Nuclear Instruments and Methods in Physics Research A 754 (2014) 36–41

Contents lists available at ScienceDirect

Nuclear Instruments and Methods in Physics Research A

journal homepage: www.elsevier.com/locate/nima

Fast decoding algorithms for coded aperture systems

Kevin Byard

Department of Economics, Faculty of Business and Law, Auckland University of Technology, Auckland 1142, New Zealand

ARTICLE INFO

Article history:
Received 22 October 2013
Received in revised form
6 April 2014
Accepted 7 April 2014
Available online 13 April 2014

Keywords:
Coded aperture
Binary arrays
Fast algorithms

ABSTRACT

Fast decoding algorithms are described for a number of established coded aperture systems. The fast decoding algorithms for all these systems offer significant reductions in the number of calculations required when reconstructing images formed by a coded aperture system and hence require less computation time to produce the images. The algorithms may therefore be of use in applications that require fast image reconstruction, such as near real-time nuclear medicine and location of hazardous radioactive spillage. Experimental tests confirm the efficacy of the fast decoding techniques.

1. Introduction

Coded aperture imaging has become the major technique for forming images in the high energy domain [1–4]. This imaging method has been proposed and used in a number of applications, most notably high energy astronomy [4] and nuclear medicine [5], although its use has also been suggested for tracking radiation contamination [6] and flaw detection in mechanical structures [7].

In the coded aperture imaging technique, an aperture consisting of opaque and transparent elements is placed between a photon emitting source and a position sensitive detector. During observation, photons not absorbed by the opaque aperture elements pass through to the detector. The result is a shadowgram on the detector which needs to be subsequently decoded to produce a reconstructed image of the source distribution. Many patterns have been proposed for the aperture. The Fresnel zone plate [1] and the random pinhole aperture [2] have largely been superseded by patterns having perfect imaging capability, where cross-correlating the aperture with the decoding function gives a perfect delta function, although a random pattern has been used on the recent *SWIFT* mission [8]. Early examples of perfect apertures are the uniformly redundant arrays (URAs) [3] and the modified uniformly redundant arrays (MURAs) [9]. Discoveries of other perfect apertures were motivated by the desire to tackle certain problems, such as having antisymmetric apertures [10,11], self-supporting apertures [12,13] and apertures with low throughput [14,15]. Although perfect apertures are often seen as desirable, it should be noted that in a practical context other effects such as imperfections in the detector, variations in detector efficiencies and various mechanical constraints, such as artifacts in the shadowgram caused by the grid structure used to support the aperture elements, can dominate noise created by using imperfect apertures [16]. Therefore the choice of aperture is not always critical.

While the perfect apertures give good images with the image quality independent of the source distribution, cross-correlating the shadowgram with the decoding function is time consuming, particularly for systems with large numbers of aperture elements and detector pixels. Typically, if there are N elements in the unit pattern of an aperture and M pixels on a detector, decoding requires N^2M^2 multiplications, with the time required to perform this decoding being approximately proportional to the same quantity. For example the *IBIS* telescope on board the *INTEGRAL* project has $N=M=53$ [17,18] and the SAX-WFC device has $N=M\simeq 256$ [19]. Therefore the decoding time will increase very rapidly with system size. In applications where economy of time is important it therefore becomes desirable to find faster methods of reconstructing the images. Examples include near real-time imaging in nuclear medicine where fast diagnosis is required (similar studies in real time medical ultrasound imaging have been done by Heimdal et al. [20] and Choe [21]). Also speed may be required in the rapid location of radioactive sources, for example during a spillage of potentially dangerous material in order that the hazard may be dealt with urgently [6].

In addition to the aforementioned cases, there are a number of non-coded aperture applications and systems that use similar correlation techniques, and which may therefore also benefit from faster decoding. These would include applications requiring very large arrays, possible examples being channel estimation for antenna systems [22], time frequency coding [23], radar applications [24], communications [25], cryptography [26] and built-in tests for very large scale integration (VLSI) circuits [27,28].

In the case of coded aperture imaging, Roques [29] has described a fast decoding algorithm for the URAs, making use of the special properties of these systems and their decoding functions. When used, Roques' method gives a substantial time saving

http://dx.doi.org/10.1016/j.nima.2014.04.011
0168-9002/

in the decoding of images produced using a URA. However, the main drawback is that URAs are limited in terms of the parameters available to the user, in particular those related to the above-mentioned desirable characteristics. Firstly URAs are not fully antisymmetric. Although mechanical attempts have been made to utilise the partial antisymmetry of URAs [30], implementing such methods in a space application may be hazardous. Secondly, all URAs have a throughput of approximately 50% and hence offer very little flexibility in terms of this particular parameter. Finally, because the URAs have such a high throughput, they do not offer much rigidity of support, unlike the self-supporting apertures. In this paper we describe fast decoding algorithms for some of the aperture patterns which possess the characteristics described. We investigate the following. Firstly we study the square MURA configurations [9], of which a subset is antisymmetric on 90° rotation [11]. Secondly we analyse those configurations described by the author that are created from products of individual one-dimensional coded aperture systems [14]. For ease of discussion, we refer to these as *product* apertures. The product apertures have low throughput and are all fully self-supporting, special cases of which are the pseudo noise product (PNP) arrays [12] and the M–P and M–M arrays [13].

