(12) PATENT **(11)** Application No. **AU 200197460 B2**
(19) AUSTRALIAN PATENT OFFICE **(10)** Patent No. **769487**

(54) Title
A method for video region tracking through three-dimensional space-time segmentation

(51)[7] International Patent Classification(s)
G06T 001/00 **G06T 007/00**

(21) Application No: **200197460** (22) Application Date: **2001.12.24**

(30) Priority Data

(31) Number	(32) Date	(33) Country
PR2333	**2000.12.28**	**AU**

(43) Publication Date : **2002.07.04**
(43) Publication Journal Date : **2002.07.04**
(44) Accepted Journal Date : **2004.01.29**

(71) Applicant(s)
Canon Kabushiki Kaisha

(72) Inventor(s)
Brian John Parker; Julian Frank Andrew Magarey

(74) Agent/Attorney
SPRUSON and FERGUSON,GPO Box 3898,SYDNEY NSW 2001

(56) Related Art
US 4845560
AU 37921/00

ABSTRACT

A METHOD FOR VIDEO REGION TRACKING THROUGH THREE-DIMENSIONAL SPACE-TIME SEGMENTATION

A method (200) of segmenting a sequence of two-dimensional frames (10_z) in a video feed is disclosed. A three-dimensional block of pixels is formed from the sequence of two-dimensional frames, with two of the dimensions (x) being spatial, and the other one (z) being related to time. Each pixel has a component or value ($f(x,z)$) as a vector of measurements. The three-dimensional block of pixels is segmented into a set of three-dimensional connected regions, so that every pixel (x,z) in the block is related to one region (O_i) in which all pixels (x,z) belonging to the same region (O_i) have pixel components ($f(x,z)$) that are homogeneous in some sense. Once the block of pixels is fully segmented into regions O_i, a next frame (10_z) is merged into a newly formed block, while a fully segmented frame is removed off the "back" of the block. The fully segmented frame is preferably the oldest frame in time of the block and is returned as output (210). The newly formed block is further segmented. This has the effect of segmenting the newly added frame as well as incorporating it into the existing segmentation.

S&F Ref: 582100

AUSTRALIA

PATENTS ACT 1990

COMPLETE SPECIFICATION

FOR A STANDARD PATENT

ORIGINAL

Name and Address of Applicant :	Canon Kabushiki Kaisha 30-2, Shimomaruko 3-chome, Ohta-ku Tokyo 146 Japan
Actual Inventor(s):	Brian John Parker, Julian Frank Andrew Magarey
Address for Service:	Spruson & Ferguson St Martins Tower,Level 35 31 Market Street Sydney NSW 2000 (CCN 3710000177)
Invention Title:	A Method for Video Region Tracking Through Three-dimensional Space-time Segmentation

ASSOCIATED PROVISIONAL APPLICATION DETAILS

[33] Country	[31] Applic. No(s)	[32] Application Date
AU	PR2333	28 Dec 2000

The following statement is a full description of this invention, including the best method of performing it known to me/us:-

A METHOD FOR VIDEO REGION TRACKING THROUGH THREE-DIMENSIONAL SPACE-TIME SEGMENTATION

Field of Invention

The present invention relates to automatic scene analysis of a video signal and, in particular, to segmentation and tracking of regions in a video feed.

Background

Video may be defined as a sequence of images. Each image is digitally represented by image data in the form of a two-dimensional array of samples (known as pixels) of some measurable quantity. Examples of directly measurable image data include luminance and chrominance of reflected light (from optical cameras), range or distance from some reference point to the imaged points (from active range sensors), or density (from tomographic scanners).

Many quantities can be derived from the raw image data. Such quantities may be referred to as metadata, this being data that is used to describe other data. Examples of such "metadata" quantities include range from passive optical range sensors, and motion from multiple images of dynamic scenes.

Image segmentation is the decomposition of an image into homogeneous entities called regions. Because human expectation is that real world objects are in some sense compact and coherent, each segment of the partitioned image consists of a region of adjacent pixels over which some property of the data (image data, metadata, or both) is uniform. Image segmentation is an important process for many subsequent image-processing tasks.

However, in the processing of video for such tasks as metadata generation, not only are the significant two-dimensional regions in each image (or frame) required, but

these regions also need to be tracked through time. This is known as the correspondence problem between frames.

Traditionally, the problem of segmentation and object tracking in a video sequence has been approached by segmenting separately each single frame in the sequence, followed by a region-matching step in order to relate every region in each frame with a region in following frames. Algorithms often used in the region-matching step include correlation, feature tracking, or explicitly estimating and modelling the motion parameters of regions between frames. These algorithms are complex and make it difficult to generalise other two-dimensional image processing results to the field of video processing.

Another approach is to treat the video sequence as a three-dimensional signal, with the three dimensions being two spatial dimensions plus time, and then to use three-dimensional segmentation. As video data typically has a very long (potentially infinite) time axis, a limitation with this approach is the size of memory and processing power required to store and process such a large quantity of data. Further, the segmentation can only commence once all the frames are available. Accordingly, this approach is only suitable for post-processing, making it unsuitable for processing of video feeds.

To partially overcome these limitations, the sequence may be split into a number of smaller three-dimensional blocks, each having a given number of frames. Each of these smaller blocks is then segmented separately. However, as the video sequence is split into smaller blocks, this approach is unsuitable for object tracking purposes, because the regions belonging to the different blocks still need to be matched in a subsequent step.

Very few of the known techniques for two-dimensional segmentation are easily extendable to three dimensions. Unseeded variational region merging is an efficient

algorithm for segmentation which trades off region homogeneity with region compactness.

With automatic segmentation by variational region merging, a difficulty often experienced is deciding when to halt the merging process. Some implementations have required a predetermined "schedule" of thresholds to govern the merging process and converge to the segmentation which minimises a cost functional. Others have removed the need for a schedule, but still require an arbitrary threshold. The use of a predetermined arbitrary threshold means the segmentation algorithm is unable to adapt to different types of images in the video feed without substantial operator intervention.

Summary of the Invention

It is an object of the present invention to substantially overcome or at least ameliorate one or more problems associated with existing arrangements.

In accordance with one aspect of the present invention there is disclosed a method for segmenting a sequence of two-dimensional frames, each frame being formed by a plurality of pixels, each said pixel being described by a vector having components each relating to a different measured image characteristic, said method comprising the steps of:

(a) forming a three-dimensional block of said pixels from the first predetermined number of said frames on said sequence of frames;

(b) segmenting said three-dimensional block of pixels using a three-dimensional region-merging segmentor to form three-dimensional regions in a segmented block of pixels;

(c) concatenating a next frame in said sequence of frames to said segmented three-dimensional block of pixels;

(d) outputting a segmented two-dimensional frame from said segmented three-dimensional block of pixels; and

(e) repeating steps (b) to (d) for each subsequent frame of said sequence.

Other aspects of the present invention are also disclosed.

Brief Description of the Drawings

An embodiment of the present invention will now be described with reference to the drawings in which:

Fig. 1 is a block diagram showing the formation of a three-dimensional block of data from a sequence of image frames;

Fig. 2 is a schematic block diagram representing processing steps of a method of video tracking;

Fig. 3 is a block diagram showing the data flow during segmentation;

Fig. 4 is a graphical representation of a main data structure used in the three-dimensional segmentation;

Fig. 5 is a graphical representation of a two-dimensional array used in the three-dimensional segmentation;

Fig. 6A is a plot of the value of the merging cost as the algorithm proceeds in a typical case;

Fig. 6B is a plot similar to Fig. 6A but simplified and shown over an entire segmentation;

Fig. 7 is a schematic block diagram representing the segmentation processing steps used in the method of video tracking shown in Fig. 2; and

Fig. 8A shows an example three-dimensional region which always results in a single segment in each of the frames;

Fig. 8B shows an example three-dimensional region which results in disconnected segments in some of the frames;

Figs. 8C and 8D show an allowable configuration and a non-allowable configuration for merging respectively;

Fig. 9 is a schematic block diagram representation of a computer system in which arrangements described may be implemented.

Detailed Description

Some portions of the description which follows are explicitly or implicitly presented in terms of algorithms and symbolic representations of operations on data within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that the above and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise, and as apparent from the following, it will be appreciated that throughout the present specification, discussions utilizing terms such as "calculating", "determining", "replacing", "generating", "initializing", "outputting", or the like, refer to the action and processes of a computer

system, or similar electronic device, that manipulates and transforms data represented as physical (electronic) quantities within the registers and memories of the computer system into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present specification also discloses apparatus for performing the operations of the methods. Such apparatus may be specially constructed for the required purposes, or may comprise a general-purpose computer or other device selectively activated or reconfigured by a computer program stored in the computer. The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose machines may be used with programs in accordance with the teachings herein. Alternatively, the construction of more specialized apparatus to perform the required method steps may be appropriate. The structure of a conventional general-purpose computer will appear from the description below.

In addition, the present specification also discloses a computer readable medium comprising a computer program for performing the operations of the methods. The computer readable medium is taken herein to include any transmission medium for communicating the computer program between a source and a designation. The transmission medium may include storage devices such as magnetic or optical disks, memory chips, or other storage devices suitable for interfacing with a general-purpose computer. The transmission medium may also include a hard-wired medium such as exemplified in the Internet system, or wireless medium such as exemplified in the GSM mobile telephone system. The computer program is not intended to be limited to any particular programming language and implementation thereof. It will be appreciated that a variety of programming languages and coding thereof may be used to implement the teachings of the arrangements described herein.

Fig. 1 shows a sequence of two-dimensional frames 10_z in a video feed. A pixel in the z-th frame 10_z has a position (x,z), where x is a spatial position vector (x,y) within the z-th frame 10_z. Each pixel has a component or value $f(x,z)$ as a vector of measurements, which typically includes colour values from an image sensor, and are typically represented in the RGB or LUV colour space. The pixel components $f(x,z)$ may also contain range data or motion data, or some combination thereof. Hereafter, let m denote the number of components in the pixel component $f(x,z)$.

Also referring to Fig. 2 where a method 200 of video region tracking is shown, in step 202 a three-dimensional block of pixels is formed, with two of the dimensions (x) being spatial, and the other one (z) being related to time. The three-dimensional block is defined by a sliding window 20, which includes N frames from the sequence of frames. Initially, frames 10_1 to 10_N are included in the block.

Accordingly, the block of pixels is a three-dimensional grid, with each interior pixel (x,z) having six direct neighbours, those being pixels directly above, below, to the left and right, and in the preceding and following frames 10_{z-1} and 10_{z+1} of the pixel (x,z) in question. Pixels at the periphery of the block each have five neighbours.

Following step 202, method 200 proceeds to step 204 where the three-dimensional block of pixels is segmented into a set of three-dimensional connected regions $\{O_i\}$, so that every pixel (x,z) in the block is related to one region O_i in which all pixels (x,z) belonging to the same region O_i have pixel components $f(x,z)$ that are homogeneous in some sense. A connected region O_i is one in which each pixel (x,z) in the region O_i can be reached from every other pixel (x,z) in that same region O_i via a neighbourhood relation. Such a requirement forbids disjoint regions O_i. The details of the segmentation follow below.

Once the block of pixels is fully segmented into regions O_i, the sliding window 20 is advanced by one frame in time in step 206. This has the effect of merging a next frame 10_z into the newly formed block, while removing a fully segmented frame off the "back" of the block. The fully segmented frame is preferably the oldest frame in time of the block and is returned as output 210. More generally, any plane within the sliding window may be returned as output 210.

The newly formed block of pixels defined by the sliding window 20 now includes N-1 frames of previously segmented pixels (x,z), and one new frame of unsegmented pixel data (termed "trivial segmentation") concatenated to the front of the block. Trivial segmentation is where every pixel (x,z) constitutes its own region O_i. In step 208 the newly formed block is further segmented. This has the effect of segmenting the newly added frame as well as incorporating it into the existing segmentation. Once the block is segmented, the method returns to step 206, where again the sliding window 20 is advanced by one frame in time.

