Abstract:
This paper presents a new absolute position measurement method using pseudo-random binary sequences. This encoding technique has the notable advantage of requiring only one code-symbol per quantization interval. This compact encoding method offers an economical solution to very-high-resolution absolute position applications: shaft encoders, linear encoders, and "automated guided vehicle" (AGV) navigation.

1. Introduction

Absolute position measurement is more convenient than its alternative, incremental measurement, as it allows for instant position recovery, which is very important in position control, robotics and machine-vision applications. Absolute position recovery also has the notable advantage of not allowing errors to accumulate as may happen in the case of incremental position recovery.

Absolute position measurement usually requires that a distinct n-bit code \([x(1), x(2), ..., x(n)]\) be marked on each quantization interval of a scale, as shown in Fig. 1. The position \(P\) of the pointer relative to the origin of this scale is then estimated by reading the specific code written on the quantization interval currently facing the pointer:

\[
P = \sum_{k=1}^{n} x(k) \cdot 2^{k-1} \cdot q.
\]

where \(q\) is the quantization step.

Of course, the resulting number of code tracks on the scale increases proportionally with the desired measuring resolution, prohibiting the use of this straightforward absolute-encoding method for the very-high-resolution applications required by contemporary industrial practice.

This paper presents a non-traditional encoding method that represents a more efficient alternative to absolute position measurement applications. The new method is based on the properties of pseudo-
random binary sequences and has the notable advantage of requiring only one bit of code per quantization interval. This advantage makes absolute position measurement more accessible for very-high-resolution applications.

2. Pseudo-Random Binary Encoding

The scale consists in this case of only one track, which is encoded one symbol per quantization step with the terms of a "pseudo-random binary sequence" (PRBS). A \((2^n-1)\)-term PRBS \(\{S(p) \mid p=0,1,\ldots,2^n-2\}\) is generated by a \(n\)bit shift register \(\{R(n),\ldots,R(1)\}\) having an appropriate modulo-two feedback as illustrated in Fig. 2 and Table 1.

The absolute position recovery is based on the PRBS window property. According to this, any \(n\)-tuple \(\{S(p+n-k) \mid k=n,\ldots,1\}\) scanned by a window \(\{x(k) \mid k=n,\ldots,1\}\) is unique and fully identifies the current position index \(p\) of the window.

As an example, a 5-bit shift register having the modulo-two feedback equation \(R(0) = R(5) + R(1)\) generates the 31-term PRBS. It is easy to verify, in Fig. 3, that any content seen through a 5-bit window sliding over this sequence is unique.

3. PRBS to Natural-Code Conversion

A conversion from the PRBS code into the more convenient natural binary representation is always necessary for practical applications. A strictly parallel code conversion would be to use a code-conversion table stored in memory. This is expensive for applications requiring high-resolution position measurement.

At the other extreme, a serial code conversion exploits the reversibility of the PRBS generating algorithm. This method is based on the idea that it is possible to find the natural value associated with any pseudo-random code by simply counting the number of reverse feedback shifts that it takes for the given pseudo-random code to arrive back into the "zero position" pattern. In this case the solution requires less hardware but is more time-consuming for high-resolution measurements. Let us consider the example illustrated in Fig. 3 when the absolute position is \(p=12\cdot q\) and the corresponding PRBS window is 11000. The serial code conversion is performed by loading the initial window contents 11000 in a 5-bit shift register with a reverse-feedback equation \(R(6)=R(1) + R(3)\). The register is then "shifted back" through the intermediate states 01100, 10110, 11011, 11101, 01110, 10111, 01011,
10101, 01010, 00101, 00010, until it reaches the original state 00001 and stops. A binary counter, cleared at the beginning of this conversion, is incremented 12 times at every back-shift, finally yielding the natural code corresponding to the measured position.

A serial-parallel code conversion consisting of a combination of the serial and parallel methods is illustrated in Fig. 4. Certain positions of the PRBS, uniformly distributed with a period of $t$, are employed as "milestones". Let it be a position index $p = m \cdot t + r$, where $m \cdot t$ is the position of the nearest "down the track milestone" $Q(m)$ and $r$ represents the position index relative to this milestone. The natural code for $r$ is found by counting the steps required to arrive by successive back-shifts from the initial code to the nearest milestone $Q(m)$. All intermediate states of this serial shift-back operation are checked in parallel against all possible "milestone" pseudo-random patterns. Thus, with this method the code conversion of the relative position index $r$ is found serially while the milestone code conversion is done in parallel.