2. Standard image reconstruction

In many cases, a coded aperture system based on a rectangular geometry is defined by a *unit pattern* of size $v \times w$ elements [3,9,14]. Exposure to the source typically generates a detector shadowgram, $P(i,j)$, where $0 \le i \le v-1$ and $0 \le j \le w-1$. The unit pattern of the decoding function is represented by $G(i,j)$ with $0 \le i \le v-1$ and $0 \le j \le w-1$. Often the full decoding function is represented by a $2v-1$ by $2w-1$ repetition of G [3,29], although in this paper we use here the modulo v and w form of the function G for ease of notation. The reconstructed image is given by

$$I(k,l) = \sum_{i=0}^{v-1} \sum_{j=0}^{w-1} P(i,j)G(i+k,j+l) \tag{1}$$

where $i+k$ is taken modulo v, $j+l$ is taken modulo w, $0 \le k \le v-1$ and $0 \le l \le w-1$. In the standard reconstruction method this double summation is completed in its entirety using a computing device. Because of the two summations over v and w terms, reconstruction of the function I therefore requires v^2w^2 multiplications, which becomes very large for large systems.

In any fast reconstruction technique, we attempt to circumvent the necessity of calculating every term in either or both the summations in Eq. (1). This can be achieved by taking advantage of the special way the decoding function is created. In his paper Roques used the special properties of the URAs to describe a fast decoding algorithm for these systems. In the next two sections we describe fast decoding algorithms for the square MURAs and the product apertures, of which the PNP, M–P and M–M apertures are special cases.

3. Square MURA apertures

The MURA apertures were introduced by Gottesman and Fenimore [9]. In their paper they describe two different MURA types, in which they mosaic onto either a linear, hexagonal or a square configuration. We here demonstrate that images created using the square configuration MURAs can be decoded using a fast algorithm. The basic idea is very similar to that of the fast decoding algorithm of URAs [29]. Let p be an odd prime and define C_p as follows:

$$C_p(i) = \left(\frac{-1}{p}\right)\left(\frac{i}{p}\right) \tag{2}$$

where (i/p) is the Legendre symbol for p and $0 \le i \le p-1$. We now recall that for a square MURA of side p elements where p is an odd prime, the unit pattern of the decoding function G is given by the following:

$$G(i,j) = \begin{cases} -1 & \text{if } i=0, j \ne 0 \\ 1 & \text{if } j=0 \\ C_p(i)C_p(j) & \text{otherwise} \end{cases} \tag{3}$$

where $0 \le i,j \le p-1$. Substituting the various values of $G(i,j)$ from Eq. (3) into Eq. (1) and rearranging gives

$$I(k,l) = \sum_{\substack{i=0 \\ i \ne -k}}^{p-1} C_p(i+k) \sum_{\substack{j=0 \\ j \ne -l}}^{p-1} P(i,j)C_p(j+l) - \sum_{\substack{j=0 \\ j \ne -l}}^{p-1} P(-k,j) + \sum_{i=0}^{p-1} P(i,-l). \tag{4}$$

The resulting expression for I is virtually identical to that for the URAs as given by Roques [29], the only difference being the nature of the signs in the second and third terms of the right hand side of Eq. (4). The methods used by Roques are therefore directly applicable to the MURAs and so any time saving gained when using the MURAs is equivalent to that quantified by Roques [29].

It is interesting to note that Gottesman and Fenimore predicted the possible existence of a fast decoding algorithm for the square MURAs, due to the inherent symmetry of these systems [9, p. 4352]. A point that makes this development particularly important is the fact that a subset of the square MURAs has been shown to be antisymmetric on 90° rotation [11]. Antisymmetric apertures are useful in the removal of systematic spatial variation in the detector background [10,31]. Therefore with a slight modification, the method described above can be used and so there exists a type of antisymmetric aperture for which fast decoding can be employed.

4. Product apertures

The product apertures are synthesised using the products of single coded aperture systems, called *primitive* systems [14]. The PNP [12], M–P and M–M [13] apertures are all subsets of the product apertures. Although some primitive systems can have orders that are non-prime (for example a PN sequence can be of order 63), because of the methods used, we are concerned here with primitive systems that have prime orders. Let G be the decoding function of a product system, composed of n primitive systems of orders $p_1, p_2, \ldots, p_n$, where the p_x are all prime. We do not need to define the dimensionality of the coded aperture system being used, since the theory described below applies to all cases, including systems with more than two dimensions. Note, however, that while the p_x do not necessarily have to be distinct, if there are m dimensions to a product system, there are at most m values of p_x that can be equal, and only then if there is no more than one primitive system of order p_x used in any one dimension.