Method 200 has the effect of, as the sliding window 20 moves along the sequence of frames, returning fully segmented frames as output 210. Each of the output frames 10_z is a two-dimensional slice through the three-dimensional block which induces a two-dimensional segmentation of the frame into a set of connected regions $\{o_i\}$. Pixels in the same and adjoining frames that belong to the same three-dimensional region O_i are automatically identified as those pixels having the same region label i in the two-dimensional segmentation. Accordingly, tracking areas in the frames belonging to the same three-dimensional region O_i provides an efficient method of tracking "objects" in a video sequence, removing the need for a subsequent algorithm to solve the correspondence problem between regions of different frames.

The segmentation of the three-dimensional block of pixels, as used in steps 204 and 208 of method 200, is now described in more detail. An assumption underlying the segmentation problem is that each pixel component $f(x,z)$ is associated with a particular state. The model used to define the states is decided upon in advance. Each state i is defined by a unknown region model parameter vector $\bar{a}_i$, with each state i being assumed to be valid over the connected region O_i. The aim of segmentation is to identify these regions O_i and the model parameters $\bar{a}_i$ for each region O_i.

The neighbourhood or adjacency rule for pixels (x,z) (or "voxels" as the three-dimensional analogue of pixels are known), known per se in the art, extends to the regions O_i. That is, a region O_i is said to be adjacent to another region O_j if some voxel in the first region O_i is a neighbour of any voxel in the second region O_j. Separating each pair of neighbouring voxels (x,z) belonging to different regions O_i and O_j is a boundary element.

A model vector of measurements $g(x,z)$ over each region O_i is assumed to be a linear projection of the model parameter n-vector $\bar{a}_i$ for that region O_i:

$$g(x,z) = A(x,z)\,\bar{a}_i\,, \qquad (x,z) \in O_i \tag{1}$$

where $A(x,z)$ is a known m by n matrix which relates the state of region O_i to the model measurements $g(x,z)$, thereby encapsulating the nature of the predefined model.

Each vector of actual pixel components $f(x,z)$ is subject to a random error $e(x,z)$ such that

$$f(x,z) = g(x,z) + e(x,z). \tag{2}$$

Further, the error $e(x,z)$ may be assumed to be drawn from a zero-mean normal (Gaussian) distribution with covariance $\Lambda(x,z)$:

$$e(x,z) \sim N(0, \Lambda(x,z)) \tag{3}$$

Fig. 3 is a block diagram showing the data flow 100 during segmentation which occurs in steps 204 and 208 shown in Fig. 2. Together with the pixel components $f(x,z)$, the m by m covariance matrix $\Lambda(x,z)$ at each pixel (x,z) is an optional additional input to the algorithm. In traditional arrangements, it has been assumed that each component of the error $e(x,z)$ is independently and identically distributed, i.e.:

$$\Lambda(x,z) = \sigma^2(x,z)I_m \ . \tag{4}$$

However, the preferred implementation generalises this to encompass disparate and possibly mutually dependent measurement error components.

Variational segmentation requires that a cost function E be assigned to each possible segmentation. A partition into regions O_i may be compactly described by a binary function $K(d)$ on the boundary elements, in which the value one (1) is assigned to each boundary element d bordering a region O_i. This function $K(d)$ is referred to as a boundary map. It should be noted that because of the requirement of region connectedness, not every boundary map $K(d)$ defines a valid segmentation.

The cost function E used in the preferred implementation is one in which a model fitting error is balanced with an overall complexity of the model. The sum of the statistical residuals of each region O_i is used as the model fitting error. Combining Equations (1), (2), and (3), the residual over region O_i as a function of the model parameters a_i is given by

$$E_i(a_i) = \sum_{(x,z)\in O_i} [f(x,z) - A(x,z)a_i]^T \Lambda^{-1}(x,z)[f(x,z) - A(x,z)a_i] \tag{5}$$

The model complexity is simply the number of region-bounding elements d. Hence the overall cost functional E may be defined as

$$E(g,K,\lambda) = \sum_i E_i(a_i) + \lambda \sum_d K(d) \ , \tag{6}$$

where the (non-negative) parameter λ controls the relative importance of model fitting error and model complexity. The contribution of the model fitting error to the cost functional E encourages a proliferation of regions, while the model complexity encourages few regions. The functional E must therefore balance the two components to achieve a reasonable result. The aim of variational segmentation is to find a minimising model measurement $\bar{\boldsymbol{g}}$ and a minimising boundary map $\overline{K}(\boldsymbol{d})$ of the overall cost functional E, for a given parameter λ value.

Note that if the region boundaries $\boldsymbol{d}$ are given as a valid boundary map $K(\boldsymbol{d})$, the minimising model parameters $\bar{\boldsymbol{a}}_i$ over each region O_i may be found by minimising the region residuals E_i. This may be evaluated using a simple weighted linear least squares calculation. Given this fact, any valid boundary map $K(\boldsymbol{d})$ will fully and uniquely describe a segmentation. Therefore, the cost function E may be regarded as a function over the space of valid edge maps (K-space), whose minimisation yields an optimal region partition $\overline{K}_\lambda$ for a given parameter λ. The corresponding minimising model parameters $\bar{\boldsymbol{a}}_i$ may then be assumed to be those which minimise the residuals E_i over each region O_i. The corresponding minimum residual for region O_i will hereafter be written as $\overline{E}_i$.

From the above it is clear that the parameter λ is critical to the result. If parameter λ is low, many boundaries are allowed, giving "fine" segmentation. As parameter λ increases, the segmentation gets coarser. At one extreme, the optimal region partition $\overline{K}_0$, where the model complexity is completely discounted, is the trivial segmentation, in which every pixel (x,z) constitutes its own region O_i, and which gives zero model fitting error $\boldsymbol{e}$. On the other hand, the optimal region partition $\overline{K}_\infty$, where the model fitting error $\boldsymbol{e}$ is completely discounted, is the null or empty segmentation in which

the entire block is represented by a single region O_i. Somewhere between these two extremes lies the segmentation $\overline{K}_\lambda$ which will appear ideal in that the regions O_i correspond to a semantically meaningful partition.

To find an approximate solution to the variational segmentation problem, a region merging strategy has been employed, wherein properties of neighbouring regions O_i and O_j are used to determine whether those regions come from the same model state, thus allowing the regions O_i and O_j to be merged as a single region O_{ij}. The region residual E_{ij} also increases after any 2 neighbouring regions O_i and O_j are merged.

Knowing that the trivial segmentation is the optimal region partition K_λ for the smallest possible parameter λ value of 0, in region merging, each pixel (x,z) in the block is initially labelled as its own unique region O_i. Adjacent regions O_i and O_j are then compared using some similarity criterion and merged if they are sufficiently similar. In this way, small regions take shape and are gradually built into larger ones.

The segmentations $\overline{K}_\lambda$ before and after the merger differ only in the two regions O_i and O_j. Accordingly, in determining the effect on the total cost functional E after such a merger, a computation may be confined to those regions O_i and O_j. By examining Equations (5) and (6), a merging cost for the adjacent region pair $\{O_i,O_j\}$ may be written as

$$t_{ij} = \frac{\overline{E}_{ij} - (\overline{E}_i + \overline{E}_j)}{l(\delta_{ij})} \tag{7}$$

where $l(\delta_{ij})$ is the area of the common boundary between three-dimensional regions O_i and O_j. If the merging cost t_{ij} exceeds parameter λ, regions O_i and O_j should be merged.

The key to efficient region merging is to compute the numerator of the merging cost t_{ij} as fast as possible. Firstly, Equation (5) is rewritten as:

$$E_j(\boldsymbol{a}_j) = (F_j - H_j \boldsymbol{a}_j)^T \Gamma_j (F_j - H_j \boldsymbol{a}_j) \quad (8)$$

where:

- H_j is an $(n_j m)$ by n matrix composed of the individual $A(\boldsymbol{x},z)$ matrices stacked on top of one another as $(\boldsymbol{x},z)$ varies over region O_j;

- F_j is a column vector of length $(n_j m)$ composed of the individual pixel component $\boldsymbol{f}(\boldsymbol{x},z)$ vectors stacked on top of one another;

- Γ_j is an $(n_j m)$ by $(n_j m)$ block diagonal matrix, where each m by m diagonal block is the inverse of the covariance $\Lambda(\boldsymbol{x},z)$ matrix at the pixel $(\boldsymbol{x},z)$ denoted by the corresponding rows in F_j.

By weighted least square theory, the minimising model parameter vector $\bar{\boldsymbol{a}}_j$ for the region O_j is given by

$$\bar{\boldsymbol{a}}_j = \kappa_j^{-1} H_j^T \Gamma_j F_j \quad (9)$$

where κ_j is the confidence in the model parameter estimate $\bar{\boldsymbol{a}}_j$, defined as the inverse of its covariance:

$$\kappa_j = \Lambda_j^{-1} = H_j^T \Gamma_j H_j \quad (10)$$

The corresponding residual is given by

$$\bar{E}_j = F_j^T \Gamma_j \left(I_{mn_j} - H_j \kappa_j^{-1} H_j^T \Gamma_j \right) F_j \quad (11)$$

When merging two regions O_i and O_j, the "merged" matrix H_{ij} is obtained by concatenating matrix H_i with matrix H_j; likewise for matrices F_{ij} and Γ_{ij}. These facts may be used to show that the best fitting model parameter vector $\bar{\boldsymbol{a}}_{ij}$ for the merged region O_{ij} is given by:

$$\bar{\boldsymbol{a}}_{ij} = \bar{\boldsymbol{a}}_i - \kappa_{ij}^{-1} \kappa_j (\bar{\boldsymbol{a}}_i - \bar{\boldsymbol{a}}_j) \quad (12)$$

where the merged confidence is

$$\kappa_{ij} = \kappa_i + \kappa_j \tag{13}$$

and the merged residual is given by

$$\bar{E}_{ij} = \bar{E}_i + \bar{E}_j + \left(\bar{\boldsymbol{a}}_i - \bar{\boldsymbol{a}}_j\right)^T \kappa_i \kappa_{ij}^{-1} \kappa_j \left(\bar{\boldsymbol{a}}_i - \bar{\boldsymbol{a}}_j\right) \quad . \tag{14}$$

Combining Equations (13) and (14), the merging cost t_{ij} in Equation (7) may be computed as:

$$t_{ij} = \frac{\left(\bar{\boldsymbol{a}}_i - \bar{\boldsymbol{a}}_j\right)^T \kappa_i \left(\kappa_i + \kappa_j\right)^{-1} \kappa_j \left(\bar{\boldsymbol{a}}_i - \bar{\boldsymbol{a}}_j\right)}{l\left(\delta_{ij}\right)} \tag{15}$$

from the model parameters and confidence of the regions O_i and O_j to be merged. The matrix to be inverted is always of size n by n, (i.e. does not increase with region size). If the merge is allowed, Equations (12) and (13) give the model parameter $\bar{\boldsymbol{a}}_{ij}$ and confidence κ_{ij} of the merged region O_{ij}.

Note that under this strategy, only Equations (12), (13), and (15) need to be applied throughout the merging process. Only the model parameters $\bar{\boldsymbol{a}}_i$ and their confidences κ_i for each region O_i are therefore required as segmentation proceeds. Further, neither the original pixel components $\boldsymbol{f}(\boldsymbol{x},z)$ nor the model structure itself (i.e. the matrices $A(\boldsymbol{x},z)$) are required.

Variational linear-model-based segmentation may thus be separated into two stages as seen in Fig. 3, those stages being an initial model fitting stage 106 where the model parameters $\bar{\boldsymbol{a}}(\boldsymbol{x},z)$ and confidences $\kappa(\boldsymbol{x},z)$ are found for the pixel components $\boldsymbol{f}(\boldsymbol{x},z)$ and covariance $\Lambda(\boldsymbol{x},z)$ at each pixel $(\boldsymbol{x},z)$, followed by a region merging stage 108.

