(12) STANDARD PATENT (11) Application No. **AU 2007249099 B2**
(19) AUSTRALIAN PATENT OFFICE

(54) Title
Block-based noise detection and reduction method with pixel level classification granularity

(51) International Patent Classification(s)
G06T 7/60 (2006.01)

(21) Application No: **2007249099** (22) Date of Filing: **2007.12.18**

(43) Publication Date: **2009.06.25**
(43) Publication Journal Date: **2009.06.25**
(44) Accepted Journal Date: **2011.12.01**

(62) Divisional of:
2007237365

(71) Applicant(s)
Canon Kabushiki Kaisha

(72) Inventor(s)
Chen, Yu-Ling;Chen, Yu-Ling

(74) Agent / Attorney
Spruson & Ferguson, Level 35 St Martins Tower 31 Market Street, Sydney, NSW, 2000

(56) Related Art
WO 2000/046748
US 6778697

ABSTRACT

BLOCK-BASED NOISE DETECTION AND REDUCTION METHOD WITH PIXEL LEVEL CLASSIFICATION GRANULARITY

A method of detecting noise in an image (1900, 2000) of a compound document is disclosed. The image comprising pixels representing at least text data (1910,1930, 2010) and image data (1920, 2020) used to form the compound document. The noise comprises at least one of halftone, bleeding and texture noise. The method partitions (120) the image into a plurality of tiles (Fig. 11) each formed of a plurality of pixels and determines (140) a set of dominant colours for each said tile and associating each pixel with one of the determined dominant colours. The method then determines (1210) for each dominant colour a fragmentation statistic (1210, Fig. 13); and applies (1220) a threshold (NoiseCnt, NoSolid, NoiseType) to each said fragmentation statistic to detect the at least one of halftone, bleeding and texture noise in the corresponding dominant colour in the tile. The method may be extended to reduce detected noise in the image by processing the colours of each said tile by (a) merging each of the colours with noise to one said colour without noise if the noise is halftone noise or texture noise; and (b) removing the colours with noise if the noise is bleeding noise. The pixels of the tile are then quantised to those remaining dominant colours of said set.

2007249099 18 Dec 2007

S&F Ref: 831907

AUSTRALIA

PATENTS ACT 1990

COMPLETE SPECIFICATION

FOR A STANDARD PATENT

Name and Address of Applicant :	Canon Kabushiki Kaisha, of 30-2, Shimomaruko 3-chome, Ohta-ku, Tokyo, 146, Japan
Actual Inventor(s):	Yu-Ling Chen
Address for Service:	Spruson & Ferguson St Martins Tower Level 35 31 Market Street Sydney NSW 2000 (CCN 3710000177)
Invention Title:	Block-based noise detection and reduction method with pixel level classification granularity

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

2007249099 18 Dec 2007

BLOCK-BASED NOISE DETECTION AND REDUCTION METHOD WITH PIXEL LEVEL CLASSIFICATION GRANULARITY

TECHNICAL FIELD

The current invention relates to the detection, and desirably the removal, of artefacts, such as halftone, bleeding and texture noise, in a scanned compound document image for document image analysis.

BACKGROUND ART

The proliferation of scanning technology combined with ever increasing computational processing power has lead to many advances in the area of document analysis and systems for such analysis. These systems may be used to extract semantic information from a scanned document, often by means of optical character recognition (OCR) technology. This technology is used in a growing number of applications such as automated form reading. The technology can also be used to improve compression of a document by selectively using an appropriate compression method depending on the content of each part of the page (eg. bitmap image, graphical object image, text, etc.). Improved document compression lends itself to applications such as archiving and electronic distribution.

Document analysis can typically be broken into three stages. The first of these stages involves pixel colour analysis. The second stage is document layout analysis, which identifies content types such as text, backgrounds and images in different regions of the page. The final stage uses the results of the first two stages and some further analysis to create a final output. The desired final output depends on the application. Some typical applications include document compression, OCR, and vectorisation.

Pixel colour analysis involves segmenting the scanned images into perceptually uniform colour regions. The most common format is a binary image which can be obtained by various adaptive thresholding methods. Binary format is simple and effective for simple document images because generally documents have dark text on light background or light text on dark background. However as colour printing becomes more advanced and widely used, and thus the choices of colours on documents also get more diverse, binary representation becomes ineffective. Some combinations of colours cannot be thresholded because they have similar luminance values. Common examples are yellow or green text on white background. Colour pixel segmentation can be used to solve this problem. There are two common methods of colour pixel segmentation. The first method is grouping similarly coloured pixels together by giving them the same label. The second method is colour quantisation. Some applications also use the combination of the two methods. Fig. 19(a) and Fig. 20(a) are two examples 1900 and 2000 of typical scanned images for document analysis. The images 1900 and 2000 each include text and a background representing text data and image data used to form a compound document. Fig. 19(b) and Fig. 20(b) are corresponding binarised images derived from Fig. 19(a) and Fig. 20(a). Fig. 19(c) and Fig. 20(c) are the typical output of colour pixel segmentation from the same input.

All the pixel segmentation methods mentioned above suffer from noise because a huge degradation on the scanned images has occurred from the original raster image processed (RIP'ed) images through the original printing and subsequent scanning processes. Artefacts such as noise in scanned documents affects the accuracy of image type classification for document analysis applications. The types of noise include halftone, registration error, bleeding and JPEG artefacts. Halftone noise is generally the most critical

type. Most document analysis applications either employ a pre-processing stage to remove halftone noise or to embed halftone detection in image type classification with a geometrically coarse classification. Other noise is normally removed by removing small blobs or regions of labelled pixels or recursively merging small blobs. The noise removal process is time consuming and implementation costly.

In document copy applications (scan then print), the detection and removal operations of halftone noise are often done at the same time by a moving window with the centre pixel being the target. The process produces a set of pixels for the centre pixel using different filters in parallel. The classification state of the centre pixel decides which of the manipulated pixels should be output for the centre pixel. The size of the window normally is quite small. While this implementation is very efficient in hardware, it suffers inaccuracy in detection due to the lack of context information. Better context information can be achieved if the window is bigger. However the hardware implementation cost increases significantly when the window size increases. A software implementation for this method is not desirable because filter operations are very slow in software.

In document analysis applications such as OCR, image region segmentation and automatic form processing etc., the halftone detection process is often embedded into image region classification. Image region classification schemes can range from a simple classification such as halftone/non-halftone to a more complicated classification such as flat/contone/rough halftone/smooth halftone/line art. The classification normally works in larger non-overlapped blocks. A block of pixels is analysed and a set of features are extracted. The block is then classified into a predefined type based on the extracted features. For a block classified as halftone, a blurring function is normally applied to the whole block to remove halftone. This method is faster than the moving window style in

software implementations and the accuracy generally is also higher because the block normally provides better context information. However this method suffers from geometrically coarse classification because each block can only have one classification type in contrast to each pixel having a classification type in the moving window style.

Geometrically coarse halftone classification affects text visual quality especially in the case of text over halftone. Without halftone removal, the background halftone noise can be incorrectly selected as foreground text and subsequently affects further text processing such as OCR. On the other hand, if a text over halftone block is halftone removed by applying a blurring function to the block, the text in that block will appear blurry. Blurry text can cause broken or missing text strokes when the image is pixel segmented into perceptually uniform colour regions. This will directly affect the OCR results.

There is a need for a pixel colour analysis technique that segments scanned images with high accuracy thereby accommodating pixel level detail.

SUMMARY

The present disclosure provides pixel level noise type classification that may be performed during pixel colour segmentation processing in a non-overlapped block raster order. Pixel level halftone classification enables crisp text pixel segmentation for text in halftone regions without blurring the text. Other types of noise are also detected in the same process. The detected noise is reduced by reducing the dominant colours and re-quantisation. The pixel level noise type classification can also be used in the printing backend process to remove halftone noise or registration error on the scanned images and improve printing quality. These approaches effectively alleviate the problems in

geometrically coarse classification with integrated noise reduction in the pixel segmentation process.

In accordance with one aspect of the present disclosure there is provided a method of detecting noise in an image of a compound document. The image comprising pixels representing at least text data and image data used to form the compound document. The noise comprises at least one of halftone, bleeding and texture noise. The method partitions the image into a plurality of tiles each formed of a plurality of pixels and determines a set of dominant colours for each said tile and associating each pixel with one of the determined dominant colours. The method then determines for each dominant colour a fragmentation statistic ; and applies a threshold to each said fragmentation statistic to detect the at least one of halftone, bleeding and texture noise in the corresponding dominant colour in the tile.

The method may be extended to reduce detected noise in the image by processing the colours of each said tile by (a) merging each of the colours with noise to one said colour without noise if the noise is halftone noise or texture noise and (b) removing the colours with noise if the noise is bleeding noise. Then the pixels of the tile are quantised to those remaining dominant colours of the set.

Other aspects are also disclosed.

BRIEF DESCRIPTION OF THE DRAWINGS

At least one embodiment of the present invention will now be described with reference to the drawings, in which:

Fig. 1 is a flow chart for a method of detecting noise in a compound document image;

Fig. 2 is the processing flow chart of the "Colour Histogram" step of Fig. 1;

Fig. 3 is intentionally absent;

Fig. 4 is the processing flow chart of the "Find Dominant Colours" step of Fig. 1;

Fig. 5 is the decision flow chart of choosing a quantisation method for use in the method of Fig. 1;

Fig. 6 is intentionally absent;

Fig. 7 is the processing flow chart of pixel quantisation using TQM_1_THRESHOLD method;

Fig. 8 is the processing flow chart of pixel quantisation using TQM_2_THRESHOLD method;

Fig. 9 is the processing flow chart of pixel quantisation using TQM_HUE_THEN_LUM method;

Fig. 10 is the processing flow chart of pixel quantisation using TQM_LUM_THEN_HUE method;

Fig. 11 is an example of a quantised map;

Fig. 12 is a flow chart representing a preferred approach to "Noise Reduction";

Fig. 13 is an example of pixel quantisation statistics;

Fig. 14 is a processing flow chart of detecting halftone evidence in each dominant colour in the method of Fig. 12;

Fig. 15 is a processing flow chart of merging colours with halftone noise to other colours in the method of Fig. 12;

Fig. 16 illustrates the updated edge ratios and border length counters of the example in Fig. 13;

Fig. 17 is a processing flow chart of the bleeding removal step of Fig. 12;

Fig. 18 is a processing flow chart of the texture noise removal step of Fig. 12;

Figs. 19(a) to 19(d) represent one example of two different coloured texts on white background;

Fig. 20(a) to 20 (d) represent one example of non-halftoned text on halftoned background; and

Fig. 21 is a schematic block diagram representation of general purpose computer system upon which the arrangements described may be performed..

DETAILED DESCRIPTION INCLUDING BEST MODE

The methods of detection and removal of noise and other artefacts in a scanned compound document may be implemented using a computer system 2100, such as that shown in Fig. 21 wherein the processes of Figs. 1 to 20 may be implemented as software, such as one or more application programs executable within the computer system 2100. In particular, the steps of the methods of noise detection and removal are effected by instructions in the software that are carried out within the computer system 2100. The instructions may be formed as one or more code modules, each for performing one or more particular tasks. The software may also be divided into two separate parts, in which a first part and the corresponding code modules performs the noise detection and removal methods and a second part and the corresponding code modules manage a user interface between the first part and the user. The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer system 2100 from the computer readable medium, and then executed by the computer system 2100. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer system 2100 preferably effects an

advantageous apparatus for noise detection and removal from scanned compound documents.

As seen in Fig. 21, the computer system 2100 is formed by a computer module 2101, input devices such as a keyboard 2102, a mouse pointer device 2103and scanner 2119, and output devices including a printer 2115, a display device 2114 and loudspeakers 2117. An external Modulator-Demodulator (Modem) transceiver device 2116 may be used by the computer module 2101 for communicating to and from a communications network 2120 via a connection 2121. The network 2120 may be a wide-area network (WAN), such as the Internet or a private WAN. Where the connection 2121 is a telephone line, the modem 2116 may be a traditional "dial-up" modem. Alternatively, where the connection 2121 is a high capacity (eg: cable) connection, the modem 2116 may be a broadband modem. A wireless modem may also be used for wireless connection to the network 2120.

The computer module 2101 typically includes at least one processor unit 2105, and a memory unit 2106 for example formed from semiconductor random access memory (RAM) and read only memory (ROM). The module 2101 also includes an number of input/output (I/O) interfaces including an audio-video interface 2107 that couples to the video display 2114 and loudspeakers 2117, an I/O interface 2113 for the keyboard 2102 and mouse 2103 and optionally a joystick (not illustrated), and an interface 2108 for the external modem 2116, scanner 2119 and printer 2115. In some implementations, the modem 2116 may be incorporated within the computer module 2101, for example within the interface 2108. The computer module 2101 also has a local network interface 2111 which, via a connection 2123, permits coupling of the computer system 2100 to a local computer network 2122, known as a Local Area Network (LAN). As also illustrated, the

local network 2122 may also couple to the wide network 2120 via a connection 2124, which would typically include a so-called "firewall" device or similar functionality. The interface 2111 may be formed by an Ethernet™ circuit card, a wireless Bluetooth™ or an IEEE 802.11 wireless arrangement.