We define the set of n functions C_x by the following two equations:

$$C_x(0) = 1 \quad \text{for all } x \tag{5}$$

$$C_x(i) = \left(\frac{-1}{p_x}\right)\left(\frac{i}{p_x}\right) \quad \text{if } i \ne 0 \tag{6}$$

where $x = 1, 2, \ldots, n$. The decoding function of a product aperture is given by

$$G(i_1, i_2, \ldots, i_n) = \prod_{x=1}^{n} C_x(i_x) \tag{7}$$

where $i_x = 0, 1, \ldots, p_x - 1$ [14]. Note that setting $n=2$ in Eq. (7) gives either a PNP, M–P or M–M aperture, depending on the choice of the primes p_x. Following an observation of a source field, let P be the resulting detector shadowgram. The reconstructed image I is given by

$$I(k_1, k_2, \ldots, k_n) = \sum_{i_1=0}^{p_1-1} \sum_{i_2=0}^{p_2-1} \cdots \sum_{i_n=0}^{p_n-1} P(i_1, i_2, \ldots, i_n) \times G(i_1+k_1, i_2+k_2, \ldots i_n+k_n) \tag{8}$$

where $0 \le k_x \le p_x - 1$ and the $i_x + k_x$ are taken modulo p_x. Comparing Eq. (8) to Eq. (1) we can see that here, if standard decoding is to be used, the number of required multiplications N_s is given by

$$N_s = \prod_{x=1}^{n} p_x^2. \tag{9}$$

This quantity becomes very large for large values of p_x.

4.1. Main optimisation algorithm

We now describe the main optimisation algorithm. It is so called because it gives by far the largest time saving of the two optimisations and so its full description is necessary. Also its understanding is important when also employing the secondary optimisation, which is described in Section 4.2.

The main optimisation algorithm arises as a result of the special way in which the decoding function G is calculated in Eq. (7). Substituting for G from Eq. (7) into Eq. (8) and rearranging gives

$$I(k_1, k_2, \ldots, k_n) = \sum_{i_1=0}^{p_1-1} C_1(i_1+k_1) \sum_{i_2=0}^{p_2-1} C_2(i_2+k_2) \ldots \times \sum_{i_n=0}^{p_n-1} P(i_1, i_2, \ldots, i_n) C_n(i_n+k_n). \tag{10}$$

The partitioning of G into the individual C_x enables fewer multiplications to be performed and hence shortens the required computing time. The continuation is an extension of the strategy employed by Roques for the URAs. Consider the last summation in Eq. (10). We denote this by

$$Q_n(i_1, i_2, \ldots, i_{n-1}, k_n) = \sum_{i_n=0}^{p_n-1} P(i_1, i_2, \ldots, i_n) C_n(i_n+k_n). \tag{11}$$

For each i_x, $x \ne n$, to calculate function Q_n requires p_n multiplications, making a total of $p_n \prod_{x=1}^{n} p_x$ multiplications to evaluate the whole of Q_n. Once Q_n has been calculated we can now employ it in Eq. (10) to give

$$I(k_1, k_2, \ldots, k_n) = \sum_{i_1=0}^{p_1-1} C_1(i_1+k_1) \sum_{i_2=0}^{p_2-1} C_2(i_2+k_2) \ldots \times \sum_{i_{n-1}=0}^{p_{n-1}-1} Q_n(i_1, i_2, \ldots, i_{n-1}, k_n) \times C_{n-1}(i_{n-1}+k_{n-1}). \tag{12}$$

We can continue in the same fashion as above by taking the last summation in Eq. (12) and setting it to

$$Q_{n-1}(i_1, i_2, \ldots, i_{n-2}, k_{n-1}, k_n) = \sum_{i_{n-1}=0}^{p_{n-1}-1} Q_n(i_1, i_2, \ldots, i_{n-1}, k_n) C_{n-1}(i_{n-1}+k_{n-1}) \tag{13}$$

which requires $p_{n-1} \prod_{x=1}^{n} p_x$ multiplications to evaluate Q_{n-1}. It is evident that we can continue this induction, with a general expression Q_y being given by

$$Q_y(i_1, i_2, \ldots, i_{y-1}, k_y, k_{y+1}, \ldots, k_n) = \sum_{i_y=0}^{p_y-1} Q_{y+1}(i_1, i_2, \ldots, i_y, k_{y+1}, \ldots, k_n) C_y(i_y+k_y) \tag{14}$$

with Q_y requiring $p_y \prod_{x=1}^{n} p_x$ multiplications. Note from Eqs. (14) and (11) that $Q_{n+1} = P$. The final stage of the induction will give

$$Q_1(k_1, k_2, \ldots, k_n) = I(k_1, k_2, \ldots, k_n) = \sum_{i_1=0}^{p_1-1} Q_2(i_1, k_2, k_3, \ldots, k_n) C_1(i_1+k_1) \tag{15}$$

which yields the final reconstructed image I. The entire process requires a total of $(\sum_{x=1}^{n} p_x)(\prod_{x=1}^{n} p_x)$ multiplications, which gives a reduction in the number of required multiplications of $(\prod_{x=1}^{n} p_x)/(\sum_{x=1}^{n} p_x)$ compared to that of the standard deconvolution method – a significant saving. As an example, if $n=3$ and $p_1 = 5$, $p_2 = 7$ and $p_3 = 11$, the number of required multiplications is reduced by a factor of 16.7.

