Modulo Transforms – an Alternative to Lifting

Sridhar Srinivasan
Microsoft Digital Media Division
sridhsri@microsoft.com

Abstract — **This paper introduces a new paradigm for the construction of reversible transforms that map integers to integers. Transform matrices with integer entries are first considered, and the modular arithmetic properties of transform coefficients is studied. It is shown that these transform coefficients are redundant in modular arithmetic. Further, this redundancy can be exploited by quantizing transform coefficients in a manner so as to produce an effective scaled transformation with unit determinant. The concept of *critical quantization* is introduced and forms the basis of a class of transforms referred to as *modulo transforms* that are reversible, effectively have rational coefficients and unit determinant. These conditions are necessary for applications such as lossless compression and reversible image rotation.**

The theory of modulo transforms is examined in depth. Analysis based on modular arithmetic shows that 2D rotations can be critically quantized, albeit with non-equal bin widths along axes. A construction procedure is derived for realizing a reversible transform with small or unit scaling factors. Further, it is shown that modulo transforms based on certain Pythagorean triples can be scalar quantized to produce a reversible, normalized, scale-free transform matrix. This theory is built on 2D rotations, and extended to larger transforms such as the 4 and 8 point DCT.

Modulo transforms and lifting are compared and contrasted in theory, and in experiments. The computational aspects of modulo transforms are also discussed in this paper.

Index Terms—**Transforms, Coding, Reversibility, Lifting, Modular Arithmetic, Number Theory**

I. Introduction

TRANSFORM domain representation of a signal finds widespread usage in digital signal processing. Integer transforms are defined as transforms that map integers to integers. Integer transforms are used in the lossless compression of integer or integer-indexed data where the original signal is perfectly recoverable from its compressed domain representation. For the compression of typical time sequences, images and video, integer transforms are chosen so as to closely approximate an efficient linear transform such as the discrete cosine transform (DCT) or Karhunen-Loeuve Transform (KLT), both of which are known to be effective for data compression on account of their energy compaction properties [1][2]. Other integer transforms include differential pulse code modulation (DPCM) or simple linear predictive coding, and the Hadamard transform defined as the pair of sum and difference of its two inputs. Non-trivial integer transforms are most often implemented using ladder structures, commonly referred to as "lifting"[1]. It has been shown that lifting techniques can approximate arbitrary linear transforms, including DCT and wavelets [3][4][5]. Lifting does suffer from a few drawbacks, including latency due to cascading stages, incremental rounding errors, and possibly inefficient implementations.

[1] Although the term lifting was originally used to denote the process of increasing the number of vanishing moments in wavelet transforms implemented with ladder networks.

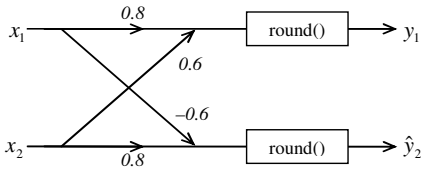


Figure 1: An example of a modulo transform – this is actually reversible!

In this paper, we introduce an alternative mechanism to lifting for implementing orthogonal linear transforms with a reversible approximation. This class of transforms is built on a foundation of modular arithmetic; these transforms are therefore called *modulo transforms*. The analysis of modulo transforms requires the development of a new theory of quantization of transform coefficients that we call *critical quantization of integer transforms*. An example of a modulo transform is shown in Figure 1. Although the transform looks like a conventional 2D rotation butterfly, it is actually reversible for integer input – integer output. The proof of this statement and the development of theory leading to it follow in the remainder of this paper. Our intent here is to present modulo transforms as an interesting theoretical innovation and to provide some evidence showing reduced rounding errors compared to lifting techniques. We recognize some computational issues (primarily the need to perform an integer division) and we propose some solutions that address these in an efficient manner.

In the next section, we review background material on reversible and efficient linear transforms. We provide some definitions and outline properties of critical quantization in Section III, upon which we build the theory of modulo transforms in Section IV. We detail a construction procedure in Section V, state Theorem 6 which proves the reversibility of the structure in Figure 1, and show examples of reversible 2 point rotations. We extend modulo transforms to larger matrices in Section VI, showing examples of reversible 4 and 8 point DCTs. We present some results in Section VII and discuss issues of computational complexity in Section VIII. Conclusions are provided in Section IX, and a complete proof of Theorem 6 in Appendix A.

II. LINEAR TRANSFORMS

A linear transform $T \in \mathrm{R}^{N \times N}$ applied to the data vector $x \in \mathrm{R}^{N}$ is defined by the equation

$$y = T\,x \tag{1}$$

where y is the transform domain representation of x. For the purpose of this paper, we refer to the transform T as orthonormal[2] if $T'T = I$, and orthogonal if $T'T = \mathrm{Diag}(\Lambda)$. For the case when $T'T = \lambda I$, we refer to T as being scaled orthonormal. Although T may be arbitrary, the interesting cases of T used in data compression are orthogonal.

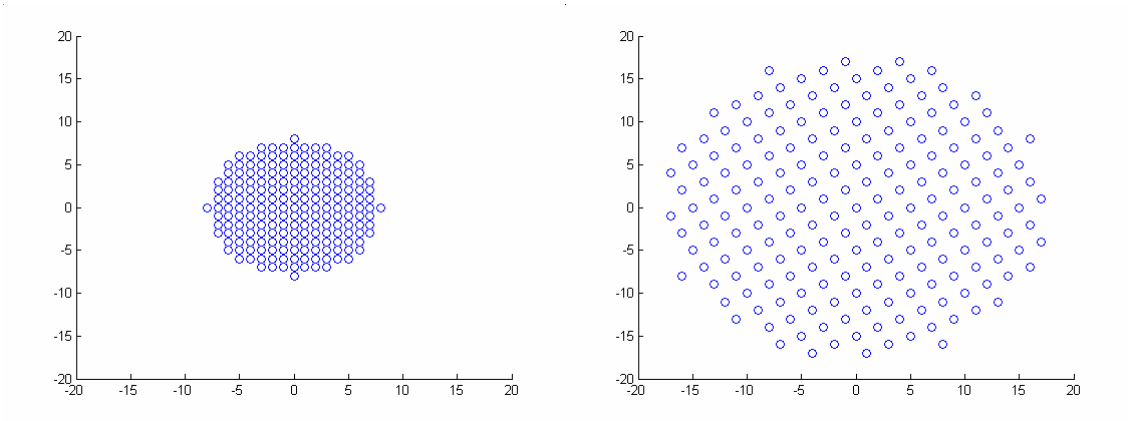
When entries of T are integers, T is an integer transform since it maps integers to integers. The original data x can be recovered from y by the inverse transformation

$$x_i = (T'y)_i \,/\, \Lambda_i \tag{2}$$

The division in (2) is component-wise. It is assumed that T is full-rank, in which case denominators in (2) do not vanish.

[2] The textbook nomenclature of such a matrix is "orthogonal", but for the purpose of this paper we define orthogonal and orthonormal to be in line with orthogonality and orthonormality of basis functions.

Any nontrivial transform with integer entries results in an expansion of the range (more correctly, the "volume") of the transformed data, compared to the volume of the original data. Consider a spherical point cloud of volume *V*, in *N* dimensions, with points located at integer lattice locations. Each point corresponds to a distinct value of the vector *x*. Upon transformation by *T*, this point cloud is reshaped into an *N* dimensional ellipsoid of volume $\mathrm{X}(T)V$, which represents the volume expanded transform domain. The volume of *N* dimensional space occupied by these points is increased by the factor $\mathrm{X}(T) = \mathrm{vol}(T) = |\det(T)| = \prod_{i=1}^{N} \sqrt{\Lambda_i}$. This denotes the *expansion factor* of the transform *T*. It is to be noted that there is no increase in the number of distinct transform domain points despite the increase of the bounding volume. An illustration of volume expansion is shown in Figure 2.



(a) (b)

Figure 2: Illustration of volume expansion accompanying an integer transform (a) two dimensional point cloud in the original domain (b) expanded two dimensional point cloud in transform domain – transform matrix is $\begin{pmatrix} 2 & 1 \\ -1 & 2 \end{pmatrix}$. Number of points in (a) and (b) is the same.

Volume expansion adversely affects the performance of lossless data compression systems. In fact, it is due to volume expansion that linear transforms with integer coefficients of the type shown in (1) are not employed in lossless data compression. To see why volume expansion is undesirable, consider the following argument:

Assume that the alphabet formed by the volume *V* size point cloud can be coded using a uniform, fixed length code. The mean codeword length is $\log_2 V$ bits for large *V*. In the transform domain, the point cloud gets mapped to a larger volume resulting in an increase of $\log_2 \mathrm{X}(T)$ bits in the mean codeword. The additional bits are due to coding a redundant alphabet – note that the number of valid transform domain points is unchanged although the volume is expanded.

It can be readily seen that not all integer lattice points in the transform domain are *valid*, where valid points are lattice points that are generated by *T* applied to integer data. A method of generating invertible representations of linear transforms has been proposed in [6], but there is a penalty to pay in terms of an expansion factor. The number of invalid lattice points as a multiple of the number of valid data points can be large and gives rise to redundancy – except for trivial cases, exploitation of this redundancy has not been attempted so far. It is a central claim of this paper that the transform domain shows periodic patterns that can be exploited to remap the expanded domain into an *N* dimension dense integer lattice with very desirable transform and compression properties. We now look at a trivial case of an integer transform that can be easily compensated for volume expansion.

2-point Hadamard Transform:

The 2-point Hadamard transform matrix is represented by

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{3}$$

An obvious property of the Hadamard transform is that the parity of the two transform coefficients y_1 and y_2 is the same, i.e. these are both even or both odd. Therefore, the last bit of either y_1 or y_2 can be left out with no loss of information [7]. This is equivalent to dividing, say y_2, by 2 and rounding towards $-\infty$. Using the point cloud argument, when y_1 and y_2 are retained to full precision, the point cloud expands by a factor of $\mathrm{X}(T) = 2$. However, when the parity redundancy is compensated by dividing either coefficient by 2 and rounding down, this volume shrinks by a factor of 2, back to *V*. Thus, there is no redundancy in the lattice grid of transform values, and no associated coding loss for the remapped or effective integer transform

$$\begin{pmatrix} y_1 \\ \hat{y}_2 \end{pmatrix} = \left\lfloor \begin{pmatrix} 1 & 1 \\ -1/2 & 1/2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right\rfloor \tag{4}$$

The rounding process introduces nonlinearity in the transform, and this is traditionally tolerated as a minor issue. Prior to the inverse transform, y_2 is regenerated from $\hat{y}_2$ by multiplying $\hat{y}_2$ by 2 and adding the parity of y_1 to the result. Pre-multiplication by *T* followed by dividing the result by 2 reconstructs the original vector $\begin{pmatrix} x_1 & x_2 \end{pmatrix}'$.

