

Description

The present invention relates to digital image processing methods and more particularly to a method and apparatus for handling encoded image data to rotate the image.

There is a considerable amount of prior art relating to methods for processing encoded image data to achieve rotation of the original image with respect to its orientation upon input to the encoding system. Varied examples are found in U. S. Patent Nos. 3,968,475; 3,976,872; 4,052,699; 4,168,488; 4,225,929; and 4,271,476; as well as in EPO published patent application 081,096 and IBM Technical Disclosure Bulletins, Vol. 13, No. 11, April 1971, page 3267; and Vol. 18, No. 8, January 1976 at pages 2633 and 2640.

In applications where the output image is displayed in raster form the image data to be processed may not be immediately available in storage for readout in raster form, that is, each binary bit may not represent a picture element or pel of the original image. The data may, for example, be stored in a run end or run length form, that is, the binary coding is representative of the transition points between black and white on the original image. In run length coding the binary numbers are indicative of the width of each black and white segment in the line of an image, while the binary numbers in run end coding cumulatively indicate successive points of transition along the line. Such representations may be compacted by a procedure such as the CCITT compression standard. One such compaction scheme is described in copending EP-A-149 124. This method includes both an encoder and a decoder. The decoder may be used to create run end representations of individual lines of an encoded image.

One method of rotating an image encoded in run representation form would be firstly to convert the coded data to raster image form using the method described in copending published patent application EP-A-149 121 and then using either the procedure described in copending U.S. patent application Serial No. 567,214 or the procedure described in copending published patent application EP-A-147 701 to rotate the image data in place. All of the preceding applications are assigned to the same assignee as the present application. While these methods are valuable in themselves, their combination would usually result in a great deal of movement of data, much of which would be unnecessary. A further problem is presented by the fact that the resulting procedure would probably be unsuitable for use on systems which allocate storage in blocks that are not large enough to contain an entire image. Consequently, for typical images, it would be more efficient to combine the steps of converting the image to raster form and rearranging the subunits of the image.

In US-4,271,476 (corresponding to EP-022490) a system and method for effecting orthogonal rotation of the scan direction of a digital raster image representation is disclosed.

This clearly does not address the same problem as that of the present invention, which creates such raster representation in rotated form from run representations of an image.

IBM Technical Disclosure Bulletin, Vol. 27, No 1B June 1984, discloses a method of image rotation by rotating raster patterns. Thereby the raster patterns are first serialized into a single bit string. Thereafter the processor transfers on a bit-by-bit basis the image-representing signals from the serialized bit string into a second serialized bit string having different positions representative of the desired rotation. The second bit string is then utilized for presenting the image in the newly rotated orientation.

In contrast the present invention, by starting from a run representation, eliminates the step of creating a serial bit stream in a first memory area by dumping the data directly into a second memory area in reverse bit order.

The present invention is directed to providing a system for combining the conversion and rearrangement of the run end or run length data and for subsequently reordering the data and rotating the subunits to create the final output images in raster form rotated by 180°, either clockwise or counterclockwise, from the original image orientation.

The prior art cited above neither teaches nor suggests the method according to claim 1 and the apparatus according to claim 4. Further embodiments of the invention are defined in the appended dependent claims.

The present invention solves the problem of taking an image in run representation form and creating a raster image which is rotated by 180° from the orientation of the original image. It particularly involves the conversion of original image data which is stored in run end or run length form to permit it to be divided and stored in discontinuous blocks.

A convenient method of rotating the input image 180° is described wherein the original image is read in run representation form a row at a time and the rows are converted to raster form and placed in storage in reverse order, with the order of the bits in each row also reversed.

The invention, which is defined in the attached claims, is described in detail below with reference to the drawings, in which

FIG. 1 is a flow chart showing the control code for the conversion/180° rotation procedure according to the present invention.

FIG. 2 is a flow chart showing the details of the conversion from run end to raster form for the 180° rotation according to the present invention.

5 Source images of the type to be operated upon with the method and apparatus of the present invention are typically documents or other material containing text or pictures in black and white or in a form that can be converted to black and white. These source images are scanned in photographic or raster form and thresholded, if necessary, to produce bilevel images that undergo suitable encoding, such as in accordance with CCITT standards, as is well known to those skilled in the art. The encoded images are then stored
10 and/or transmitted for decoding and display in, for example, a teleconferencing system where the transmission means may be telephone lines. The received data may be decoded in run end or run length form. To convert this binary run end data to raster form, it is necessary to generate the bits so that each bit in a line of data represents a successive black or white picture element or pel in the output image.