The interfaces 2108 and 2113 may afford both serial and parallel connectivity, the former typically being implemented according to the Universal Serial Bus (USB) standards and having corresponding USB connectors (not illustrated). Storage devices 2109 are provided and typically include a hard disk drive (HDD) 2110. Other devices such as a floppy disk drive and a magnetic tape drive (not illustrated) may also be used. An optical disk drive 2112 is typically provided to act as a non-volatile source of data. Portable memory devices, such optical disks (eg: CD-ROM, DVD), USB-RAM, and floppy disks for example may then be used as appropriate sources of data to the system 2100.

The components 2105 to 2113 of the computer module 2101 typically communicate via an interconnected bus 2104 and in a manner which results in a conventional mode of operation of the computer system 2100 known to those in the relevant art. Examples of computers on which the described arrangements can be practised include IBM-PC's and compatibles, Sun Sparcstations, Apple Mac™ or alike computer systems evolved therefrom.

Typically, the application programs discussed above are resident on the hard disk drive 2110 and read and controlled in execution by the processor 2105. Intermediate storage of such programs and any data fetched from the networks 2120 and 2122 may be accomplished using the semiconductor memory 2106, possibly in concert with the hard disk drive 2110. In some instances, the application programs may be supplied to the user encoded on one or more CD-ROM and read via the corresponding drive 2112, or

alternatively may be read by the user from the networks 2120 or 2122. Still further, the software can also be loaded into the computer system 2100 from other computer readable media. Computer readable media refers to any storage medium that participates in providing instructions and/or data to the computer system 2100 for execution and/or processing. Examples of such media include floppy disks, magnetic tape, CD-ROM, a hard disk drive, a ROM or integrated circuit, a magneto-optical disk, or a computer readable card such as a PCMCIA card and the like, whether or not such devices are internal or external of the computer module 2101. Examples of computer readable transmission media that may also participate in the provision of instructions and/or data include radio or infra-red transmission channels as well as a network connection to another computer or networked device, and the Internet or Intranets including e-mail transmissions and information recorded on Websites and the like.

The second part of the application programs and the corresponding code modules mentioned above may be executed to implement one or more graphical user interfaces (GUIs) to be rendered or otherwise represented upon the display 2114. Through manipulation of the keyboard 2102 and the mouse 2103, a user of the computer system 2100 and the application may manipulate the interface to provide controlling commands and/or input to the applications associated with the GUI(s).

Documents may be using the computer 2101 to the printer 2115 and may be scanned using the scanner 2119. Scanned documents may be stored electronically in the store 2109. The printer 2115 and scanner 2119 may collectively form a photocopier or otherwise be parts of a multi-function device (MFD). Alternatively scanned documents may be accessed via the networks 2120 and/or 2122.

Fig. 1 shows a flowchart of a method 100 of detecting noise in a compound document image. The method 100 is preferably implemented as software executable within the computer system 2100 as discussed above. The method 100 receives an input image 110 which is partitioned in step 120 into uniform sized tiles, each preferably of size 32 x 32 (1024) pixels when the input resolution is 300dpi. Different tile sizes can also be used. A tile may also be termed a block of the image. The tiles are preferably processed in raster order – that is from left to right, and top to bottom. The tiling of step 120 may be performed on the entirety of the image, or upon a band of the image in raster order. The following steps 130-180 of the method 100 are then performed on each tile, again preferably in raster order.

Step 130 involves a Colour Histogram process which reduces a tile of continuous tone pixels into an indexed image, essentially counting the number of pixels in the tile that accord with specific colours. YCbCr is the preferred colour space for the step 130, and also the entirety of the method 100, although other colour spaces may be used at one or more stages. In a preferred implementation the indexed image is formed of a palette of less than 33 colours and an indexed map. The number of colours in a palette depends on the similarity of colours in the tile. The colour value for each palette colour depends on the colour composition in the tile. For example a tile with text on plain background may only uses less than eight palette colours while a tile with halftoned image may require 32 palette colours. Since the number of colours in the palette is in this implementation limited to about 3% of the number of pixels in the tile, quantisation of the pixel colours to the palette colours is required.

Once the palette is produced, step 140, Find Dominant Colours, creates a set of dominant colours from the colour palette to represent the dominant information bearing

elements in the tile. Due to the physical size of a tile (2.7mm*2.7mm, being 32 x 32 pixels at 300 dpi), a tile normally can be represented with less than five dominant colours without losing the information bearing elements. For example, a tile with black and pink text on white background will need three dominant colours to represent the information bearing elements. Fig. 19(d) shows an image portion having three dominant colours (1910, 1920 and 1930) that represent the information bearing elements (two different coloured text items 1910 and 1930 and a background colour 1920). Fig. 20(d) shows an image portion with two dominant colours (2010 and 2020) that represent the information bearing elements (text 2010 and background 2020). A tile in a photograph or lineart region may need more than four dominant colours to represent the tile more visually correctly. However since the colours in the image regions are not important from the perspective of document analysis, applying a limitation of the number of dominant colours per tile generally does not pose a problem in respect of loss of image quality, however such can improve processing performance.

Returning to Fig. 1, step 150, Quantise to Dominant Colours, follows in which each pixel in a tile is colour quantised to one of the dominant colours and a quantised map is produced. The quantised map forms a record of the information bearing elements for document image analysis. However the information bearing elements are often polluted by the noise introduced through the printing and scanning process. Noise reduction on the quantised map is desirable to filter out the noise from the information bearing elements so that the document image analysis can produce better results. Subject to the quantity and distribution of quantised colours in the map, the preceding steps operate to identify at least pixels of each tile as noise colours and non-noise colours, the noise colours being one of halftone noise, texture noise and bleeding noise.

In Step 160, Noise Reduction, the quantised map is examined for halftone, bleeding and texture noise. Because the information bearing elements, from the perspective of document image analysis, are text, tables or drawings, if any of the noise is found, the existing dominant colours will be modified in order to remove the noise. Noise can be suppressed by reducing the number of dominant colours. For example black text on white background should be represented by two dominant colours – black and white. However in reality step 140 may produce three dominant colours – black, grey and white due to bleeding noise. If such noise is not removed before the actual document analysis takes place, the grey portion can either be merged to the black or white information bearing elements by the text analysis. In either case, the text quality is affected because the text will look either too thick or too thin. When the grey portion is removed through the pixel segmentation process by removing the grey dominant colour, the pixel segmentation quality is much better because the grey portion is evenly distributed between the black and white.

Step 170 then checks whether the number of dominant colours has changed. This is achieved by ascertaining the number of dominant colours remaining at step 160. In this regard, step 130 starts with a palette of 32 colours, and step 140 reduces that to a maximum of 4 colours. Step 160 can reduce that number further to between 1 and 3 colours depending on the pixel content of the tile. If the number of dominant colours has changed, the previously described process Quantised to Dominant Colours is performed again in step 180 to re-quantise pixels to the new set of dominant colours. Step 190 then decides whether there are more tiles to be processed. If so, processing returns to step 130. If not, the processing of the method 100 ends.

Each processing step and module in Fig. 1 will be described in further detail.

Colour Histogram

Fig. 2 shows the processing steps of Colour Histogram step 130 which reduces the continuous tone input image in a tile (32x32 of YCbCr pixels) into an indexed image. The colour histogram module 130 produces a palette of approximate colours after all pixels in a tile are examined once in pixel raster order. The colour histogram is formed of 32 bins arranged in 8 luminance bands and 4 colour columns. Each input pixel is placed into one of the 32 bins by a predetermined mapping method. Each colour bin has a colour accumulator, a pixel counter and a registration ID which is set by the YCbCr value of the first pixel placed into the bin. The predetermined mapping method may be changed depending on the composition of the pixels in a tile. As a result, there is no predetermined colour for each bin. The palette is decided by the pixel composition and the order by which the colours of pixels are processed.

The process of step 130 is performed for each pixel in a tile and starts with step 210, where a predetermined mapping is calculated using the equation below.

$$band = Y / 32$$

$$column = (|U - REF\_U| + |V - REF\_V|) * NORMALISING\_FACTOR[band]$$

$$mapped = band * 8 + column$$

The values of REF_U and REF_V are the chrominance of grey: that is REF_U = 128 and REF_V = 128 for 8-bit RGB input data. NORMALISING_FACTOR for each band is pre-calculated using the chosen REF_U and REF_V for normalising each band into 4 bins from the RGB colour space. The NORMALISING_FACTOR can be generated using, for example, the 17 numbered lines of the pseudo code below.

```
1     Set max_dist [0:7] to 0
2     for each r in 0 to 255
3     {
4         for each g in 0 to 255
5         {
6             for each b in 0 to 255
7             {
8                 c = RGB2YUV(r, g, b);
9                 band = c.y / 32;
10                dist = |c.u – REF_U | + |c.v – REF_V |;
11                if (dist > max_dist [band])
12                    max_dist [band] = dist;
13            }
14        }
15    }
16    for each band in 0 to 7
17        NORMALISING_FACTOR [band] = (1 / max_dist [band]) * 4
```

As each pixel is examined and assigned to a bin once only, each bin is registered with the colour value of the first pixel being placed into the bin. To prevent using up the bins in a band prematurely, step 215 checks whether the mapped bin is empty before placing a new pixel into the bin. If the mapped bin is empty, the process continues to step 225 where value "threshold" is assigned to old_err as the mapping error. Setting old_err to threshold will make the mapped bin the preference for placing the current pixel unless a non-empty bin with mapping error less than "threshold" is found. "Threshold" is defined as 1/8 of max_dist in each band as defined in the pseudo code above.

If the mapped bin is not empty, the process continues to step 220 to calculate the mapping error as min(|u - bin[mapped].u|, |v - bin[mapped].v|), where (u, v) are the chrominance values of the current input pixel and (bin[mapped].u, bin[mapped].v) are

the registration ID of the mapped bin. Next the process continues to step 230 where the mapping error is compared against the threshold. If the mapped bin has mapping error below the threshold, process continues to step 250. Otherwise the process continues to step 233 where a search starting from column 0 to 3 on the mapped band is performed to find a bin with the smallest mapping error. Step 233 starts a local loop and step 235 which follows calculates the mapping error in the same fashion as performed in step 220. Step 240 compares the calculated error, new_err, with old_err (which may be "threshold" if the mapped bin is empty) If new_err is smaller than old_err, step 245 identifies the loop increment "i" as the mapped value, otherwise the loop continues to the next value. When the local loop 233 ends, "mapped" is the final mapped bin. Step 250 follows.

In step 250, the mapped bin is checked again to determine if it is empty. If the mapped bin is empty, the mapped bin is registered at step 255 with the (u, v) values of the current input pixel. Otherwise the process continues to step 260 where the colour values of the current input pixel are accumulated to the mapped bin and the pixel count of the mapped bin is incremented. The bin mapping for each pixel is recorded as a 5-bit index and it forms an indexed map after a whole tile is processed. At the end of the pixel loop, the palette is generated from the average colours of the non-empty bins, thereby concluding the histogram step 130.

Find Dominant Colours

Fig. 4 shows the processing steps of Finding Dominant Colours step 140 from the palette produced by the Colour Histogram step 130. The process 140 determines the dominant colours to represent the information bearing elements for the current tile. The process 140 involves finding the initial colours from the palette and a sequence of refinement steps. Initially step 410 indentifies two initial colours, colour0 and colour1,

being the darkest and brightest colours of the palette. Step 420 then checks if the palette spreads more than four bands. When the palette spreads more than four bands, there is a higher chance that the tile may need three dominant colours to represent the information bearing elements. The third initial colour selection is essential since it will pick up an important but not densely populated colour from the palette as a dominant colour. If the palette spreads over more than 4 bands, step 430 searches for the third initial colour by testing each of the remaining palette colours using the condition below. If palette[n] satisfies the condition, palette[n] is chosen as the third initial colour. In the remaining description, uv_dist(C1, C2) is defined as |C1.u – C2.u| + |C1.v – C2.v| where C1 and C2 are two colours.

Condition: diff0 = uv_dist(|palette[n], colour0), and

diff1 = uv_dist(|palette[n], colour1)

where diff0 > 20 && diff1 > 20 && (diff0 + diff1) > 100

Step 437 then identifies a loop that refines the initial colours to the final dominant colours. The refinement only uses the most populated N bins that are not already included in the initial colours. The selection of N depends on the number of palette colours. The most populated N bins are denoted as P[N] where N ranges from 4 to 6. In step 445, bin colour P[i] may be merged to the existing dominant colours. The colour distance between P[i] and each of the existing dominant colours is calculated and if the distance is smaller than a predefined threshold (eg. 20), P[i] is merged to that existing dominant colour. Step 450 then checks if the merging attempt in step 445 was successful. If yes, the processing continues for the next colour, P[i+1]. Otherwise the processing proceeds to

step 455 to check if P[i] is very important. A palette bin is considered important if its colour difference to each of the existing dominant colours is greater than a predefined threshold (eg. 255), and its pixel count is also greater than a predefined threshold (eg. 30). If P[i] is important and the number of dominant colours (DC) is less than 4, a new dominant colour is created from P[i] in step 460. Otherwise the processing proceeds to step 465. If P[i] is substantial in the pixel count (eg. pixel count greater than 37) and its colour difference to each of the existing dominant colours is less than a predefined threshold (eg. 53) P[i] is merged in step 470 to one of the existing dominant colours with the closest colour distance. Otherwise P[i] is discarded and the processing continues by returning to step 445 for the next colour, P[i+1]. Discarding insubstantial palette colours can produce less polluted dominant colours. When the most populated N bins are all processed, the dominant colour generation process 140 is complete.

