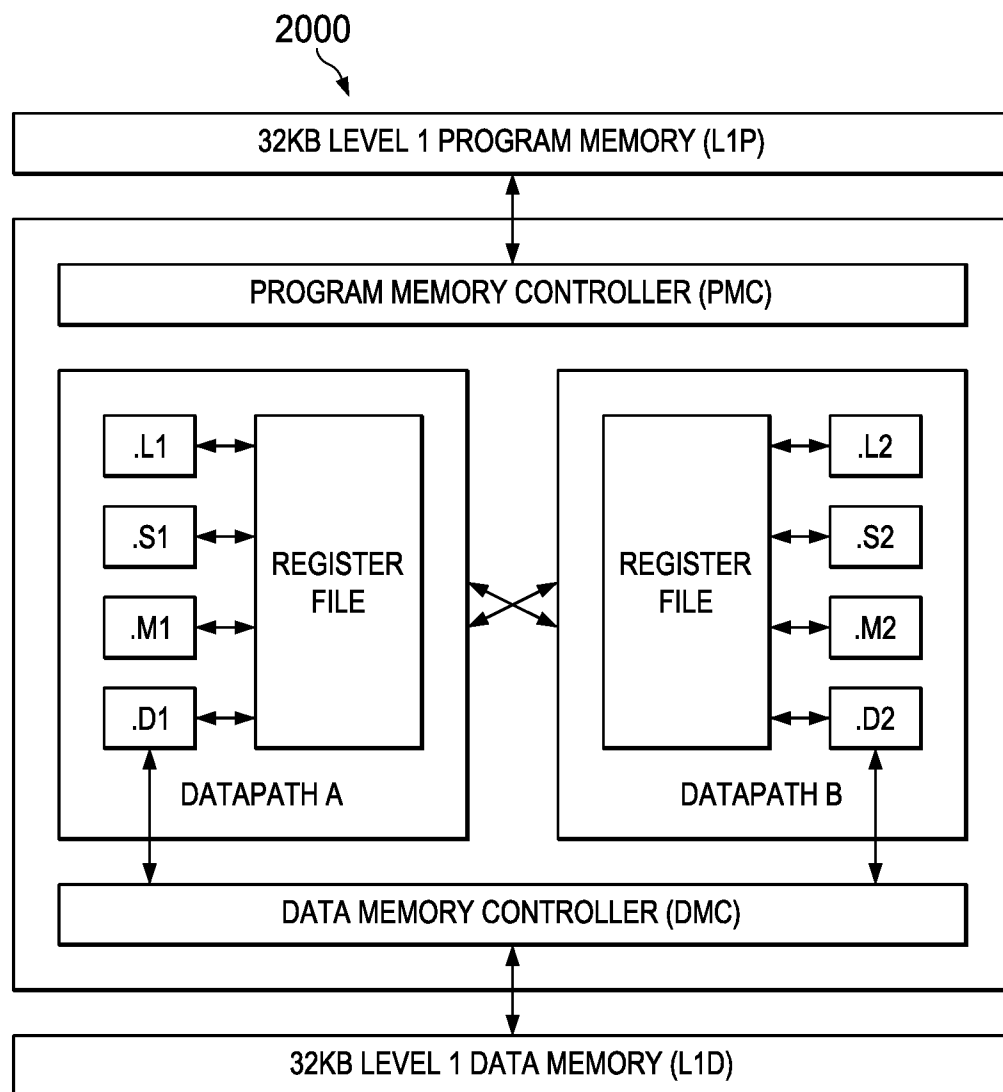


FIG. 20

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 2016/041159

A. CLASSIFICATION OF SUBJECT MATTER														
G06K 9/62 (200.01) G06F 15/173 (2006.01) 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)														
G06K 9/00, 9/62, G06F 15/00, 15/16, 15/173, 15/80														
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched														
Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)														
PatSearch (RUPTO internal), USPTO, PAJ, Esp@cenet, DWPI, EAPATIS, PATENTSCOPE														
C. DOCUMENTS CONSIDERED TO BE RELEVANT														
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.												
A	US 8923585 B1 (GIVEN IMAGING LTD.) 30.12.2014, col. 3, line 65-col. 4, line 11, col. 29, line 26-col. 30, line 10	1-20												
A	US 5805915 A (INTERNATIONAL BUSINESS MACHINES CORPORATION) 08.09.1998, abstract	1-20												
A	US 8533129 B2 (YAHOO! INC.) 10.09.2013, abstract, claim 1	1-20												
☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.														
<table border="0"> <tr> <td>* Special categories of cited documents:</td> <td>"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</td> </tr> <tr> <td>"A" document defining the general state of the art which is not considered to be of particular relevance</td> <td>"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</td> </tr> <tr> <td>"E" earlier document but published on or after the international filing date</td> <td>"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</td> </tr> <tr> <td>"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)</td> <td>"&" document member of the same patent family</td> </tr> <tr> <td>"O" document referring to an oral disclosure, use, exhibition or other means</td> <td></td> </tr> <tr> <td>"P" document published prior to the international filing date but later than the priority date claimed</td> <td></td> </tr> </table>			* Special categories of cited documents:	"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	"A" document defining the general state of the art which is not considered to be of particular relevance	"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	"E" earlier document but published on or after the international filing date	"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	"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)	"&" document member of the same patent family	"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	
* Special categories of cited documents:	"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													
"A" document defining the general state of the art which is not considered to be of particular relevance	"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													
"E" earlier document but published on or after the international filing date	"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													
"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)	"&" document member of the same patent family													
"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														
Date of the actual completion of the international search		Date of mailing of the international search report												
14 September 2016 (14.09.2016)		13 October 2016 (13.10.2016)												
Name and mailing address of the ISA/RU: Federal Institute of Industrial Property, Berezhkovskaya nab., 30-1, Moscow, G-59, GSP-3, Russia, 125993 Facsimile No: (8-495) 531-63-18, (8-499) 243-33-37		Authorized officer R. Lopatkina Telephone No. 499-240-25-91												