

(51) International Patent Classification:
G06T 7/00 (2006.01) **G06K 9/64** (2006.01)(21) International Application Number:
PCT/KR2009/002579(22) International Filing Date:
15 May 2009 (15.05.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10-2008-0045572 16 May 2008 (16.05.2008) KR(71) Applicant (for all designated States except US):
POSTECH ACADEMY-INDUSTRY FOUNDATION [KR/KR]; San 31, Hyoja-dong, Nam-gu, Pohang-shi, Kyungsangbuk-do 790-784 (KR).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **JEONG, Hong** [KR/KR]; 4-104 Gysu Apt., Jigok-dong, Nam-gu, Pohang-shi, Kyungsangbuk-do 790-390 (KR). **PARK, Sung Chan** [KR/KR]; 2-1504 Graduate School Apt., Jigok-dong, Nam-gu, Pohang-shi, Kyungsangbuk-do 790-390 (KR). **NA, Intae** [KR/KR]; Men's Dormitory 19-530, POSTECH, Hyoja-dong, Nam-gu, Pohang-shi Kyungsangbuk-do 790-784 (KR).(74) Agent: **JANG, Seong Ku**; 19th Fl., Trust Tower, 275-7 Yangjae-dong, Seocho-gu, Seoul 137-130 (KR).

(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, BR, BW, BY, BZ, CA, CH, 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, IS, JP, KE, KG, KM, KN, KP, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

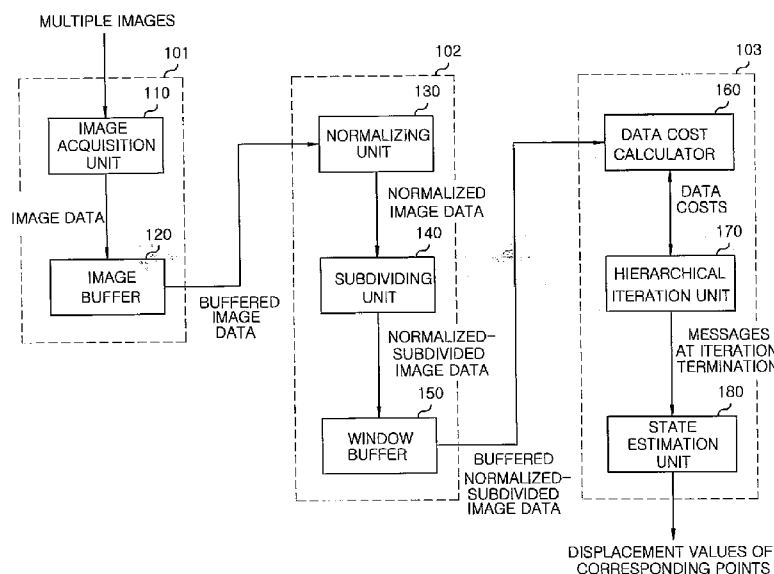
(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— without international search report and to be republished upon receipt of that report (Rule 48.2(g))

(54) Title: PARALLEL IMAGE PROCESSING APPARATUS AND METHOD FOR IMAGE MATCHING USING INTENSITY ADJUSTMENT

[Fig. 1]



(57) Abstract: A parallel image processing apparatus includes an image acquisition block, a preprocessing block and an image matching block. The image acquisition block includes: an image acquisition unit for extracting image data from input images; and an image buffer for temporarily storing the image data. The preprocessing block includes: a normalizing unit for normalizing the image data; a subdividing unit for subdividing the normalized image data; and a window buffer for temporarily storing the normalized-subdivided image data. The image matching block includes: a data cost calculator for computing data costs by using the normalized-subdivided image data; a hierarchical iteration unit for computing messages at iteration termination by using the data costs through hierarchical iteration operations; and a state estimation unit for collecting the messages by nodes to estimate a hidden state of each node and output a displacement value of a point corresponding to the hidden state.

Description

PARALLEL IMAGE PROCESSING APPARATUS AND METHOD FOR IMAGE MATCHING USING INTENSITY ADJUSTMENT

Technical Field

- [1] The present invention relates to a parallel image processing apparatus and method for image matching based on intensity adjustment; and, more particularly, to a parallel image processing apparatus and method that allow fast and reliable image matching through adjusting intensities corresponding to temporal or spatial variations in illumination between images.

Background Art

- [2] Finding or estimating matching pixels or regions between multiple images can be performed in two general ways. In local matching, corresponding points are estimated using only locally restricted information based on pixels or pixel windows. In global matching, corresponding points are estimated using global information collected from entire images.
- [3] Compared to local matching schemes, global matching schemes require more computations and higher system complexity, but show significantly lower error rates and produce stable results against occlusion. A global matching problem can be solved using belief propagation (BP), which enables Bayesian estimation on hidden Markov models with low error rates through iterative operations. In belief propagation on a graph composed of nodes and branches connecting the nodes, probability messages are repeatedly exchanged between neighboring nodes to estimate hidden states with minimum probabilistic energy.
- [4] Global matching based on belief propagation has demonstrated higher performance in comparison to other matching schemes, but has difficulty in real-time processing owing to large memory requirements and high computational complexity. To solve these problems, an improved belief propagation technique employing layered architectures and distance transformations has been proposed, and a real-time system using parallel graphics processing units (GPU) has been implemented on the basis of the improved technique. However, the real-time system has shown a limited performance improvement owing to requirements of high clock rates and memory bandwidths restricting the degree of parallelism. A recently proposed technique known as fast belief propagation (FBP) has enhanced the iterative processing architecture of conventional belief propagation to handle the requirements of high memory bandwidth and capacity, and has enabled achievement of a higher degree of parallelism through very large scale

integration (VLSI) technology. An image matching system based on the fast belief propagation has shown a significant performance improvement in image matching with reduced consumption of memory space and processing power in comparison to existing systems based on belief propagation.

- [5] Existing image matching systems based on belief propagation produce acceptable results for images captured under ideal indoor illumination, but may generate totally erroneous results for images that contain regions with pixel brightness biases due to external factors such as reflected light, shadows of hidden areas, unbalanced camera settings, and relative positions of light sources and cameras. This is because the data cost expressed in absolute values of differences between intensities cannot determine similarity between regions having similar intensity distributions but also having brightness biases.
- [6] Thus, to implement an image matching system based on belief propagation that produces robust performance results under various lighting conditions, it is necessary to remove brightness biases through brightness normalization in regions having intensity differences due to biases. Accordingly, many techniques based on belief propagation apply normalized cross-correlation to the data cost for better results.
- [7] Correlation based on pixel windows is used to measure image similarity, and is represented using the square of the Euclidean distance between feature vectors of pixel windows in images being compared. Because this correlation-based similarity measurement involves multi-pixel comparisons, it can produce a more reliable result in comparison to a scheme employing the absolute value of brightness differences between single pixels, but may not work in the presence of biases due to irregular illumination in whole or in part of patterns of similar intensities. In a scheme employing normalized cross-correlation, feature vectors whose means are zero are normalized and only direction vectors are taken for use, thereby minimizing the influence of biases on the matching process.
- [8] However, a system employing normalized cross-correlation has a high computational complexity of $O(n^4)$ due to a large number of multiplications, divisions, and square roots, and may be not suitable for parallel processing based on VLSI technology.