Quantise To Dominant Colours

The quantising to dominant colours processes of steps 150 and 180 involves selecting a suitable quantisation method and quantising pixels using the selected quantisation method. There are five methods of quantising pixels to the dominant colours, the choice of which depends on the composition of the dominant colours. Fig. 5 shows a process 610 for choosing the quantisation method. The four pre-determined dominant colour compositions determined in step 140 are tested in sequence as shown in Table 1 below. Dominant colours are named C0, C1, C2 and C3, where C0 is the darkest and C1 is the brightest. When the dominant colour composition does not satisfy any of the four pre-determined dominant colour compositions, a TQM_FULL_MAP method is used. Each of the quantisation methods will be explained in further details in the next paragraph.

Table 1:

Condition name	Colour composition	Condition
TQM_1_THRESHOLD	Two colours with distinctive luminance difference	2 dominant colours AND (C1.y – C0.y) > 50
TQM_2_THRESHOLD	Three colours with distinctive luminance difference but less chrominance difference	2 dominant colours AND (C1.y – C0.y) > 50
TQM_HUE_THEN_LUM	Three colours, C2 has distinctive chrominance difference to C0 and C1 and its luminance is close to the average of C0 and C1	3 dominant colours AND |C2.y - avg(C0, C1).y| < 30 AND uv_dist(C2, C0) > 40 AND uv_dist(C2, C1) > 40
TQM_LUM_THEN_HUE	Three colours, C2 has distinctive chrominance difference to either C0 or C1 and its luminance is close to either C0 or C1	3 dominant colours AND |C2.y - avg(C0, C1).y| >= 30 AND (uv_dist(C2, C0) > 40 OR uv_dist(C2, C1) > 40)

The sequence of testing noted above and indicated in Table 1 is illustrated in Fig. 5 where the process 610 initially tests two dominant colours against TQM_1_THRESHOLD in step 510. Where this is not met, the three dominant colours are tested in step 520. If

none satisfy, the TQM_FULL_MAP method is required. If the three colours do satisfy, these are progressively tested at steps 530, 540 and 550 to ascertain which other method is to be selected.

Fig. 7 shows the quantisation processing steps 700 of the TQM_1_THRESHOLD method. In step 710 a threshold is calculated from the average luminance of C0 and C1. The process then enters a pixel loop. In step 720, the luminance value of the current pixel C[i] in the tile is compared against the threshold calculated in step 710. If the result is greater, the current pixel is quantised to colour C1 and '1' is output as the quantised label. Otherwise the current pixel is quantised to colour C0 and '0' is output as the quantised label.

Fig. 8 shows the quantisation processing steps 800 of the TQM_2_THRESHOLD method. In step 810, two thresholds are calculated. The process then enters a pixel loop for each pixel in the tile. In step 820, the luminance value of the current pixel C[i] is compared against the lower threshold, ThresholdL. If the result is smaller, the current pixel is quantised to colour C0 and '0' is output as the quantised label. Otherwise the processing advances to step 840 and compares the luminance value of the current pixel C[i] against the higher threshold, ThresholdH. If the result is greater, the current pixel is quantised to colour C1 and '1' is output as the quantised label. Otherwise the current pixel is quantised to colour C2 and '2' is output as the quantised label.

Fig. 9 shows the quantisation processing steps 900 of the TQM_HUE_THEN_LUM method. In step 910 a threshold is calculated from the average luminance of colours C0 and C1. The process then enters a pixel loop over all pixels in the tile. In step 920 the chrominance distance from the current pixel C[i] to colours C0, C1 and C2 are calculated and compared. If it is closest to colour C2, the current pixel is quantised to colour C2

and '2' is output as the quantised label. Otherwise the processing advances to step 940 and compares the luminance value of the current pixel C[i] against the threshold. If the result is greater, the current pixel is quantised to C1 and '1' is output as the quantised label. Otherwise the current pixel is quantised to colour C0 and '0' is output as the quantised label.

Fig. 10 shows the quantisation processing steps 1000 of the TQM_LUM_THEN_HUE method. In step 1010 a threshold is calculated from the average luminance of colours C0 and C1. In step 1015 the luminance of colour C2 is compared against colours C0 and C1. If it is closer to colour C0, the process 1000 enters the pixel loop in step 1020. Otherwise the process enters the pixel loop in 1040, both loops being performed over all pixels in the tile. In step 1025 the luminance of the current pixel C[i] is compared against the threshold. If it is greater, the current pixel is quantised to colour C1 and '1' is output as the quantised label. Otherwise the process continues to step 1030 and compares the chrominance distance between the current pixel to colour C0 and to colour C2. If chrominance distance is closer to colour C2, the current pixel is quantised to colour C2 and '2' is output as the quantised label. Otherwise the current pixel is quantised to colour C0 and '0' is output as the quantised label. In step 1045 the luminance of the current pixel C[i] is compared against the threshold. If it is smaller, the current pixel is quantised to colour C0 and '0' is output as the quantised label. Otherwise the process continues to step 1030 and compares the chrominance distance between the current pixel to colour C1 and to colour C2. If chrominance distance is closer to colour C2, the current pixel is quantised to colour C2 and '2' is output as the quantised label. Otherwise the current pixel is quantised to colour C1 and '1' is output as the quantised label.

For the TQM_FULL_MAP method, the colour distances (eg. Manhattan) for each pixel in the current tile to all dominant colours are calculated. These distances are preferably "Manhattan" distances measured along each axis required to traverse between the pixels. The pixel is then quantised to the dominant colour with the smallest distance.

Noise Reduction

Fig. 12 shows the processing flow of Noise Reduction process 160. The input to this module is a quantised map with labels representing the dominant colours of each pixel. Fig. 11 is an example of a quantised map for a tile, in this case, an 8x8 tile having three dominant colours C0, C1 and C2. The noise reduction processing of Fig. 12 starts from step 1210 where pixel quantisation statistics are calculated. Pixel quantisation statistics represent a matrix of data of the fragmentation of the dominant colours in a tile. In the particular described embodiment, the fragmentation matrix data and statistics include an edge ratio for each quarter of the tile, an edge ratio for the whole tile, a border length between colours, a pixel count for each quarter of the tile and a pixel count for the whole tile. The edge ratio is defined as the number of border counts divided by the number of pixels for a dominant colour.

Fig. 13 shows examples of edge ratios for an 8 x 8 tile with four dominant colours C0, C1, C2 and C3. The tile in this example may be from an image portion (cf. text portion) of a scanned compound document and the edge ratios for the tile are seen to form an 8 x8 matrix representing the fragmentation of the regions of dominant colours. The borders are highlighted in black and limited to within the tile only. The borders define regions within the tile of pixels quantised to each dominant colour. The border length of a colour C0 is 20 and the pixel count of the colour C0 is 15, being the number of pixels enclosed by the corresponding border/region. As a result the edge ratio is 20/15. Edge

ratio indicates the degree of pixel fragmentation. The higher the edge ratio is the more fragmented a dominant colour is. The fragmentation indicates the noise state of the current tile. Edge ratios are calculated in every quarter of the tile and the whole tile for each dominant colour. The edge ratio statistics for each dominant colour in this example are also shown in Fig. 13. The border length between colours is denoted as edge[colour1]_[colour2]. For example border length between colours C0 and C1 is denoted as edge0_1 as shown in Fig.13.

After the pixel quantisation statistics are calculated, the process 160 of Fig. 12 proceeds to step 1220 to detect halftone noise for each dominant colour as described below. The output of step 1220 is the existence of halftone noise found in each dominant colour. Next in step 1230, the dominant colours with halftone noise are merged to other dominant colours. Once the dominant colours with halftone noise are merged, and the border lengths between those merged dominant colours are updated in step 1240. The process 160 then continues to step 1250 to remove bleeding noise, if it exists, as described below. Next in Step 1260, texture noise will be removed if it exists. The details of each step will be explained further next.

Fig. 14 shows the details of step 1220. The existence of halftone noise is determined by the fragmentation state of each dominant colour in each quarter of the tile and as a whole. The process 1220 starts from step 1410 where variables NoiseCnt and NoSolid are reset. NoiseCnt indicates how many quarters are fragmented. NoSolid is a flag indicating the whole tile is fragmented. Each dominant colour is tested for halftone noise evidence in turn in step 1415. A loop over each quarter in step 1420 tests the quarter fragmentation state for the current dominant colour where NoiseCnt and NoSolid are updated according to the statistics obtained from step 1210. In step 1425 the quarter edge

ratio, EdgeRatioQtr, for each dominant colour is compared against a threshold. If it is greater than the threshold, NoiseCnt for the dominant colour is incremented. The threshold is a function of the quarter pixel count. Otherwise the process 1220 enters step 1430 and compares EdgeRatioQtr against another threshold for solidness testing. If it is smaller, NoSolid is set to false. Once the process finishes checking all quarter edge ratios for every dominant colour, it enters another colour loop, formed by step 1445, to consolidate the halftone noise evidence for each dominant colour. If NoiseCnt is higher than a threshold or EdgeRatio is higher than another threshold, NoiseType is set to HALFTONE.

Fig. 15 shows the details of step 1230 of Fig. 12. The process starts from step 1510 to test whether the tile can be flattened. Flattening involves reducing the number of dominant colours to a single, solid colour. Specifically, when NoSolid is true and the number of halftone noise colours equals to the number of dominant colours, the test of step 1510 is satisfied and the tile is flattened to an average colour of all colours in the tile in step 1515. Otherwise the process advances to step 1520 to test whether the tile should be made bi-level. When the number of halftone noise colours is one less than the number of dominant colours, or there is one halftone noise colour among three dominant colours, the tile is made bi-level. Step 1540 and loop 1545 form the bi-level process which starts by choosing two base colours in step 1540. Colours C0 and C1 are the default two base colours. Colour C0 is be replaced by colour C2 if its pixel count is smaller than colour C2 and the pixel count of colour C1 is more than twice of colour C0. If colour C0 is not replaced, the same test is applied to colour C1. Next the process 1230 enters loop 1545 to merge the remaining colours to the two base colours. The colours are merged to one of the two base colours based on the following sequence. First, step 1550 tests the border length of the edge between the two dominant colours, essentially the edge that differentiates the

corresponding adjacent regions of dominant colour. When the base colour with more pixels does not have twice as many pixels as the other one, the colour is merged to the base colour with the longer shared border in step 1595. Otherwise the process moves to step 1560 to calculate the luminance difference between the current colour and two base colours. If the luminance is very close to the 1st base colour, the current colour is merged to the 1st base colour in step 1580. Otherwise it tests whether the current luminance is very close to the 2nd base colour. If so, the current colour is merged to the 2nd base colour in step 1585. Otherwise the process moves to step 1575 to decide whether the current colour is almost equally close to the 1st and the 2nd base colour in luminance. If so, the colour is merged to the base colour with the longer shared border in step 1595. Otherwise the colour is merged to the base colour with closer luminance (Euclidean, ie. line –of-sight) distance in step 1590. If the tile can not be made bi-level in step 1520, the method 1230 proceeds to step 1535 which operates to merge each colour with halftone noise to one of the other colours depending on the longest shared border.

Returning to Fig. 12, once the halftone noise colours are merged, the process 160 continues to step 1240 to update the border length counters and edge ratios. Only the whole tile edge ratios are updated which can be done by updating the border length counters depending on how the dominant colours are merged. For example, Colour2 in the Fig. 13 example has been merged to Colour1, Edge0_2 is therefore accumulated to Edge0_1 and Edge2_3 accumulated to Edge1_3. The result of the update for the example in Fig. 13 is shown in Fig. 16.

After step 1240, the process continues to step 1250 for bleeding removal, which is shown in detail in Fig.17. To prevent false bleeding detection, which may result in removing a genuine dominant colour, the process 1250 commences with steps 1710

and 1720 to decide whether bleeding removal should be performed on the current tile. For 3 coloured tiles, the bleeding removal is only enabled when the 3rd colour satisfies the following conditions:

Small pixel count
AND
(
Quite different from the 3rd dominant colour of the tile above or left
OR
Close to the 1st dominant colour
OR
Close to the 2nd dominant colour
)

The measures for "quite different" and "close" used in the condition above may be determined using one or more colour distance thresholds. The thresholds may be predetermined based upon the type of image or expected colours being detected (eg. black text on solid colour background). Further, a value of "small" may be established or predetermined based on the size of the tile. For example, for an 8 x 8 tile, "small" may be set to be a region having less than 4 pixels.

For 4-coloured tiles, the bleeding removal is only enabled when both the left tile and above tile have less than four colours. If the tile satisfies the bleeding removal conditions, the process continues to step 1730 to check in turn whether colours 2 and 3 are bleeding colours. In this regard, step 1730 operates by which steps 1740-1780 are performed once for colour 2, and once for colour 3. Otherwise the process 1250 terminates. Step 1740 checks whether the colour has not been merged by halftone merging and has a high edge ratio. If not, the process continues to process the next colour.

Otherwise the process continues to step 1750 and then step 1760 where the longest border length of the current colour to a darker colour and brighter colour are assigned to edge0 and edge1. Next, step 1770 tests whether edge0 and edge1 are both greater than a threshold. The threshold is calculated using the edge count of the current colour. If the test is true, the current colour is removed at step 1780. Otherwise the process 1250 continues to process the next colour until both colour 2 and 3 have been processed.

