**(12) STANDARD PATENT APPLICATION** (11) Application No. **AU 2004242549 A1**
**(19) AUSTRALIAN PATENT OFFICE**

(54) Title
**Enhanced approach of m-array decoding and error correction**

(51)[7] International Patent Classification(s)
**G06K 009/70** **G06F 003/033**

(21) Application No: **2004242549** (22) Date of Filing: **2004.12.31**

(30) Priority Data

| (31) Number | (32) Date | (33) Country |
|---|---|---|
| **10752109** | **2004.01.06** | **US** |

(43) Publication Date: **2005.07.21**
(43) Publication Journal Date: **2005.07.21**

(71) Applicant(s)
**Microsoft Corporation**

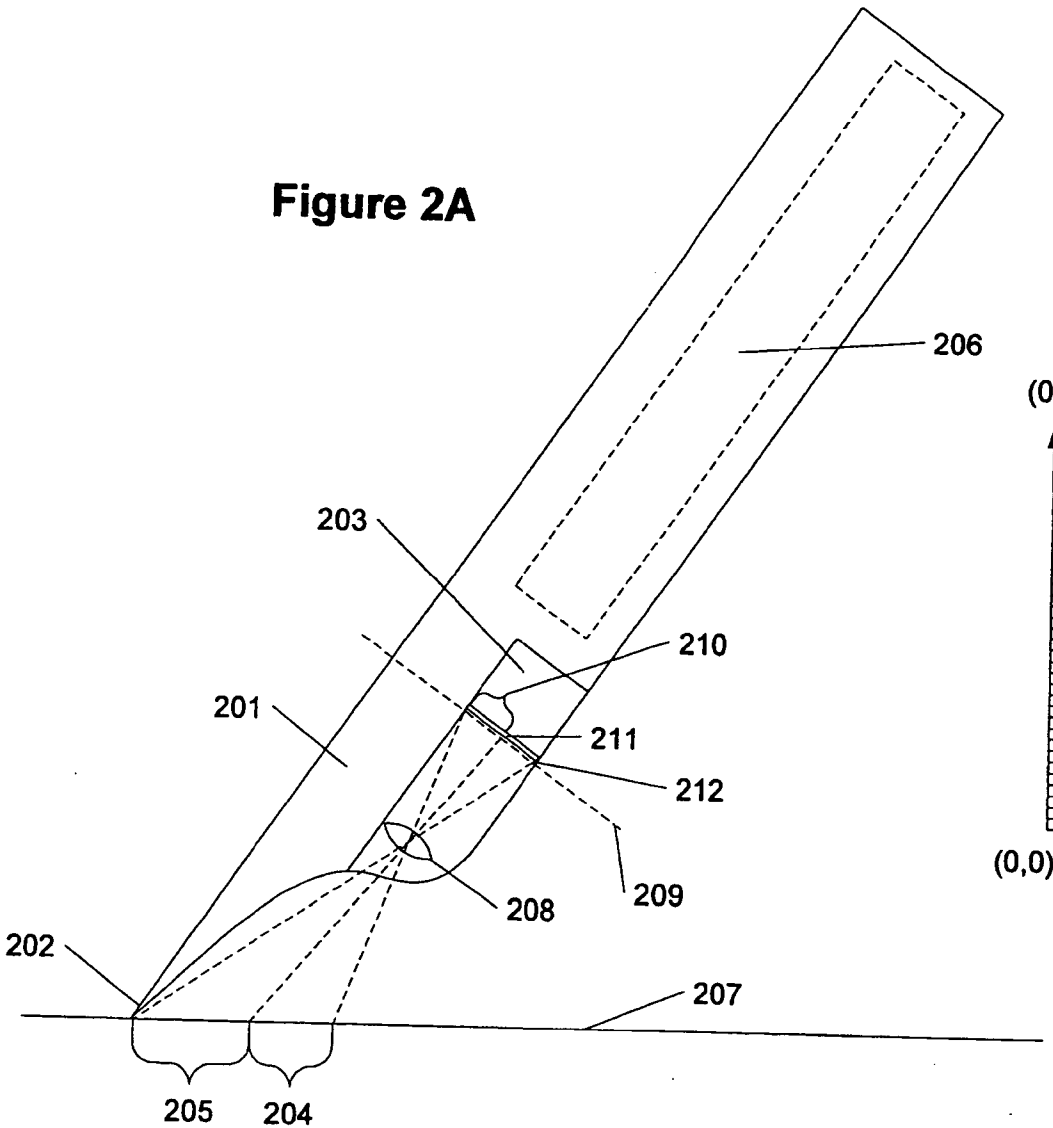
(72) Inventor(s)
**Ma, Xiaoxu; Wang, Qiang; Wang, Jian; Lin, Zhouchen; Li, Yue**

(74) Agent / Attorney
**Davies Collison Cave, 1 Nicholson Street, MELBOURNE, VIC, 3000**

## ABSTRACT

A process and apparatus for determining the location of a captured array from a larger image is described. A non-repeating sequence may be folded into a non-repeating array in which the array is unique for every neighboring window of a given size. A portion of the array of the neighboring window may be captured and a subset of extracted bits corresponding to the captured array is decoded to identify error bits. The location of the captured array is determined within the non-repeating array by further processing the decoded bits.

Figure 2A



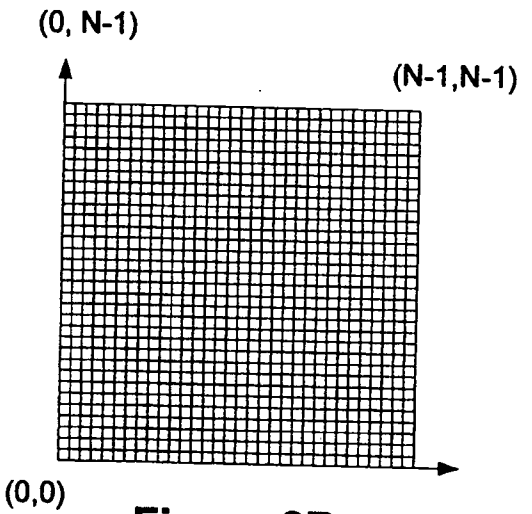




Figure 2B

2004242549 31 Dec 2004

**AUSTRALIA**

PATENTS ACT 1990

**COMPLETE SPECIFICATION**

NAME OF APPLICANT(S)::

**Microsoft Corporation**

ADDRESS FOR SERVICE:

**DAVIES COLLISON CAVE**
Patent Attorneys
1 Nicholson Street,Melbourne, 3000, Australia

INVENTION TITLE:

Enhanced approach of m-array decoding and error correction

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

## Technical Field

[01] The present invention relates to interacting with a medium using a digital pen. More particularly, the present invention relates to determining the location of a digital pen during interaction with one or more surfaces.

## Background

[02] Computer users are accustomed to using a mouse and keyboard as a way of interacting with a personal computer. While personal computers provide a number of advantages over written documents, most users continue to perform certain functions using printed paper. Some of these functions include reading and annotating written documents. In the case of annotations, the printed document assumes a greater significance because of the annotations placed on it by the user. One of the difficulties, however, with having a printed document with annotations is the later need to have the annotations entered back into the electronic form of the document. This requires the original user or another user to wade through the annotations and enter them into a personal computer. In some cases, a user will scan in the annotations and the original text, thereby creating a new document. These multiple steps make the interaction between the printed document and the electronic version of the document difficult to handle on a repeated basis. Further, scanned-in images are frequently non-modifiable. There may be no way to separate the annotations from the original text. This makes using the annotations difficult. Accordingly, an improved way of handling annotations is needed.

[03] One technique of capturing handwritten information is by using a pen whose location may be determined during writing. One pen that provides this capability is the Anoto pen by Anoto Inc. This pen functions by using a camera to capture an image of paper encoded with a predefined pattern. An example of the image pattern is shown in Figure 11. This pattern is used by the Anoto pen (by Anoto Inc.) to determine a location of a pen on a

piece of paper. However, it is unclear how efficient the determination of the location is with the system used by the Anoto pen. To provide efficient determination of the location of the captured image, a system is needed that provides efficient decoding of the captured image.

## Summary

[04] Aspects of the present invention seek to provide solutions to at least one of the issues mentioned above, or at least provide a useful alternative, and may enable one to locate a position or positions of the captured image on a viewed document with a predefined pattern. The viewed document may be on paper, LCD screen or any other medium with the predefined pattern. Aspects of the present invention include a decoding process that permits efficient decoding of a captured image, providing for efficient determination of the location of the image.

[05] With one aspect of the invention, a decoding process tactfully selects a subset of bits from bits extracted from the captured image. With another aspect of the invention, a process adjusts the number of iterations that the decoding process executes. With another aspect of the invention, a process determines the X,Y coordinates of the location of the extracted bits so that the X,Y coordinates are consistent with a local constraint such as a destination area. These and other aspects of the present invention will become known through the following drawings and associated description.

## Brief Description of Drawings

[06] The foregoing summary of the invention, as well as the following detailed description of preferred embodiments, is better understood when read in conjunction with the accompanying drawings, which are included by way of example, and not by way of limitation with regard to the claimed invention.

[07] Figure 1 shows a general description of a computer that may be used in conjunction with embodiments of the present invention.

[08] Figures 2A and 2B show an image capture system and corresponding captured image in accordance with embodiments of the present invention.

[09] Figures 3A through 3F show various sequences and folding techniques in accordance with embodiments of the present invention.

[10] Figures 4A through 4E show various encoding systems in accordance with embodiments of the present invention.

[11] Figures 5A through 5D show four possible resultant corners associated with the encoding system according to Figures 4A and 4B.

[12] Figure 6 shows rotation of a captured image portion in accordance with embodiments of the present invention.

[13] Figure 7 shows various angles of rotation used in conjunction with the coding system of Figures 4A through 4E.

[14] Figure 8 shows a process for determining the location of a captured array in accordance with embodiments of the present invention.

[15] Figure 9 shows a method for determining the location of a captured image in accordance with embodiments of the present invention.

[16] Figure 10 shows another method for determining the location of captured image in accordance with embodiments of the present invention.

[17] Figure 11 shows a representation of encoding space in a document according to prior art.

[18] Figure 12 shows a flow diagram for decoding extracted bits from a captured image in accordance with embodiments of the present invention.

[19] Figure 13 shows bit selection of extracted bits from a captured image in accordance with embodiments of the present invention.

[20] Figure 14 shows an apparatus for decoding extracted bits from a captured image in accordance with embodiments of the present invention.

## Detailed Description

[21] Aspects of the present invention relate to determining the location of a captured image in relation to a larger image. The location determination method and system described herein may be used in combination with a multi-function pen.

[22] The following is separated by subheadings for the benefit of the reader. The subheadings include: terms, general-purpose computer, image capturing pen, encoding of array, decoding, error correction, and location determination.

### *Terms*

[23] Pen - any writing implement that may or may not include the ability to store ink. In some examples, a stylus with no ink capability may be used as a pen in accordance with embodiments of the present invention.

[24] Camera - an image capture system that may capture an image from paper or any other medium.

### *General Purpose Computer*

[25] Figure 1 is a functional block diagram of an example of a conventional general-purpose digital computing environment that can be used to implement various aspects of the present invention. In Figure 1, a computer 100 includes a processing unit 110, a system memory 120, and a system bus 130 that couples various system components including the system memory to the processing unit 110. The system bus 130 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus,

and a local bus using any of a variety of bus architectures. The system memory 120 includes read only memory (ROM) 140 and random access memory (RAM) 150.

[26] A basic input/output system 160 (BIOS), containing the basic routines that help to transfer information between elements within the computer 100, such as during start-up, is stored in the ROM 140. The computer 100 also includes a hard disk drive 170 for reading from and writing to a hard disk (not shown), a magnetic disk drive 180 for reading from or writing to a removable magnetic disk 190, and an optical disk drive 191 for reading from or writing to a removable optical disk 192 such as a CD ROM or other optical media. The hard disk drive 170, magnetic disk drive 180, and optical disk drive 191 are connected to the system bus 130 by a hard disk drive interface 192, a magnetic disk drive interface 193, and an optical disk drive interface 194, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for the personal computer 100. It will be appreciated by those skilled in the art that other types of computer readable media that can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROMs), and the like, may also be used in the example operating environment.

[27] A number of program modules can be stored on the hard disk drive 170, magnetic disk 190, optical disk 192, ROM 140 or RAM 150, including an operating system 195, one or more application programs 196, other program modules 197, and program data 198. A user can enter commands and information into the computer 100 through input devices such as a keyboard 101 and pointing device 102. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner or the like. These and other input devices are often connected to the processing unit 110 through a serial port interface 106 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). Further still, these devices may be coupled directly to the system bus 130 via an appropriate interface (not shown). A monitor 107 or other type of display device is also connected to the system

bus 130 via an interface, such as a video adapter 108. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers. In a preferred embodiment, a pen digitizer 165 and accompanying pen or stylus 166 are provided in order to digitally capture freehand input. Although a direct connection between the pen digitizer 165 and the serial port is shown, in practice, the pen digitizer 165 may be coupled to the processing unit 110 directly, via a parallel port or other interface and the system bus 130 as known in the art. Furthermore, although the digitizer 165 is shown apart from the monitor 107, it is preferred that the usable input area of the digitizer 165 be co-extensive with the display area of the monitor 107. Further still, the digitizer 165 may be integrated in the monitor 107, or may exist as a separate device overlaying or otherwise appended to the monitor 107.