Disclosure of Invention

Technical Problem

- [9] In view of the above, the present invention provides a parallel image processing apparatus and method that allow reliable image matching in real time under various conditions through adjusting intensities.

Technical Solution

- [10] In accordance with a first aspect of the present invention, there is provided a parallel

- image processing apparatus for image matching, including:
- [11] an image acquisition block;
 - [12] a preprocessing block; and
 - [13] an image matching block,
 - [14] wherein the image acquisition block includes: an image acquisition unit for extracting image data from at least one input image; and an image buffer for temporarily storing the extracted image data;
 - [15] wherein the preprocessing block includes: a normalizing unit for normalizing the image data received from the image buffer; a subdividing unit for subdividing, according to a system window size, the normalized image data into normalized-subdivided image data; and a window buffer for temporarily storing the normalized-subdivided image data; and
 - [16] wherein the image matching block includes: a data cost calculator for computing data costs by applying normalized cross-correlation to the normalized-subdivided image data received from the window buffer; a hierarchical iteration unit for computing, for all nodes, messages at iteration termination by using the data costs received from the data cost calculator through fast belief propagation based hierarchical iteration operations; and a state estimation unit for collecting the messages by nodes to estimate for each node a state having the minimum accumulated value of the messages as a hidden state of the node and output a displacement value of a point corresponding to the hidden state.
 - [17] In accordance with a second aspect of the present invention, there is provided a parallel image processing method for image matching, including:
 - [18] extracting image data from at least one input image captured by multiple cameras;
 - [19] temporarily storing the extracted image data in an image buffer;
 - [20] normalizing the image data stored in the image buffer;
 - [21] subdividing, according to a system window size, the normalized image data into normalized-subdivided image data;
 - [22] temporarily storing the normalized-subdivided image data in a window buffer; and
 - [23] computing data costs by applying normalized cross-correlation to the normalized-subdivided image data stored in the window buffer;
 - [24] computing, for all nodes, messages at iteration termination by using the data costs through fast belief propagation based hierarchical iteration operations; and
 - [25] collecting the messages by nodes to estimate for each node a state having the minimum accumulated value of the messages as a hidden state of the node and output a displacement value of a point corresponding to the hidden state.

Advantageous Effects

- [26] According to the present invention, the image matching system employing normalized cross-correlation uses a hierarchical operation architecture for fast belief propagation, thereby enabling rapid convergence of information from a distance. Thus, in comparison to existing belief propagation based systems, the required number of iterative operations is significantly reduced, leading to saving of the memory space and reduction of the computation time.
- [27] In addition, fast belief propagation can be readily parallelized using VLSI, and it is possible to implement a robust real-time image matching system that exhibits stable performance in the presence of biases due to differences in lighting.

Brief Description of Drawings

- [28] The above and other objects and features of the present invention will become apparent from the following description of embodiments given in conjunction with the accompanying drawings, in which:
- [29] Fig. 1 illustrates a block diagram of an image processing apparatus in accordance with an embodiment of the present invention;
- [30] Fig. 2 illustrates a block diagram of the normalizing unit of the apparatus of Fig. 1;
- [31] Fig. 3 illustrates a block diagram of the window buffer of the apparatus of Fig. 1;
- [32] Fig. 4 illustrates a block diagram of the data cost calculator of the apparatus of Fig. 1;
- [33] Fig. 5 is a block diagram of the hierarchical iteration unit of the apparatus of Fig. 1;
- [34] Fig. 6 illustrates an exemplary hierarchical graph structure for hierarchical belief propagation;
- [35] Fig. 7 illustrates fast belief propagation based on the hierarchical graph structure of Fig. 6;
- [36] Fig. 8 illustrates a flowchart of an image processing method for the apparatus of Fig. 1; and
- [37] Fig. 9 illustrates a flowchart of the hierarchical iteration procedure in the method of Fig. 8.

Best Mode for Carrying out the Invention

- [38] Hereinafter, embodiments of the present invention will be described in detail with reference to the accompanying drawings, which form a part hereof.
- [39] Fig. 1 illustrates a block diagram of an image processing apparatus 100. The image processing apparatus 100 includes an image acquisition block 101, a preprocessing block 102 and an image matching block 103, which are connected in series. The image acquisition block 101 includes an image acquisition unit 110 and an image buffer 120. In the image acquisition block 101, image data acquired from multiple input images is buffered and then provided to the preprocessing block 102. The preprocessing block 102 includes a normalizing unit 130, a subdividing unit 140 and a window buffer 150.

In the preprocessing block 102, the image data is normalized into normalized image data and subdivided into normalized-subdivided image data. The normalized-subdivided image data is buffered and then provided to the image matching block 103. The image matching block 103 includes a data cost calculator 160, a hierarchical iteration unit 170 and a state estimation unit 180. In the image matching block 103, data costs are computed with the normalized-subdivided image data through a hierarchical iteration process and displacement values of corresponding points are output.

[40] In the image acquisition block 101, the image acquisition unit 110 and the image buffer 120 are connected together in series.

[41] The image acquisition unit 110 extracts from multiple input images captured by multiple cameras image data in a form of, e.g., streaming data or a video file, and sends the extracted image data to the image buffer 120. The image acquisition unit 110 may be a frame grabber or a video decoder. To use two or more cameras, the image acquisition unit 110 may further include a camera trigger and an image transmission synchronizing unit. The input images are digital images captured by cameras having the same resolution, and the original image and comparison image have the same size.

[42] The image buffer 120 is a memory, serving as a ring buffer, for storing in sequence a series of the image data from the image acquisition unit 110 and outputting the same upon request. The image buffer 120 serves as an intermediate storage indispensable for image processing based on pixel windows.

[43] In the preprocessing block 102, the normalizing unit 130, the subdividing unit 140 and the window buffer 150 are connected together in series.

[44] Fig. 2 illustrates a block diagram of the normalizing unit 130. The normalizing unit 130 includes a deviation calculator 240, an L1-norm calculator 250 and a divider 260.

[45] The deviation calculator 240 includes a scan-line buffer 210, a mean calculator 220 and a subtractor 230. The scan-line buffer 210 temporarily stores the image data received from the image acquisition block 101 in units of scan-lines, and provides the same as scan image data values to the mean calculator 220 and the subtractor 230. The mean calculator 220 computes a mean of the scan image data values received from the scan-line buffer 210, and outputs the computed mean to the subtractor 230. The subtractor 230 computes deviations by subtracting the mean computed by the mean calculator 220 from the scan image data values received from the scan-line buffer 210, and outputs the computed deviations between pixel values and their mean to the L1-norm calculator 250 and the divider 260. This operation sets an average of the deviations, i.e., an average coordinates of feature vectors of the image data, to zero, the average coordinates being for use in computing a normalized cross-correlation coefficient.