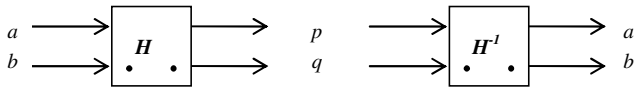


Figure 3: Representation of 2 point Hadamard transform with no volume expansion, and its inverse

The two point Hadamard transform with no volume expansion is represented as shown in Figure 3, according to the input-output equations shown below:

Forward Hadamard transform

$$\begin{aligned} p &= a + b \\ q &= (a - b) >> 1 \end{aligned} \tag{5}$$

Inverse Hadamard transform

$$\begin{aligned} a &= ((p + 1) >> 1) + q \\ b &= (p >> 1) - q \end{aligned} \tag{6}$$

In line with "C" language syntax, the ">>" operation is a signed bit shift to the right. It can be verified that equations (5) and (6) are inverses. In Figure 3, the dots represent the input and output terms *b* and *q*. The linearized transform matrices for $\boldsymbol{H}$ and $\boldsymbol{H^{-1}}$ are $\begin{bmatrix} 1 & 1 \\ -1/2 & 1/2 \end{bmatrix}$ and $\begin{bmatrix} 1/2 & -1 \\ 1/2 & 1 \end{bmatrix}$ respectively. Both these matrices have unit volume.

The same procedure can be implemented by means of the ladder structure shown in Figure 2.

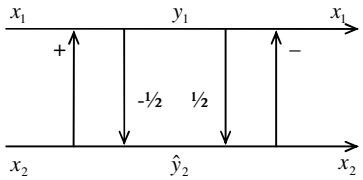


Figure 4: Hadamard transform with no volume expansion implemented through lifting

Next, we introduce the theory of critical quantization central to the ideas proposed in this paper.

III. DEFINITIONS

Definition 1: The mapping $F : x \in \mathrm{Z}^N \rightarrow y \in \mathrm{Z}^N$ is said to be reversible if $F(x_1) = F(x_2) \Rightarrow x_1 = x_2$.

Definition 2: The quantization **Q** of space Y is a partition of Y into some number *K* of discrete cells or quantization bins. *K* may be countably infinite.

Definition 3: A valid transform point is a point in the transform space Y that is mapped by an element of the original domain X. For the purpose of this paper, $\mathrm{X} = \mathrm{Z}^N$, and the transform *T* is assumed to be invertible, i.e. $\det(T) \neq 0$.

Definition 4: Q is a critical quantization of the transform space Y if every quantization bin contains one and only one valid transform point generated by transforming the integer lattice X.

Hence, the critical quantization **Q** maps the integer lattice X into an integer index specifying the quantization bin to which the transformed point belongs. This mapping is one-to-one. It is easy to see that the above definition of criticality implies that any transform *T* can be critically quantized by situating the quantization bins around the *K* permissible values of Y. However, this approach is rather ad-hoc and of no practical value. The only interesting partitions of Y are those that roughly correspond to independent component-wise uniform quantization of the space Y.

When **Q** shows this structure, the partition bins can themselves be remapped into an integer lattice of the same dimensionality as Y. This is equivalent to a component-wise quantization of the transform coefficient *y* to a reduced or "shrunk" integer space. In some implementations of **Q** proposed in this paper, the quantization bins may be perturbed by a small distance for a subset of transform domain points. Whether due to this perturbation or due to rounding, quantization introduces a small nonlinearity in *T*. However, for the purpose of analysis, we shall assume the overall quantized transform to be linear even when there are subtle non-linearities. This is not any different from lifting based techniques which are subtly non-linear implementations of linear transforms. When the quantization bin widths are integers in each dimension, then the entries of the resulting linearized transform are rational numbers.

Although criticality qualifies the quantization process, it is closely linked to the transform itself and may be considered as part of the transform definition. Therefore, it becomes meaningful to refer to transforms that are critically quantized – in which case an alternative definition of critical quantization is applicable.

Definition 5: *T* is a critically quantized transform if a one-to-one mapping exists between the integer lattice representing the domain of *T* and the integer lattice representing the range of *T*, i.e. $\mathrm{X} \leftrightarrow \mathrm{Y}$. Therefore, a critically quantized transform *T* defines a reversible mapping.

The two properties of critically quantized *T* are reversibility and volume preservation. Reversibility of *T* is evident, since the domain and range are mapped one-to-one, each transform domain point in Y is generated by one and only one point in X. Given the transform coefficient *y*, the point *x* generating this transform coefficient is known without ambiguity.

Volume preservation is the property by which there are no "holes" in the transform domain, holes being transform domain integer lattice points that are not mapped by any point in X. Volume preservation is an essential property for transforms used in data compression; in the event of volume expansion, redundant information will need to be encoded causing loss of compression efficiency.

Theorem 1: $\mathrm{vol}(T) = |\det(T)| = 1$, when *T* is a critically quantized linear transform.

Proof: A sphere of unit volume is expanded to a sphere of volume $\mathrm{vol}(T) = |\det(T)|$ by the application of the linear operator *T*. A point cloud of volume *K* in the original domain gets mapped to a point cloud of volume $K\,\mathrm{vol}(T)$ in the transform domain. When $\mathrm{vol}(T) < 1$, one-to-one mapping is violated since the number of transform domain points is strictly less than the number of original lattice points within *K*. Likewise, when $\mathrm{vol}(T) > 1$, holes are introduced in the transform domain thereby violating the one-to-one mapping.

The converse of Theorem 1 does not hold since it is easy to come up with linear transforms *T* with unit determinant, but whose quantization rule ambiguates the transform domain.

Lemma 1: Cascades of critically quantized transforms are critically quantized.

Proof: Let $T = T_2 \cdot T_1$, and the spaces X, Y and Z represent the original domain, range of T_1, and range of T_2 respectively. From Definition 5, we know that X ↔ Y and Y ↔ Z, therefore X ↔ Z, and *T* is critically quantized.

As expected, $\mathrm{vol}(T) = |\det(T)| = |\det(T_1)\det(T_2)| = \mathrm{vol}(T_1) \cdot \mathrm{vol}(T_2) = 1$.

In the remainder of this paper, we imply quantization to mean *N* dimensional vector quantization. Further, it is desirable for the quantization to be uniform and separable. In other words, the quantization bins are ideally uniform scalar partitions of the vector space into *N* dimensional rectangular volumes. Under critical quantization, each of these regions contains only one valid transform point.

When quantization is separable and uniform scalar, the effect of quantization along a certain dimension is basically a rounded division by the bin width along that particular dimension. For the purpose of analysis of the transform, this division can be pulled into the transform itself. For instance, the critically quantized 2-point Hadamard transform exploiting the redundancy of the parity bit has the equivalent linearized transform matrix shown in (4), which is non-expansive with unit determinant. The extension of critical quantization to general integer transforms has not been considered in prior research. This paper studies the patterns of redundancies in general orthogonal integer transformations in a rigorous manner, showing how arbitrary transforms can also be quantized critically and thereby used for data compression. The parity bit redundancy of the 2-point Hadamard transform falls out as a special case of the "modulo transform" introduced in the next sections.

It can be seen that the two transform coefficients in (4) are unequally scaled. Cascades of 2-D Hadamard transforms, with each stage exploiting the parity bit redundancy, give rise to more complicated critically quantized transforms due to the unequal scaling of coefficients.

Definition 6: The rounded division operation sdiv is defined for integer *a* and *n* as

$$\mathrm{sdiv}(a,n) = \left\lfloor \frac{a + \lfloor n/2 \rfloor}{n} \right\rfloor \tag{7}$$

Definition 7: The signed modulo operation smod is defined as

$$\mathrm{smod}(a,n) = a - n\,\mathrm{sdiv}(a,n) \quad \in \quad \left\{ -\lfloor n/2 \rfloor \ldots \lfloor (n-1)/2 \rfloor \right\} \tag{8}$$

When quantization bins are centered around multiples of n, the quantized index of a is $\left\lfloor \frac{a + \lfloor n/2 \rfloor}{n} \right\rfloor$, which is given by the sdiv operation. The location of a within its quantization bin with respect to its center is given by the signed modulus smod. The range of smod(a, n) is centered around zero for odd n. For even n, a consistent rounding rule is chosen for all a, for instance rounding is always towards $-\infty$.

IV. Plane Rotations – Reversible Butterflies

The simplest non-trivial linear orthogonal transform is a two-point transform. A two point transform operates on a vector of length two. This vector may be considered as a point on a two dimensional plane. An orthonormal transform rotates this vector on this plane about the origin while preserving its length, whereas an orthogonal transform scales the rotated vector as well. A two point transform is represented as a "butterfly" in the signal flow graph as shown in Figure 5. Let us now consider the following two point transform or rotation matrix:

$$R = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \tag{9}$$

For R to be orthonormal, $c^2 + s^2 = 1$. The theory of lifting provides reversible implementations based on decomposing R into a product of three normal triangular matrices, i.e. three shears. This has been used for the fast rotation of images [8]. Let us relax the restriction of orthonormality of R, and instead restrict s and c to be integers. The resulting rotation is not unit norm (except for the trivial cases of $s = \pm 1$ or $c = \pm 1$). c and s may be derived from rational approximations to the desired rotation angle cosine and sine terms, such that they have no common factors. The resulting R is scaled orthonormal. Clearly, R is an expansive transformation with an expansion factor of $D = c^2 + s^2 \in \mathrm{Z}^+$. In the remainder of this section, we demonstrate how R can be reformulated as a non-expansive transformation.

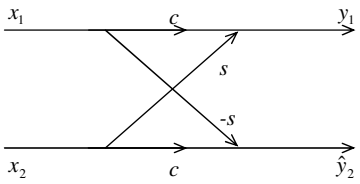


Figure 5: Butterfly corresponding to Equation (10)

Lemma 2: R is rank-deficient in modular $D = c^2 + s^2$ arithmetic.

Proof: $\det(R) = c^2 + s^2 = D$, which is zero in modular D arithmetic.

Theorem 2: For the transformation $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, the transform domain coefficients y_1 and y_2 are redundant in modular D arithmetic.