Returning to Fig. 12, after bleeding removal 1250, the noise reduction process 160 continues to step 1260 to remove texture noise, if there is more than one colour remaining. Background textures are preferably removed from the pixel segmentation output because they are not information bearing elements. Fig. 18 shows the details of the texture noise removal process 1260. The processing starts at step 1810 by deciding whether to reduce colours or flatten the tile based on whether there are only two remaining dominant colours. When there are only two remaining dominant colours, step 1820 is taken to decide whether the tile can be flattened. When the border length between the two remaining colours is very long and their colour distance is not far apart, the tile will be flattened. Otherwise no action is taken and the process 1260 ends. When there are still more than two remaining dominant colours (ie. 3 or 4 colours), a loop 1825 is performed to reduce colours. Steps 1830 and 1840 calculate the weighted distances, dist0 and dist1, to colour0 and colour1 respectively for colour c. The weighted distance is the colour distance scaled by the shared border length between colour0 or colour1 and colour c and their pixel count difference. Next the process moves to step 1845. If dist0 or dist1 is smaller than a threshold, colour c is merged to colour0 or colour1 in step 1850. Otherwise the process continues to the next colour until all colours have been processed.

As a consequence of the processing described above, various forms of noise in a pixel-based image can be detected and reduced in a fashion that permits a modified image to be more readily segmented. This can aid pixel classification for OCR and may also be used for optimised image compression. The block-based processing performed upon tiles of the image accommodates and allows the processing to adapt to a range of colours and classes across the entire image.

INDUSTRIAL APPLICABILITY

The arrangements described are applicable to the computer and data processing industries and particularly for the removal of noise from images required to be segment or classified to permit easier and more reliable segmentation and classification.

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

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

The claims defining the invention are as follows:

1. A method of detecting noise in an image of a compound document, said image comprising pixels representing at least text data and image data used to form the compound document, said noise comprising at least one of halftone, bleeding and texture noise, the method comprising the steps of:

(a) partitioning the image into a plurality of tiles each formed of a plurality of pixels;

(b) determining a set of dominant colours for each said tile and associating each pixel with one of the determined dominant colours;

(c) determining for each dominant colour a fragmentation statistic; and

(d) applying a threshold to each said fragmentation statistic to detect the at least one of halftone, bleeding and texture noise in the corresponding dominant colour in the tile.

2. A method according to claim 1 wherein step (b) comprises quantising a colour of pixels to one of said dominant colours.

3. A method according to claim 1 wherein step (c) comprises determining for each tile a fragmentation matrix representing a statistic for each said dominant colour.

4. A method according to claim 1, 2, or 3 wherein the fragmentation statistic comprises an edge ratio for each said dominant colour.

5. A method according to claim 1 wherein step (c) comprises identifying a border forming a region associated with each said dominant colour and the fragmentation statistic comprises a measure of a distance about each said region.

6. A method according to claim 5 wherein said statistic comprises a ratio of said measure with a number of pixels enclosed by the forming the corresponding region

7. A method according to claim 1 wherein the method is performed in a raster fashion of the tiles in the image.

8. A method of reducing noise in an image of a compound document, said noise comprising at least one of halftone, bleeding and texture noise, said method comprising the steps of:

(1) detecting noise in said image according to the method of any one of claims 1 to 7, said detecting thereby identifying at least pixels of each said tile as pixels having colours with noise and pixels having colours without noise, the noise being one of halftone noise, texture noise and bleeding noise;

(2) processing the colours of each said tile by:

(2a) merging each of the colours with noise to one said colour without noise if the noise is halftone noise or texture noise;

(2b) removing the colours with noise if the noise is bleeding noise; and

(3) quantising the pixels of the tile to those remaining dominant colours of said set.

9. A method according to claim 8 when dependent on claim 5 wherein step (2a) comprises merging the colour of a first region with the colour of an adjacent second region sharing a longest border.

10. A method according to claim 8 or 9 wherein step (2a) comprises comparing a luminance distance between a region having one dominant colour and adjacent regions with each having others of the dominant colours and merging colours of the regions with the smaller luminance distance.

11. A method according to claim 8, 9 or 10 wherein step (2b) comprises comparing a luminance of edges between a current region and adjacent brighter and darker regions against a threshold and removing the current colour where both edge luminances exceed the threshold.

12. A method according to claim 11 where the threshold is adapted based on an edge count of the current colour.

13. Computer apparatus for detecting noise in an image of a compound document, said image comprising pixels representing at least text data and image data used to form the compound document, said noise comprising at least one of halftone, bleeding and texture noise, the apparatus comprising:

means for partitioning the image into a plurality of tiles each formed of a plurality of pixels;

means for determining a set of dominant colours for each said tile and associating each pixel with one of the determined dominant colours;

means for determining for each dominant colour a fragmentation statistic; and

means for applying a threshold to each said fragmentation statistic to detect the at least one of halftone, bleeding and texture noise in the corresponding dominant colour in the tile.

14. Computer apparatus according to claim 13 further comprising:

means for processing the colours of each said tile by:

(a) merging each of the colours with noise to one said colour without noise if the noise is halftone noise or texture noise; and

(b) removing the colours with noise if the noise is bleeding noise; and

means for quantising the pixels of the tile to those remaining dominant colours of said set, to thereby reduce noise in the image of the compound document.

15. A computer readable medium having a computer program recorded thereon, said program being executable in a computer to detect noise in an image of a compound document, said image comprising pixels representing at least text data and image data used to form the compound document, said noise comprising at least one of halftone, bleeding and texture noise, the program comprising:

code for partitioning the image into a plurality of tiles each formed of a plurality of pixels;

code for determining a set of dominant colours for each said tile and associating each pixel with one of the determined dominant colours;

code for determining for each dominant colour a fragmentation statistic; and

code for applying a threshold to each said fragmentation statistic to detect the at least one of halftone, bleeding and texture noise in the corresponding dominant colour in the tile.

16. A computer readable medium according to claim 15 where the program further comprises:

code for processing the colours of each said tile by:

(a) merging each of the colours with noise to one said colour without noise if the noise is halftone noise or texture noise; and

(b) removing the colours with noise if the noise is bleeding noise; and

code for quantising the pixels of the tile to those remaining dominant colours of said set, to thereby reduce noise in the image of the compound document.

17. A method of detecting noise in an image of a compound document, the method being substantially as described herein with reference to any one of the embodiments as that embodiment is illustrated in the drawings.

18. A method of reducing noise in an image of a compound document, the method being substantially as described herein with reference to any one of the embodiments as that embodiment is illustrated in the drawings.