[46] Here, a pixel window can be set to contain 2^k pixels (k is a natural number) for

averaging through simple bit-shifting, in which case the mean calculator 220 includes an accumulator 222 and a shift register 224. The accumulator 222 computes the sum of the scan image data values received from the scan-line buffer 210 and temporarily holds the computed sum. Since the number of pixels is a power of 2, the shift register 224 can perform division on the value received from the accumulator 222 by simple bit-shift operation, thereby enhancing computation speed.

[47] The L1-norm calculator 250 includes an absolute value calculator 252 and an adder 254. L1-norm, also known as city block distance, is a measure for a vector size, and is given by a sum of absolute differences (SAD) of respective coordinates of the vector. The absolute value calculator 252 receives zero-mean feature vector, i.e., the deviations between pixel values and their mean from the deviation calculator 240, and removes signs of the deviations to output absolute values to the adder 254. The adder 254 computes an L1-norm of the feature vector by summing the absolute values, and outputs the computed L1-norm to the divider 260. The use of the L1-norm, not a Euclidean norm, reduces computation time and the number of logic devices.

[48] The divider 260 divides the zero-mean feature vector, i.e., the deviations between pixel values and their mean, by the L1-norm of the feature vector. Through this operation, the feature vector is converted into a direction vector, and output as the normalized image data to the subdividing unit 140.

[49] Referring back to Fig. 1, the subdividing unit 140 subdivides the normalized image data according to the pixel window size, and outputs the subdivision results as normalized-subdivided image data to the window buffer 150.

[50] The window buffer 150 temporarily stores in sequence the normalized-subdivided image data received from the subdividing unit 140, and outputs the same to the image matching block 103. When the image matching block 103 is designed to have a parallel architecture, the window buffer 150 can also have a parallel architecture in order for data cost synchronization.

[51] Fig. 3 illustrates a block diagram of a window buffer 150 having a parallel architecture. The window buffer 150 includes N parallel buffers 320-1 to 320-N, each of the buffers 320-1 to 320-N having serially-connected S window memories, where S is equal to the number of states in a belief propagation system. Each of the window memories 310-11 to 310-NS stores all pixels in a single pixel window. That is, the window buffer 150 has the N parallel buffers 320-1 to 320-N having a depth S. The buffers 320-1 to 320-N operate in parallel in a given state, and the window memories 310-11 to 310-NS are updated gradually according to state changes.

[52] The above-described architecture of the preprocessing block 102 prevents waste of logic devices that may be caused by introduction of normalized cross-correlation into a VLSI parallel computation system, and facilitates pixel window matching.

- [53] In the image matching block 103, the data cost calculator 160, the hierarchical iteration unit 170 and the state estimation unit 180 are connected together in series.
- [54] Fig. 4 illustrates a block diagram of the data cost calculator 160. The data cost calculator 160 includes parallel N processing modules 410-1 to 410-N, a hierarchical data cost evaluator 420 receiving outputs in parallel from the processing modules 410-1 to 410-N and a data cost memory 430 connected in series to the hierarchical data cost evaluator 420. The data cost calculator 160 corresponds to a single processing group on a hierarchical graph of M levels and includes N processing modules 410-1 to 410-N connected in parallel, where M is a natural number and N equals 2^{M-1} .
- [55] Each of the processing modules 410-1 to 410-N includes an absolute difference calculator, an accumulator and a shift register, which are connected together in series. Each of the absolute difference calculators 412-1 to 412-N receives the normalized-subdivided image data and computes absolute differences between original window pixel values and comparison window pixel values. Each of the accumulators 414-1 to 414-N adds the absolute differences together and temporarily holds the sum. Each of the shift registers 416-1 to 416-N halves the sum from the accumulators 414-1 to 414-N by bit shifting, and outputs the result to the hierarchical data cost evaluator 420.
- [56] The hierarchical data cost evaluator 420 receives N outputs in parallel from the processing modules 410-1 to 410-N, computes data costs through hierarchical operations based on fast belief propagation, and outputs the computed data costs to the data cost memory 430.
- [57] The data cost memory 430 stores the data costs received from the hierarchical data cost evaluator 420, and outputs the same to the hierarchical iteration unit 170.
- [58] Fig. 5 illustrates a block diagram of the hierarchical iteration unit 170. The hierarchical iteration unit 170 includes K processing groups 510-1 to 510-K connected in parallel and an iteration termination determiner 520. Each of the processing groups 510-1 to 510-K has a local buffer, a processing element and a layer buffer, and computes and updates messages for a node in a 2-dimensional graph through a hierarchical iterative operation based on fast belief propagation.
- [59] Each of the processing elements 514-1 to 514-K along with each of the local buffers 512-1 to 512-K computes a message of a node on a graph having a size equal to that of an image, and outputs the computed message to each of the layer buffers 516-1 to 516-K. Here, the message is computed on the basis of the data costs, a message at a previous iteration and a smoothness cost function between nodes. Each of the layer buffers 516-1 to 516-K temporarily stores the message, and output the same to the iteration termination determiner 520. The iteration termination determiner 520 determines whether to terminate the hierarchical iterative operation whenever the layer buffers 516-1 to 516-K are updated. If it is determined not to terminate the iteration,

the messages received from the layer buffers 516-1 to 516-K are provided to the processing groups 510-1 to 510-K as messages at a previous iteration. If it is determined to terminate the iteration, the messages received from the layer buffers 516-1 to 516-K are provided to the state estimation unit 180 as messages at iteration termination.

- [60] Fig. 6 illustrates an exemplary hierarchical graph structure for hierarchical belief propagation. For fast belief propagation, hierarchical belief propagation is used to obtain converged results with a fixed small number of iterations. In hierarchical belief propagation, for a graph of a given size, upper-level graphs having smaller numbers of nodes are successively created by undersampling, and then belief messages are iteratively computed along the graph hierarchy from the highest level graph to the original graph in a coarse-to-fine manner. In Fig. 6, the graph at level $k+1$ is obtained by undersampling the graph at level k , and can represent approximate information on the graph at level k with $1/4$ the number of nodes.
- [61] Fig. 7 illustrates fast belief propagation based on the hierarchical graph structure of Fig. 6. For fast belief propagation, message passing on graphs is iteratively performed in parallel using local buffers and layer buffers in the direction of image scan-lines. When iteration for a higher-level graph is terminated, a computed message for a node in the higher-level graph are used to initialize messages for corresponding nodes in the lower-level graph. This process enables information collected through iterations at a higher-level graph to be utilized at a lower-level graph, and can reduce the number of message passing iterations by rapidly collecting information from distant nodes.
- [62] In a hierarchical graph structure with M levels, a node of the highest level graph contains information on $2^{M-1} \times 2^{M-1}$ nodes of the lowest level graph. If an image width is given as C where C is a multiple of 2^{M-1} , the maximum number of necessary parallel processing elements is C and the minimum number thereof is $C/2^{M-1}$. Hence, processing elements can be divided into $C/2^{M-1}$ processing groups each having 2^{M-1} processing elements. Figs. 3 and 4 respectively illustrate the window buffer 150 and the data cost calculator 160 having such parallel architecture.
- [63] Referring back to Fig. 1, the state estimation unit 180 collects the messages at iteration termination from the hierarchical iteration unit 170. For each node, the state estimation unit 180 estimates a state having the minimum accumulated value of the messages as a hidden state of the node, and outputs a displacement value of a point corresponding to the hidden state.
- [64] The estimated states of nodes are displacement values between coordinates of original image pixels and those of corresponding comparison image pixels, and these values are output as a 2-dimensional array having the same size as the image size. The displacement values of corresponding points correspond to discrete coordinates of a

2-dimensional array having the same size as the input reference image size, and refers to differences in image coordinates between reference image pixels and corresponding comparison image pixels.

