PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06K 9/48

A1

(11) International Publication Number: WO 94/08315

(43) International Publication Date: 14 April 1994 (14.04.94)

(21) International Application Number: PCT/US93/09159

(22) International Filing Date: 24 September 1993 (24.09.93)

(30) Priority data:
07/956,056 2 October 1992 (02.10.92) US

(71) Applicant: INTERNATIONAL REMOTE IMAGING SYSTEMS, INC. [US/US]; 9162 Eton Avenue, Chatsworth, CA 91311-5805 (US).

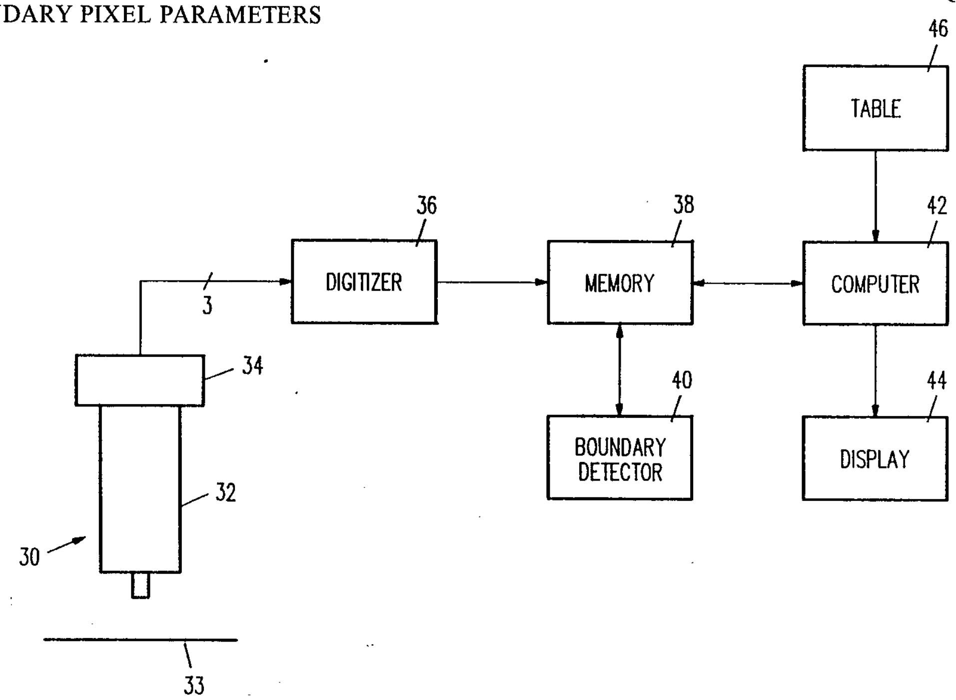
(72) Inventor: KASDAN, Harvey, Lee ; 5414 Sunnyslope Avenue, Van Nuys, CA 91401 (US).

(74) Agents: YIN, Ronald, L. et al.; Limbach & Limbach, 2001 Ferry Building, San Francisco, CA 94111 (US).

(81) Designated States: AU, CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND APPARATUS FOR IDENTIFYING AN OBJECT USING AN ORDERED SEQUENCE OF BOUNDARY PIXEL PARAMETERS



(57) Abstract

The present invention relates to a method and apparatus for identifying an examined object having a discernible boundary. An image of the object is formed (34) and segmented into a plurality of pixels (36). The boundary of the object in the image is detected (40) and the parameters of the boundary pixels form an ordered sequence: p1...pk...pN, where p1 is the parameter value of the first boundary pixel; where pN is the parameter value of the last boundary pixel; N is the total number of boundary pixels and k is the index to the N pixels. The pk data point is determined (42) such that k satisfies the relationship (I): $i-1/M < k/N \leq i/M$, where M is the total number of quantile partitions and i is the ith quantile partition. The object is identified based upon the pk data point value, which is the ith quantile in M partitions developed from the above relationship.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**FR**	France	**MR**	Mauritania
AU	Australia	**GA**	Gabon	**MW**	Malawi
BB	Barbados	**GB**	United Kingdom	**NE**	Niger
BE	Belgium	**GN**	Guinea	**NL**	Netherlands
BF	Burkina Faso	**GR**	Greece	**NO**	Norway
BG	Bulgaria	**HU**	Hungary	**NZ**	New Zealand
BJ	Benin	**IE**	Ireland	**PL**	Poland
BR	Brazil	**IT**	Italy	**PT**	Portugal
BY	Belarus	**JP**	Japan	**RO**	Romania
CA	Canada	**KP**	Democratic People's Republic of Korea	**RU**	Russian Federation
CF	Central African Republic			**SD**	Sudan
CG	Congo	**KR**	Republic of Korea	**SE**	Sweden
CH	Switzerland	**KZ**	Kazakhstan	**SI**	Slovenia
CI	Côte d'Ivoire	**LI**	Liechtenstein	**SK**	Slovak Republic
CM	Cameroon	**LK**	Sri Lanka	**SN**	Senegal
CN	China	**LU**	Luxembourg	**TD**	Chad
CS	Czechoslovakia	**LV**	Latvia	**TG**	Togo
CZ	Czech Republic	**MC**	Monaco	**UA**	Ukraine
DE	Germany	**MG**	Madagascar	**US**	United States of America
DK	Denmark	**ML**	Mali	**UZ**	Uzbekistan
ES	Spain	**MN**	Mongolia	**VN**	Viet Nam
FI	Finland				

METHOD AND APPARATUS FOR IDENTIFYING AN OBJECT USING AN ORDERED SEQUENCE OF BOUNDARY PIXEL PARAMETERS

****

Technical Field

The present invention relates to a method and an apparatus for identifying an object under examination, wherein an image of the object is formed. More particularly, the present invention relates to a method and an apparatus for identifying an object based upon the object having a discernable boundary and a novel feature termed "quantile".

Background Of The Invention

Methods and apparatuses for identifying an object are well-known in the art. The art of identifying biological samples is replete with various techniques for identifying the type of biological samples under examination. See, for example, U.S. Patent No. 4,175,860. Formation of histograms of various parameters are also known. See for example U. S. Patent No. 3,851,156.

Heretofore, one of the known methods for identifying an object is to use the size of the object. Other parameters of the object which can be used to identify an object include color, internal optical density, and average intensity within the boundary of the object.

In some cases, however, it is not possible to identify an object uniquely or to differentiate two objects if certain of their parameters yield the same result. Thus, for example, if two objects have the same size and have the same internal optical density and have the same average visible light intensity, then it is not possible to identify one of the objects or to differentiate these two objects, based upon these parameters. In addition, some of these parameters are computationally intensive requiring considerable amount of time or computer resources. Thus, the present invention contemplates a novel method and apparatus for identifying an object or to differentiate two objects based upon a new parameter which is termed "quantile".

Summary Of The Invention

In the present invention, a method of identifying an object, under examination, which has a discernible boundary is disclosed. The method comprises the steps of forming an image of the object. The image is segmented to form a plurality of pixels. The boundary of the object under examination in the image is then detected. A measurement is made of a parameter of each pixel within the boundary detected. The parameter values of the pixels measured within the boundary detected are formed in an ordered sequence. The particular measured parameter value P_k, where k is an index of the ordered sequence, is determined such that the index k satisfies the following relationship:

$$\frac{i-1}{M} < \frac{k}{N} \leq \frac{i}{M}$$

N - is the total number of pixels;

M - is the total number of quantile partitions

i - index corresponding to the i^{th} fraction of partition or i^{th} quantile

The object under examination is then identified based upon the P_k parameter value which is the i^{th} quantile within M partitions developed from the above relationship associated with the object.

The present invention also contemplates an apparatus to carry out the foregoing method.

Brief Description Of The Drawings

Figure 1 is a schematic graphical illustration of two objects "a" and "b" which have the same size and shape and the same average intensity but which are different.

Figure 2 is a graph showing a histogram of intensity of light versus the number of pixels in the boundary detected.

Figure 3 is a graph showing a cumulative histogram of intensity of light versus the cumulative number of pixels in the boundary detected, and with the quantile feature determined for each object.

Figure 4 is a table of pre-determined relationships or quantiles of various known objects.

Figure 5 is an ordered sequence of the values of the parameter measured for the pixels in the boundary.

Figure 6 is a block diagram of an apparatus suitable for carrying out the method of the present invention.

Detailed Description Of The Drawings

Referring to Figure 1, there is shown a schematic diagram of an image 10 of two objects: object "A" 12 and "B" 14. Although the description set forth hereinafter is with reference to the method of the present invention to distinguish object "A" from object "B", it can be seen that the method of the present invention is equally applicable to identifying a single object which is under examination.

Each of the objects 12 and 14 has a discernible boundary 18 and 20, respectively. As shown in Figure 1, each of the objects 12 and 14 has the same size and shape. Although for illustration purposes, objects 12 and 14 are shown as having the same size and shape, it is not necessary to the practice of the present invention. The purpose of illustrating the method of the present invention using objects 12 and 14 having the same size and shape is to show how the method of the present invention can be used to distinguish objects 12 and 14 under examination when other parameters, such as size and shape, cannot be used.

Objects 12 and 14, of course, represent images of real objects, such as cells, genes or other biological or non-biological objects. In the method of the present invention, an image 10 is taken of the objects 12 and 14. The image 10 can be formed by using an electronic video camera, such as a CCD (the specific details of an apparatus suitable to carry out the method of the present invention will be discussed hereinafter). The image 10 is then filtered in accordance with the different color filters that can be used to distinguish the type of color of the object. Each of the different color images is segmented, to form a plurality of pixels. Each pixel is then digitized. The boundary of each of the objects 12 and 14 is determined. This can be done, for example, by the method disclosed in U.S. Patent No. 4,538,299, which is incorporated herein by reference.

Once the boundary of each of the objects 12 and 14 is determined, one of the parameters of each pixel within the boundary is measured. One parameter can be the intensity of visible light. Thus, the intensity of visible light at each pixel, positioned within the boundary positions 18 and 20, is measured.

One prior art method is to form a histogram of intensity versus the number of pixels. Referring to Figure 2, there is shown two histograms. The histogram of Figure 2a corresponds to the object 12 whereas the histogram of Figure 2b corresponds to the object 14. As can be seen from Figure 1, object 12 has a substantially uniform intensity (the dark image indicates brightness) throughout the entire region within the boundary 18. Thus, the histogram, as shown in Figure 2a, shows substantially N_1 pixels each having an intensity of I_1. In contrast, object 14 has a small area 16 that has greater intensity than the uniform intensity I_1 of object 12. The histogram, as shown in Figure 2b, shows that N_2 number of pixels in the spot 16 have substantially an intensity value of I_2, with $N_1 > N_2$ and $I_2 > I_1$.