4.2. Secondary optimisation

The special properties of the Legendre symbol can be used to achieve a further optimisation of the decoding algorithm of the product apertures, in an idea that is similar to that used by Roques in the case of the URAs [29]. We refer to this as *secondary* optimisation. We here utilise the fact that consecutive members of the $C_i(x)$ in Eq. (6) are sometimes equal. This point can be used to achieve a further time saving as follows. During the induction method of the main optimisation algorithm in Section 4.1, consider a point in the induction given by Eq. (14). Using this equation we can write

$$\begin{aligned} Q_y(i_1, i_2, \ldots, i_{y-1}, k_y, k_{y+1}, \ldots, k_n) &= Q_y(i_1, i_2, \ldots, i_{y-1}, k_y - 1, k_{y+1}, \ldots, k_n) \\ &+ \sum_{i_y=0}^{p_y-1} Q_{y+1}(i_1, i_2, \ldots, i_y, k_{y+1}, \ldots, k_n) \\ &\times [C_y(i_y+k_y) - C_y(i_y+k_y-1)] \end{aligned} \tag{16}$$

and so, starting in sequence with $Q_y(i_1, i_2, \ldots, i_{y-1}, 0, k_{y+1}, \ldots, k_n)$, once $Q_y(i_1, i_2, \ldots, i_{y-1}, k_y - 1, k_{y+1}, \ldots, k_n)$ has been calculated then $Q_y(i_1, i_2, \ldots, i_{y-1}, k_y, k_{y+1}, \ldots, k_n)$ can also be calculated. However, upon studying (16) it is clear that if the quantity $C_y(i_y+k_y) - C_y(i_y+k_y-1)$ is zero, then it becomes unnecessary to perform a summation to calculate $Q_y(i_1, i_2, \ldots, i_{y-1}, k_y, k_{y+1}, \ldots, k_n)$, thus reducing the overall number of multiplications required. This point was also used by Roques for the URAs [29]. For all primes p_y, the number h_y of times the quantity $C_y(i_y+k_y) - C_y(i_y+k_y-1)$ is nonzero under the criteria of Eqs. (5) and (6) is given by

$$h_y = \frac{p_y - (-1/p_y)}{2} \tag{17}$$

where $(-1/p_y)$ is the Legendre symbol of -1 with respect to the prime p_y. Therefore, for large p the number of multiplications required to perform each summation is reduced by approximately a half.

When combining the main and secondary optimisation methods the time saving is significant, particularly for systems with large n. If both optimisation methods are used the required total number of multiplications N_f is given by

$$N_f = \left[n + \sum_{i=1}^{n} \frac{(p_i - 1) h_i}{p_i} \right] \prod_{x=1}^{n} p_x. \tag{18}$$

This compares to the number of multiplications for the standard decoding given in Eq. (9).

Examples of the reduction in the number of required multiplications for some configurations are given in Table 1, where the ratio N_f/N_s is given for a configuration of n primitive systems, each of

Table 1
Ratio N_f/N_s for n primitive systems, each of order p.

p	n			
	2	3	4	5
5	0.21	0.062	0.017	4.2×10^{-3}
11	0.11	0.015	1.7×10^{-3}	2.0×10^{-4}
23	0.047	3.1×10^{-3}	1.8×10^{-4}	9.7×10^{-6}
53	0.019	5.3×10^{-4}	1.3×10^{-5}	3.2×10^{-7}
101	9.9×10^{-3}	1.5×10^{-4}	1.9×10^{-6}	2.4×10^{-8}

order p. The table shows that in all cases there is a significant reduction in the number of multiplications required. For small systems, with $p_x\lesssim11$ and/or $n\lesssim3$, reductions of approximately one to two orders of magnitude are available. For medium to large systems the reductions are of several orders of magnitude.

5. PNP, M–P and M–M apertures

We demonstrate the methods of the previous sections by considering the specific cases of the PNP [12], M–P and M–M [13] apertures. These will occur when $n=2$ with the orders of the two primitive systems being p_1 and p_2. Using these parameters with Eq. (10) gives

$$I(k_1,k_2)=\sum_{i_1=0}^{p_1-1}C_1(i_1+k_1)\sum_{i_2=0}^{p_2-1}P(i_1,i_2)C_2(i_2+k_2). \tag{19}$$

Note that the number of multiplications to calculate I for all k_1 and k_2 has been reduced from $p_1^2p_2^2$ to $p_1p_2(p_1+p_2)$, meaning that we have achieved the saving of the main optimisation algorithm of Section 4.1. At this point we use Eq. (11) to give

$$Q_2(i_1,k_2)=\sum_{i_2=0}^{p_2-1}P(i_1,i_2)C_2(i_2+k_2) \tag{20}$$

which is the representation of the last summation in Eq. (19). When calculating Q_2 in Eq. (20) we utilise the secondary optimisation by first calculating $Q_2(i_1,0)$ and then only calculating subsequent $Q_2(i_1,k_2)$ if $C_2(i_2+k_2)-C_2(i_2+k_2-1)\neq0$. By Eq. (17) this will be $h_2=[p_2-(-1/p_2)]/2$ times, with a saving of approximately a half over that of using the first optimisation only.