- [65] Hereinafter, an image processing method for the image processing apparatus 100 will be described with reference to a flowchart in Figs. 8 and 9 along with Figs. 1 to 5.
- [66] Referring to Fig. 8, the image acquisition unit 110 receives multiple input images captured by multiple cameras to extract image data therefrom (step S801). The extracted image data may be in the form of streaming data or a video file. The extracted image data is temporarily stored in the image buffer 120 (step S803).
- [67] In the deviation calculator 240 of the normalizing unit 130, the mean calculator 220 receives scan image data values from the scan-line buffer 210 to compute the mean of the scan image data values, and the subtractor 230 outputs deviations between the scan image data values and the mean thereof. In the L1-norm calculator 250, the absolute value calculator 252 computes the absolute values of the deviations, and the adder 254 computes the L1-norm by summing the absolute values. Then, the divider 260 divides the deviations by the L1-norm to produce normalized image data (step S805).
- [68] The subdividing unit 140 subdivides the normalized image data according to the pixel window size (step S807), and the window buffer 150 temporarily stores normalized-subdivided image data (step S809). The window buffer 150 may have N parallel buffers 320-1 to 320-N of a depth S.
- [69] The data cost calculator 160 computes data costs (step S811). The number of the processing modules 410-1 to 410-N of the data cost calculator 160, which are connected in parallel, may be the same as that of the buffers 320-1 to 320-N the window buffer 150.
- [70] Fig. 9 illustrates a flowchart of a hierarchical iteration procedure in the method of Fig. 8, the hierarchical iteration procedure following the step S811. The hierarchical iteration procedure processes the data costs calculated in the step S811 to generate messages at iteration termination (step S813). To be specific, the processing groups 510-1 to 510-K computes messages of nodes of the graph having a size equal to the image size on the basis of the data costs, messages at the previous iteration and a smoothness cost function between nodes (step S901). The layer buffers 516-1 to 516-K temporarily store the messages computed in the step S901 (step S903). The layer buffers 516-1 to 516-K provide the messages temporarily stored in the step S903 to the iteration termination determiner 520. The iteration termination determiner 520 determines whether to terminate the hierarchical iterative operation whenever the layer buffers 516-1 to 516-K are updated (step S905). If it is determined not to terminate the iteration, the messages received from the layer buffers 516-1 to 516-K 515 are provided to the processing groups 510-1 to 510-K as messages at the previous

iteration. If it is determined to terminate the iteration, the messages received from the layer buffers 516-1 to 516-K are provided to the state estimation unit 180 as messages at iteration termination.

[71] Referring back to Fig. 8, the state estimation unit 180 collects the messages at iteration termination by nodes, and for each node, estimates a state having the minimum accumulated value of the messages as the hidden state of the node (step S815). The state estimation unit 180 outputs displacement values of points corresponding to the hidden states.

[72] While the invention has been shown and described with respect to the embodiments, it will be understood by those skilled in the art that various changes and modification may be made without departing from the scope of the invention as defined in the following claims.

Claims

- [1] A parallel image processing apparatus for image matching, comprising:
an image acquisition block;
a preprocessing block; and
an image matching block,
wherein the image acquisition block includes: an image acquisition unit for extracting image data from at least one input image; and an image buffer for temporarily storing the extracted image data;
wherein the preprocessing block includes: a normalizing unit for normalizing the image data received from the image buffer; a subdividing unit for subdividing, according to a system window size, the normalized image data into normalized-subdivided image data; and a window buffer for temporarily storing the normalized-subdivided image data; and
wherein the image matching block includes: a data cost calculator for computing data costs by applying normalized cross-correlation to the normalized-subdivided image data received from the window buffer; a hierarchical iteration unit for computing, for all nodes, messages at iteration termination by using the data costs received from the data cost calculator through fast belief propagation based hierarchical iteration operations; and a state estimation unit for collecting the messages by nodes to estimate for each node a state having the minimum accumulated value of the messages as a hidden state of the node and output a displacement value of a point corresponding to the hidden state.
- [2] The apparatus of claim 1, wherein the image data extracted by the image acquisition unit is in a form of streaming data or a video file.
- [3] The apparatus of claim 1, wherein the image acquisition unit has a frame grabber or a video decoder.
- [4] The apparatus of claim 3, wherein the image acquisition unit further has a camera triggering device and an image transmission synchronizing device.
- [5] The apparatus of claim 1, wherein the image buffer is a ring buffer.
- [6] The apparatus of claim 1, wherein the normalizing unit includes:
a deviation calculator for calculating deviations, as a feature vector of the image data received from the image buffer, between image data values of the image data received from the image buffer and a mean value thereof;
an L1-norm calculator for calculating an L1-norm of the feature vector; and
a divider for dividing the feature vector by the L1-norm.
- [7] The apparatus of claim 6, wherein the deviation calculator includes:
a scan-line buffer for receiving the image data stored in the image buffer by

- scan-lines and temporarily storing the received image data;
a mean calculator for calculating the mean value of the image data values; and
a subtractor for subtracting the mean value from the image data values.
- [8] The apparatus of claim 7, wherein the image data has a size of a power of 2, and wherein the mean calculator includes:
an accumulator for temporarily storing the image data values; and
a shift register for computing the mean value of the image data values by bit-shifting output of the accumulator.
- [9] The apparatus of claim 6, wherein the L1-norm calculator includes:
an absolute value calculator for calculating absolute values of the deviations received from the deviation calculator; and
an adder for adding the absolute values together.
- [10] The apparatus of claim 1, wherein the window buffer includes one or more parallel buffers connected in parallel and the data cost calculator includes parallel processing modules identical in number to the parallel buffers of the window buffer.
- [11] The apparatus of claim 10, wherein each of the parallel buffers includes one or more serially-connected window memories.
- [12] The apparatus of claim 1, wherein the hierarchical iteration unit operates in a fast belief propagation method and includes:
one or more processing groups, connected in parallel, for performing the hierarchical iteration operations; and
an iteration termination determiner for determining whether to terminate the hierarchical iteration operations.
- [13] The apparatus of claim 12, wherein each of the processing groups includes: a local buffer; a processing element for computing, along with the local buffer, a message of a node on a graph having a size equal to a size of the input image on the basis of the data costs, messages at the previous iteration and a smoothness cost function between nodes; and a layer buffer for temporarily storing the computed message and outputting the message to the iteration termination determiner;
wherein the processing groups computes the messages in sequence in a direction of image scan-lines through parallel-processing; and
wherein the iteration termination determiner determines whether to terminate the iteration operations whenever the layer buffers are updated.
- [14] The apparatus of any one of claims 1 to 13, wherein the image data is in a form of window pixels.
- [15] A parallel image processing method for image matching, comprising:

