US006535632B1

(12) **United States Patent**
Park et al.

(10) **Patent No.:** **US 6,535,632 B1**
(45) **Date of Patent:** **Mar. 18, 2003**

(54) **IMAGE PROCESSING IN HSI COLOR SPACE USING ADAPTIVE NOISE FILTERING**

(75) Inventors: **Hyun Wook Park**, Seattle, WA (US); **Lakshmanan Gopishankar**, Seattle, WA (US); **Shijun Sun**, Seattle, WA (US); **Yongmin Kim**, Seattle, WA (US)

(73) Assignee: **University of Washington**, Seattle, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/386,538**

(22) Filed: **Aug. 30, 1999**

Related U.S. Application Data

(63) Continuation of application No. 09/323,501, filed on Jun. 1, 1999, and a continuation of application No. 09/233,894, filed on Jan. 20, 1999, now Pat. No. 6,272,250, and a continuation of application No. 09/216,691, filed on Dec. 18, 1998, now Pat. No. 6,243,494, and a continuation of application No. 09/216,692, filed on Dec. 18, 1998, now Pat. No. 6,301,387.

(51) **Int. Cl.**[7] **G06K 9/00**; G06K 9/34; G06K 9/40; G06K 9/56; H04N 1/46

(52) **U.S. Cl.** **382/164**; 382/162; 382/165; 382/167; 382/173; 382/205; 382/254; 382/260; 382/266; 382/282; 358/515; 358/520; 358/538

(58) **Field of Search** 382/103, 162, 382/164–167, 173, 181, 191, 205, 232, 239, 242, 243, 248, 254, 260–266, 274–276, 282, 199; 348/30, 169, 606, 625, 630, 649; 358/447, 515, 520, 521, 532; 375/240.02, 240.08, 538

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,819,077	A	*	4/1989	Kikuchi et al.	348/71
5,949,914	A	*	9/1999	Yuen	382/254
5,959,693	A	*	9/1999	Wu et al.	348/624
5,995,164	A	*	11/1999	Tsyrganovich	348/649
6,226,401	B1	*	5/2001	Yamafuji et al.	382/165
6,229,580	B1	*	5/2001	Inoue	348/649

OTHER PUBLICATIONS

Mital, et al discloses "colour vision for industrial applications", IEEE, pp. 519–522, 1989.*

Ladley, et al discloses "fundamentals of true–color image processing", IEEE, pp. 791–795, 1990.*

Lin, et al discloses "color image segmentation using modified HSI system for road following", IEEE, pp. 1998–2003, 1991.*

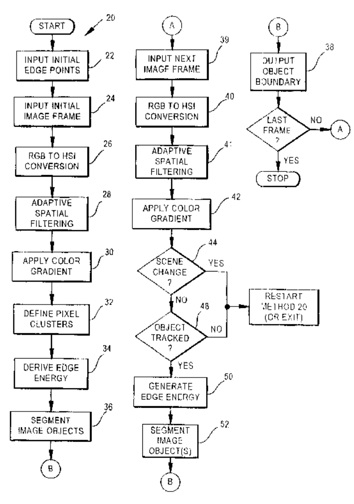
(List continued on next page.)

Primary Examiner—Leo Boudreau
Assistant Examiner—Daniel G. Mariam
(74) *Attorney, Agent, or Firm*—Steven P. Koda, Esq.

(57) **ABSTRACT**

Adaptive noise filtering is applied to an image frame of HSI data to reduce and more uniformly distribute noise while preserving image feature edges. An adaptive spatial filter includes a plurality of averaging kernels. An appropriate kernel is selected for each pixel for each of the hue and saturation components. A set of thresholds are defined for selecting the kernel for the hue component. Another set of thresholds are defined for selecting the kernel for the saturation component. The kernel for the saturation component is selected by comparing the intensity component to the saturation component thresholds. The kernel for the hue component is selected by comparing the product of intensity component and the saturation component to the hue component thresholds. A color gradient operation is applied to the filtered HSI data to aid in detecting image object boundaries. Object segmentation and other image processing techniques may be performed on the filtered HSI data.

32 Claims, 13 Drawing Sheets



OTHER PUBLICATIONS

Welch, et al discloses "image processing using the HSI color space", IEEE, pp. 722–725, 1991.*

Parsiani discloses "isolated noise detection and reduction in color images", IEEE, pp. 741–743, 1993.*

Sobottka, et al discloses "segmentation and tracking of faces in color images", IEEE, pp. 236–241, 1996.*

Sparling, et al discloses "photoelastic analysis using chromatic interpretation of digitized video", IEEE, pp. 417–418, 1997.*

Terillon, et al discloses "automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments", IEEE, pp. 112–117, 1998.*

Herodotou, et al discloses "a color segmentation scheme for object–based video coding", IEEE, pp. 25–29, 1998.*

Carron et al.; "Color Edge Detector Using Jointly Hue, Saturation and Intensity," IEEE Publ. 0–8186–6950–0/94; 1994.

Gonzalez et al.; "Digital Image Processing," Addison–Wesley Pub Co.; pp. 228–237; 1992.

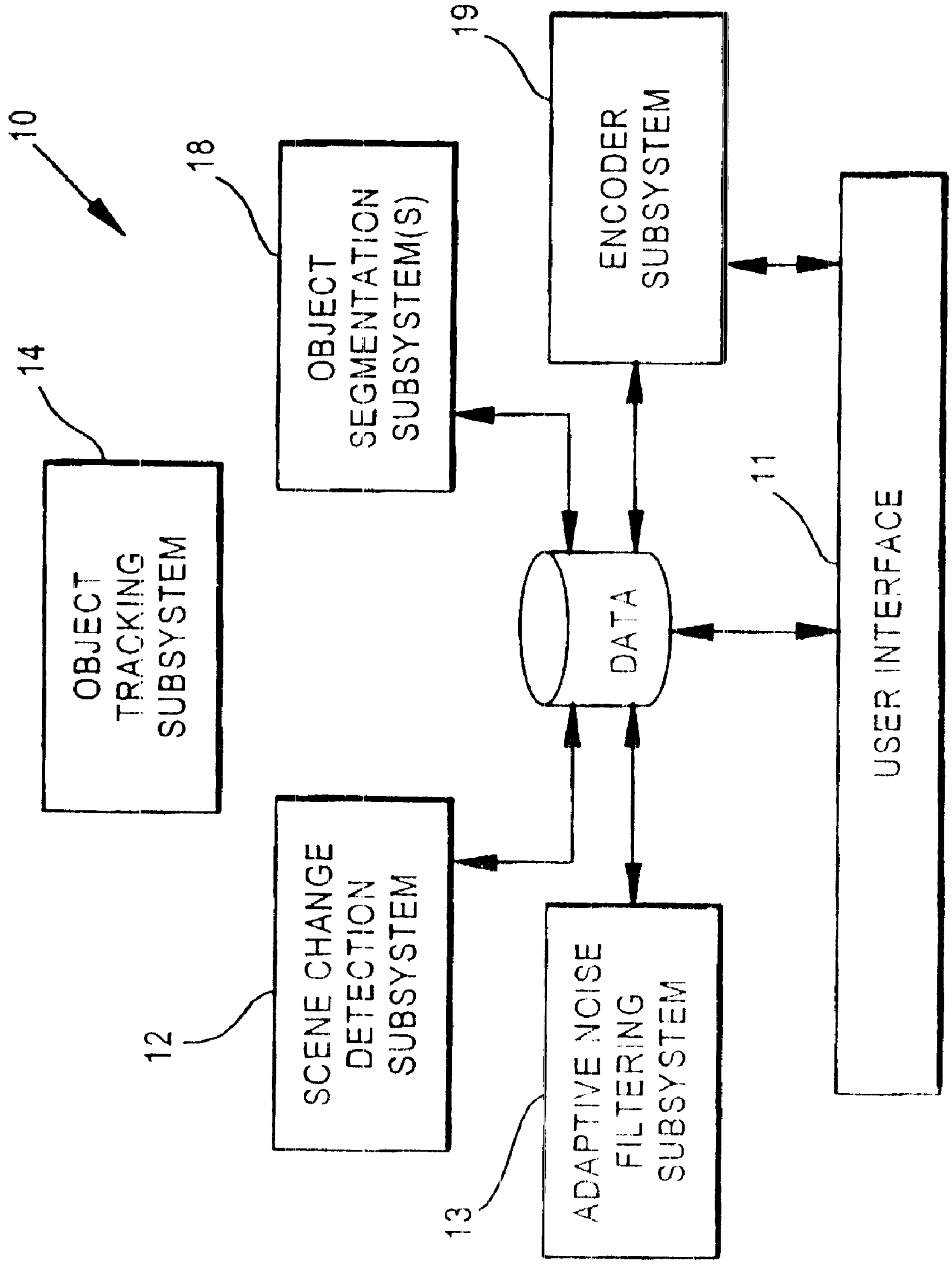
Canny, John; "A Computational Approach to Edge Detection;" IEEE Transactions on Pattern Analysis and MAchine Intelligence, vol. PAMI–8, No. 6; Nov. 1986.

Sapiro, Guillermo; "Note Color Snakes;" Computer Vision and Image Understanding, vol. 68, No. 2; pp. 247–253; Nov. 1997.

Papoulis, Athanasios; "Probality, Random Variables, and Stochastic Processes" Third Edition, McGraw–Hill, Inc.; 1991.

* cited by examiner

FIG. 1


10
14
OBJECT TRACKING SUBSYSTEM
18
OBJECT SEGMENTATION SUBSYSTEM(S)
19
ENCODER SUBSYSTEM
DATA
11
USER INTERFACE
12
SCENE CHANGE DETECTION SUBSYSTEM
13
ADAPTIVE NOISE FILTERING SUBSYSTEM

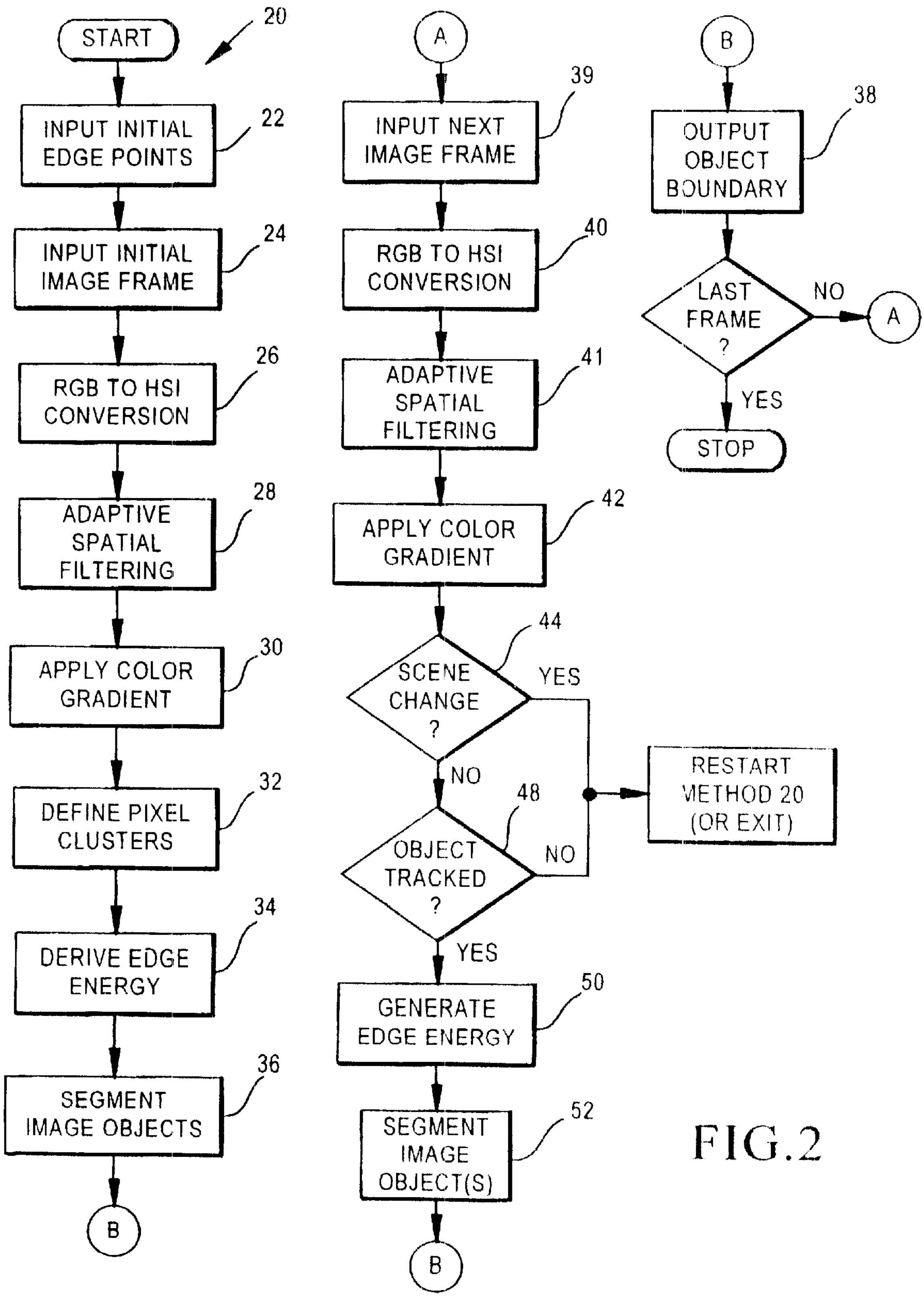

20
START
INPUT INITIAL EDGE POINTS
22
INPUT INITIAL IMAGE FRAME
24
RGB TO HSI CONVERSION
26
ADAPTIVE SPATIAL FILTERING
28
APPLY COLOR GRADIENT
30
DEFINE PIXEL CLUSTERS
32
DERIVE EDGE ENERGY
34
SEGMENT IMAGE OBJECTS
36
B
A
INPUT NEXT IMAGE FRAME
39
RGB TO HSI CONVERSION
40
ADAPTIVE SPATIAL FILTERING
41
APPLY COLOR GRADIENT
42
SCENE CHANGE ?
44
YES
NO
OBJECT TRACKED ?
48
NO
YES
RESTART METHOD 20 (OR EXIT)
GENERATE EDGE ENERGY
50
SEGMENT IMAGE OBJECT(S)
52
B
B
OUTPUT OBJECT BOUNDARY
38
LAST FRAME ?
NO
A
YES
STOP

FIG.2

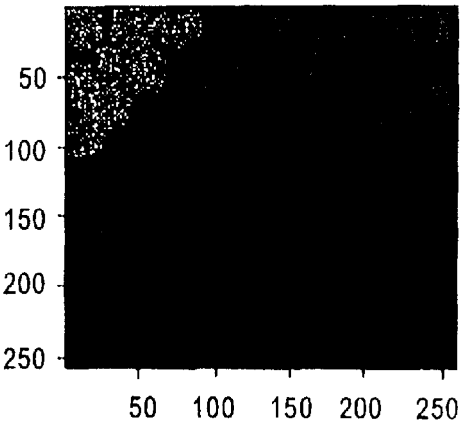


FIG.3A

FIG.3B

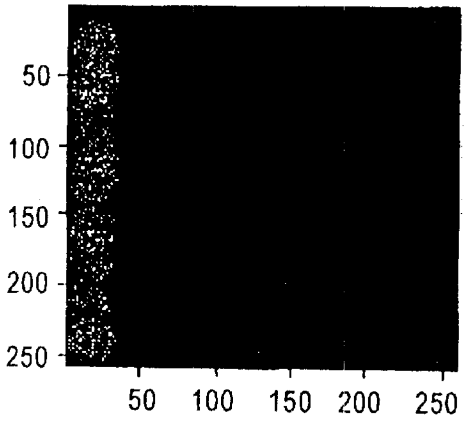
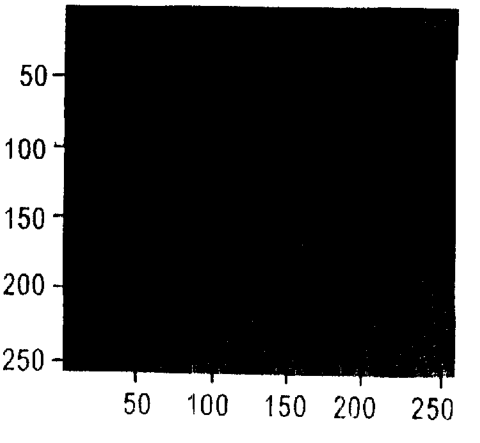