Once $Q_2(i_1,k_2)$ has been calculated as a function of i_1 we combine Eqs. (19) and (20) to give

$$I(k_1,k_2)=Q_1(k_1,k_2)=\sum_{i_1=0}^{p_1-1}C_1(i_1+k_1)Q_2(i_1,k_2) \tag{21}$$

which is the representation of Eq. (15) for $n=2$. Again, we use the secondary optimisation, calculating $Q_1(0,k_2)$ and then only calculating subsequent $Q_1(k_1,k_2)$ if $C_1(i_1+k_1)-C_1(i_1+k_1-1)\neq0$, which only occurs $h_1=[p_1-(-1/p_1)]/2$ times. Completing the summations for Eq. (21) gives the final reconstructed image.

Combining the induction method of Section 4.1 with the secondary optimisation method of Section 4.2 we calculate from Eqs. (17) and (18) that the total number of multiplications required is

$$N_{f2}=2p_1p_2+\frac{p_1(p_2-1)[p_2-(-1/p_2)]}{2}+\frac{p_2(p_1-1)[p_1-(-1/p_1)]}{2} \tag{22}$$

which using Eq. (9) compares with the number required for the standard decoding of

$$N_{s2}=p_1^2p_2^2. \tag{23}$$

Comparing the ratio N_{f2}/N_{s2} for various values of $p_1=p_2$ in Eqs. (22) and (23) yields the values in the second column of Table 1.

6. Time factors

The previous sections discuss the considerable reduction in the number of multiplications required when employing both the main and secondary optimisation methods over that of the standard method. However, the simple counting of the numbers of multiplications and comparing N_f to N_s only partly explains the observed computational speed up. This is because some multiplications require slightly different procedures to others. For this reason, it is convenient to divide Eq. (18) into two separate quantities as follows. At the commencement of the secondary optimisation algorithm, calculating the starting point $Q_y(i_1,i_2,\ldots,i_{y-1},0,k_{y+1},\ldots,k_n)$, which for ease of notation we shall hereafter refer to as Q_y^0, using Eq. (14) requires $\prod_{x=1}^{n}p_x$ multiplications for a given y and hence the following number of multiplications for all $1\leq y\leq n$:

$$N_0=n\prod_{x=1}^{n}p_x. \tag{24}$$

Now, calculating the rest of $Q_y(i_1,i_2,\ldots,i_{y-1},k_y,k_{y+1},\ldots,k_n)$, hereafter referred to as Q_y^k, for all k_y and all y using Eq. (16) for those cases when $C_y(i_y+k_y)-C_y(i_y+k_y-1)\neq0$ requires the following number of multiplications:

$$N_k=\left[\sum_{i=1}^{n}\frac{(p_i-1)h_i}{p_i}\right]\prod_{x=1}^{n}p_x \tag{25}$$

where $N_0+N_k=N_f$.

We first study the calculation of Q_y^0. Here, Eq. (14) with $k_y=0$ shows that a number of multiplications are required to be performed, as indeed are similarly required to calculate I using the standard method of Eq. (8). Study of Eq. (14) shows that each of the N_0 multiplications to calculate the Q_y^0 requires, among other things, the interrogation of the one-dimensional vector C_y, which has an argument of $i_y+k_y=i_y$, since $k_y=0$. This interrogation takes a small amount of time before the actual multiplication. However, in the case of the calculation of I in Eq. (8), it is the n-dimensional array G that is required to be interrogated, which takes longer than the interrogation of a one-dimensional vector such as C_y. In addition each multiplication in Eq. (8) suffers further from the requirement to calculate not one, but n arguments $i_1+k_1, i_2+k_2, \ldots, i_n+k_n$, prior to each multiplication. Now, while the calculations of the i_x+j_x can be optimised by nesting them outside some of the summation loops, both of the above mentioned processes nevertheless result in a larger overall time cost, and hence a reduction in the observed number of multiplications per second in the calculation of I in Eq. (8) as compared to the calculation of Q_y^0 in Eq. (14).