In the case of a zero-order model ($n=m$), e.g. RGB or LUV colour vectors used directly, the initial model-fitting stage 106 is trivial:

$$\bar{a}(x,z) = f(x,z) \tag{16}$$

$$\kappa(x,z) = \Lambda^{-1}(x,z)\,. \tag{17}$$

In the case of higher-order models, e.g. piecewise planar fitting on range data, model parameters $\bar{a}(x,z)$ and confidences $\kappa(x,z)$ at each pixel (x,z) may be obtained in any manner desired.

At the initialisation of the region merging stage 108, all adjacent region pairs O_i and O_j are determined and their corresponding merging cost t_{ij} evaluated. All the adjacent region pairs O_i and O_j are then "sorted" by a priority queue into ascending order of the merging costs t_{ij}. Region merging then involves popping a region pair O_i and O_j off the top of the priority queue (i.e. the region pair O_i and O_j with the lowest merging cost t_{ij}), merging this region pair O_i and O_j to form a new region O_{ij} with data members updated as described above.

Similarly, any boundaries that are now duplicated, and hence will be forming a multi-graph, are replaced with a new merged boundary, that has the sum of the areas of the two constituent boundaries.

The merging cost of all boundaries adjacent to O_{ij} are then updated, rearranging their order in the priority queue. The priority queue may be an exact priority queue such as a Fibonacci heap, or an approximate priority queue such as a bucket sorted bounded heap.

Note that it is this global ordering of the boundary merging cost t_{ij} that ensures that newly added unsegmented frame data will be preferentially segmented and merged into the existing three-dimensional segmentation.

This region-growing segmentation stage 108 may be shown to have run-time complexity of order ($M \log M$), where M is the number of pixels in a frame (or "slice" in the case where the three dimensions are spatial) assuming that the number of

neighbouring regions remains small relative to M, and provided the sorting and insertion can be done in "log time". This can be guaranteed if the list structure is maintained in computer memory in a structure called a priority queue.

To calculate a stopping point for this algorithm, note that this region-growing algorithm effectively provides a value at each merge operation, namely the merging cost t_{ij} of the pair being merged. It is therefore possible to build up a sequence of merging cost t_{ij} values as the algorithm proceeds. The algorithm halts if this merging cost t_{ij} value exceeds a threshold λ_{stop} at any time. If the algorithm were not halted, segmentation would continue until only one region remains.

In the preferred implementation, the value of the threshold λ_{stop} is automatically determined from the first block of N frames in the sequence of frames it is applied to.

To illustrate how the threshold λ_{stop} is determined, it is noted that, as merging proceeds, the merging cost t_{ij} of the regions O_i and O_j being merged generally increases. Fig. 6A shows a plot of merging cost t_{ij} during part of a segmentation of a real three-dimensional block of pixels, and Fig. 6B shows an artificial plot of the merging cost t_{ij} over time for an entire region merging process. As can be seen from Figs. 6A and 6B, the increase in merging cost t_{ij} is not purely monotonic. In fact, the overall rise in merging cost t_{ij} is punctuated by departures from monotonicity, which herein are termed local minima. A local minimum represents the collapse of what might be termed a self-supporting group of adjacent regions. Such occurs if one boundary within the group is removed, and the merging costs t_{ij} for adjacent boundaries then suddenly reduce. In effect, the hypothesis that the regions of the group are part of the same object is confirmed as more regions merge and so the merging cost t_{ij} decreases. The result is that all the boundaries in the group are removed in quick succession. These self-supporting groups tend to represent the internal structure of objects and background clutter. A

measure of merit such as the number of boundaries removed, or their total area, or the maximum (absolute or relative) decrease in merging cost t_{ij} may be assigned to each local minimum.

The point immediately after a local minimum, being a return to substantial monotonicity, is termed herein a stable configuration. Visually, a stable configuration represents a point in the region merging process at which an area of internal object structure or background clutter has just been removed, and is thus a good potential halting point. Each stable configuration has an associated merging cost t_{ij}. Fig. 6A also shows local minima and stable configurations.

If a complete pass is made through the segmentation, in which all regions are merged until only one region (the whole block) remains, all local minima and stable configurations for the block of frames may be found automatically by analysing the values of the merging cost t_{ij}. Significant local minima, being those whose measure of merit exceeds a certain threshold ζ, are flagged. As can be seen in Fig. 6B, during the early stages of region merging, local minima are common, giving the plot an erratic behaviour. As the regions become more established and substantial, the local minima frequency reduces until the null segmentation is reached (i.e. the block forms a single region). Those segmentations approaching the null will however be useful since the number of regions will be manageable computationally and most likely will be visually perceptible. As indicated above, a stable point is a desirable location to cease region merging, and Fig. 6B illustrates the identification of a limited number of candidate stopping locations, those being λ_{stop_1} , λ_{stop_2} , λ_{stop_3}, at significant stables near the null segmentation. The last significant stable configuration λ_{stop_1} is typically chosen as the threshold λ_{stop}, although any of the limited number of candidate stopping locations may be selected depending on the particular video feed and/or application being processed.

Further, where the image has a large number of local minima (e.g. hundreds, thousands or more), the limited number of candidate stopping positions may be limited (eg. in the "tens").

To achieve this stable configuration during the region-growing segmentation stage 108, the merging of regions must stop once the merging cost t_{ij} exceeds the threshold λ_{stop}. To restore this state, the region merging segmentation stage 108 need only reverse its last few merging operations by restoring the algorithm state appropriately. Alternatively, the region merging segmentation may be run again from the start, halting when the value of merging cost t_{ij} reaches the threshold λ_{stop}.

The latter, more expensive alternative is used wherein a first pass of the region merging segmentation is used to determine the threshold λ_{stop}, and subsequent passes halt once the merging cost t_{ij} reaches the threshold λ_{stop}.

Fig. 7 shows the region-growing segmentation stage 108, which forms part of the data flow 100 (Fig. 3) during segmentation in more detail. It is again noted that the data flow 100 occurs during the segmentation steps 204 and 208 shown in Fig. 2. In the preferred implementation, the first pass only occurs in step 204 as it is assumed that subsequent frames added to the block of frames have similar characteristics to those frames 10_1 to 10_N included in the initial block.

The region-growing segmentation stage 108 starts at step 302 and proceeds to step 304 which receives the model parameters $\bar{a}(x,z)$ and model confidences $\kappa(x,z)$ for each pixel (x,z) in the block of N frames from the model fitting stage 106 (Fig. 5). The region-growing segmentation stage 108 starts with the trivial segmentation where each pixel (x,z) forms its own region O_i. Step 306 then determines all adjacent region pairs O_i and O_j, and computes the merging cost t_{ij} according to Equation (15) for each of the

boundaries between adjacent region pairs O_i and O_j. Step 308 inserts the boundaries with merging cost t_{ij} into a priority queue T in priority order.

Step 310 takes the first entry from the priority queue T(1) and merges the corresponding region pair O_i and O_j (i.e. the region pair O_i and O_j with the lowest merging cost t_{ij}) to form a new region O_{ij}. Step 312 records the merging cost t_{ij} in a list L.

Step 314 identifies all boundaries between regions O_l adjoining either of the merged regions O_i and O_j and merges any duplicate boundaries, adding their areas. Step 318 follows by calculating a new merging cost $t_{ij,l}$ for each boundary between adjacent regions O_{ij} and O_l. The new merging costs $t_{ij,l}$ effectively reorder the priority queue into the final sorted queue in step 318.

During the first pass, the region-growing segmentation stage 108 then proceeds to step 324 where it is determined whether null segmentation has been reached. This is done by determining whether more than one region remains. If the null segmentation has not been reached, then control returns to step 310. Steps 310 to 324 are repeated until null segmentation is reached.

When all regions have been combined into the null segmentation, step 324 passes control to step 326 which then identifies the merging cost t_{ij}, stored in list L, corresponding to the last stable configuration. This is assigned the threshold λ_{stop}. This concludes the first pass.

Control returns to step 304 where the segmentation is started again with the trivial segmentation where each pixel (x,z) forms its own region O_i. Steps 306 to 318 follow as described above where the pixels are merged to form regions in the second pass.

With threshold λ_{stop} determined, step 318 passes control to step 322 to determine if the merging has reached the stopping point. This is done by determining whether the merging cost t_{ij} corresponding to the regions O_i and O_j at the top of the priority queue T

(entry T(1)) has a value greater than the threshold λ_{stop}. If the merging has reached a stopping point, then the region-growing segmentation stage 108 ends in step 330. Alternatively, control is returned to step 310 and steps 310 to 322 are repeated, merging the two regions with the lowest merging cost t_{ij} every cycle, until the stopping point is reached.

In order to apply the region-growing segmentation stage 108 described above to a wide variety of data sources and models, it is desirable to use an adaptive measure of merit to determine whether a significant local minimum has been reached when analysing, in step 326, the merging costs t_{ij} stored in list L. In the preferred implementation, the measure of merit used corresponds to the number of boundaries removed during a local minimum.

Fig. 4 shows a graphical representation of a data structure 500 for use with the region-growing segmentation stage 108, which is essentially an adjacency list representation of the segmentation graph. The data structure 500 includes a region list 510, a boundary list 520 for each region in the region list 510, and a priority queue 530. The region list 510 contains for each region O_i a region identifier, co-ordinates (x,z) of all pixels in that region O_i, the model parameter vector $\bar{a}_i$ and the model confidence κ_i. The priority queue 530 contains and orders boundaries B_{ij}. These are data structures which store the merging costs t_{ij} of two regions O_i and O_j, the area $l(\delta_{ij})$ of the common boundary between regions O_i and O_j, and pointers to the corresponding regions O_i and O_j. Each boundary list 520 contains a list of pointers, with each pointer pointing to an entry in the priority queue corresponding to the boundary between adjacent regions O_i and O_j under consideration. This data structure 500 allows the region-growing segmentation stage 108 to efficiently store all the data and parameters required for the segmentation.

Fig. 5 shows a graphical representation of a two-dimensional array D, which is maintained during steps 206 and 208. It conceptually corresponds to the two-dimensional intersection of the segmented three-dimensional regions O_i with the frame at the "front" of the sliding window. This is the last frame that was added to the block of N frames. Each pixel position in the two-dimensional array D stores a pointer to the segmented three-dimensional region O_i that intersects it, allowing constant time lookup of the adjacent region, so that a new frame can be concatenated to the graph data structure 500 in constant time per pixel.

The region-merging segmentation stage 108 described above starts with the trivial segmentation where each pixel (x,z) forms its own region O_i. In order to lower memory cost and to decrease runtime, an initial region-splitting preprocessor may be used. The region-splitting preprocessor preprocesses a frame 10_z into square regions of pixels before adding it to the three-dimensional block. Accordingly, the region merging segmentation stage 108 starts with fewer regions than in the above implementation.

The preprocessor utilises a tiled quadtree decomposition of the newly added frame 10_z. Rather than adding the frame 10_z as individual pixels, the preprocessor attempts to add squares of MxM pixels in scan-line order, where M is some predetermined small power of two, such as 16. For each square of MxM pixels it is determined whether the pixel components $f(x,z)$ of the pixels contained in the square all have values within a predetermined small tolerance. If this condition is met, the square is added in its entirety as a single region. Alternatively, the square is subdivided into 4 equally sized smaller blocks. Each of these smaller blocks are themselves tested for homogeneity, and added as a single region if so, otherwise the process is continued recursively until single pixel-sized regions are reached.

Note that by adding square (or rectangular) blocks, a frame 10_z may be tiled in scan-line order, and hence the neighbouring regions can be quickly determined and the regions added in constant time. This would not hold for arbitrarily shaped regions.