[28] The computer 100 can operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 109. The remote computer 109 can be a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 100, although only a memory storage device 111 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 112 and a wide area network (WAN) 113. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[29] When used in a LAN networking environment, the computer 100 is connected to the local network 112 through a network interface or adapter 114. When used in a WAN networking environment, the personal computer 100 typically includes a modem 115 or other means for establishing a communications over the wide area network 113, such as the Internet. The modem 115, which may be internal or external, is connected to the system bus 130 via the serial port interface 106. In a networked environment, program modules depicted relative to the personal computer 100, or portions thereof, may be stored in the remote memory storage device.

[30] It will be appreciated that the network connections shown are illustrative and other techniques for establishing a communications link between the computers can be used. The existence of any of various well-known protocols such as TCP/IP, Ethernet, FTP, HTTP, Bluetooth, IEEE 802.11x and the like is presumed, and the system can be operated in a client-server configuration to permit a user to retrieve web pages from a web-based server. Any of various conventional web browsers can be used to display and manipulate data on web pages.

### *Image Capturing Pen*

[31] Aspects of the present invention include placing an encoded data stream in a displayed form that represents the encoded data stream. (For example, as will be discussed with Figure 4B, the encoded data stream is used to create a graphical pattern.) The displayed form may be printed paper (or other physical medium) or may be a display projecting the encoded data stream in conjunction with another image or set of images. For example, the encoded data stream may be represented as a physical graphical image on the paper or a graphical image overlying the displayed image (e.g., representing the text of a document) or may be a physical (non-modifiable) graphical image on a display screen (so any image portion captured by a pen is locatable on the display screen).

[32] This determination of the location of a captured image may be used to determine the location of a user's interaction with the paper, medium, or display screen. In some aspects of the present invention, the pen may be an ink pen writing on paper. In other aspects, the pen may be a stylus with the user writing on the surface of a computer display. Any interaction may be provided back to the system with knowledge of the encoded image on the document or supporting the document displayed on the computer screen. By repeatedly capturing images with a camera in the pen or stylus as the pen or stylus traverses a document, the system can track movement of the stylus being controlled by the user. The displayed or printed image may be a watermark associated with the blank or content-rich paper or may be a watermark associated with a displayed image or a fixed coding overlying a screen or built into a screen.

[33] Figures 2A and 2B show an illustrative example of pen 201 with a camera 203. Pen 201 includes a tip 202 that may or may not include an ink reservoir. Camera 203 captures an image 204 from surface 207. Pen 201 may further include additional sensors and/or processors as represented in broken box 206. These sensors and/or processors 206 may also include the ability to transmit information to another pen 201 and/or a personal computer (for example, via Bluetooth or other wireless protocols).

[34] Figure 2B represents an image as viewed by camera 203. In one illustrative example, the field of view of camera 203 (i.e., the resolution of the image sensor of the camera) is 32x32 pixels (where $N$=32). In the embodiment, a captured image (32 pixels by 32 pixels) corresponds to an area of approximately 5 mm by 5 mm of the surface plane captured by camera 203. Accordingly, Figure 2B shows a field of view of 32 pixels long by 32 pixels wide. The size of $N$ is adjustable, such that a larger N corresponds to a higher image resolution. Also, while the field of view of the camera 203 is shown as a square for illustrative purposes here, the field of view may include other shapes as is known in the art.

[35] The images captured by camera 203 may be defined as a sequence of image frames $\{I_i\}$, where $I_i$ is captured by the pen 201 at sampling time $t_i$. The sampling rate may be large or small, depending on system configuration and performance requirement. The size of the captured image frame may be large or small, depending on system configuration and performance requirement.

[36] The image captured by camera 203 may be used directly by the processing system or may undergo pre-filtering. This pre-filtering may occur in pen 201 or may occur outside of pen 201 (for example, in a personal computer).

[37] The image size of Figure 2B is 32x32 pixels. If each encoding unit size is 3x3 pixels, then the number of captured encoded units would be approximately 100 units. If the encoding unit size is 5x5 pixels, then the number of captured encoded units is approximately 36.

[38] Figure 2A also shows the image plane 209 on which an image 210 of the pattern from location 204 is formed. Light received from the pattern on the object plane 207 is focused by lens 208. Lens 208 may be a single lens or a multi-part lens system, but is represented here as a single lens for simplicity. Image capturing sensor 211 captures the image 210.

[39] The image sensor 211 may be large enough to capture the image 210. Alternatively, the image sensor 211 may be large enough to capture an image of the pen tip 202 at location 212. For reference, the image at location 212 is referred to as the virtual pen tip. It is noted that the virtual pen tip location with respect to image sensor 211 is fixed because of the constant relationship between the pen tip, the lens 208, and the image sensor 211.

[40] The following transformation $F_{S\to P}$ transforms position coordinates in the image captured by camera to position coordinates in the real image on the paper:

$$L_{paper} = F_{S\to P}(L_{Sensor})$$

[41] During writing, the pen tip and the paper are on the same plane. Accordingly, the transformation from the virtual pen tip to the real pen tip is also $F_{S\to P}$:

$$L_{pentip} = F_{S\to P}(L_{virtual-pentip})$$

[42] The transformation $F_{S\to P}$ may be estimated as an affine transform. This simplifies as:

$$F'_{S\to P} = \begin{Bmatrix} \dfrac{s_x \sin\theta_y}{\cos\theta_x \sin\theta_y - \cos\theta_y \sin\theta_x}, & -\dfrac{s_x \cos\theta_y}{\cos\theta_x \sin\theta_y - \cos\theta_y \sin\theta_x}, & 0 \\ -\dfrac{s_y \sin\theta_x}{\cos\theta_x \sin\theta_y - \cos\theta_y \sin\theta_x}, & \dfrac{s_y \cos\theta_x}{\cos\theta_x \sin\theta_y - \cos\theta_y \sin\theta_x}, & 0 \\ 0, & 0, & 1 \end{Bmatrix}$$

as the estimation of $F_{S\to P}$, in which $\theta_x$, $\theta_y$, $s_x$, and $s_y$ are the rotation and scale of two orientations of the pattern captured at location 204. Further, one can refine $F'_{S\to P}$ by matching the captured image with the corresponding real image on paper. "Refine"

means to get a more precise estimation of the transformation $F_{S\to P}$ by a type of optimization algorithm referred to as a recursive method. The recursive method treats the matrix $F'_{S\to P}$ as the initial value. The refined estimation describes the transformation between $S$ and $P$ more precisely.

**[43]** Next, one can determine the location of virtual pen tip by calibration.

**[44]** One places the pen tip 202 on a fixed location $L_{pentip}$ on paper. Next, one tilts the pen, allowing the camera 203 to capture a series of images with different pen poses. For each image captured, one may obtain the transformation $F_{S\to P}$. From this transformation, one can obtain the location of the virtual pen tip $L_{virtual-pentip}$:

$$L_{virtual-pentip} = F_{P\to S}(L_{pentip})$$

where $L_{pentip}$ is initialized as (0, 0) and

$$F_{P\to S} = (F_{S\to P})^{-1}$$

**[45]** By averaging the $L_{virtual-pentip}$ obtained from each image, a location of the virtual pen tip $L_{virtual-pentip}$ may be determined. With $L_{virtual-pentip}$, one can get a more accurate estimation of $L_{pentip}$. After several times of iteration, an accurate location of virtual pen tip $L_{virtual-pentip}$ may be determined.

**[46]** The location of the virtual pen tip $L_{virtual-pentip}$ is now known. One can also obtain the transformation $F_{S\to P}$ from the images captured. Finally, one can use this information to determine the location of the real pen tip $L_{pentip}$:

$$L_{pentip} = F_{S\to P}(L_{virtual-pentip})$$

### *Encoding of Array*

[47] A two-dimensional array may be constructed by folding a one-dimensional sequence. Any portion of the two-dimensional array containing a large enough number of bits may be used to determine its location in the complete two-dimensional array. However, it may be necessary to determine the location from a captured image or a few captured images. So as to minimize the possibility of a captured image portion being associated with two or more locations in the two-dimensional array, a non-repeating sequence may be used to create the array. One property of a created sequence is that the sequence does not repeat over a length (or window) $n$. The following describes the creation of the one-dimensional sequence then the folding of the sequence into an array.

## Sequence Construction

[48] A sequence of numbers may be used as the starting point of the encoding system. For example, a sequence (also referred to as an $m$-sequence) may be represented as a q-element set in field $F_q$. Here, $q=p^n$ where $n \geq 1$ and $p$ is a prime number. The sequence or $m$-sequence may be generated by a variety of different techniques including, but not limited to, polynomial division. Using polynomial division, the sequence may be defined as follows:

$$\frac{R_l(x)}{P_n(x)}$$

[49] where $P_n(x)$ is a primitive polynomial of degree $n$ in field $F_q[x]$ (having $q^n$ elements). $R_l(x)$ is a nonzero polynomial of degree $l$ (where $l<n$) in field $F_q[x]$. The sequence may be created using an iterative procedure with two steps: first, dividing the two polynomials (resulting in an element of field $F_q$) and, second, multiplying the remainder by $x$. The computation stops when the output begins to repeat. This process may be implemented using a linear feedback shift register as set forth in an article by Douglas W. Clark and Lih-Jyh Weng, "Maximal and Near-Maximal Shift Register Sequences: Efficient Event Counters and Easy Discrete Logarithms," IEEE Transactions on Computers 43.5 (May

1994, pp 560-568). In this environment, a relationship is established between cyclical shifting of the sequence and polynomial $R_l(x)$: changing $R_l(x)$ only cyclically shifts the sequence and every cyclical shifting corresponds to a polynomial $R_l(x)$. One of the properties of the resulting sequence is that, the sequence has a period of $q^n - 1$ and within a period, over a width (or length) $n$, any portion exists once and only once in the sequence. This is called the "window property". Period $q^n - 1$ is also referred to as the length of the sequence and $n$ as the order of the sequence.

[50] The process described above is but one of a variety of processes that may be used to create a sequence with the window property.

## Array Construction

[51] The array (or $m$-array) that may be used to create the image (of which a portion may be captured by the camera) is an extension of the one-dimensional sequence or $m$-sequence. Let $A$ be an array of period $(m_1, m_2)$, namely $A(k + m_1, l) = A(k, l + m_2) = A(k, l)$. When an $n_1 \times n_2$ window shifts through a period of $A$, all the nonzero $n_1 \times n_2$ matrices over $F_q$ appear once and only once. This property is also referred to as a "window property" in that each window is unique. A widow may then be expressed as an array of period $(m_1, m_2)$ (with $m_1$ and $m_2$ being the horizontal and vertical number of bits present in the array) and order $(n_1, n_2)$.

[52] A binary array (or $m$-array) may be constructed by folding the sequence. One approach is to obtain a sequence then fold it to a size of $m_1 \times m_2$ where the length of the array is $L = m_1 \times m_2 = 2^n - 1$. Alternatively, one may start with a predetermined size of the space that one wants to cover (for example, one sheet of paper, 30 sheets of paper or the size of a computer monitor), determine the area ($m_1 \times m_2$), then use the size to let $L \geq m_1 \times m_2$, where $L = 2^n - 1$.

[53] A variety of different folding techniques may be used. For example, Figures 3A through 3C show three different sequences. Each of these may be folded into the array shown as

Figure 3D. The three different folding methods are shown as the overlay in Figure 3D and as the raster paths in Figures 3E and 3F. We adopt the folding method shown in Figure 3D.

[54] To create the folding method as shown in Figure 3D, one creates a sequence $\{a_i\}$ of length $L$ and order $n$. Next, an array $\{b_{kl}\}$ of size $m_1 \times m_2$, where gcd($m_1$, $m_2$) =1 and $L = m_1 \times m_2$, is created from the sequence $\{a_i\}$ by letting each bit of the array be calculated as shown by equation 1:

$$b_{kl} = a_i \text{ , where } k = i \bmod(m_1), l = i \bmod(m_2), \; i = 0, \cdots, L-1. \qquad (1)$$

[55] This folding approach may be alternatively expressed as laying the sequence on the diagonal of the array, then continuing from the opposite edge when an edge is reached.

[56] Figure 4A shows sample encoding techniques that may be used to encode the array of Figure 3D. It is appreciated that other encoding techniques may be used. For example, an alternative coding technique is shown in Figure 11.

[57] Referring to Figure 4A, a first bit 401 (for example, "1") is represented by a column of dark ink. A second bit 402 (for example, "0") is represented by a row of dark ink. It is appreciated that any color ink may be used to represent the various bits. The only requirement in the color of the ink chosen is that it provides a significant contrast with the background of the medium to be differentiable by an image capture system. The bits in Figure 4A are represented by a 3x3 matrix of cells. The size of the matrix may be modified to be any size as based on the size and resolution of an image capture system. Alternative representation of bits 0 and 1 are shown in Figures 4C-4E. It is appreciated that the representation of a one or a zero for the sample encodings of Figures 4A-4E may be switched without effect. Figure 4C shows bit representations occupying two rows or columns in an interleaved arrangement. Figure 4D shows an alternative arrangement of the pixels in rows and columns in a dashed form. Finally Figure 4E shows pixel

representations in columns and rows in an irregular spacing format (e.g., two dark dots followed by a blank dot).