19. Computer apparatus adapted to perform the method of claim 17 or 18.

20. A computer program adapted to perform the method of claim 17 or 18.

Dated this 17th day of DECEMBER 2007

CANON KABUSHIKI KAISHA

Patent Attorneys for the Applicant

Spruson&Ferguson

2007249099 18 Dec 2007

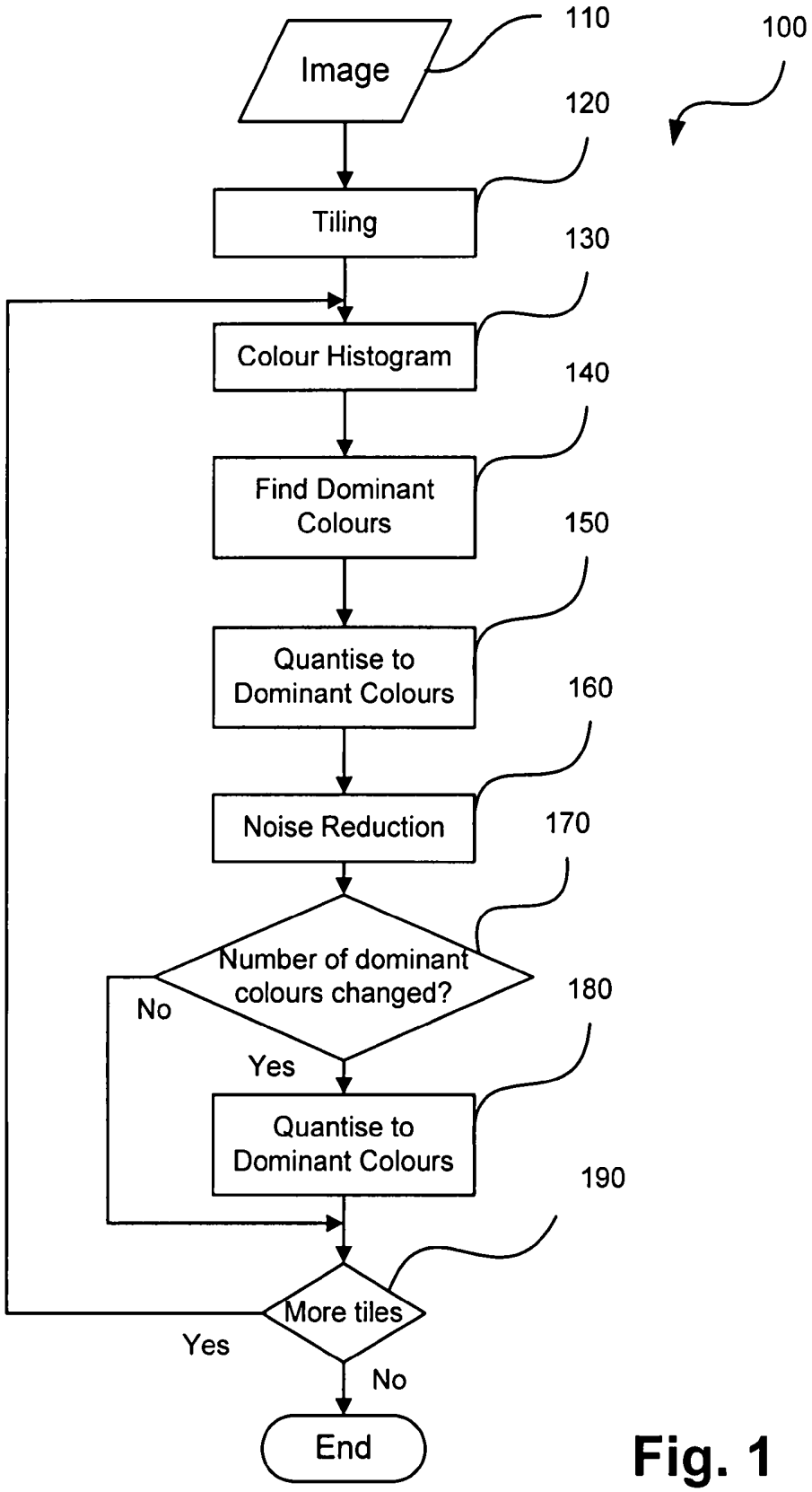


Fig. 1

2007249099 18 Dec 2007

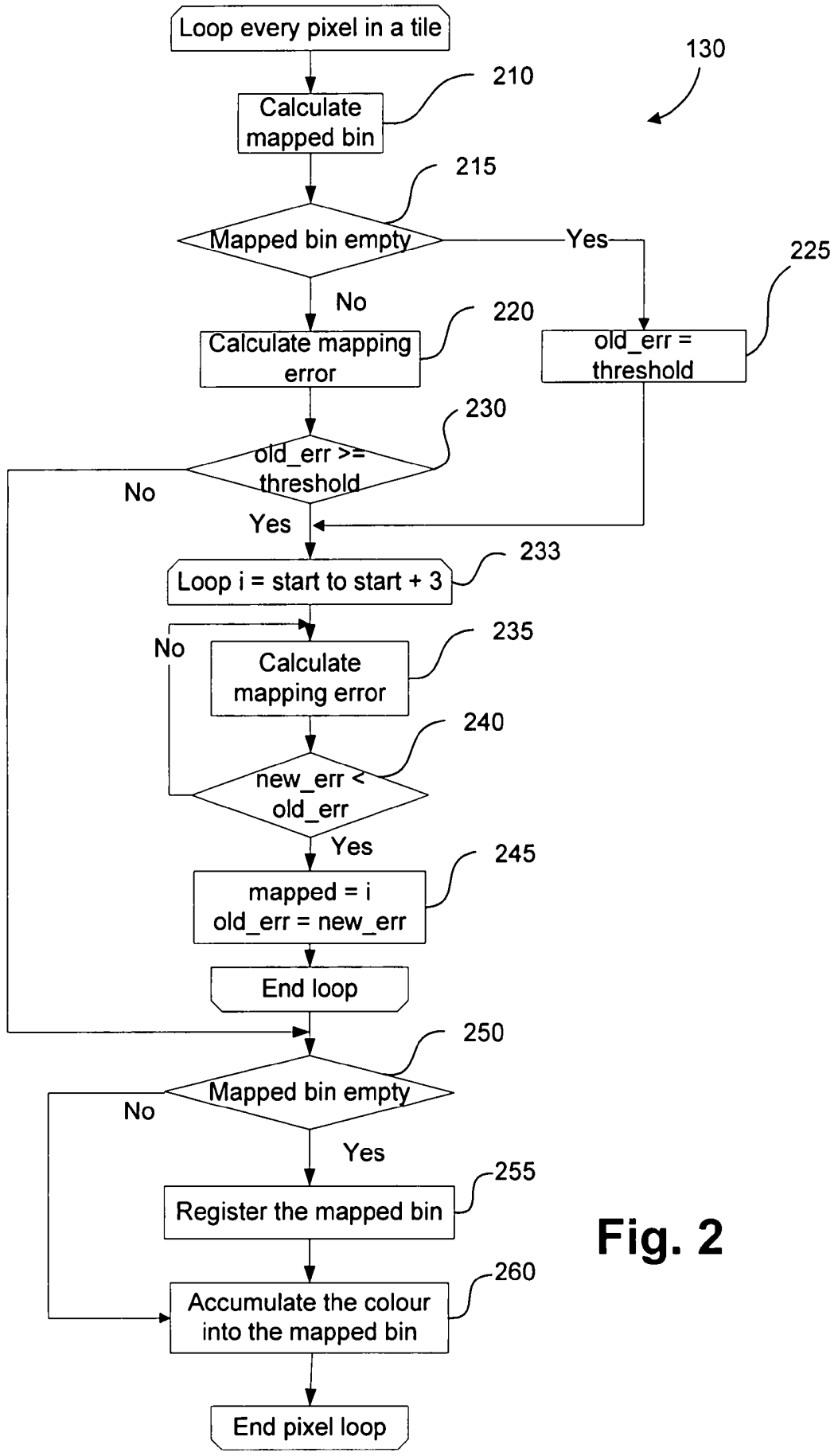


Fig. 2

2007249099 18 Dec 2007

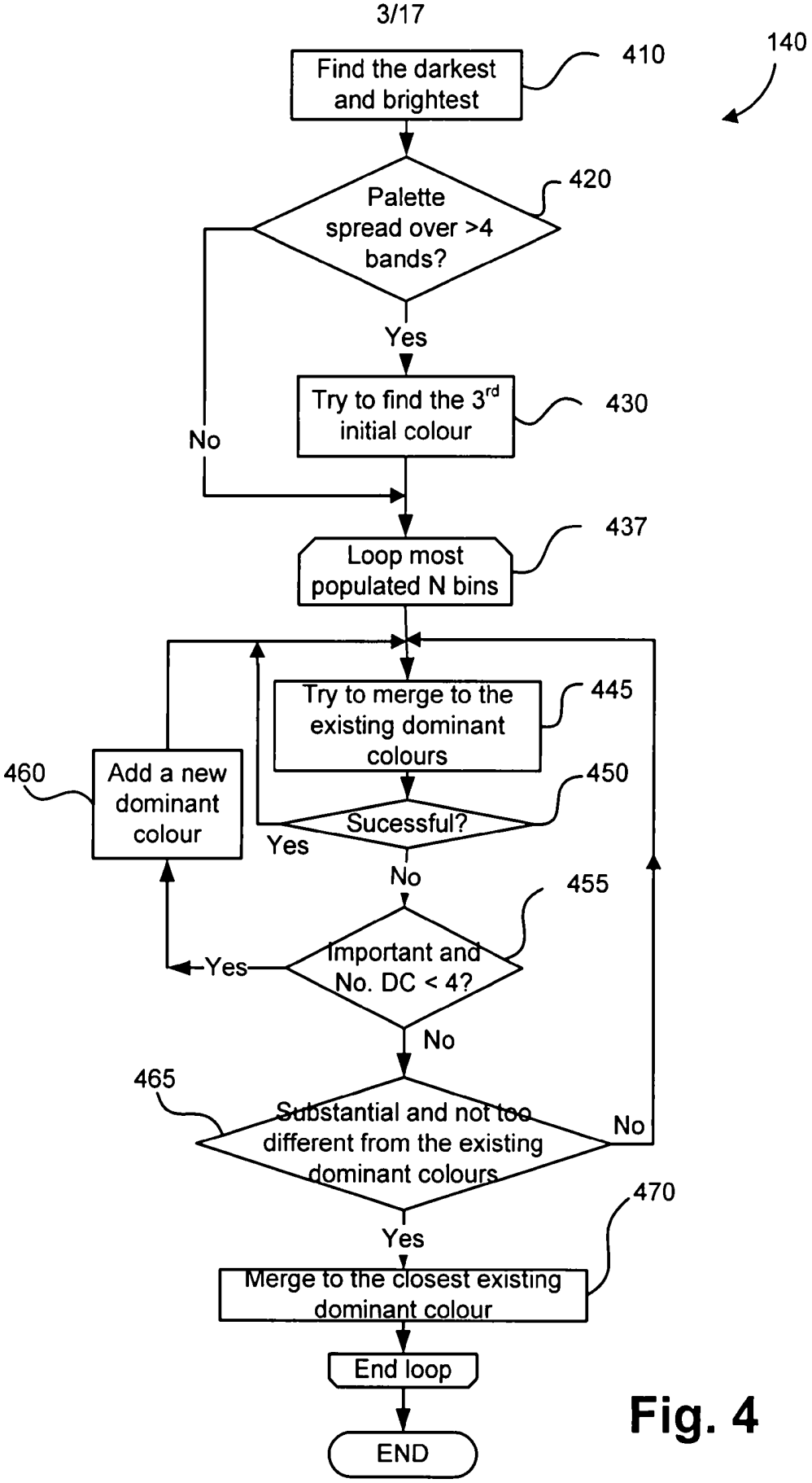


Fig. 4

2007249099 18 Dec 2007

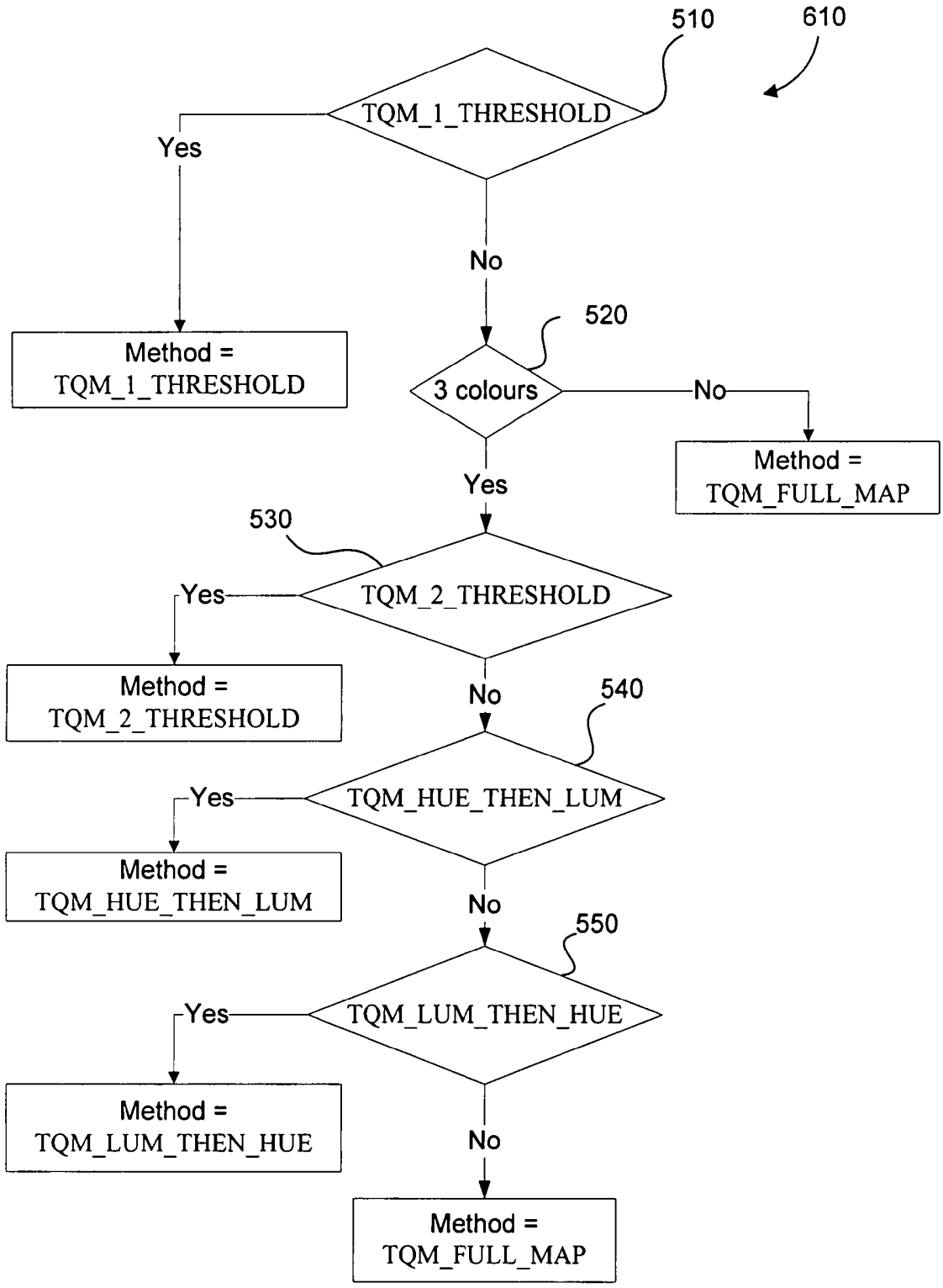


Fig. 5

2007249099 18 Dec 2007

Fig. 3

Intentionally absent

Fig. 6

Intentionally absent

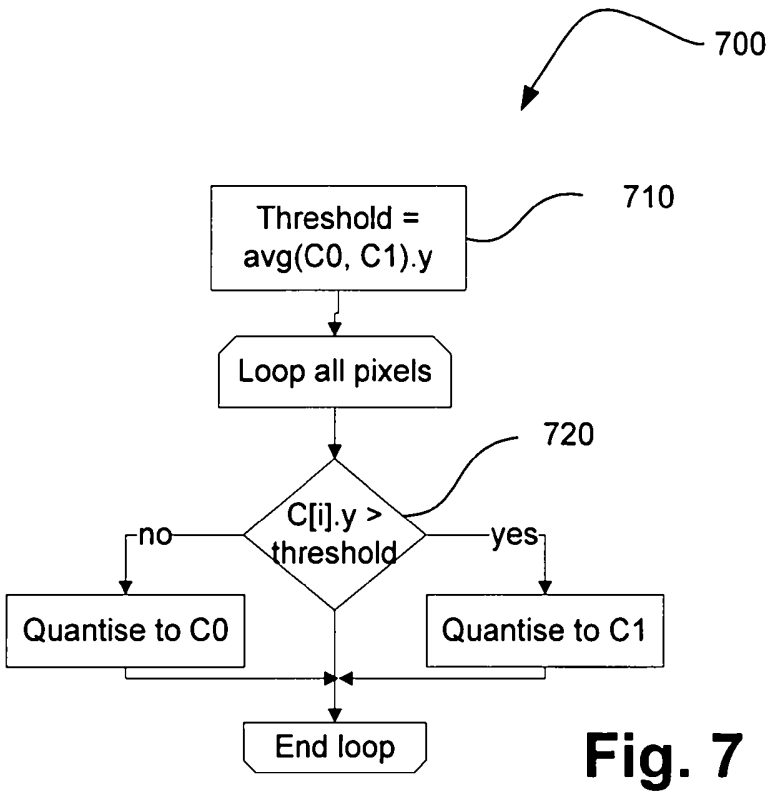


Fig. 7

2007249099 18 Dec 2007

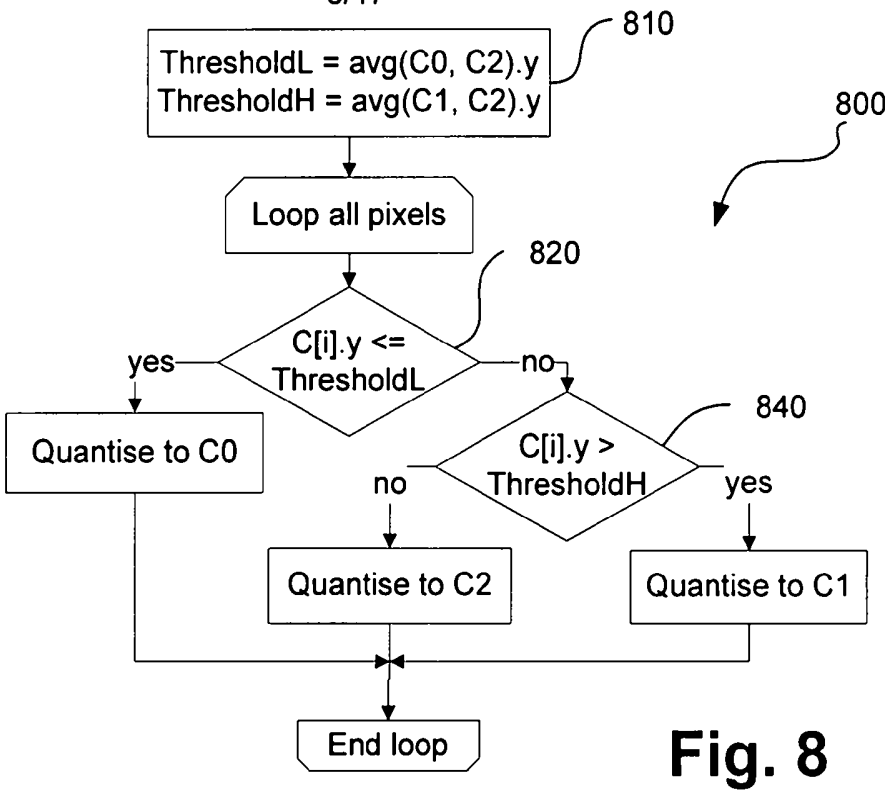


Fig. 8

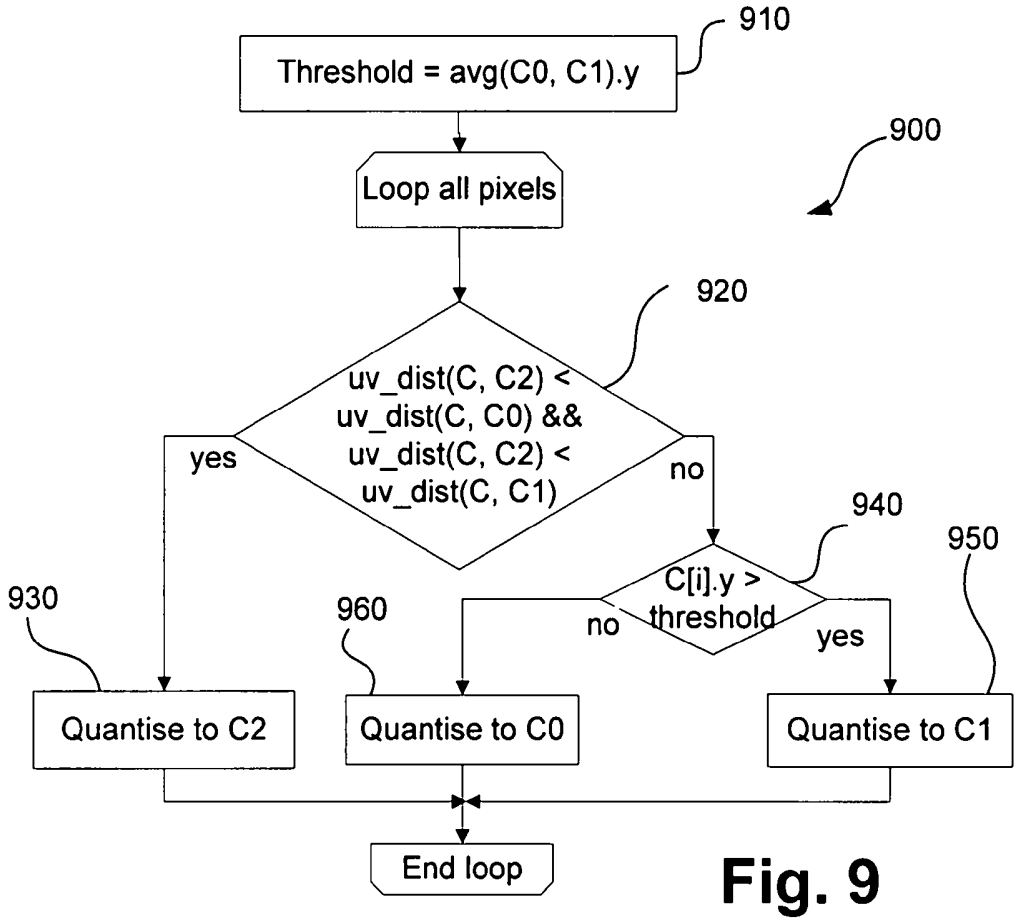