The method 200 for tracking regions in a sequence of frames relies on the fact that there exists some "connectivity" between corresponding regions in consecutive frames. This requirement may not be met when the sequence includes small, fast moving objects. An adequate temporal sampling rate is required for such objects to remain connected in the three-dimensional space-time domain. Typical sampling rates of 25 Hz (or 50 Hz for interlaced video) are, in fact, adequate for tracking most objects of interest, such as moving humans.

If the temporal sampling rate is fixed and small, and fast moving objects need to be tracked, then one of the following options may be used:

1. Filter the original analog video signal temporally before temporal sampling. This introduces motion blur, and method 200 takes advantage of such motion blur, as it assists in causing connectivity in the three dimensional space-time;

2. The effect of option 1 above may be approximated by digital processing by super-sampling, where frames are captured at a higher rate than is required. Super-sampling is then followed by down-sampling by averaging several frames together; or

3. Motion-guided frame interpolation, known per se in the art, can be used to generate intermediate frames between given frames to increase the effective temporal sampling rate of data passed into the algorithm of method 200. To perform motion-guided frame interpolation, an estimate of the motion field is required, which may be calculated using any one of a number of known algorithms.

A problem that is often encountered when performing region merging on video as a three-dimensional block of voxels is that connected regions in the three-dimensions

may be *non-simple* along the time axis, which results in disconnected regions in the two-dimensional segmented frame returned in step 210 (Fig. 2). Fig. 8A shows a region A which always results in a single segment s_z in each of the frames 10_z. However, the region B shown in Fig. 8B results in disconnected segments s_{z1} and s_{z2} in frame 10_z. This may adversely affect later image processing algorithms using the segmented data of the frames 10_z.

A typical approach to this problem is to rectify the returned two-dimensional segmented frames by post-processing, whereby connectivity of each labelled region O_j of the segmented frame 10_z is tested. Each disconnected segment is then either relabelled or removed. This form of post-processing is slow and significantly complicates algorithms it is applied to.

A preferred approach is to intervene during the three-dimensional region-growing segmentation stage 108 (Fig. 3). Regions O_j that are non-simple along the time axis with the sliding window 20 are prevented from being generated, where "simple" means that a cross-section along the time axis will give a single two-dimensional connected component. Hence, disconnected segments such as s_{z1} and s_{z2} shown in Fig. 8B cannot occur.

Before merging neighbouring regions O_i and O_j in step 310 (Fig. 7), it is tested whether the resulting region O_{ij} would comply with an "allowed configuration" defined below. If the resulting region O_{ij} would not comply with the allowed configuration, then the merger of regions O_i and O_j is prevented by effectively setting the merging cost t_{ij} to infinity.

Referring to Fig. 8C, a minimum and maximum extent of a region A along the time axis is defined as RA_{tmin} and RA_{tmax} respectively. Similarly, the minimum and maximum extent of a region B along the time axis is defined as RB_{tmin} and RB_{tmax}

respectively. A three-dimensional boundary separating regions A and B has a time extent ($Boundary_{tmin}$, $Boundary_{tmax}$). Let w_{min} be the minimum extent of the sliding window in time and w_{max} the maximum extent. An allowed configuration, as shown in Fig. 8C, is defined as configurations wherein:

$[\max(RA_{tmin},w_{min})=\max(Boundary_{tmin},w_{min})$ OR $\max(RB_{tmin},w_{min})=\max(Boundary_{tmin},w_{min})]$ AND $[\min(RB_{tmax},w_{max})=\min(Boundary_{tmax},w_{max})$ OR $\min(RA_{tmax},w_{max})=\min(Boundary_{tmax},w_{max})]$. All other configurations are disallowed.

An example of such a disallowed configuration is shown in Fig. 8D. As can be seen from the illustration, a merger between regions A and B will not result in an allowable configuration according to the above condition.

The calculation of the extent RB_{tmin} and RB_{tmax} of a region B along the time axis of all the regions, as well as the extent $Boundary_{tmin}$ and $Boundary_{tmax}$ of the boundaries is easily calculated in the initialisation step 304, as the extent of pixels or regions and their boundaries are trivially known. When merging two regions, the extent of the new region may be updated in constant time by assigning the maximum and minimum extents of the two constituent regions. Boundaries may be updated in a similar manner.

All initial pixels are simple in time, and by maintaining the condition that all regions O_i remain simple at each merging step, then the final three-dimensional segmentation must be simple, i.e. can not result in disconnected two-dimensional segments in cross-section.

For isotropic three-dimensional data, the presumption that the number of neighbouring regions remains small is correct and the region-growing segmentation stage 108 has $O(M \log M)$ run-time. However, when concatenating unsegmented data onto the previously segmented three-dimensional sliding window, this presumption does not hold, as the data is anisotropic, and the number of neighbouring regions can grow to large sizes.

To maintain a reasonable O($M \log M$) run-time, the boundedness of the number of neighbouring regions needs to be guaranteed. This is done by extending the region-growing segmentation stage 108 so that the merging cost t_{ij} is set to infinity if either of the two adjoining regions O_i and O_j has a number of neighbouring regions greater than some threshold. This effectively removes the affected regions O_j and O_j from merging until such time as their number of neighbouring regions has shrunk below some predetermined bound, ensuring an O($M \log M$) run-time.

The method 200 is not limited to analysing regions in a video feed, but may also be used for segmentation of other forms of three-dimensional data, such as data in three spatial dimensions. The method 200 is particularly useful where segmentation is required for large quantities of three-dimensional data, such as resulting from computed tomography (CT) or magnetic resonance imaging (MRI). Segmentation of the whole three-dimensional block using a traditional three-dimensional segmentor would require a very large data storage facility and would be slow. By moving the whole three-dimensional block through a sliding window which defines a smaller block of data, segmentation is performed on the smaller block of data. Every time the sliding window is advanced, a "slice" of data from the whole three-dimensional block is added to the smaller block and a fully segmented "slice" of data is produced as output. The sliding window is moved over the whole three-dimensional block, until a completely segmented block of data is formed.

The method 200 of video tracking described above may be practiced using a programmable device, and are preferably practiced using a conventional general-purpose computer system 400, such as that shown in Fig. 9 wherein methods 200 may be implemented as software, such as an application program executing within the computer system 400. In particular, the steps of methods 200 are effected by instructions in the

software that are carried out by the computer. The software may be divided into two separate parts; one part for carrying out the video tracking method and another part to manage the user interface between the latter and the user. The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer preferably effects an advantageous apparatus for video tracking in accordance with the embodiments of the invention.

The computer system 400 comprises a computer module 401, input devices such as a keyboard 402 and mouse 403, output devices including a printer 415 and a display device 414. A Modulator-Demodulator (Modem) transceiver device 416 is used by the computer module 401 for communicating to and from a communications network 420, for example connectable via a telephone line 421 or other functional medium. The modem 416 can be used to obtain access to the Internet, and other network systems, such as a Local Area Network (LAN) or a Wide Area Network (WAN).

The computer module 401 typically includes at least one processor unit 405, a memory unit 406, for example formed from semiconductor random access memory (RAM) and read only memory (ROM), input/output (I/O) interfaces including a video interface 407, and an I/O interface 413 for the keyboard 402 and mouse 403 and optionally a joystick (not illustrated), and an interface 408 for the modem 416. A storage device 409 is provided and typically includes a hard disk drive 410 and a floppy disk drive 411. A magnetic tape drive (not illustrated) may also be used. A CD-ROM drive 412 is typically provided as a non-volatile source of data. The components 405 to 413 of the computer module 401, typically communicate via an interconnected bus 404

and in a manner which results in a conventional mode of operation of the computer system 400 known to those in the relevant art. Examples of computers on which the embodiments can be practised include IBM-PC's and compatibles, Sun Sparcstations or alike computer systems evolved therefrom.

Typically, the application program of the preferred embodiment is resident on the hard disk drive 410 and read and controlled in its execution by the processor 405. Intermediate storage of the program and any data fetched from the network 420 may be accomplished using the semiconductor memory 406, possibly in concert with the hard disk drive 410. In some instances, the application program may be supplied to the user encoded on a CD-ROM or floppy disk and read via the corresponding drive 412 or 411, or alternatively may be read by the user from the network 420 via the modem device 416. Still further, the software can also be loaded into the computer system 400 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer module 401 and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including e-mail transmissions and information recorded on websites and the like. The foregoing is merely exemplary of relevant computer readable mediums. Other computer readable media may be practiced without departing from the scope and spirit of the invention.

The methods described may alternatively be implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub functions and for example incorporated in a digital video camera 450. Such dedicated hardware may include graphic processors, digital signal processors, or one or more microprocessors and associated memories. As seen, the camera 450 includes a display screen 452 which can be used to display the segmented frames of information regarding then same. In this

fashion, a user of the camera may record a video sequence, and using the processing methods described above, create metadata that may be associate with the video sequence to conveniently describe the video sequence, thereby permitting the video sequence to be used or otherwise manipulated with a specific need for a user to view the video sequence. A connection 448 to the computer module 401 may be utilised to transfer data to and/or from the computer module 401 for performing the video tracking process.

Industrial Applicability

It is apparent from the above that the embodiment(s) of the invention are applicable to the video processing industries where video sequences may require cataloguing according to their content.

The foregoing describes only one embodiment/some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiment(s) being illustrative and not restrictive.

In the context of this specification, the word "comprising" means "including principally but not necessarily solely" or "having" or "including" and not "consisting only of". Variations of the word comprising, such as "comprise" and "comprises" have corresponding meanings.

The claims defining the invention are as follows:

1. A method for segmenting a sequence of two-dimensional frames, each frame being formed by a plurality of pixels, each said pixel being described by a vector having components each relating to a different measured image characteristic, said method comprising the steps of:

(a) forming a three-dimensional block of said pixels from the first predetermined number of said frames on said sequence of frames;

(b) segmenting said three-dimensional block of pixels using a three-dimensional region-merging segmentor to form three-dimensional regions in a segmented block of pixels;

(c) concatenating a next frame in said sequence of frames to said segmented three-dimensional block of pixels;

(d) outputting a segmented two-dimensional frame from said segmented three-dimensional block of pixels; and

(e) repeating steps (b) to (d) for each subsequent frame of said sequence.

2. A method as claimed in claim 1 wherein said sequence of frames is from a video feed and said output frame is the oldest segmented frame in time.

3. A method as claimed in claim 2 comprising the further step of:

(d1) tracking regions in said sequence of frames by tracking corresponding regions in said output frames.

4. A method as claimed in any one of claims 1 to 3 wherein each pixel is further described by a corresponding error covariance, and step (b) comprises the sub-steps of:

(b1) for each said pixel, fitting each said component and the corresponding error covariance to a predetermined linear model to obtain a set of model parameters and corresponding confidence representations;

(b2) statistically analysing said sets of model parameters and corresponding confidence representations to derive said segmented block of pixels that minimises a predetermined cost function.

5. A method according to claim 4 wherein step (b2) comprises the sub-steps of:

merging neighboring regions in an order using said sets of model parameters and confidence representations until an optimal merging criterion is reached, said order being determined by a variational cost function.

6. A method according to claim 5 wherein said order is determined by dividing a minimum covariance-normalised vector distance between adjacent regions of said segmentation by an area of a common boundary between adjacent regions, and ordering the resulting quotients.

7. A method according to claim 5 whereby regions having more than a predetermined number of neighbouring regions are excluded from merging.

8. A method according to claim 4 wherein step (b2) comprises the sub-steps of:

arranging a sequence of neighbouring region pairs in an order using said sets of model parameters and confidence representations, said order being determined by a variational cost function;

determining whether a merger of a first pair of neighbouring regions in said sequence would result in a configuration wherein every cross-section along the time axis provides a single two-dimensional connected component; and

merging said pair of neighbouring regions if said merger would result in said configuration, or alternatively moving said pair of neighbouring regions to a substantially last position in said sequence.