[58] Referring back to Figure 4A, if a bit is represented by a 3×3 matrix and an imaging system detects a dark row and two white rows in the 3×3 region, then a zero is detected (or one). If an image is detected with a dark column and two white columns, then a one is detected (or a zero).

[59] Here, more than one pixel or dot is used to represent a bit. Using a single pixel (or bit) to represent a bit is fragile. Dust, creases in paper, non-planar surfaces, and the like create difficulties in reading single bit representations of data units. However, it is appreciated that different approaches may be used to graphically represent the array on a surface. Some approaches are shown in Figures 4C through 4E. It is appreciated that other approaches may be used as well. One approach is set forth in Figure 11 using only space-shifted dots.

[60] A bit stream is used to create the graphical pattern 403 of Figure 4B. Graphical pattern 403 includes 12 rows and 18 columns. The rows and columns are formed by a bit stream that is converted into a graphical representation using bit representations 401 and 402. Figure 4B may be viewed as having the following bit representation:

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

### *Decoding*

[61] When a person writes with the pen of Figure 2A or moves the pen close to the encoded pattern, the camera captures an image. For example, pen 201 may utilize a pressure sensor as pen 201 is pressed against paper and pen 201 traverses a document on the paper. The image is then processed to determine the orientation of the captured image

with respect to the complete representation of the encoded image and extract the bits that make up the captured image.

[62] For the determination of the orientation of the captured image relative to the whole encoded area, one may notice that not all the four conceivable corners shown in Figure 5A-5D can present in the graphical pattern 403. In fact, with the correct orientation, the type of corner shown in Figure 5A cannot exist in the graphical pattern 403. Therefore, the orientation in which the type of corner shown in Figure 5A is missing is the right orientation.

[63] Continuing to Figure 6, the image captured by a camera 601 may be analyzed and its orientation determined so as to be interpretable as to the position actually represented by the image 601. First, image 601 is reviewed to determine the angle $\theta$ needed to rotate the image so that the pixels are horizontally and vertically aligned. It is noted that alternative grid alignments are possible including a rotation of the underlying grid to a non-horizontal and vertical arrangement (for example, 45 degrees). Using a non-horizontal and vertical arrangement may provide the probable benefit of eliminating visual distractions from the user, as users may tend to notice horizontal and vertical patterns before others. For purposes of simplicity, the orientation of the grid (horizontal and vertical and any other rotation of the underlying grid) is referred to collectively as the predefined grid orientation.

[64] Next, image 601 is analyzed to determine which corner is missing. The rotation amount $o$ needed to rotate image 601 to an image ready for decoding 603 is shown as $o$ = ($\theta$ plus a rotation amount {defined by which corner missing}). The rotation amount is shown by the equation in Figure 7. Referring back to Figure 6, angle $\theta$ is first determined by the layout of the pixels to arrive at a horizontal and vertical (or other predefined grid orientation) arrangement of the pixels and the image is rotated as shown in 602. An analysis is then conducted to determine the missing corner and the image 602 rotated to the image 603 to set up the image for decoding. Here, the image is rotated 90 degrees

counterclockwise so that image 603 has the correct orientation and can be used for decoding.

[65] It is appreciated that the rotation angle $\theta$ may be applied before or after rotation of the image 601 to account for the missing corner. It is also appreciated that by considering noise in the captured image, all four types of corners may be present. We may count the number of corners of each type and choose the type that has the least number as the corner type that is missing.

[66] Finally, the code in image 603 is read out and correlated with the original bit stream used to create image 403. The correlation may be performed in a number of ways. For example, it may be performed by a recursive approach in which a recovered bit stream is compared against all other bit stream fragments within the original bit stream. Second, a statistical analysis may be performed between the recovered bit stream and the original bit stream, for example, by using a Hamming distance between the two bit streams. It is appreciated that a variety of approaches may be used to determine the location of the recovered bit stream within the original bit stream.

[67] Once one has the recovered bits, one needs to locate the captured image within the original array (for example, the one shown in Figure 4B). The process of determining the location of a segment of bits within the entire array is complicated by a number of items. First, the actual bits to be captured may be obscured (for example, the camera may capture an image with handwriting that obscures the original code). Second, dust, creases, reflections, and the like may also create errors in the captured image. These errors make the localization process more difficult. In this regard, the image capture system may need to function with non-sequential bits extracted from the image. The following represents a method for operating with non-sequential bits from the image.

[68] Let the sequence (or $m$-sequence) $\mathbf{I}$ correspond to the power series $I(x) = 1/P_n(x)$, where $n$ is the order of the $m$-sequence, and the captured image contains $K$ bits of $\mathbf{I}$ $\mathbf{b} = (b_0 \quad b_1 \quad b_2 \quad \cdots \quad b_{K-1})^t$, where $K \geq n$ and the superscript $t$ represents a transpose

of the matrix or vector. The location $s$ of the $K$ bits is just the number of cyclic shifts of $\mathbf{I}$ so that $b_0$ is shifted to the beginning of the sequence. Then this shifted sequence $\mathbf{R}$ corresponds to the power series $x^s / P_n(x)$, or $\mathbf{R} = T^s(\mathbf{I})$, where $T$ is the cyclic shift operator. We find this $s$ indirectly. The polynomials modulo $P_n(x)$ form a field. It is guaranteed that $x^s \equiv r_0 + r_1 x + \cdots r_{n-1} x^{n-1} \bmod(P_n(x))$. Therefore, we may find $(r_0, r_1, \cdots, r_{n-1})$ and then solve for $s$.

[69] The relationship $x^s \equiv r_0 + r_1 x + \cdots r_{n-1} x^{n-1} \bmod(P_n(x))$ implies that $\mathbf{R} = r_0 + r_1 T(\mathbf{I}) + \cdots + r_{n-1} T^{n-1}(\mathbf{I})$. Written in a binary linear equation, it becomes:

$$\mathbf{R} = \mathbf{r}^t \mathbf{A} \tag{2}$$

where $\mathbf{r} = (r_0 \quad r_1 \quad r_2 \quad \cdots \quad r_{n-1})^t$, and $\mathbf{A} = \left(\mathbf{I} \quad T(\mathbf{I}) \quad \cdots \quad T^{n-1}(\mathbf{I})\right)^t$ which consists of the cyclic shifts of $\mathbf{I}$ from 0-shift to (n-1)-shift. Now only sparse $K$ bits are available in $\mathbf{R}$ to solve $\mathbf{r}$. Let the index differences between $b_i$ and $b_0$ in $\mathbf{R}$ be $k_i$, $i = 1,2,\cdots,k-1$, then the 1st and $(k_i + 1)$-th elements of $\mathbf{R}$, $i = 1,2,\cdots,k-1$, are exactly $b_0$, $b_1$, ..., $b_{k-1}$. By selecting the 1st and $(k_i + 1)$-th columns of $\mathbf{A}$, $i = 1,2,\cdots,k-1$, the following binary linear equation is formed:

$$\mathbf{b}^t = \mathbf{r}^t \mathbf{M} \tag{3}$$

where $\mathbf{M}$ is an $n \times K$ sub-matrix of $\mathbf{A}$.

[70] If $\mathbf{b}$ is error-free, the solution of $\mathbf{r}$ may be expressed as:

$$\mathbf{r}^t = \tilde{\mathbf{b}}^t \tilde{\mathbf{M}}^{-1} \tag{4}$$

[71] where $\tilde{\mathbf{M}}$ is any non-degenerate $n \times n$ sub-matrix of $\mathbf{M}$ and $\tilde{\mathbf{b}}$ is the corresponding sub-vector of $\mathbf{b}$.

[72] With known **r**, we may use the Pohlig-Hellman-Silver algorithm as noted by Douglas W. Clark and Lih-Jyh Weng, "Maximal and Near-Maximal Shift Register Sequences: Efficient Event Counters and Easy Discrete Logorithms," IEEE Transactions on Computers 43.5 (May 1994, pp 560-568) to find $s$ so that $x^s \equiv r_0 + r_1 x + \cdots r_{n-1} x^{n-1} \bmod(P_n(x))$.

[73] As matrix **A** (with the size of $n$ by $L$, where $L=2^n - 1$) may be huge, we should avoid storing the entire matrix **A**. In fact, as we have seen in the above process, given extracted bits with index difference $k_i$, only the first and $(k_i + 1)$-th columns of **A** are relevant to the computation. Such choices of $k_i$ is quite limited, given the size of the captured image. Thus, only those columns that may be involved in computation need to saved. The total number of such columns is much smaller than $L$ (where $L=2^n - 1$ is the length of the $m$-sequence).

### *Error Correction*

[74] If errors exist in **b**, then the solution of **r** becomes more complex. Traditional methods of decoding with error correction may not readily apply, because the matrix **M** associated with the captured bits may change from one captured image to another.

[75] We adopt a stochastic approach. Assuming that the number of error bits in **b**, $n_e$, is relatively small compared to $K$, then the probability of choosing correct $n$ bits from the $K$ bits of **b** and the corresponding sub-matrix $\tilde{\mathbf{M}}$ of **M** being non-degenerate is high.

[76] When the n bits chosen are all correct, the Hamming distance between $\mathbf{b}^t$ and $\mathbf{r}^t\mathbf{M}$, or the number of error bits associated with **r**, should be minimal, where **r** is computed via equation (4). Repeating the process for several times, it is likely that the correct **r** that results in the minimal error bits can be identified.

[77] If there is only one **r** that is associated with the minimum number of error bits, then it is regarded as the correct solution. Otherwise, if there is more than one **r** that is associated

with the minimum number of error bits, the probability that $n_e$ exceeds the error correcting ability of the code generated by **M** is high and the decoding process fails. The system then may move on to process the next captured image. In another implementation, information about previous locations of the pen can be taken into consideration. That is, for each captured image, a destination area where the pen may be expected next can be identified. For example, if the user has not lifted the pen between two image captures by the camera, the location of the pen as determined by the second image capture should not be too far away from the first location. Each **r** that is associated with the minimum number of error bits can then be checked to see if the location *s* computed from **r** satisfies the local constraint, i.e., whether the location is within the destination area specified.

[78] If the location *s* satisfies the local constraint, the X, Y positions of the extracted bits in the array are returned. If not, the decoding process fails.

[79] Figure 8 depicts a process that may be used to determine a location in a sequence (or *m*-sequence) of a captured image. First, in step 801, a data stream relating to a captured image is received. In step 802, corresponding columns are extracted from **A** and a matrix **M** is constructed.

[80] In step 803, *n* independent column vectors are randomly selected from the matrix **M** and vector **r** is determined by solving equation (4). This process is performed $Q$ times (for example, 100 times) in step 804. The determination of the number of loop times is discussed in the section Loop Times Calculation.

[81] In step 805, **r** is sorted according to its associated number of error bits. The sorting can be done using a variety of sorting algorithms as known in the art. For example, a selection sorting algorithm may be used. The selection sorting algorithm is beneficial when the number $Q$ is not large. However, if $Q$ becomes large, other sorting algorithms (for example, a merge sort) that handle larger numbers of items more efficiently may be used.

[82] The system then determines in step 806 whether error correction was performed successfully, by checking whether multiple **r**'s are associated with the minimum number of error bits. If yes, an error is returned in step 809, indicating the decoding process failed. If not, the position $s$ of the extracted bits in the sequence (or $m$-sequence) is calculated in step 807, for example, by using the Pohig-Hellman-Silver algorithm.

[83] Next, the (X,Y) position in the array is calculated as: $x = s \bmod m_1$ and $y = s \bmod m_2$ and the results are returned in step 808.

### *Location Determination*

[84] Figure 9 shows a process for determining the location of a pen tip. The input is an image captured by a camera and the output may be a position coordinates of the pen tip. Also, the output may include (or not) other information such as a rotation angle of the captured image.

[85] In step 901, an image is received from a camera. Next, the received image may be optionally preprocessed in step 902 (as shown by the broken outline of step 902) to adjust the contrast between the light and dark pixels and the like.

[86] Next, in step 903, the image is analyzed to determine the bit stream within it.

[87] Next, in step 904, $n$ bits are randomly selected from the bit stream for multiple times and the location of the received bit stream within the original sequence (or $m$-sequence) is determined.

[88] Finally, once the location of the captured image is determined in step 904, the location of the pen tip may be determined in step 905.

[89] Figure 10 gives more details about 903 and 904 and shows the approach to extract the bit stream within a captured image. First, an image is received from the camera in step 1001. The image then may optionally undergo image preprocessing in step 1002 (as shown by

2004242549 31 Dec 2004

the broken outline of step 1002). The pattern is extracted in step 1003. Here, pixels on the various lines may be extracted to find the orientation of the pattern and the angle $\theta$.

[90] Next, the received image is analyzed in step 1004 to determine the underlying grid lines. If grid lines are found in step 1005, then the code is extracted from the pattern in step 1006. The code is then decoded in step 1007 and the location of the pen tip is determined in step 1008. If no grid lines were found in step 1005, then an error is returned in step 1009.

### *Outline of Enhanced Decoding and Error Correction Algorithm*

[91] With an embodiment of the invention as shown in Figure 12, given extracted bits 1201 from a captured image (corresponding to a captured array) and the destination area, a variation of an *m*-array decoding and error correction process decodes the X,Y position. Figure 12 shows a flow diagram of process 1200 of this enhanced approach. Process 1200 comprises two components 1251 and 1253.