extracting image data from at least one input image captured by multiple cameras;
temporarily storing the extracted image data in an image buffer;
normalizing the image data stored in the image buffer;
subdividing, according to a system window size, the normalized image data into normalized-subdivided image data;
temporarily storing the normalized-subdivided image data in a window buffer;
and
computing data costs by applying normalized cross-correlation to the normalized-subdivided image data stored in the window buffer;
computing, for all nodes, messages at iteration termination by using the data costs through fast belief propagation based hierarchical iteration operations; and
collecting the messages by nodes to estimate for each node a state having the minimum accumulated value of the messages as a hidden state of the node and output a displacement value of a point corresponding to the hidden state.

[16] The method of claim 15, wherein the extracted image data is in a form of streaming data or a video file.

[17] The method of claim 15, wherein the image data is extracted by an image acquisition unit having a frame grabber or a video decoder.

[18] The method of claim 17, wherein the image acquisition unit further has a camera triggering device and an image transmission synchronizing device.

[19] The method of claim 15, wherein the image buffer is a ring buffer.

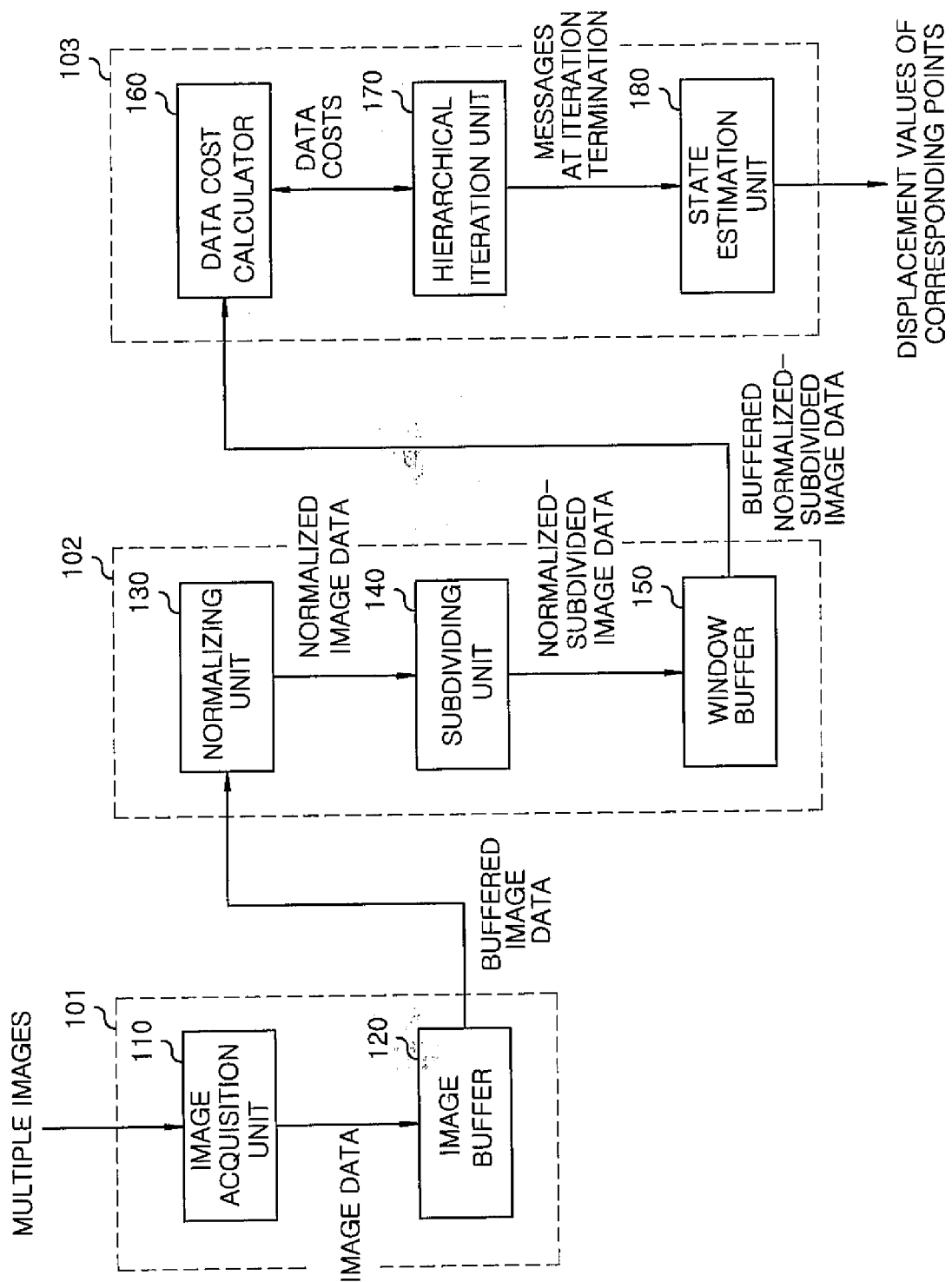
[20] The method of claim 15, wherein said normalizing the image data includes:
calculating deviations, as a feature vector of the image data received from the image buffer, between image data values of the image data received from the image buffer and a mean value thereof;
calculating an L1-norm of the feature vector; and
dividing the feature vector by the L1-norm.

[21] The method of claim 15, wherein said calculating the deviations includes:
receiving the image data stored in the image buffer by scan-lines and temporarily storing the received image data in a scan-line buffer;
calculating the mean value of the image data values; and
subtracting the mean value from the image data values.