9. A method according to any one of claims 1 to 8 comprising the further step of:

maintaining a two-dimensional array of pointers to regions in said segmented block of frames, said array having entries corresponding with said pixels, thereby allowing a constant time look-up of neighbouring regions when concatenating a new frame in step (c).

10. A method according to claim any one of claims 1 to 9 wherein step (c) comprises the sub-steps of:

dividing said next frame into tiled square regions of pixels;

quadtree decomposing said tiled regions if said pixel components are not within a predetermined variance; and

adding said tiled regions to said segmented block of pixels.

11. A method according to any one of claims 1 to 10, wherein said plurality of vector components comprise at least two of colour, range and motion.

12. An apparatus for segmenting a sequence of two-dimensional frames, each frame being formed by a plurality of pixels, each said pixel being described by a vector having components each relating to a different measured image characteristic, said apparatus comprising:

means for forming a three-dimensional block of said pixels from the first predetermined number of said frames on said sequence of frames;

segmenting means for segmenting said three-dimensional block of pixels using a three-dimensional region-merging segmentor to form three-dimensional regions in a segmented block of pixels;

concatenating means for concatenating a next frame in said sequence of frames to said segmented three-dimensional block of pixels;

output means for outputting a segmented two-dimensional frame from said segmented three-dimensional block of pixels; and

means for activating said segmenting means, said concatenating means and said output means for each subsequent frame of said sequence.

13. An apparatus as claimed in claim 12 wherein said sequence of frames is from a video feed and said output frame is the oldest segmented frame in time.

14. An apparatus as claimed in claim 13 further comprising:

means for tracking regions in said sequence of frames by tracking corresponding regions in said output frames.

15. An apparatus as claimed in any one of claims 12 to 14 wherein each pixel is further described by a corresponding error covariance, and said segmenting means comprises:

fitting means for, for each said pixel, fitting each said component and the corresponding error covariance to a predetermined linear model to obtain a set of model parameters and corresponding confidence representations;

analysing means for statistically analysing said sets of model parameters and corresponding confidence representations to derive said segmented block of pixels that minimises a predetermined cost function.

16. An apparatus according to claim 15 wherein said analysing means comprises:

means for merging neighboring regions in an order using said sets of model parameters and confidence representations until an optimal merging criterion is reached, said order being determined by a variational cost function.

17. An apparatus according to claim 16 wherein said order is determined by dividing a minimum covariance-normalised vector distance between adjacent regions of said segmentation by an area of a common boundary between adjacent regions, and ordering the resulting quotients.

18. An apparatus according to claim 16 whereby regions having more than a predetermined number of neighbouring regions are excluded from merging.

19. An apparatus according to claim 15 wherein said analysing means comprises:

means for arranging a sequence of neighboring region pairs in an order using said sets of model parameters and confidence representations, said order being determined by a variational cost function;

means for determining whether a merger of a first pair of neighboring regions in said sequence would result in a configuration wherein every cross-section along the time axis provides a single two-dimensional connected component; and

means for merging said pair of neighboring regions if said merger would result in said configuration, or alternatively moving said pair of neighboring regions to a substantially last position in said sequence.

20. An apparatus according to claim any one of claims 12 to 19 wherein said concatenating means comprises:

means for dividing said next frame into tiled square regions of pixels;

means for quadtree decomposing said tiled regions if said pixel components are not within a predetermined variance; and

means for adding said tiled regions to said segmented block of pixels.

21. A program stored in a memory medium for segmenting a sequence of two-dimensional frames, each frame being formed by a plurality of pixels, each said pixel being described by a vector having components each relating to a different measured image characteristic, said program comprising:

code for forming a three-dimensional block of said pixels from the first predetermined number of said frames on said sequence of frames;

code for segmenting said three-dimensional block of pixels using a three-dimensional region-merging segmentor to form three-dimensional regions in a segmented block of pixels;

code for concatenating a next frame in said sequence of frames to said segmented three-dimensional block of pixels;

code for outputting a segmented two-dimensional frame from said segmented three-dimensional block of pixels; and

code for repeating said code for segmenting, code for concatenating and code for outputting for each subsequent frame of said sequence.

22. A method of segmenting an image, said method being substantially as described herein with reference to Figs. 1 to 9 of the drawings.

23. An apparatus for segmenting an image, said apparatus being substantially as described herein with reference to Figs. 1 to 9 of the drawings.

24. A program for segmenting an image, said program being substantially as described herein with reference to Figs. 1 to 9 of the drawings.

Dated 24 December, 2001
CANON KABUSHIKI KAISHA
Patent Attorneys for the Applicant/Nominated Person
SPRUSON & FERGUSON

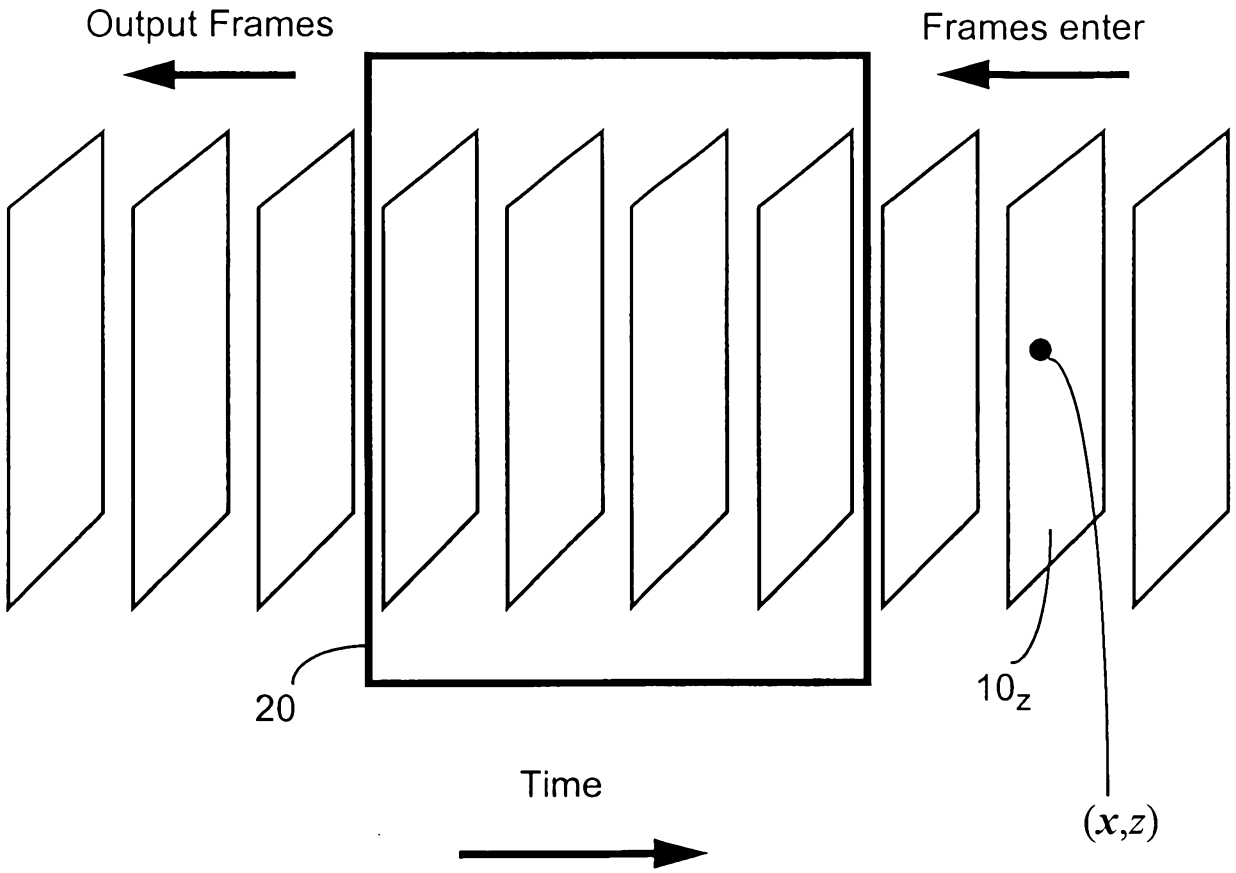

Fig. 1

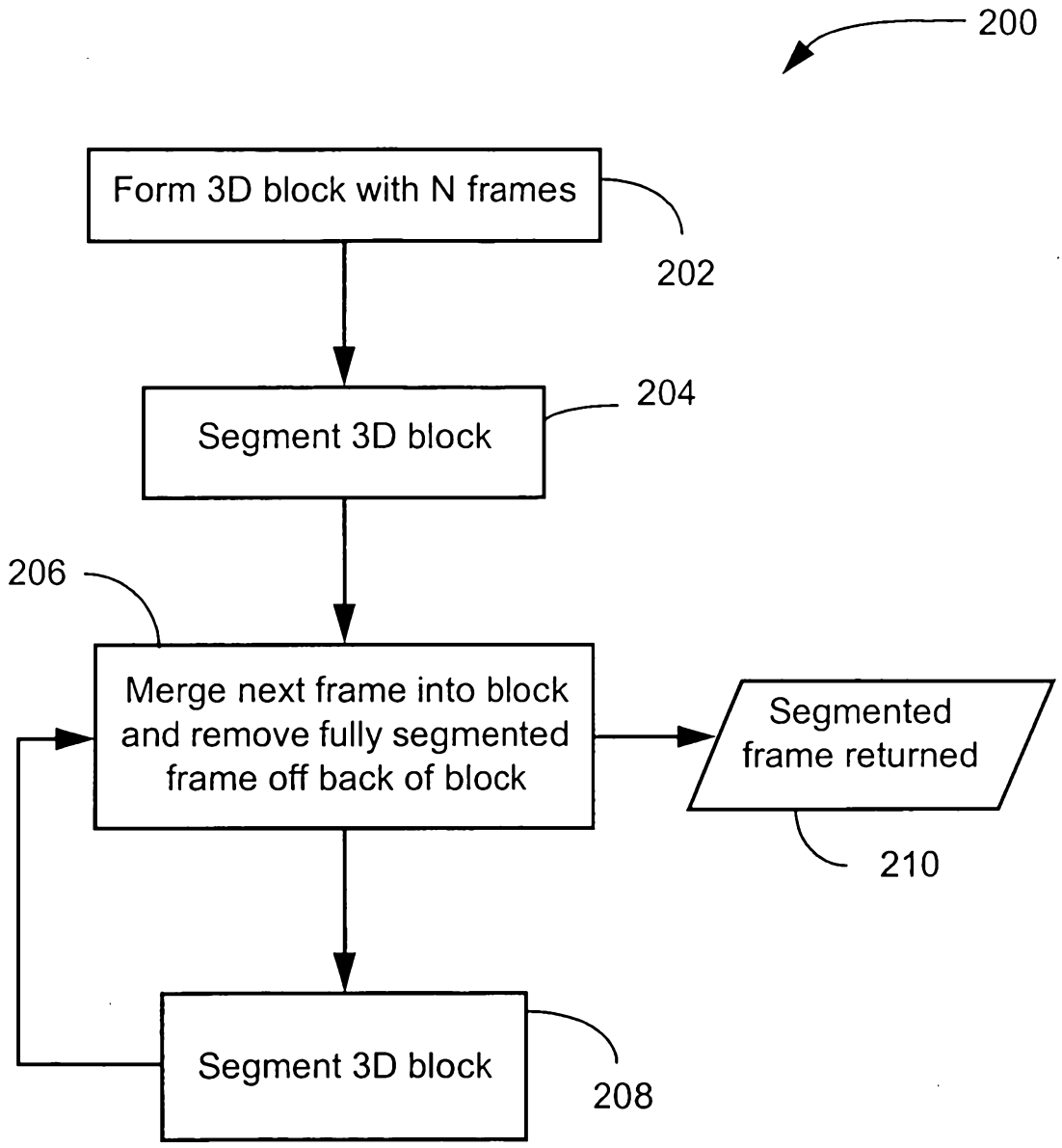
200
Form 3D block with N frames
202
Segment 3D block
204
206
Merge next frame into block and remove fully segmented frame off back of block
Segmented frame returned
210
Segment 3D block
208