- ***Decode Once.*** Component 1251 includes three parts.
  - ➢ random bit selection: randomly selects a subset of the extracted bits 1201 (step1203)
  - ➢ decode the subset (step 1205)
  - ➢ determine X,Y position with local constraint (step1209)
- ***Decoding with Smart Bit Selection.*** Component 1253 includes four parts.
  - ➢ smart bit selection: selects another subset of the extracted bits (step1217)
  - ➢ decode the subset (step 1219)
  - ➢ adjust the number of iterations (loop times) of step 1217 and step 1219 (step 1221)
  - ➢ determine X,Y position with local constraint (step 1225)

[92] The embodiment of the invention utilizes a discreet strategy to select bits, adjusts the number of loop iterations, and determines the X,Y position (location coordinates) in accordance with a local constraint, which is provided to process 1200. With both components 1251 and 1253, steps 1205 and 1219 ("Decode Once") utilize equation (4) to compute **r**.

Let $\hat{\mathbf{b}}$ be decoded bits, that is:

$$\hat{\mathbf{b}}^t = \mathbf{r}^t\mathbf{M} \qquad (5)$$

The difference between **b** and $\hat{\mathbf{b}}$ are the error bits associated with **r**.

[93] Figure 12 shows a flow diagram of process 1200 for decoding extracted bits 1201 from a captured image in accordance with embodiments of the present invention. Process 1200 comprises components 1251 and 1253. Component 1251 obtains extracted bits 1201 (comprising *K* bits) associated with a captured image (corresponding to a captured array). In step 1203, *n* bits (where *n* is the order of the *m*-array) are randomly selected from extracted bits 1201. In step 1205, process 1200 decodes once and calculates **r**. In step 1207, process 1200 determines if error bits are detected for **b**. If step 1207 determines that there are no error bits, X,Y coordinates of the position of the captured array are determined in step 1209. With step 1211, if the X,Y coordinates satisfy the local constraint, i.e., coordinates that are within the destination area, process 1200 provides the X,Y position (such as to another process or user interface) in step 1213. Otherwise, step 1215 provides a failure indication.

[94] If step 1207 detects error bits in **b**, component 1253 is executed in order to decode with error bits. Step 1217 selects another set of *n* bits (which differ by at least one bit from the *n* bits selected in step 1203) from extracted bits 1201. Steps 1221 and 1223 determine the number of iterations (loop times) that are necessary for decoding the extracted bits. Step 1225 determines the position of the captured array by testing which candidates obtained

in step 1219 satisfy the local constraint. Steps 1217-1225 will be discussed in more details.

### *Smart Bit Selection*

[95] Step 1203 randomly selects $n$ bits from extracted bits 1201 (having $K$ bits), and solves for $\mathbf{r}_1$. Using equation (5), decoded bits can be calculated. Let $I_1 = \{k \in \{1,2,\cdots,K\} \mid b_k = \hat{b}_k\}$, $\bar{I}_1 = \{k \in \{1,2,\cdots,K\} \mid b_k \neq \hat{b}_k\}$, where $\hat{b}_k$ is the k$^{th}$ bit of $\hat{\mathbf{b}}$, $B_1 = \{b_k \mid k \in I_1\}$ and $\bar{B}_1 = \{b_k \mid k \in \bar{I}_1\}$, that is, $B_1$ are bits that the decoded results are the same as the original bits, and $\bar{B}_1$ are bits that the decoded results are different from the original bits, $I_1$ and $\bar{I}_1$ are the corresponding indices of these bits. It is appreciated that the same $\mathbf{r}_1$ will be obtained when any $n$ bits are selected from $B_1$. Therefore, if the next $n$ bits are not carefully chosen, it is possible that the selected bits are a subset of $B_1$, thus resulting in the same $\mathbf{r}_1$ being obtained.

[96] In order to avoid such a situation, step 1217 selects the next $n$ bits according to the following procedure:

1. Choose at least one bit from $\bar{B}_1$ 1303 and the rest of the bits randomly from $B_1$ 1301 and $\bar{B}_1$ 1303, as shown in Figure 13 corresponding to bit arrangement 1351. Process 1200 then solves $\mathbf{r}_2$ and finds $B_2$ 1305, 1309 and $\bar{B}_2$ 1307, 1311 by computing $\hat{\mathbf{b}}_2^t = \mathbf{r}_2^t \mathbf{M}_2$.
2. Repeat step 1. When selecting the next $n$ bits, for every $\bar{B}_i$ (i = 1, 2, 3..., $x$-1, where $x$ is the current loop number), there is at least one bit selected from $\bar{B}_i$. The iteration terminates when no such subset of bits can be selected or when the loop times are reached.

2004242549 31 Dec 2004

### *Loop Times Calculation*

[97] With the error correction component 1253, the number of required iterations (loop times) is adjusted after each loop. The loop times is determined by the expected error rate. The expected error rate $p_e$ in which not all the selected $n$ bits are correct is:

$$p_e = \left(1 - \frac{C_{K-n_e}^n}{C_K^n}\right)^{lt} \approx -e^{-lt\left(\frac{K-n}{K}\right)^{n_e}} \qquad (6)$$

where $lt$ represents the loop times and is initialized by a constant, $K$ is the number of extracted bits from the captured array, $n_e$ represents the minimum number of error bits incurred during the iteration of process 1200, $n$ is the order of the $m$-array, and $C_K^n$ is the number of combinations in which $n$ bits are selected from $K$ bits.

[98] In the embodiment, we want $p_e$ to be less than $e^{-5} = 0.0067$. In combination with (6), we have:

$$lt_i = \min\left(lt_{i-1}, \frac{5}{\left(\frac{K-n}{K}\right)^{n_e}} + 1\right) \qquad (7)$$

Adjusting the loop times may significantly reduce the number of iterations of process 1253 that are required for error correction.

### *Determine X, Y Position with Local Constraint*

[99] In steps 1209 and 1225, the decoded position should be within the destination area. The destination area is an input to the algorithm, and it may be of various sizes and places or simply the whole $m$-array depending on different applications. Usually it can be predicted by the application. For example, if the previous position is determined, considering the writing speed, the destination area of the current pen tip should be close to the previous position. However, if the pen is lifted, then its next position can be anywhere. Therefore, in this case, the destination area should be the whole m-array. The correct X,Y position is determined by the following steps.

[100] In step 1224 process 1200 selects $\mathbf{r}_i$ whose corresponding number of error bits is less than:

$$N_e = \frac{\log_{10}\left(\frac{3}{lt}\right)}{\log_{10}\left(\frac{K-n}{K}\right) \times \log_{10}\left(\frac{10}{lr}\right)} \tag{8}$$

where *lt* is the actual loop times and *lr* represents the Local Constraint Rate calculated by:

$$lr = \frac{\text{area of the } destination \text{ area}}{L} \tag{9}$$

where $L$ is the length of the m-array.

[101] Step 1224 sorts $\mathbf{r}_i$ in ascending order of the number of error bits. Steps 1225, 1211 and 1212 then finds the first $\mathbf{r}_i$ in which the corresponding X,Y position is within the destination area. Steps 1225, 1211 and 1212 finally returns the X,Y position as the result (through step 1213), or an indication that the decoding procedure failed (through step 1215).

### *Illustrative Example of Enhanced Decoding and Error Correction Process*

[102] An illustrative example demonstrates process 1200 as performed by components 1251 and 1253. Suppose $n$=3, $K$=5, $\mathbf{I} = (I_0 \quad I_1 \quad \cdots \quad I_6)^t$ is the $m$-sequence of order $n$=3. Then

$$\mathbf{A} = \begin{pmatrix} I_0 & I_1 & I_2 & I_3 & I_4 & I_5 & I_6 \\ I_6 & I_0 & I_1 & I_2 & I_3 & I_4 & I_5 \\ I_5 & I_6 & I_0 & I_1 & I_2 & I_3 & I_4 \end{pmatrix} \tag{10}$$

Also suppose that the extracted bits $\mathbf{b} = (b_0 \quad b_1 \quad b_2 \quad b_3 \quad b_4)^t$, where $K$=5, are actually the $s^{th}$, $(s+1)^{th}$, $(s+3)^{th}$, $(s+4)^{th}$, and $(s+6)^{th}$ bits of the $m$-sequence (these numbers are actually modulus of the m-array length $L = 2^n - 1 = 2^3 - 1 = 7$). Therefore

$$\mathbf{M} = \begin{pmatrix} I_0 & I_1 & I_3 & I_4 & I_6 \\ I_6 & I_0 & I_2 & I_3 & I_5 \\ I_5 & I_6 & I_1 & I_2 & I_4 \end{pmatrix} \tag{11}$$

which consists of the $0^{th}$, $1^{st}$, $3^{rd}$ , $4^{th}$, and $6^{th}$ columns of **A**. The number $s$, which uniquely determines the X,Y position of $b_0$ in the $m$-array, can be computed after solving $\mathbf{r} = (r_0 \quad r_1 \quad r_2)^t$ that are expected to fulfill $\mathbf{b}^t = \mathbf{r}^t\mathbf{M}$. Due to possible error bits in **b**, $\mathbf{b}^t = \mathbf{r}^t\mathbf{M}$ may not be completely fulfilled.

[103] Process 1200 utilizes the following procedure. Randomly select $n$=3 bits, say $\tilde{\mathbf{b}}_1^t = (b_0 \quad b_1 \quad b_2)$, from **b**. Solving for $\mathbf{r}_1$:

$$\tilde{\mathbf{b}}_1^t = \mathbf{r}_1{}^t\tilde{\mathbf{M}}_1 \tag{12}$$

where $\tilde{\mathbf{M}}_1$ consists of the 0th, 1st, and 2nd columns of **M**. (Note that $\tilde{\mathbf{M}}_1$ is an $n \times n$ matrix and $\mathbf{r}_1^t$ is a $1 \times n$ vector so that $\tilde{\mathbf{b}}_1^t$ is a $1 \times n$ vector of selected bits.)

[104] Next, decoded bits are computed:

$$\hat{\mathbf{b}}_1{}^t = \mathbf{r}_1{}^t\mathbf{M} \tag{13}$$

where **M** is an $n \times K$ matrix and $\mathbf{r}_1^t$ is a $1 \times n$ vector so that $\hat{\mathbf{b}}_1^t$ is a $1 \times K$ vector. If $\hat{\mathbf{b}}_1$ is identical to **b**, i.e., no error bits are detected, then step 1209 determines the X,Y position and step 1211 determines whether the decoded position is inside the destination area. If so, the decoding is successful, and step 1213 is performed. Otherwise, the decoding fails as indicated by step 1215. If $\hat{\mathbf{b}}_1$ is different from **b**, then error bits in **b** are detected and component 1253 is performed. Step 1217 determines the set $B_1$, say $\{b_0\ b_1\ b_2\ b_3\}$, where the decoded bits are the same as the original bits. Thus, $\overline{B}_1 = \{b_4\}$ (corresponding to bit arrangement 1351 in Figure 13). Loop times ($lt$) is initialized to a constant, e.g., 100, which may be variable depending on the application. Note that the number of error bits corresponding to $\mathbf{r}_1$ is equal to 1. Then step 1221 updates the loop time ($lt$) according to equation (7), $lt_1 = \min(lt, 13) = 13$.

[105] Step 1217 next chooses another $n$=3 bits from **b**. If the bits all belong to $B_1$, say $\{b_0 \quad b_2 \quad b_3\}$, then step 1219 will determine $\mathbf{r}_1$ again. In order to avoid such repetition, step 1217 may select, for example, one bit $\{b_4\}$ from $\overline{B}_1$, and the remaining two bits $\{b_0 \quad b_1\}$ from $B_1$.

[106] The selected three bits form $\tilde{\mathbf{b}}_2^t = (b_0 \quad b_1 \quad b_4)$. Step 1219 solves for $\mathbf{r}_2$:

$$\tilde{\mathbf{b}}_2^t = \mathbf{r}_2^t \tilde{\mathbf{M}}_2 \tag{14}$$

where $\tilde{\mathbf{M}}_2$ consists of the 0th, 1st, and 4th columns of **M**.

[107] Step 1219 computes $\hat{\mathbf{b}}_2^t = \mathbf{r}_2^t \mathbf{M}$. Find the set $B_2$, e.g., $\{b_0 \quad b_1 \quad b_4\}$, such that $\hat{\mathbf{b}}_2$ and **b** are the same. Then $\overline{B}_2 = \{b_2 \quad b_3\}$ (corresponding to bit arrangement 1353 in Figure 13). Step 1221 updates the loop times ($lt$) according to equation (7). Note that the number of error bits associated with $\mathbf{r}_2$ is equal to 2. Substituting into (7), $lt_2 = \min(lt_1, 32) = 13$.

[108] Because another iteration needs to be performed, step 1217 chooses another $n$=3 bits from **b**. The selected bits shall not all belong to either $B_1$ or $B_2$. So step 1217 may select, for example, one bit $\{b_4\}$ from $\overline{B}_1$, one bit $\{b_2\}$ from $\overline{B}_2$, and the remaining one bit $\{b_0\}$.

[109] The solution of **r**, bit selection, and loop times adjustment continues until we cannot select any new $n$=3 bits such that they do not all belong to any previous $B_i$'s, or the maximum loop times $lt$ is reached.