Once the histogram of each object 12 and 14 has been formed, a cumulative histogram is then developed. The cumulative histogram is formed by summing the number of pixels that has at least a certain intensity value. If N = f(I), where N is the number of pixels having an intensity level I, then Cum N = F(I) where Cum N is the cumulative number of pixels having an intensity level ≤I, and where

$$F(I) = \sum_{j=o}^{I} f(j)$$

The cumulative histograms of the histograms shown in Figure 2a and 2b, are shown in Figure 3.

The particular parameter value, such as intensity of visible light, associated with a cumulative fraction of all pixels, is termed a quantile. Thus, from Figure 3, the 50th quantile is

I_3 for object 12 and I_4 for object 14. It should be noted that unlike conventional percentile representations, which are even increments of discrete percents, quantile representations are not limited to discrete, even increments. Thus, an object may be identified by its ith, jth, or kth quantile, where i, j, and k do not necessarily occur in discrete, even increments.

Since the quantile relationship, shown in Figure 3, associated with each objects 12 and 14 is different, the objects 12 and 14 can be distinguished from one another, based upon this developed quantile relationship. Alternatively, to uniquely identify a particular object, such as object 14, the quantile relationship associated with the object 14 can be compared to a table of pre-determined relationship of other known objects whose quantile relationships have been pre-determined. The table of pre-determined quantile relationships of other known objects can be based upon experimental results. The comparison of the quantile relationship to the table of pre-determined quantile relationships would then serve to identify uniquely the type of the object that is under examination.

For the purpose of distinguishing the objects 12 and 14 from one another or to identify an unknown object by comparing the quantile relationship, it is not necessary to compare each ordered pair of numbers (Q,I) (where Q is the quantile number and I is intensity associated therewith), to each other (in the case of two particles) or to a table (in the case of attempting to uniquely identify an object). One method is to compare the associated intensity value of the object under examination at a particular quantile number to the associated intensity value of another object at the same quantile number, or to a table of associated intensity values of identified objects at the same quantile number. Thus, as shown in Figure 3, the 50th quantile for the object 12 has an intensity value of I_3 and the same 50th quantile for the object 14 has the value I_4. These intensity values can be compared to each other to distinguish one object from another. Alternatively, the 50th quantile of one of the objects can be compared to a table of intensity values of identified particles whose 50th quantile have been pre-determined. See Figure 4. By comparing the 50th quantile of the unknown object to the table, the unknown object can be identified.

Of course, if we are dealing with objects of biological particles having statistical variations in intensity, it may not be possible to determine precisely the intensity associated with a particular quantile for the same particle under all cases of examination. Thus, a table relationship may encompass a range of values of intensity for a particular quantile such that the object having that range of intensity values, can be uniquely identified. The range of values can be developed from representative samples identified from experimental results.

The foregoing example illustrates a method of distinguishing objects wherein the average intensity of the two objects 12 and 14 are indistinguishable. However, using the intensity of visible light and the quantiles developed therefor, the objects can be distinguished from one another. Parameters, other than intensity of light can also be used to distinguish the particles. Thus, another parameter suitable for use is differentiating by color representation. Color can be represented by three primary colors of red, blue, and green. Alternatively, color can also be represented by hue, intensity, and saturation; or by cyan, magenta and yellow.

In one particular embodiment, the difference in color representation of the primary colors is used. The parameter of log(a) - log(b) may be employed, where a, and b are intensities of red and blue primary colors, respectively. Other possible combinations include: a being green and b being red; or a being green and b being blue.

The problem with the foregoing method of calculating quantiles is that a cumulative histogram must be formed. This necessitates the creation of a plurality of discrete "bins". A bin is defined by a cumulative fraction of all the pixels from a first fraction to a second fraction. As can be seen from Figure 3, the entire value of the cumulative histogram from 0 to I_T must be divided into a number of discrete bins. The i^{th} quantile is the value of I corresponding to the i^{th} cumulative fraction of the pixels. For parameters, such as log(a) - log(b), as mentioned hereinabove, the range of values can be enormous (due to the nature of the logarithmic operation). Thus, the choice of the boundary of the bins may become difficult.

In the method of the present invention, quantiles can be calculated without calculating a cumulative histogram. In the method of the present invention, the values of the parameters measured for each of the pixels within the boundary detected are placed in an ordered sequence. This is shown in Figure 5. As previously discussed, for the example used in the method of the present invention, the objects 12 and 14 are assumed to be the same size and shape. Thus, each of those objects 12 and 14 would be segmented into the same number of total pixels, N. The parameter, for example of intensity of visible light, for each of the pixels within the boundary is then placed in an ordered sequence. P_1 is the value of the pixel having the lowest intensity of visible light. P_N is the value of the pixel having the largest value of the parameter, such as intensity visible light. The values of the parameter of the pixels measured are then formed in an ordered sequence. A value of P_k represents the parameter of the pixel between the smallest value and the largest value with k being the index to the N pixels.

Thereafter, k is chosen such that k satisfies the following relationship:

$$\frac{i-1}{M} < \frac{k}{N} \leq \frac{i}{M}$$

where M is the total number of quantile partitions and i is the i^{th} fraction of partition or i^{th} quantile. For the i^{th} chosen quantile, once k is determined, the intensity value of the data point P_k would be the intensity value associated with the i^{th} quantile, or simply the value of the ith quantile. Thereafter, the object under examination can be distinguished from other objects based upon the i^{th} quantile.

Alternatively, the object under examination can be distinguished from all other objects based upon a table of predetermined values of corresponding i^{th} quantile. This can be accomplished even where the two objects do not have the same number of pixels. Thus, for example, object 1 has N_1 pixels and Object 2 has N_2 pixels. The parameters of the pixels are placed in an ordered sequence as follows:

$P_1 \ldots P_{k1} \ldots P_{N1}$ - object 1 having N_1 pixels

$P_1 \ldots P_{k2} \ldots P_{N2}$ - object 2 having N_2 pixels

where k is the index. For each object, the P_k data point is determined such that k satisfies the following:

$$\frac{i-1}{M} < \frac{k}{\text{Total No. of pixels for the object}} \leq \frac{i}{M}$$

M - total number of quantile partitions;

i - i^{th} fraction of partition or i^{th} quantile partition

where for Objects 1 and 2 i and M are chosen to be the same. The resultant value of P_k for each object is compared to one another to differentiate the objects. Of course, as previously discussed, Objects 1 and 2 can be the same type of object even if P_{k1} does not equal P_{k2}. There can be a range of values for a particular quantile such that objects having values within that range will still be the same type.

As can be seen from the foregoing, with the method of the present invention, no cumulative histogram calculation is made. As a result, there is no need to set the size of the bin to calculate the cumulative histogram. A simple comparative relationship to determine the index of the data points is determined from which the value of the data point of the parameter measured is then obtained which corresponds to the quantile of interest.

An example of the method of the present invention can be seen with reference to the following Table 1, with an example of three particles: Particle 1, Particle 2, and Particle 3, each having 50 pixels of varying intensity of the same parameter.

TABLE 1

Pixel#	Particle 1	Particle 2	Particle 3
1	6.55	3.77	2.64
2	6.98	5.56	2.60
3	12.24	6.45	1.92
4	9.38	4.39	1.82
5	8.28	5.28	2.25
6	12.66	6.50	2.46
7	8.04	3.14	2.23
8	6.46	5.78	2.83
9	10.12	4.18	2.15
10	7.90	7.58	2.83
11	8.94	5.45	1.98
12	9.99	4.47	2.07
13	8.78	5.21	2.30
14	9.16	4.55	1.86
15	15.12	5.21	2.26
16	13.26	3.47	1.90
17	9.04	4.25	8.44
18	14.82	4.65	6.72
19	11.36	4.25	4.98
20	8.64	5.96	5.61
21	30.53	6.16	6.36
22	33.61	4.61	4.70
23	30.36	3.42	9.32
24	30.21	4.38	9.30
25	31.12	2.87	5.86
26	27.71	6.49	6.96
27	32.21	4.38	2.96
28	33.27	5.74	4.74
29	34.31	6.84	8.42
30	41.23	3.43	6.32
31	26.76	6.03	6.72
32	27.83	4.92	8.90
33	30.93	5.23	6.64
34	29.12	6.59	4.02
35	26.33	4.19	5.53
36	30.59	4.87	8.84
37	33.00	4.01	6.19
38	29.07	6.34	8.75
39	26.01	3.96	4.17
40	36.96	6.95	6.91
41	34.02	91.44	91.44
42	30.15	90.50	90.50
43	30.13	95.96	95.96
44	21.58	92.91	92.91
45	34.80	88.72	88.72
46	21.75	97.00	97.00
47	27.42	94.56	94.56
48	30.47	94.97	94.97
49	28.36	94.35	94.35
50	29.58	93.19	93.19

The next table, Table 2, shows various statistical values computed for the three particles.

TABLE 2

	Particle 1	Particle 2	Particle 3
Mean	22.182	22.682	22.518
Standard Error	1.504	5.053	5.073
Median	27.088	5.314	6.025
Mode	#N/A	#N/A	#N/A
Standard Deviation	10.637	35.729	35.872
Variance	113.141	1276.597	1286.779
Kurtosis	-1.580	0.421	0.402
Skewness	-0.219	1.547	1.535
Range	34.763	94.129	95.341
Minimum	6.464	2.872	1.660
Maximum	41.227	97.001	97.001
Sum	1109.125	1134.118	1125.918
Count	50	50	50

The next table, Table 3, shows the creation of a plurality (10) of "bins" with each bin containing a count of the number of pixels having a value between a first count to a second count. Thus, Bin #1 contains all the pixels having intensity values between 0 and 10. For particle #1, there would be 13 such pixels. For particles #2 and #3, there would be 40 and 40 respectively. The 10 bins are chosen to partition the maximum range of pixel intensity values into 10 fractions. Since the maximum range of intensity value is zero to 41.23, 97.00, and 97.00 for particles #1, #2 and #3 respectively, the 10 bins are chosen to delineate pixel intensity values of 0-10-20-... -90-100. The column entitled Cumulative Histogram is a cumulative count (shown as a %) of the total number of pixels having an intensity value between zero and the value associated with the Bin #. As can be seen from Table 3, particles #2 and #3 are indistinguishable based upon cumulative histogram. Further, as previously discussed, for intensity values based upon log( ), the value can be large. Thus, the range of pixel intensity values will be large, rendering difficult the division of the intensity values into a manageable number of bins.