A suitable system for creating a raster image from an image in run end form, as previously indicated, is
15 described in copending application EP-A-149 121. The system therein creates a raster image, and the rotation algorithms described in copending applications EP-A-149 119 and EP-A-147 701 may then be used to rotate the raster image in place or not in place. The present invention is an improvement on these procedures because it eliminates a great deal of the rearrangement of the data and because it may be used to create an image which does not occupy a single large block of storage, but rather may occupy a number
20 of smaller blocks. The latter consideration is important when the procedures are to be implemented on a processor which allocates storage in small chunks, so that it is not practical to obtain a block of contiguous storage large enough to apply one of the rotation functions to the whole image. The present invention takes advantage of the fact that initially the image does not exist in raster form, allowing the process of creating a raster image to do some of the rearrangement required by the rotations.

25 However, even if the rotated image is to occupy a large area of contiguous storage, so that the combination of procedures cited above could be used, it will often be more efficient to apply the system of the invention to perform the two steps simultaneously. For the 180° rotation, this can be done by specifying that the list of output blocks includes one block that is large enough to contain the entire image.

The preferred embodiments of the invention assume that the run representation form of the original
30 (input) image is contained in a series of buffers, one buffer for each line or row of image data. For the purposes of description, each buffer will be assumed to contain a series of halfwords containing the run end data. It will be apparent to those skilled in the art that a run length representation could be used with minor changes to the procedure as described. The 16 bits in a halfword of run end data are adequate to represent images with widths up to 32K as positive displacements. The first halfword in each line buffer gives the
35 number of bytes of run end data plus two bytes for the count (i.e. the total number of bytes of data in the buffer); this is followed by three zero run ends, an arbitrary number of pairs of white/black run ends, and two additional copies of the last black run end. If the image row begins with a black run, the first white run end must be specified as zero. If the image row ends with a white run, the last black run end will be the same as the last white run end, so that there are three copies of the last real run end. For example, the run
40 end buffer containing the halfword entries

```

24  0  0  0  0 10 27 28 32 32 32 32
45 16  0  0  0 32 32 32 32
   32  0  0  0  0  1  2  3  8  9 12 28 31 32 32 32

```

would correspond to the following bit image:

```

50
   11111111 11000000 00000000 00010000
   00000000 00000000 00000000 00000000
55 10100000 10001111 11111111 11110001

```

Run end buffers such as those described above may be obtained from various sources; for example, the decoder described in copending application Ser. No. 571,239 creates representations of image lines in this form, as does the method for converting raster data to run representation form described in copending application Ser. No. 567,218. The latter is assigned to the same assignee as the present application. Data may be fed into the procedure of the present invention one line at a time; the present invention does not require multiple lines (in run representation form) of the image to be rotated to be simultaneously available. The conversion of the run representation data to raster form by the present invention may be interleaved with the creation of the run representation data.

The simplest rotation is the 180° rotation. It is possible to perform this operation with an amount of effort roughly equivalent to that used to convert the image to raster form by an appropriate modification to the conversion procedure and a change in the processing order. In the normal conversion of an image in run representation to raster form, a pointer is positioned at the beginning of the storage allocated for the rotated (output) image. The first line of the input image is obtained, and the run end buffer is read from left to right as the runs are filled in in the output image moving from left to right. The pointer then addresses the beginning of the second line of the output image and the next line is converted. Processing continues, reading from the top of the image to the bottom for both the input and output images.

To rotate by 180° while converting to raster, parts of this process are reversed. A pointer is positioned at the end of the storage allocated for the output image. The first line of the input image is obtained. The run end buffer is read from left to right, but the runs are filled in in the output image moving from right to left, reversing the pattern of bits in the line described by the run end buffer. The output pointer is then positioned at the end of the next-to-last output line and the second input line is obtained and converted to raster form in the same manner. The process continues, working toward the top of the output image as the bottom of the input image is approached. When converting/rotating an image into multiple blocks of storage, the output pointer must be repositioned whenever it reaches the beginning of a block.