Regarding the calculations of the remaining Q_y^k, study of Eq. (16) shows that it is still necessary to interrogate C_y, and indeed this is required twice to obtain $C_y(i_y+k_y)-C_y(i_y+k_y-1)=D_y$, which also requires calculation of the two arguments i_y+k_y and i_y+k_y-1. However, the calculation of D_y is required only once before being stored and used to calculate the remainder of $\sum_{i_y=0}^{p_y-1}Q_{y+1}(i_1,i_2,\ldots,i_y,k_{y+1},\ldots,k_n)\times D_y$ in Eq. (16). Using Eq. (17), there are $\sum_{y=1}^{n}h_y$ nonzero values of D_y for all $1\leq y\leq n$, and each nonzero D_y serves in the performance of $(\prod_{x=1}^{n}p_x)/p_y$ multiplications during the entire calculation of $Q_1=I$. In all the total number of multiplications required is N_k whereas the total number of interrogations of the C_y vectors, with arguments, is only

$$N_i=2\sum_{y=1}^{n}p_y(p_y-1). \tag{26}$$

Therefore the required number of interrogations of the vectors C_y, including the calculations of their arguments, is reduced by the factor N_i/N_k compared to that of calculating the Q_y^0 and so the number of multiplications per second to calculate the Q_y^k is higher

Table 2
Ratio calculations for $n=3$ and $p_1=p_2=p_3=p$.

p	N_f/N_s	N_i/N_k
5	0.062	0.2
7	0.039	0.071
11	0.015	0.030
13	8.9×10^{-3}	0.026
17	5.2×10^{-3}	0.015
19	4.6×10^{-3}	0.011
23	3.1×10^{-3}	0.007
29	1.8×10^{-3}	0.005
31	1.7×10^{-3}	0.004
37	1.1×10^{-3}	0.003
41	8.9×10^{-4}	0.002
43	8.5×10^{-4}	0.002

than that of the Q_y^0 which is in turn higher than that of the standard method of Eq. (8). As an example, if $n=3$ and $p_1=5$, $p_2=7$ and $p_3=11$, the ratio N_i/N_m is 0.085. N_i/N_m decreases with higher values of p_y. Table 2 shows the ratios of multiplications and interrogations using $n=3$ and $p_1=p_2=p_3=p$, where N_s and N_f are calculated using Eqs. (9) and (18) respectively. In this table both the ratios N_f/N_s and N_i/N_k decrease with p which will lead to a significant speed up in the computation with increasing p.

Once all the multiplications to calculate Q_y for all y have been completed, it only remains to then add $Q_y(i_1,i_2,\ldots,i_{y-1},k_y-1,k_{y+1},\ldots,k_n)+\sum_{i_y=0}^{p_y-1}Q_{y+1}(i_1,i_2,\ldots,i_y,k_{y+1},\ldots,k_n)\times[C_y(i_y+k_y)-C_y(i_y+k_y-1)]$ for $1\le y\le n$ as per Eq. (16). This final addition entails an amount of extra time T_a outside of the multiplication processes, but this is very small in comparison to the rest of the computation of I, as demonstrated in the next section.

7. Experimental tests

Experimental tests were conducted to investigate the time savings achievable by the fast decoding algorithm in a practical context. We study the case $n=3$, with $p_1=p_2=p_3=p$, measuring the time taken to complete the decoding for both the standard and fast algorithms, assuming both the main and secondary optimisations are used. The tests were done using Mathematica 8 on Windows 7 Enterprise, with a 3.2 GHz processor, a RAM of 8 GB and a 64 bit operating system. The results of the tests for primes $5\le p\le43$ are given in Table 3, where T_s and T_f were the measured computational times required to complete the standard and fast decoding respectively. Also shown in the table are the observed numbers of multiplications per second using the standard method of Eq. (8), and when calculating Q_y^0 using Eq. (14) and Q_y^k using Eq. (16). The extra time required for the addition part of Eq. (16), T_a, is also given.

The results indicate that the fast decoding algorithm does afford a significant time saving when employed practically over the standard decoding, due mainly to the reduced number of multiplications required when using the fast method. In addition, however, the observed numbers of multiplications per second is also shown to increase as we move from the standard method to fast decoding, with the calculations for Q_y^k being even more rapid than those for Q_y^0, as explained in the discussion in Section 6. Using the fast method does require the extra time T_a to perform a small matrix addition as per Eq. (16), but Table 3 shows that this was observed to be very small in all cases. Therefore, in practice, the time savings are more optimistic than the ratio of N_f/N_s would indicate, as shown by the final column in Table 3. For the case of these particular parameters, the total further saving

Table 3
Results of experimental tests for $n=3$ and $p_1=p_2=p_3=p$. The times indicated are in seconds.

p	T_s	T_f	T_f/T_s	Multiplications per second ($\times10^3$)			T_a	$\frac{T_f/T_s}{N_f/N_s}$
				Standard	Q_y^0	Q_y^k		
5	0.05	0	0	340	–	–	0	0
7	0.33	0	0	360	–	–	0	0
11	4.65	0.08	0.02	381	125	473	0	1.15
13	13	0.09	7.4×10^{-3}	382	412	468	0	0.83
17	63	0.23	3.7×10^{-3}	383	–	547	0.031	0.71
19	123	0.45	3.7×10^{-3}	382	447	543	0.048	0.80
23	390	0.95	2.4×10^{-3}	379	380	546	0.061	0.79
29	1552	2.11	1.4×10^{-3}	383	472	542	0.126	0.76
31	2360	2.95	1.2×10^{-3}	376	478	531	0.157	0.75
37	6763	5.54	8.2×10^{-4}	379	463	538	0.265	0.75
41	12 443	8.26	6.6×10^{-4}	382	457	541	0.359	0.74
43	16 558	10.47	6.3×10^{-4}	382	478	537	0.421	0.75