TABLE 3

Bin #	Intensity Value of Pixels	Particle 1		Particle 2		Particle 3	
		Histogram	Cum.Hist.	Histogram	Cum.Hist.	Histogram	Cum.Hist.
1	0-10	13	26.00%	40	80.00%	40	80.00%
2	10-20	7	40.00%	0	80.00%	0	80.00%
3	20-30	12	64.00%	0	80.00%	0	80.00%
4	30-40	17	98.00%	0	80.00%	0	80.00%
5	40-50	1	100.00%	0	80.00%	0	80.00%
6	50-60	0	100.00%	0	80.00%	0	80.00%
7	60-70	0	100.00%	0	80.00%	0	80.00%
8	70-80	0	100.00%	0	80.00%	0	80.00%
9	80-90	0	100.00%	1	82.00%	1	82.00%
10	90-100	0	100.00%	9	100.00%	9	100.00%

The following table, Table 4, shows the table of pixel intensity values (Table 1) rearranged in accordance with the method of the present invention. The pixel intensity values are ordered in a sequential order, from highest to lowest. (Of course, the ordered sequence can be from the lowest to the highest.) In this example, P_{50} for each of the three particles have the values 6.46, 2.87 and 1.66, respectively, while P_1 is 41.23, 97.00 and 97.00 respectively.

TABLE 4

Rank	Percent	Particle 1 Pixel#	Particle 1 Int.	Particle 2 Pixel#	Particle 2 Int.	Particle 3 Pixel#	Particle 3 Int.
1	100.00%	30	41.23	46	97.00	46	97.00
2	97.95%	40	36.96	43	95.96	43	95.96
3	95.91%	45	34.80	48	94.97	48	94.97
4	93.87%	29	34.31	47	94.56	47	94.56
5	91.83%	41	34.02	49	94.35	49	94.35
6	89.79%	22	33.51	50	93.19	50	93.19
7	87.75%	28	33.27	44	92.91	44	92.91
8	85.71%	37	33.00	41	91.44	41	91.44
9	83.67%	27	32.31	42	90.50	42	90.50
10	81.63%	25	31.12	45	88.72	45	88.72
11	79.59%	33	30.93	10	7.58	23	9.32
12	77.55%	36	30.59	40	6.95	24	9.30
13	75.51%	21	30.53	29	6.84	32	8.90
14	73.46%	48	30.47	34	6.59	36	8.84
15	71.42%	23	30.36	6	6.50	38	8.75
16	69.38%	24	30.21	26	6.49	17	8.44
17	67.34%	42	30.15	3	6.45	29	8.42
18	65.30%	43	30.13	21	6.16	26	6.96
19	63.26%	50	29.58	31	6.03	40	6.91
20	61.22%	34	29.12	20	5.96	18	6.72
21	59.16%	38	29.07	8	5.78	31	6.72
22	57.14%	49	28.36	28	5.74	33	6.64
23	55.10%	32	27.83	2	5.56	21	6.36
24	53.06%	26	27.71	11	5.45	30	6.32
25	61.02%	47	27.42	38	5.34	37	6.19
26	48.97%	31	26.76	5	5.28	25	5.86
27	46.93%	25	26.33	33	5.23	20	5.81
28	44.89%	29	26.01	15	5.21	35	5.53
29	42.85%	46	21.75	13	5.21	19	4.98
30	40.81%	44	21.58	32	4.92	28	4.74
31	38.77%	15	15.12	36	4.87	22	4.70
32	36.73%	18	14.82	18	4.65	39	4.17
33	34.69%	16	13.26	22	4.61	34	4.02
34	32.65%	6	12.66	14	4.55	27	2.96
35	30.61%	3	12.24	12	4.47	8	2.83
36	28.57%	19	11.36	4	4.39	1	2.64
37	26.53%	9	10.12	27	4.38	2	2.60
38	24.48%	12	9.99	24	4.38	6	2.46
39	22.44%	4	9.38	19	4.25	13	2.30
40	20.40%	14	9.16	17	4.25	15	2.25
41	18.36%	17	9.04	35	4.19	5	2.25
42	16.32%	11	8.94	9	4.18	7	2.23
43	14.28%	13	8.78	37	4.01	9	2.15
44	12.24%	20	8.64	39	3.96	12	2.07
45	10.20%	1	8.55	1	3.77	11	1.98
46	8.16%	5	8.28	16	3.47	3	1.92
47	6.12%	7	8.04	30	3.43	16	1.90
48	4.08%	10	7.90	23	3.42	14	1.86
49	2.04%	2	6.98	7	3.14	4	1.82
50	0.00%	8	6.46	25	2.87	10	1.66

where Rank is the order in sequence by the intensity of the pixel, Pixel# is the number of the pixel from Table 1, Intensity is the value of the intensity for the corresponding Pixel# from Table 1, and Percent is the percent of particles that would have intensity value shown under Intensity or less.

Thus, the pixel having the Rank of 3 for Particle 1 is Pixel# 45 from Table 1. Pixel# 45 for Particle 1 has an intensity value of 34.80. Further 95.91% of all pixels in Particle 1 have intensity values of 34.80 or less.

If the number of "bins" or fractions, M is chosen to be 10 (the same as the example based upon a cumulative histogram calculation), and N-number of pixels = 50, then

$$\frac{i-1}{10} < \frac{k_i}{50} \leq \frac{i}{10}$$

where i is the i^{th} bin. If i=10

$$\frac{9}{10} < \frac{k_{10}}{50} \leq 1$$

k_{10} = 46, 47, 48, 49, 50.

If we choose k_{10} = 50, then P_{k10} = 6.46, 2.87, 1.66 for particles #1, #2 and #3, respectively.

Since P_{k10} = 2.87 for particle #2 clearly differs from P_{k10} = 1.66 for particle #3, particles #2 and #3 can be differentiated, while other statistical based analysis such as histogram, and cumulative histogram are unable to differentiate the particles. Furthermore, although a single value was compared, clearly a range of values can be compared. Note that the same holds true for P_{k9} (K_9 = 45) = 8.55, 3.77, 1.98 and P_{k8} (K_8 =40) = 9.16, 4.25, 2.25 for particles #1, #2 and #3 respectfully.

An apparatus 30 suitable for carrying out the method of the present invention is shown in Figure 6. The apparatus 30 comprises a microscope 32, which is focused on an examination area 33. The examination area can be a microscopic slide or a flow cell such as that disclosed in U.S. Patent No. 4,338,024. A camera 34 is attached to the microscope 32 and is adapted to image a portion of a suspension having a particle therein. The camera 34 is preferably of a raster scan type and may be a CCD camera model XC-711 manufactured by Sony. The camera 34 can form an electrical image of the field of view as seen through the microscope 32. Further, the camera 34 segments the image into a plurality of pixels, with an electrical signal corresponding to each pixel of the image. The camera 34 also outputs a plurality of signals (3) one for each of the colors (Red, Blue, and Green).

Each of the signals from the camera 34 is supplied to a digitizer 36, which digitizes the image intensity of each pixel into an electrical signal representing the grey scale value. Preferably, the pixel is digitized into a gray scale value between 0 and 255, inclusive.

From the digitizer, the digitized gray scale value is supplied to a memory 38, where the values are stored. The memory 36 can be a RAM. A boundary detector 40, such as that disclosed in U.S. Patent No. 4,538,299 operates on the image in the memory 38 and detects the boundary of the image detected. The result is also stored in the memory 38.

From the memory 38, a computer 42 operates on the image and measures a parameter, such as intensity of light, of each pixel within the boundary detected.

In the event the apparatus 30 seeks to determine the identity of a single particle under examination, then the particular quantile value derived by the computer 42 is compared to the table of stored values 46 of previously identified particles. The table 46 can be a ROM or another memory. The result can be displayed on the display 44.

In the event the apparatus 30 seeks to differentiate a plurality of objects under examination, then a quantile is developed for the same parameter for each of the pixels in the boundary of each of the images of the different objects. From the quantile, the particles under examination can be differentiated from one another. The result can be displayed on the display 44.