Proof: Consider the quantity $c\, y_1 - s\, y_2$ evaluated as follows

$$c\, y_1 - s\, y_2 = \left(c^2 + s^2\right) x_1 - 0\, x_2 = 0 \bmod D \tag{10}$$

We refer to equation (11) as the *modulo relationship*. Since c and s have no common factors, c, s and D have no pairwise common factors. There exist inverses c^{-1} and s^{-1} in modular D arithmetic such that $c^{-1}c = s^{-1}s = 1 \bmod D$. The modulo relationship can be rewritten as

$$y_1 = c^{-1} s\, y_2 \bmod D \tag{11}$$

which implies that in modular D arithmetic one transform component can be recovered from the other, proving their redundancy.

The modulo relationship also implies it's dual:

$$c\, y_2 + s\, y_1 = 0 \bmod D \tag{12}$$

since $c\, y_2 + s\, y_1 = c^{-1}\left(c^2 y_2 + c\, s\, y_1\right) = c^{-1}\left(-s^2 y_2 + c\, s\, y_1\right) = c^{-1} s\left(c\, y_1 - s\, y_2\right) = 0 \bmod D$.

For a base D, any integer k can be represented in two parts corresponding to its quotient and remainder with divisor D. When a consistent rounding rule is applied for both positive and negative k, the remainder corresponds to k mod D. Let the quotient be written as k div D. Thus,

$$k = (k \operatorname{div} D) D + (k \bmod D) \tag{13}$$

The functions sdiv and smod may be used in place of div and mod in this equation.

Corollary 1: When transform coefficients y_1 and y_2 are represented as quotient and remainder with divisor D, $(y_1 \operatorname{div} D)$ and y_2 are sufficient to reconstruct y_1, and therefore the original data point $(x_1 \quad x_2)'$.

Proof: Since we know that $y_1 = c^{-1} s\, y_2 \bmod D$, the quantity $(y_1 \bmod D)$ can be computed from the given information using the modulo relationship in equation (14). y_1 can be reconstructed using equation (14), which together with given y_2 is the exact transform domain value. The original data point is obtained by inverting the transform. The roles of y_1 and y_2 can be interchanged without loss of generality.

The above sufficient representation realizes the effective linearized transform $\hat{T} = \begin{pmatrix} c/D & s/D \\ -s & c \end{pmatrix}$, which has unit volume and is critically quantized. This proves that any scaled orthonormal two point transform T can be reformulated as a non-expansive or critically quantized $\hat{T}$ by exploiting the modulo relationship, although the resulting $\hat{T}$ may be only orthogonal.

Theorem 3: Any transform domain point satisfying the modulo relationship (11) or (13) is a valid transform domain point generated by a pair of integers in the spatial domain.

Proof: Since T is full rank, there exists a point $(x_1 \quad x_2)'$ in the original space that generates $(y_1 \quad y_2)'$ in the transform domain. It needs to be shown that $(x_1 \quad x_2)'$ is an integer lattice point to prove the above theorem. Substituting $(x_1 \quad x_2)'$ into (11), we get the condition $D\,x_1 = k\,D$ for some integer k. Since $D \neq 0$, $x_1 = k \in \mathrm{Z}$.

Further, x_2 can be calculated from the two scalar equations derived from the transform components y_1 and y_2 as follows

$$x_2 = \frac{y_1 - c\,x_1}{s} = \frac{y_2 + s\,x_1}{c} \tag{14}$$

Since c and s have no common factors, and the numerators and denominators of (15) are integers, the equation (15) only permits integer solutions, i.e. $x_2 \in \mathrm{Z}$, proving the claim.

Definition 8: A *sliver* is defined as a 1×D or a D×1 area in the transform domain, containing D consecutive integer lattice points. In order to prevent any ambiguity, the periphery of a sliver does not pass through any integer lattice points.

Theorem 4: A sliver in the transform domain contains exactly one valid transform domain point.

Proof: Consider the quantity $c\,y_1 - s\,y_2$ defined on the domain $(y_1 \quad y_2)$. Since c, s and D do not have any pairwise common factors, the quantity evaluates to the values $\{0,1 \ldots D-1\}$, not necessarily in that order, in modular D arithmetic for the D points on the sliver. Only one point on the sliver evaluates to 0 mod D, and therefore there is one and only one valid transform domain point on the sliver, regardless of orientation or location of the sliver.

The entire transform domain can therefore be partitioned into slivers. Each sliver is known to contain one and only one transform domain point. Therefore, the address or location of each sliver contains all the information necessary and sufficient to reconstruct the valid transform domain point, thereby reconstructing the corresponding original spatial domain point. When all slivers are aligned and oriented in the same direction, the address of each sliver can be represented, for instance, by $(y_1, \mathrm{sdiv}(y_2, D))$. In this case, slivers form the quantization partitions and lead to a critical quantization of the transform domain space. Each quantization partition contains exactly one valid transform point, and the effective transform matrix for the above representation is $\hat{T} = \begin{pmatrix} c & s \\ -s/D & c/D \end{pmatrix}$, which has unit volume. This is true for any arbitrary integer 2D rotation matrix, provided c and s do not have any common factors.

Corollary 2: Any 2D rotation transform can be approximated by a critically quantized transform. Critically quantized transforms exploiting the modulo relationship are referred to here as *modulo transforms*.

Proof: A 2D rotation matrix of the type $\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$ can be approximated arbitrarily closely by an integer rotation matrix $\begin{pmatrix} c & s \\ -s & c \end{pmatrix}$, such that c and s have no common factors. The scaling introduced by integerizing the rotation is compensated for by critical quantization, although the compensation factor is unequal between the transform components.

Example: Let us take another look at the 2-point Hadamard transform. In this case, $c = s = 1$, and $D = 2$. A valid transform domain point $(y_1 \quad y_2)'$ satisfies the modulo relationship (13), which in this case is $y_1 - y_2 = 0 \bmod 2$. This is no surprise – the modulo relationship merely states that y_1 and y_2 have the same parity. The transform domain space can be partitioned into 1×2

slivers, in other words the critically quantized pair $\left(y_1, \left\lfloor \frac{y_2}{2} \right\rfloor \right)$ contains all the information, and no more, to reconstruct the original data. Hence, the dropping of the last bit of one transform coefficient of the 2-point Hadamard transform while maintaining reversibility of the operation is just a special case of critical quantization using 1×*D* or *D*×1 slivers.

Quantization partitioning based on such slivers has a major drawback. The effective norms of the two basis functions are skewed by a factor of *D*, which is at a minimum 2 for the 2-point Hadamard transform, and increases rapidly when large numbers are used for the matrix entries *c* and *s*. When norms are dissimilar, multiple stage transformations become more complicated to design since the scaling of different data paths is vastly different. Arithmetic precision becomes an issue since the dynamic range of intermediate quantities is highly skewed. Multiple code tables may have to be used to best match the dissimilar statistics, and this can adversely impact codec complexity.

The undesirable effects of using an elongated *D*×1 quantization bin can be minimized if the bins have a smaller aspect ratio. The ideal situation is when partition bins are $\sqrt{D} \times \sqrt{D}$ in size, in which case the resulting matrix $\hat{T}$ is orthonormal. Reshaping the quantization bins from slivers to square or near-square bins is a non-trivial process, which we examine in the next section. We look at some properties of valid transform domain points, and design a construction procedure that can be used to critically quantize the transform domain into more even shaped rectangular regions. Subsequently, we look at perfectly square quantization partitions that can be guaranteed to be critical when specific Pythagorean triples (of *c*, *s* and $\sqrt{D}$) are used.

V. Construction of Arbitrary Quantizers

Definition 9: The *constellation* of a transform *T* is the pattern of lattice points formed by the valid transform domain coefficients *T* in the transform domain space.

Theorem 5: The constellation of *T* for a two dimensional orthogonal integer transform is periodic in either dimension with a period $D = \mathrm{vol}(T)$.

Proof: Assume $(y_1 \quad y_2)'$ is the point on the constellation generated by $(x_1 \quad x_2)'$. The points $(y_1 \pm D \quad y_2)'$ and $(y_1 \quad y_2 \pm D)'$ also exist in the constellation, being generated by $(x_1 \pm c \quad x_2 \pm s)'$ and $(x_1 \mp s \quad x_2 \pm c)'$ respectively. The period can be no less than *D* since this hypothesis will in direct violation to Theorem 4.

Any further analysis of the transform domain can therefore be restricted to a square of size *D*×*D*. As long as constellation points within this square can be remapped to their quantized representation in a reversible manner, the entire transform retains reversibility. In other words, if the quantization bins are also periodic in *D* in both dimensions, and each quantization bin within a *D*×*D* period contains one constellation point, the quantization is critical and the resulting transform can be formulated to be reversible and non-expansive.

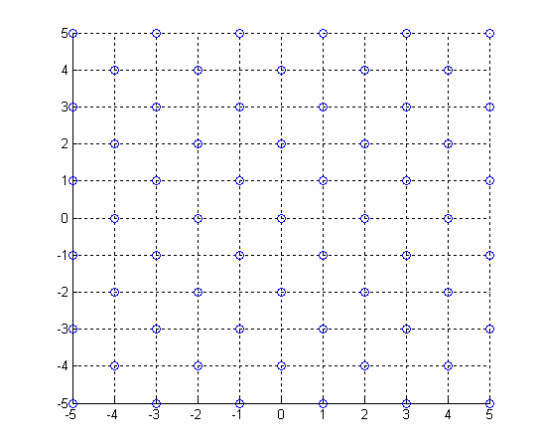
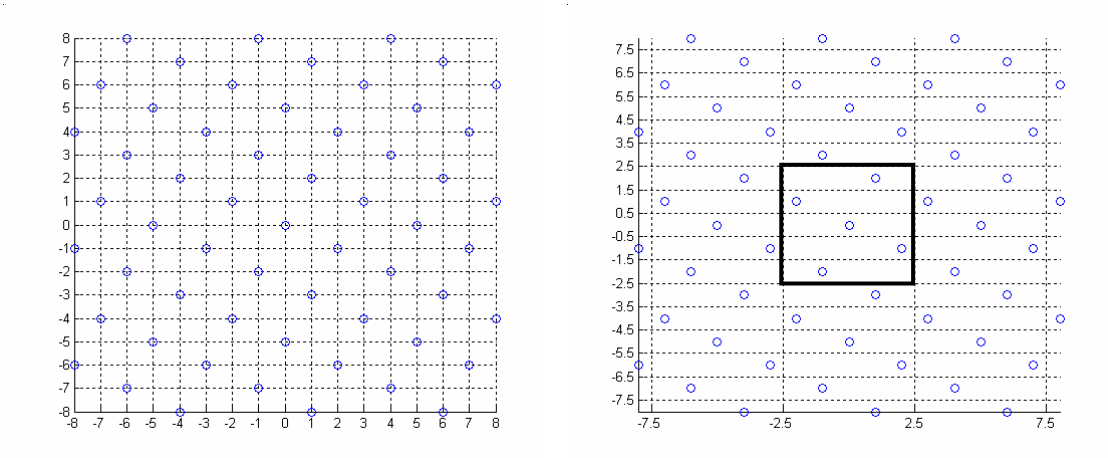


Figure 6: Constellation of 2-point Hadamard transform, showing periodicity of 2 in either direction

Figure 6 and Figure 7 present a graphical representation of Theorem 5. The rectangular grid of integers in two dimensions is remapped to the constellation in the transform domain. Circles denote valid transform coefficients, and intersecting grid lines correspond to integer coordinates. Figure 6 shows the 2 point Hadamard transform constellation with $D = 2$, and Figure 7(a) shows the constellation of the transform $\begin{pmatrix} 2 & 1 \\ -1 & 2 \end{pmatrix}$ with $D = 5$. The latter transform is used in the H.264 video format [9]. Periodicity of the constellations in both directions, with periods of 2 and 5 respectively, is evident. Figure 7(b) shows a 5×5 periodic tile partitioned into five slivers, each sliver containing one constellation point, demonstrating Theorem 4.