amounts to approximately 25% over and above that calculated from Eqs. (9) and (18) alone for the parameters studied. The results for small p contain some anomalies which we might expect when dealing with very short time measurements. For example, the dashes in Table 3 are due to the presence of reported times of zero for some cases, which will evidently not be the case in reality. The results tend to 'settle down' for larger values of p, converging on the 25% extra time saving as described above. As an example of a real telescope, the time taken for one decoding of an image obtained by the *IBIS* instrument on the *INTEGRAL* satellite [17,18] using the same computer system took 21.2 s using the standard decoding technique, but only 0.4 s using the proposed method.

It is clear that the use of Fast Fourier Transforms to decode the images can also yield fast results. In the examples given, FFTs were also applied and in each case the same results were reproduced using less than 1 s of computing time. However the method described provides a transparent alternative which is simpler, preserves the linear cross-correlation approach and, in the context of real coded aperture systems of current size, offers decoding times that are both many orders of magnitude faster than standard decoding and which are also competitive with the FFTs – the longest decoding of Table 3 took less than 11 s.

8. Conclusions

Following Roques' work with URAs [29], this paper presents fast decoding algorithms for a number of other coded aperture systems, namely the square configuration MURAs, which includes the antisymmetric MURAs, the product apertures and the PNP, M–P and M–M apertures. The technique, which utilises the special methods by which the various configurations are constructed, reduces significantly the number of calculations required when they are employed in coded aperture systems. Results indicate that for small configurations with the $p_x\lesssim11$ and/or $n\lesssim3$, savings of approximately one to two orders of magnitude are available, and for medium to large systems the available savings are of several orders of magnitude.

Experimental tests not only confirm the efficacy of the fast decoding technique but also verify the further practical improvement above that indicated by a simple counting of the number of calculations given in Table 1, as discussed in Section 6. These extra improvements arise for two reasons. Firstly, the part of the fast decoding technique that calculates Q_y^0 scores over the standard

method by virtue of the use of one-dimensional vectors for the decoding, instead of the n dimensions of the standard method. For Q_y^0, it is only necessary to calculate one argument, which is then used to interrogate only a one-dimensional vector, whereas with the standard method, it is necessary prior to each multiplication to calculate n arguments, which in addition requires the more lengthy interrogation of an n-dimensional array. Although some of these problems could be alleviated by more optimal programming, there is still a significant decrease in the required number of multiplications per second when calculating Q_y^0 using the fast method. Secondly, when calculating Q_y^k, the need to calculate arguments and interrogate the vector $C_y(i_y+k_y)$ compared to Q_y^0 is reduced by a factor of N_i/N_k, which results in an even higher number of multiplications per second, as shown in Table 3. When performing the experimental tests for the particular aperture parameters studied, the overall time saving settles down to approximately 25% for system sizes that exceed the anomalous smaller values of p. In all cases the validity of the fast decoding method is confirmed and for system sizes of the order of those real coded aperture instruments currently in use [18,19], the results are competitive with those of Fast-Fourier Transforms, and hence offer a viable alternative that is simple and transparent.

The techniques described clearly increase the number of coded aperture systems that can benefit from fast decoding. However, of no less import is the point that the methods also widen the range of possible system parameters that can now benefit from fast decoding, such as the aperture symmetry, shape and throughput. Therefore fast decoding is rendered available for some systems that possess special properties not enjoyed by the URAs. Because the technique is now available to the square MURAs [9], a subset of which has been shown to be antisymmetric on 90° rotation [11], fast decoding is now available if an antisymmetric aperture is desired. Also, because the technique applies to the self-supporting PNP, M–P or M–M apertures, it is possible to use fast decoding while enjoying the benefits of these self-supporting apertures, such as rigidity of structure [12] or active collimation [32]. Note also that these latter systems possess a much greater range of eccentricity of rectangular pattern than that available to the almost square URAs. Finally, the product apertures typically have low throughput values [14]. For applications working in a low-background environment, such as nuclear medicine imaging, it may not be necessary to have an aperture with a large throughput [33], and often a low throughput aperture may be selected due to its simplicity of design, for which fast decoding now becomes available.

The fast decoding techniques may be of use in any coded aperture application requiring fast image reconstruction. Examples include near real-time nuclear medicine imaging and the detection and the rapid location of hazardous material, such as radioactive spillage. Also, it should always be borne in mind that there are a number of non-coded aperture applications where the fast cross-correlation of binary arrays is desirable. These potentially include the use of antenna systems, radar applications, time frequency coding, communications, cryptography and built-in tests for VLSI circuits.

Acknowledgements

The author wishes to thank Dr. Shaun Cooper for his help in writing this paper and Jordan Alexander for some excellent comments and suggestions.