The computer 42 carries out its tasks in accordance with a computer program. A copy of a program to compute quantiles written in the C language for execution under the MS-DOS operating system is as follows:

```
/*      cca.h()         Block Diagram Software                  cca.h()
**
**      Programmer:     H. L. Kasdan
**
**      Copyright :     1987 by Harvey L. Kasdan
**
*************************************************************************
**
**      NAME
**              cca.h - chain code analysis header file
**
**      SYNOPSIS
**
**      DESCRIPTION
**
**      SEE ALSO
**
**      DIAGNOSTICS
**
**      BUGS
**
**      REVISION HISTORY
**
*************************************************************************
*/
/*
*************************************************************************
**                      DEFINES
*************************************************************************
*/
#define MAXX 60
#define MAXY 60
/*
*************************************************************************
**                      STRUCTURE DEFINITION
*************************************************************************
```

```
*/
struct ccdef {
            char ccang;                 /* angular position of next point 0 - 7 */
            int x;                      /* x-position of current point */
            int y;                      /* y-position of current point */
            };                                  /* chain code definition structure */

typedef char n8group[9];                /* 8-group of characters (pixels) */

typedef struct {
              int px;                           /* x-coordinate */
              int py;                           /* y-coordinate */
              } point;                          /* point coordinates */

typedef point n8points[9];              /* array of central point and 8 neighbors */

/*
****************************************************************************
**                       SUBROUTINE DECLARATIONS
****************************************************************************
*/
int ccabox0();
int ccabrk0();
int ccaconsc();
int ccaconss();
int ccadcon0();
int ccadetp0();
int ccadilate();
int ccadiss1();
int ccadisst();
int ccaenter();
int ccaerod1();
int ccaerode();
int ccafilin();
int ccafili1();
int ccafili2();
void ccafil10();
int ccan8bf0();
int ccan8fol();
```

```
/*      ndap.h()          Block Diagram Software          ndap.h()
**
**      Programmer:       H. L. Kasdan
**
**      Copyright :       (C) 1988 by
**                        International Remote Imaging Systems Inc.
**
*************************************************************************
**
**      NAME
**                ndap.h - NIH data analysis header file
**
**      SYNOPSIS
**
**      DESCRIPTION
**
**      SEE ALSO
**
**      DIAGNOSTICS
**
**      BUGS
**
**      REVISION HISTORY
**
*************************************************************************
*/
/*
*************************************************************************
**                      DEFINES
*************************************************************************
```

```
*/
#define CAXMAX 50                       /* cell array x-maximum */
#define CAYMAX 50                       /* cell array y-maximum */
/*
****************************************************************************
**                    STRUCTURE DEFINITION
****************************************************************************
*/
struct imageO {
            char fname[14];                     /* cell data file name and extension */
         int cellno;                    /* cell number */
         long int riod;                 /* red iod */
         long int giod;                 /* green iod */
         long int biod;                 /* blue iod */
         long int riot;                 /* red iot */
         long int giot;                 /* green iot */
         long int biot;                 /* blue iot */
         int carea;                     /* cell area */
            int window;                 /* box window size */
         int thresh;                    /* edge trace threshold */
         char ctype[12];                /* cell type */
         int cxO;                       /* cell trace starting x-position */
         int cyO;                       /* cell trace starting y-position */
         int cxmin;                     /* cell trace minimum x-position */
         int cymin;                     /* cell trace minimum y-position */
         int cxmax;                     /* cell trace maximum x-position */
```

```
        int cymax;                  /* cell trace maximum y-position */
        int cpie;                   /* cell points in edge */
        char ccc[256];              /* cell chain code */
        unsigned char crimg[CAYMAX][CAXMAX];        /* cell red image */
        unsigned char cgimg[CAYMAX][CAXMAX];        /* cell green image */
        unsigned char cbimg[CAYMAX][CAXMAX];        /* cell blue image */
        unsigned char cnrimg[CAYMAX][CAXMAX];       /* cell normalized red image */
        unsigned char cngimg[CAYMAX][CAXMAX];       /* cell normalized green image */
        unsigned char cnbimg[CAYMAX][CAXMAX];       /* cell normalized blue image */
        int lgmlr[CAYMAX][CAXMAX];        /* log green minus log red image */
        int lbmlg[CAYMAX][CAXMAX];        /* log blue minus log green image */
        int lrmlb[CAYMAX][CAXMAX];        /* log red minus log blue image */
        int qclgmlr[50];   /* log green minus log red cell quantiles */
        int qclbmlg[50];   /* log blue minus log green cell quantiles */
        int qclrmlb[50];   /* log red minus log blue cell quantiles */
        int mclgmlr;       /* log green minus log red cell pixel mean */
        int mclbmlg;       /* log blue minus log green cell pixel mean */
        int mclrmlb;       /* log red minus log blue cell pixel mean */
        int sdclgmlr;      /* log green minus log red cell pixel std dev */
        int sdclbmlg;      /* log blue minus log green cell pixel std dev */
        int sdclrmlb;      /* log red minus log blue cell pixel std dev */
        char ccimg[CAYMAX][CAXMAX];       /* cell chain code image */
        struct ccdef cccimg[256];         /* cell chain code struct */
          };     /* image0 - cell image data structure */
struct hist0 {
          unsigned int rhist0[256];       /* red image histogram */
          unsigned int ghist0[256];       /* green image histogram */
          unsigned int bhist0[256];       /* blue image histogram */
          unsigned int rihist0[256];      /* red image interior histogram */
          unsigned int gihist0[256];      /* green image interior histogram */
          unsigned int bihist0[256];      /* blue image interior histogram */
          unsigned int rehist0[256];      /* red image exterior histogram */
          unsigned int gehist0[256];      /* green image exterior histogram */
          unsigned int behist0[256];      /* blue image exterior histogram */
          };             /* hist0 - cell image data histogram structure */
struct chist0 {
```

```
               unsigned int crhist0[256];      /* red image cum histogram */
               unsigned int cghist0[256];      /* green image cum histogram */
               unsigned int cbhist0[256];      /* blue image cum histogram */
               unsigned int crihist0[256];     /* red image interior cum histogram */
               unsigned int cgihist0[256];     /* green image interior cum histogram */
               unsigned int cbihist0[256];     /* blue image interior cum histogram */
               unsigned int crehist0[256];     /* red image exterior cum histogram */
                unsigned int cgehist0[256];    /* green image exterior cum histogram */
                unsigned int cbehist0[256];    /* blue image exterior cum histogram */
                };              /* chist0 - cell image data histogram structure */
struct qtl0 {
                unsigned int qr0[50];  /* red image quantiles */
                unsigned int qg0[50];  /* green image quantiles */
                unsigned int qb0[50];  /* blue image quantiles */
                unsigned int qri0[50]; /* red image interior quantiles */
                unsigned int qgi0[50]; /* green image interior quantiles */
                unsigned int qbi0[50]; /* blue image interior quantiles */
                unsigned int qre0[50]; /* red image exterior quantiles */
                unsigned int qge0[50]; /* green image exterior quantiles */
                unsigned int qbe0[50]; /* blue image exterior quantiles */
```

```
                unsigned int mrO;       /* red image mean */
                unsigned int mgO;       /* green image mean */
                unsigned int mbO;       /* blue image mean */
                unsigned int mriO;      /* red image interior mean */
                unsigned int mgiO;      /* green image interior mean */
                unsigned int mbiO;      /* blue image interior mean */
                unsigned int mreO;      /* red image exterior mean */
                unsigned int mgeO;      /* green image exterior mean */
                unsigned int mbeO;      /* blue image exterior mean */
                unsigned int sdrO;      /* red image std deviation */
                unsigned int sdgO;      /* green image std deviation */
                unsigned int sdbO;      /* blue image std deviation */
                unsigned int sdriO;     /* red image interior std deviation */
                unsigned int sdgiO;     /* green image interior std deviation */
                unsigned int sdbiO;     /* blue image interior std deviation */
                unsigned int sdreO;     /* red image exterior std deviation */
                unsigned int sdgeO;     /* green image exterior std deviation */
                unsigned int sdbeO;     /* blue image exterior std deviation */
                };              /* qtlO - cell image data quantile structure */
/*
****************************************************************
**                    SUBROUTINE DECLARATIONS
****************************************************************
*/
char *lbltbl();
```

```
/*      ndaphO()            Analysis Software                   ndaphO()
**
**      Programmer:         H. L. Kasdan
**
**      Copyright :         1988 by
**                          International Remote Imaging Systems, Inc.
**
*************************************************************************
**
**      NAME
**              ndaphO - image histogram routine
**
**      SYNOPSIS
**              #include <stdio.h>
**              #include <string.h>
**              #include "cca.h"
**              #include "ndap.h"
**              int ndaphO( int mode, struct imageO *p_imageO,
**                                  struct histO *p_histO, struct chistO *p_chistO )
**              int mode;                       histogram formation mode
**              struct imageO *p_imageO;        pointer to imageO data structure
**              struct histO *p_histO;          pointer to histO data structure
**              struct chistO *p_chistO;        pointer to chistO data structure
**
**      DESCRIPTION
**              Routine generates histogram to histO data structure
**              using data from imageO data structure according to
**              specified mode.
**              mode                operation
**              ----                ---------
**              0                   use un-normalized image data from imageO
**              1                   use normalized image data from imageO
**
**      SEE ALSO
```

```
**
**   DIAGNOSTICS
**
**   BUGS
**
**   REVISION HISTORY
**            900507  hlk      modified for *.URN file precursor to use window size
**
*****************************************************************************
*/
#include <stdio.h>
#include <string.h>
#include "cca.h"
#include "ndap.h"
#define FALSE 0
#define TRUE 1
#define PMAXX 50
#define PMAXY 50
#define PMINX 0
#define PMINY 0
int ndaph0( mode, p_image0, p_hist0, p_chist0 )
   int mode;                     /* histogram formation mode */
   struct image0 *p_image0;      /* pointer to image0 data structure */
   struct hist0 *p_hist0;        /* pointer to hist0 data structure */
   struct chist0 *p_chist0;      /* pointer to chist0 data structure */
   {
   /*
   *****************************************************************************
   **                  LOCAL STORAGE
   *****************************************************************************
```

```
*/
int err = 0;                                    /* error return */
int i;                          /* generic index */
int x;                          /* x-coordinate value */
int y;                          /* y-coordinate value */
/*
*************************************************************************
**          Do Histogram According to Mode
*************************************************************************
*/
switch (mode)
  {
  case 1:                       /* normalized image is data source */

    for ( i = 0; i < 256; i++)
   {
      p_hist0->rhist0[i] = 0;
      p_hist0->ghist0[i] = 0;
      p_hist0->bhist0[i] = 0;
      p_hist0->rihist0[i] = 0;
      p_hist0->gihist0[i] = 0;
      p_hist0->bihist0[i] = 0;
      p_hist0->rehist0[i] = 0;
      p_hist0->gehist0[i] = 0;
      p_hist0->behist0[i] = 0;
   } /* end - for ( i = 0; i < 256; i++) */

    for (y = PMINY; y < p_image0->window; y++)
   {
   for (x = PMINX; x < p_image0->window; x++)
     {
        p_hist0->rhist0[(int)p_image0->cnrimg[y][x]] += 1;
        p_hist0->ghist0[(int)p_image0->cngimg[y][x]] += 1;