[110] Suppose that process 1200 calculates five $\mathbf{r}_i$ ($i$=1,2,3,4,5), with the number of error bits corresponding to 1, 2, 4, 3, 2, respectively. (Actually, for this example, the number of error bits cannot exceed 2, but the illustrative example shows a larger number of error

bits to illustrate the algorithm.) Step 1224 selects $\mathbf{r}_i$'s, for example, $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_4, \mathbf{r}_5$, whose corresponding numbers of error bits are less than $N_e$ shown in (8).

[111] Step 1224 sorts the selected vectors $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_4, \mathbf{r}_5$ in ascending order of their error bit numbers: $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_5, \mathbf{r}_4$. From the sorted candidate list, steps 1225, 1211 and 1212 find the first vector $\mathbf{r}$, for example, $\mathbf{r}_5$, whose corresponding position is within the destination area. Step 1213 then outputs the corresponding position. If none of the positions is within the destination area, the decoding process fails as indicated by step 1215.

### *Apparatus*

[112] Figure 14 shows an apparatus 1400 for decoding extracted bits 1201 from a captured array in accordance with embodiments of the present invention. Apparatus 1400 comprises bit selection module 1401, decoding module 1403, position determination module 1405, input interface 1407, and output interface 1409. In the embodiment, interface 1407 may receive extracted bits 1201 from different sources, including a module that supports camera 203 (as shown in Figure 2A). Bit selection module 1401 selects $n$ bits from extracted bits 1201 in accordance with steps 1203 and 1217. Decoding module 1403 decodes the selected bits ($n$ bits selected from the $K$ extracted bits as selected by bit selection module 1401) to determine detected bit errors and corresponding vectors $\mathbf{r}_i$ in accordance with steps 1205 and 1219. Decoding module 1403 presents the determined vectors $\mathbf{r}_i$ to position determination module 1405. Position determination module 1405 determines the X,Y coordinates of the captured array in accordance with steps 1209 and 1225. Position determination module 1405 presents the results, which includes the X,Y coordinates if successful and an error indication if not successful, to output interface 1409. Output interface 1409 may present the results to another module that may perform further processing or that may display the results.

[113] Apparatus 1400 may assume different forms of implementation, including modules utilizing computer-readable media and modules utilizing specialized hardware such as an application specific integrated circuit (ASIC).

[114] As can be appreciated by one skilled in the art, a computer system with an associated computer-readable medium containing instructions for controlling the computer system can be utilized to implement the exemplary embodiments that are disclosed herein. The computer system may include at least one computer such as a microprocessor, digital signal processor, and associated peripheral electronic circuitry.

[115] Although the invention has been defined using the appended claims, these claims are illustrative in that the invention is intended to include the elements and steps described herein in any combination or sub combination. Accordingly, there are any number of alternative combinations for defining the invention, which incorporate one or more elements from the specification, including the description, claims, and drawings, in various combinations or sub combinations. It will be apparent to those skilled in the relevant technology, in light of the present specification, that alternate combinations of aspects of the invention, either alone or in combination with one or more elements or steps defined herein, may be utilized as modifications or alterations of the invention or as part of the invention. It may be intended that the written description of the invention contained herein covers all such modifications and alterations.

[116] Throughout this specification and the claims which follow, unless the context requires otherwise, the word "comprise", and variations such as "comprises" or "comprising", will be understood to imply the inclusion of a stated integer or step or group of integers or steps but not the exclusion of any other integer or step or group of integers or steps.

[117] The reference to any prior art in this specification is not, and should not be taken as, an acknowledgement or any form of suggestion that that prior art forms part of the common general knowledge in Australia.

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:

1. A method for determining a location of a captured array, the method comprising:

(A) obtaining extracted bits that are associated with the captured array;

(B) determining position coordinates if there are no error bits; and

(C) if there are error bits, determining the position coordinates from a portion of the extracted bits by non-repetitive bit selections, wherein the position coordinates are consistent with a local constraint.

2. The method of claim 1, wherein (B) comprises:

(i) selecting a first subset from the extracted bits;

(ii) decoding the first subset; and

(iii) in response to (ii), if no error bits are detected, determining the position coordinates of the captured array.

3. The method of claim 2, wherein (ii) utilizes a first matrix equation $\tilde{\mathbf{b}}^t = \mathbf{r}^t\tilde{\mathbf{M}}$, and wherein $\tilde{\mathbf{M}}$ is a sub-matrix of $\mathbf{M}$ in order to determine a vector $\mathbf{r}$.

4. The method of claim 3, wherein (ii) further utilizes a second matrix equation $\hat{\mathbf{b}}^t = \mathbf{r}^t\mathbf{M}$ in order to determine decoded bits.

5. The method of claim 2, wherein (iii) comprises:

(1) comparing decoded bits with the extracted bits.

6. The method of claim 1, wherein (C) comprises:

(i) if an error bit is detected, selecting a different subset from the extracted bits, wherein at least one bit of the different subset is from previous sets of error bits;

(ii) decoding with corresponding bits of the different subset;

(iii) in response to (ii), determining whether another decoding iteration shall be performed;

(iv) if another decoding iteration shall be performed, selecting another subset from the extracted bits and repeating (ii); and

(v) if another decoding iteration shall not be performed, determining the position coordinates of the captured array.

7. The method of claim 6, wherein calculated bits given by $\hat{\mathbf{b}}' = \mathbf{r}'\mathbf{M}$ are different from the extracted bits.

8. The method of claim 2, wherein (B) further comprises:

(iv) verifying that the position coordinates are within a destination area.

9. The method of claim 6, wherein (C) further comprises:

(vi) verifying that the position coordinates are within a destination area .

10. The method of claim 2, wherein (i) comprises:

(1) randomly choosing constituent bits of the first subset from the extracted bits.

11. The method of claim 6, wherein (i) comprises:

(1) choosing the corresponding bits of the different subset from the extracted bits in a manner as to whether decoded bits satisfy a matrix equation $\mathbf{b}^t=\mathbf{r}^t\mathbf{M}$ in a previous iteration of the method; and

(2) computing how many different bits there are between calculated bits given by $\hat{\mathbf{b}}^t = \mathbf{r}^t\mathbf{M}$ and the extracted bits.

12. The method of claim 6, wherein (v) comprises:

(1) selecting a determined vector $\mathbf{r}_i$ if the determined vector corresponds to a number of error bits that is less than a threshold; and

(2) in response to (1), ordering a plurality of determined vectors in an ascending order by a corresponding number of error bits.

13. The method of claim 12, wherein (v) further comprises:

(3) in response to (2), finding a first solution that corresponds to the position coordinates within a destination area.

14. The method of claim 13, wherein (v) further comprises:

(4) if no solutions are located within the destination area, indicating a failure of decoding.

15. The method of claim 6, wherein (iii) comprises:

(1) adjusting a required number of iterations of the method based on an expected error rate of error bits.

16. The method of claim 6, wherein (ii) comprises:

(1) determining a vector $\mathbf{r}$ by utilizing a first matrix equation $\tilde{\mathbf{b}}^t = \mathbf{r}^t\tilde{\mathbf{M}}$;

(2) calculating decoded bits by utilizing a second matrix equation $\hat{\mathbf{b}}^t = \mathbf{r}^t\mathbf{M}$, wherein the vector is determined by (1); and

(3) comparing the decoded bits with the extracted bits to find a number of error bits.

17. A computer-readable medium having computer-executable instructions for performing the method as recited in claim 1.

18. A computer-readable medium having computer-executable instructions for performing the method as recited in claim 2.

19. A computer-readable medium having computer-executable instructions for performing the method as recited in claim 6.

20. An apparatus that determines position coordinates of a captured array, comprising:

(a) a bit selection module that applies a smart strategy to choose a subset of bits that has at least one bit from previous sets of error bits, the subset being selected from extracted bits that correspond to the captured array;

(b) a decoding module that processes the subset of bits to determine error information regarding the extracted bits and that determines whether another iteration of decoding is necessary from the error information; and

(c) a position determination module that processes the error information to determine position coordinates of the captured array, wherein the position coordinates are within a destination area.

21. The apparatus of claim 20, further comprising:

an input interface that receives the extracted bits and presents the extracted bits to the bit selection module for processing.

22. The apparatus of claim 20, wherein the decoding module calculates a location matrix $\mathbf{r}_i$ for the $i^{th}$ iteration and determines error bits by comparing decoded bits determined from $\hat{\mathbf{b}}' = \mathbf{r}'\mathbf{M}$ with the extracted bits.

23. The apparatus of claim 20, further comprising:

an output interface that provides the position coordinates of the captured array, wherein the position coordinates are determined by the position determination module.

24. A method for determining location coordinates of a captured array, the method comprising:

(A) receiving extracted bits that are associated with the captured array;

(B) selecting a first bit subset from the extracted bits;

(C) decoding the first bit subset;

(D) if no error bits are detected, determining the location coordinates of the captured array, the location coordinates being within a destination area;

(E) selecting a different subset from the extracted bits, wherein at least one bit of the different subset does not satisfy a matrix equation $\hat{\mathbf{b}}^t = \mathbf{r}^t\mathbf{M}$;

(F) decoding selected bits of the different subset;

(G) adjusting a number of iterations for performing (F), the number being adjusted according to results from (F);

(H) if another decoding iteration shall be performed, repeating (E)-(G); and

(I) if another decoding iteration shall not be performed, determining the location coordinates of the captured array, wherein the location coordinates are within the destination area.

25. A method substantially as hereinbefore described with reference to the drawings.

26. An apparatus substantially as hereinbefore described with reference to the drawings.

27. A computer-readable medium substantially as hereinbefore described with reference to the drawings.

28. The steps, features, compositions and compounds disclosed herein or referred to or indicated in the specification and/or claims of this application, individually or collectively, and any and all combinations of any two or more of said steps or features.

DATED this THIRTY FIRST day of DECEMBER 2004

Microsoft Corporation

by DAVIES COLLISON CAVE

Patent Attorneys for the applicant(s)