(a) (b)

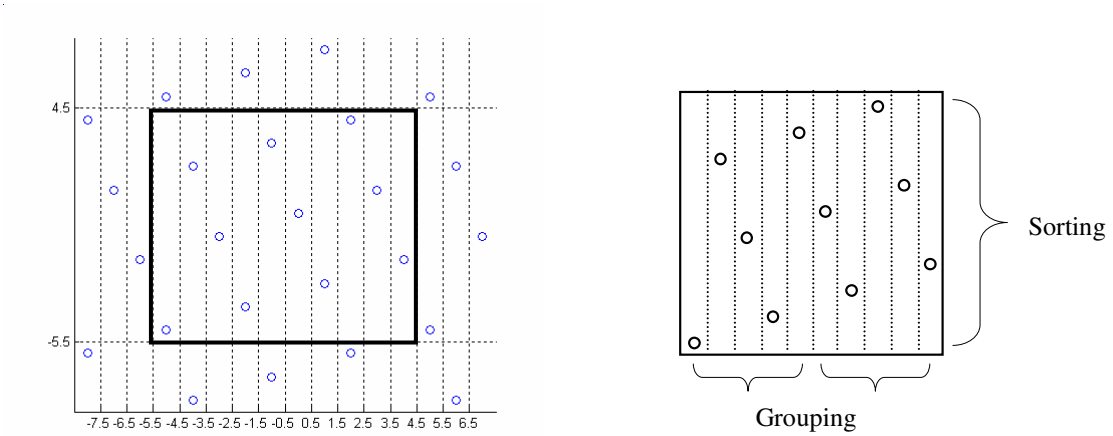
Figure 7: (a) $R\{2,1\}$ constellation shown on integer sampling grid and (b) $R\{2,1\}$ constellation showing 5x1 slivers marked by dashed lines. The inscribed square shows the periodic $D \times D$ tile. In this case, $D = 5$.

Construction: A simple remapping technique can be used to achieve more equitable quantization regions. Assume that D is composed of two integral factors D_1 and D_2, i.e. $D = D_1 D_2$. First, the transform domain is partitioned into $D \times D$ tiles. Since the transform domain is periodic in D in either dimension, the following procedure is applied to each $D \times D$ tile.

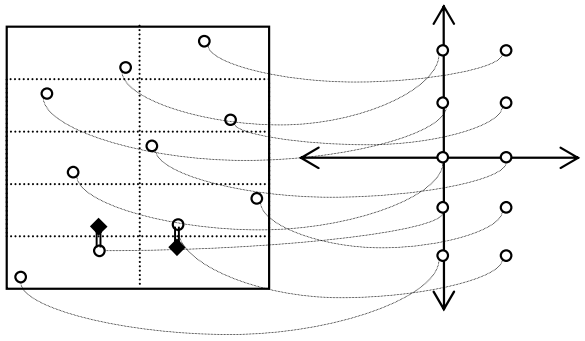
Each tile is partitioned along one direction into D slivers, each sliver containing one constellation point. D_1 of these slivers are grouped together, so as to partition the $D \times D$ tile into D_2 rectangular regions, each region containing D_1 constellation points. The D_1 constellation points within each rectangle are sorted along the long dimension and an address k denoting their sort order assigned to them. In the other dimension, their address l is a number between 0 and $D_2 - 1$, identifying their $D_1 \times D_2$ partition.

The sorting of D_1 constellation points is unique, since no two points within the $D \times D$ tile may share either coordinate. Therefore, there is a one to one mapping between constellation points and a composite index (k, l) taking on integer values in the $\{0,1\ldots D_1 - 1\} \times \{0,1\ldots D_2 - 1\}$, within the $D \times D$ tile. The second coordinate l is computed by quantizing the coordinate within the tile into D_2 equal sized intervals.

The sorting order roughly corresponds to a scalar quantization, give or take a few points that fall into the "wrong" bin. Figure 8 illustrates the construction procedure using the transform matrix $\begin{pmatrix} 3 & -1 \\ 1 & 3 \end{pmatrix}$. This is the simplest transform that gives rise to a composite number for D, which in this case is 10. The constellation and periodic tile centered at the origin is shown in Figure 8 (a), and (b) shows the grouping of vertically oriented slivers within each tile into two groups of five slivers each. The five constellation points in each group are sorted to produce a quantization index between 0 and 4. Figure 8(c) shows the final quantization partitions with the effective displacement of two constellation points that gives rise to uniform partitioning depicted by arrows. Finally, the right hand side of Figure 8(c) shows how the tile can be remapped to a 5×2 densely packed grid. The effective transform matrix is $\begin{pmatrix} \frac{3}{5} & -\frac{1}{5} \\ \frac{1}{2} & \frac{3}{2} \end{pmatrix}$, which has unit volume and is critically quantized.



(a) (b)



(c)

Figure 8: (a) $R\{3,-1\}$ constellation showing unique transform domain points in 10×1 slivers marked by dashed lines. Inscribed square is a 10×10 periodic tile, (b) quantization of $R\{3,-1\}$ using 5×2 regions following the construction procedure of Section V, (c) result of construction – 10×10 periodic tile shown on left, corresponding quantized grid points shown on right. The diamonds in the tile mark constellation points that are virtually moved as a result of sorting.

Further, when D is a square, the construction procedure can be used to group $\sqrt{D}$ slivers and sort them into $\sqrt{D}$ bins. The effective transform in this case is

$$\hat{T} = \frac{1}{\sqrt{D}} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \tag{15}$$

which is orthonormal. Thus, this construction can be used to generate effective orthonormal transforms for all primitive Pythagorean triples [10] which are integer triples (c, s, $\sqrt{D}$) satisfying $D = c^2 + s^2$ with no common factors. It can be shown that all Pythagorean triples can be generated by the formula

$$\begin{aligned} c &= 2\,a\,b \\ s &= a^2 - b^2 \\ \sqrt{D} &= a^2 + b^2 \end{aligned} \tag{16}$$

c and s may be interchanged to overcome the inherent constraints of even c and odd s in (17). Without loss of generality, only rotations between 0 and $\pi/4$ are of interest, since other rotations can be realized by negating the rotation angle or adding an arbitrary number of right angle rotations which are trivial transforms of the form $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^n$, or a combination of both. Any given rotation angle θ can be approximated to an arbitrary degree of accuracy by the substitution

$$\frac{b}{a} \cong \tan\left(\frac{\theta}{2}\right) \tag{17}$$

to generate candidate integer values for a and b, which are then used to compute the Pythagorean triple.

In practice, this construction procedure can be implemented in a computer using small lookup tables and straightforward division based quantization, as follows

$$\begin{aligned} y &= \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = T\,x \\ y_D &= y_1 \bmod D \\ \hat{y} &= \left\lfloor \frac{y + \mathrm{L}(y_D)}{\sqrt{D}} \right\rfloor \end{aligned} \tag{18}$$

where the modulo operation is component-wise, and L is a lookup table of size D taking on small values typically in the range $\left\{-\sqrt{D} \cdots \sqrt{D}\right\}$. The inverse transform is similar to the forward transform. Since the construction procedure is identical for each tile, the computational complexity in implementing such a transform is not excessive. Typically, only a small number of constellation points need to be "nudged" into an adjacent partition – in other words, several entries of L are zero, and the lookup table has a sparse structure.

This brings us to the next question – are there any situations where constellation points naturally fall into their desired $\sqrt{D} \times \sqrt{D}$ quantization bins – in other words are there any situations where L in (19) is all-zero ? We present a powerful result in Theorem 6 that claims that Pythagorean triplets satisfying a certain condition give rise to a constellation that is uniformly *and critically* quantized using $\sqrt{D} \times \sqrt{D}$ bins.

Theorem 6: Any Pythagorean triple satisfying $c = \pm\left(\sqrt{D} - 1\right)$ or $s = \pm\left(\sqrt{D} - 1\right)$ is critically quantized using uniform quantizers with square bins of size $\sqrt{D} \times \sqrt{D}$, when the quantization grid is arranged such that the origin lies at the center of its bin.

The proof of Theorem 6 is rather complicated, and is presented in Appendix A.

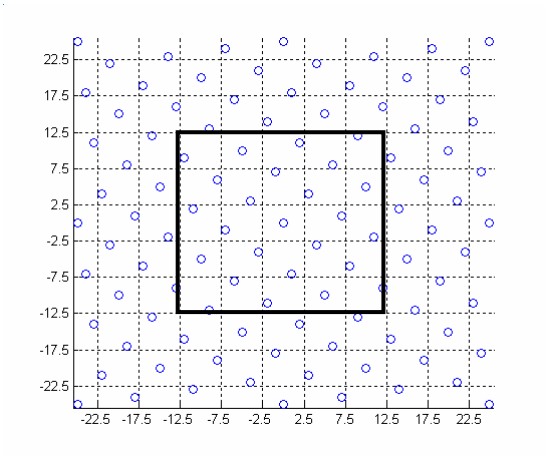
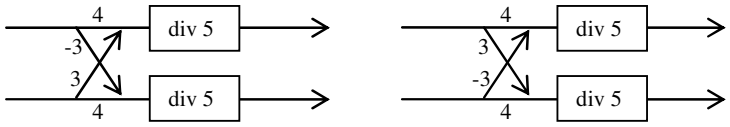


Figure 9: Demonstration of Theorem 6 for the simplest Pythagorean triad $R(4, 3)$ – one and only one constellation point lies in each 5×5 quantization bin, and the resulting transform is both critically quantized and orthonormal.