Fig. 2

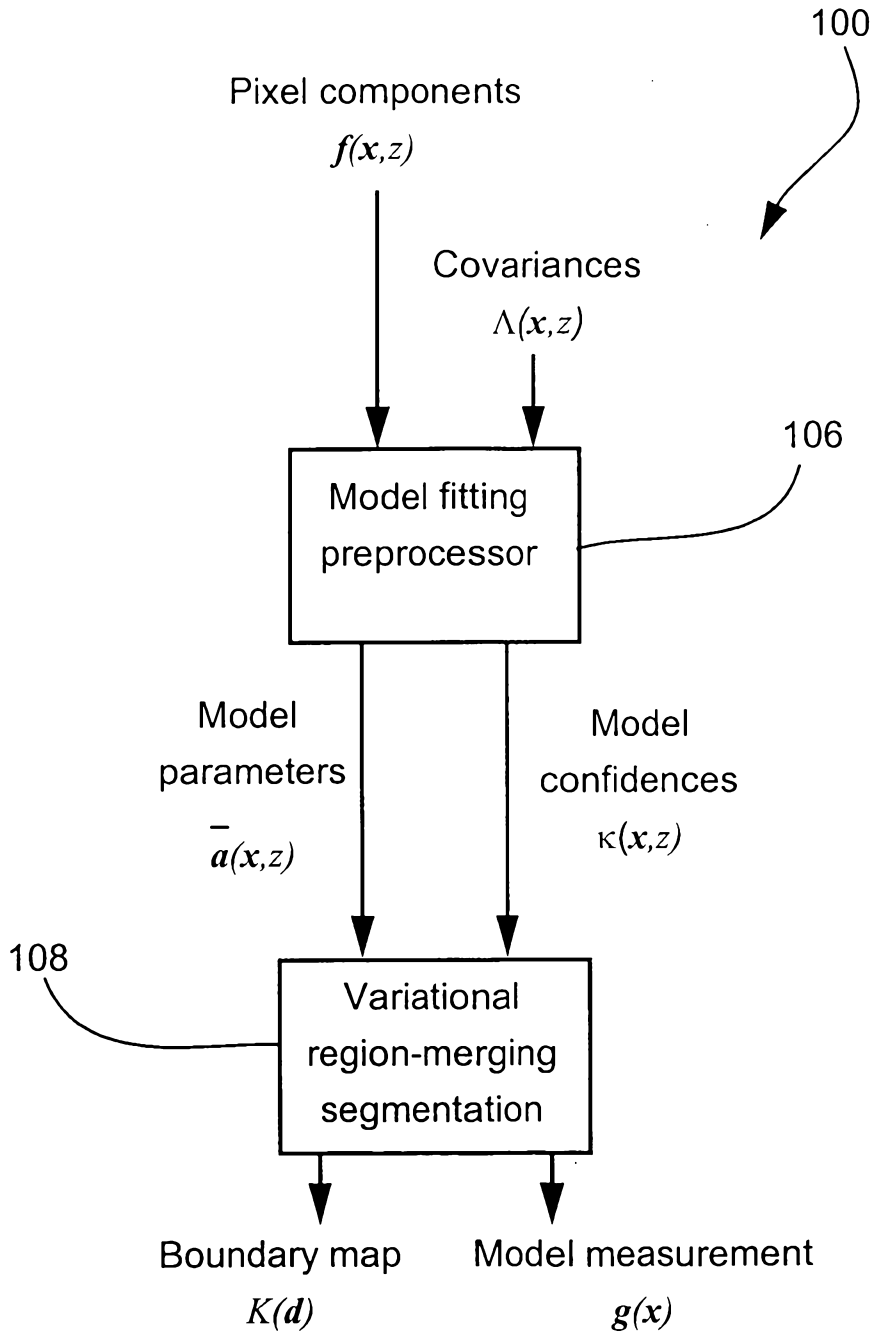
100
Pixel components
f(x,z)
Covariances
Λ(x,z)
106
Model fitting preprocessor
Model parameters
ā(x,z)
Model confidences
κ(x,z)
108
Variational region-merging segmentation
Boundary map
K(d)
Model measurement
g(x)

Fig. 3

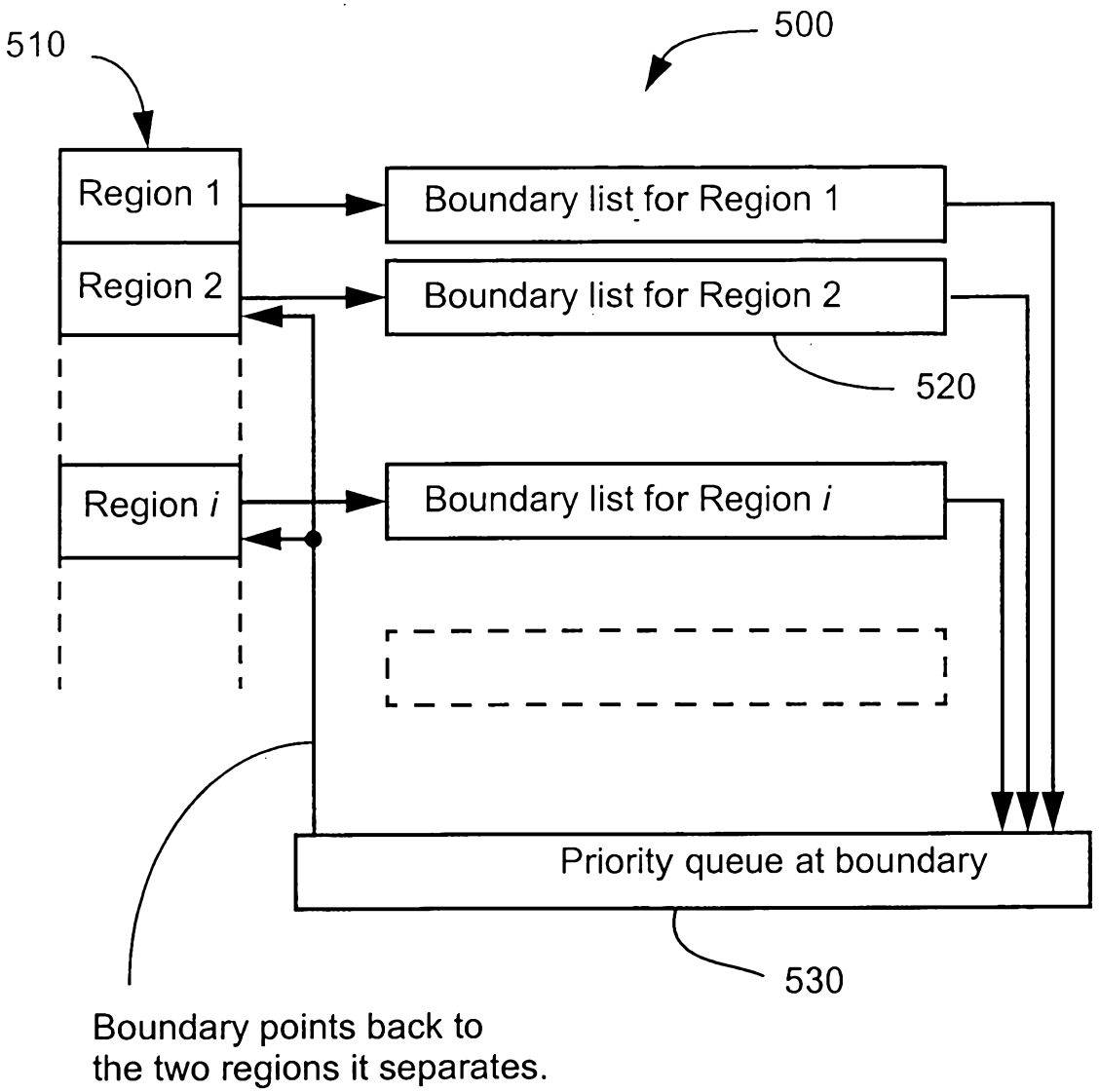
510
500
Region 1
Region 2
Region i
Boundary list for Region 1
Boundary list for Region 2
520
Boundary list for Region i
Priority queue at boundary
530
Boundary points back to
the two regions it separates.