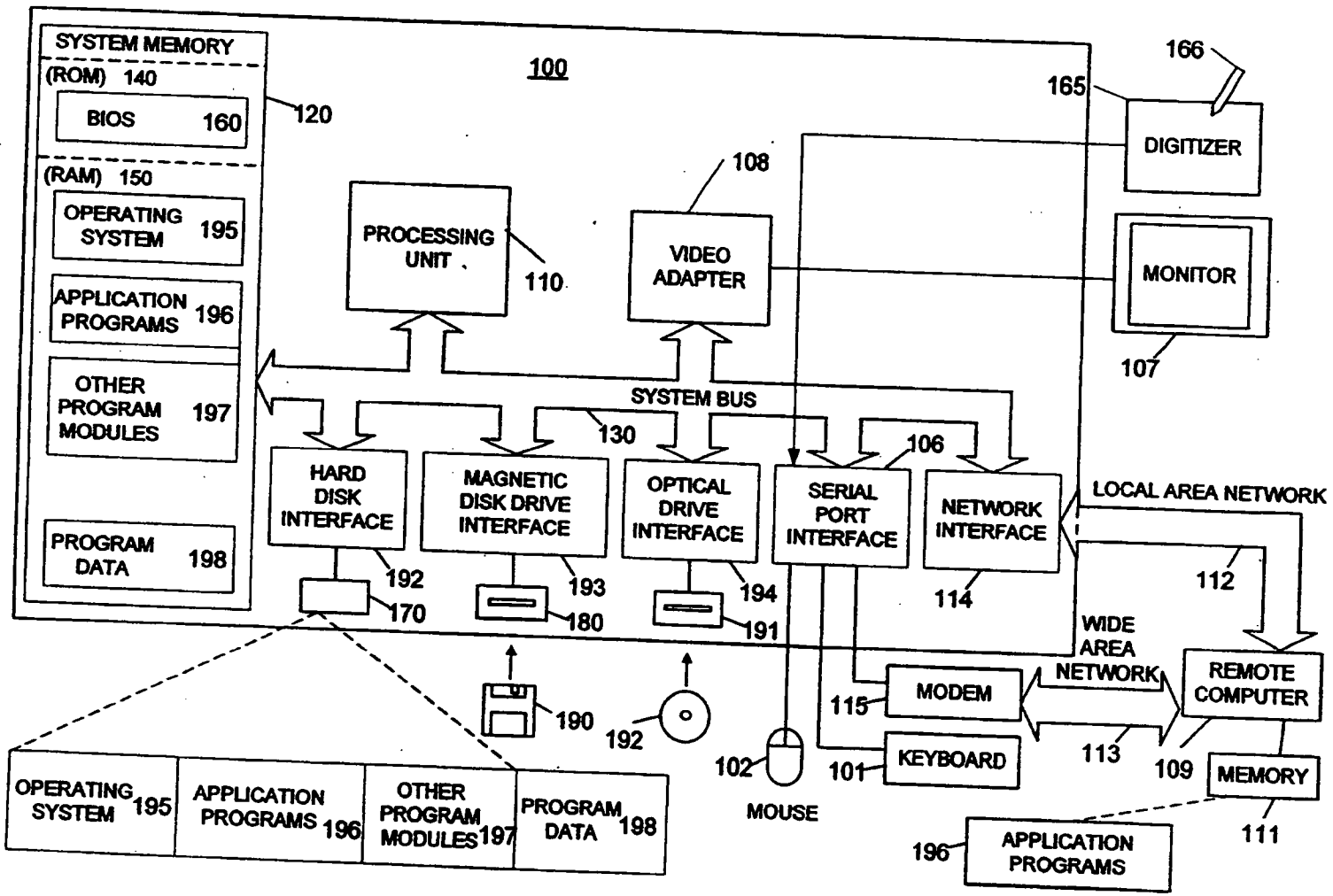
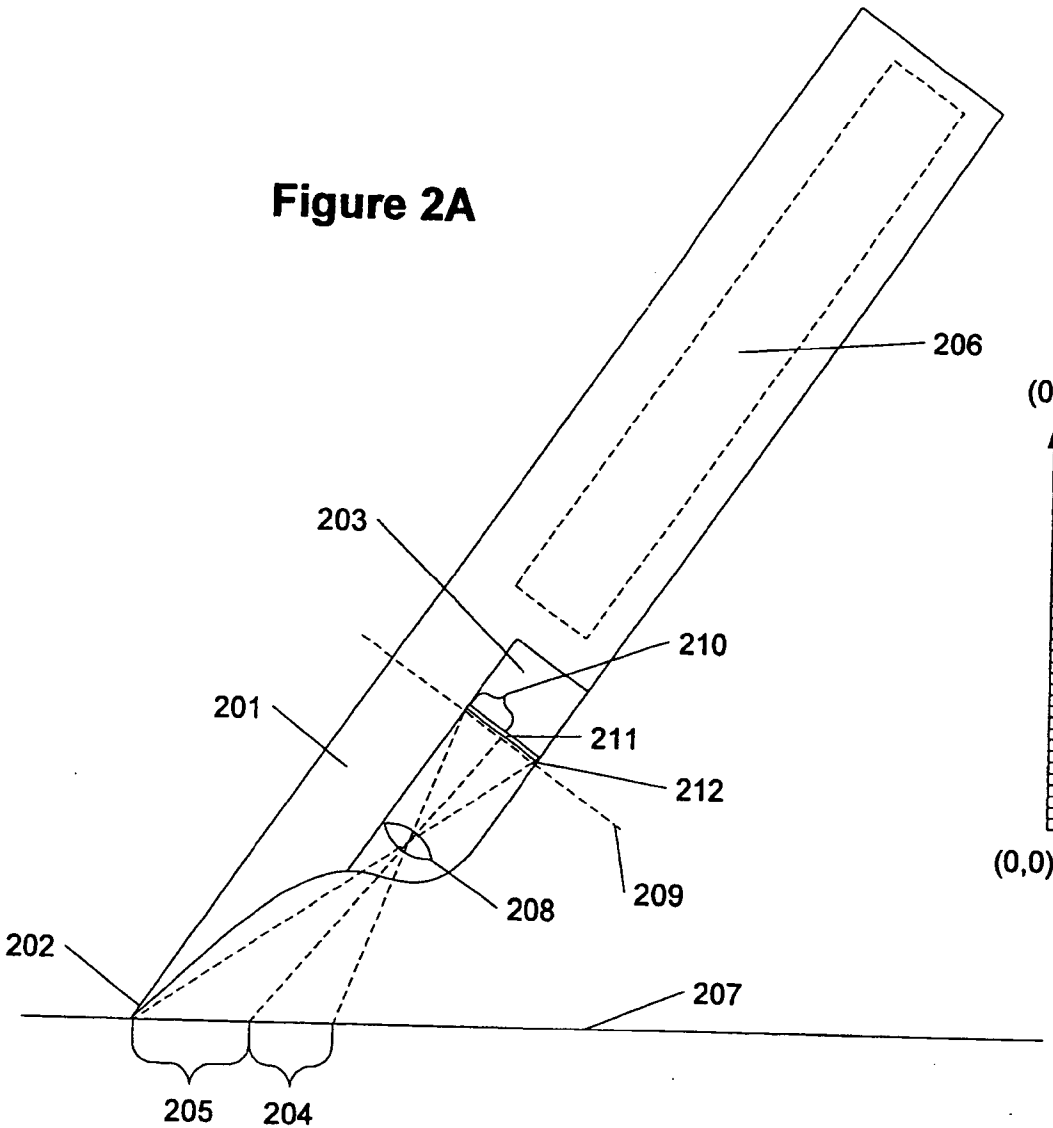

SYSTEM MEMORY
(ROM) 140
BIOS 160
120
(RAM) 150
OPERATING SYSTEM 195
APPLICATION PROGRAMS 196
OTHER PROGRAM MODULES 197
PROGRAM DATA 198
100
PROCESSING UNIT 110
108
VIDEO ADAPTER
166
165
DIGITIZER
MONITOR
107
SYSTEM BUS
130
106
HARD DISK INTERFACE
MAGNETIC DISK DRIVE INTERFACE
OPTICAL DRIVE INTERFACE
SERIAL PORT INTERFACE
NETWORK INTERFACE
LOCAL AREA NETWORK
112
192
193
194
114
170
180
191
WIDE AREA NETWORK
190
192
MODEM
115
REMOTE COMPUTER
102
101
KEYBOARD
113
109
MEMORY
MOUSE
111
196
APPLICATION PROGRAMS
OPERATING SYSTEM 195
APPLICATION PROGRAMS 196
OTHER PROGRAM MODULES 197
PROGRAM DATA 198