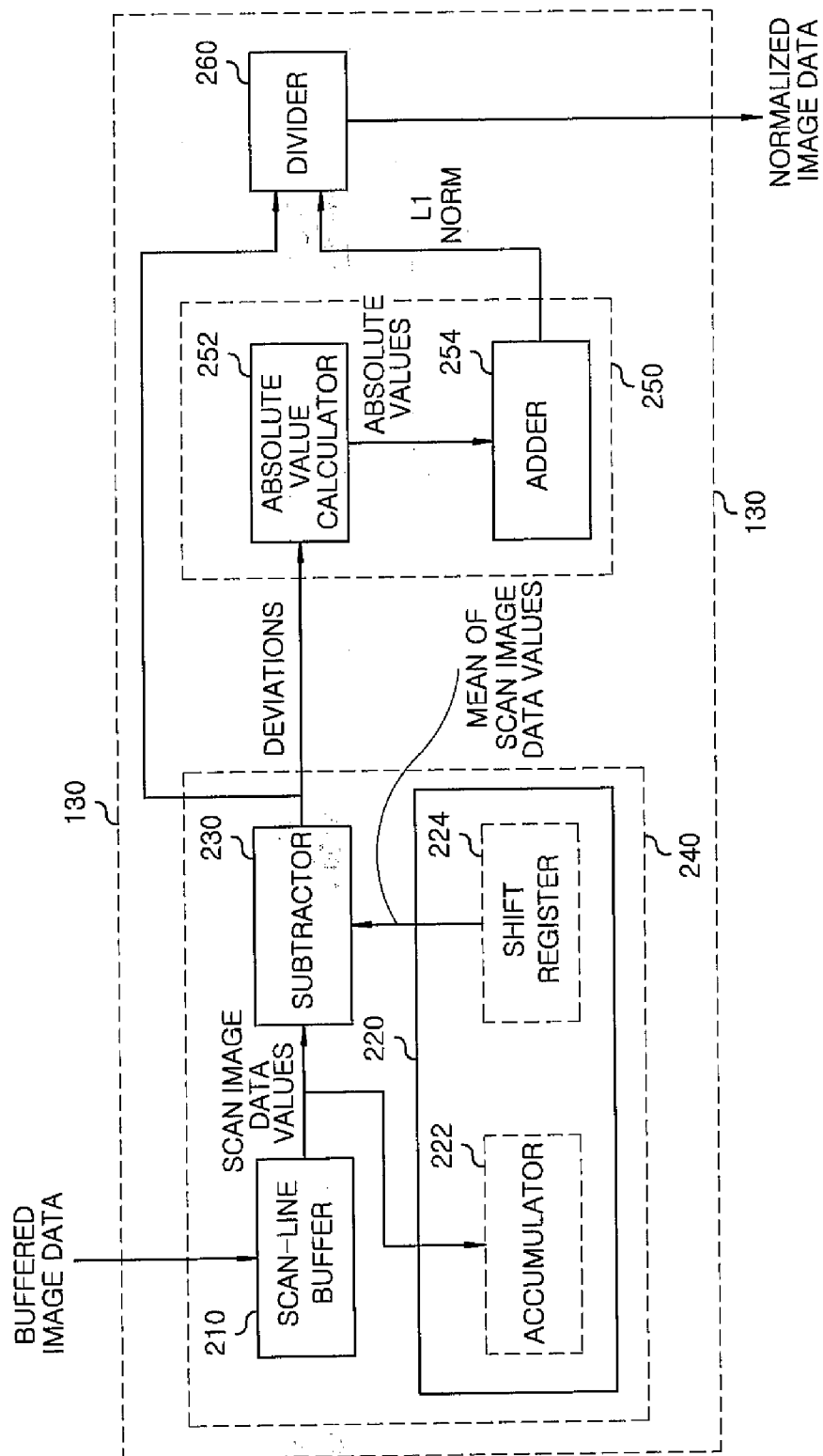
[22] The method of claim 21, wherein the image data has a size of a power of 2, and wherein said calculating the mean value includes:
temporarily storing the image data values in an accumulator; and
computing the mean value of the image data values by bit-shifting output of the accumulator.

- [23] The method of claim 20, wherein said calculating the L1-norm includes:
calculating absolute values of the deviations; and
adding the absolute values together.
- [24] The method of claim 15, wherein the window buffer includes one or more
parallel buffers connected in parallel and the data costs are computed by a data
cost calculator having parallel processing modules identical in number to the
parallel buffers of the window buffer.
- [25] The method of claim 24, wherein each of the parallel buffers includes one or
more serially-connected window memories.
- [26] The method of claim 15, wherein the hierarchical iteration operations includes:
computing messages of nodes of a graph having a size equal to a size of the input
image on the basis of the data costs, messages at the previous iteration and a
smoothness cost function between the nodes;
temporarily storing the computed messages in layer buffers; and
determining whether to terminate the iteration operations whenever the layer
buffers are updated.
- [27] The method of claim 26, wherein the messages are computed by processing
groups each having a local buffer, a processing element and a layer buffer, and
wherein said computing the messages are parallel-processed in sequence in a
direction of image scan-lines.
- [28] The method of any one of claims 15 to 27, wherein the image data is in a form of
window pixels.

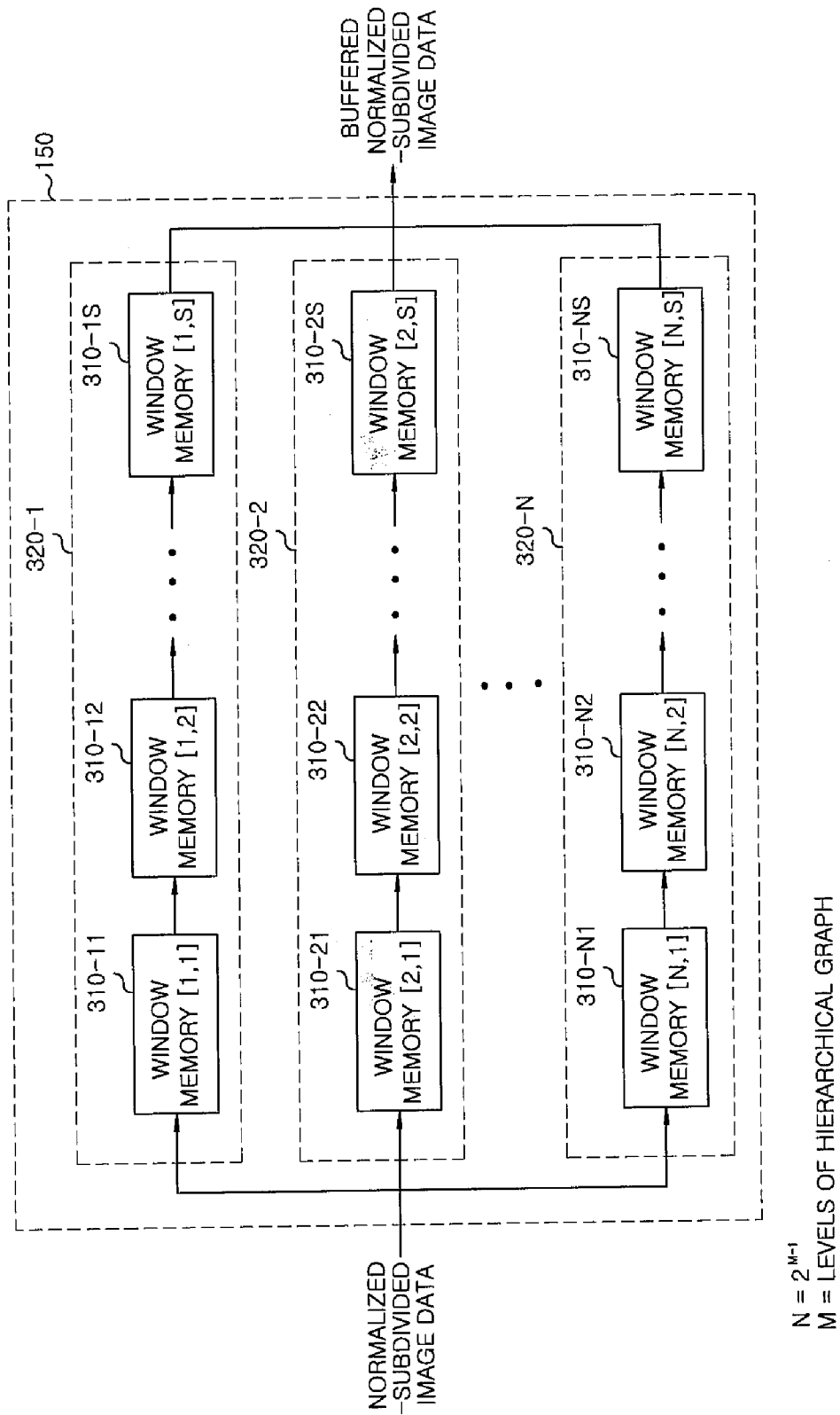
[Fig. 1]