A hardware/time cost analysis illustrated in Fig. 5 could eventually be used to optimize the number of "milestones" (i.e., the degree of parallelism) for a specific application. The temporal cost associated with the serial back-shift matching operation increases linearly with the milestone period $t$ whereas the hardware cost is inversely proportional with $t$. The graph of combined total cost has a minimum between 1 and $2^n - 1$, and the value of $t$ that minimizes the total-cost function is:

$$t_{opt} = \left(\frac{k_1}{k_4 \cdot 2^n - 1}\right)^{1/2}$$

Fig. 6 illustrates this serial-parallel code-conversion method for the same 31-position PRBS encoded track as in Fig. 3. The four milestones $Q(0)$, $Q(1)$, $Q(2)$, and $Q(3)$ used in this case are paced with a period $t=8$ and correspond to position indices $p=0$, $p=8$, $p=16$, and $p=24$. The parallel milestone-identification module implements the following PRBS-to-natural code-conversions 00001/0000, 11101/0100, 01111/10000, and 11010/11000. The PRBS 5-tuple 11110 that corresponds to the current position index $p=18$ is back-shifted two times, $r=2$, until it reaches the state 01111, which is recognized by the parallel milestone module as $m \cdot t = 16$. This will finally yield the correct value of the absolute position index $p = m \cdot t + r = 16 + 2 = 18$.

4. Applications

Requiring only one bit of code per quantization step, the PRBS encoding method is an attractive alternative for the implementation of high-resolution absolute encoders. Most of the absolute-position encoders manufactured today are based on the transverse encoding method illustrated in Fig. 1. As a result, the resulting number of code tracks of these encoders increases proportionally with their binary
resolution. The PRBS encoding discussed in this paper has the distinct advantage of requiring only one code track for any desired resolution. This property becomes quite attractive for the implementation of very-high-resolution absolute encoders.

The PRBS encoding also provides a practical solution to the recovery of the absolute position of an AGV anywhere on the guide-path.

AGVs are driverless carriers that follow predetermined routes specified by buried cables or by floor-painted guide-paths. Due to the prohibitive complexity of the straightforward absolute encoding (Fig. 1), the existent AGVs have to rely on odometry to estimate their position. Since the errors encountered in this method are cumulative, these AGVs may eventually lose their position, which is a serious handicap in situations where they must avoid obstacles or interact with other robots, machine tools, etc. Attempts were made to compensate for this drawback by using either optical calibration methods or code labeling of specially designed locations. However, these solutions cannot be used for the recovery of vehicle's position at any point on the guide-path, and their applicability is restricted to a limited number of a priori defined reference points.

Fig. 7 shows an experimental optically guided AGV built to test the described absolute-position recovery method using PRBS encoding. It consists of a tricycle-type robot with the front wheel used for both driving and steering. A three-track guidance configuration (consisting of guide-path, synchronization track, and PRBS code track) is painted on the floor. The total length of this track layout was divided into 512 steps of equal size (q=1 cm), which resulted in a measurement resolution of n=9 bits. In order to reduce the number of code-reading heads, the 9 bits of the pseudo-random position window are not read in parallel but instead are assembled sequentially in a 9-bit shift register. A sensing board with optical reading heads for guide-path tracking (two heads), synchronization (one head), and position-code reading (one head) was specially built and attached to the front wheel. The guide-path tracking is implemented as a servo system driven by an error signal obtained by the comparison of the signals coming from the two guide-path tracking reading heads.

Fig. 8 illustrates a computer-vision method for AGV absolute-position recovery using the pseudo-random encoding. While the optically guided AGV illustrated in Fig. 7 needs three tracks (guide-path, code track, and synchronization track) the vision-based AGV needs only one track.