FIG.3C



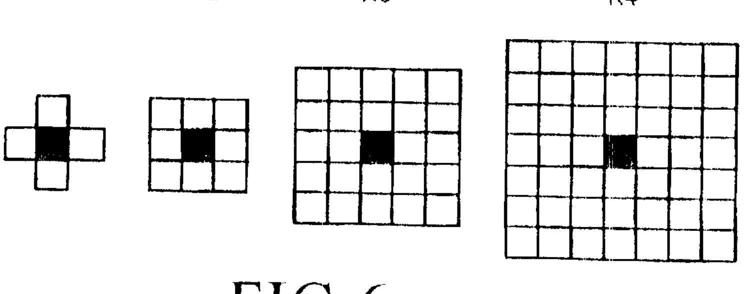
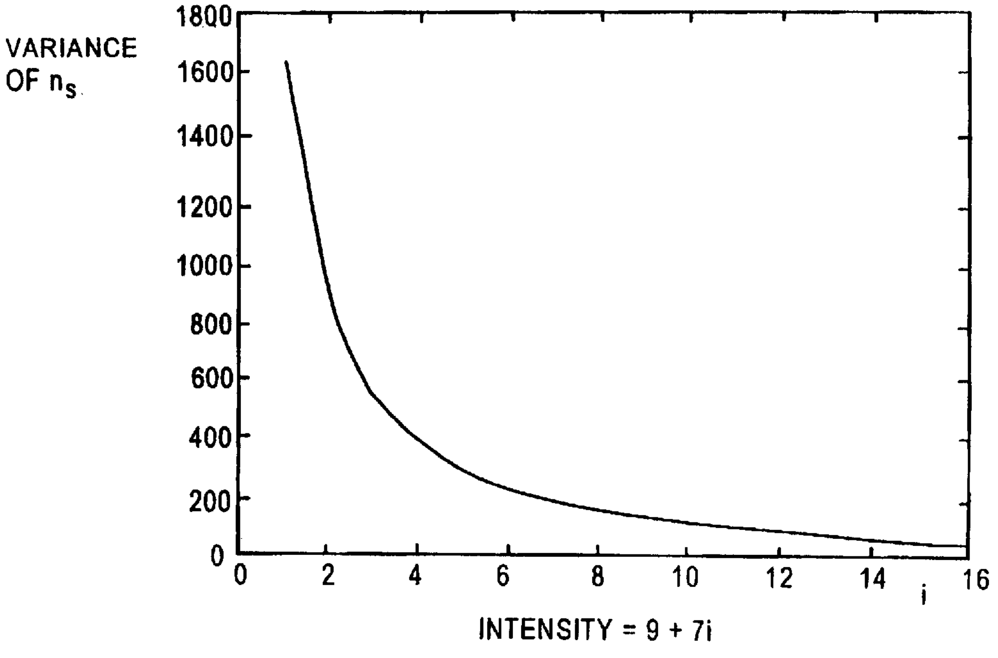


FIG.6

FIG.4


VARIANCE OF n_s
1800
1600
1400
1200
1000
800
600
400
200
0
0 2 4 6 8 10 12 14 16
i
INTENSITY = 9 + 7i

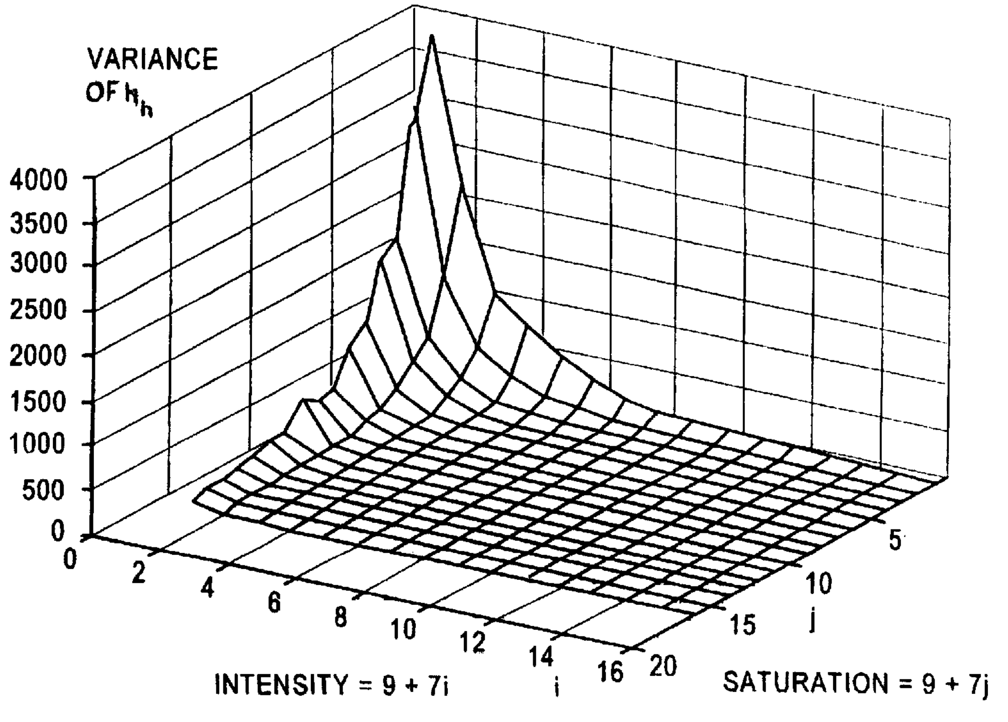

VARIANCE OF n_h
4000
3500
3000
2500
2000
1500
1000
500
0
0 2 4 6 8 10 12 14 16
i
INTENSITY = 9 + 7i
5 10 15 20
j
SATURATION = 9 + 7j

FIG.5

FIG.7

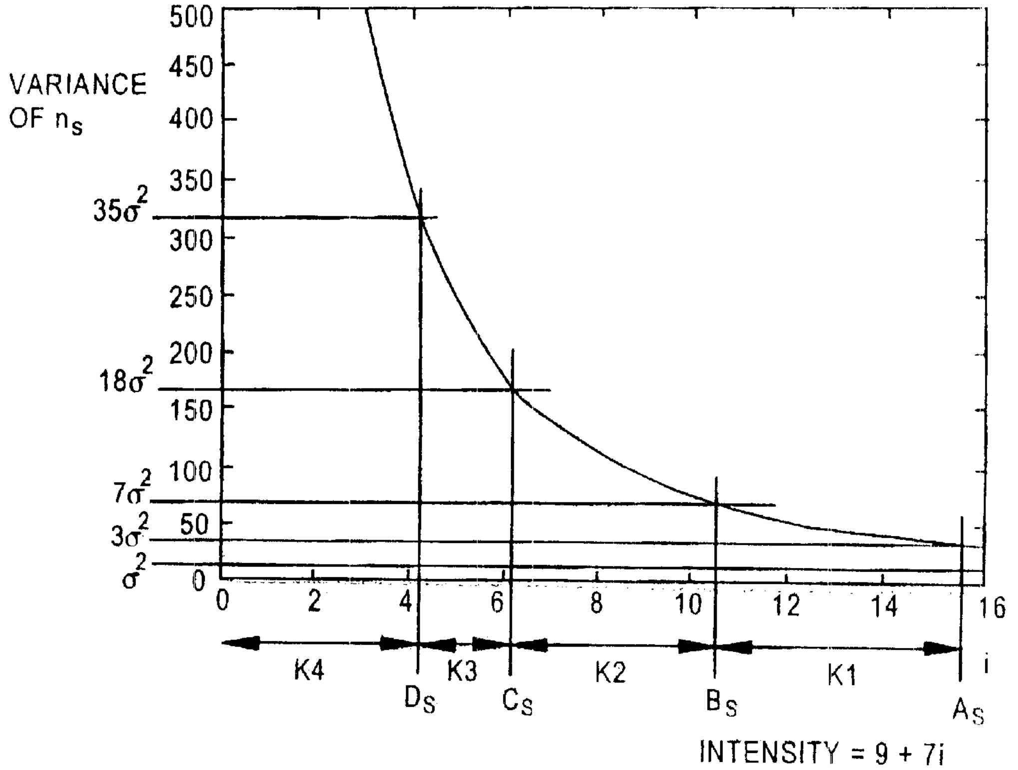


FIG.8A

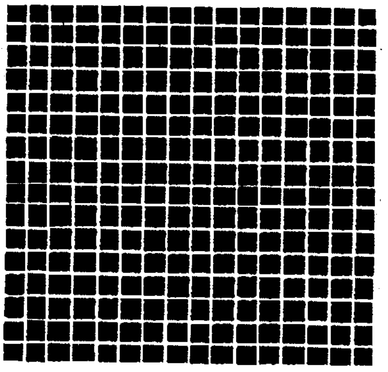


FIG.8B

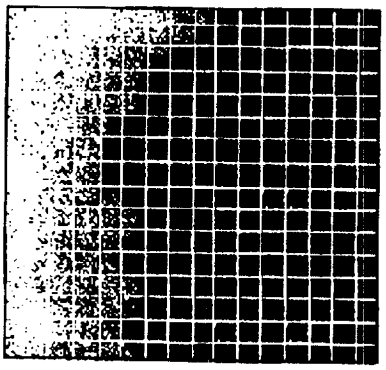
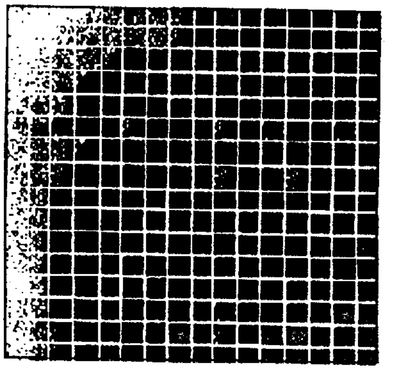


FIG.8C



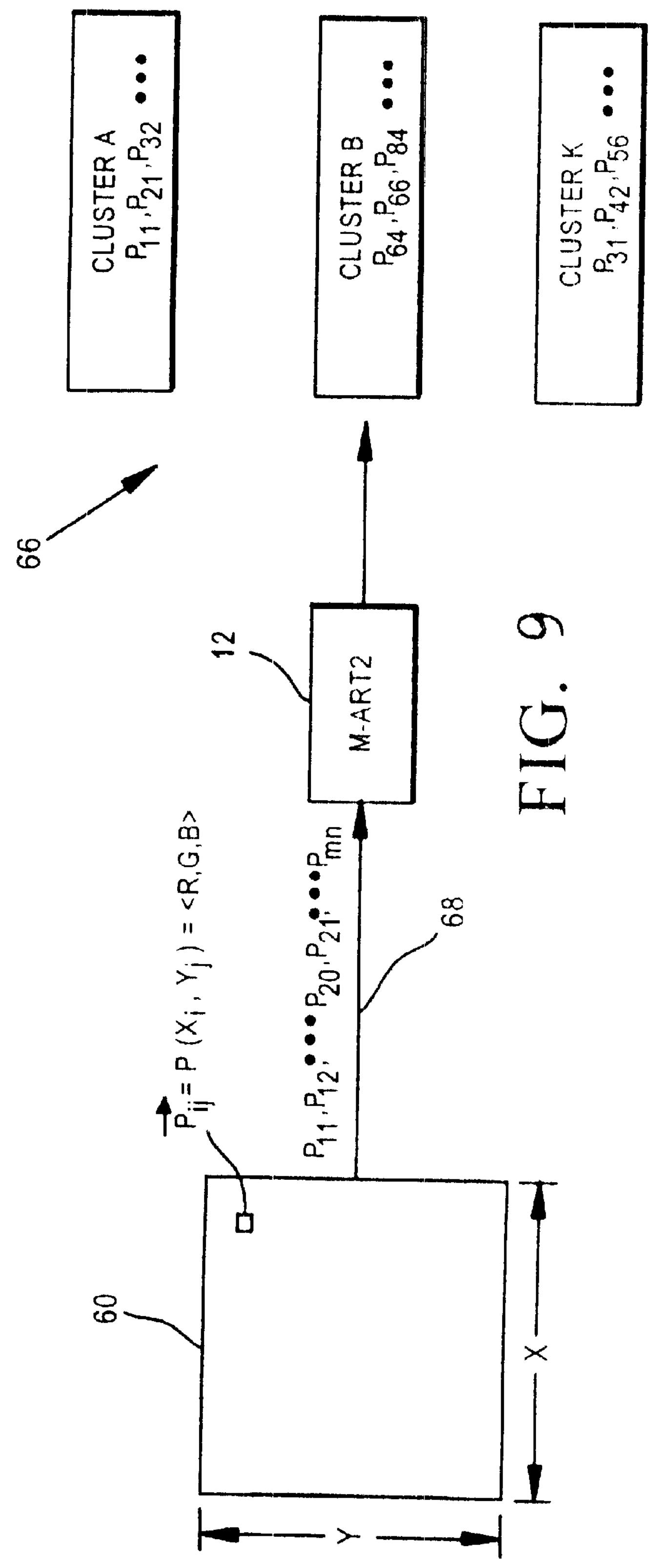
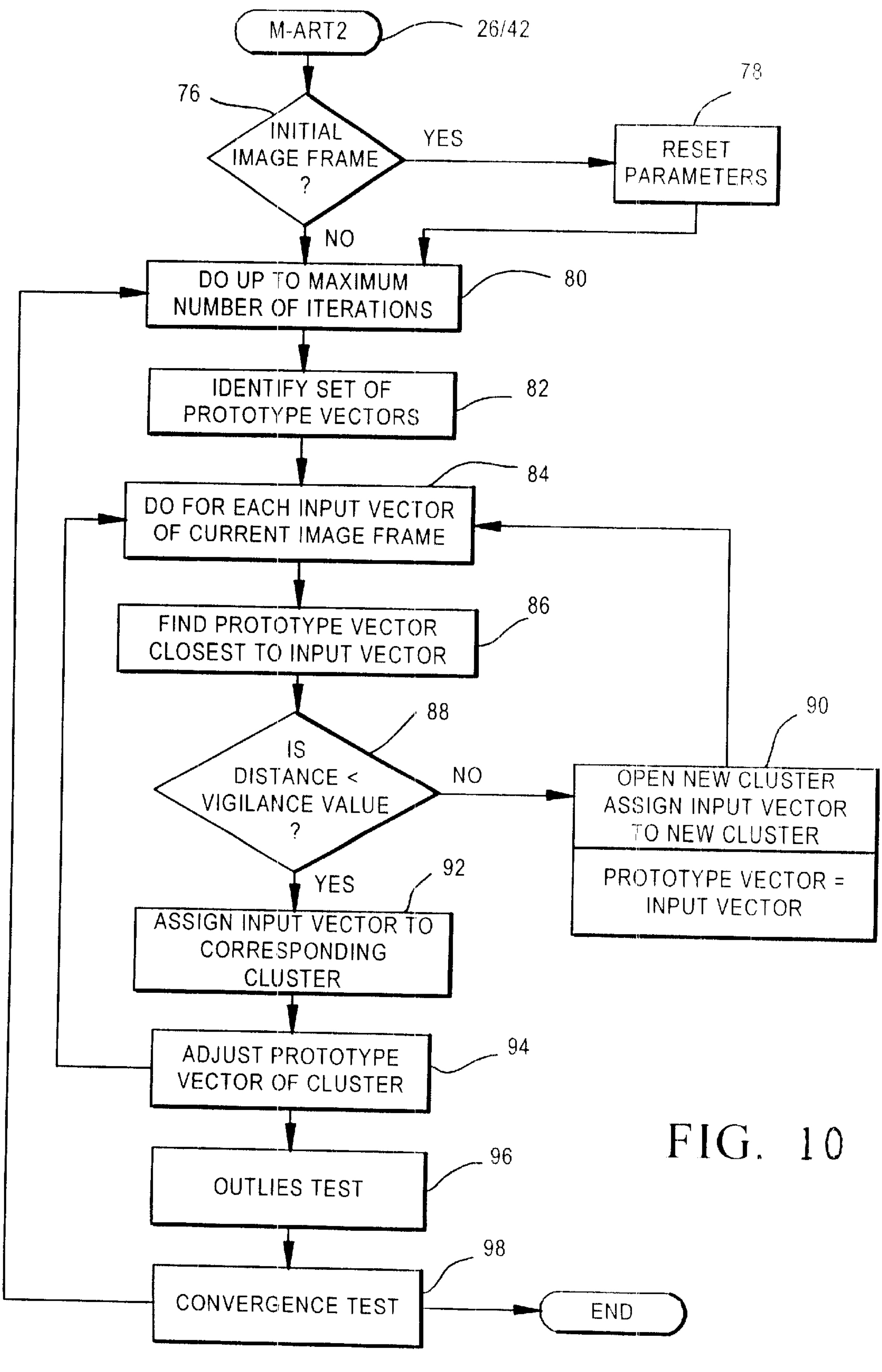

CLUSTER A
P11,P21,P32 •••
CLUSTER B
P64,P66,P84 •••
CLUSTER K
P31,P42,P56 •••
66
12
M-ART2
Pij = P (Xi , Yj) = <R,G,B>
P11,P12,•••P20,P21,•••Pmn
68
60
X
Y

FIG. 9



M-ART2
26/42
76
INITIAL IMAGE FRAME ?
YES
78
RESET PARAMETERS
NO
DO UP TO MAXIMUM NUMBER OF ITERATIONS
80
IDENTIFY SET OF PROTOTYPE VECTORS
82
84
DO FOR EACH INPUT VECTOR OF CURRENT IMAGE FRAME
FIND PROTOTYPE VECTOR CLOSEST TO INPUT VECTOR
86
88
IS DISTANCE < VIGILANCE VALUE ?
NO
90
OPEN NEW CLUSTER ASSIGN INPUT VECTOR TO NEW CLUSTER
PROTOTYPE VECTOR = INPUT VECTOR
YES
92
ASSIGN INPUT VECTOR TO CORRESPONDING CLUSTER
ADJUST PROTOTYPE VECTOR OF CLUSTER
94
OUTLIES TEST
96
98
CONVERGENCE TEST
END