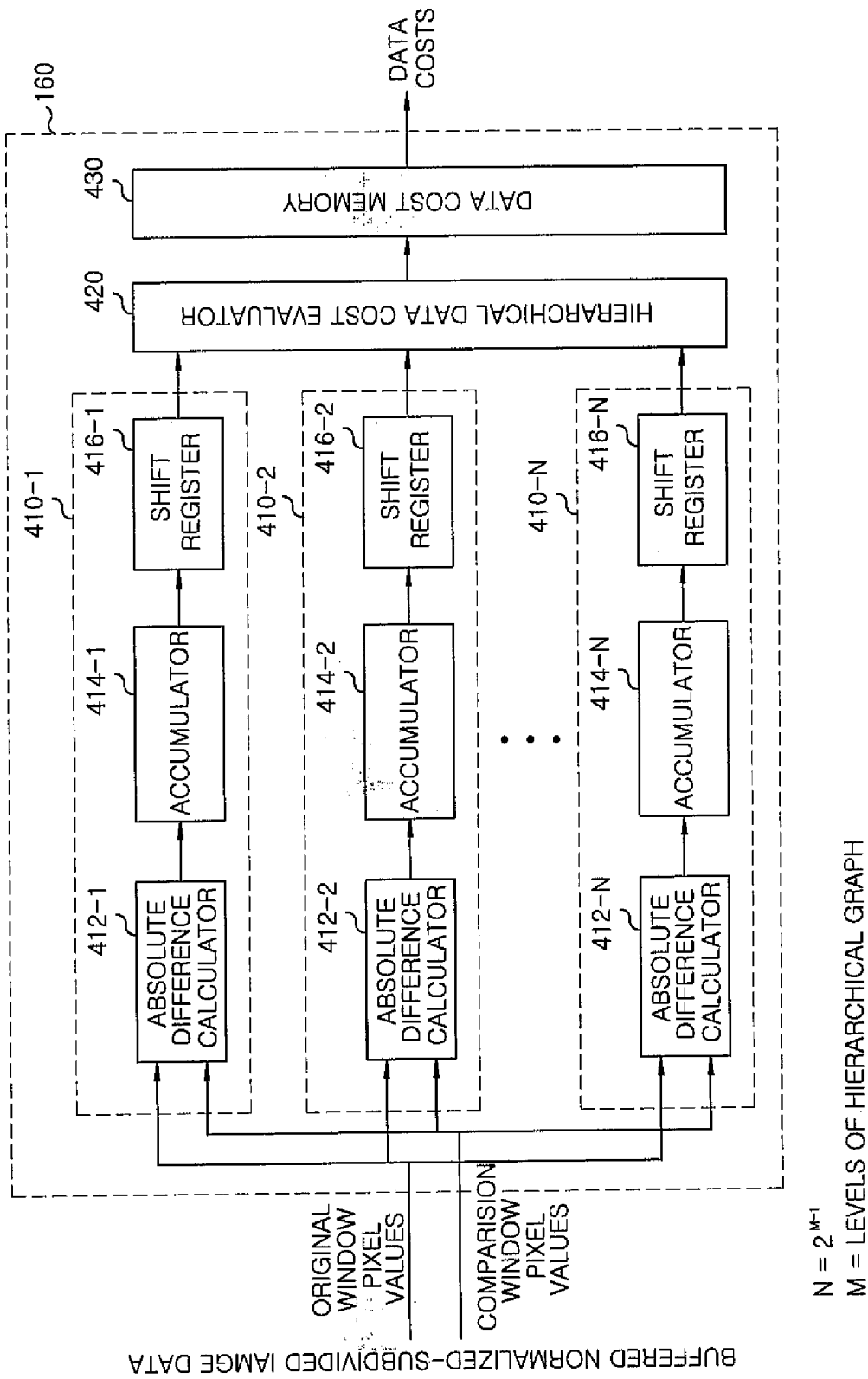
[Fig. 2]