The details of the preferred embodiment of the 180° rotation will now be described with reference to Figures 1 and 2. This embodiment assumes that the output image is to occupy one or more blocks in storage, with each block containing some number of complete image lines and no partial lines. The number of bytes of image data which are to be placed in a particular block is given by the function NB ("number of bytes"). Variables used in the procedure are:

BLKPTR	Pointer to list of block addresses
BLKPT	Pointer addressed by BLKPTR; points to a block
LAST	Pointer to output byte currently being constructed
RPTR	Pointer to white/black run end pair currently being converted
WHITEEND	White run end addressed by RPTR
BLACKEND	Black run end addressed by RPTR
REND	Run end of run being converted
CUM	Output byte being constructed
BITS	Number of image pels (bits) whose values have been determined but not yet stored
LASTBYTE	Output byte addressed by LAST
IC	Number of bits per line (assumed to be a multiple of 8)
T	Temporary value

Two tables, SBLACKS and EBLACKS, are used by the conversion. Each contains eight entries with indices running from zero to seven. The entries, in hexadecimal, are:

SBLACKS:	FF FE FC F8 F0 E0 C0 80
EBLACKS:	00 01 03 07 0F 1F 3F 7F

SBLACKS is used to start a black run after some number of white pels (the white pels are placed in the low-order bits of the output byte because the raster data is constructed moving right to left). It is also used to invert all of the bits except the first (low-order) ones; the index gives the number of bits which are not to be changed. EBLACKS is used to construct the end of a black run; again, the pels in the run occupy the low-order bits in the output byte.

Figure 1 shows the control code for the 180° rotation. The end of the block list, i.e. the address of the block which will contain the last lines of the output image, is addressed, and LAST is positioned to point to the last byte in the area which will contain output data. The output area is zeroed so that it will not be necessary to store zero bytes when long white runs are converted.

The loop which converts lines is then entered. A new run end buffer is obtained. At this point it may be desirable to check the run end buffer to ensure that it describes a line containing the correct number of bits. RPTR is then set to address the first white/black run end pair, which is at an offset of eight bytes into the buffer. The line is converted to raster form, decrementing the pointer LAST as each byte is stored. After the line conversion is complete, LAST is compared to the address of the beginning of the current block. If LAST is greater, then there is space to convert another line into the current block, so the loop repeats. Otherwise it is necessary to obtain another block and reset LAST. A test is made to determine whether all of the output blocks have been filled with image data; if so, then processing is complete. Otherwise BLKPTR is altered to address the previous output block, LAST is reset to point to the last byte in the data area of that block, the data area in the block is cleared, and the loop to convert lines is reentered.

Figure 2 illustrates the process used to convert one line of run end data to reversed raster form. Initially the previous run end, the current output byte, and the bit count are all set to zero. To convert a white run, BITS is incremented by the length of the white run (the difference between WHITEEND and the old black run end REND), and the white run end is placed in REND. If the total bit count is less than 8, then the bits not yet determined are made black (ones) and control passes to the portion of the code which converts a black run. Otherwise, the current output byte is stored and LAST is backed up to address the previous output byte. The bit counter is decremented by 8 for the byte stored. If the bit count is still not less than 8, then the output run must contain one or more white (zero) bytes. Since the output area was cleared before the line conversion began, it is not necessary to store zero bytes; it is necessary only to back LAST up by the appropriate amount to skip over the bytes, and to reduce the bit counter by the number of bits so skipped over. The current output byte is then set up to contain a number of "leading" (i.e. low-order) zeroes determined by the bit counter, with the rest of the bits black.

To convert a black run, BITS is incremented by the length of the black run (the difference between BLACKEND and the old white run end saved in REND). The run end buffer pointer RPTR is incremented to point to the next white/black run end pair after the black run end has been placed in REND. If the bit count is less than 8, then REND is tested to see if the end of the line has been reached. If so, then the bit count is zero (since the line width is assumed to be a multiple of 8 pels), so there is no more data to store and the line conversion is complete. If the end of the line has not been reached, then the output bits not yet determined are inverted so that they become zeroes (white) and control passes to the code which converts a white run.

If the addition of the black run length causes the bit count to be 8 or greater, then the completed output byte in CUM is stored, LAST is backed up to address the next byte, and BITS is decremented by 8 for the eight bits stored. If BITS is still 8 or greater, then some number of black (X'FF') bytes must be placed in the output image. A black byte is stored, LAST is backed up, and BITS is decremented. This cycle repeats until BITS is less than 8. The black run end value is then tested to determine whether the end of the line has been reached; if so, processing is complete. Otherwise, the "leading" bits of the new output byte CUM are made black, and control passes to the code to convert the next white run end.