**Figure 1**

Figure 2A



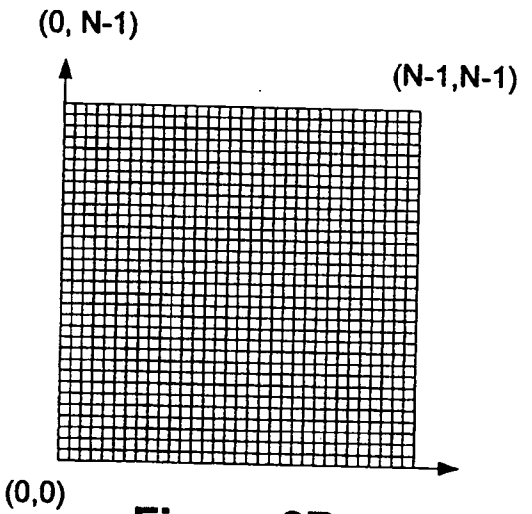




Figure 2B

**Figure 3A** 0000010 00 01100 0101 001 111010 0 0111001 0 010110 111 0110 01101 01 0111111

**Figure 3B** 0000000 0011101 0100111 1110100 0111010 1000101 1110100 0100111 0011101

**Figure 3C** 000100100 001111110 010111101 010011001 011100111 001011010 011000011

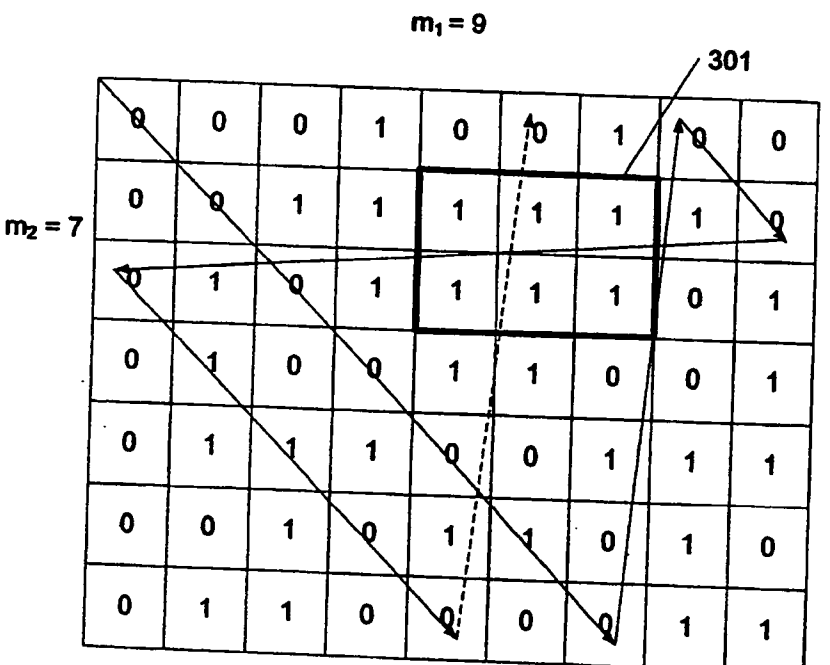



**Figure 3D**



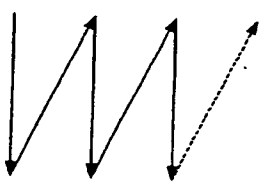


**Figure 3E**

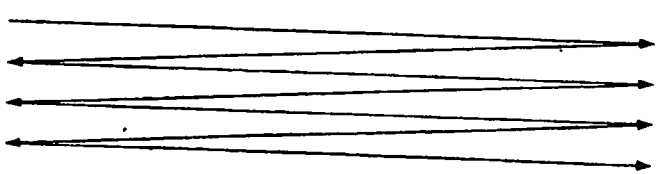


**Figure 3F**

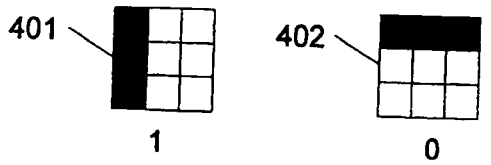



Figure 4A

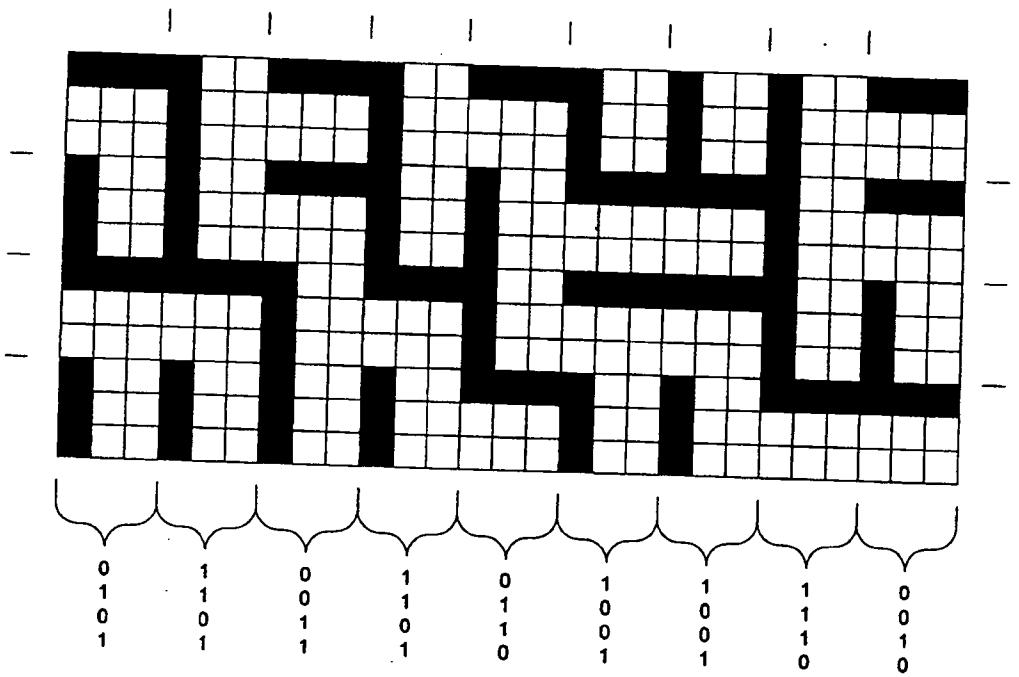



Figure 4B

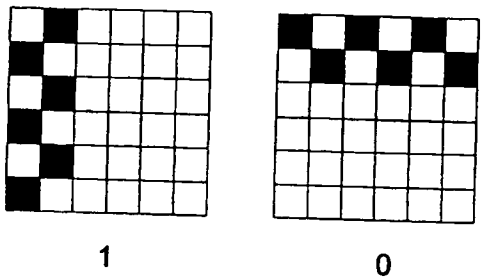



Figure 4C

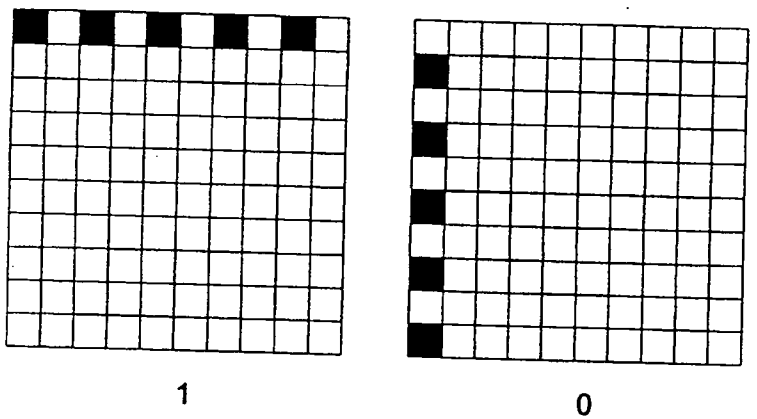



Figure 4D

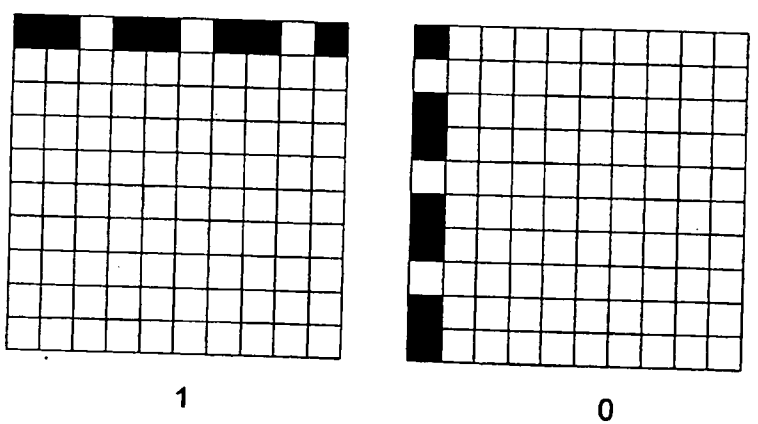



Figure 4E

Figure 5A

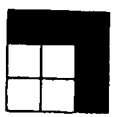


Figure 5B

Figure 5C

Figure 5D

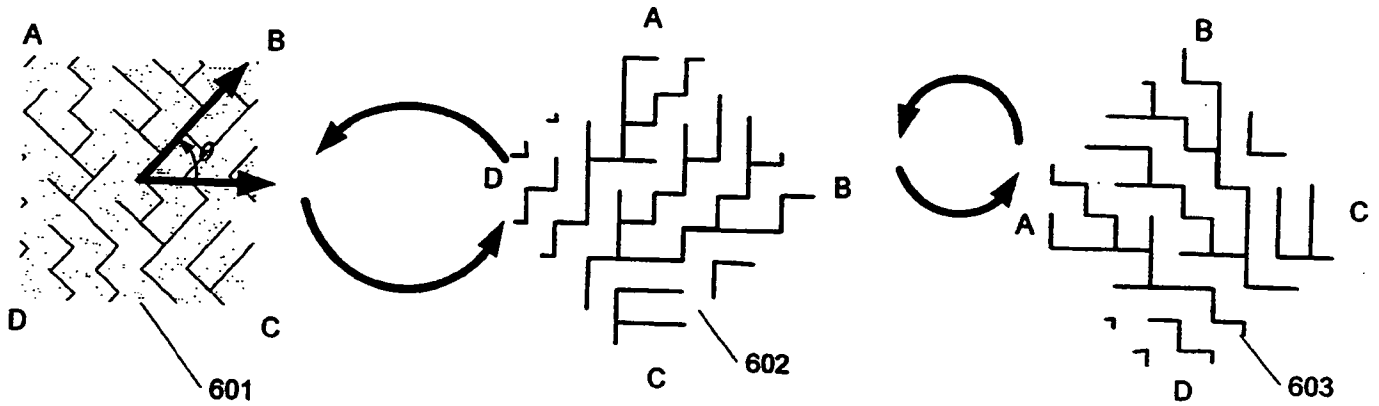



**Figure 6**

$$o=\theta+\begin{cases}0-no\ a\\ \frac{\pi}{2}-no\ b\\ \pi-no\ c\\ \frac{3\pi}{2}-no\ d\end{cases}$$

**Figure 7**

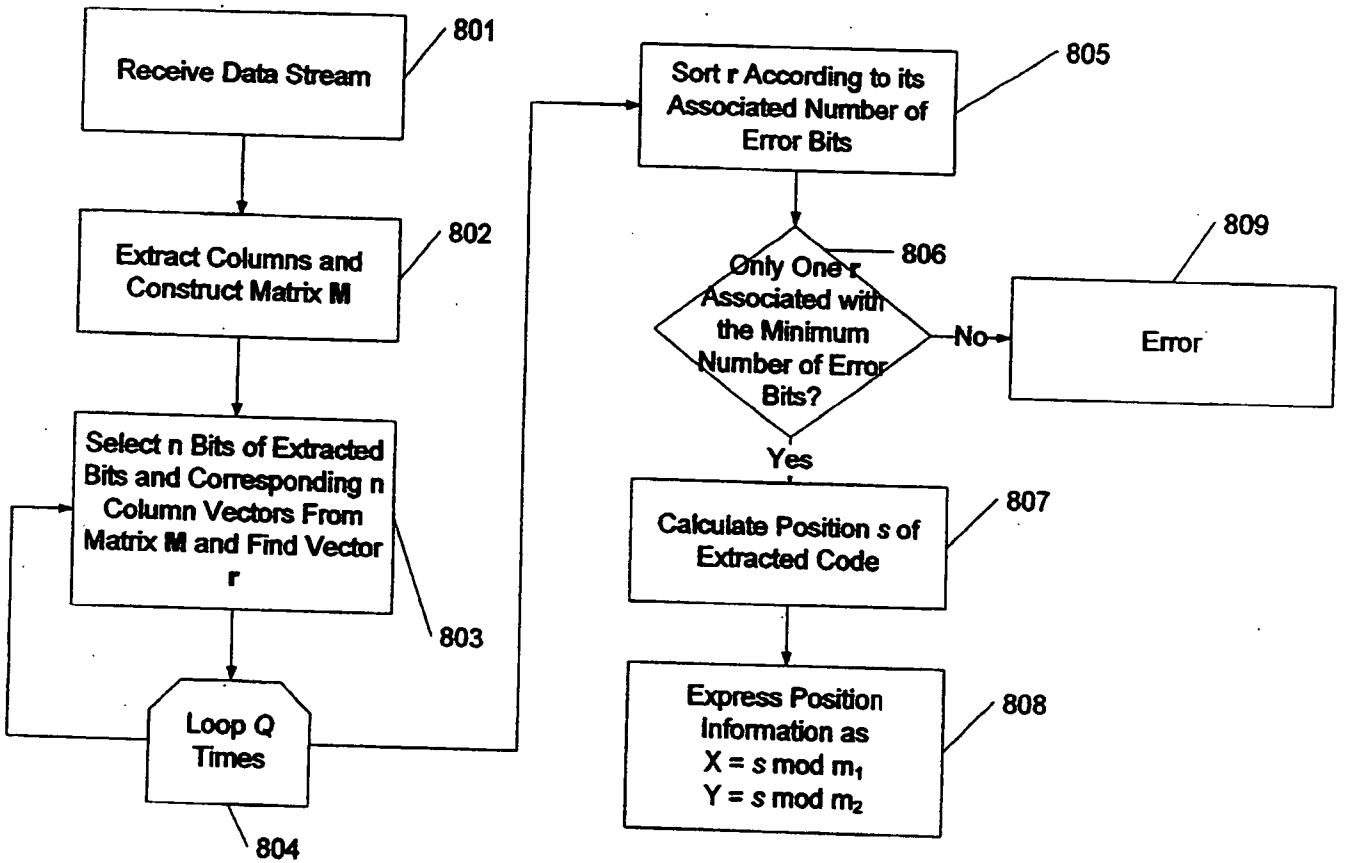

Receive Data Stream
801
Extract Columns and Construct Matrix M
802
Select n Bits of Extracted Bits and Corresponding n Column Vectors From Matrix M and Find Vector r
803
Loop Q Times
804
Sort r According to its Associated Number of Error Bits
805
Only One r Associated with the Minimum Number of Error Bits?
806
No
Error
809
Yes
Calculate Position s of Extracted Code
807
Express Position Information as X = s mod m1 Y = s mod m2
808

**Figure 8**

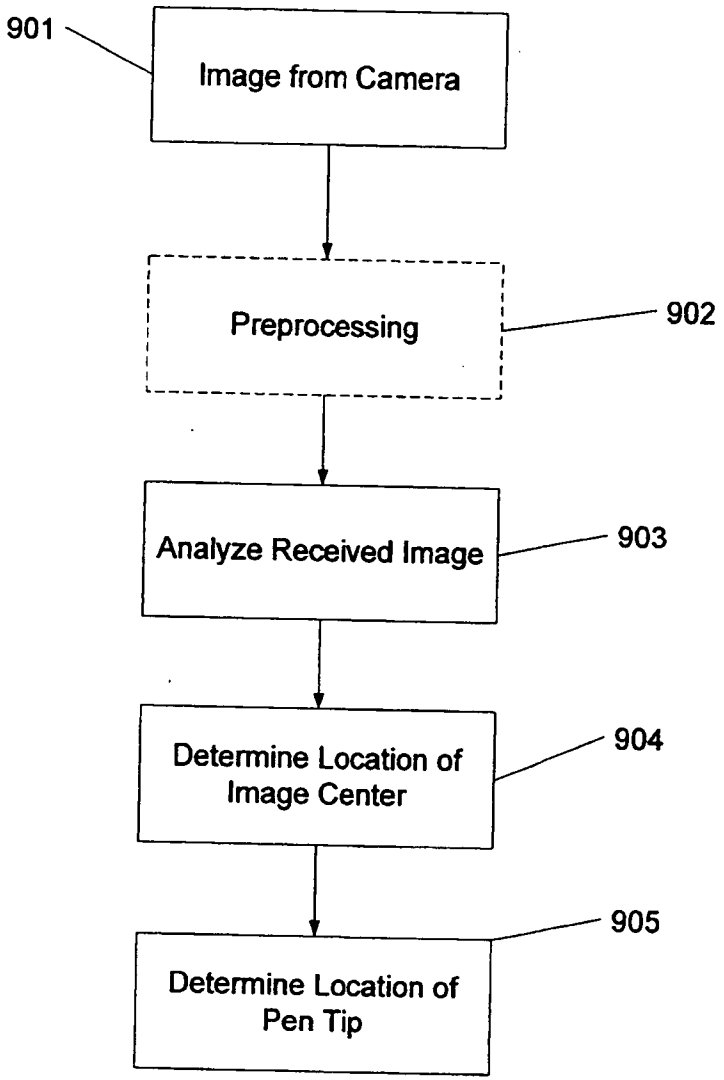

901
Image from Camera
Preprocessing
902
Analyze Received Image
903
Determine Location of Image Center
904
Determine Location of Pen Tip
905

**Figure 9**

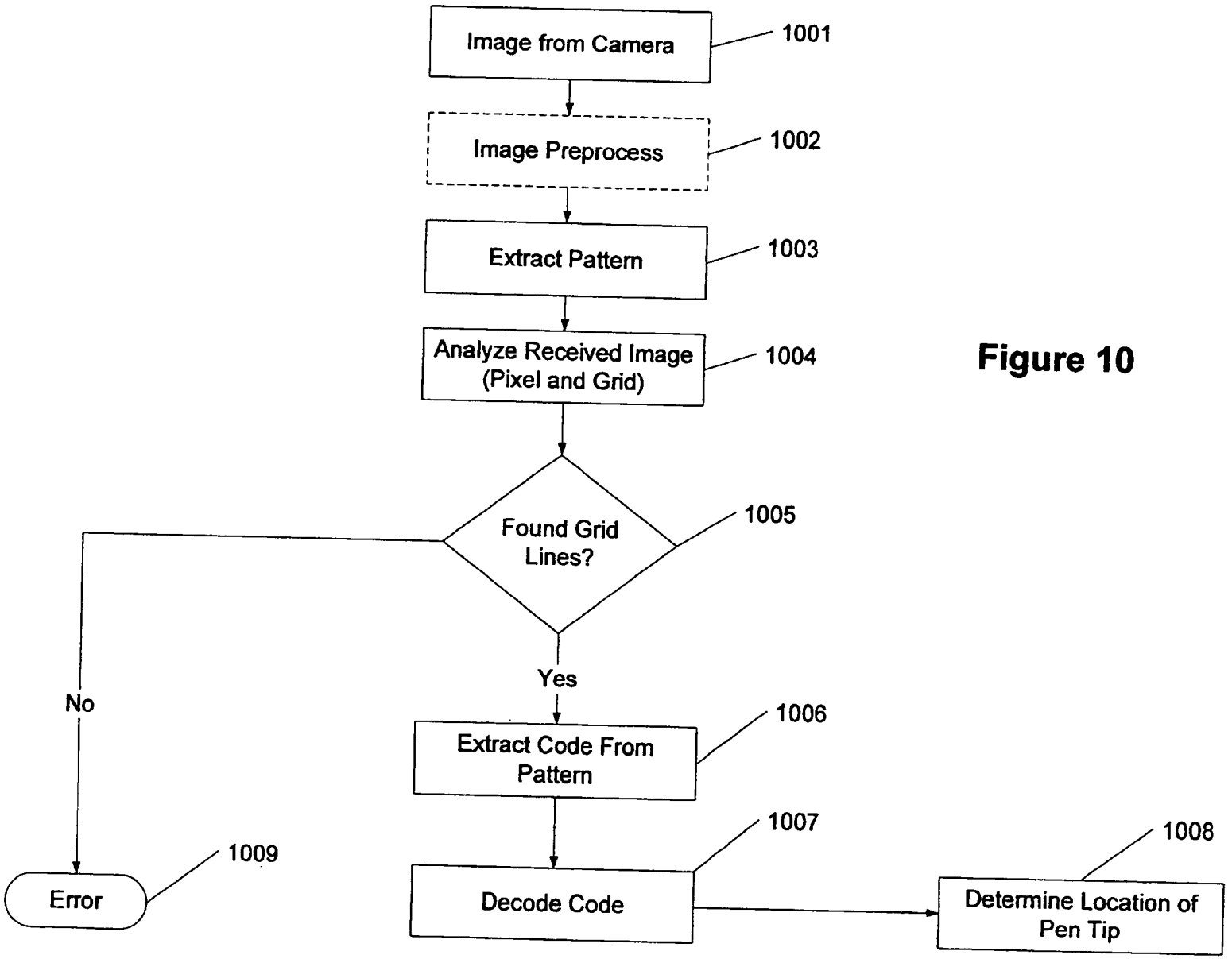


Image from Camera
1001
Image Preprocess
1002
Extract Pattern
1003
Analyze Received Image (Pixel and Grid)
1004
Found Grid Lines?
1005
No
Yes
Extract Code From Pattern
1006
Decode Code
1007
Determine Location of Pen Tip
1008
Error
1009