Theorem 6 is demonstrated for the transform matrix $T = \begin{pmatrix} 4 & 3 \\ -3 & 4 \end{pmatrix}$, which is the smallest non-trivial Pythagorean triple that satisfies the condition. Figure 9 shows the constellation generated by this transform, with a periodic tile centered at the origin. Five uniform quantization partitions are made in the horizontal and vertical directions of the tile, and each bin contains one and only one constellation point. Thus, the transform is uniformly and critically quantized.



Forward rotation R (4, 3) Inverse rotation R (4, -3)

Figure 10: Forward and inverse Pythagorean triple rotations $R(4, 3)$ and $R(4, -3)$

The inverse transform of T is merely $\begin{pmatrix} 4 & -3 \\ 3 & 4 \end{pmatrix}$ which is also critically quantized using a separable uniform quantizer. It can be shown that the original data can be recovered with no error by the critically quantized inverse transform. The flowgraphs of the forward and inverse transform are shown in Figure 10, where the box labeled "div5" performs the operation sdiv(k, 5) on the input k.

Theorem 6 is satisfied when the primitive Pythagorean triple is generated by choosing s and c such that

$$c = \frac{s^2 - 1}{2}, \tag{19}$$

which produces rotation angles

$$\theta = \arctan\left(\frac{2s}{s^2 - 1}\right) \tag{20}$$

As the value of s increases, these angles get progressively smaller. s is odd and c is even, and their roles may be reversed. There are countably infinite permissible rotation angles corresponding to the odd values of s. The first ten triples and angles that can be

realized are shown in Table 1. It can be inferred that there exist good approximations to small rotation angles. Larger rotations may be realized as a finite-length cascade of smaller rotations of the form (21), although there is computational penalty in doing so.

Index	s	c	$c+1$	θ
1	3	4	5	0.643501
2	5	12	13	0.394791
3	7	24	25	0.283794
4	9	40	41	0.221314
5	11	60	61	0.18132
6	13	84	85	0.153544
7	15	112	113	0.133136
8	17	144	145	0.117512
9	19	180	181	0.105166
10	21	220	221	0.095166

Table 1: Pythagorean triples satisfying the conditions of Theorem 6

An example of a two-stage modulo transform is considered here. The 2-point Hadamard transform is a particularly difficult rotation to realize – in some sense, it is the most extreme angle since numerically larger angles can be realized by a combination of right-angle flips and smaller rotations. The best two-stage approximation of the desired rotation angle $\pi/4$ is obtained by two successive rotations of $\theta = \arctan\left(\frac{2\times 5}{5^2-1}\right) = 0.394791$. Each rotation is based on the primitive Pythagorean triple (5, 12, 13). The effective rotation is by the matrix

$$\begin{pmatrix} 12 & 5 \\ -5 & 12 \end{pmatrix}^2 = \begin{pmatrix} 119 & 120 \\ -120 & 119 \end{pmatrix}. \tag{21}$$

Clearly, the effective transform matrix is close to the 2 point Hadamard transform although it is significantly more computationally demanding than the simple parity bit reduction technique. Figure 11 shows the forward transform realizing the cascade of two modulo transform steps resulting in an orthonormal and reversible transform.

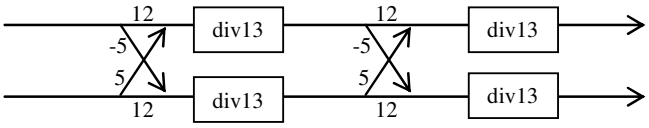


Figure 11: Modulo transform approximation to the Hadamard transform

Let us now contrast this structure with lifting for producing a normalized output. The three shear decomposition for implementing a 2 point Hadamard transform is given by

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1-\sqrt{2} & 1 \end{pmatrix}\begin{pmatrix} 1 & 1/\sqrt{2} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1-\sqrt{2} & 1 \end{pmatrix} \tag{22}$$

One possible computationally friendly lifting approximation to (23) is considered below

$$\begin{pmatrix} 1 & 0 \\ 13/32 & 1 \end{pmatrix}\begin{pmatrix} 1 & 45/64 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 13/32 & 1 \end{pmatrix} = \begin{pmatrix} 0.7144 & 0.7031 \\ -0.6965 & 0.7144 \end{pmatrix} \tag{23}$$

The DC leakage of the 2 point Hadamard transform is ideally zero. In other words, the AC transform component of a DC signal (for instance the input vector [1 1]') should be zero. For the two stage modulo transform realization, the DC leakage for the input [1 1]' is 0.0059, whereas the DC leakage for the lifting structure shown in (24) is 0.0179. By this metric, the modulo transform structure of (22) is better than the lifting structure of (24) in approximating the normalized 2 point Hadamard transform. In practice however, even this degree of closeness to the Hadamard transform may be insufficient for the first stage of a forward transform due to nonzero DC leakage. For subsequent steps in a cascade, these approximations are more likely to work well.

In the next section, we look at extending these results to higher dimensions. Specifically, we examine realizations of the 4 and 8 point DCTs using modulo transform steps.

VI. Transforms in Higher Dimensions

It is known that an $N\times N$ orthonormal transform can be implemented with up to $N(N-1)/2$ plane rotations (and a possible mirror flip) [11]. Each plane rotation affects two components of the data vector. T can be represented as $T = \Pi_{i,j} R_{i,j}$, where $R_{i,j}$ is a matrix, shown below, that performs a 2D rotation on the i^{th} and j^{th} components of the data

$$R_{i,j} = \begin{pmatrix} 1 & 0 & 0 & & 0 & 0 \\ 0 & c_{ij} & 0 & & s_{ij} & 0 \\ 0 & 0 & 1 & & 0 & 0 \\ & & & \ddots & & \\ 0 & -s_{ij} & 0 & & c_{ij} & 0 \\ 0 & 0 & 0 & & 0 & 1 \end{pmatrix} \tag{24}$$

The non-trivial entries are in the i^{th} and j^{th} columns and rows of the matrix. The angles defined by these entries are called Euler angles. An orthogonal matrix T with integer entries may be factorized similarly, but the individual rotation matrices may not be constrained to integer entries. In this case, the nontrivial entries are integerized by scaling them up, which will be assumed here.

Theorem 7: Transformation $T = \Pi_{i,j} R_{i,j}$ is quantized critically if all $R_{i,j}$ s are quantized critically.

Proof: This is a direct application of Lemma 1.

In theory, therefore, it can be concluded that any higher order orthonormal integer transform can be approximated by a cascade of modulo transforms, thereby ensuring reversibility. In practice, some degree of simplification is possible by allowing non-orthonormal intermediate transforms to produce an overall transformation T which may in turn be only orthogonal. Subsequent steps such as transformation along the other dimension (when 2 and higher dimensional data is being processed) or entropy coding will take into account the denormalized nature of transform coefficients. In the following section, we consider modulo transform implementations of two commonly used transforms, the 4 point and the 8 point DCT. The implementations use a mix of normalized and denormalized rotations.

4-point DCT: The even basis functions of the 4 point DCT are trivially generated by a cascade of 2-point Hadamard transform steps. The odd basis requires a rotation by the angle π/8, which is approximately realized by the Pythagorean triad (5, 12, 13). Based on these observations, the implementation shown in Figure 12 can be derived.

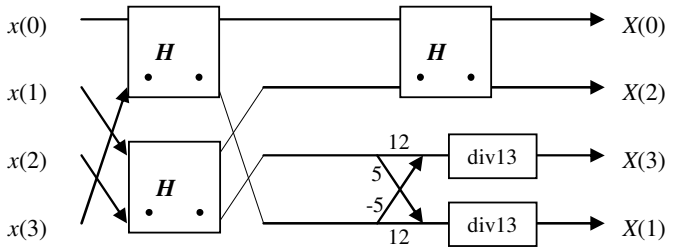


Figure 12: Modulo transform 4-point DCT

The resulting transform T is shown below:

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 12/26 & 5/26 & -5/26 & -12/26 \\ 1/2 & -1/2 & -1/2 & 1/2 \\ -5/26 & 12/26 & -12/26 & 5/26 \end{bmatrix} \tag{25}$$

The resulting basis functions have unequal norms, viz. $\begin{pmatrix} 2 & 2^{-1/2} & 1 & 2^{-1/2} \end{pmatrix}$. Other implementations can be derived by repositioning the divisions by 2, and/or by implementing the Hadamard steps as a cascade of modulo transform butterflies, or using lifting in some steps.

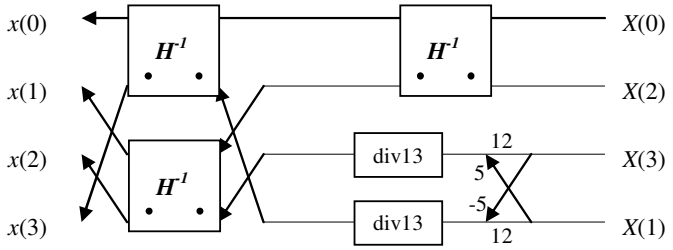


Figure 13: Inverse of modulo transform 4-point DCT

The inverse of the above transform is easy to derive. The inverse of the modulo transform based Hadamard transform has been discussed earlier. The inverse of the rotation realized by the Pythagorean triad (5, 12, 13) is the rotation by the same triad in the opposite direction. The forward and inverse rotations differ in the signs of the off-diagonal terms.

Table 2 presents a comparison of the coding gains of the true 4 point DCT and the approximation shown in Figure 12. It can be seen that there is no performance loss, although lossless reversibility of the transform is maintained.

Coding gain	DCT	MT-DCT
ρ = 0.95	7.5701 dB	7.5702 dB
ρ = 0.90	5.3870 dB	5.3872 dB
ρ = 0.85	4.1453 dB	4.1456 dB

Table 2: Coding gains of 4-point DCT and its modulo transform approximation shown in Figure 12

8-point DCT: Based on the degree of accuracy desired, several approximations to the 8 point DCT can be realized using modulo transforms. We present one such structure in this section.

The even basis functions of the 8 point DCT, starting with the DC basis, are based on the 4 point DCT. Figure 14 shows a relatively simple implementation of the 8 point DCT. This uses a modification of Loeffler's factorization [12] to produce the odd basis functions. Even basis functions are generated by the 4-point DCT implemented as shown in Figure 12. At a minimum, two non-trivial rotations are needed to produce the odd basis functions. In Chen's factorization [13], the first rotation by $\pi/4$ must be normalized – this requires at least two modulo transform stages. Subsequently two more non-trivial rotations are needed, increasing both the number of modulo transform stages and the number of cascading steps. Loeffler's factorization can be modified so that the intermediate Hadamard stages implement non-$\pi/4$ rotations. Moreover, the first rotation stage by $\pi/16$ can be ignored, giving the structure shown in Figure 14. Outputs are denormalized, as for the 4 point DCT. All three non-trivial odd basis rotations use the simplest Pythagorean triad, thereby minimizing the divisor, which is desirable for implementing the division as a multiplication and bit shift.