FIG. 10

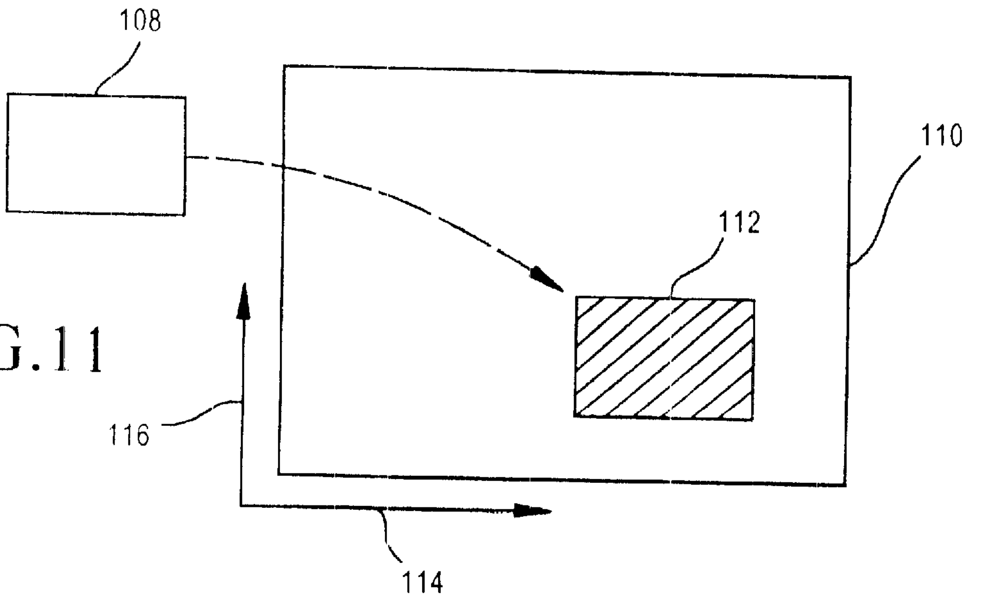

108
110
112
116
114

FIG.11

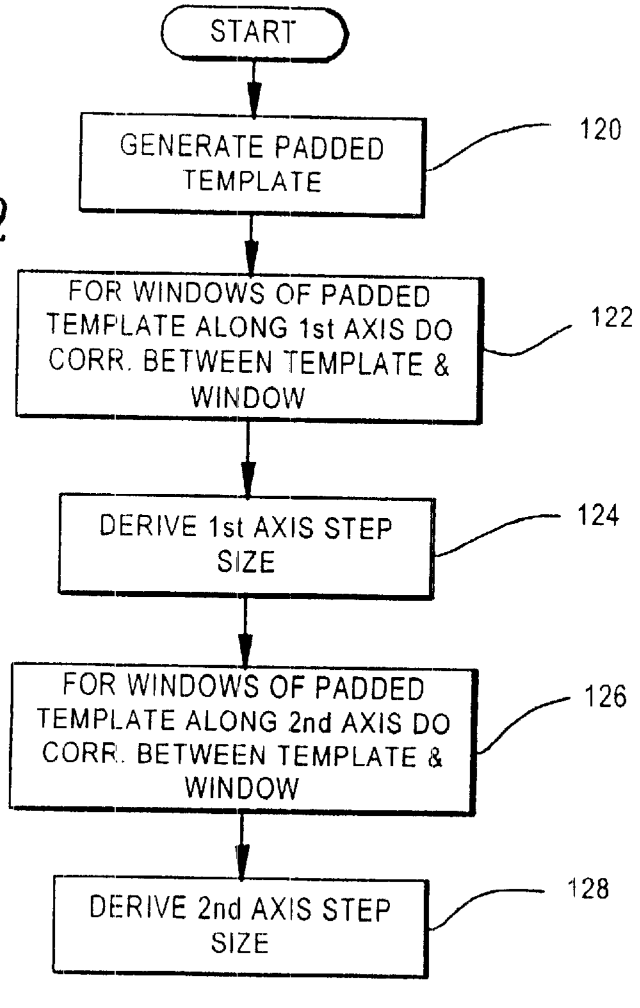
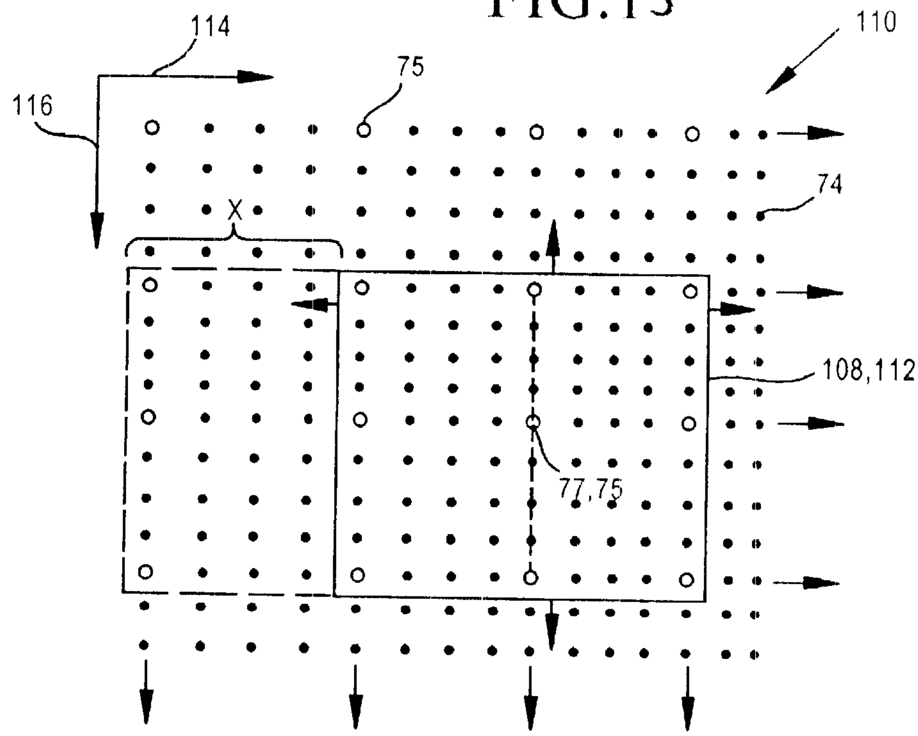

START
GENERATE PADDED TEMPLATE
120
FOR WINDOWS OF PADDED TEMPLATE ALONG 1st AXIS DO CORR. BETWEEN TEMPLATE & WINDOW
122
DERIVE 1st AXIS STEP SIZE
124
FOR WINDOWS OF PADDED TEMPLATE ALONG 2nd AXIS DO CORR. BETWEEN TEMPLATE & WINDOW
126
DERIVE 2nd AXIS STEP SIZE
128

FIG.12

FIG. 13


114
116
75
110
X
74
108,112
77,75

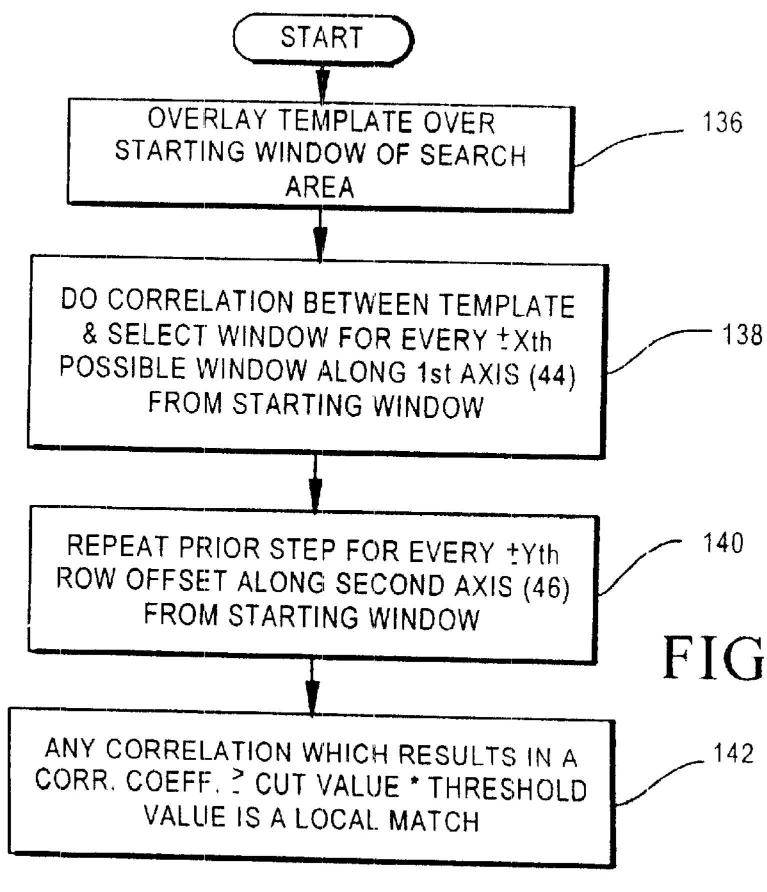
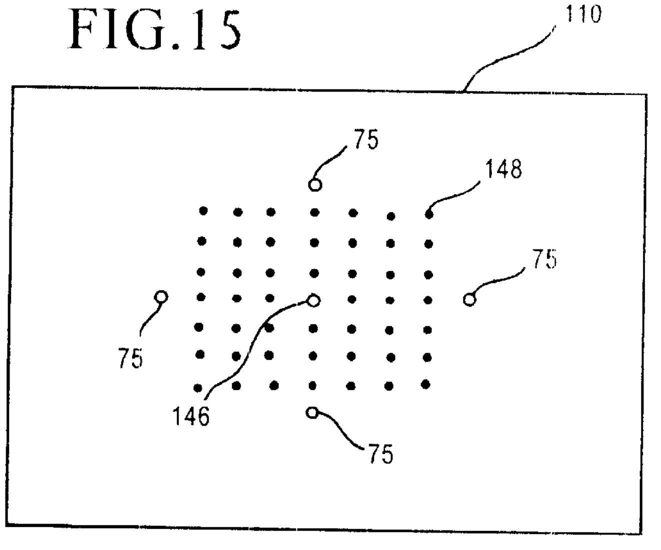

START
OVERLAY TEMPLATE OVER STARTING WINDOW OF SEARCH AREA
136
DO CORRELATION BETWEEN TEMPLATE & SELECT WINDOW FOR EVERY ±Xth POSSIBLE WINDOW ALONG 1st AXIS (44) FROM STARTING WINDOW
138
REPEAT PRIOR STEP FOR EVERY ±Yth ROW OFFSET ALONG SECOND AXIS (46) FROM STARTING WINDOW
140
ANY CORRELATION WHICH RESULTS IN A CORR. COEFF. ≥ CUT VALUE * THRESHOLD VALUE IS A LOCAL MATCH
142

FIG. 14

FIG.15


110
75
148
75
75
146
75

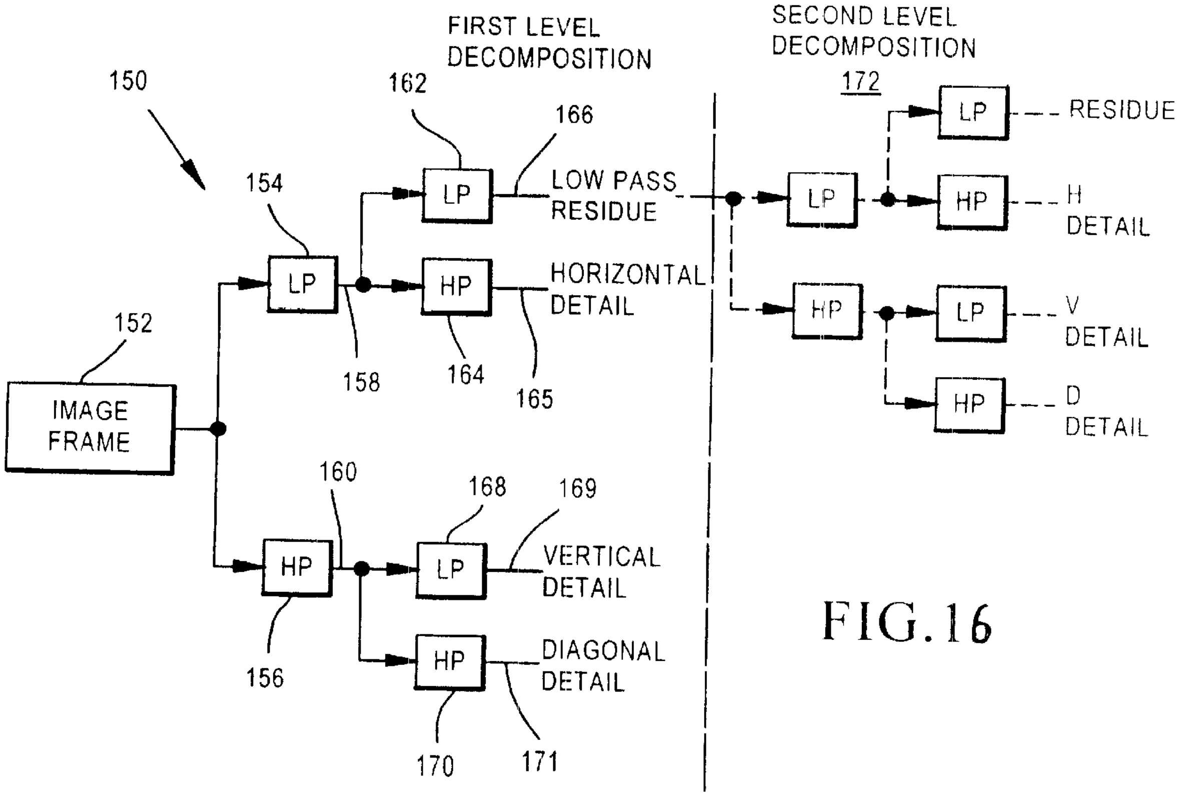

FIRST LEVEL
DECOMPOSITION
SECOND LEVEL
DECOMPOSITION
172
150
162
166
154
LP
LP
LOW PASS
RESIDUE
HP
HORIZONTAL
DETAIL
152
158
164
165
IMAGE
FRAME
160
168
169
HP
LP
VERTICAL
DETAIL
HP
DIAGONAL
DETAIL
156
170
171
LP
RESIDUE
LP
HP
H
DETAIL
HP
LP
V
DETAIL
HP
D
DETAIL

FIG.16

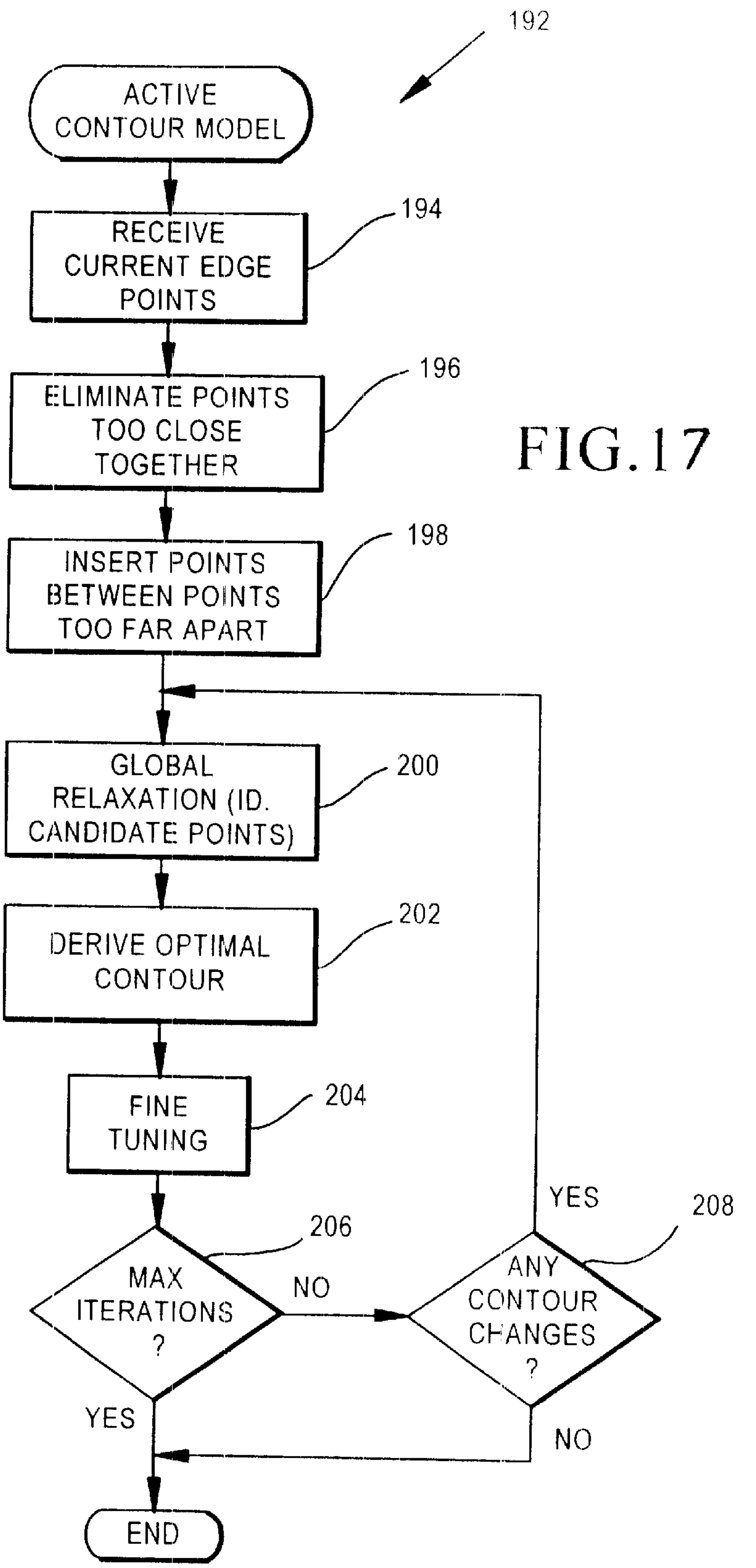

192
ACTIVE CONTOUR MODEL
RECEIVE CURRENT EDGE POINTS
194
ELIMINATE POINTS TOO CLOSE TOGETHER
196
INSERT POINTS BETWEEN POINTS TOO FAR APART
198
GLOBAL RELAXATION (ID. CANDIDATE POINTS)
200
DERIVE OPTIMAL CONTOUR
202
FINE TUNING
204
MAX ITERATIONS ?
206
NO
YES
ANY CONTOUR CHANGES ?
208
YES
NO
END