Claims

1. A method of rotating an image in run end or run length form by 180° from the orientation of an original image to create an image in raster form, characterized by the following steps:
 beginning the 180° rotation operation by positioning pointer means at a first end of an original image stored in one of said run forms and at a second end, opposite to said first end, of a storage means allocated for an output image in raster form;
 storing a plurality of run representations of a line of said original image in a buffer;
 initializing a line of output data whereby all bits in the output area are set to a first logical level;
 converting each run representation in said buffer to a picture element representation in which the bits in said buffer are read in a first direction and the converted run representation bits are filled in in the storage allocated for the output image moving in a second direction opposite to said first direction;
 storing said picture representation in raster form of said line;
 repeating the steps of storing, initializing, converting and storing for each of the remaining lines of said original image, progressing from said first end of said original image to said second end and from said second end of the output image to said first end, until said original image has been converted to a bit map representation rotated 180° in the storage means allocated to the output image.
2. A method as claimed in Claim 1, in which said initializing step includes the step of testing an output pointer to determine whether a discontinuity in the output image storage area has been reached, and in



The loop which converts lines is then entered. A new run end buffer is obtained. At this point it may be desirable to check the run end buffer to ensure that it describes a line containing the correct number of bits. RPTR is then set to address the first white/black run end pair, which is at an offset of eight bytes into the buffer. The line is converted to raster form, decrementing the pointer LAST as each byte is stored. After the line conversion is complete, LAST is compared to the address of the beginning of the current block. If LAST is greater, then there is space to convert another line into the current block, so the loop repeats. Otherwise it is necessary to obtain another block and reset LAST. A test is made to determine whether all of the output blocks have been filled with image data; if so, then processing is complete. Otherwise BLKPTR is altered to address the previous output block, LAST is reset to point to the last byte in the data area of that block, the data area in the block is cleared, and the loop to convert lines is reentered.

Figure 2 illustrates the process used to convert one line of run end data to reversed raster form. Initially the previous run end, the current output byte, and the bit count are all set to zero. To convert a white run, BITS is incremented by the length of the white run (the difference between WHITEEND and the old black run end REND), and the white run end is placed in REND. If the total bit count is less than 8, then the bits not yet determined are made black (ones) and control passes to the portion of the code which converts a black run. Otherwise, the current output byte is stored and LAST is backed up to address the previous output byte. The bit counter is decremented by 8 for the byte stored. If the bit count is still not less than 8, then the output run must contain one or more white (zero) bytes. Since the output area was cleared before the line conversion began, it is not necessary to store zero bytes; it is necessary only to back LAST up by the appropriate amount to skip over the bytes, and to reduce the bit counter by the number of bits so skipped over. The current output byte is then set up to contain a number of "leading" (i.e. low-order) zeroes determined by the bit counter, with the rest of the bits black.

To convert a black run, BITS is incremented by the length of the black run (the difference between BLACKEND and the old white run end saved in REND). The run end buffer pointer RPTR is incremented to point to the next white/black run end pair after the black run end has been placed in REND. If the bit count is less than 8, then REND is tested to see if the end of the line has been reached. If so, then the bit count is zero (since the line width is assumed to be a multiple of 8 pels), so there is no more data to store and the line conversion is complete. If the end of the line has not been reached, then the output bits not yet determined are inverted so that they become zeroes (white) and control passes to the code which converts a white run.

If the addition of the black run length causes the bit count to be 8 or greater, then the completed output byte in CUM is stored, LAST is backed up to address the next byte, and BITS is decremented by 8 for the eight bits stored. If BITS is still 8 or greater, then some number of black (X'FF') bytes must be placed in the output image. A black byte is stored, LAST is backed up, and BITS is decremented. This cycle repeats until BITS is less than 8. The black run end value is then tested to determine whether the end of the line has been reached; if so, processing is complete. Otherwise, the "leading" bits of the new output byte CUM are made black, and control passes to the code to convert the next white run end.