References

[1] L. Mertz, N.O. Young, Fresnel transformations of images, in: K.J. Habell (Ed.), Proceedings of the International Conference on Optical Instruments and Techniques, Chapman and Hall, London, UK, 1961, pp. 305–310.
[2] R.H. Dicke, The Astrophysical Journal Letters 153 (1968) L101.
[3] E.E. Fenimore, T.M. Cannon, Applied Optics 17 (1978) 337.
[4] E. Caroli, J.B. Stephen, G. Di Cocco, L. Natalucci, A. Spizzichino, Space Science Review 45 (1987) 349.
[5] R. Accorsi, F. Gasparini, R. Lanza, IEEE Transactions on Nuclear Science NS-48 (2001) 2411.
[6] M. Woodring, D. Beddingfield, D. Souza, G. Entine, M. Squillante, J. Christian, A. Cogan, Nuclear Instruments and Methods in Physics Research Section A 505 (2003) 415.
[7] S. Thangavelu, M.A. Hussein, Applied Radiation and Isotopes 65 (2006) 189.
[8] N. Gehrels, G. Chincarini, P. Giommi, K.O. Mason, J.A. Nousek, A.A. Wells, N.E. White, S.D. Barthelmy, et al., The Astrophysical Journal 611 (2004) 1005.
[9] S.R. Gottesman, E.E. Fenimore, Applied Optics 28 (1989) 4344.
[10] M.H. Finger, T.A. Prince, Hexagonal uniformly redundant arrays for coded-aperture imaging, in: F.C. Jones (Ed.), Proceeding of the 19th International Cosmic Ray Conference, vol. 3, Scientific and Technical Information Branch, NASA, Washington, 1985, pp. 295–298.
[11] K. Byard, Experimental Astronomy 2 (1992) 227.
[12] S.R. Gottesman, E.J. Schneid, IEEE Transactions on Nuclear Science NS-33 (1986) 745.
[13] K. Byard, Nuclear Instruments and Methods in Physics Research Section A 322 (1992) 97.
[14] K. Byard, Nuclear Instruments and Methods in Physics Research Section A 336 (1993) 262.
[15] K. Byard, Applied Optics 51 (2012) 3453.
[16] G.K. Skinner, Applied Optics 47 (2008) 2739.
[17] A. Goldwurm, P. Goldoni, A. Gros, J. Stephen, L. Foschini, F. Gianotti, L. Natalucci, G. De Cesare, M. Del Santo, Gamma-ray imaging with the coded mask IBIS telescope, in: A. Gimenez, V. Reglero, C. Winkler (Eds.), Exploring the Gamma-Ray Universe, Proceedings of 4th INTEGRAL Workshop, ESA SP-459, 2001, pp. 497–500.
[18] A. Goldwurm, P. David, L. Foschini, A. Gros, P. Laurent, A. Sauvageon, A.J. Bird, L. Lerusse, N. Produit, Astronomy and Astrophysics Letters 411 (2003) 223.
[19] J.J.M. in't Zand, J. Heise, R. Jager, Astronomy and Astrophysics 288 (1994) 665.
[20] A. Heimdal, A. Stoylen, H. Torp, T. Skjaerpe, Journal of the American Society of Echocardiography 11 (1998) 1013.
[21] J.W. Choe, IEEE Transactions on Medical Imaging 32 (2013) 1258.
[22] S. Yang, J. Wu, IEEE Transactions on Vehicular Technology 51 (1986) 1271.
[23] S.W. Golomb, H. Taylor, IEEE Transactions on Information Theory 28 (1982) 600.
[24] G. Weathers, E.M. Holliday, IEEE Transactions on Aerospace and Electronic Systems 19 (1983) 369.
[25] T. Kohda, H. Aihara, Binary sequences using chaotic dynamics and their applications to communications, in: T. Hikihara (Ed.), IUTAM Symposium on 50 Years of Chaos: Applied of Theoretical, vol. 5, Elsevier B.V., Red Hook, New York, USA, 2012, pp. 46–48.
[26] C. Hernandez-Goya, A. Fuster-Sabater, Balancedness in binary sequences with cryptographic applications, in: Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science, vol. 4967, 2008, pp. 499–508.
[27] P.H. Bardell, W.H. McAnney, IEEE Transactions on Computers 35 (1986) 653.
[28] C.C. Wang, IEEE Transactions on Computers 39 (1990) 258.
[29] J.P. Roques, Applied Optics 26 (1987) 3862.
[30] U.B. Jayanthi, J. Braga, Nuclear Instruments and Methods in Physics Research Section A 322 (1991) 685.
[31] W.R. Cook, M. Finger, T.A. Prince, E.C. Stone, IEEE Transactions on Nuclear Science NS-31 (1984) 771.
[32] A. Johansson, B.L. Beron, L. Campbell, R. Eichler, P. Gorodetsky, R. Hofstadter, E.B. Hughes, S. Wilson, IEEE Transactions on Nuclear Science NS-27 (1980) 375.
[33] A.R. Gourlay, J.B. Stephen, Applied Optics 22 (1983) 4042.