FIG. 17

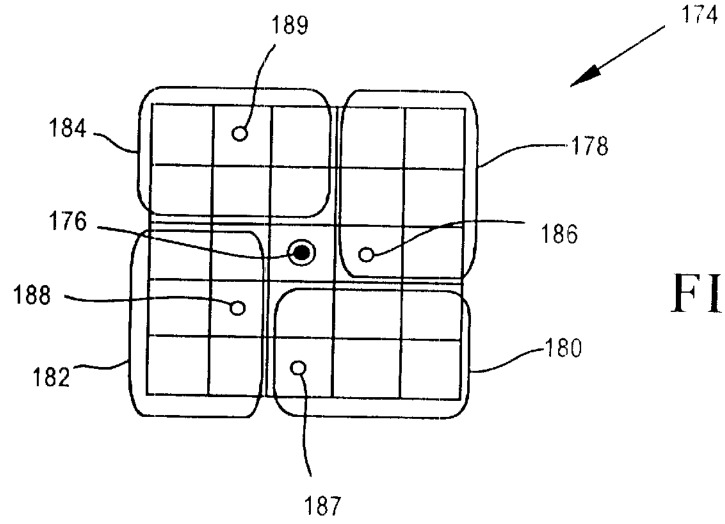

174
189
184
178
176
186
188
182
180
187

FIG. 18

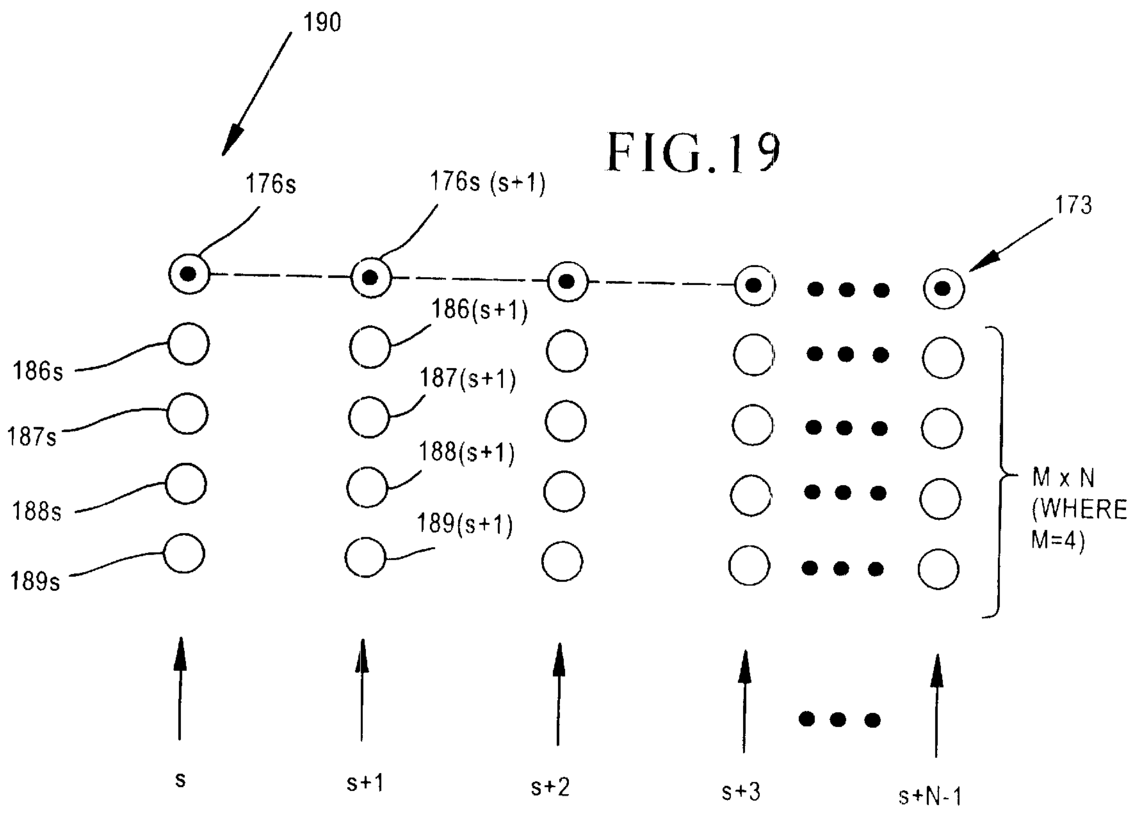

190
176s
176s (s+1)
173
186(s+1)
187(s+1)
188(s+1)
189(s+1)
186s
187s
188s
189s
M x N
(WHERE
M=4)
s
s+1
s+2
s+3
s+N-1

FIG. 19

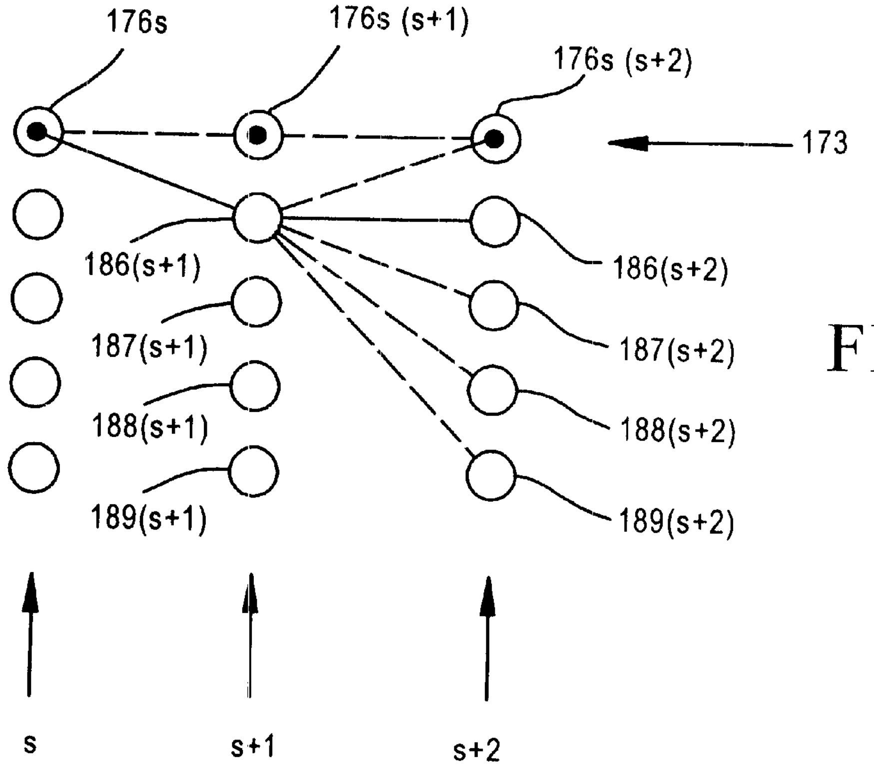

176s
176s (s+1)
176s (s+2)
173
186(s+1)
187(s+1)
188(s+1)
189(s+1)
186(s+2)
187(s+2)
188(s+2)
189(s+2)
s
s+1
s+2

FIG. 20

IMAGE PROCESSING IN HSI COLOR SPACE USING ADAPTIVE NOISE FILTERING

CROSS REFERENCE TO RELATED APPLICATIONS

This invention is a continuation of and related to U.S. patent application Ser. No. 09/216,692 filed Dec. 18, 1998 (now U.S. Pat. No. 6,301,387 issued Oct. 9, 2001) of Sun et al. for "Template Matching Using Correlative Auto-Predictive Search;" U.S. patent application Ser. No. 09/216,691 filed Dec. 18, 1998 (now U.S. Pat. No. 6,243,494 issued Jun. 5, 2001) of Sun et al. for "Template Matching in Three Dimensions Using Correlative Auto-Predictive Search;" U.S. patent application Ser. No. 09/233,894 filed Jan. 20, 1999 (now U.S. Pat. No. 6,272,250 issued Aug. 7, 2001) of Sun et al. for "Color Clustering for Scene Change Detection and Object Tracking in Video Sequences;" and U.S. patent application Ser. No. 09/323,501 filed Jun. 1, 1999 of Sun et al. for "Video Object Segmentation Using Active Contour Modelling With Global Relaxation." The content of such applications are incorporated herein by reference and made a part hereof.

BACKGROUND OF THE INVENTION

This invention relates to color image processing techniques such as object tracking and image segmentation, and more particularly to a process for filtering HSI data for object tracking and image segmentation.

Color image processing techniques often are used in image enhancement, video encoding, video editing and computer vision applications. Image tracking relates to the identification of an image object each frame in a sequence of image frames, such as in a sequence of motion video frames. Image segmentation is used to identify boundaries and edges of image objects in an image frame.

HSI refers to the Hue, Saturation, Intensity color model for presenting color data. There are many different color models (also referred to as color domains or color spaces) developed for the representation and manipulation of color data. Color monitors typically use a Red, Green, Blue (RGB) color model. Color printers typically use a Cyan, Yellow, Magenta (CYM) or a Cyan, Yellow, Magenta, Black (CYMK) color model. Color television broadcast signals typically use a luminance, intensity, color difference (YIQ) color model, where I and Q relate to chrominance.

The Hue Saturation Intensity (HSI) color model closely resembles the color sensing properties of human vision. The intensity component is related to the luminance component decoupled from the color. The hue and saturation components are related to the way in which a human perceives color. Such relation to human vision makes it desirable to use the HSI color model for color image processing techniques, such as image enhancement and image segmentation.

The input image data for color image processing techniques typically is in RGB format. Unfortunately the transformation from RGB to HSI color space and from HSI to RGB color space is very nonlinear and complicated in comparison to the conversion formulas among the other color models. As an example, when an RGB image is degraded by random noise, the nonlinearity in the conversion formulae causes the noise distribution in HSI color space to be nonuniform. Further, the noise distribution in HSI color space depends on the intensity and saturation values of the input data. For example, when the intensity

value is small, the noise in the saturation and hue is large. This creates problems in using the HSI color model for image processing techniques, such as image enhancement and image segmentation. Accordingly, there is a need for a method which reduces the magnitude of the noise or the nonuniformity of the noise variance in HSI color space.

With regard to object tracking, it is known to use data clustering methods, such as found in pattern learning and recognition systems based upon adaptive resonance theory (ART). Adaptive resonance theory, as coined by Grossberg, is a system for self-organizing stable pattern recognition codes in real-time data in response to arbitrary sequences of input patterns. (See "Adaptive Pattern Classification and Universal Recoding: II . . . ," by Stephen Grossberg, Biological Cybernetics 23, pp. 187–202 (1976)). It is based on the problem of discovering, learning and recognizing invariant properties of a data set, and is somewhat analogous to the human processes of perception and cognition. The invariant properties, called recognition codes, emerge in human perception through an individual's interaction with the environment. When these recognition codes emerge spontaneously, as in human perception, the process is said to be self-organizing.

With regard to image segmentation, active contour models, also known as snakes, have been used for adjusting image features, in particular image object boundaries. In concept, active contour models involve overlaying an elastic curve onto an image. The curve (i.e., snake) deforms itself from an initial shape to adjust to the image features. An energy minimizing function is used which adapts the curve to image features such as lines and edges. The function is guided by external constraint forces and image forces. The best fit is achieved by minimizing a total energy computation of the curve. The energy computation is derived from (i) energy terms for internal tension (stretching) and stiffness (bending), and (ii) potential terms derived from image features (edges; corners). A pressure force also has been used to allow closed contours to inflate. Conventionally, iterations are applied to get the entire contour to converge to an optimal path.

SUMMARY OF THE INVENTION

According to the invention, adaptive noise filtering is applied to an image frame of HSI data to reduce and more uniformly distribute noise while preserving image feature edges. In one implementation for a sequence of image frames, such filtering allows for improved image object tracking ability and improved image object segmentation.

According to one aspect of the invention, it has been found that in transforming an RGB image into HSI color space, noise present in the RGB image is nonuniformly distributed within the resulting HSI image. In particular the hue and saturation components have what may be considered to be a Cauchy distribution of noise where mean and variance do not exist. As a result, a noise distribution model has been determined experimentally.

According to another aspect of this invention, the HSI data is filtered using an adaptive spatial filter having a plurality of averaging kernels. An appropriate kernel is selected for each pixel for each of the hue and saturation components. A set of thresholds are defined for selecting the kernel for the hue component. Another set of thresholds are defined for selecting the kernel for the saturation component.

According to another aspect of this invention, the kernel for the saturation component is selected by comparing the intensity component to the saturation component thresholds.

According to another aspect of this invention, the kernel for the hue component is selected by comparing the product of intensity component and the saturation component to the hue component thresholds.

According to another aspect of this invention, a color gradient operation is applied to the filtered HSI data to aid in detecting image object boundaries.

According to another aspect of the invention, a method is provided for segmenting an image frame of pixel data, in which the image frame includes a plurality of pixels. For each pixel of the image frame, the corresponding pixel data is converted into hue, saturation, intensity color space. The HSI pixel data then is filtered with the adaptive spatial filters. Object segmentation then is performed to define a set of filtered HSI pixel data corresponding to the image object. The image frame then is encoded in which pixel data corresponding to the image object is encoded at a higher bit rate than other pixel data.

An advantage of the invention is that image segmentation techniques are performed in HSI color space where color sensing properties more closely resemble human vision. According to another advantage of this invention, object boundaries are preserved while noise level is significantly reduced and the noise variance is made more uniform.

These and other aspects and advantages of the invention will be better understood by reference to the following detailed description taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a system for performing adaptive noise filtering, video segmentation and object tracking according to an embodiment of this invention;

FIG. **2** is a flow chart of a method for processing a sequence of image frames to perform adaptive noise filtering, object tracking and image segmentation according to an embodiment of this invention;

FIGS. **3***a*–**3***c* are sample images of noise in hue, saturation and intensity components, respectively;

FIG. **4** is a chart of saturation component noise variance versus intensity;

FIG. **5** is a 3D graph of hue component noise variance versus intensity and saturation;

FIG. **6** is a diagram depicting multiple filtering kernels in the adaptive spatial filter according to an embodiment of this invention;

FIG. **7** is a chart showing sample thresholds for selecting a filtering kernel for filtering the saturation component;

FIGS. **8***a*–**8***c* are sample HSI images of an image without noise, an image with noise which has not been filtered and an image with noise which has been filtered, where in each case a color gradient operation has been applied;

FIG. **9** is a diagram of an input, processing, output sequence for the scene change detection subsystem of FIG. **1** to obtain image edges;

FIG. **10** is a flow chart for a method of pattern learning and recognition implemented by the scene change detection subsystem of FIG. **1**;

FIG. **11** is a diagram of a template and search area for performing a correlative autopredictive search (CAPS);

FIG. **12** is a flow chart of a process for determining CAPS step sizes according to an implementation of the object tracking subsystem of FIG. **1**;

FIG. **13** is a diagram of a search area of data points with a window area to be tested against a template;

FIG. **14** is a flow chart of a process for performing a fast search of the search area to identify local matches between a template and a subset of window areas of the search area;

FIG. **15** is a diagram of center data points for windows in the vicinity of the local template match to be tested for a better match (also shown are center points for nearby windows tested during the fast search);

FIG. **16** is a diagram of a quadrature modelling filter for decomposing an image to achieve detailing images and a low pass residue;

FIG. **17** is a flow chart of an active contour modelling process for segmenting an image;

FIG. **18** is a diagram of a 5×5 pixel domain about a current edge point (pixel) used for selecting other candidate points which might be used in place of the current edge point;

FIG. **19** is a diagram of potential edge points processed to preserve one optimal path for an image object boundary; and

FIG. **20** is a partial travel path of the contour in the process of being derived from the set of points of FIG. **19**.

DESCRIPTION OF SPECIFIC EMBODIMENTS

Overview

FIG. **1** shows a system **10** for adaptive noise filtering and image object segmentation and tracking according to one embodiment of the invention. System **10** includes a user interface **11**, an adaptive noise filtering subsystem **13**, a subsystem **12** for detecting changes in scene (e.g., a modified adaptive resonance theory —2 (M-ART2) subsystem), an object tracking subsystem **14** (e.g., a 2D or 3D correlative auto-predictive search (CAPS) subsystem), an object segmentation subsystem **18** (e.g., an edge energy derivation subsystem and an active contour modelling subsystem), and an encoder subsystem **19**.