Claims

1. A method of rotating an image in run end or run length form by 180° from the orientation of an original image to create an image in raster form, characterized by the following steps:
beginning the 180° rotation operation by positioning pointer means at a first end of an original image stored in one of said run forms and at a second end, opposite to said first end, of a storage means allocated for an output image in raster form;
storing a plurality of run representations of a line of said original image in a buffer;
initializing a line of output data whereby all bits in the output area are set to a first logical level;
converting each run representation in said buffer to a picture element representation in which the bits in said buffer are read in a first direction and the converted run representation bits are filled in the storage allocated for the output image moving in a second direction opposite to said first direction;
storing said picture representation in raster form of said line;
repeating the steps of storing, initializing, converting and storing for each of the remaining lines of said original image, progressing from said first end of said original image to said second end and from said second end of the output image to said first end, until said original image has been converted to a bit map representation rotated 180° in the storage means allocated to the output image.
2. A method as claimed in Claim 1, in which said initializing step includes the step of testing an output pointer to determine whether a discontinuity in the output image storage area has been reached, and in



that case resetting said pointer to address the next section of the output image area.

3. A method as claimed in Claim 1 or 2, in which said first end of said original image is the upper left hand corner of said original image and said second end of the output image area is the lower right hand corner of said output image area.
4. An apparatus for rotating an image in run end or run length form by 180° from the orientation of an original image to create an image in raster form, comprising
 buffer means for containing run representation data of an original image;
 means for providing an output image area;
 pointer means positionable at a first end of an original image stored in one of said run forms and at a second end, opposite to said first end, of a storage means allocated for an output image in raster form;
 means for storing a plurality of run representations of a line of said original image in said buffer means;
 means for initializing a line of output data whereby all bits in the output area are set to a first logical level;
 means for converting each run representation in said buffer to a picture element representation in which the bits in said buffer are read in a first direction and the converted run representation bits are filled in in the storage allocated for the output image moving in a second direction opposite to said first direction;
 means for storing said picture representation in raster form of said line; and
 means for repeating the steps of storing, initializing, converting and storing for each of the remaining lines of said original image, progressing from said first end of said original image to said second end and from said second end of the output image to said first end, until said original image has been converted to a bit map representation rotated 180° in the storage means allocated to the output image.
5. An apparatus as claimed in Claim 4, in which said means for initializing comprises means for setting all bits in the output area to zero.
6. An apparatus as claimed in Claim 4, in which said means for initializing comprises means for testing an output pointer to determine whether a discontinuity in the output image storage area has been reached, and in that case resetting said pointer to address the next section of the output image area.
7. An apparatus as claimed in Claim 4, 5 or 6, in which said first end of said original image is the upper left hand corner of said original image and said second end of the output image area is the lower right hand corner of said output image area.

Patentansprüche

1. Verfahren zum Drehen eines Bildes in Laufende- oder Lauflängendarstellung um 180° zu der Ausrichtung eines ursprünglichen Bildes, um ein Bild in Rasterdarstellung zu erzeugen, **dadurch gekennzeichnet, daß** es die folgenden Schritte aufweist:
 Beginnen der 180°-Drehung durch Positionieren von Zeigermitteln an ein erstes Ende eines ursprünglichen Bildes, das in einer der Laufdarstellungen gespeichert ist, und an ein zweites Ende, das dem ersten gegenüberliegt, in einem Speichermittel, dem das Ausgabebild in Rasterdarstellung zugeordnet ist;
 Speichern einer Mehrzahl von Laufdarstellungen einer Linie des ursprünglichen Bildes in einem Buffer;
 Initialisieren einer Linie mit Ausgabedaten, wodurch alle Bits des Ausgabebereichs an einen ersten logischen Pegel gesetzt werden;
 Umwandeln jeder Laufdarstellung in dem Buffer in eine Bildelementdarstellung, bei welcher die Bits in dem Buffer in einer ersten Richtung gelesen werden und die umgewandelten Laufdarstellungsbits durch Bewegen in eine zweite Richtung, die der ersten entgegengesetzt ist, in den Speicher gefüllt werden, dem das Ausgabebild zugeordnet ist;
 Speichern der Bilddarstellung der Linie in Rasterdarstellung;
 Wiederholen der Schritte zum Speichern, Initialisieren, Umwandeln und Speichern jeder der restlichen Linien des ursprünglichen Bildes, Fortschreiten von dem ersten Ende des ursprünglichen Bildes zu dem zweiten Ende und von dem zweiten Ende des Ausgabebildes zu dem ersten Ende, bis das ursprüngliche Bild in eine Bitmap-Darstellung umgewandelt wurde, die in dem Speichermittel, dem das Ausgabebild zugeordnet ist, um 180° gedreht ist.