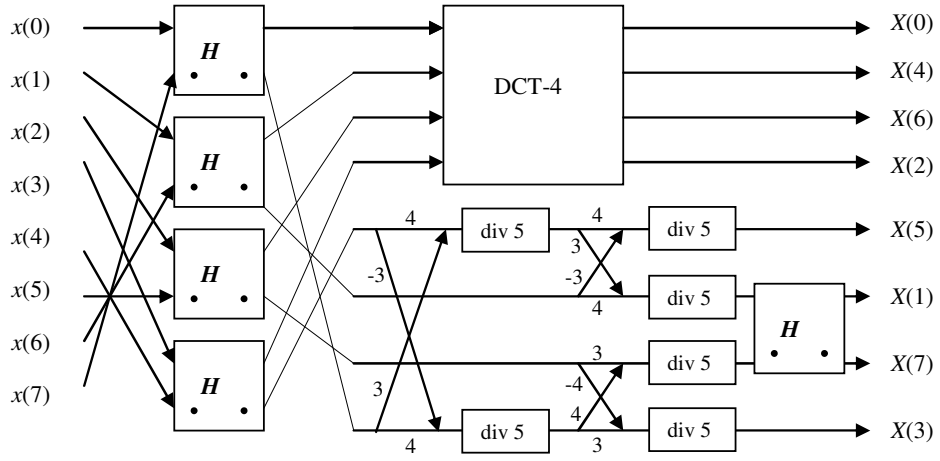


Figure 14: A modulo transform approximation to the 8-point DCT

The coding gains of the true 8-point DCT and its approximation of Figure 14 are shown in Table 3. There is a moderate drop in performance in this particular modulo transform approximation. Better structures may be derived to more closely match the coding gain of the 8-point DCT, although these approximations will involve more computations and/or larger Pythagorean triads in the generation of the odd transform coefficients.

Coding gain	DCT	MT-DCT
$\rho = 0.95$	8.8259 dB	8.6305 dB
$\rho = 0.90$	6.2761 dB	6.1029 dB
$\rho = 0.85$	4.8267dB	4.6742 dB

Table 3: Coding gains of 8-point DCT and its critically quantized approximation of Figure 8

Multi-dimensional DCTs: The approximations of the 4 and 8 point DCT may be applied in succession on two dimensional data to generate approximate and reversible 4×4, 4×8, 8×4 and 8×8 DCTs. Although the norm of each transform coefficient is different after the transform along the first dimension, the transform along the second dimension is applied to terms with equivalent scaling. In other words, in the second dimension, the transform is performed across the same-indexed coefficients, meaning that the norms are balanced at the input to the second dimension. Thus, multidimensional DCTs and other transforms can be realized as a cascade of critically quantized one-dimensional transforms, as long as the original transform is separable.

However, it must be kept in mind that the imbalance in norms will multiply with the number of separable stages. For instance, a 4×4 DCT implemented as two stages of the approximation shown in Figure 12 will produce transform coefficients scaled by one of $\left\{2^{-1}, 2^{-1/2}, 1, 2^{1/2}, 2, 4\right\}$. The ratio of the largest to the smallest scale is 8 for the two dimensional transform, whereas it is $\sqrt{8}$ for the one dimensional stage. The inverse of a multi-dimensional modulo transforms is obtained by applying the inverse of each stage of the forward transform, in the reverse order of the forward transform.

Matrix lifting has been proposed as a solution for implementing reversible transforms with larger transform matrices [14][15]. This has the advantage of reducing the rounding error. Due to space limitations, we have not explored the analogous extension of modulo transforms to higher dimensions, although it can be noted that the properties of the transform coefficients in modulo det(*T*) arithmetic apply equally to larger matrices.

VII. Discussion and Results

So far, we have developed the theoretical framework for realizing reversible integer transforms that do not suffer from volume expansion. Modulo transforms, based on the principle of critical quantization, present an alternative to lifting for implementing lossless codecs and other signal processing algorithms. In theory, both lifting and modulo transforms can achieve a given orthogonal transform to an arbitrary degree of accuracy. Performance metrics such as absolute difference, coding gain etc. are not very meaningful in ranking one over the other. We implemented a lossy / lossless image encoder using both lifting and modulo transform techniques. Neither showed a statistically significant superiority in terms of the rate-distortion metric. Small tweaks to the entropy coding component masked any differences in the transform – moreover it was necessary to tune the entropy coding for each transform type in light of the basis function norms and rounding error behavior. Our conclusion is that for the purpose of data compression, both lifting and modulo transforms fare equally well.

From the computation perspective, lifting is by and large less demanding than modulo transforms and may be considered superior on that account. Lifting does suffer from cascading delays due to multiple lifting steps. However, modulo transforms have to deal with integer division by a constant odd number. These divisions may be expensive to implement on conventional computational hardware, although we provide some suggestions for speeding up divisions in the next section. It may also be kept in mind that non bitexact decoders may be implemented without integer division for decoders based on modulo transforms where a mildly inexact reconstruction is acceptable.

One of the fundamental applications of the 2 point rotation transform is, obviously, for planar rotation. The evolution of the lifting algorithm can be traced back to the three shear algorithm for implementing image rotations for computer graphics [8]. Although the primary aim of the three shear algorithm is to facilitate computational speed and parallelism, the lifting steps provide reversibility to the rotation operation by forming a one-to-one mapping between the original image plane coordinates and the rotated destination image coordinates. This is useful for applications such as image editing where rotations can be reversed so as to recover the original data. Implementing reversible image rotations using modulo transforms and using lifting allows us to contrast them visually, and to highlight a key difference between the two.

The three point lifting algorithm is based on a cascade of three steps – therefore it may be surmised that the rounding error due to shear steps accumulates with the number of steps. In the worst case, the absolute rounding error of one of the components can be as high as 1, and 0.5 for the other component. The elementary modulo transform is not a cascade of steps. From the theoretical analysis presented here, it can be seen that the rounding error is bounded by 0.5 in absolute value for each component. The

question is whether this has any bearing on the quality of a rotated image, and evidence for an answer can be easily evaluated by an experiment.

We choose a rotation angle of π/8. This is implemented using a modulo transform defined by the Pythagorean triad (5, 12, 13), and by a lifting scheme that uses floating point arithmetic for the shear computation with rounding. Figure 15 (a) and (b) show the image "lena" rotated using modulo transform and lifting respectively. The effect of rounding error can be seen in the latter, particularly at linear edges such as the hat, face and also image periphery. The modulo transform is visually superior to lifting in this respect.

Further, a second rotation operation is repeated on the rotated images. Figure 16 (a) and (b) respectively show "lena" rotated twice using the modulo transform and lifting steps described above, to produce an image with a rotation angle of approximately π/4. As before, the difference in subjective quality is evident. Figure 16 (c) shows "lena" rotated by π/4 using one set of lifting steps (i.e. three shears). Subjectively, this is equivalent to the two stage modulo transform. Finally, Figure 17 (a) and (b) respectively show the error buildup over four stages of rotation using modulo transform and lifting. The overall rotation angle is approximately π/2, and the image is right angle rotated back to its original orientation in the figure. The subjective quality difference can be easily seen throughout the image, and one can conclude that the rounding behavior of modulo transforms is less visually objectionable than that due to lifting.

VIII. Computational Aspects

Modulo transforms, more specifically those based on Pythagorean triads satisfying the conditions of Theorem 6, are implemented with integer multiplications and integer divisions. Without loss of generality, let us assume that $c = \sqrt{D} - 1$. Therefore, the two point modulo transform is generated by the following

$$\begin{aligned} y_1 &= \mathrm{sdiv}\left(c\,x_1 + s\,x_2,\, c+1\right) \\ y_2 &= \mathrm{sdiv}\left(-s\,x_1 + c\,x_2,\, c+1\right) \end{aligned} \tag{26}$$

A commonly used method of realizing integer division by a constant is by doing a multiply and shift. The multiplier is related to the reciprocal of the divisor (in this case $c + 1$). An offset is added prior to shift to do rounded division as required here. The size of the accumulator for the multiply operation is the bit range of the dividend plus a constant. Assuming the input range is $\pm R$ in each dimension, the dividend range in (27) is $\pm(c + s)\cdot R$. The dividend range can be reduced to $\pm(s + 1)\cdot R$ noting that (27) is equivalent to

$$\begin{aligned} y_1 &= x_1 + \mathrm{sdiv}\left(-x_1 + s\,x_2,\, c+1\right) \\ y_2 &= x_2 - \mathrm{sdiv}\left(s\,x_1 + x_2,\, c+1\right) \end{aligned} \tag{27}$$

Although (28) still involves two divisions, the multiply and shift division can be implemented using a smaller accumulator. Further, let the quotient and remainder of the first division be q_1 and r_1 (the remainder is trivial to compute given the quotient). Therefore, we have

$$-x_1 + s\,x_2 = q_1(c+1) + r_1 \tag{28}$$

Multiplying both sides by s, and noting the relationship between s and c, it can be shown that

$$s\,x_1 + x_2 = (c+1)\cdot(2\,x_2 - s\,q_1) - s\,r_1 \tag{29}$$

Therefore, the second division can be written as

$$\text{sdiv}\left(s\,x_1 + x_2,\, c+1\right) = \left(2\,x_2 - s\,q_1\right) - \text{sdiv}\left(s\,r_1,\, c+1\right) \tag{30}$$

The range of the dividend on the right hand side is bounded by a constant value $\pm s \cdot c/2$, and therefore it can be implemented with either a much smaller accumulator, or by using a lookup table.

Although it may appear at the first glance that lifting is easy to implement because of the lack of the division operator, this conclusion is conditional on certain observations. First, the lifting step which is the off diagonal shear term must be implemented in high precision arithmetic for the desired rotation to be accurate. Integerized approximations are often used, but they result in an effective linearized lifting transformation that is not a pure rotation. In contrast, the modulo transform *always* results in a rotation – even when the rotation angle is only approximately achieved. Second, the lifting based butterfly consists of a cascade of three steps which must be completed in sequence for the final output. On the other hand, the modulo transform components can be independently evaluated without being dependent upon intermediate values in the signal flow graph. This means that the modulo transform has less latency – an issue which may favor overall system delay and computational performance. Finally, the division-by-constant operation in modulo transforms may be speeded up by a variety of means [23][24][25], or alternatively the transform may be implemented in floating point arithmetic as shown in Figure 1, as a means of reducing the computational burden for VLSI or on certain architectures.