The adaptive noise filtering subsystem **13** converts input image frame data from RGB or another input format into HSI format, then filters the HSI data and applies a colored gradient to the filtered data. In other embodiments the adaptive noise filtering subsystem **13** need not be combined with the other subsystems for scene change detection, object tracking, object segmentation, energy derivation or encoding, but may stand alone with the user interface **11**, or be combined with one or more of the same or other subsystems to form an alternative system for image processing.

The M-ART2 subsystem **12** serves to detect scene changes in a sequence of image frames. The CAPS subsystem **14** serves to identify an object in a given image frame. The CAPS subsystem also serves to track the object among a sequence of input image frames. A motion vector of the tracked object is maintained. The edge energy subsystem serves to calculate the edge energy for an image object to be modelled. The active contour modelling subsystem serves to segment an image object and accurately model an edge boundary of the image object being tracked. When an operator completes enhancements, editing or filtering of a video sequence, the encoder subsystem **19** encodes/compresses the finalized video sequence into a desired format.

The various subsystems are implemented in software on one or more host computing devices or are integrated into an embedded system. Preferably the functions of the various subsystems are performed by programmed digital computers of the type which are well known in the art. A host computer system for embodiments of the invention typically includes a display monitor, a keyboard, a pointing/clicking device, one or more processors or multiprocessors, random access memory (RAM), a non-volatile storage device such as a hard disk drive, and other devices such as a communication or network interface (e.g., modem; ethernet adapter), a transportable storage media drive, such as a floppy disk drive, CD-ROM drive, zip drive, bernoulli drive or other magnetic, optical or other storage media. The various components interface and exchange data and commands through one or more busses. The computer system receives information by entry through the keyboard, pointing/clicking device, a network interface or another input device or input port. The computer system may be any of the types well known in the art, such as a mainframe computer, minicomputer, or microcomputer. To speed up computations, (e.g., convolutions, correlations) parallel processing may be implemented.

FIG. **2** shows a system flow chart of a method **20** for (i) applying an adaptive noise filtering process HSI data and (ii) tracking and segmenting an image object defined by such data according to an embodiment of this invention. Although tracking and segmentation are described below as being performed on the filtered data, the filtering process may be applied, instead, for an alternative image processing system in which alternative image processing techniques are implemented.

Input to the method at steps **22** and **24** are initial edge points and an initial image frame. In one application the initial edge points are selected manually by an operator using a conventional video editing application interface. In another application the edge points are derived automatically and fed into a method embodiment of this invention.

At steps **26–30** the adaptive noise filtering subsystem **13** performs the steps of converting the image data into HSI format (step **26**), applying adaptive spatial filtering to the HSI data (step **28**) and applying a color gradient to the filtered HSI data (step **30**). The resulting HSI data then is analyzed at step **32** using the scene change detection subsystem **12**. In one embodiment, a modified applied resonance theory —2 (M-ART2) process is executed as part of step **32** to define clusters of image pixels. The M-ART2 process is described below in a separate section. At step **34**, the object segmentation subsystem **18** derives the edge energy of the input edge boundary is derived. Then at step **36** the subsystem **18** applies an active contour model to segment the edge boundary and accurately model the object boundary. The active contour model is described below in a separate section. At step **38** the modelled image object boundary is output. In some embodiments the output is written to a buffer, a file, and/or to a display. In various embodiments the RGB to HSI conversion step **26**, the adaptive spatial filtering step **28** and the color gradient step **30** may occur at any step prior to the image segmentation steps (i.e., steps **34** and **36**).

Iterative processing then is performed for subsequent image frames. In some embodiments each image frame is processed. In other embodiments, image frames are periodically or aperiodically sampled. At step **39** the next image frame to be processed is input to the method implementation **20**. At steps **40–42** the adaptive noise filtering subsystem **13** performs the steps of converting the image data into HSI format (step **40**), applying adaptive spatial filtering to the HSI data (step **41**) and applying a color gradient to the filtered HSI data (step **42**). The resulting HSI data then is analyzed at step **44** using the scene change detection subsystem **12** to determine whether there has been a change in scene. If a scene change is detected at step **44**, then the method **20** is complete, or is re-initialized to track another image object. If a scene change has not occurred, then the image object is identified from the image frame using a correlative auto-predictive search (CAPS) process. The CAPS process is described below in a separate section. If at step **48** the image object is not found using the CAPS process, then the tracking method **20** terminates or re-initializes for tracking another object. If the object is identified, then the edge energy for the object boundary is derived at step **50**. Then at step **52** an active contour model is applied to segment the image boundary and accurately model the object boundary. At the next step, step **38** the modelled image boundary is output. As described above for the initial image frame, in some embodiments the output is written to a buffer, a file, and/or to a video screen. The process then repeats steps **38–52** for another image frame. As a result, an image object is segmented and tracked over many image frames. Thereafter, in some embodiments an encoding process is applied to encode the data into a desired format (e.g., MPEG-4 video).

Adaptive Noise Filtering in HSI Color Space

One of the functions of the Filtering Subsystem **13** is to convert the input image data into HSI format. Typically, the input image data is in RGB format. In one embodiment the following equations are implemented to convert from RGB format to HSI format:

$$H = \cos^{-1}\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{0.5}} \tag{I}$$

$$S = 1 - \frac{3}{R+G+B}[\min(R,G,B)] \tag{II}$$

$$I=\tfrac{1}{3}(R+G+B) \tag{III}$$

where R, G and B are the respective RGB components of the input data;

min (R,G,B) denotes a function for the minimum of R, G and B;

the ranges S, I, R, G and B are in [0,1], while H is in degrees (0 to 360°);

Hue=H, where B<G,

Hue=360−H where B>G

Nonlinearity of Noise in HSI Conversion:

For an input image with data in RGB format, noise occurs in the RGB color space. It is assumed that random gaussian noise with zero mean and σ^2 variance occurs in the RGB image data. In addition, the noise in each RGB color component is assumed to be independent from one another and also from the image data signal. As shown in Eqs. (I)–(III), the RGB-to-HSI conversion equations are nonlinear. For example, the noise variance of intensity (I) is $\sigma^2/3$. However, the noise variances in hue and saturation cannot be defined analytically since they have a kind of Cauchy distribution, where mean and variance do not exist. Therefore, the noise characteristics of hue and saturation have been evaluated experimentally.

In order to measure the noise variance of hue and saturation and to analyze the noise dependency on the image data, several sample images are created in the HSI color space. In one embodiment a 256×256-pixel sample image is divided into 16×16 blocks with each block having 16×16 pixels. Each block in one sample image has constant HSI values as defined below:

$H(i,j)=64$ for $1 \leqq i \leqq 16$, $1 \leqq j \leqq 16$

$S(i,j)=9+7j$ for $1 \leqq i \leqq 16$, $1 \leqq j \leqq 16$

$I(i,j)=9+7i$ for $1 \leqq i \leqq 16$, $1 \leqq j \leqq 16$

where i and j are block numbers in horizontal and vertical directions, respectively. The sample image has an intensity value increasing horizontally while the saturation value increases vertically. The experiment is repeated with several different hue values.

In each experiment the sample image in the HSI color space is converted to the RGB color space, and random Gaussian noise is added to each RGB color component. The noise has a Gaussian distribution with zero mean and σ^2 variance. The image with noise in the RGB color space is reconverted to the HSI color space and the noise characteristics are analyzed. Noise in the HSI color space is computed as follows:

$$\begin{bmatrix} n_h \\ n_s \\ n_i \end{bmatrix} = RGB \text{ to } HSI \begin{bmatrix} R+n_r \\ G+n_g \\ B+n_b \end{bmatrix} - RGB \text{ to } HSI \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{(IV)}$$

where, RGB to HSI[ ] corresponds to the conversion Eqs. I–III from RGB to HSI;

(n_r, n_g, n_b) are the noises in RGB color components, respectively; and

(n_h, n_s, n_i) are the noises in HSI color components, respectively.

FIGS. **3***a*–**3***c* show the noise distribution of the Hue, Saturation and Intensity components respectively for H=64. In this example, the noise that is added to the RGB image, (n_r, n_g, n_b), has a variance of 9. As shown in FIG. **3***b*, the noise in the saturation component (n_s) depends on the intensity value, (i.e., it is large when the intensity is small at the left side of FIG. **3***b*). The noise in the hue component (n_h) depends on the intensity and saturation values, (i.e., it is large when the intensity and saturation values are small in the upper-left corner of FIG. **3***a*.

To show the relationship between noise and the image data, the variance of noises in saturation and hue is analyzed with respect to the intensity and saturation values. In FIG. **4**, the variance of n_s is plotted with respect to the intensity value, which is approximately proportional to $1/\text{Intensity}^2$. The variance of n_s also depends on the hue and saturation values, but their effects are negligible in comparison with that from the intensity value. FIG. **4** plots the mean value of the variance of n_s with different hue and saturation values. FIG. **5** shows the variance of n_h with respect to the intensity and saturation values. The variance of n_h also depends on the hue value itself, but this dependency is negligible compared with that on the intensity and saturation values. Accordingly, in applicant's model noise in the saturation component is taken to be proportional to the value of the intensity component. Noise in the hue component is taken to be proportional to the value in the intensity and saturation components.

Adaptive Spatial Filtering:

At steps **28** and **41** (see FIG. **2**) an adaptive spatial filtering method is executed to reduce the noise in the image data signal. According to the method kernel size of an averaging filter is adapted to make the noise distribution in the HSI color space more uniform while preserving image edge information. The kernel size is adapted based on noise variance.

Referring to FIG. **6**, a kernel is selected from a set **60** of kernels **K1** to **K4** for each pixel according to the intensity and saturation values. In one embodiment saturation component threshold values (A_s, B_s, C_s, D_s) for filtering the saturation component are defined based on the noise analysis results in FIG. **4**. For example, the filter kernel **K1** is applied when the variance on n_s is between $3\sigma^2$ and $7\sigma^2$. Then the noise variance after filtering with the K1 kernel is between $3\sigma^2/5$ and $7\sigma^2/5$.

Similarly, the **K2**, **K3**, and **K4** kernels are used when the variance ranges of n_s are $[7\sigma^2, 18\sigma^2]$, $[18\sigma^2, 35\sigma^2]$, and $[35\sigma^2, \ ]$, respectively. The threshold values of A_s, B_s, C_s, and D_s are selected from the intensity axis (of FIG. **4**) to correspond (in the n_s variance axis) to $3\sigma^2$, $7\sigma^2$, $18\sigma^2$, $35\sigma^2$, respectively as shown in FIG. **7**.

Hue component threshold values (A_h, B_h, C_b, D_h) for filtering the hue component are defined based on the noise analysis results in FIG. **5**. The hue component threshold values (A_h, B_h, C_h, and D_h) are selected from FIG. **5** by using $3\sigma^2$, $7\sigma^2$, $18\sigma^2$, and $35\sigma^2$ as transition points in the n_h variance axis. In alternative embodiments the number of filter kernels and/or their shapes and coefficient values may be varied or increased, in which case the new threshold values are determined to make the noise distribution more uniform. When the number of filter kernels increases, the noise distribution is made more uniform, and the noise variance is further reduced for extremely small intensity and/or saturation values.

Once the saturation component threshold values (A_s, B_s, C_s, and D_s) are established, the saturation component of the HSI image is filtered adaptively by the filter kernel selected for each pixel based on its intensity value according to equation (V) below:

$$\text{filter kernel for } S(x, y) = \begin{cases} \text{no filter,} & \text{for } A_s < I(x, y) \\ K1, & \text{for } B_s < I(x, y) \le A_s \\ K2, & \text{for } C_s < I(x, y) \le B_s \\ K3, & \text{for } D_s < I(x, y) \le C_s \\ K4, & \text{for } I(x, y) \le D_s \end{cases} \quad \text{(V)}$$

where (x,y) are the horizontal and vertical coordinates of a respective image pixel. After the saturation component is filtered, the hue component can be filtered in a similar way using equation (VI) below. However, the filter kernel for each hue pixel is adaptively selected based on the product of intensity and saturation values as follows:

$$\text{filter kernel for } H(x, y) = \begin{cases} \text{no filter,} & \text{for } A_h < I(x, y)S(x, y) \\ K1, & \text{for } B_h < I(x, y)S(x, y) \le A_h \\ K2, & \text{for } C_h < I(x, y)S(x, y) \le B_h \\ K3, & \text{for } D_h < I(x, y)S(x, y) \le C_h \\ K4, & \text{for } I(x, y)S(x, y) \le D_h \end{cases} \quad \text{(VI)}$$

where S(x,y) is the saturation component after filtering using Eq. (V). The adaptive spatial filtering improves the saturation and hue noise characteristics significantly by reducing noise level and by making the noise distribution more uniform. The smoothing filters reduce the random noise and smooth the image details. To avoid blurring the image details, in one embodiment an image edge-preserving procedure (equation VII) is applied during adaptive filtering as follows:

$$\text{filter coefficient at } (u, v) = \begin{cases} 0, & \text{if } |I(u, v) - I(x, y)| > 2\sigma \\ 1, & \text{if } |I(u, v) - I(x, y)| \le 2\sigma \end{cases} \quad \text{(VII)}$$

where (x,y) is the center pixel of the kernel, (i.e., the pixel to be filtered), and (u,v) are other pixels in the filter kernel. In equation (VII), σ is the standard deviation of noise in the RGB color space. If the threshold value in equation (VII) is too large, the image edges end up being smoothed by the adaptive spatial filtering. It has been found that the threshold value of 2σ was effective to handle about 90% of noise in the intensity component because the variance of n_i is $\sigma^2/3$. In various applications, the noise variance, σ^2, is measured or estimated in an RGB image.

Applying Color Gradient:

A color gradient image of the filtered input frame is derived at steps **30** and **42**. The color gradient image is obtained by applying a derivative of Gaussian (DOG) operator to each HSI pixel component in the filtered image. Equation (VIII) below characterizes the application of the color gradient to the filtered image resulting from step **28** or step **41**:

$$c(x, y) = \sqrt{\frac{\nabla H(x, y)^2 + \nabla S(x, y)^2 + \nabla I(x, y)^2}{3}} \qquad \text{(VIII)}$$

where

$$\nabla H(x,y)^2=(H(x,y)*G_h(x,y))^2+(H(x,y)*G_v(x,y))^2;$$

$$\nabla S(x,y)^2=(S(x,y)*G_h(x,y))^2+(S(x,y)*G_v(x,y))^2; \text{ and}$$

$$\nabla I(x,y)^2=(I(x,y)*G_h(x,y))^2+(I(x,y)*G_v(x,y))^2.$$

In equation (VIII), $G_h(x,y)$ and $G_v(x,y)$ are the gradient operators in the horizontal and the vertical directions, respectively. The symbol * denotes a convolution operation.

FIG. **8***a* shows a color gradient output image for a sample HSI image in which there is no noise present. FIG. **8***b* shows a color gradient output image for the same sample HSI image, but in which there is noise present. The adaptive filtering is not performed for the image of FIG. **8***b*. FIG. **8***c* shows a color gradient output image for the same HSI image with the same noise present as in FIG. **8***b*, but where the adaptive filtering steps **28** or **41** are performed prior to applying the color gradient. As evidenced in FIGS. **8***b* and **8***c*, the noise is definitely reduced in the color gradients with the adaptive spatial filtering.

Pixel Clustering and Scene Change Detection

In one embodiment the scene change detection subsystem **12** is based upon a method of modified applied resonance theory as described in the commonly-assigned U.S. patent application Ser. No. 09/233,894, filed Jan. 20, 1999 for “Color Clustering for Scene Change Detection and Object Tracking in Video Sequences.” The content of such application is incorporated herein by reference and made a part hereof.

The subsystem **12** performs pattern learning and recognition on a sequence of input image frames. Referring to FIG. **9**, the subsystem **12** processes a current image frame **60** grouping the image frame contents into clusters **66**. The image frame **60** is formed by an array of image pixels P. For a raster type image frame, the image pixels are arranged into y rows and x columns. In various embodiments the image pixels are color image pixels coded according to a standard red, green, blue coding scheme (e.g., NTSC), a standard yellow, magenta, cyan and black coding scheme (YMCK), a standard luminosity, chrominance, brightness coding scheme (e.g., YUV), the hue saturation intensity color scheme (HSI), or some other color coding scheme. For the embodiment for the process of FIG. **2** the conversion of RGB data to HSI data occurs prior to the M-ART2 26 steps. Accordingly, HSI data is used for such embodiment. In various embodiments the RGB to HSI conversion may occur at any step prior to the image segmentation steps (i.e., steps **28** and **30** to generate edge energy **28** and apply active contour model **30**).

Each image frame is a set of data points. Each pixel is a data point. A data point is referred to herein as an input vector. Input vector P_{ij} corresponds to pixel P (x_i,y_j) which for an HSI coding scheme has a value (H,S,I). The subsystem **12** processes a sequence **68** of input vectors P corresponding to a given set of data points (i.e., a current image frame **60**). The input vectors P are grouped into clusters **66**.