2. Verfahren nach Anspruch 1, bei welchem der Initialisierungsschritt den Schritt zum Testen eines Ausgabezeigers enthält, um zu bestimmen, ob in dem Ausgabebildbereich eine Unterbrechung erreicht wurde, und in diesem Fall, Rücksetzen des Zeigers, um den nächsten Abschnitt des Ausgabebildbereichs zu adressieren.
- 5 3. Verfahren nach Anspruch 1 oder 2, bei welchem das erste Ende des ursprünglichen Bildes die linke obere Ecke des ursprünglichen Bildes und das zweite Ende des Ausgabebildbereichs die rechte untere Ecke des Ausgabebildbereichs ist.
- 10 4. Einrichtung zum Drehen eines Bildes in Laufende- oder Lauflängendarstellung um 180° von der Ausrichtung eines ursprünglichen Bildes, um ein Bild in Rasterdarstellung zu erzeugen, die aufweist:
 - Buffermittel, die Laufdarstellungsdaten eines ursprünglichen Bildes enthalten;
 - Mittel zum Liefern eines Ausgabebildbereichs;
 - 15 Zeigermittel, die an einem ersten Ende eines ursprünglichen Bildes, das in einer der Laufdarstellungen gespeichert ist, und an einem zweiten Ende, das dem ersten gegenüber liegt, in einem Speichermittel positioniert werden können, dem das Ausgabebild in Rasterdarstellung zugeordnet ist;
 - Mittel zum Speichern einer Mehrzahl von Laufdarstellungen einer Linie des ursprünglichen Bildes in einem Buffer;
 - 20 Mittel zum Initialisieren einer Linie mit Ausgabedaten, wodurch alle Bits des Ausgabebereichs an einen ersten logischen Pegel gesetzt werden;
 - Mittel zum Umwandeln jeder Laufdarstellung in dem Buffer in eine Bildelementdarstellung, bei welcher die Bits in dem Buffer in einer ersten Richtung gelesen werden und die umgewandelten Laufdarstellungsbits durch Bewegen in eine zweite Richtung, die der ersten entgegengesetzt ist, in den Speicher gefüllt werden, dem das Ausgabebild zugeordnet ist;
 - 25 Mittel zum Speichern der Bilddarstellung der Linie in Rasterdarstellung und
 - Mittel zum Wiederholen der Schritte zum Speichern, Initialisieren, Umwandeln und Speichern jeder der restlichen Linien des ursprünglichen Bildes, Fortschreiten von dem ersten Ende des ursprünglichen Bildes zu dem zweiten Ende und von dem zweiten Ende des Ausgabebildes zu dem ersten Ende, bis das ursprüngliche Bild in eine Bitmap-Darstellung umgewandelt wurde, die in dem Speichermittel, dem
 - 30 das Ausgabebild zugeordnet ist, um 180° gedreht ist.
5. Einrichtung nach Anspruch 4, bei welcher das Mittel zum Initialisieren Mittel zum Setzen aller Bits in dem Ausgabebereich auf Null aufweist.
- 35 6. Einrichtung nach Anspruch 4, bei welcher das Mittel zum Initialisieren Mittel zum Testen eines Ausgabezeigers enthält, um zu bestimmen, ob in dem Ausgabebildbereich eine Unterbrechung erreicht wurde, und in diesem Fall, Rücksetzen des Zeigers, um den nächsten Abschnitt des Ausgabebildbereichs zu adressieren.
- 40 7. Einrichtung nach Anspruch 4, 5 oder 6, bei welchem das erste Ende des ursprünglichen Bildes die linke obere Ecke des ursprünglichen Bildes ist und das zweite Ende des Ausgabebildbereichs die rechte untere Ecke des Ausgabebildbereichs ist.