Fig. 9

2007249099 18 Dec 2007

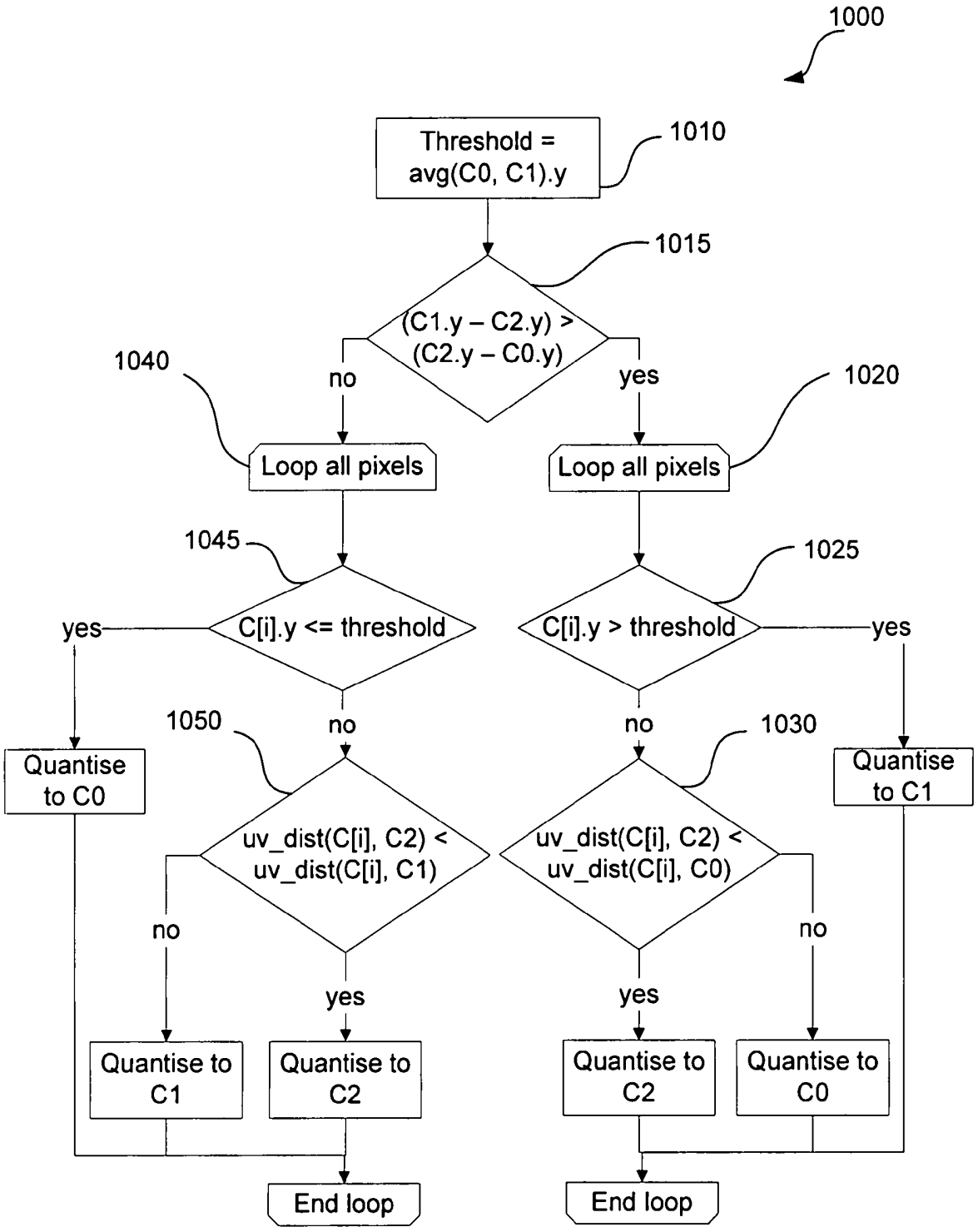
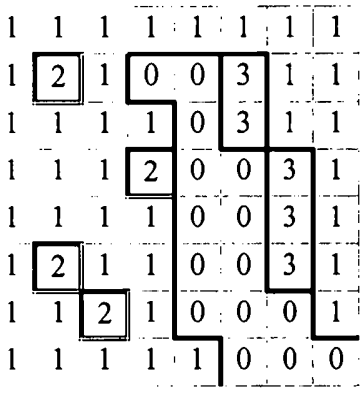


Fig. 10

1	1	1	1	1	1	1	1
1	2	1	0	0	1	1	1
1	1	1	1	0	1	1	1
1	1	2	1	0	0	1	1
1	1	1	1	0	0	1	1
1	2	1	1	0	0	1	1
1	1	2	1	0	0	0	1
1	1	1	1	1	0	0	0

Fig. 11



Quarter tile edge ratio for C0 = 5/1, 5/4, NA, 7/10
Quarter tile edge ratio for C1 = 12/13, 5/12, 12/14, 7/6
Quarter tile edge ratio for C2 = 7/2, NA, 9/2, NA
Quarter tile edge ratio for C3 = NA, 9/3 NA, 5/2

Edge0_1 = 12 Edge1_2 = 15
Edge0_2 = 1 Edge1_3 = 7
Edge0_3 = 7 Edge2_3 = 0

Whole tile edge ratio for C0 = (Edge0_1+Edge0_2+Edge0_3)/15 = 20/15
Whole tile edge ratio for C1 = ((Edge0_1+Edge1_2+Edge1_3) = 34/45
Whole tile edge ratio for C2 = (Edge0_2+Edge1_2+Edge2_3) = 16/4
Whole tile edge ratio for C3 = (Edge0_3+Edge1_3+Edge2_3) = 14/5