Fig. 4

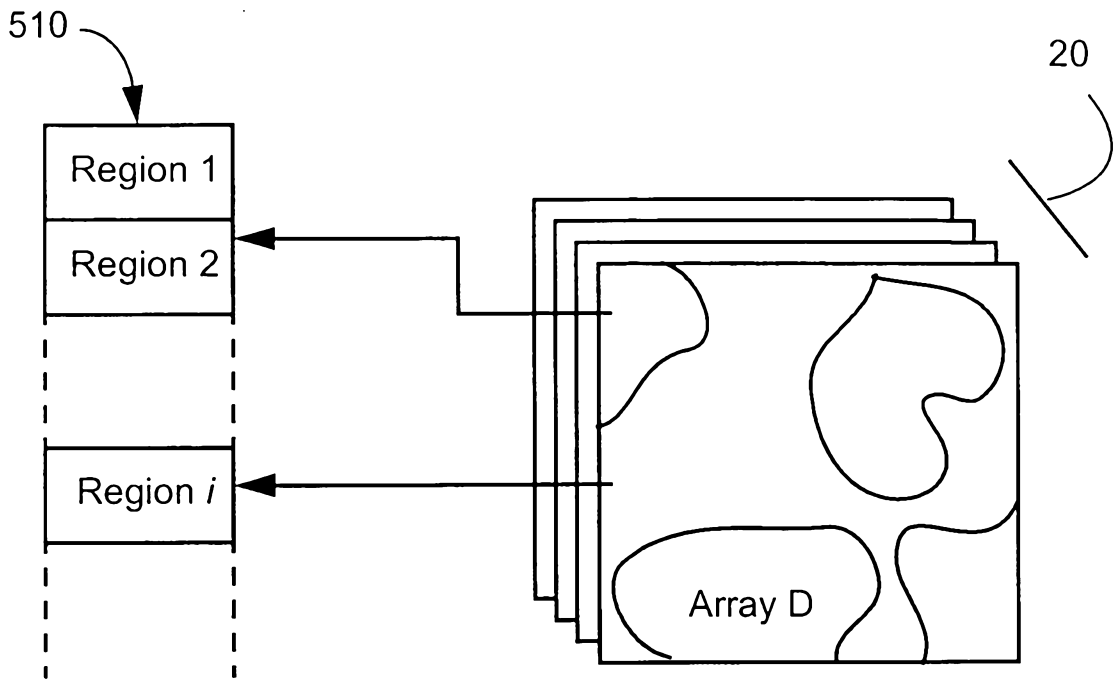

Fig. 5

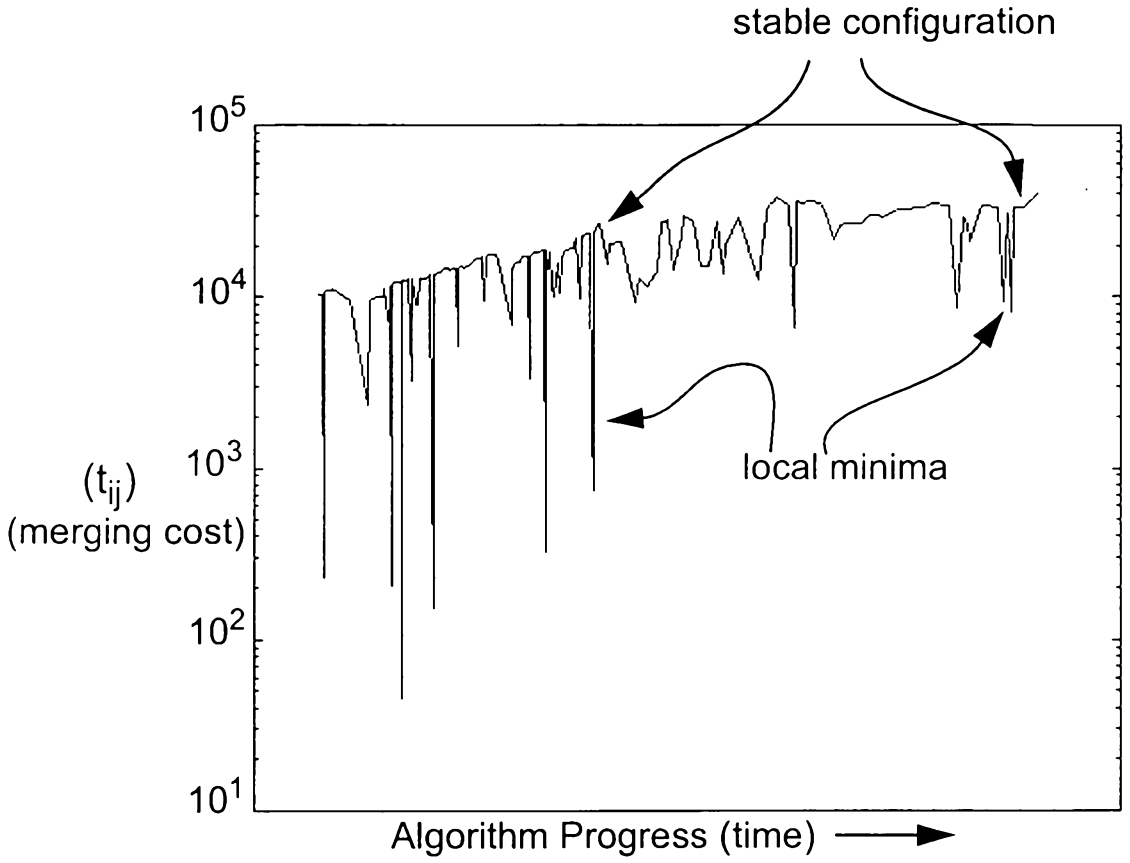
stable configuration
10^5
10^4
10^3
10^2
10^1
(t_{ij})
(merging cost)
local minima
Algorithm Progress (time)

Fig. 6A

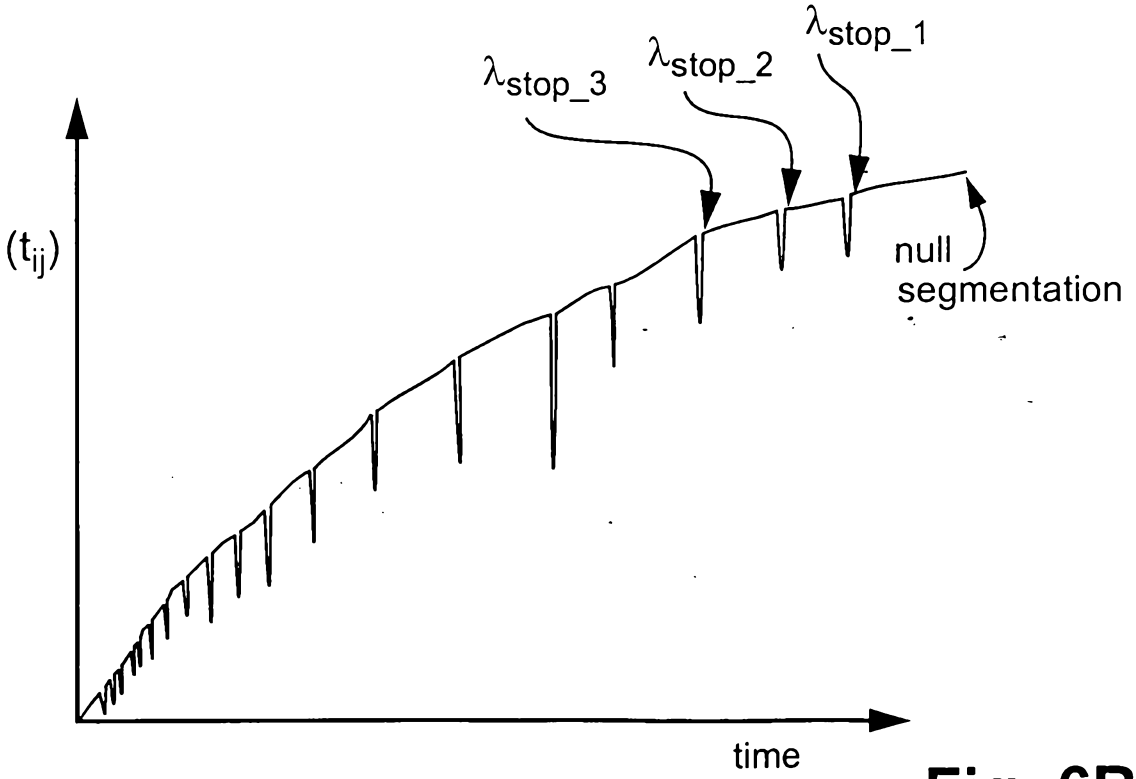
λ_{stop_1}
λ_{stop_2}
λ_{stop_3}
(t_{ij})
null
segmentation
time

Fig. 6B

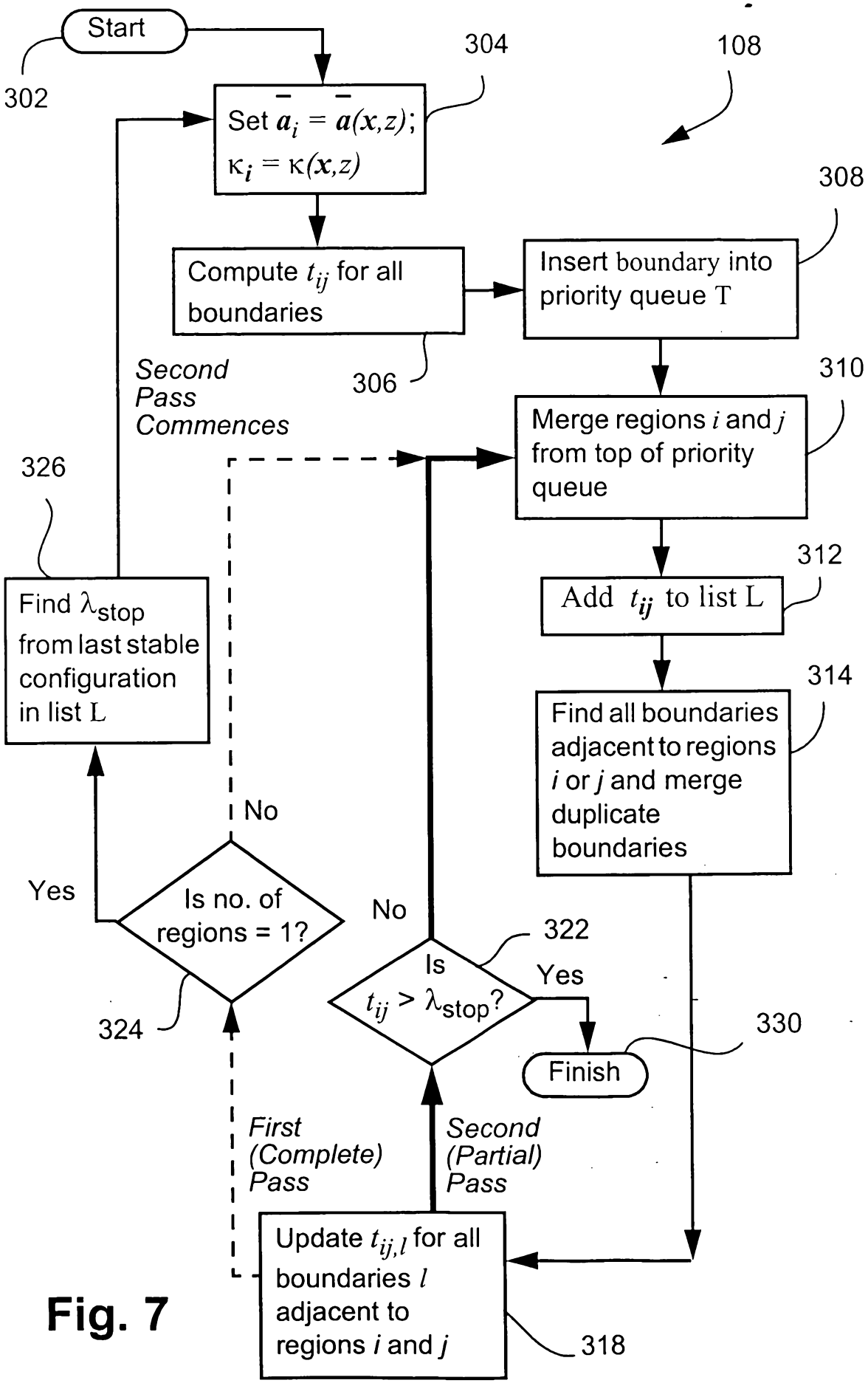
Start
302
304
Set $\bar{a}_i = \bar{a}(x,z)$; $\kappa_i = \kappa(x,z)$
108
308
Compute t_{ij} for all boundaries
Insert boundary into priority queue T
306
Second Pass Commences
310
Merge regions i and j from top of priority queue
326
312
Add t_{ij} to list L
Find λ_{stop} from last stable configuration in list L
314
Find all boundaries adjacent to regions i or j and merge duplicate boundaries
No
Yes
Is no. of regions = 1?
No
322
Is $t_{ij} > \lambda_{stop}$?
Yes
330
Finish
324
First (Complete) Pass
Second (Partial) Pass
Update $t_{ij,l}$ for all boundaries l adjacent to regions i and j
318

Fig. 7

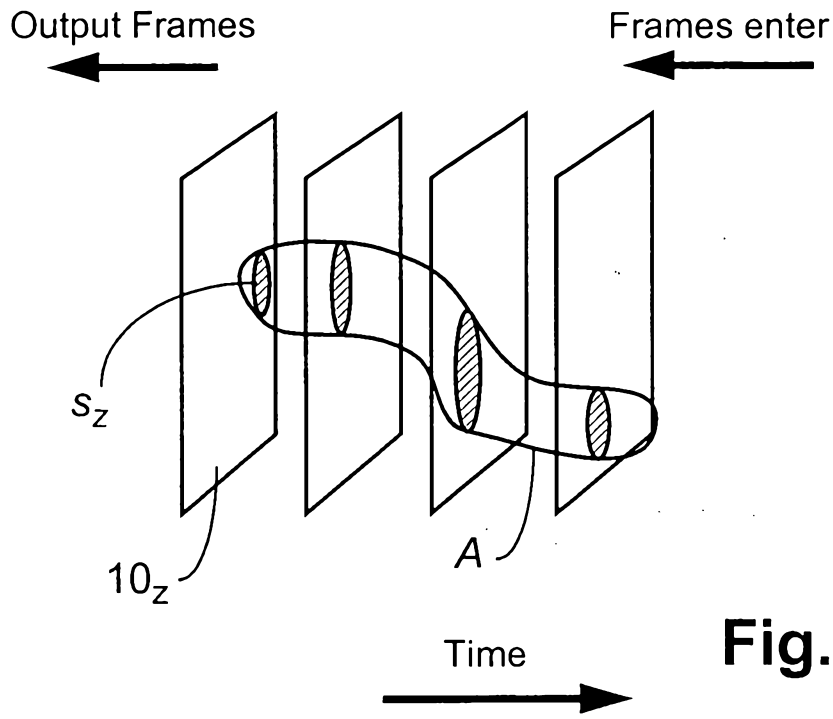
Output Frames
Frames enter
s_z
10_z
A
Time