         p_hist0->bhist0[(int)p_image0->cnbimg[y][x]] += 1;
     if ((p_image0->ccimg[y][x] == 'x') ||
         (((p_image0->ccimg[y][x] >= '0') && (p_image0->ccimg[y][x] < '8'))))
       {
       /*
```

```
    **          Do Interior
    */
        p_hist0->rihist0[(int)p_image0->cnrimg[y][x]] += 1;
        p_hist0->gihist0[(int)p_image0->cngimg[y][x]] += 1;
        p_hist0->bihist0[(int)p_image0->cnbimg[y][x]] += 1;
    } /* end - if (... || ...) */
  else
    {
    /*
    **          Do Exterior
    */
        p_hist0->rehist0[(int)p_image0->cnrimg[y][x]] += 1;
        p_hist0->gehist0[(int)p_image0->cngimg[y][x]] += 1;
        p_hist0->behist0[(int)p_image0->cnbimg[y][x]] += 1;
    } /* end - else (... || ...) */
     } /* end - for (x = PMINX; x < p_image0->window; x++) */
} /* end - for (y = PMINY; y < p_image0->window; y++) */

  /*
  **          Compute Cumulative Histogram
  */
  p_chist0->crhist0[0] = p_hist0->rhist0[0];
  p_chist0->cghist0[0] = p_hist0->ghist0[0];
  p_chist0->cbhist0[0] = p_hist0->bhist0[0];
  p_chist0->crihist0[0] = p_hist0->rihist0[0];
  p_chist0->cgihist0[0] = p_hist0->gihist0[0];
  p_chist0->cbihist0[0] = p_hist0->bihist0[0];
  p_chist0->crehist0[0] = p_hist0->rehist0[0];
  p_chist0->cgehist0[0] = p_hist0->gehist0[0];
  p_chist0->cbehist0[0] = p_hist0->behist0[0];

  for ( i = 1; i < 256; i++)
 {
```

```
      p_chist0->crhist0[i] = p_hist0->rhist0[i] + p_chist0->crhist0[i - 1];
      p_chist0->cghist0[i] = p_hist0->ghist0[i] + p_chist0->cghist0[i - 1];
      p_chist0->cbhist0[i] = p_hist0->bhist0[i] + p_chist0->cbhist0[i - 1];
      p_chist0->crihist0[i] = p_hist0->rihist0[i] + p_chist0->crihist0[i - 1];
      p_chist0->cgihist0[i] = p_hist0->gihist0[i] + p_chist0->cgihist0[i - 1];
      p_chist0->cbihist0[i] = p_hist0->bihist0[i] + p_chist0->cbihist0[i - 1];
      p_chist0->crehist0[i] = p_hist0->rehist0[i] + p_chist0->crehist0[i - 1];
      p_chist0->cgehist0[i] = p_hist0->gehist0[i] + p_chist0->cgehist0[i - 1];
      p_chist0->cbehist0[i] = p_hist0->behist0[i] + p_chist0->cbehist0[i - 1];
  } /* end - for ( i = 0; i < 256; i++) */
    break;

  case 0: default:                    /* use un-normalized image data */

    for ( i = 0; i < 256; i++)
  {
      p_hist0->rhist0[i] = 0;
      p_hist0->ghist0[i] = 0;
      p_hist0->bhist0[i] = 0;
      p_hist0->rihist0[i] = 0;
      p_hist0->gihist0[i] = 0;
      p_hist0->bihist0[i] = 0;
      p_hist0->rehist0[i] = 0;
      p_hist0->gehist0[i] = 0;
      p_hist0->behist0[i] = 0;
  } /* end - for ( i = 0; i < 256; i++) */

    for (y = PMINY; y < p_image0->window; y++)
  {
  for (x = PMINX; x < p_image0->window; x++)
    {
```

```
            p_hist0->rhist0[(int)p_image0->crimg[y][x]] += 1;
            p_hist0->ghist0[(int)p_image0->cgimg[y][x]] += 1;
            p_hist0->bhist0[(int)p_image0->cbimg[y][x]] += 1;
        if ((p_image0->ccimg[y][x] == 'x') ||
            (((p_image0->ccimg[y][x] >= '0') && (p_image0->ccimg[y][x] < '8'))))
          {
          /*
          **            Do Interior
          */
              p_hist0->rihist0[(int)p_image0->crimg[y][x]] += 1;
              p_hist0->gihist0[(int)p_image0->cgimg[y][x]] += 1;
              p_hist0->bihist0[(int)p_image0->cbimg[y][x]] += 1;
          } /* end - if (... || ...) */
        else
          {
          /*
          **            Do Exterior
          */
              p_hist0->rehist0[(int)p_image0->crimg[y][x]] += 1;
              p_hist0->gehist0[(int)p_image0->cgimg[y][x]] += 1;
              p_hist0->behist0[(int)p_image0->cbimg[y][x]] += 1;
          } /* end - else (... || ...) */
            } /* end - for (x = PMINX; x < p_image0->window; x++) */
      } /* end - for (y = PMINY; y < p_image0->window; y++) */

        /*
        **              Compute Cumulative Histogram
        */
        p_chist0->crhist0[0] = p_hist0->rhist0[0];
        p_chist0->cghist0[0] = p_hist0->ghist0[0];
        p_chist0->cbhist0[0] = p_hist0->bhist0[0];
        p_chist0->crihist0[0] = p_hist0->rihist0[0];
        p_chist0->cgihist0[0] = p_hist0->gihist0[0];
        p_chist0->cbihist0[0] = p_hist0->bihist0[0];
        p_chist0->crehist0[0] = p_hist0->rehist0[0];
        p_chist0->cgehist0[0] = p_hist0->gehist0[0];
        p_chist0->cbehist0[0] = p_hist0->behist0[0];
```

```
      for ( i = 1; i < 256; i++)
    {
        p_chist0->crhist0[i] = p_hist0->rhist0[i] + p_chist0->crhist0[i - 1];
        p_chist0->cghist0[i] = p_hist0->ghist0[i] + p_chist0->cghist0[i - 1];
        p_chist0->cbhist0[i] = p_hist0->bhist0[i] + p_chist0->cbhist0[i - 1];
        p_chist0->crihist0[i] = p_hist0->rihist0[i] + p_chist0->crihist0[i - 1];
        p_chist0->cgihist0[i] = p_hist0->gihist0[i] + p_chist0->cgihist0[i - 1];
        p_chist0->cbihist0[i] = p_hist0->bihist0[i] + p_chist0->cbihist0[i - 1];
        p_chist0->crehist0[i] = p_hist0->rehist0[i] + p_chist0->crehist0[i - 1];
        p_chist0->cgehist0[i] = p_hist0->gehist0[i] + p_chist0->cgehist0[i - 1];
        p_chist0->cbehist0[i] = p_hist0->behist0[i] + p_chist0->cbehist0[i - 1];
    } /* end - for ( i = 0; i < 256; i++) */
      break;
    } /* end - switch (mode) */

  return(err);

  } /* end - int ndaph0(char *cs) */
```

```
/*    ndapctq0()        Analysis Software               ndapctq0()
**
**    Programmer:       H. L. Kasdan
**
**    Copyright :       1988 by
**                      International Remote Imaging Systems, Inc.
**
*************************************************************************
**
**    NAME
**            ndapctq0 - produce cum hist quantiles and statistics routine
**
**    SYNOPSIS
**            #include <stdio.h>
**            #include <string.h>
**            #include "cca.h"
**            #include "ndap.h"
**            int ndapctq0( struct chist0 *p_chist0, struct qt10 *p_qt10 )
**            struct chist0 *p_chist0;        pointer to chist0 data structure
**            struct qt10 *p_qt10;     pointer to qt10 data structure
**
**    DESCRIPTION
**            Routine generates quantiles of cell
**            image data using data from chist0 data structure.
**
**    SEE ALSO
**
**    DIAGNOSTICS
**
**    BUGS
**
**    REVISION HISTORY
**
*************************************************************************
```

```
*/
#include <math.h>
#include <stdio.h>
#include <string.h>
#include "cca.h"
#include "ndap.h"
#define BNPCT 95
#define BOFF 28
#define GNPCT 95
#define GOFF 21
#define FALSE 0
#define NVALUE 200                          /* normalized maximum image value */
#define PMAXX 46
#define PMAXY 46
#define PMINX 2
#define PMINY 2
#define RNPCT 95
#define ROFF 25
#define TRUE 1

void ndapqnti();
void ndapmsdi();

int ndapctq0( p_chist0, p_qt10 )
  struct chist0 *p_chist0;   /* pointer to chist0 data structure */
  struct qt10 *p_qt10;       /* pointer to qt10 data structure */
  {

  /*
  ***************************************************************************
  **                  LOCAL STORAGE
  ***************************************************************************
  */
  int err = 0;                            /* error return */
  int i;                     /* generic index */
  int n;                     /* number of interior pixels - length of index
                                array */
  /*
  ***************************************************************************
  **                  Compute Statistics for 9 images
  ***************************************************************************
```

```
*/
/*
**                      r Cell Image
*/
ndapqnti(&p_chist0->crhist0[0], &p_qt10->qr0[0]);
ndapmsdi(&p_chist0->crhist0[0],
        &p_qt10->mr0, &p_qt10->sdr0);

/*
**                      g Cell Image
*/
ndapqnti(&p_chist0->cghist0[0], &p_qt10->qg0[0]);
ndapmsdi(&p_chist0->cghist0[0],
        &p_qt10->mg0, &p_qt10->sdg0);

/*
**                      b Cell Image
*/
ndapqnti(&p_chist0->cbhist0[0], &p_qt10->qb0[0]);
ndapmsdi(&p_chist0->cbhist0[0],
        &p_qt10->mb0, &p_qt10->sdb0);

/*
**                      ri Cell Image
*/
ndapqnti(&p_chist0->crihist0[0], &p_qt10->qri0[0]);
ndapmsdi(&p_chist0->crihist0[0],
        &p_qt10->mri0, &p_qt10->sdri0);

/*
**                              gi Cell Image
*/
ndapqnti(&p_chist0->cgihist0[0], &p_qt10->qgi0[0]);
ndapmsdi(&p_chist0->cgihist0[0],
        &p_qt10->mgi0, &p_qt10->sdgi0);
/*
```

```
**                              bi Cell Image
*/
ndapqnti(&p_chist0->cbihist0[0], &p_qt10->qbi0[0]);
ndapmsdi(&p_chist0->cbihist0[0],
      &p_qt10->mbi0, &p_qt10->sdbi0);

/*
**                              re Cell Image
*/
ndapqnti(&p_chist0->crehist0[0], &p_qt10->qre0[0]);
ndapmsdi(&p_chist0->crehist0[0],
      &p_qt10->mre0, &p_qt10->sdre0);

/*
**                              ge Cell Image
*/
ndapqnti(&p_chist0->cgehist0[0], &p_qt10->qge0[0]);
ndapmsdi(&p_chist0->cgehist0[0],
      &p_qt10->mge0, &p_qt10->sdge0);

/*
**                              be Cell Image
*/
ndapqnti(&p_chist0->cbehist0[0], &p_qt10->qbe0[0]);
ndapmsdi(&p_chist0->cbehist0[0],
      &p_qt10->mbe0, &p_qt10->sdbe0);


return(err);

} /* end - int ndapctq0(char *cs) */

/*  ndapqnti()       Analysis Software                  ndapqnti()
**
**  Programmer:      H. L. Kasdan
**
```

```
**  Copyright :      1988 by
**          International Remote Imaging Systems, Inc.
**
*****************************************************************************
**
**  NAME
**          ndapqnt1 - compute quantiles from cumulative histogram
**
**  SYNOPSIS
**          #include <stdio.h>
**          #include <string.h>
**          #include "cca.h"
**          #include "ndap.h"
**          void ndapqnt1( unsigned int chist[256] int quant[50])
**               unsigned int chist[256];        cumulative histogram array
**          int quant[50];                   arrray in which 50 quantiles
**                                           will be returned
**
**  DESCRIPTION
**          Routine orders determines quantiles from cumulative histogram
**               array.
**
**  SEE ALSO
**
**  DIAGNOSTICS
**
**  BUGS
**
**  REVISION HISTORY
**
*****************************************************************************
*/
void ndapqnt1( chist, quant)
  unsigned int chist[256];        /* cumulative histogram array */
  int quant[50];              /* arrray in which 50 quantiles
                                 will be returned */
```

```
{
/*
**********************************************************************
**                          LOCAL STORAGE
**********************************************************************
*/
long int cj;                    /* target cum hist value for next quantile */
int i;
int j0 = 0;                         /* quantile index */
long int 150 = 50;              /* number of quantiles */
long int maxnum = chist[255];       /* total number of values in histgram */
int n = 256;                    /* size of histogram array */
/*
**                          Generate Quantiles
*/
cj = (((long int)(j0 + 1)) * maxnum) / 150;
for (i = 1; (i <= n) && (j0 < 50); i++)
  {
  while (((unsigned int)cj <= chist[i]) && (j0 < 50))
    {
    quant[j0++] = i;
    cj = (((long int)(j0 + 1)) * maxnum) / 150;
    } /* end - while ((jnext - j0++) > 1) */

  } /* end - for (i = 1; i <= n; i++) */
} /* end - void ndapqntl( int n, int arrin[], int indx[]) */


/*  ndapmsdl()      Analysis Software                   ndapmsdl()
**
**  Programmer:     H. L. Kasdan
**
**  Copyright :     1988 by
**          International Remote Imaging Systems, Inc.
**
*********************************************************************
**
**  NAME
```

```
**          ndapmsdl - compute mean and std deviation
**
**  SYNOPSIS
**          #include <math.h>
**          #include <stdio.h>
**          #include <string.h>
**          #include "cca.h"
**          #include "ndap.h"
**          void ndapmsdl( unsigned int chist[256], int *mean,
**                             int *sdev)
**              unsigned int chist[256];        cum histogram array
**          int *mean;                         mean value of specified
**                                             elements
**          int *sdev;                         standard deviation of
**                                             specified elements
**
**  DESCRIPTION
**          Routine computes mean and standard deviation of
**          image from cumulative histogram.
**
**  SEE ALSO
**
**  DIAGNOSTICS
**
**  BUGS
**
**  REVISION HISTORY
**
*****************************************************************************
*/
void ndapmsdl( chist, mean, sdev)
  unsigned int chist[256];       /* cum histogram array */
  int *mean;                          /* mean value of specified
                                 elements */
  int *sdev;                          /* standard deviation of
                                 specified elements */
```

```
{
/*
************************************************************************
**                              LOCAL STORAGE
************************************************************************
*/
int chvalue;                          /* current histogram value */
double di;                            /* double index */
double dmean;                         /* running first moment */
int dopixel;                          /* previous temp pixel value */
double dpixel;                 /* temp pixel value */
double dnt = (double)chist[255];      /* total number of points in histogram */
double dsdev;                         /* double std dev */
double dsm;                           /* running second moment */
int i;
int n = 256;                      /* histogram array size */
/*
**                              Compute Moments
```

```
*/
dmean = 0.;
dsm   = 0.;
dopixel = 0.;
for (i = 0; (i < n); i++)
  {
  chvalue = (int)chist[i];