Each cluster **66** is a learned or a recognized pattern. For a first set of input data (i.e., an initial image frame) there is no prior information for allocating the data points into clusters. Thus, the patterns are learned. For subsequent sets of data points (e.g., subsequent images in a sequence of image frames), the patterns previously learned may be used. Specifically, data points for a current set of data points (image frame) are tested to try and recognize the prior patterns in the new set of data points. The process for analyzing the subsequent sets of data points is a recognition process. During the recognition process, the previous learned patterns also are updated and modified based upon the new data.

Pattern Learning and Recognition:

Referring to FIG. **10**, a flow chart of the pattern learning and recognizing process (also see steps **32** and **44** of FIG. **2**) commences at step **76**. If the current image frame is an initial image frame, then at step **78** various parameters are reset. Further, if the current image frame is an initial image frame then there are no clusters that have been started.

The current image frame **60** is processed in an iterative manner (step **80**). At step **82**, an initial set of prototype vectors for this processing iteration of the current image frame is obtained. There is a prototype vector for each cluster defined. If the current image frame is an initial image frame, then there are no prototype vectors. The prototype vector is a weighted centroid value based upon a history of input vectors allocated to the corresponding cluster.

The process for allocating input vectors into clusters is performed for each input vector (step **84**). Such process is based upon a minimum distance measure. In various embodiments an euclidean distance, an absolute distance or some other distance measure is used. In one embodiment the euclidean distance is used. An input vector is allocated to a cluster to which it has a minimal euclidean distance with the cluster's prototype vector. At step **86**, the prototype vector closest to the input vector is found. As a self-organizing control for allocating data into clusters, a vigilance parameter, also referred to herein as a vigilance value, is used. A vigilance test is performed at step **88**. If the minimum euclidean distance is not less than the vigilance value, then a new cluster is defined at step **90**. The input vector is assigned to such new cluster and becomes the initial prototype vector for such new cluster. If the minimum euclidean distance is less than the vigilance value, then the input vector is assigned to the cluster corresponding to the closest prototype vector at step **92**. Thus, an input vector is allocated to a preexisting cluster or a new cluster.

For a new learning and recognition process, there are no prototype vectors to start with. Thus, the first input vector will define an initial prototype vector for a first cluster. The minimum distance between the next input vector and the prototype vectors will be to the first prototype vector (since at this point in the example there is only one prototype vector). If such minimum distance exceeds the vigilance value, then the second input vector becomes an initial prototype vector for a second cluster. If, however, such minimum distance is within the vigilance value distance, then the second input vector is allocated to the first cluster.

If the second input vector is allocated to the first cluster, then the prototype vector for such first cluster is modified at step **94**. The modified prototype vector for the first cluster becomes the weighted centroid value for all data points among the first cluster, based upon the following equation:

$$w_k^{(new)} = \frac{P(x, y) + w_k^{(old)} \|cluster_k^{(old)}\|}{\|cluster_k^{(old)}\| + 1}$$

where, $W_k^{(new)}$=new prototype vector for cluster k=new centroid value;

$W_k^{(old)}$=old prototype vector for cluster k=old centroid value;

P(x,y)=input vector;

$\| cluster_k^{(old)}\|$=number of vectors in cluster k.

The influence of the new input vector in the cluster has a weighted influence on the prototype vector of the cluster. The weight is proportional to the number of input vectors in the cluster, and thus, corresponds to a statistical centroid. This process for updating the prototype vector provides a self-scaling feature to the cluster learning and recognition process.

This process is used for allocating each input vector of the current image frame. Once all the input vectors have been allocated in a given iteration, testing is performed to determine whether another iteration is needed and whether outlier clusters are present.

For an initial data set where no information is previously stored, one or more initial clusters are defined as above. An iterative process is used, however, to achieve a self-stabilizing quality to the clusters. Specifically, once the entire data set has been processed, allocating the input vectors into clusters, another iteration of allocating the input vectors into clusters is performed. Prior to performing another iteration, however, the clusters are analyzed for quantity in an outlier test (see step **96**). According to such test, any cluster having less than a prescribed threshold number of input vector members is discarded. More specifically the prototype vector is discarded and thus not used in finding a minimum distance to input vectors during a subsequent iteration. The input vectors in the discarded cluster are considered to be outliers (e.g., noise).

Consider, for example, a data set including 30,000 data values. Also, consider that after the first iteration, a first cluster has 20,000 members, a second cluster has 8,000 members, a third cluster has 1985 members, and a fourth cluster has 15 members. In this example, assume the prescribed threshold value is 64. Because cluster **4** has less than 64 input vector members, it is discarded. It is expected that many of the input vectors in this fourth cluster will be allocated into another cluster during a subsequent reiteration. Note that this is an example, and that the threshold value may be prescribed as a matter of design, or based upon empirical analysis.

For the next iteration the prototype vectors from the remaining clusters of the prior iteration are retained (step **82** of next iteration). In our example above, the prototype vectors from the first three clusters are retained, while the prototype vector from the fourth cluster is discarded. Each input vector then is re-allocated to a cluster during this subsequent iteration by determining the prototype vector to which it has a minimum euclidean distance. If such minimum distance is less than the vigilance value, then the input vector is allocated to the cluster corresponding to that prototype vector. If such minimum distance exceeds the vigilance value, then the input vector defines a prototype vector for a new cluster. According to various embodiments, either the same or a different vigilance value is used during the subsequent iterations.

Upon identifying a cluster into which an input vector is allocated during a subsequent iteration, the prototype vector (i.e., weighted centroid) for such cluster is recalculated. During the subsequent iteration the number of input vectors in the cluster is not reset, but remains at its last count from the prior iteration. Thus, the weighting influence of the current input vector is less during the subsequent iteration than during the prior iteration.

After the subsequent iteration is complete, like in the prior iteration, any cluster having fewer than a prescribed threshold number of input vector members is discarded (step **96**). The clusters then are tested for convergence (step **98**) to see if the number of input vector members in each cluster has significantly changed. If the number has not changed significantly, then the iterative process is complete. In this sense, the process is self-stabilizing. If a cluster was discarded for such iteration, such discarded cluster is considered to be an outlier and the members are considered as noise.

The number of cluster members is considered to change significantly if it has changed by more than a prescribed number of data points or prescribed percentage, whichever is larger. Such number and percentage are defined empirically. If the number of members has changed significantly then a new iteration is performed (step **80**). In the new iteration, the remaining (e.g., non-discarded) prototype vectors from the immediately prior iteration are used as the initial prototype vectors for each remaining cluster (step **82**). The iterations continue until, either the number of members in each cluster is not changed significantly (convergence test at step **98**), or a prescribed maximum number of iterations has occurred. Such maximum number of iterations is determined as a matter of design or empirically.

For a current image frame which is subsequent to an initial image frame, the prototype vectors correspond to the final prototype vectors from the preceding image frame processed among the sequence of image frames being processed. Each input vector in such current image frame is allocated to a cluster by determining the prototype vector to which it has a minimum euclidean distance (step **86**). If such minimum distance is less than the vigilance value (step **88**), then the input vector is allocated to the cluster corresponding to that prototype vector (step **92**). If such minimum distance exceeds the vigilance value, then the input vector defines a prototype vector for a new cluster (step **90**). A new cluster corresponds to a new prototype pattern. According to various embodiments, either the same or a different vigilance value is used for the subsequent image frames in the sequence relative to that used for an initial image frame. In a preferred embodiment, the vigilance value is increased for the subsequent data sets, relative to that for the initial data set.

Upon identifying a cluster into which an input vector is allocated, the prototype vector (i.e., centroid) for such cluster is recalculated. The number of input vectors in the cluster is held over from the processing of the prior image frame. Thus, the prototype vector is a weighted centroid based upon multiple iterations of multiple image frames in a sequence of image frames.

After all the input vectors of the current data set have been allocated into clusters, another iteration of allocating the input vectors into clusters is performed. Prior to performing another iteration, however, the clusters are analyzed for quantity in the outlier test (step **96**). Any cluster having less than a prescribed threshold number of input vector members is discarded as described above for the initial data set. For the subsequent iteration the prototype vectors from the remaining clusters of the first iteration are retained. Each input vector then is re-allocated to a cluster during the subsequent iterations in the same manner as described above.

Each image frame in the sequence is similarly processed. In a preferred embodiment, the starting prototype vectors for allocating input vectors of a current data set are the final prototype vectors obtained during processing of the immediately prior data set. Further the count of the number of input vectors in a clusters is held over from prior iterations and prior image frames. New clusters defined as the sequence of data clusters continue correspond to new prototype patterns. New prototype patterns may occur in an image sequence, for example, due to an image object insertion, deletion or change.

Detecting Scene Changes Within a Sequence of Image Frames:

In the course of processing a sequence of image frames of a common scene, it is expected that much of the image content is similar from image frame to image frame. As a result, the defined clusters will be similar from image frame to image frame. The hold over of the count of input vectors in a cluster used in weighting the centroid of the cluster is based upon such assumption. If while processing a given image frame however, it is determined that the prototype vectors for each one of several clusters have changed beyond a threshold amount, then it is considered that the scene being imaged has changed. Specifically, upon processing any given image frame, if more than a prescribed number of prototype vectors has changed by more than a predetermined amount, then a scene change is considered to have occurred.

A scene change is determined by tracking a cluster change ratio from image frame to image frame. Specifically, after the iterative processing of input vectors for a current image frame is complete, the cluster rate of change for that image frame is derived. Cluster rate of change is derived in a preferred embodiment using the following equation:

$$R^f = \frac{\sum_{k=1}^{n_c^f} |N_k^f - N_k^{f-1}|}{N_{total}}$$

where, R^f=cluster change ratio for image frame f;

N_k^f=number of input vectors in cluster k of frame f (actual number, not the count used in prototype vector centroid which counts input vector for each iteration);

N_{total}=total number of input vectors in image frame f; and

n_c^f=number of clusters in frame f.

Note that if the k-th cluster in frame f is a new cluster, then N_k^{f-1} is simply zero. A scene change is identified at step **44** (see FIG. **9**) when the cluster change ratio for an image frame f exceeds a prescribed value, (e.g., 5%–10%). The prescribed value is determined empirically or be design and may exceed the example values of 5%–10%.

If a scene change is detected for a current image frame f, then the method **20** terminates, or is restarted (at step **22**) with the current image frame f set to be an initial frame. Image frame f then is re-processed as the current frame. Since it is an initial frame, parameters are reset at step **78**. Specifically, the prototype vectors are discarded. Thus at step **82** there are no prototype vectors. As a result, during processing of the first input vector, such input vector will define a new cluster and become the prototype vector for such cluster (step **90**). Additional cluster then are defined based upon whether the current input vector is farther than the vigilance value distance away from the prototype vector (s). Note that initially there are no prior input vectors in each new cluster (cluster count=0 when first deriving the weighted centroid of a new cluster).

Correlative Auto-Predictive Search (CAPS)—Object Tracking

A preferred embodiment of the correlative auto-predictive search process is described in the commonly-assigned U.S. patent application Ser. No. 09/216,692, filed Dec. 18, 1998 (now U.S. Pat. No. 6,301,387 issued on Oct. 9, 2001) for "Template Matching Using Correlative Auto-Predictive Search." The content of such application is incorporated herein by reference and made a part hereof.

The CAPS process is executed for image frames following an initial image frame. The object to be tracked has been defined during processing of the initial image frame. The object location is updated (by the CAPS process) during processing of subsequent image frames. The initial object or the updated object from the prior frame serves as a template for locating the object in the current image frame. Referring to FIG. **11**, the object being tracked serves as a template **108** while the current image frame serves as a search area **110**. The template **108** is overlaid onto a window **112** within the search area **110**. A motion vector is maintained which identifies the change in location of the object from one frame to the next. In some embodiments the motion vector derived from the previous frame is used to select a starting window **112**.

The template **108** data points are compared to the window's **112** data points to determine if the data points correlate to a desired degree. If they do, then a match for the template has been found. In a search area **110** formed by 'm' rows of 'n' data points, a template formed by 'k' rows of 'p' data points may be placed over (m−k+1)*(n−p+1) potential windows **112**.

To reduce the number of windows **112** that the template **108** is compared with, an effective step size is derived from the template. According to a 2-dimensional implementation embodiment, a step size along a first axis **114** is derived and a step size along a second axis **116** is derived. Rather then compare the template to every possible window of the search area **110**, the template **108** is moved along either or both of the first axis **114** and second axis **116** by the corresponding first axis step size or second axis step size.

Once the desired step sizes are derived, then the template **108** is compared to the various windows **112** of the search area **110** at the step size increments during a fast search process. In one embodiment the comparison is a correlation function of the template **108** and the window **112** and results in a correlation coefficient. Any window **112** in which the correlation coefficient with the template **108** is found to exceed a specific value is a local match for the template. In a preferred embodiment the specific value is the cut value times a threshold value.

Next, a full search then is performed in the vicinity of any location which is a local match. A full search of such vicinity encompasses performing a correlation between the template and every potential search area window between the local match location window and the windows at the prior and next step in each of the horizontal and vertical axes. For example, if the horizontal step size is 3 pixels and the vertical step size is 4 pixels, then correlations are performed for windows ±1 pixel and ±2 pixels along the horizontal axis and ±1 pixel, ±2 pixels and ±3 pixels along the vertical axis. In addition correlations are performed for windows off the axes within the area delineated by the step sizes. Thus, the full search of the vicinity of the local match for this example includes (2*2+1)*(2*3+1)−1=34 correlations between the template and the search area. Any locations among the local match locations and the locations tested during the full search of the vicinity which exceed the threshold value are considered template matches. In some embodiments, only the location having the highest correlation is considered a match. In other embodiments there may be multiple matches. Thus, the top matches or all matches above the threshold are selected as resultant matches.

Determining Step Size:

To determine effective step sizes, the template **108** itself is analyzed. Referring to FIG. **12**, at a first step **120** the template **108** is padded with additional data points to achieve a padded template. For circular padding, multiple copies of the template **108** are used to increase the template size. The number of copies may vary for differing embodiments. In a preferred embodiment there are at least 9 full copies of the template in the circularly padded template. In another embodiment, a padded template is achieved by linear padding. For linear padding, data points are added in which each data point has a common value. The common value is a padding constant. In one embodiment the padding constant may be 0 or another fixed value. In a preferred embodiment the padding constant is derived from the data values of the various data points which make up the template **108**. For example, in one embodiment an average data value is derived for all the temple **108** data points using any of various averaging techniques. This average value serves as the padding constant. For image data, the added data points are pixels and the padding constant is a pixel intensity and/or color. Preferably the center window of the padded template formed by linear padding also is formed by the original template **108**.

Referring again to FIG. **12**, at another step **122** the template **108** is correlated to various windows of the padded template. Because the center of the padded template equals the original template **108**, it is known that the correlation between the template **108** and the center window is 1.0. Thus, that correlation need not be calculated. It is already known. For a two dimensional analysis, a correlation between the original template **108** and windows of the padded template are derived for windows along either of such axes **114**, **116** moving in either direction away from the center window. The step size for selecting adjacent window to evaluate is one data point. Consider for example a template which is 40 pixels by 60 pixels and a padded template which is 120 pixels by 180 pixels. The step size is one pixel. Starting from the center window, there are 40 potential windows in a first direction along the first axis **114** and 40 potential windows in a second, opposite direction along the same axis **114**. In step **122** a correlation is performed between the template and the select windows. As the selected window changes along the first axis **114** in the first direction, the resulting correlation coefficient is likely to decrease below 1.0. Eventually there will be a window where the correlation coefficient falls to a prescribed cut-off value. Such cut-off value may vary for differing embodiment, but preferably is less than a threshold value which identifies an estimated match between a window and the template. A window will be found in the padded template in each direction along axis **114** where the cut-off criteria is met.

Rather than perform a correlation for each potential window along the first axis **114**, correlations are performed for windows along the axis **114** away from the center window in each direction until a window is identified in such direction where the correlation coefficient intersects the cut-off value. For two dimensional analysis, there is a cut-off point found in each direction from the center window along the first axis **114**. The distance between those two windows in data points is the width along the first axis.

Referring to FIG. **12**, at step **124** the first axis step size is derived from the width along the first axis **114** between windows which have a correlation to the template **108** equal to or less than the prescribed cut-off value. The step size along the first axis **114** is a fraction of the width. In a preferred embodiment, one-half the width is taken as the step size for the given axis. In other embodiments, the step size is taken as the entire width or some other fraction of the width.

In steps **126** and **128** the correlations are repeated along the second axis **116** in two opposing directions to find a width along the second axis **116**. For two dimensional analysis, there is a cut-off point found in each direction from the center window along the second axis **116**. The distance between those two windows in data points is the width along the second axis. A fraction of this distance is taken as the step size for the corresponding axis (e.g., first axis, or horizontal, step size; second axis, or vertical, step size). In a preferred embodiment, one-half the width is taken as the step size. In other embodiments, the step size is taken as the entire width or some other fraction of the width. Preferably, the step size along the second axis **116** is derived in the same manner as the step size along the first axis **114**. The step sizes are referred to herein as correlative auto-predictive search ('CAPS') step sizes.

Fast Search:

Once the CAPS step sizes have been derived, a fast search is performed comparing the template **108** to the search area **110**. It is a fast search in the sense that not every potential window of the search area is compared to the template. Referring to FIG. **13**, the search area **110** is shown as an array of data points **74**, **75** such as image pixels points. The two CAPS step sizes are used for selecting windows from the search area **110** to be compared to the template. The data points in the search area **110** about which the template is centered during successive steps are designated with an open circle and part number **75**. Other data pints in the points which are not center points are designated as a data point **74**.