Fig. 13

2007249099 18 Dec 2007

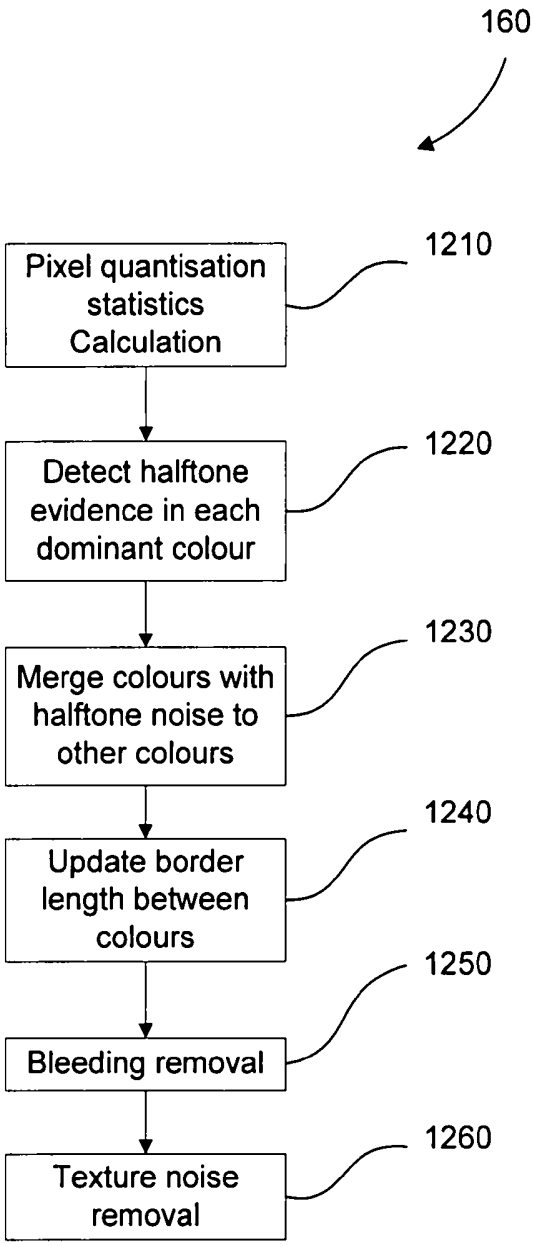


Fig. 12

2007249099 18 Dec 2007

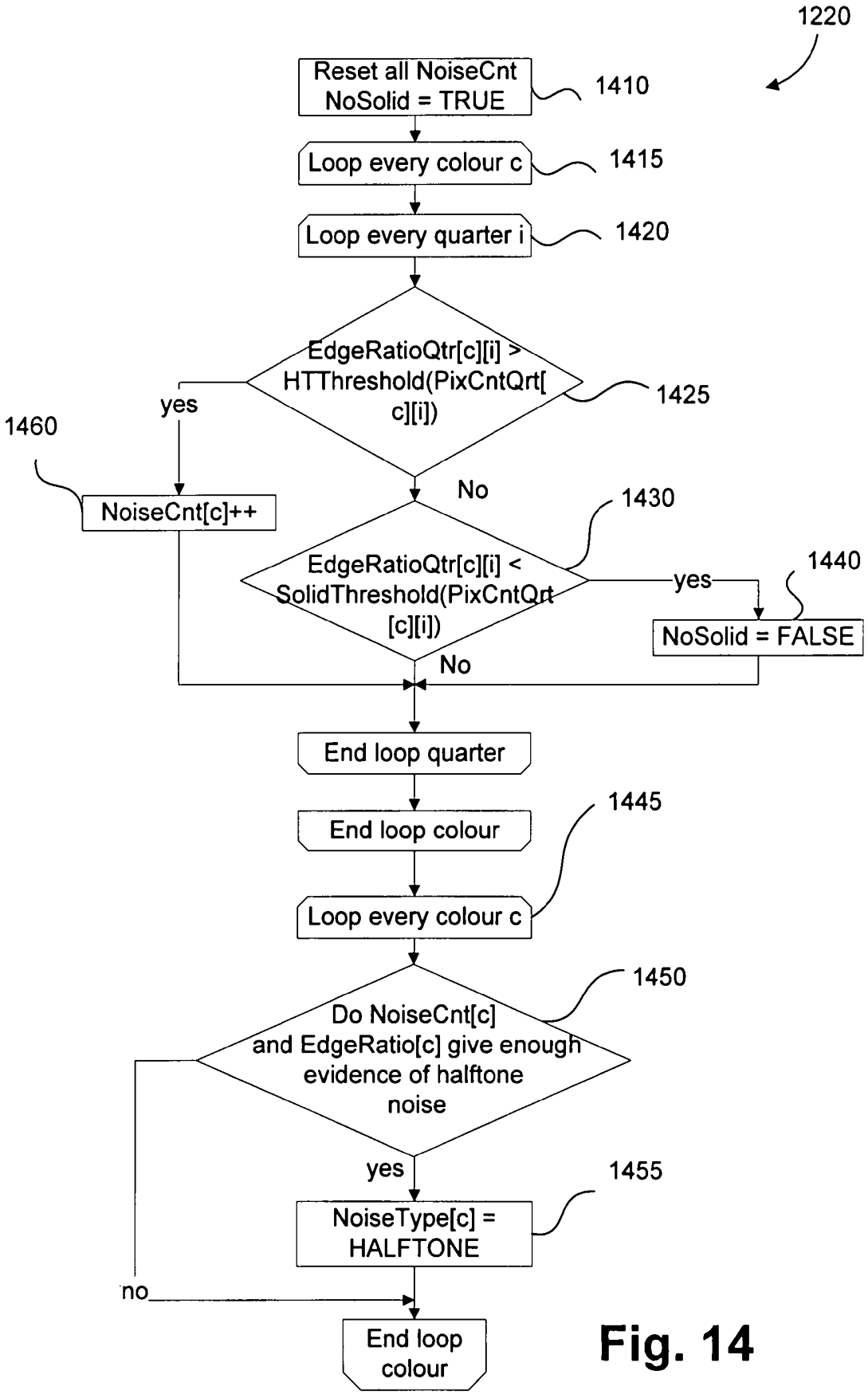


Fig. 14

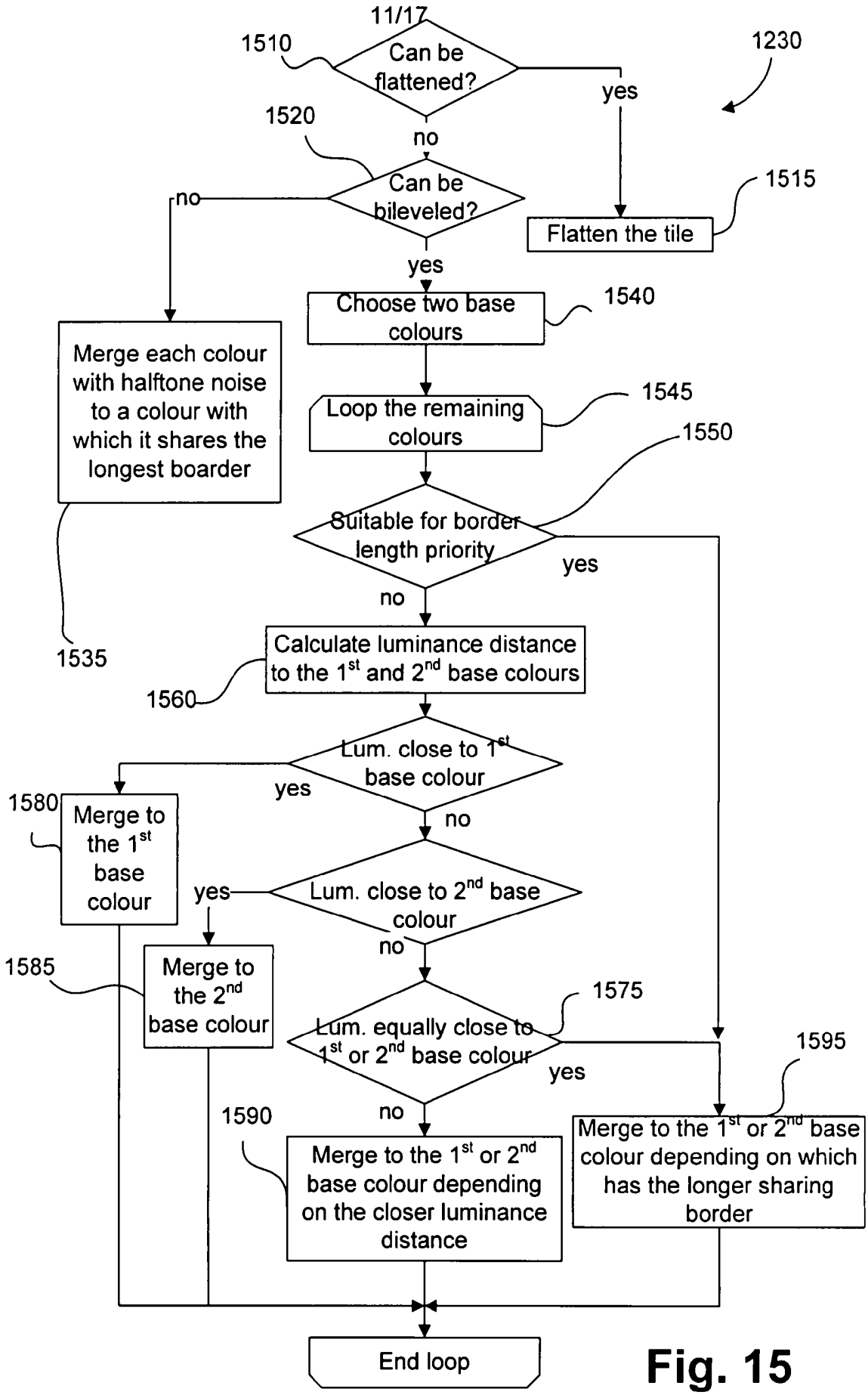
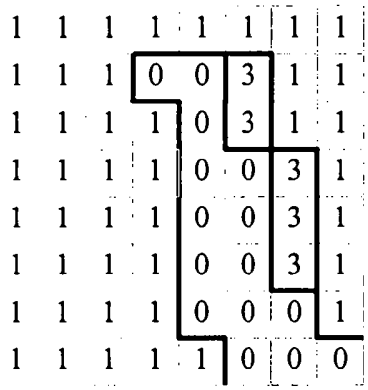

1510
Can be flattened?
yes
1230
no
1520
Can be bileveled?
no
1515
Flatten the tile
yes
Choose two base colours
1540
Merge each colour with halftone noise to a colour with which it shares the longest boarder
Loop the remaining colours
1545
1550
Suitable for border length priority
yes
no
1535
Calculate luminance distance to the 1st and 2nd base colours
1560
Lum. close to 1st base colour
yes
1580
Merge to the 1st base colour
no
Lum. close to 2nd base colour
yes
1585
Merge to the 2nd base colour
no
1575
Lum. equally close to 1st or 2nd base colour
yes
1595
Merge to the 1st or 2nd base colour depending on which has the longer sharing border
no
1590
Merge to the 1st or 2nd base colour depending on the closer luminance distance
End loop

Fig. 15

2007249099 18 Dec 2007



Edge0_1 = 13 Edge1_2 = 0
Edge0_2 = 0 Edge1_3 = 7
Edge0_3 = 7 Edge2_3 = 0

Whole tile edge ratio for C0 = (Edge0_1+Edge0_2+Edge0_3)/15 = 20/15
Whole tile edge ratio for C1 = ((Edge0_1+Edge1_2+Edge1_3) = 20/49
Whole tile edge ratio for C2 = NA
Whole tile edge ratio for C3 = (Edge0_3+Edge1_3+Edge2_3) = 14/5