  dpixel = (double)(chvalue - dopixel);
  if (dpixel != 0. )
    {
    di = (double)i;
    dmean += dpixel * di;
    dsm += dpixel * di * di;
    } /* end - if (dpixel != 0. ) */
  dopixel = chvalue;
  } /* end - for (i = 1; i <= n; i++) */
dmean /= dnt;
*mean = (int)dmean;
dsm /= dnt;
dsdev = dsm - (dmean * dmean);
dsdev = (dsdev >= (double)0. ) ? dsdev : (double)0.;
dsdev = sqrt(dsdev);
*sdev = (int)dsdev;
} /* end - void ndapmsd1( int n, int arrin[], int indx[]) */
```

```
/*      ndapmlqO()          Analysis Software                   ndapmlqO()
**
**      Programmer:         H. L. Kasdan
**
**      Copyright :         1988 by
**                          International Remote Imaging Systems, Inc.
**
*************************************************************************
**
**      NAME
**              ndapmlqO - produce log image quantiles and statistics routine
**
**      SYNOPSIS
**              #include <stdio.h>
**              #include <string.h>
**              #include "cca.h"
**              #include "ndap.h"
**              int ndapmlqO( struct imageO *p_imageO )
**              struct imageO *p_imageO;        pointer to imageO data structure
**
**      DESCRIPTION
**              Routine generates quantiles of log difference cell
**              image data using data from imageO data structure.
**              Modified heapsort algorithm is used to order pixel data.
**
**      SEE ALSO
**
**      DIAGNOSTICS
**
**      BUGS
**
**      REVISION HISTORY
**              900507  hlk     modified for *.URN precursor to use window value
**
*************************************************************************
```

```
*/
#include <math.h>
#include <stdio.h>
#include <string.h>
#include "cca.h"
#include "ndap.h"
#define BNPCT 95
#define BOFF 28
#define GNPCT 95
#define GOFF 21
#define FALSE 0
#define NVALUE 200                    /* normalized maximum image value */
#define PMAXX 46
#define PMAXY 46
#define PMINX 2
#define PMINY 2
#define RNPCT 95
#define ROFF 25
#define TRUE 1

void indexx();
void ndapqntO();
void ndapmsdO();

int ndapmlqO( p_imageO )
  struct imageO *p_imageO;    /* pointer to imageO data structure */
  {
  /*
  ************************************************************************
  **                     GLOBAL STORAGE
  ************************************************************************
  */
  extern int ltabO[256];      /* precomputed log table values */

  /*
  ************************************************************************
  **                     LOCAL STORAGE
  ************************************************************************
  */
  int bnvalue;                          /* blue pixel normalizing value */
```

```
int err = 0;                         /* error return */
int i;                          /* generic index */
int indx[CAXMAX * CAYMAX + 1];       /* interior pixel index array */
int gnvalue;                         /* green pixel normalizing value */
long int lptmp;                 /* temporary pixel value */
int n;                          /* number of interior pixels - length of index
                                   array */
int rnvalue;                         /* red pixel normalizing value */
int x;                          /* x-coordinate value */
int y;                          /* y-coordinate value */
/*
**************************************************************************
**                    Generate Index Array
**************************************************************************
*/
n = 0;                          /* index count */

for ( y = 0; y < p_image0->window; y++ )
  {
  for ( x = 0; x < p_image0->window; x++ )
    {
    if ((p_image0->ccimg[y][x] == 'x') ||
      (((p_image0->ccimg[y][x] >= '0') && (p_image0->ccimg[y][x] < '8'))))
   {
    indx[++n] = (y * CAXMAX) + x;
   } /* end - if (... || (...) && (...) ) */

     } /* end - for ( x = 0; x < p_image0->window; x++ ) */
   } /* end - for ( y = 0; y < p_image0->window; y++ ) */
 /*
 **                       lgmlr Cell Image
 */
 indexx(n, &p_image0->lgmlr[0][0], indx);  /* order index array for lgmlr */
 ndapqnt0(n, &p_image0->lgmlr[0][0], indx, &p_image0->qclgmlr[0]);
 ndapmsd0(n, &p_image0->lgmlr[0][0], indx,
      &p_image0->mclgmlr, &p_image0->sdclgmlr);
```

```
/*
**                          lbmlg Cell Image
*/
indexx(n, &p_image0->lbmlg[0][0], indx);  /* order index array for lbmlg */
ndapqnt0(n, &p_image0->lbmlg[0][0], indx, &p_image0->qclbmlg[0]);
ndapmsd0(n, &p_image0->lbmlg[0][0], indx,
     &p_image0->mclbmlg, &p_image0->sdclbmlg);

/*
**                          lrmlb Cell Image
*/
indexx(n, &p_image0->lrmlb[0][0], indx);  /* order index array for lrmlb */
ndapqnt0(n, &p_image0->lrmlb[0][0], indx, &p_image0->qclrmlb[0]);
ndapmsd0(n, &p_image0->lrmlb[0][0], indx,
     &p_image0->mclrmlb, &p_image0->sdclrmlb);


return(err);

} /* end - int ndapmlq0(char *cs) */

/*  indexx()          Analysis Software                   indexx()
**
**  Programmer:       H. L. Kasdan
**
**  Copyright :       1988 by
**            International Remote Imaging Systems, Inc.
**
*************************************************************************
**
**  NAME
**            indexx - order index array for log image data
**
**  SYNOPSIS
**            #include <stdio.h>
**            #include <string.h>
**            #include "cca.h"
**            #include "ndap.h"
```

```
**          void indexx( int n, int arrin[], int indx[])
**          int n;                              number of elements to be ordered
**          int arrin[];                        array containing elements to
**                                              be ordered
**          int indx[];                         array containing indecies of
**                                              elements in arrin that will
**                                              be ordered. This index array
**                                              will be ordered in place.
**
**   DESCRIPTION
**          Routine orders elements in array arrin specified in
**          index array, indx.
**
**   SEE ALSO
**
**   DIAGNOSTICS
**
**   BUGS
**
**   REVISION HISTORY
**
**************************************************************************
*/
void indexx( n, arrin, indx)
  int n;                      /* number of elements to be ordered */
  int arrin[];                        /* array containing elements to
                                  be ordered */
  int indx[];                         /* array containing indecies of
                                  elements in arrin that will
                                  be ordered. This index array
                                  will be ordered in place.  */

  {
  /*
  ****************************************************************************
  **                            LOCAL STORAGE
  ****************************************************************************
```

```
*/
int i;
int indxt;
int ir;
int j;
int l;
int q;
/*
**                              Do heapsort
*/
l = (n >> 1) + 1;
ir = n;

for (;;)
  {

  if (l > 1)
    q = arrin[(indxt = indx[--l])];
  else
    {
    q = arrin[(indxt = indx[ir])];
    indx[ir] = indx[1];
    if (--ir == 1)
  {
  indx[1] = indxt;
  return;
      } /* end - if (--ir == 1) */
    } /* end - else (l > 1) */

  i = l;
  j = l << 1;

  while (j <= ir)
    {
    if (j < ir && arrin[indx[j]] < arrin[indx[j + 1]]) j++;
    if (q < arrin[indx[j]])
  {
```

```
    indx[i] = indx[j];
        j += (i=j);
        } /* end - if (q < arrin[indx[j]]) */
      else j = ir + 1;
      } /* end - while (j <= ir) */

    indx[i] = indxt;

    } /* end - for (;;) */

  } /* end - void indexx( int n, int arrin[], int indx[]) */

/*  ndapqnt0()       Analysis Software                 ndapqnt0()
**
**  Programmer:      H. L. Kasdan
**
**  Copyright :      1988 by
**            International Remote Imaging Systems, Inc.
**
*************************************************************************
**
**  NAME
**            ndapqnt0 - order index array for log image data
**
**  SYNOPSIS
**            #include <stdio.h>
**            #include <string.h>
**            #include "cca.h"
**            #include "ndap.h"
**            void ndapqnt0( int n, int arrin[], int indx[], int quant[50])
**            int n;                          number of elements to be ordered
**            int arrin[];                    array containing elements to
**                                            be ordered
**            int indx[];                     array containing indecies of
**                                            elements in arrin that will
**                                            be ordered. This index array
**                                            will be ordered in place.
**            int quant[50];                  arrray in which 50 quantiles
**                                            will be returned
**
```

```
**  DESCRIPTION
**          Routine orders determines quantiles from elements in array
**          arrin according to index array, indx.
**
**  SEE ALSO
**
**  DIAGNOSTICS
**
**  BUGS
**
**  REVISION HISTORY
**
*****************************************************************************
*/
void ndapqnt0( n, arrin, indx, quant)
  int n;                    /* number of elements to be ordered */
  int arrin[];                     /* array containing elements to
                             be ordered */
  int indx[];                      /* array containing indecies of
                             elements in arrin that will
                             be ordered. This index array
                             will be ordered in place. */
  int quant[50];            /* arrray in which 50 quantiles
                             will be returned */


  {
  /*
  *****************************************************************************
  **                         LOCAL STORAGE
  *****************************************************************************
  */
  int i;
  int j0 = 0;                       /* quantile index */
  int jnext;                        /* target quantile index */
  /*
  **                         Generate Quantiles
  */
```

```
  quant[0] = arrin[indx[1]];
  j0 = 1;
  for (i = 1; (i <= n) && (j0 < 50); i++)
    {
    jnext = (i * 50) / n;
    while (((jnext - j0) > 0) && (j0 < 50))
      {
      quant[j0++] = arrin[indx[i]];
      } /* end - while ((jnext - j0++) > 1) */

    } /* end - for (i = 1; i <= n; i++) */
 } /* end - void ndapqnt0( int n; int arrin[], int indx[]) */


/*  ndapmsd0()      Analysis Software               ndapmsd0()
**
**  Programmer:     H. L. Kasdan
**
**  Copyright :      1988 by
**          International Remote Imaging Systems, Inc.
**
*****************************************************************************
**
**  NAME
**          ndapmsd0 - order index array for log image data
**
**  SYNOPSIS
**          #include <math.h>
**          #include <stdio.h>
**          #include <string.h>
**          #include "cca.h"
**          #include "ndap.h"
**          void ndapmsd0( int n, int arrin[], int indx[], int *mean,
                           int *sdev)
```

```
**          int n;                          number of elements to be ordered
**          int arrin[];                    array containing elements to
**                                          be ordered
**          int indx[];                     array containing indecies of
**                                          elements in arrin that will
**                                          be used.
**          int *mean;                      mean value of specified
**                                          elements
**          int *sdev;                      standard deviation of
**                                          specified elements
**
**  DESCRIPTION
**          Routine orders computes mean and standard deviation of
**          elements in array arrin specified in index array, indx.
**
**  SEE ALSO
**
**  DIAGNOSTICS
**
**  BUGS
**
**  REVISION HISTORY
**
*****************************************************************************
*/
void ndapmsdO( n, arrin, indx, mean, sdev)
  int n;                  /* number of elements to be ordered */
  int arrin[];                    /* array containing elements to
                          be ordered */
  int indx[];                     /* array containing indecies of
                          elements in arrin that will
                          be ordered. This index array
                          will be ordered in place. */
  int *mean;                      /* mean value of specified
                          elements */
  int *sdev;                      /* standard deviation of
                          specified elements */


  {
  /*
```

```
***************************************************************************
**                              LOCAL STORAGE
***************************************************************************
*/
double dmean;                           /* running first moment */
double dpixel;                  /* temp pixel value */
double dsdev;                           /* double std dev */
double dsm;                             /* running second moment */
int i;
/*
**                              Compute Moments
*/
dmean = 0.;
dsm   = 0.;
for (i = 1; (i <= n); i++)
  {
  dpixel = (double)arrin[indx[i]];
  dmean += dpixel;
  dsm += dpixel * dpixel;
  } /* end - for (i = 1; i <= n; i++) */
dmean /= (double)n;
*mean = (int)dmean;
dsm /= (double)n;
dsdev = dsm - (dmean * dmean);
dsdev = (dsdev >= (double)0.) ? dsdev : (double)0.;
dsdev = sqrt(dsdev);
*sdev = (int)dsdev;
} /* end - void ndapmsd0( int n, int arrin[], int indx[]) */
```