Fig. 8A

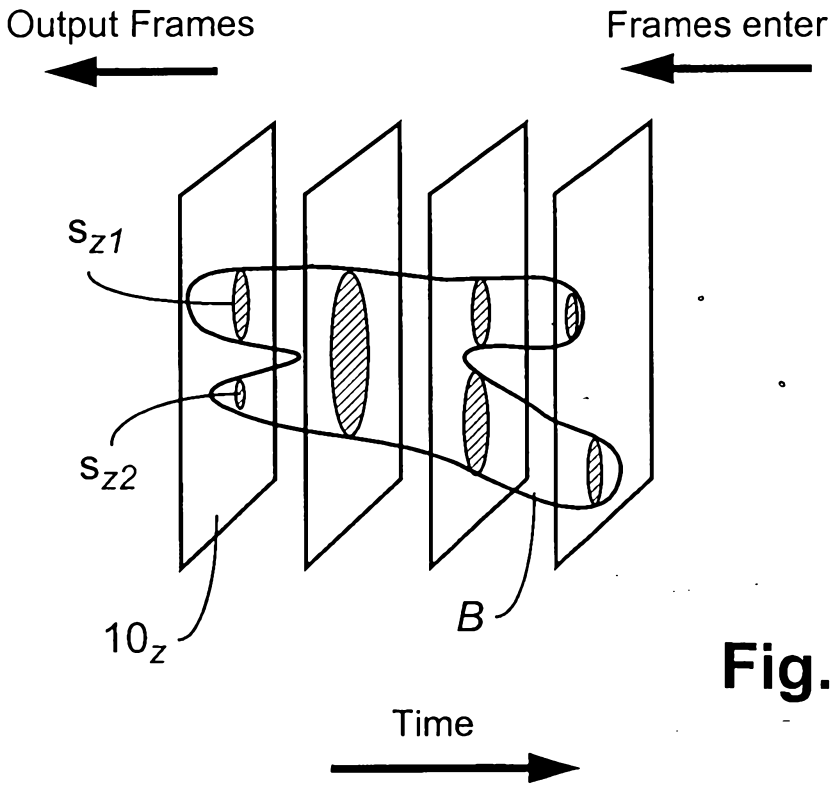
Output Frames
Frames enter
s_{z1}
s_{z2}
10_z
B
Time

Fig. 8B

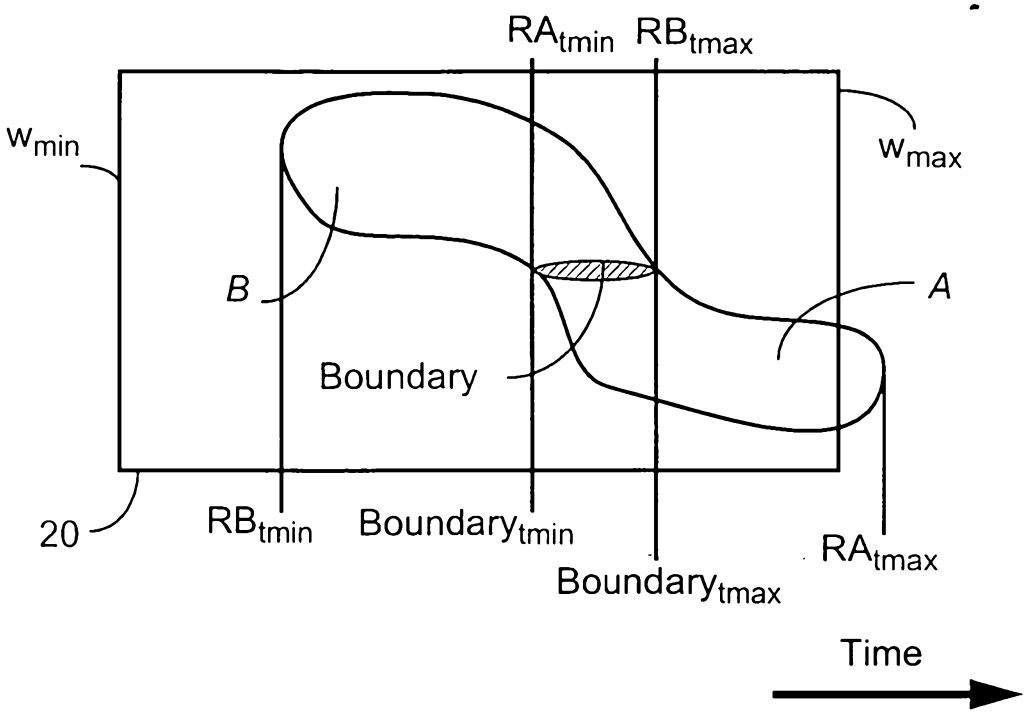
RA$_{tmin}$ RB$_{tmax}$
w$_{min}$
w$_{max}$
B
A
Boundary
20
RB$_{tmin}$
Boundary$_{tmin}$
Boundary$_{tmax}$
RA$_{tmax}$
Time

Fig. 8C

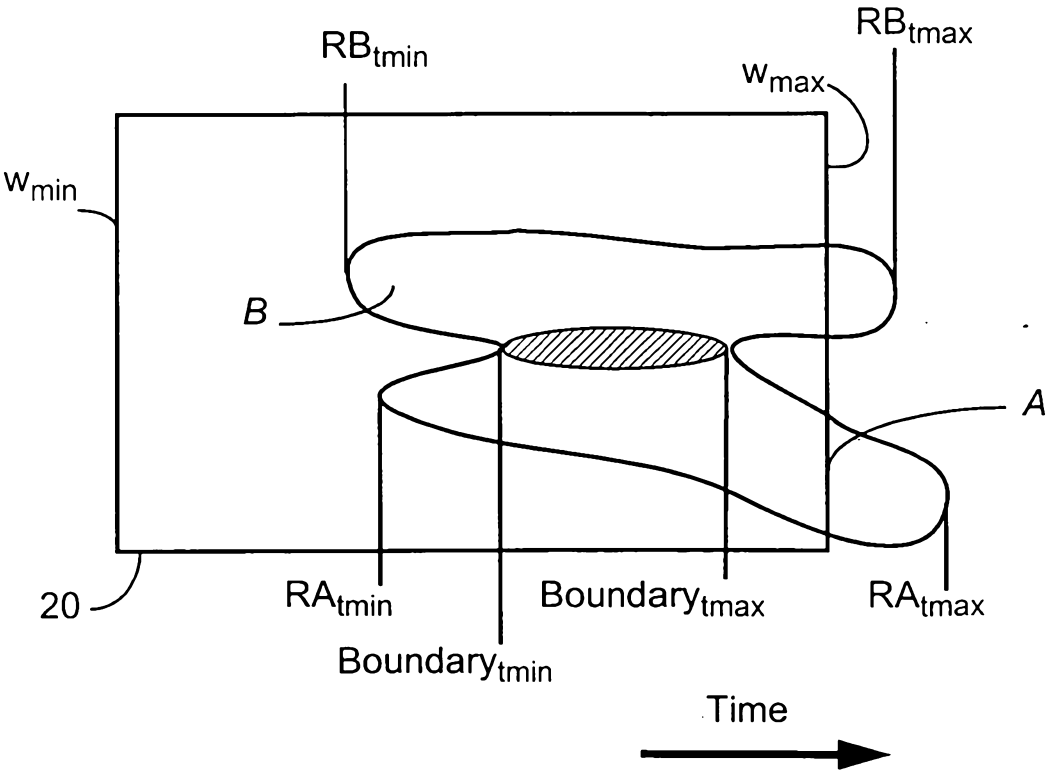
RB$_{tmin}$
RB$_{tmax}$
w$_{max}$
w$_{min}$
B
A
20
RA$_{tmin}$
Boundary$_{tmax}$
RA$_{tmax}$
Boundary$_{tmin}$
Time

Fig. 8D

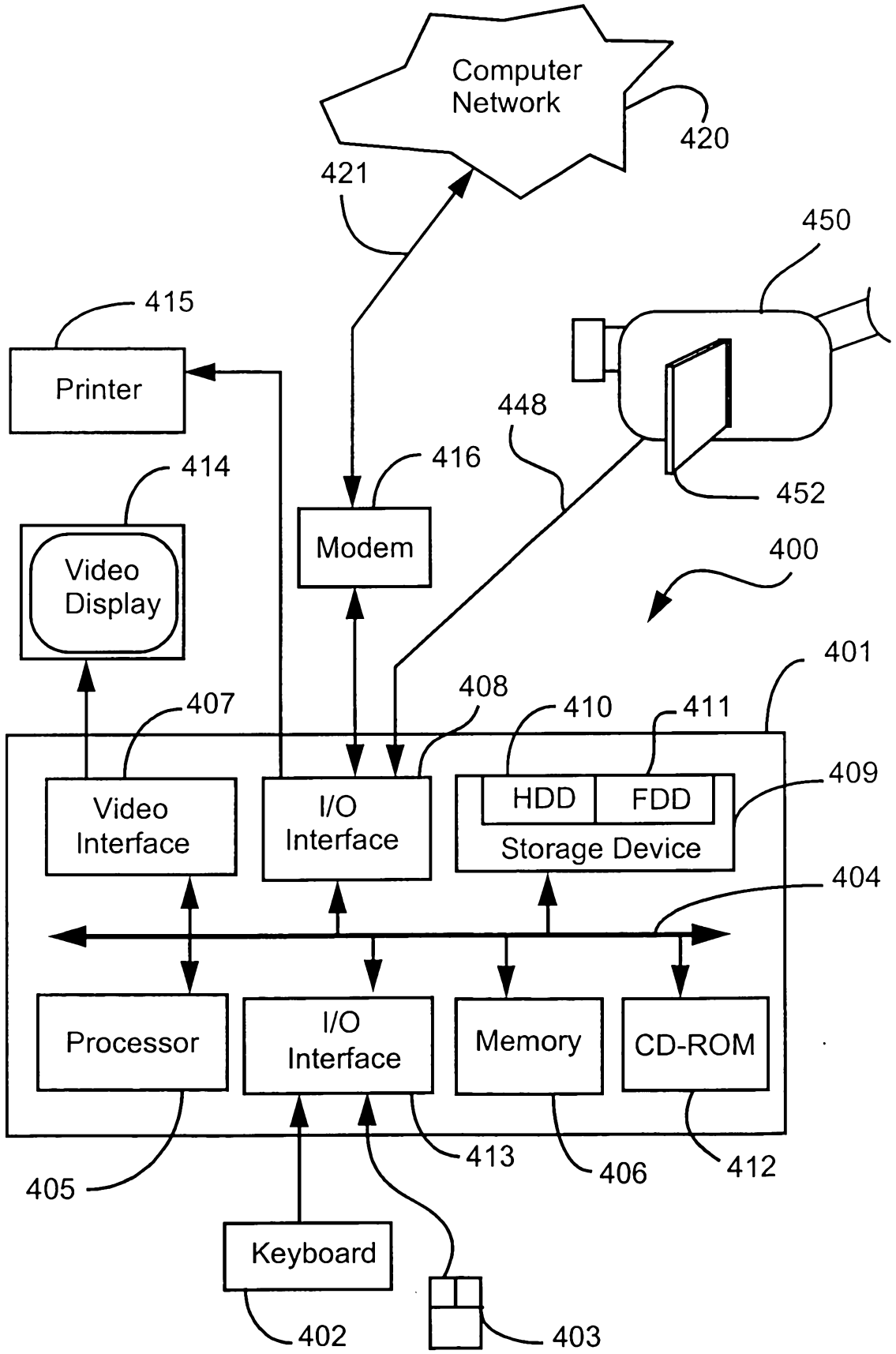
Computer Network
420
421
450
415
Printer
448
414
416
452
Video Display
Modem
400
401
408
407
410
411
409
Video Interface
I/O Interface
HDD
FDD
Storage Device
404
Processor
I/O Interface
Memory
CD-ROM
413
406
412
405
Keyboard
402
403

Fig. 9