IX. Conclusion

The paper presents a new paradigm for achieving lossless and efficient transforms that is built on reversible butterfly operations. We have explored the theoretical aspects of integer transforms and have presented some interesting properties in modular arithmetic. Based on these properties, we have shown that certain integer rotations can be critically quantized with no loss of information – this forms the basis of modulo transforms. Modulo transforms are nonexpansive and are therefore suitable for use in the lossless compression of data as a viable alternative to lifting. We have not observed any statistical advantage or disadvantage in using modulo transforms as opposed to lifting for data compression. Yet we are able to contrast them in the reversible rotation of images, and the benefit of modulo transforms can be easily visualized.

We have also attempted to address some issues with the implementation of modulo transforms to make it computationally competitive with lifting. In summary, modulo transforms present a means of achieving reversible integer transforms, and may be a viable alternative to lifting in certain situations. Since this is a new paradigm in transform theory, we expect further investigation by the community to address relevant issues and to fully derive the benefits of modulo transform for real world applications.

Appendix A – Proof of Theorem 6

The terms *tile* and *bin* are used interchangeably and refer to a contiguous square area in the transform domain. Tile boundaries are not coincident with lattice points. The entire transform domain is partitioned into regularly spaced tiles. Unless otherwise mentioned, tiles form a square grid.

Definition 10: The *origin tile* is defined as the $\sqrt{D} \times \sqrt{D}$ tile in the transform domain centered at the origin.

Definition 11: The *local coordinate* is the relative coordinate of a point within its $\sqrt{D} \times \sqrt{D}$ tile. Therefore, the local coordinate is a vector that takes on values within the origin tile. The local coordinate (z_1, z_2) of (y_1, y_2) is given by the following:

$$(z_1, z_2) = \left(\text{smod}\left(y_1, \sqrt{D}\right), \text{smod}\left(y_2, \sqrt{D}\right)\right) \tag{31}$$

Lemma 3: All constellation points have a local coordinate (z_1, z_2) that satisfies $c\,z_2 + s\,z_1 = k\sqrt{D} \bmod D$ for an integer k that is fixed for a particular tile. The condition

$$c\,z_2 + s\,z_1 = k\sqrt{D} \bmod D \tag{32}$$

is referred to as the *local modulo relationship* (LMR). Equivalently for an arbitrary l,

$$c\,z_2 + s\,z_1 = k\sqrt{D} + l\,D \tag{33}$$

Proof: The modulo relationship for the constellation point (y_1, y_2) is $c\,y_2 + s\,y_1 = 0 \bmod D$. Setting

$$y_i = k_i\sqrt{D} + \text{smod}\left(y_i, \sqrt{D}\right) = k_i\sqrt{D} + z_i \text{ for } i = 1 \text{ and } 2,$$

the modulo relationship can be written as

$$\begin{aligned} c\,y_2 + s\,y_1 &= (c\,k_2 + s\,k_1)\sqrt{D} + (c\,z_2 + s\,z_1) = 0 \bmod D \\ \Rightarrow (c\,z_2 + s\,z_1) &= -(c\,k_2 + s\,k_1)\sqrt{D} \bmod D \\ \Rightarrow (c\,z_2 + s\,z_1) &= \left(\left(-(c\,k_2 + s\,k_1) \bmod \sqrt{D}\right)\cdot\sqrt{D}\right) \bmod D \\ &= \left(k\cdot\sqrt{D}\right) \bmod D \end{aligned}$$

Since k_1 and k_2 are constants for a certain tile, the value of k is also constant for that tile, proving the Lemma.

Definition 12: *Local constellation points* are defined as points lying in the $\sqrt{D}\times\sqrt{D}$ origin tile that satisfy the LMR equation. The *residue* of a local constellation point is the value k in its corresponding LMR equation (33).

Definition 13: The *modulo offset* of a tile **T** is the value $k \in \mathbf{Z}_{\sqrt{D}}$ given by the following equation, where **T** is situated at an offset of $\left(k_1\sqrt{D}, k_2\sqrt{D}\right)$ from the origin tile.

$$\left(-(c\,k_2 + s\,k_1) \bmod \sqrt{D}\right) = k \tag{34}$$

Lemma 4: Any point (z_1, z_2) satisfying LMR, i.e. any local constellation point, implies the existence of a constellation point (y_1, y_2) such that the local coordinate of (y_1, y_2) is (z_1, z_2).

Proof: Pick $(y_1, y_2) = (z_1, z_2) - \sqrt{D}\,(0, c^{-1}k)$ where k is the corresponding variable of the LMR for (z_1, z_2), and the inverse of c is in modular $\sqrt{D}$ arithmetic. The modulo relationship at (y_1, y_2) evaluates to

$$\begin{aligned} c\,y_2 + s\,y_1 &= (c\,z_2 + s\,z_1) - c\,k\,c^{-1}\sqrt{D} \\ &= \left(k\sqrt{D} + n_1\,D\right) - k\sqrt{D}\left(1 + n_2\sqrt{D}\right) \\ &= (n_1 - k\,n_2)D = 0 \bmod D \end{aligned}$$

for arbitrary integers n_1 and n_2. From Theorem 3, (y_1, y_2) is a constellation point, thereby proving the statement.

Lemma 5: If the LMR is satisfied for a certain $k \in \mathbf{Z}_{\sqrt{D}}$ by two distinct local coordinate points (z_1, z_2) and $(\hat{z}_1, \hat{z}_2)$, then a certain tile **T** contains two constellation points, and vice versa.

Proof: Assume $c\,z_2 + s\,z_1 = k\sqrt{D} \bmod D$ and $c\,\hat{z}_2 + s\,\hat{z}_1 = k\sqrt{D} \bmod D$ for a certain $k \in \mathbf{Z}_{\sqrt{D}}$. The points $\left(z_1, z_2 - k\left(c^{-1}\right)\sqrt{D}\right)$ and $\left(\hat{z}_1, \hat{z}_2 - k\left(c^{-1}\right)\sqrt{D}\right)$ both satisfy the modulo relationship and therefore are valid constellation points. These points also lie on the same tile, proving the first statement. The converse is trivial.

Lemma 6: The modulo offset k in equation (34) corresponding to any $\sqrt{D}$ horizontally or vertically adjacent tiles takes on all values in $\mathbf{Z}_{\sqrt{D}}$.

Proof: Consider the equation (34). Since c, s and $\sqrt{D}$ do not have any common factors, the $\sqrt{D}$ successive values of $-c\,k_2$ or $-s\,k_1$ evaluate to distinct integers in modulo $\sqrt{D}$ arithmetic, not necessarily in any order.

When the transform domain is partitioned into a square grid, it can be shown that the sequence of local coordinates of constellation points in a horizontal row of tiles repeats for each successive row with a shift (and likewise for vertical columns). The shift is zero at every $\sqrt{D}$ rows or columns, in other words, the local coordinate is periodic in every $\sqrt{D}$ rows or columns (as can be expected, since the whole constellation is periodic in both directions with period D). This observation is not critical to the proof of Theorem 6 and is therefore not proven.

Theorem 8: Each $\sqrt{D}\times\sqrt{D}$ quantization bin or tile in the transform domain contains one and only one constellation point if and only if LMR is satisfied by one local coordinate (z_1, z_2) for each $k \in \mathbf{Z}_{\sqrt{D}}$.

Proof: First, we prove that if each tile contains one and only one constellation point, this implies that the LMR is satisfied by only one local point for each k. Assume that each tile contains one constellation point and that the LMR is satisfied by multiple local points for a certain $k_0 \in \mathbf{Z}_{\sqrt{D}}$. This is in direct violation to Lemma 5, and is therefore impossible. Next, assume that each tile contains one constellation point and that the LMR is not satisfied by any local points for a certain $k_1 \in \mathbf{Z}_{\sqrt{D}}$. Then, there exists a tile $\mathbf{T}$ situated at an offset of $\left(0, -c^{-1}k_1\sqrt{D}\right)$, where the inverse of c is in modular $\sqrt{D}$ arithmetic. The interior lattice points of $\mathbf{T}$ are $\left(z_1, z_2 - c^{-1}k_1\sqrt{D}\right)$. The modulo discriminant of the points in $\mathbf{T}$ is

$$c\left(z_2 - c^{-1}k_1\sqrt{D}\right) + s\,z_1 \bmod D = \left(c\,z_2 + s\,z_1\right) - k_1\sqrt{D} \bmod D \neq 0$$

and therefore $\mathbf{T}$ contains no constellation point, violating the assumption.

In order to prove the converse, assume that the LMR is satisfied once by a local (z_1, z_2) for each $k \in \mathbf{Z}_{\sqrt{D}}$. From Lemma 4 and Lemma 6, any set of $\sqrt{D}$ successive tiles contain a constellation point in each tile, which is possible only when every tile contains at least one constellation point. Moreover since from Theorem 4 there are only D constellation points in a square of $D \times D$, every tile contains exactly one constellation point, proving the theorem.

Theorem 8 relates the existence and uniqueness of constellation points within a tile with the existence of a unique point in the origin tile with a given residue $k \in \mathbf{Z}_{\sqrt{D}}$. It is required that for the conditions of Theorem 8 to hold, the residue of each origin tile point satisfying LMR should evaluate to a unique number, i.e. no two points within the origin tile may have the same residue $k\sqrt{D}$, and all values of $k \in \mathbf{Z}_{\sqrt{D}}$ should occur. Origin tile points not satisfying LMR are irrelevant – in fact, it can be seen that constellation points will never happen with these local coordinates. This provides a rapid check for determining uniqueness of constellation points within a tile, or the permissibility of critical quantization using strict scalar quantization and no tweaking of coefficients.

Theorem 8 is illustrated in Table 4. The transform used is the simplest Pythagorean triad $R(4, 3)$. Residues are computed over the 5×5 origin tile. It can be seen that residues take on each multiple of 5 from -10 to 10 once and only once in the table.

11	-11	-8	**-5**	-2
-10	-7	-4	-1	2
-6	-3	**0**	3	6
-2	1	4	7	**10**
2	**5**	8	11	-11

Table 4: Residues evaluated over the origin tile for $R(4, 3)$ – values satisfying LMR are bolded.

Lemma 7: Local constellation points are distributed symmetrically around the origin, and their residues are odd-symmetric. The origin is always on the local constellation and has the residue zero.

Proof: If (z_1, z_2) is a local constellation point, we have that

$$c\, z_2 + s\, z_1 = k\sqrt{D} \bmod D$$

i.e.

$$c\,(-z_2) + s\,(-z_1) = -k\sqrt{D} \bmod D$$

At the origin, $(z_1, z_2) = (0,0)$, which produces the residue 0, proving the lemma. This is illustrated in the example shown in Table 4.