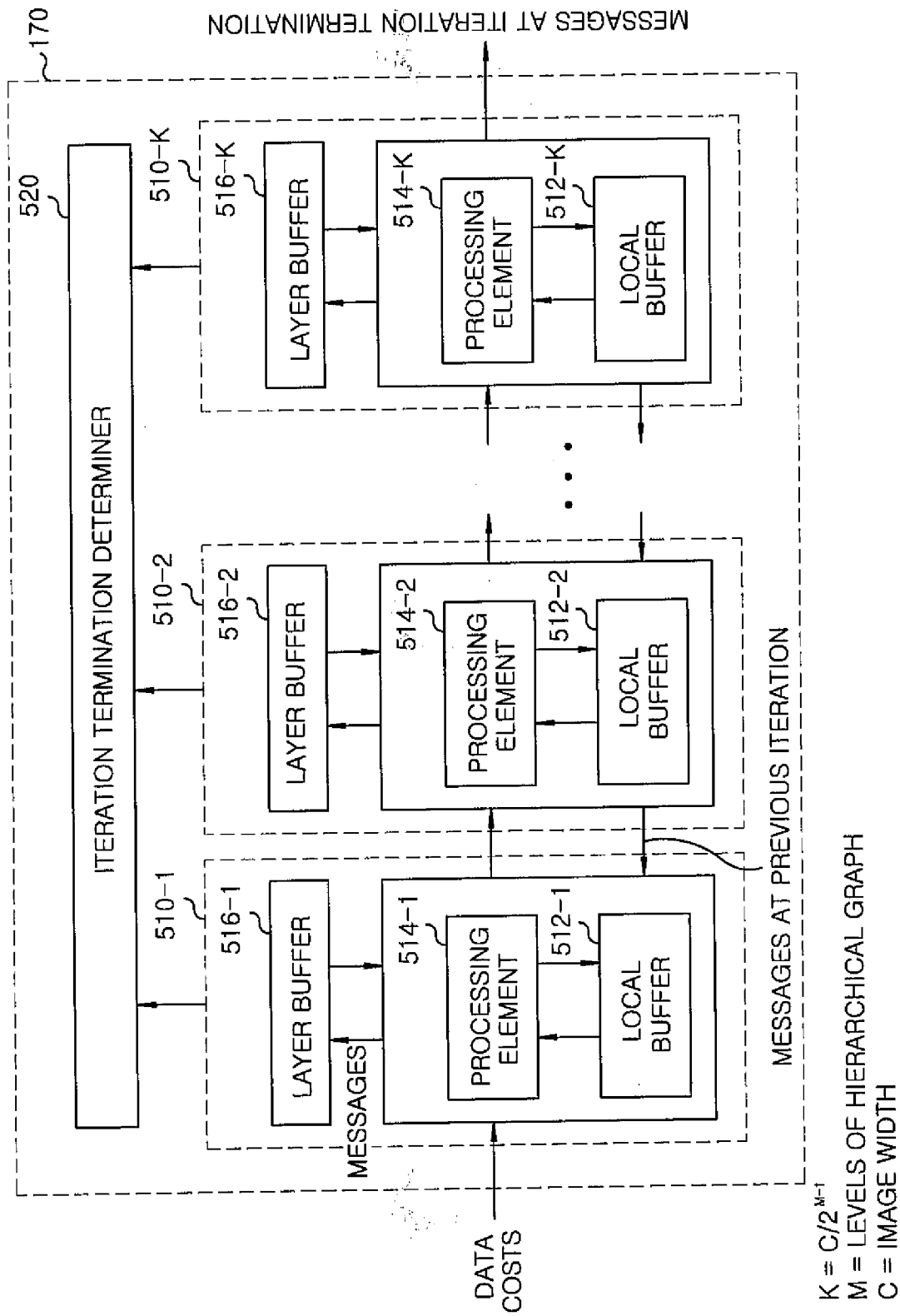
[Fig. 3]



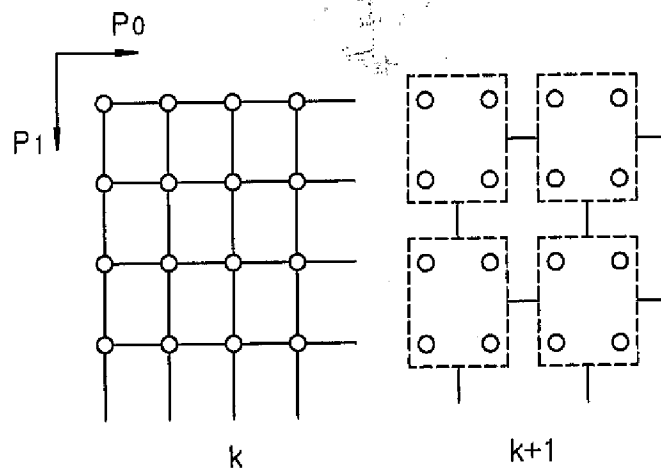
[Fig. 4]



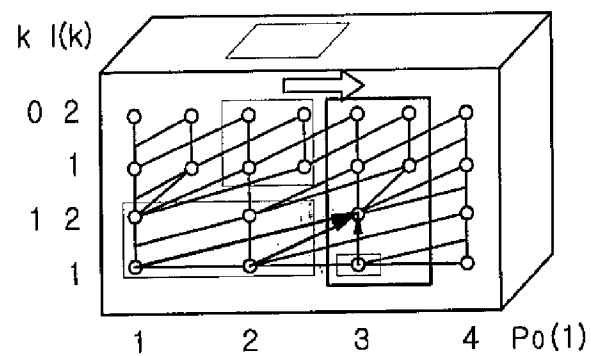
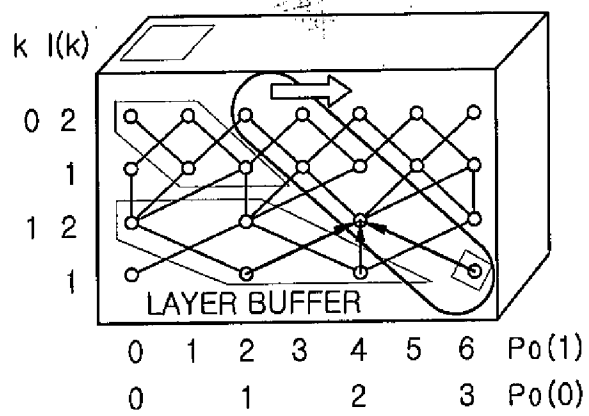
[Fig. 5]



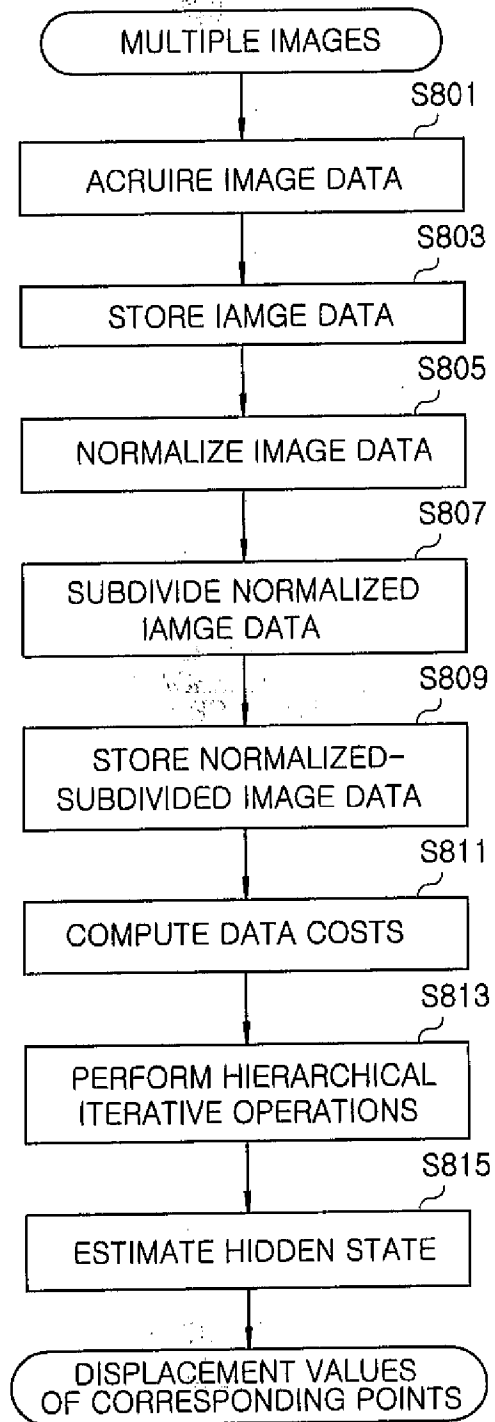
[Fig. 6]



[Fig. 7]



[Fig. 8]



[Fig. 9]