Referring to FIG. **14**, at a step **136** the template **108** (see FIG. **11**) is overlaid to a starting window **112** of the search area **110**. The starting window can be any window of the search area. In a preferred embodiment the starting window **112** is selected by predicting the object location with the motion vector, derived for the previous frame. In one embodiment a linear prediction calculation is implemented, although other more complex prediction algorithms also may be used.

At step **138** a correlation is performed between the template **108** and the starting window and every +/−x-th window along the first axis **114**, where x is the first axis step size. Thus, for a horizontal axis step size of 'x', the template is shifted along the horizontal axis **114** by x data points at a time. More specifically, a center point **77** of the template **108** coincides with a given pixel **75** for a given iteration. The template then is moved to center over another data point **74** that is x points away from the given pixel **75** along the horizontal axis **114**. The template **108** is moved in each direction along the axis **114** using the first step size of x. A correlation is performed at each step.

At step **140** the shifting along the first axis **114** and testing of windows is performed for a template center point repositioned over every y-th row of data points. Specifically, once the initial row of the search area has been tested, the template **108** is moved along the second axis **116** to another row that is y data points away, where y is the second axis step size. This next row then is tested by shifting along the first axis **114** using the first axis step size. A correlation is performed at each iteration. Then another row is tested which is y data points away along the second axis **116**. In this manner the template is shifted by the second step size along the second axis **116** and by the first step size along the first axis **114** to select windows to be tested during the fast search. For example, in a search area which is 400 pixels by 400 pixels, and where the first axis step size is four and the second axis step size is four, there are 100*100=10,000 windows tested during the fast search.

Of the tested windows, at step **142** the window location for any correlation which resulted in a correlation coefficient which is greater than or equal to the product of the cut value times a predetermined threshold value is considered a local match. In a preferred embodiment the cut value is the same for each axis. Where the cut value used along one axis differs from the cut value used along the other axis, either cut value may be used. Alternatively, an average of the cut values may be used. The threshold value is a predetermined value and signifies the minimum correlation coefficient acceptable to designate a window as being a match for the template. Typical values are 0.8 and 0.9. The specific value may vary based upon the search area or type of date. The specific value may be determined empirically for different types of data or search area characteristics.

Local Full Search:

Once the fast search is complete (or during the course of the fast search), a local full search is performed about each of the local matches. For a given window of the search area **110** which is a local match, the windows which are within a 2-dimensional area bounded by the step sizes (for the respective axes) are tested by a local full search. Note that the windows which are exactly a step size away along either axis **114**, **116** were already tested during the fast search. To do the local full search we test all the intermediary windows in the area between the local match and the windows plus or minus one step size away along either axis **114**, **116**. For example, given a first axis step size of x and a second axis step size of y, the windows having a center point which are +/−0, 1, 2, . . . , x−1 data points away from the locally matched window along the first axis, and +/−0, 1, 2, . . . , y−1 data points away from the locally matched window along the second axis, are tested during the full search. Although, the local match need not be recorrelated.

Referring to FIG. **15**, the window corresponding to the local match has a center data point **146**. The template is moved at a step interval of one data point in either direction along either axis up to but not including the data point which in one step size away. As the template is moved over this area, the windows tested during the local full search will have a center data point **148**. FIG. **15** shows all the center points **148** for a given local full search as black dots for an implementation in which the first axis step size is four and the second axis step size is four. FIG. **15** shows the nearby center points from the fast search as open dots **75**.

A correlation is performed between the template **108** and each window in the vicinity of the local match. For the vicinity shown in FIG. **15** in which the step is four, there are **48** additional windows tested. Any of the additional **48** windows or the local match which has a correlation coefficient which equals or exceeds the threshold value is a match of the template. Alternatively, of the windows where the correlation coefficient exceeds the threshold value, only the window or windows having the highest correlation coefficient(s) are selected as matched. For example, one or more windows may have the same correlation coefficient which is highest. As another example the windows corresponding to the top 'n' correlation coefficients may be selected, where each window correlation coefficient also exceeds the threshold value.

Once the template match is found, the corresponding window in the search area is the object being tracked. The relative position of the object within the search area **110** for the current image frame is compared to the relative position of the object in the search area for the prior image frame. The motion vector is derived/updated from the relative positions to define the movement of the object. In one embodiment, the vector is a linear vector derived from respective midpoints of the object from the two image frames. In another embodiment a more complex vector analysis is performed to identify rotation or other two-dimensional or three-dimensional motion of the object being tracked.

In one embodiment the area of the image frame corresponding to the template match is output to the object segmentation subsystem **16**, where the edge potential energy of the object boundary is derived. In addition, a set of data points along the periphery of the template match is sampled to serve as an estimate of the current image object boundary. Such estimate is input to the object segmentation subsystem **18**.

Implementing the Correlation Function:

The correlation coefficient for a correlation between two data sets 'a' and 'b' is defined below. The data set 'a' is the template **108**. The data set 'b' is a window of the padded template (or of a rotational offset of the padded template) for the process of finding the CAPS step sizes. The data set 'b' is a window of the search area **110** (or of a rotational offset of the search area) for the process of identifying candidate locations, potential template matches or template matches. Each of data sets 'a' and 'b' may be a matrix, image or another set of data points. The correlation coefficient, corr is:

$$corr = \frac{E\{[a - E(a)] * [b - E(b)]\}}{sd(a) * sd(b)}$$

which may be simplified to

$$corr = \frac{E(a * b) - E(a) * E(b)}{sd(a) * sd(b)}$$

where E(x)=expected value of data set (x)

sd(x)=standard deviation of data set (x)

and corr is between −1.0 and +1.0.

Edge Energy

Referring to FIG. **2**, edge energy is generated at steps **34** and **50**. More particularly, it is edge potential energy which is derived. Various measures of potential energy may be implemented. In one embodiment a multiple level wavelet detection algorithm is used to extract high frequency components of an image. The high frequency details are analyzed to identify image object edges. In a preferred embodiment Haar wavelet detection is used.

The input to be processed to derive edge potential energy is an image. In one embodiment the image is the entire image frame. In other embodiments, the image is an image object (e.g., the template match area found by the tracking subsystem **14**). The derived edge potential energy is an array of potential energy for each data point (pixel) of the image.

The input image is decomposed by filtering the image with a quadrature mirror filter (QMF) pair which brings out the image details, while simultaneously smoothing the image. The QMF pair includes a high pass filter for bringing out the image details, and a low pass filter for smoothing the image. Referring to FIG. **16** a multiple level QMF decomposition **150** of an image frame **152** is shown. The image frame **152** is passed through a low pass filter **154** and a high pass filter **156** to obtain a low pass component **158** and a high pass component **160**. These components, in turn, are filtered. The low pass component **158** is passed through a low pass filter **162** and a high pass filter **164**. The output of low pass filter **162** is lowpass residue **166**. The output of high pass filter **164** is the horizontal detail **165** of the image frame **152**.

In parallel, the high pass component **160** is passed through a low pass filter **168** and a high pass filter **170**. The output of the low pass filter **168** is the vertical detail **169** of the image frame **152**. The output of the high pass filter **170** is the diagonal detail **171** of the image frame **152**. The low pass residue **166** and the three detailing images **165**, **169**, **171** are the first level QMF decomposition of the image frame **152**. In some embodiments a second level QMF decomposition **172** also is performed in which the low pass residue **166** is input similarly through two stages of low pass and high pass filters to achieve a second-level, low-pass residue and three detailing images (horizontal detail, vertical detail and diagonal detail). In some embodiments the same filters may be used in the second level decomposition as were used in the first level decomposition. for example, the low pass residue **166** is merely input to filters **154**, **156** instead of the image frame **152**.

The high pass filtering function is a wavelet transformation (ψ), while the low pass filtering function is a scaling function (ϕ) corresponding with the wavelet. The scaling function causes smoothing, while the three wavelets bring out the image details.

The scaling function and wavelet transforms in one dimensional space are given by the equations below:

$$\phi_{a,b}(x) = \frac{1}{\sqrt{a}}\phi\left(\frac{x-b}{a}\right),\ a > b,\ b \in R$$

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right),\ a > 0,\ b \in R$$

where, $\phi_{a,b}(x)$ is the family of scaling function at sale a and translated by b;

$\psi_{a,b}(x)$ is the family of wavelets at scale a and translated by b;

a is the scaling factor;

b is the translation desired

ϕ is $\phi_{0,0}$; and

ψ is $\psi_{0,0}$.

Two dimensional wavelets are defined as tensor products of the one-dimensional wavelets. The two-dimensional scaling function is $\phi(x,y)=\phi(x)*\phi(y)$. The two-dimensional wavelets are:

$$\psi_1(x,y)=\phi(x)*\psi(y)$$

$$\psi_2(x,y)=\phi(y)*\psi(x)$$

$$\psi_3(x,y)=\psi(x)*\psi(y)$$

Although the scaling may be varied from one level of decomposition to another, in one embodiment such scaling is not varied.

A first level QMF decomposition is performed. For a second level decomposition the low pass residue **166** of the first level decomposition is analyzed without further downsampling. In some embodiments additional levels of decomposition may be obtained by passing the low pass residue of the prior level through a two stage filtering process (similar to that for the prior levels).

For any given level of decomposition there are four images: the low pass residue, the vertical detail, the horizontal detail and the diagonal detail. The horizontal and vertical detail are gradients of the image along x and y axes. The magnitude of the image is taken at every level of decomposition. The diagonal details have been omitted in one embodiment, because they did not contribute significantly.

In a preferred embodiment up to five levels of decomposition are used for each color component of the image frame, in which the low pass residue from the prior stage is input to the filters **154**, **156** to generate image details and residue for the current stage. Preferably, only data from the even levels (e.g., levels **2**, **4**, and **6**) are used to avoid half-pixel shifts in the edge energy. The integration of the multiple levels and multiple channel (color component) data is guided by their principle component. In one implementation the ratio of multiple-level edge gradients is selected as 1:2:4:8:16 for the five levels of decomposition. With respect to the color components (Y, Cr, Cb), edge gradient ratios of 1:1:1 are used.

In a preferred embodiment the horizontal detail and vertical detail of a given level (i) of decomposition are combined to generate the edge potential energy (EPE) for that level as follows:

$$EPE\ (i)=\text{sqrt}\ [\text{horizontal detail}^2(i)+\text{vertical detail}^2(i)]$$

where i=i-th level of decomposition. For an embodiment in which 5 levels of decomposition are executed, the total edge potential energy (EPE) for a given color component are summed together:

$$EPE_c=EPE_c(2)+2*EPE_c(4)+4*EPE_c(6)+8*EPE_c(8)+16*EPE_c(10)$$

where c is the color component being processed. The overall edge potential energy for the entire frame, inclusive of all color components is the weighted sum of the energy from the different color components. For a weighting factor of (1, 1, 1) the total potential energy is given by:

$$\text{Total Edge Potential Energy}=EPE_y+EPE_{cr}+EPE_{Cb}$$

where Y, Cr and Cb are the color components. In other embodiments R,G and B color components or those of another color component model may be used. The weighting factor may vary depending on the color components model being used.

The total edge potential energy is an array having an energy value for each pixel of the image processed. The edge potential energy is input to the active contour model for use in object segmentation. In some embodiments the edge energy is input to the CAPS process. When providing an input to the CAPS process, the edge energy is being used to predict where the object being tracked is located in a current image frame. For such an embodiment, the "Generate Edge Energy" step **50** is executed prior to the tracking step **48** (see FIG. **2**).

Note that in various embodiments, the edge potential energy is derived before or after the CAPS model executes.

When the edge potential energy is calculated first, a predicted location for the image object may be derived with the edge potential energy as an input to the tracking subsystem **14**. When the CAPS model executes first, the image being processed for edge potential energy is the template matched portion of the image frame.

Object Segmentation

Once an image object has been identified, the image boundary (i.e., edge) is segmented to more accurately model the object edges. In one embodiment the object segmentation subsystem **18** is based upon an active contour modelling method. However, other segmentation methods are known and may be substituted. The active contour modelling method (see subsystem **18**) performs segmentation at step **52** (FIG. **2**) to segment the image object boundary.

Input to the active contour model is the derived total edge potential energy and a current image object boundary. The total edge potential energy is derived at step **50** (see FIG. **2**). For an initial frame the current image object boundary is the boundary input to the system at step **22** (see FIG. **2**). The set of data points for the current image object boundary are used by the active contour model at step **36** (see FIG. **2**).

For subsequent image frames, the current image object boundary is derived by the tracking subsystem **14**, as described above. The set of data points for the current image object boundary are used by the active contour model at step **52** (see FIG. **2**).

Referring to FIG. **17**, a flow chart **192** of the active contour model includes a first step **194** at which edge points are received by the object segmentation subsystem **18**. The number of input edge points may vary. At step **196**, the edge points which are too close together are eliminated, (i.e., less than a first threshold distance apart). In one embodiment points are considered too close together when they are less than 2.5 pixels apart. In other embodiments the distance may be smaller or larger. At step **198** additional points are added by interpolation where the adjacent points are too far apart, (i.e., greater than a second threshold distance apart). In one embodiment points are considered too far apart together when they are greater than 6.0 pixels apart. In other embodiments the distance may be smaller or larger than 6.0 while being larger than the first threshold distance.

At this stage of the process there are a given number of current edge points, as modified from the input edge points. Although the number of edge points may vary from contour to contour, we will describe the process for N current edge points. At step **200** the subsystem **18** performs global relaxation on the N current edge points. To do so, for each current edge point, M candidate points are selected from a box around the current edge point. In one embodiment M equals 4, although in various embodiments the number of candidate points may vary. In one embodiment a 5×5 box is used. However, the size of the box may vary. A larger box leads to a more flexible contour, but more computation time. The shape of the box may be square, rectangular or another shape.

Referring to FIG. **18**, a 5×5 box **174** of pixels surrounding the current edge point **176** is divided into four regions **178**, **180**, **182**, **184**. Within each region there are 6 pixels. One of those 6 pixels is selected in each region to be a candidate pixel ('point') which may potentially replace the current edge point **176** as an object boundary edge point. Thus, 4 candidate points **186–189** are selected for each current edge point **176**. In alternative embodiments a different number of candidate points, or another method of selecting candidate points, may be used.

For a given region **78**, the candidate point is the pixel among the 6 potential points which has the highest edge potential energy. For an image object boundary which has N current edge points, and where there are M (e.g., four) alternative candidate points for each one of the N points, there are $(M+1)^N$ (e.g., 5^N) possible contours from which to select the modelled image object boundary. At step **202** a travel algorithm is applied to the current edge points with the alternative candidate points to select an optimal contour path. FIG. **19** shows a travel path diagram for the possible contours. There are (M+1 (e.g., 5) points in each column. The five points correspond to a current edge point **176** and four candidate edge points **186**, **189** for such current edge point **176**. The number of points in each row (which also equals the number of columns) corresponds to N.

To choose the optimal image object boundary, a starting location **190** on the current contour is selected. Such location **190** corresponds to any given current edge point **176** and its M=4 candidate edge points **186–189**. From each of such M+1=5 points, an optimal path is derived. Of the 5 resulting paths the most optimal path then is selected to be the modelled object boundary. The process for deriving the optimal path is the same for each of the M+1 paths to be derived.

Referring to FIG. **20**, consider a path that is to start from edge point **176**s. A segment of the path is constructed by advancing to one of the M+1 points in the adjacent column s+1. Thus, one choice is to step to point **176**(s+1). Another choice is to step to candidate point **186**(s+1). The others choices include **187**(s+1), **188**(s+1) and **189**(s+1). Only one choice is selected. The choice is made by determining for which of the M+1=5 points in column (s+1) the resulting path has the least difference in energy (e.g., the most energy savings). The selected point is preserved along with a distance of how far such point is from the current point in column s+1. Consider an example where point **186**(s+1) is selected. Such point is preserved along with a distance value (e.g., in pixels) of far many pixels such point is from the point **176**(s+1).

Similarly, to construct the next segment of the path a point among the M+1 points in column s+2 is selected. For each segment along the path only one of the M+1=5 potential segments are preserved, along with a distance from such point to the current point **176** in the same column.

The same process is performed to derive a path which starts from point **186**s. A first segment of the path is constructed by advancing to one of the M+1 points in the adjacent column s+1. One choice is to step to point **176**(s+1). Another choice is to step to candidate point **186**(s+1). The others choices include **187**(s+1), **188**(s+1) and **189**(s+1). Only one choice is selected. The choice is made by determining for which of the M+1=5 points in column (s+1) the resulting path has the most difference in energy relative to the current contour **173**. The selected point is preserved along with a distance of how far such point is from the current point in column s+1. Respective paths starting from point **187**s, **188**s and **189**s, respectively are constructed in the same manner. The M+1 resulting paths then are compared to see which one is the most optimal path, (e.g., most difference in energy relative tot he current contour **173**).

According to this method, rather than perform 5^N computations—one for each one of the potential contours—only (M+1)*(M+1)*N)—(e.g., 5*(5*N))—computations occur.