Fig. 16

2007249099 18 Dec 2007

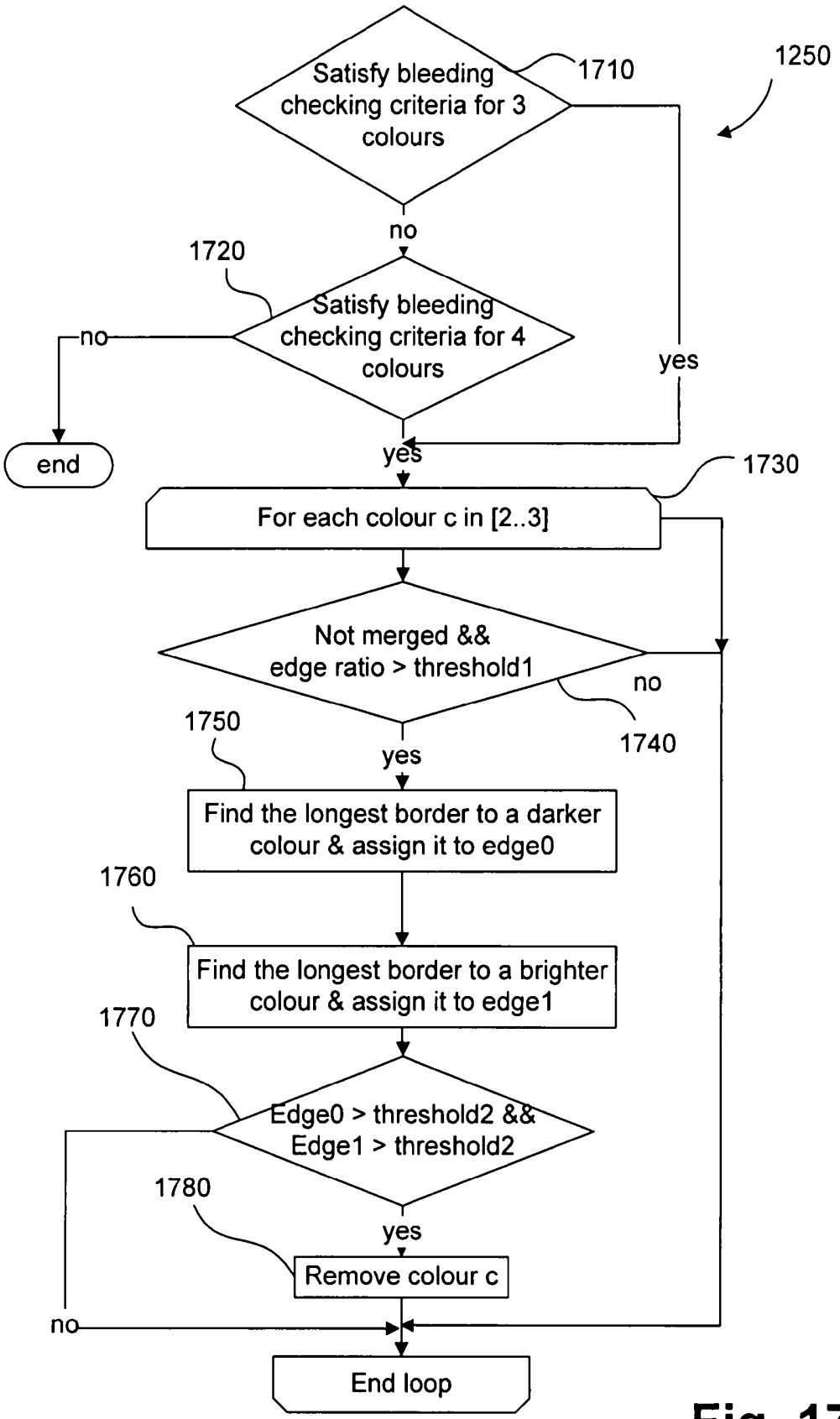


Fig. 17

2007249099 18 Dec 2007

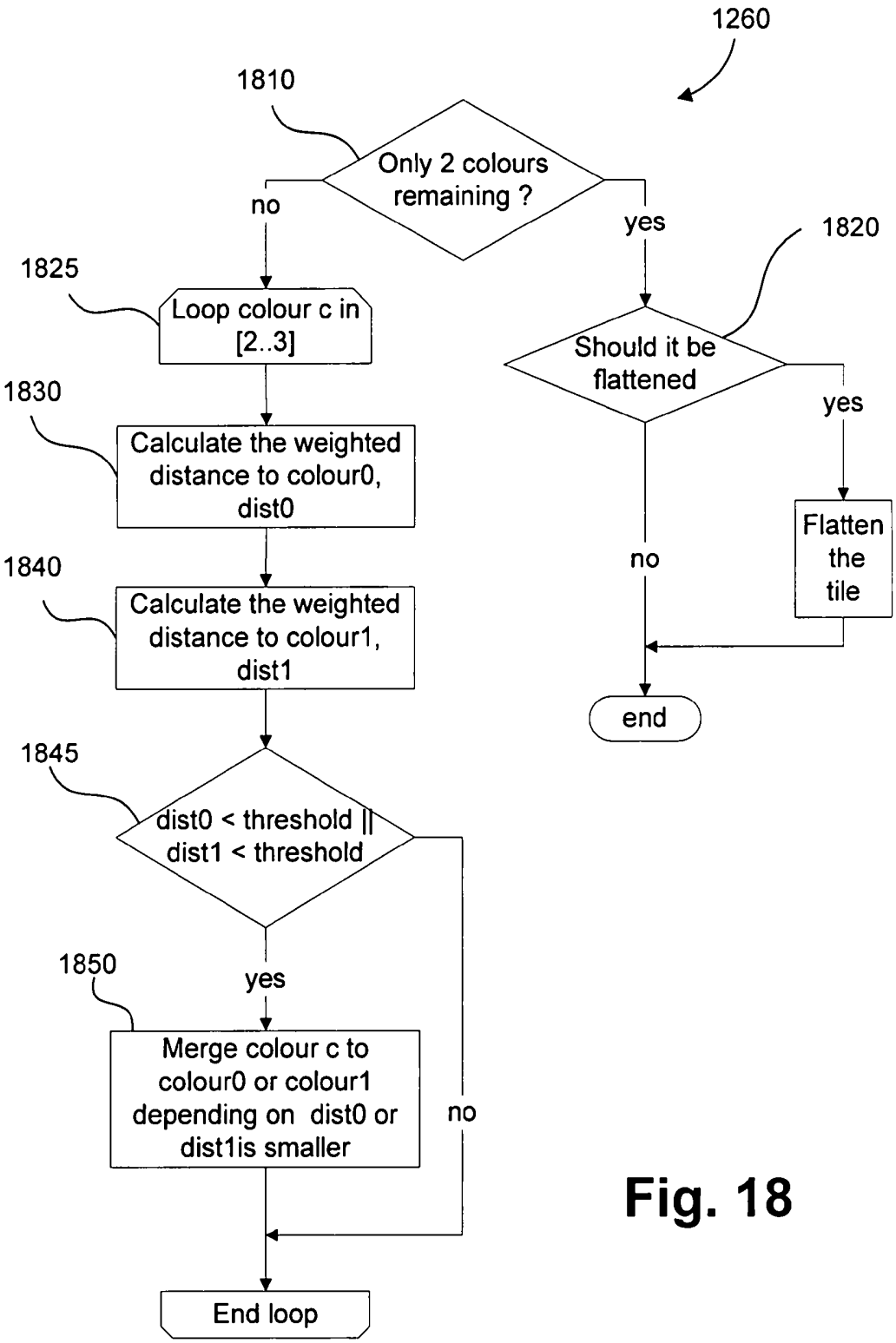


Fig. 18

2007249099 18 Dec 2007

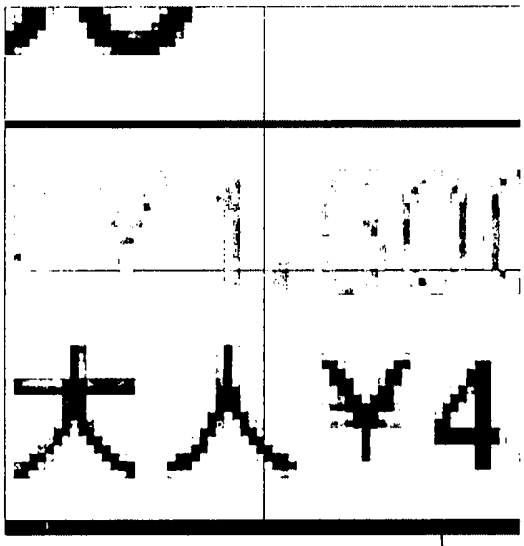


Fig. 19(a)
(Prior Art)

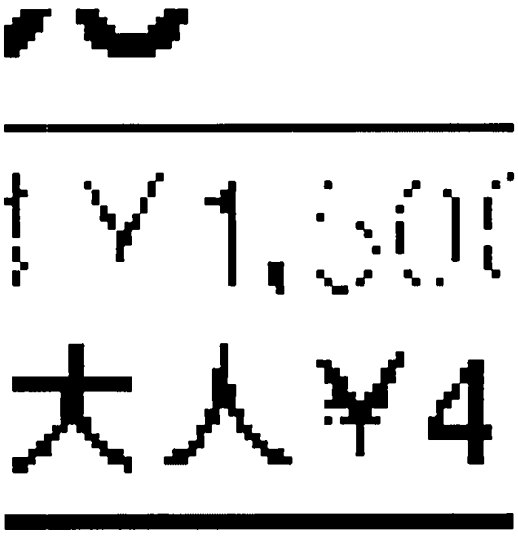


Fig. 19(b)
(Prior Art)

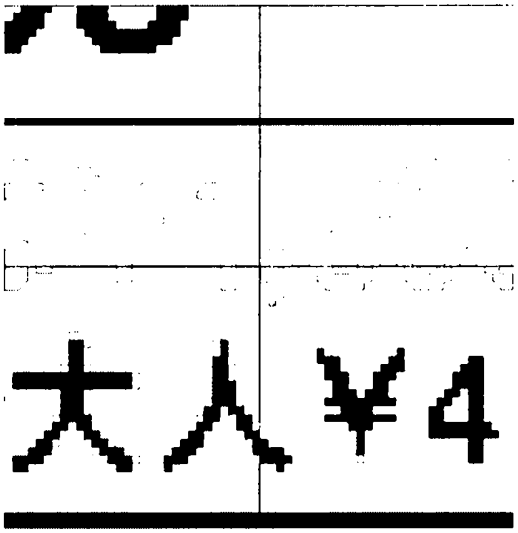


Fig. 19(c)
(Prior Art)

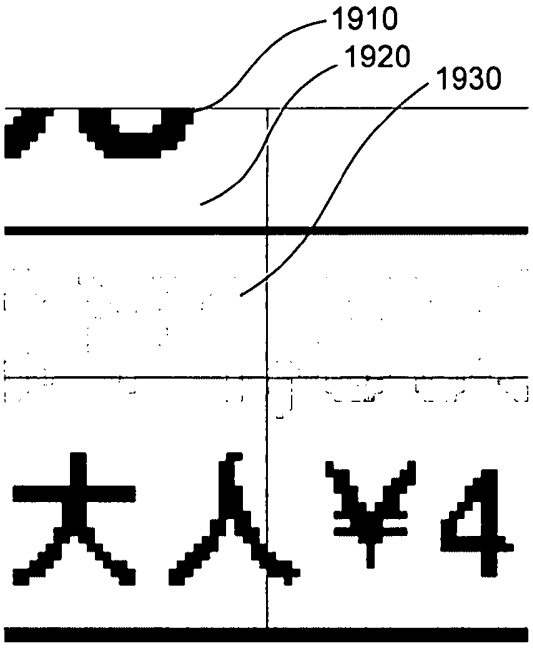
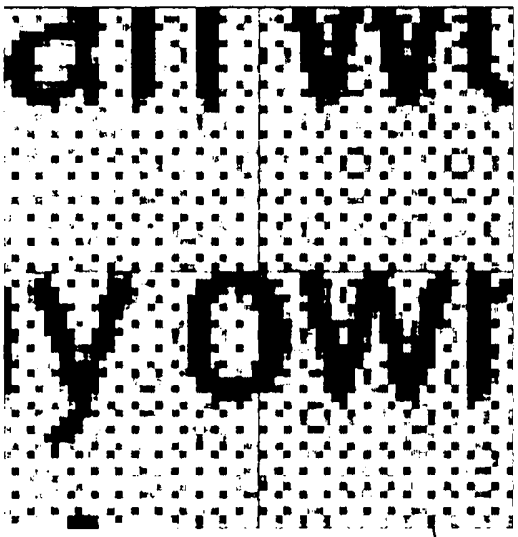


Fig. 19(d)

1900

2007249099 18 Dec 2007



2000

Fig. 20(a)
(Prior Art)

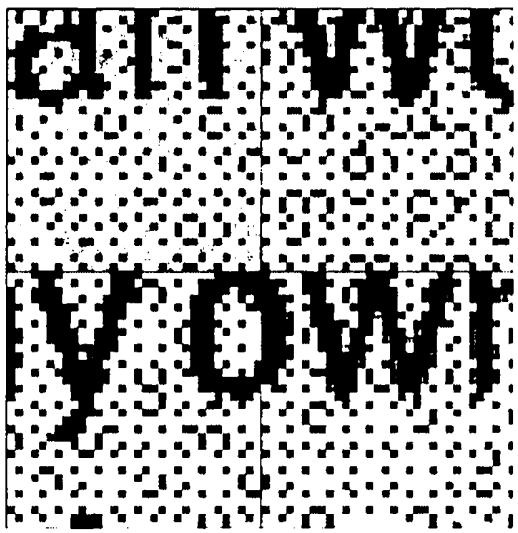


Fig. 20(c)
(Prior Art)

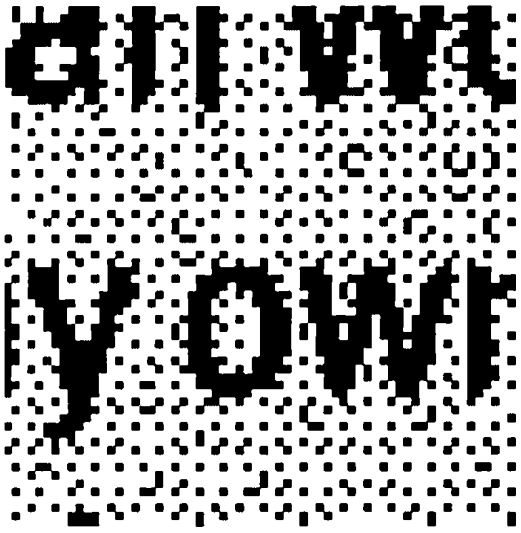


Fig. 20(b)
(Prior Art)

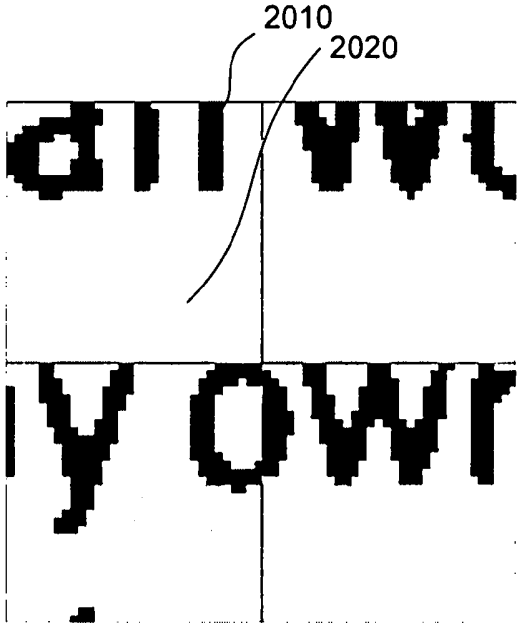


Fig. 20(d)

2007249099 18 Dec 2007

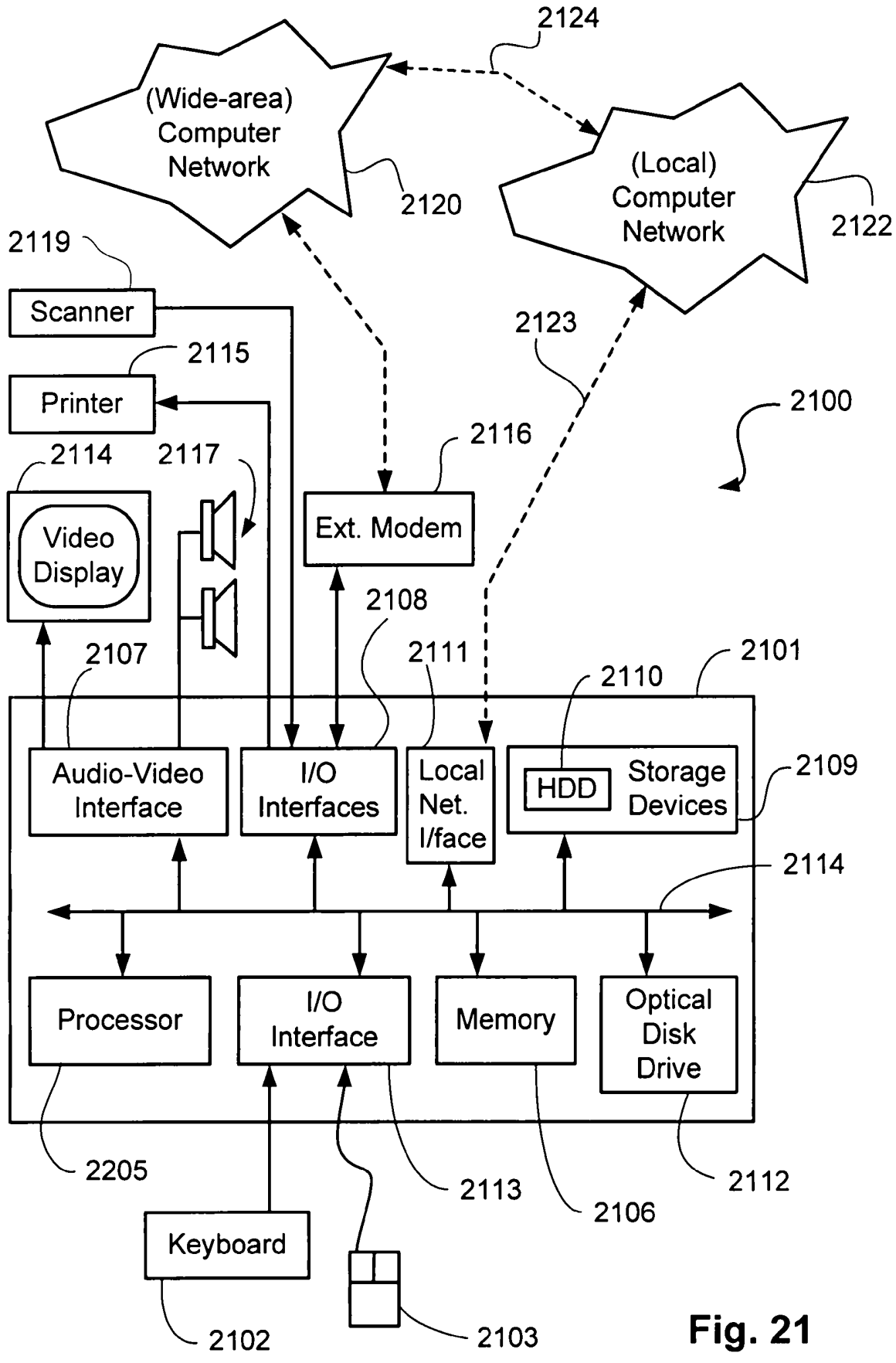


Fig. 21