Revendications

- 45 1. Méthode de rotation d'une image sous formes de longueur de bloc ou de fin de bloc sur 180° à partir de l'orientation d'une image originale pour créer une image sous forme de trame, caractérisée par les étapes suivantes:
 - commencer l'opération de rotation de 180° en positionnant un moyen pointeur à une première
 - 50 extrémité d'une image originale emmagasinée dans une desdites formes et à une deuxième extrémité, opposée à ladite première extrémité, d'un moyen d'emmagasinage attribué à une image de sortie sous forme de trame;
 - emmagasiner une pluralité de représentations de blocs d'une ligne de ladite image originale dans un dispositif tampon;
 - 55 initialiser une ligne de données de sortie par où tous les bits de la zone de sortie sont instaurés à un premier niveau logique;
 - convertir chaque représentation de blocs dans ledit dispositif tampon en une représentation d'éléments d'image dans laquelle les bits dans ledit dispositif tampon sont lus dans une première



direction et les bits de représentation de blocs convertis sont insérés dans la mémoire attribuée à l'image de sortie se déplaçant dans une deuxième direction opposée à ladite première direction;

emmagasiner ladite représentation d'image sous forme de trame de ladite ligne;

répéter les étapes d'emmagasiner, initialiser, convertir et emmagasiner pour chacune des lignes restantes de ladite image originale, en progressant de ladite première extrémité de ladite image originale jusqu'à ladite deuxième extrémité et de ladite deuxième extrémité de l'image de sortie jusqu'à ladite première extrémité, jusqu'à ce que ladite image originale ait été convertie en une représentation de mappe binaire soumise à rotation sur 180° dans le moyen d'emmagasiner attribué à l'image de sortie.

2. Méthode selon la revendication 1, dans laquelle ladite étape d'initialiser comprend l'étape de tester un pointeur de sortie pour déterminer si une discontinuité dans la zone d'emmagasiner a été atteinte et, dans ce cas, restaurer ledit pointeur pour adresser la section suivante de la zone d'image de sortie.

3. Méthode selon les revendications 1 ou 2, dans laquelle ladite première extrémité de ladite image originale est le coin supérieur gauche de ladite image originale et ladite deuxième extrémité de la zone d'image de sortie est le coin inférieur droit de ladite zone d'image de sortie.

4. Appareil de rotation d'une image sous formes de longueur de bloc ou de fin de bloc sur 180° à partir de l'orientation d'une image originale pour créer une image sous forme de trame, comprenant:

des moyens tampons pour contenir des données de représentation de blocs d'une image originale;

des moyens pour fournir une zone d'image de sortie;

des moyens pointeurs pouvant être positionnés à une première extrémité d'une image originale emmagasinée dans une desdites formes et à une deuxième extrémité, opposée à ladite première extrémité, d'un moyen d'emmagasiner attribué à une image de sortie sous forme de trame;

des moyens pour emmagasiner une pluralité de représentations de blocs d'une ligne de ladite image originale dans lesdits moyens tampons;

des moyens pour initialiser une ligne de données de sortie par où tous les bits de la zone de sortie sont instaurés à un premier niveau logique;

des moyens pour convertir chaque représentation de blocs dans ledit tampon en une représentation d'éléments d'image dans laquelle les bits dans ledit tampon sont lus dans une première direction et les bits de représentation de blocs convertis sont insérés dans le moyen d'emmagasiner attribué à l'image de sortie se déplaçant dans une deuxième direction opposée à ladite première direction;

des moyens pour emmagasiner ladite représentation d'image sous forme de trame de ladite ligne;

et
des moyens pour répéter les étapes d'emmagasiner, initialiser, convertir et emmagasiner pour chacune des lignes restantes de ladite image originale, en progressant de ladite première extrémité de ladite image originale jusqu'à la deuxième extrémité et de ladite deuxième extrémité de l'image de sortie jusqu'à ladite première extrémité, jusqu'à ce que ladite image originale ait été convertie en une représentation de mappe binaire soumise à rotation sur 180° dans le moyen d'emmagasiner attribué à l'image de sortie.

5. Appareil selon la revendication 4, dans lequel lesdits moyens d'initialiser comprennent des moyens pour instaurer à zéro tous les bits dans la zone de sortie.

6. Appareil selon la revendication 4, dans lequel lesdits moyens pour initialiser comprennent des moyens pour tester un pointeur de sortie pour déterminer si une discontinuité dans l'image de sortie a été atteinte et, dans ce cas, restaurer ledit pointeur pour adresser la section suivante de la zone d'image de sortie.

7. Appareil selon les revendications 4, 5 ou 6, dans lequel ladite première extrémité de ladite image originale est le coin supérieur gauche de ladite image originale et ladite deuxième extrémité de la zone d'image de sortie est le coin inférieur droit de ladite zone d'image de sortie.