The energy difference between a contour which steps to the current point for a given point among the 5 potential points at a given step is derived as follows:

$$\Delta E_i = \sum_{i=1}^{N} \delta E_i$$

$$\delta E_i = f_i^1 - f_i^0 - f_i^0 * \frac{d_i^1 + d_i^2 + d_i^3 - d_i^0}{d_i^0}$$

$$= f_i^1 - f_i^0 * \frac{d_i^1 + d_i^2 + d_i^3}{d_i^0}$$

where,

$$f(u1, u2, v1, v2) = \int_{(u1,u2)}^{(v1,v2)} TEPE * ds$$

$$f_i^0 = f(x_i, y_i, x_{i+1}, y_{i+1})$$

$$f_i^1 = f(a_i, b_i, a_{i+1}, b_{i+1})$$

$$d_i^0 = |(x_i, y_i) - (x_{i+1}, y_{i+1})|$$

$$d_i^1 = |(a_i, b_i) - (a_{i+1}, b_{i+1})|$$

$$d_i^2 = |[(a_i, b_i) - (a_{i+1}, b_{i+1})] - [(x_i, y_i) - (x_{i+1}, y_{i+1})]|$$

$$d_i^3 = |(a_i, b_i) - (x_i, y_i)|$$

where TEPE=total edge potential energy

ds=derivative with respect to s (s=length of contour segment between two points)

f_i^0 represents the integral of the total edge potential energy along the i-th segment of the current contour;

f_i^1 represents the integral of the total edge potential energy along the i-th segment of the candidate contour;

d_i^0 represents the length of the i-th segment of the current contour;

d_i^1 represents the length of the i-th segment of the candidate contour;

d_i^2 represents the distance between the two segments when we look at them as vectors;

d_i^3 represents the distance between the i-th current contour point and the i-th candidate contour point.

The terms d_i^0 and d_i^1 correspond to tension in the contour. The term d_i^2 corresponds to stiffness for keeping the shape of the modelled contour similar to the current contour. The term d_i^3 corresponds to pressure to keep the candidate contour close to the current contour. The optimal contour is the one having the optimal ΔE. In one embodiment this is the maximum ΔE. In other embodiments negative TEPE is used instead, so optimum becomes the minimum ΔE.

At the completion of step **202**, the optimal contour is a polygon. As a result, the points identified at step **202** selected from the travel algorithm, may or may not be on the actual smooth object boundary. Thus, fine tuning is performed at step **204**.

Each segment of the optimal contour includes the points selected using the travel algorithm as end points, along with the pixels in between. The pixels in between although not part of the travel problem are part of the input image being processed. In the fine tuning process the pixel along the segment having the highest edge potential energy is selected as the most reliable point of such group for being on the actual object boundary. A most reliable point is selected for each segment of the polygon (i.e., optimal contour path output from the travel algorithm). Points then are selected to be filled in between the most reliable points using the criteria: (i) a new point should be 8 connected to a previously selected boundary point, and (ii) the distance of the new boundary point to the next boundary point should be less than the distance from the previous boundary point to the next boundary point.

Once the object boundary has been fine tuned, the active contour process is repeated with the object boundary of the prior iteration being the current edge points. Global relaxation then is performed again at step **200** to select alternative candidate points for the current edge points. Then the travel algorithm is reapplied at step **202**, followed by fine tuning at step **204**. After the fine tuning step, at step **206** an iteration count is tested to determine if a maximum number of iterations have been executed. If a maximum number of iterations has occurred, then the edge points making up the fine tuned boundary are the image object boundary points output at step **38** (see FIG. **2**). If not, then at step **208** the contour is checked to see if it has changed from the prior iteration. If it has not changed then the edge points making up the fine tuned boundary are the image object boundary points. If the contour has changed, then the process is repeated commencing at step **200** with the global relaxation process.

Exemplary implementations of the object segmentation methods include, but are not limited to video encoding, video editing and computer vision. For example, the segmentation and modelling may be performed in the context of MPEG-4 video encoding and content based video editing in which video objects from different video clips are grouped together to form a new video sequence. As an example of a computer vision application, the segmentation and modelling may be performed with limited user assistance to track a target (e.g., military or surveillance context). Once the target is locked with user assistance, the tracking and segmentation methods automatically provide information to follow the target.

Encoder Subsystem

When other processing is complete, the encoder subsystem **19** is activated to encode and compress the finalized image frame or video sequence into a desired format. In one embodiment a MPEG-4 encoder is implemented.

In one embodiment the operator is able to analyze the output quality by viewing peak signal to noise ratios per color component or per number of bit encoding. In addition, the operator can alter some encoding parameters and view the results for many different encodings to find the encoding settings that provide the desired trade-off to achieve a satisfactory image quality at some number of bits encoded per pixel. By segmenting the object image the operator is able to provide more bits for encoding the segmented image object(s) then for the other portions of the image frame(s). Thus, increased precision is achieved for the image object(s) of interest.

Meritorious and Advantageous Effects

An advantage of the invention is that image segmentation techniques are performed in HSI color space where color sensing properties more closely resemble human vision. According to another advantage of this invention, object boundaries are preserved while noise level is significantly reduced and the noise variance is made more uniform.

Although preferred embodiments of the invention have been illustrated and described, various alternatives, modifications and equivalents may be used. Therefore, the foregoing description should not be taken as limiting the scope of the inventions which are defined by the appended claims.

What is claimed is:

1. A method for segmenting an image frame of pixel data, the image frame including a plurality of pixels, the pixel data corresponding to the pixels, the method comprising:

for each pixel of the image frame, converting the corresponding pixel data into hue, saturation, intensity color space to achieve HSI pixel data having a hue component, a saturation component and an intensity component;

filtering the HSI pixel data to achieve filtered HSI pixel data, wherein said filtering includes: respectively selecting for each one HSI pixel of the image frame, based upon a value of the corresponding intensity component of said each one HSI pixel of the image frame, a first filter kernel from a plurality of filter kernels; and respectively filtering the saturation component of each one HSI pixel using the first filter kernel selected for said one HSI pixel;

identifying presence of an image object in the image frame; and

segmenting the image frame to define a set of filtered HSI pixel data corresponding to the image object.

2. The method of claim 1, further comprising the step of:

encoding the image frame, wherein the pixel data corresponding to the image object is encoded at a higher bit rate than other pixel data corresponding to another portion of the image frame.

3. The method of claim 1, further comprising the step of performing a color gradient operation on the filtered HSI pixel data using a derivative of Gausssian operator; and wherein the step of segmenting comprises segmenting the image frame after the color gradient operation is performed, wherein the set of filtered HSI pixel data corresponding to the image object is filtered HSI pixel data which has received the color gradient operation.

4. The method of claim 1 for segmenting a plurality of image frames included within a motion video sequence of image frames, wherein the steps of converting, filtering, identifying, segmenting and encoding are performed on each one image frame among the plurality of image frames.

5. The method of claim 1, wherein the step of filtering the HSI pixel data comprises:

applying an averaging filter having a kernel size adapted for each pixel, the averaging filter for increasing uniformity of noise distribution of the pixel data within hue, saturation, intensity color space.

6. The method of claim 1, in which said HSI pixel data filtering further comprises:

respectively selecting for each one HSI pixel of the image frame, based upon a product of the intensity component and the saturation component of said each one HSI pixel of the image frame, a second filter kernel from the plurality of filter kernels; and

respectively filtering the hue component of said each one HSI pixel using the second filter kernel selected for said each one HSI pixel.

7. The method of claim 1, in which the step of segmenting comprises applying an active contour model to define an edge of the image object.

8. The method of claim 1, in which the step of identifying the image object comprises identifying a first set of filtered HSI pixels corresponding to the image object, and in which the step of segmenting comprises:

identifying a second set of N filtered HSI pixels corresponding to an initial estimate of a desired contour of the image object, wherein said second set define a current object contour, the first set including at least the second set of HSI filtered pixels;

deriving edge potential energy for the first set of filtered HSI pixels;

refining the current object contour into the desired contour using the current object contour and the derived edge potential energy.

9. A system for segmenting an image frame of pixel data, the image frame including a plurality of pixels, the pixel data corresponding to the pixels, the system comprising:

a processor which converts, for each pixel of the image frame, the corresponding pixel data into hue, saturation, intensity color space to achieve HSI pixel data having a hue component, a saturation component and an intensity component;

a selector which respectively selects for each one HSI pixel of the image frame, based upon a value of the corresponding intensity component of said each one HSI pixel of the image frame, a first filter kernel from a plurality of filter kernels;

a filter receiving the HSI pixel data which generates filtered HSI pixel data, the filter including a saturation component filter which respectively filters the saturation component of each one HSI pixel using the first filter kernel selected for said one HSI pixel;

a processor which identifies presence of an image object in the image frame;

a processor which segments the image frame to define a set of filtered HSI pixel data corresponding to the image object.

10. The system of claim 9, further comprising:

an encoder which encoding the image frame, wherein the pixel data corresponding to the image object is encoded at a higher bit rate than other pixel data corresponding to another portion of the image frame.

11. The system of claim 9, further comprising a processor which performs a color gradient operation on the filtered HSI pixel data using a derivative of Gausssian operator; and wherein the segmented image frame is segmented after the color gradient operation is performed and the set of filtered HSI pixel data corresponding to the image object is filtered HSI pixel data which has received the color gradient operation.

12. The system of claim 9 in which a plurality of image frames included within a motion video sequence of image frames are segmented and encoded, and further comprising a processor which tracks the image object among the plurality of image frames.

13. The system of claim 9, wherein the HSI pixel filter comprises:

an averaging filter having a plurality of kernels of differing kernel size, wherein the kernel size is adapted for each pixel to increase uniformity of noise distribution within hue, saturation, intensity color space.

14. The system of claim 9, in which the selector is a first selector and further comprising a second selector which respectively selects for each one HSI pixel of the image frame, based upon a product of the intensity component and the saturation component of said each one HSI pixel of the image frame, a second filter kernel from the plurality of filter kernels; and wherein said HSI pixel data filter further includes:

a hue component filter which respectively filters the hue component of said each one HSI pixel using the second filter kernel selected for said each one HSI pixel.

15. The system of claim 9, in which the processor which segments applies an active contour model to define an edge of the image object.

16. The system of claim 9, in which the processor which identifies the image object identifies a first set of filtered HSI pixels corresponding to the image object, and in which the processor which segments comprises:

means for identifying a second set of N filtered HSI pixels corresponding to an initial estimate of a desired contour of the image object, wherein said second set defines a current object contour, the first set including at least the second set of HSI filtered pixels;

means for deriving edge potential energy for the first set of filtered HSI pixels;

means for refining the current object contour into the desired contour using the current object contour and the derived edge potential energy.

17. A computer readable storage medium for storing processor-executable instructions and processor-accessible data for segmenting an image frame of pixel data, the image frame including a plurality of pixels, the pixel data corresponding to the pixels, the medium comprising:

means which converts, for each pixel of the image frame, the corresponding pixel data into hue, saturation, intensity color space to achieve HSI pixel data having a hue component, a saturation component and an intensity component;

means for filtering the HSI pixel data to generate filtered HSI pixel data; wherein said filtering means includes: means respectively selecting for each one HSI pixel of the image frame, based upon a value of the corresponding intensity component of said each one HSI pixel of the image frame, a first filter kernel from a plurality of filter kernels; and means for respectively filtering the saturation component of each one HSI pixel using the first filter kernel selected for said one HSI pixel; and

means for identifying presence of an image object in the image frame.

18. The medium of claim 17, further comprising:

means for segmenting the image frame to define a set of filtered HSI pixel data corresponding to the image object.

19. The storage medium of claim 17, further comprising:

means for encoding the image frame, wherein the pixel data corresponding to the image object is encoded at a higher bit rate than other pixel data corresponding to another portion of the image frame.

20. The storage medium of claim 17, further comprising means which performs a color gradient operation on the filtered HSI pixel data using a derivative of Gaussian operator; and wherein the segmented image frame is segmented after the color gradient operation is performed and the set of filtered HSI pixel data corresponding to the image object is filtered HSI pixel data which has received the color gradient operation.

21. The storage medium of claim 17, wherein the HSI pixel filtering means comprises:

a plurality of kernels of differing kernel size, wherein the kernel size is adapted for each pixel to increase uniformity of noise distribution within hue, saturation, intensity color space.

22. The storage medium of 17, in which the selecting means is a first selecting means and wherein the HSI pixel filtering means further comprises:

a second selecting means which respectively selects for each one HSI pixel of the image frame, based upon a product of the intensity component and the saturation component of said each one HSI pixel of the image frame, a second filter kernel from the plurality of filter kernels; and

a hue component filter which respectively filters the hue component of said each one HSI pixel using the second filter kernel selected for said each one HSI pixel.

23. A method for filtering an image portion, the image portion comprising a plurality of pixel data, the method comprising the steps of:

converting the plurality of pixel data into hue, saturation, intensity color space to achieve HSI pixel data having a hue component, a saturation component and an intensity component;

respectively selecting and applying for each one pixel of the HSI pixel data, a first filter kernel from a plurality of filter kernels, said first kernel filtering the saturation component of said each one pixel of the HSI pixel data to achieve a filtered saturation component of the HSI pixel data; and

respectively selecting and applying for each one pixel of the HSI pixel data, a second filter kernel from the plurality of filter kernels, said second kernel filtering the hue component of said each one pixel of the HSI pixel;

wherein said selecting the first filter kernel to filter the saturation component comprises testing the intensity component of the corresponding HSI pixel data against a set of threshold values to determine which filter kernel among the plurality of filter kernels is applied to filter the saturation component.

24. The method of claim 23, in which the step of selecting the second filter kernel to filter the hue component comprises testing a product of the intensity component and the filtered saturation component of the corresponding HSI pixel data against a set of threshold values to determine which filter kernel among the plurality of filter kernels is applied to filter the hue component.

25. The method of claim 23, in which the image portion is an image frame among a sequence of image frames, the method further comprising the steps of:

identifying presence of an image object in the image frame;

segmenting the image frame to define a set of filtered HSI pixel data corresponding to the image object.

26. The method of claim 25, further comprising the step of performing a color gradient operation on the filtered HSI pixel data using a derivative of Gaussian operator; and wherein the step of segmenting comprises segmenting the image frame after the color gradient operation is performed, wherein the set of filtered HSI pixel data corresponding to the image object is filtered HSI pixel data which has received the color gradient operation.

27. A system for filtering an image portion, the image portion comprising a plurality of pixel data, the system comprising:

a processor which converts the plurality of pixel data into hue, saturation, intensity color space to achieve HSI pixel data having a hue component, a saturation component and an intensity component; and

an averaging filter having a kernel size adapted for each pixel, the averaging filter increasing uniformity of noise distribution of the HSI pixel data, the averaging filter comprising:

means for respectively selecting and applying for each one pixel of the HSI pixel data, a first filter kernel from a plurality of filter kernels, said first kernel filtering the saturation component of said each one pixel of the HSI pixel data to achieve a filtered saturation component of the HSI pixel data; and

means for respectively selecting and applying for each one pixel of the HSI pixel data, a second filter kernel from the plurality of filter kernels, said second kernel filtering the hue component of said each one pixel of the HSI pixel;

wherein the means for selecting the first filter kernel to filter the saturation component comprises means for testing the intensity component of the corresponding HSI pixel data against a set of threshold values to determine which filter kernel among the plurality of filter kernels is applied to filter the saturation component.

28. The system of claim **27**, in which the means for selecting the second filter kernel to filter the hue component comprises means for testing a product of the intensity component and the filtered saturation component of the corresponding HSI pixel data against a set of threshold values to determine which filter kernel among the plurality of filter kernels is applied to filter the hue component.

29. The system of claim **27**, in which the image portion is an image frame among a sequence of image frames, the system further comprising:

means for identifying presence of an image object in the image frame; and

means for segmenting the image frame to define a set of filtered HSI pixel data corresponding to the image object.

30. The system of claim **29**, further comprising: means for performing a color gradient operation on the filtered HSI pixel data using a derivative of Gaussian operator; and wherein the segmenting means comprises means for segmenting the image frame after the color gradient operation is performed, wherein the set of filtered HSI pixel data corresponding to the image object is filtered HSI pixel data which has received the color gradient operation.

31. A computer readable storage medium for storing processor-executable instructions and processor-accessible data for filtering an image portion, the image portion comprising a plurality of pixel data, the medium comprising:

means for converting the plurality of pixel data into hue, saturation, intensity color space to achieve HSI pixel data having a hue component, a saturation component and an intensity component;

means for respectively selecting and applying for each one pixel of the HSI pixel data, a first filter kernel from a plurality of filter kernels, said first kernel filtering the saturation component of said each one pixel of the HSI pixel data to achieve a filtered saturation component of the HSI pixel data; and

means for respectively selecting and applying for each one pixel of the HSI pixel data, a second filter kernel from the plurality of filter kernels, said second kernel filtering the hue component of said each one pixel of the HSI pixel;

wherein the means for selecting the first filter kernel to filter the saturation component comprises means for testing the intensity component of the corresponding HSI pixel data against a set of threshold values to determine which filter kernel among the plurality of filter kernels is applied to filter the saturation component.

32. The medium of claim **31**, in which the means for selecting the second filter kernel to filter the hue component comprises means for testing a product of the intensity component and the filtered saturation component of the corresponding HSI pixel data against a set of threshold values to determine which filter kernel among the plurality of filter kernels is applied to filter the hue component.

* * * * *