**Figure 10**

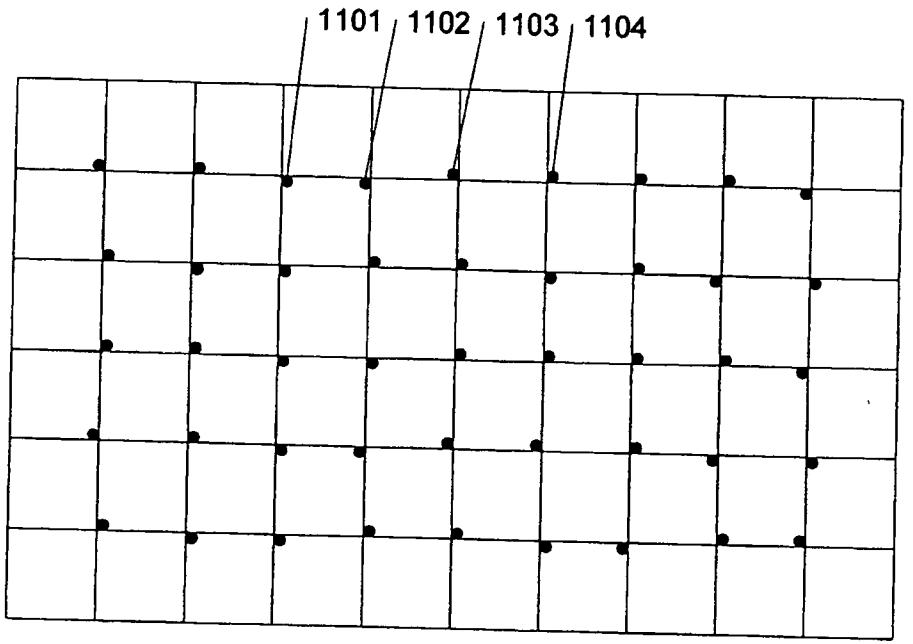



**Figure 11 (Prior Art)**

1200

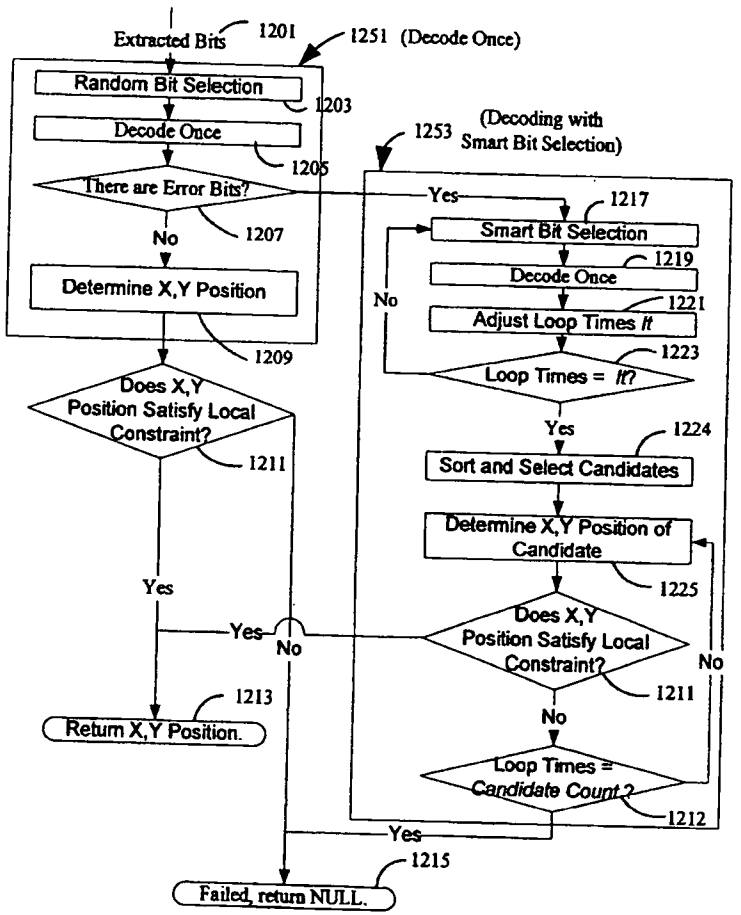

Extracted Bits
1201
1251 (Decode Once)
Random Bit Selection
1203
Decode Once
1205
There are Error Bits?
1207
No
Yes
Determine X,Y Position
1209
Does X,Y Position Satisfy Local Constraint?
1211
Yes
No
Return X,Y Position.
1213
1253 (Decoding with Smart Bit Selection)
Smart Bit Selection
1217
Decode Once
1219
Adjust Loop Times It
1221
Loop Times = It?
1223
No
Yes
Sort and Select Candidates
1224
Determine X,Y Position of Candidate
1225
Does X,Y Position Satisfy Local Constraint?
1211
Yes
No
No
Loop Times = Candidate Count ?
1212
Yes
Failed, return NULL.
1215

FIG. 12

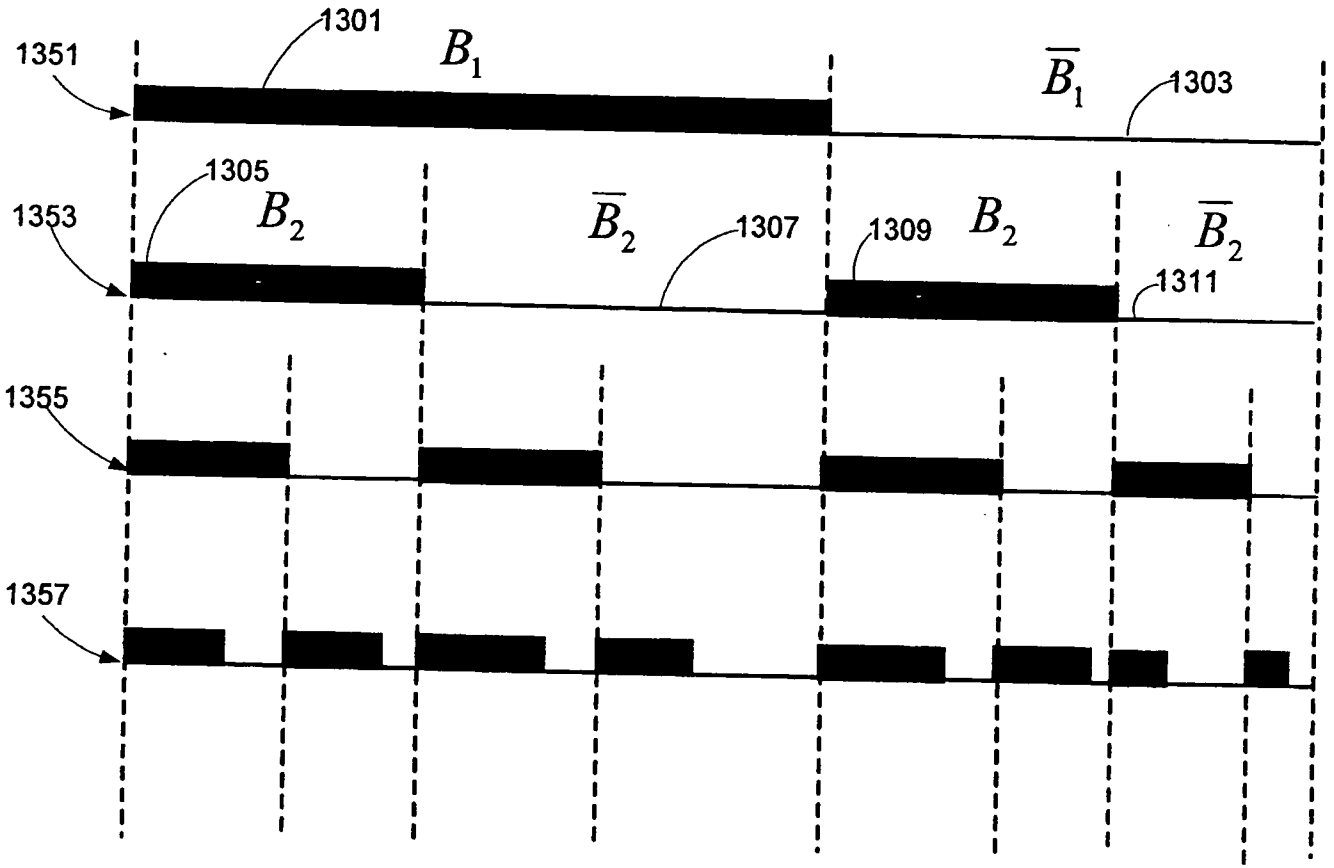

1301
1351
$B_1$
$\overline{B}_1$
1303
1305
1353
$B_2$
$\overline{B}_2$
1307
1309
$B_2$
$\overline{B}_2$
1311
1355
1357

FIG. 13

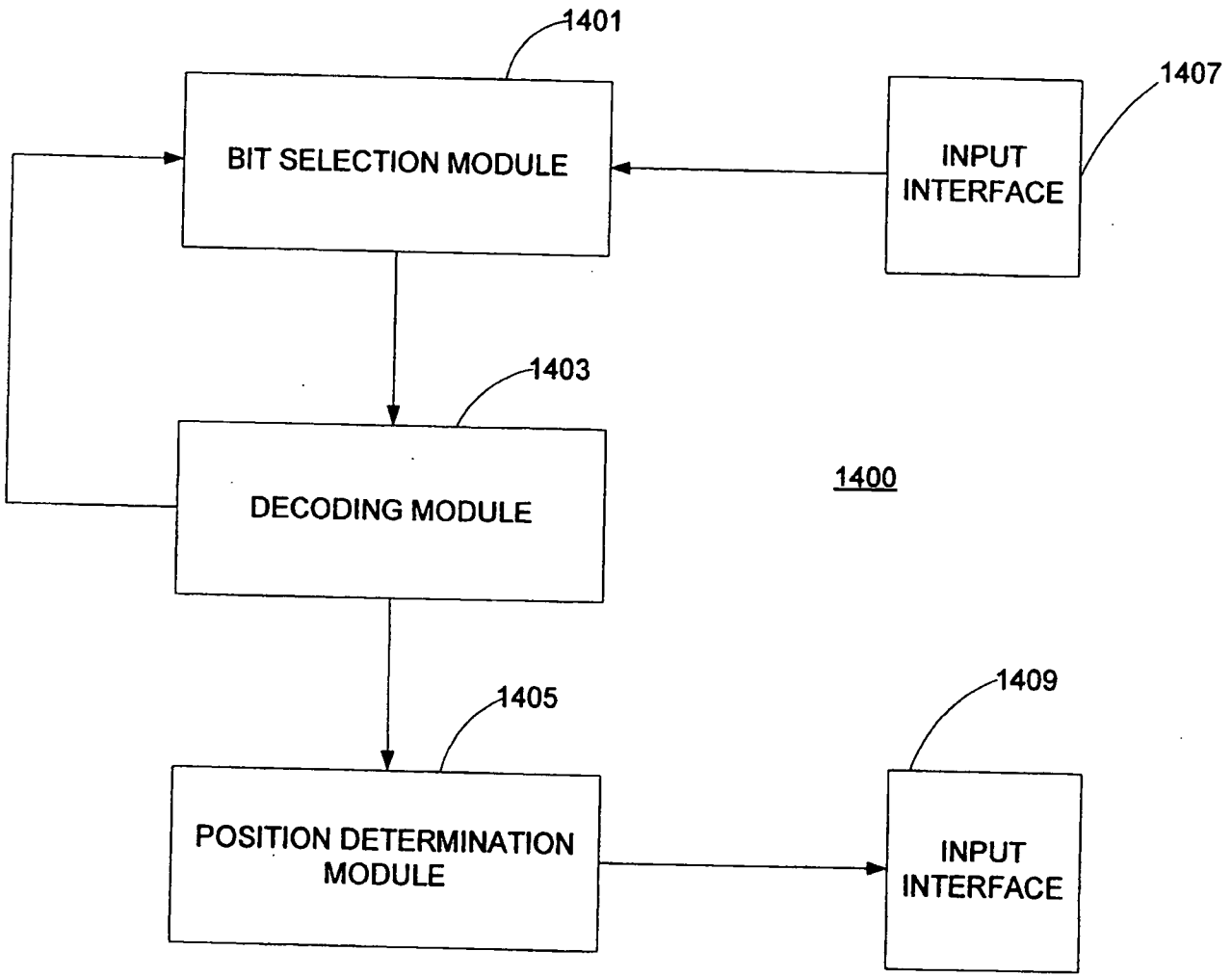

1401
BIT SELECTION MODULE
1407
INPUT INTERFACE
1403
DECODING MODULE
1400
1405
POSITION DETERMINATION MODULE
1409
INPUT INTERFACE

FIG. 14