a threshold detector and changing to white the pixels that don't meet the threshold.

A further understanding of the nature and advantages of the inventions herein may be realized by reference to the remaining portions of the specification and the attached drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a document digitizing system using a defect defection system according to the present invention;

FIG. **2** is a block diagram further detailing the defect detection system shown in FIG. 1;

FIGS. **3**(*a*)–(*c*) graphically depict a defect detection process;

FIG. **4** is a flowchart of a defect detection process according to the present invention;

FIG. **5** illustrates the results of an actual defect detection experiment;

FIG. **6** is a block diagram of an inefficient use of FIFOs;

FIG. **7** is a memory map of a circular stack for storing a defect list;

FIG. **8** is a diagram illustrating an alternate method for detecting defects based on a collection of document images; and

FIG. **9** is a block diagram of a page frame apparatus for detecting both white and black defects automatically.

DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. **1** is a block diagram of a digital document imaging system **10** with defect correction according to the present invention. System **10** includes a scanner **12**, a defect detection system **14**, an image restoration system **16**, and optionally, raw image storage **18**. The output from scanner **12** is a digital image Which is supplied to defect detection system **14**, image restoration system **16** and, if used, storage **18**. The digital image is a digitized input image. The corrected output document image, or images, **22** is supplied by image restoration system **16**. Defect detection system **14** is coupled to image restoration system **16** to provide defect locations.

In operation, defect detection system **14** will typically have accumulated knowledge of the defects present in the scanning mechanism of scanner **12**. Of course, upon first use after reset, reconfiguration or initialization, defect detection system **14** will not know the location of defects, but must accumulate this knowledge over several documents.

Assuming defect detection system **14** has been operating long enough to accumulate knowledge of the defect locations in the scanner **12**, document **20** is scanned by the scanning mechanism of scanner **12**, which supplies the digital image of document **20** to defect detection system **14** and either storage **18** or image restoration system **16**. Defect detection system **14** accumulates and refines knowledge about defect locations based on the digital images it receives. Image restoration system **16** uses this information about the defect locations to automatically retouch the digital image provided from scanner **12** and supply the corrected document image as image **22**. Image **22**, in some embodiments, is an electronically transmitted sequence of digital data, while in other embodiments, image **22** is translated into a printed document **22**. The former embodiments are useful in computer applications, while the latter application is useful in office copier applications where a corrected copy of an original document is the desired output.

FIG. **2** is a block diagram showing defect detection system **14** in further detail. FIG. **2** shows a digital image **30**, a dark pixel identifier **32**, a defect memory controller **34**, cursor registers **38**, defect history buffer **40** and an output block of defect locations **42**. FIG. **2** also shows the interconnections among the various blocks, such as an input of pixel identifier **32** coupled to receive digital image **30** and an output to provide dark pixel locations to controller **34**. Controller **34** reads, modifies and deletes entries from defect history buffer **40** and reads cursor registers **38**. Controller **34** also outputs block **42**. Controller **34** and buffer **40** are shown coupled through two unidirectional data paths **44**, **46**, to highlight the interaction of controller **34** and buffer **40** which results in efficient packing of buffer **40**. This is explained below in connection with FIGS. **6** and **7**.

In general, the operation of defect detection system **14** is as follows. Dark pixel identifier **32** identifies all the dark pixels in digital image **30** and passes their locations to controller **34**. Buffer **40** contains a list of all tentatively defective pixel locations. An entry in the defect list includes an identification of the location of the defect and thus a tentatively defective pixel in a specific image such as digital image **30**. An entry also includes a count of the number of consecutive dark pixels found at that location in a series of documents, and other flags and fields as explained below. Controller **34** increments the count for an entry if the pixel at that location is dark and zeroes the count if the pixel is light or removes the defect from list **40** entirely. In an alternate embodiment, the count is not zeroes upon the occurrence of a fight pixel, but is only reduced. Table 1 shows the contents of a typical defect list.

TABLE 1

Location	Count	Status Flags
l_0	c_0	f_0
l_1	c_1	f_1
l_2	c_2	f_2
...	...	...

Location field values can be stored as scalar position values or as row/column coordinates. For particularly efficient memory usage, the location fields need contain only offsets relative to the location of a prior entry.

In a specific embodiment, buffer **40** holds a maximum of 60,000 tentative defect entries, and each entry is stored in 20 bits of memory. Of the 20 bits assigned to an entry, 15 bits are allocated for storage of the location offset and the other 5 bits are allocated for storage of a status field, from which the count and status flags can be derived. Table 2 shows the meaning of each of the 32 possible status field values.

Each time a document is scanned, the entries are passed from buffer **40** to controller **34** along data path **44**, modified or deleted by controller **34**, and passed back to buffer **40** along data path **46**. This allows buffer **40** to remain packed even while entries are being deleted, as explained in connection with FIGS. **6** and **7**. In some embodiments, the controller merely directs the transfer of data within buffer **40** without it actually moving it to controller **34**.