WHAT IS CLAIMED IS:

1. A method of identifying an object under examination, said object has a discernible boundary; said method comprising the steps of:

a) forming an image of said object;

b) segmenting said image to form a plurality of pixels;

c) detecting the boundary of said object under examination in said image;

d) measuring a parameter of each pixel within the boundary detected;

e) forming an ordered sequence of the parameter of the pixel measured, within the boundary detected;

$$P_1 \ldots P_k \ldots P_N$$

P_1 - first value of the parameter of the pixel within the boundary detected;

P_N - last value of the parameter of the pixel within the boundary deducted;

N - total number of pixels within the boundary detected;

k - index to the N pixels;

f) determining the P_k data point(s) such that k satisfies the following relationship; and

$$\frac{i-1}{M} < \frac{k}{N} \leq \frac{i}{M}$$

M - total number of quantile partitions;

i - i^{th} fraction of partition or i^{th} quantile partitions;

g) identifying said object under examination, based upon the relationship developed from step (f) associated with said object.

2. The method of Claim 1 wherein P_1 is the smallest value and P_N is the largest value.

3. The method of Claim 1, wherein said identifying step further comprising:

comparing said relationship of step (f) associated with said object under examination, to a table of pre-determined relationships of other identified objects to identify the object under examination.

4. The method of Claim 3, wherein said table of pre-determined relationships of other identified objects comprises a range of values.

5. The method of Claim 3, wherein said table of predetermined relationships of other identified object comprises a single value.

6. The method of Claim 4, wherein said table of pre-determined relationships are based upon experimental results.

7. The method of Claim 6, wherein said table of predetermined relationships of other identified object comprises a single value.

8. The method of Claim 1, wherein said parameter is intensity of visible light.

9. The method of Claim 1, wherein said parameter is a function of color representations at the same position.

10. The method of Claim 9, wherein said color representation is based upon three primary colors.

11. The method of Claim 9, wherein said color representation is based upon hue, intensity, and saturation of colors.

12. The method of Claim 10, wherein said parameter is

$$\log (a) - \log (b)$$

where a and b are the intensities of two different colors.

13. The method of Claim 12 wherein a is red color and b is blue color.

14. The method of Claim 12, wherein a is blue color and b is green color.

15. The method of Claim 12, wherein a is green color and b is red color.

16. A method of differentiating a plurality of different objects under examination, wherein each object has a discernible boundary; said method comprising the steps of:

a) forming a plurality of images, with each image of each of said plurality of objects;

b) segmenting each of said plurality of images to form a plurality of pixels;

c) detecting the boundary of each object under examination in each of said images;

d) measuring a parameter of each pixel within the boundary of each object detected;

e) forming an ordered sequence of the parameter of the pixel measured, within the boundary detected for each object;

$P_1 \ldots P_{k1} \ldots P_{N1}$ - object 1 having N_1 pixels

$P_1 \ldots P_{k2} \ldots P_{N2}$ - object 2 having N_2 pixels

f) determining, for each ordered sequence formed, the P_k data point(s) such that k satisfies the following; and

$$\frac{i-1}{M} < \frac{k}{\text{Total No. of pixels for the object}} \leq \frac{i}{M}$$

M - total number of quantile partitions;
i - i^{th} fraction of partition or i^{th} quantile partition;

g) comparing the P_k data point(s) of each ordered sequence for each object to one another, for identical variables of i and M, to differentiate the objects.

17. The method of Claim 16, wherein said parameter is intensity of visible light.

18. The method of Claim 16, wherein said parameter is a function of color representations at the same position.

19. The method of Claim 18, wherein said color representation is based upon three primary colors.

20. The method of Claim 18, wherein said color representation is based upon hue, intensity, and saturation of colors.

21. The method of Claim 19, wherein said parameter is

$$\log(a) - \log(b)$$

where a and b are the intensities of two different colors.

22. The method of Claim 21 wherein a is red color and b is blue color.

23. The method of Claim 21, wherein a is blue color and b is green color.

24. The method of Claim 21, wherein a is green color and b is red color.

25. An apparatus for identifying an object under examination, said object has a discernible boundary; said apparatus comprising:

a) means for forming an image of said object;

b) means for segmenting said image to form a plurality of pixels;

c) means for detecting the boundary of said image of said object under examination;

d) means for measuring a parameter of each pixel within the boundary detected;

e) means for forming an ordered sequence of the parameter of the pixel measured, within the boundary detected;

$$P_1 \ldots P_k \ldots P_N$$

P_1 - first value of the parameter of the pixel within the boundary detected;

P_N - last value of the parameter of the pixel within the boundary deducted;

N - total number of pixels within the boundary detected;

k - index to the N pixels;

f) means for determining the P_k data point(s) such that k satisfies the following relationship; and

$$\frac{i-1}{M} < \frac{k}{N} \leq \frac{i}{M}$$

M - total number of quantile partitions;
i - i^{th} fraction of partition or i^{th} quantile partition;

g) means for identifying said object under examination, based upon the relationship of P_k to i^{th} quantile in M partitions developed from step (f) associated with said object.

26. The apparatus of Claim 25 wherein P_1 is the smallest value and P_N is the largest value.

27. The apparatus of Claim 25, wherein said means for identifying further comprising:

means for comparing said relationship of step (f) associated with said object under examination, to a table of pre-determined relationships of other identified objects to identify the object under examination.

28. The apparatus of Claim 27, wherein said table of pre-determined relationships of other identified objects comprises a range of values.

29. The apparatus of Claim 26, wherein said parameter is intensity of visible light.

30. The apparatus of Claim 29, wherein said parameter is a difference of color representations at the same position.

31. The apparatus of Claim 30, wherein said color representation is based upon three primary colors.

32. The apparatus of Claim 30, wherein said color representation is based upon hue, intensity, and saturation of colors.

33. The apparatus of Claim 31, wherein said parameter is

$$\log (a) - \log (b)$$

where a and b are the intensities of two different primary colors.

34. The apparatus of Claim 33, wherein a is red color and b is blue color.

35. The apparatus of Claim 33, wherein a is blue color and b is green color.

36. The apparatus of Claim 33, wherein a is green color and b is red color.

37. An apparatus for differentiating a plurality of objects under examination, each of said objects has a discernible boundary; said apparatus comprising:

a) means for forming an image of each of said plurality of objects to form a plurality of images;
b) means for segmenting each of said plurality of images to form a plurality of pixels;
c) means for detecting the boundary of each of said plurality of objects under examination in each of said plurality of images;
d) means for measuring a parameter of each pixel within the boundary of each object detected;
e) means for forming an ordered sequence of the parameter of the pixel measured, within the boundary detected for each object;

$P_1 \ldots P_{k1} \ldots P_{N1}$ - object 1 having N_1 pixels
$P_1 \ldots P_{k2} \ldots P_{N2}$ - object 2 having N_2 pixels

f) means for determining, for each ordered sequence formed, the P_k data point(s) such that k satisfies the following; and

$$\frac{i-1}{M} < \frac{k}{\text{Total No. of pixels for the object}} \leq \frac{i}{M}$$

M - total number of quantile partitions;
i - i^{th} fraction of partition or i^{th} quantile partition;

g) means for comparing the P_k data point(s) of each ordered sequence for each object to one another, for identical variables of i and M, to differentiate the objects.

38. The apparatus of Claim 37 wherein P_1 is the smallest value and P_N is the largest value.

39. The apparatus of Claim 37, wherein said means for identifying further comprising:

means for comparing said relationship of step (f) associated with said object under examination, to a table of pre-determined relationships of other identified objects to identify the object under examination.

40. The apparatus of Claim 39, wherein said table of pre-determined relationships of other identified objects comprises a range of values.

41. The apparatus of Claim 38, wherein said parameter is intensity of visible light.

42. The apparatus of Claim 41, wherein said parameter is a difference of color representations at the same position.

43. The apparatus of Claim 42, wherein said color representation is based upon three primary colors.

44. The apparatus of Claim 42, wherein said color representation is based upon hue, intensity, and saturation of colors.

45. The apparatus of Claim 43, wherein said parameter is

$$\log (a) - \log (b)$$

where a and b are the intensities of two different primary colors.