Two distinct 20 mm-by-30 mm graphical symbols H and E are used to mark on the floor (with a 30 mm step) the binary values 1 and 0 within the encoding 255-bit PRBS generated by an 8-bit shift register. Because of the symbol's E asymmetry it is possible to determine the orientation of the guide-
Images of this code track, provided by a video-camera mounted on the vehicle, are processed in order to recognize and locate the binary symbols. It is necessary to recognize at least 8 subsequent symbols in order to find the 8-bit window needed to recover the AGV's absolute position.

5. Conclusion

Absolute PRBS encoding discussed in this paper has the notable advantage of requiring only one code track and one code-symbol per quantization interval. This compact encoding method offers an economical solution to very-high-resolution absolute-position measurement applications involving, for example, shaft encoders, linear encoders, and AGV navigation.

Acknowledgment: This work was partially supported by the Natural Sciences and Engineering Research Council of Canada. The author thanks his colleagues F.C.A. Groen, J. Basran, N. Trif, and S.K. Yeung for their help and assistance over the years in development of various applications of the described PRBS absolute encoding.

References


Fig. 1  The straightforward approach to absolute position encoding.

\[ R(0) = R(n) \oplus c(n-1) \cdot R(n-1) \oplus \ldots \oplus c(1) \cdot R(1) \]

Fig. 2  Modulo-2 direct-feedback shift register
Table 1 Feedback equations for PRBS generation

<table>
<thead>
<tr>
<th>Shift register length $n$</th>
<th>Feedback for direct PRBS $R(0)= R(n) \oplus c(n-1) \cdot R(n-1) \oplus \ldots \oplus c(1) \cdot R(1)$</th>
<th>Feedback for reverse PRBS $R(n+1)= R(1) \oplus b(2) \cdot R(2) \oplus \ldots \oplus b(n) \cdot R(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$R(0) = R(4) \oplus R(1)$</td>
<td>$R(5) = R(1) \oplus R(2)$</td>
</tr>
<tr>
<td>5</td>
<td>$R(0) = R(5) \oplus R(2)$</td>
<td>$R(6) = R(1) \oplus R(3)$</td>
</tr>
<tr>
<td>6</td>
<td>$R(0) = R(6) \oplus R(1)$</td>
<td>$R(7) = R(1) \oplus R(2)$</td>
</tr>
<tr>
<td>7</td>
<td>$R(0) = R(7) \oplus R(3)$</td>
<td>$R(8) = R(1) \oplus R(4)$</td>
</tr>
<tr>
<td>8</td>
<td>$R(0) = R(8) \oplus R(4) \oplus R(3) \oplus R(2)$</td>
<td>$R(9) = R(1) \oplus R(3) \oplus R(4) \oplus R(5)$</td>
</tr>
<tr>
<td>9</td>
<td>$R(0) = R(9) \oplus R(4)$</td>
<td>$R(10) = R(1) \oplus R(5)$</td>
</tr>
<tr>
<td>10</td>
<td>$R(0) = R(10) \oplus R(3)$</td>
<td>$R(11) = R(1) \oplus R(4)$</td>
</tr>
</tbody>
</table>

$\mathbf{p} = 0$  
5  10 ▼  15  20  25  30  

$\mathbf{PRBS} = 0000101011101100011111001101001$

Fig. 3 Absolute position recovery on a 31-position PRBS encoded track; the 00001 window corresponds to $p=0$ and the 11000 window corresponds to $p=12$. 
\{x(k)=S(p+n-k)\mid k=n,\ldots,1\}

Pseudo-random n-tuple corresponding to the position index \( p = m \cdot t + r \)

Fig. 4 Serial-parallel PRBS to Natural-Code conversion
**Fig. 5** Serial-parallel code conversion costs as a function of the distance \( t \) between milestones. \( k_1 \) is the equipment cost associated with each milestone, \( k_2 \) is the basal hardware cost for the serial back-shift operations, \( k_3 \) is the basal temporal cost for a fully parallel solution, and \( k_4 \) is the temporal cost associated with each back-shift operation.

**Fig. 6** Serial-parallel code conversion of the absolute position \( p=18 \) on a 31-position PRBS encoded track with four milestones.
Fig. 7  Optically guided AGV on a pseudo-random encoded guide path.
Fig. 8  Vision guided AGV on a pseudo-random encoded track.