46. The apparatus of Claim 45, wherein a is red color and b is blue color.

47. The apparatus of Claim 45, wherein a is blue color and b is green color.

48. The apparatus of Claim 45, wherein a is green color and b is red color.

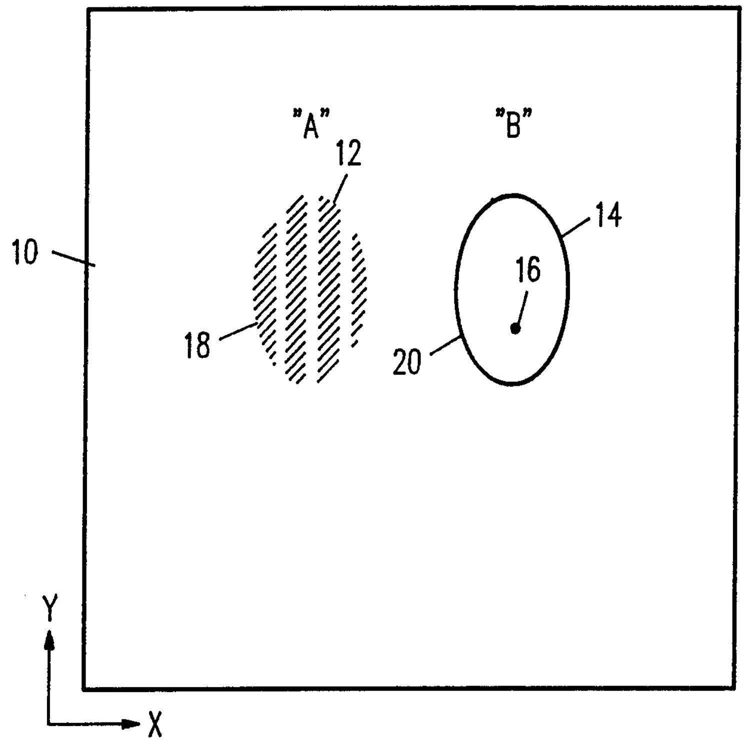


FIG. 1

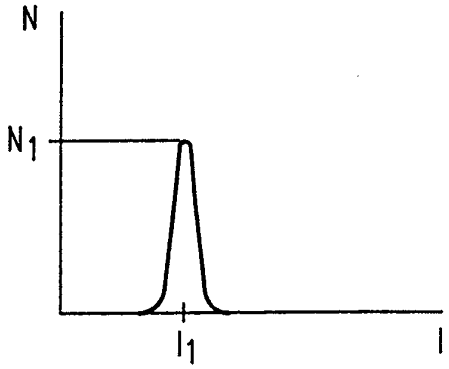


FIG. 2a

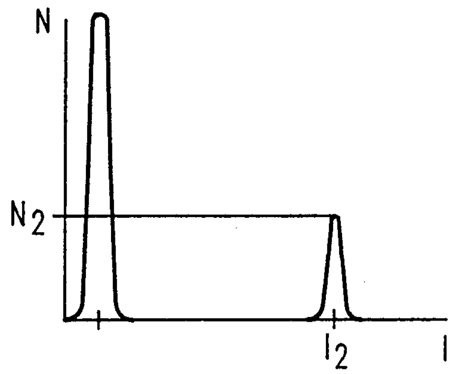


FIG. 2b

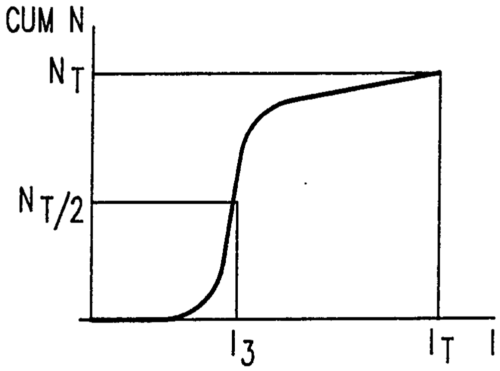


FIG. 3a

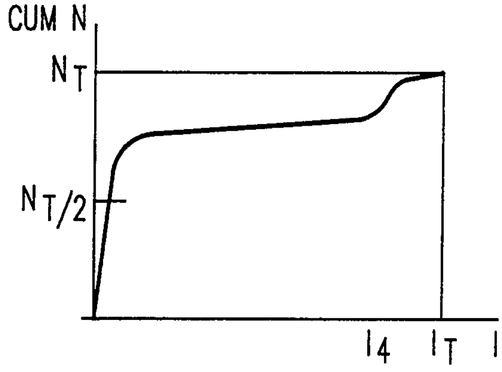


FIG. 3b

50th QUANTILE	INTENSITY	OBJECT
	I_a	xxx
	⋮	⋮
	I_3	12
	⋮	⋮
	I_4	14
	⋮	⋮
	I_x	xxx

FIG. 4

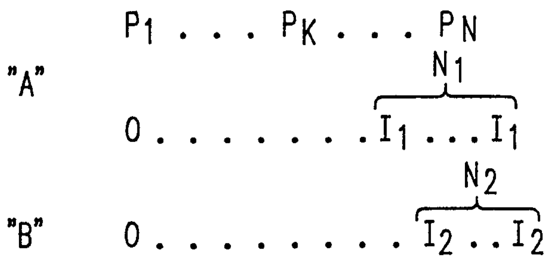


FIG. 5

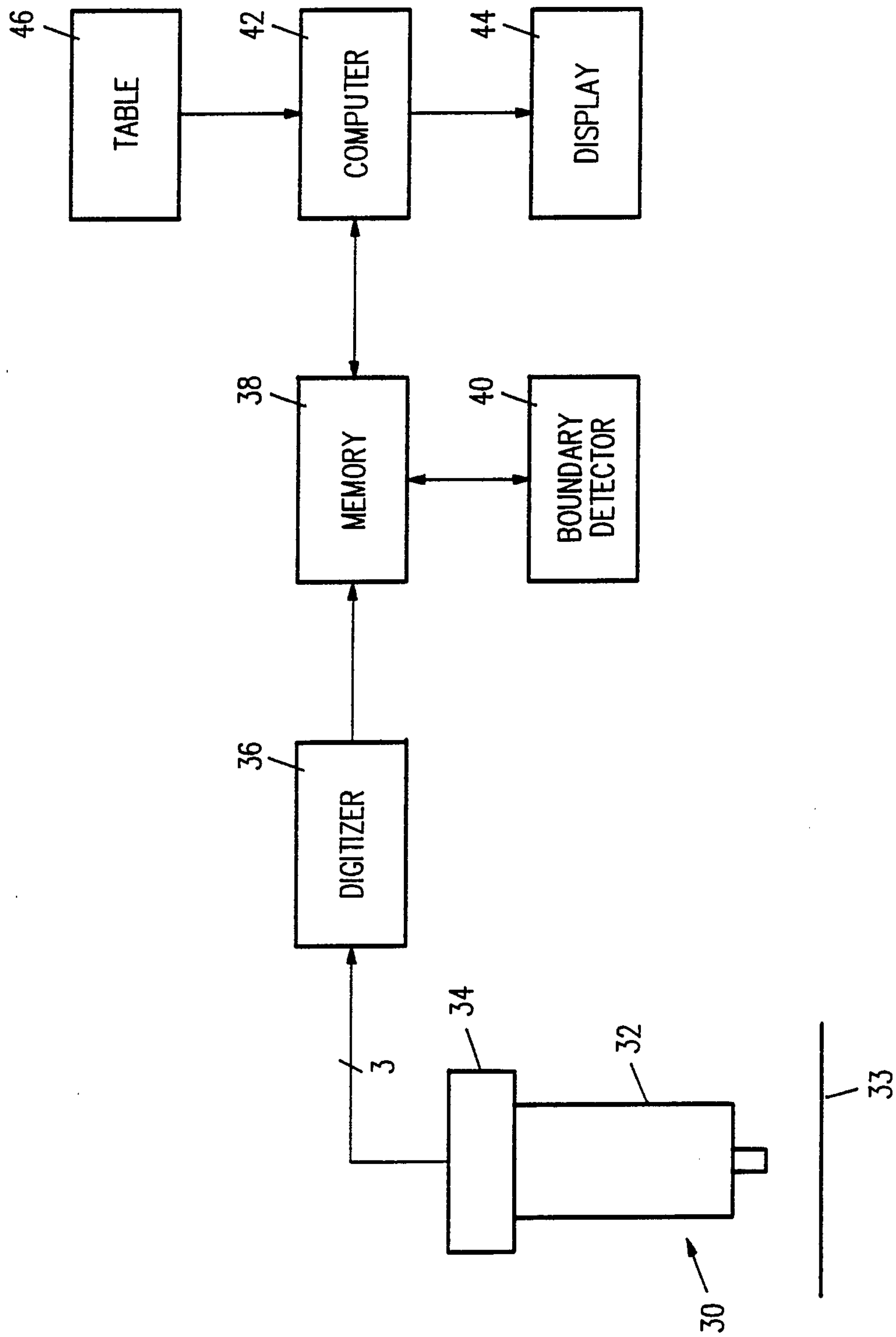

46
TABLE
42
COMPUTER
44
DISPLAY
38
MEMORY
40
BOUNDARY DETECTOR
36
DIGITIZER
34
3
32
33
30

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/09159

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :G06K 09/48
US CL :382/22, 18

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 382/22, 18, 6, 17, 21, 54

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

CHECKED REFERENCES CITED IN US PATENT 5,123,055

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

Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO, A, WO90/14639 (KASDAN ET AL.) 29 NOVEMBER 1990, whole document.	1-48
A	US, A, 4,075,604 (DEGASPERI) 21 FEBRUARY 1978, see fig. 3 and col. 1, line 63 - col. 2, line 13; col. 4, lines 12-31.	1, 16, 25, 37
A	US, A, 5,121,338 (LODDER) 09 JUNE 1992, see Fig. 2 and col. 6 - col 8, line 25; col. 5, lines 8-29.	1-9, 16, 18, 25-30, 37-42
A	US, A, 4,573,197 (CRIMMINS) 25 FEBRUARY 1986, see Abstract, Figs. 1-3.	1, 16, 25, 37
A	US, A, 3,705,383 (FRAYER) 05 DECEMBER 1972, see col. 20, lines 41-56.	8-10, 17-19, 29-31, 41-43

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:

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

"E" earlier document published on or after the international filing date

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
08 November 1993	FEB 07 1994

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	LARRY J. PRIKOCKIS
Facsimile No. NOT APPLICABLE	Telephone No. (703) 305-4791

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US, A, 4,782,389 (MAYWEATHER III) 01 NOVEMBER 1988, see Abstract, Fig. 1.	1, 16, 25, 37