Lemma 8: If two local constellation points have the coordinates (z_1, a) and $(-z_1, b)$, then $a = -b$.

Proof: From Lemma 7, $(-z_1, -a)$ is a local constellation point. From Lemma 4 and Theorem 4, the $\sqrt{D} \times 1$ and $1 \times \sqrt{D}$ partitions of the origin tile (corresponding to slivers) contain only one local constellation point each. Therefore, $a = -b$.

Proof of Theorem 6: There are four possible conditions that satisfy the assumptions of Theorem 6, viz. $s = \pm\left(\sqrt{D} - 1\right)$ and $c = \pm\left(\sqrt{D} - 1\right)$. However, it is sufficient to consider one case, $c = \sqrt{D} - 1$, since the other cases are obtained by switching the signs of c and s (i.e. a mirror flip), and/or by reversing their roles (i.e. a transpose), both of which are reversible operations. We will prove Theorem 6 by contradiction.

Assume that $c = \sqrt{D} - 1$, and a quantization bin or tile **T** contains at least two constellation points thereby violating the statement of Theorem 6. From Lemma 5, there exist at least two points (z_1, z_2) and $(\hat{z}_1, \hat{z}_2)$ both of which satisfy the LMR for a certain $k \in \mathbf{Z}_{\sqrt{D}}$. Let us pick the two points (z_1, z_2) and $(\hat{z}_1, \hat{z}_2)$ that have the smallest squared distance within the origin tile that produce the residue k.

The LMR for (z_1, z_2) gives us the following

$$c\, z_2 + s\, z_1 = k\sqrt{D} \bmod D$$

Substituting the value of c, we get

$$c\, z_2 + s\, z_1 = \left(\sqrt{D} - 1\right) z_2 + s\, z_1$$

Let us expand out the second term as follows, where $b = \text{smod}\left(s\, z_1, \sqrt{D}\right)$ and $a = \text{sdiv}\left(s\, z_1, \sqrt{D}\right)$:

$$s\, z_1 = a\sqrt{D} + b \tag{35}$$

Then, we have the following condition:

$$c\,z_2 + s\,z_1 = \sqrt{D}\,(z_2 + a) - z_2 + b = k\sqrt{D} \bmod D \tag{36}$$

Since the modulo operation smod is defined as being zero-centered, we restrict $\frac{1-\sqrt{D}}{2} \le z_2, b \le \frac{\sqrt{D}-1}{2}$, which limits the value of $z_2 - b$ to

$$1-\sqrt{D} \le z_2 - b \le \sqrt{D} - 1$$

The only solution to equation (37) that produces zero modulo $\sqrt{D}$ is $b = z_2$, simplifying it to

$$\sqrt{D}\,(b+a) = k\sqrt{D} \bmod D$$

which further reduces to

$$(k - a - b) = 0 \bmod \sqrt{D} \tag{37}$$

Likewise, substituting

$$s\,\hat{z}_1 = \hat{a}\sqrt{D} + \hat{b}$$

we get the following relationship for the point $(\hat{z}_1, \hat{z}_2)$

$$\left(k - \hat{a} - \hat{b}\right) = 0 \bmod \sqrt{D} \tag{38}$$

Subtracting equation (39) from (38) gives us

$$\begin{aligned} &\left(a + b - \hat{a} - \hat{b}\right) = 0 \bmod \sqrt{D} \\ &\Rightarrow \left(b - \hat{b}\right) = -(a - \hat{a}) + l\sqrt{D} \end{aligned} \tag{39}$$

for an arbitrary integer *l*. However, since *a* and *b* are range bounded, *l* can only take on three values viz. -1, 0, and 1. Now consider the quantity $s(z_1 - \hat{z}_1)$

$$\begin{aligned} s(z_1 - \hat{z}_1) &= (a - \hat{a})\sqrt{D} + \left(b - \hat{b}\right) \\ &= (a - \hat{a})\left(\sqrt{D} - 1\right) + l\sqrt{D} \\ &= c(a - \hat{a}) + l\sqrt{D} \end{aligned} \tag{40}$$

Case 1 – *l* = 0: When *l* = 0, it is easy to see that the only solution in $\Delta z_1 = (z_1 - \hat{z}_1)$ and $\Delta a = (a - \hat{a})$ is, due to the absence of common factors between *c* and *s*,

$$\begin{aligned} \Delta z_1 &= c\,m \\ \Delta a &= s\,m \end{aligned} \tag{41}$$

Since Δz_1 is range bounded to $\pm\left(\sqrt{D} - 1\right)$, the only valid solutions for Δz_1 are $\Delta z_1 = 0$ or $\Delta z_1 = \pm\left(\sqrt{D} - 1\right)$. In the first case, the points (z_1, z_2) and $(\hat{z}_1, \hat{z}_2)$ are identical and thereby violate the underlying assumption. When $\Delta z_1 = \pm\left(\sqrt{D} - 1\right)$, z_1 and $\hat{z}_1$ are uniquely determined to be $\pm\frac{\sqrt{D}-1}{2}$. From Lemma 7 and Lemma 8, the residue at (z_1, z_2) and $(\hat{z}_1, \hat{z}_2)$ is zero. This violates the assumption that the points (z_1, z_2) and $(\hat{z}_1, \hat{z}_2)$ have the smallest squared distance between themselves, since the point (0, 0) is a local constellation point with zero residue and lies between (z_1, z_2) and $(\hat{z}_1, \hat{z}_2)$.

Case 2 – *l* = ±1: When *l* = ±1, the equation is a linear diophantine equation in $\Delta z_1 = (z_1 - \hat{z}_1)$ and $\Delta a = (a - \hat{a})$ as shown

$$s\,\Delta z_1 - c\,\Delta a = \pm\sqrt{D} \tag{42}$$

This equation has the following integer solutions for Δz_1 and Δa :

$$\begin{aligned} \Delta z_1 &= \pm s + c\,m \\ \Delta a &= \pm 1 + s\,m \end{aligned} \tag{43}$$

for an arbitrary integer *m*. Since Δz_1 is range bounded to $\pm\left(\sqrt{D}-1\right)$, i.e. to ± *c*, the only integers that produces meaningful solutions for Δz_1 are *m* = 0, and *m* = –1 when $s > 0$ and *m* = 1 when $s < 0$. The corresponding solutions of Δa are 1 and $1-|s|$.

For the rest of this discussion, we will assume that *l* = 1. The case of *l* = –1 merely interchanges the two points under consideration.

Also by subtracting the LMR equation (33) for *z* and $\hat{z}$, we get

$$s\left(z_1 - \hat{z}_1\right) + c\left(z_2 - \hat{z}_2\right) = 0 \bmod D \tag{44}$$

From (42) and (44), we get the relationship

$$c\left(\Delta z_2 + \Delta a\right) + \sqrt{D} = 0 \bmod D$$

where $\Delta z_2 = \left(z_2 - \hat{z}_2\right)$ and $\Delta a = \left(a - \hat{a}\right)$. Replacing *c* by $\sqrt{D}-1$, we get

$$-\left(\Delta z_2 + \Delta a\right) + \left(\Delta z_2 + \Delta a + 1\right)\sqrt{D} = 0 \bmod D \tag{45}$$

Δz_2 is likewise range bounded to $\pm\left(\sqrt{D}-1\right)$. Let *q* denote the term $\left(\Delta z_2 + \Delta a\right)$, expressed as

$$\left(\Delta z_2 + \Delta a\right) = q = t_2 + t_1\sqrt{D} \tag{46}$$

where $t_2 = \mathrm{smod}\left(q, \sqrt{D}\right)$ and $t_1 = \mathrm{sdiv}\left(q, \sqrt{D}\right)$. Equation (46) can be rewritten as

$$\begin{aligned} -t_2 - t_1\sqrt{D} + \left(t_2 + t_1\sqrt{D} + 1\right)\sqrt{D} &= lD \\ \Rightarrow t_1 D + \left(1 + t_2 - t_1\right)\sqrt{D} - t_2 &= lD \end{aligned} \tag{47}$$

for an arbitrary *l*. This equation has only one integer solution, viz. $t_2 = 0, t_1 = 1$. Therefore, equation (46) has the only solution

$$\left(\Delta z_2 + \Delta a\right) = t_2 + t_1\sqrt{D} = \sqrt{D} \tag{48}$$

We know that Δa satisfies equation (43) and its only integer solutions are $\Delta a = 1$ and $\Delta a = 1 - |s|$. The corresponding solutions for Δz_2 in equation (48) are $\Delta z_2 = \sqrt{D} - 1$ and $\Delta z_2 = \sqrt{D} - 1 + |s|$. The latter is an invalid solution since Δz_2 is range bounded. The only valid solution for Δz_2 is therefore $\Delta z_2 = \sqrt{D} - 1$, and this uniquely identifies z_2 and $\hat{z}_2$ to be $\pm\dfrac{\sqrt{D}-1}{2}$.

As for the case when *l* = 0, from Lemma 7 and Lemma 8, the residue at $\left(z_1, z_2\right)$ and $\left(\hat{z}_1, \hat{z}_2\right)$ is zero. This violates the assumption that the points $\left(z_1, z_2\right)$ and $\left(\hat{z}_1, \hat{z}_2\right)$ have the smallest squared distance between themselves, since the point (0, 0) is a local constellation point with zero residue and lies between $\left(z_1, z_2\right)$ and $\left(\hat{z}_1, \hat{z}_2\right)$. This proves Theorem 6.

Conjecture: No primitive Pythagorean triples not satisfying $c = D - 1$ can be critically quantized using uniform quantizers with square bins of size $\sqrt{D} \times \sqrt{D}$. This conjecture is not critical to the fundamental theory introduced in this paper. However, it is probably of academic interest to the reader in this context.

REFERENCES

[1] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transform", *IEEE Trans. Comp.*, vol. C-23, pp. 90-93, Jan 1974.

[2] A. K. Jain, *Image Processing*, Prentice Hall, 1989.

[3] F. A. M. L. Bruekers, and A. W. M. van den Enden, "New Networks for Perfect Inversion and Perfect Reconstruction", in *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 1, Jan 1992.

[4] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in *Proc. SPIE 2569*, 1995, pp. 68-79.

[5] A. R. Calderbank, I. Daubechies, W. Sweldens, and B-L. Yeo, "Lossless Image Compression using Integer to Integer Wavelet Transforms," in *Proc. IEEE Intl. Conf. on Image Processing*, vol. 1, pp. 596-599, 1997.

[6] G. Plonka, "A Global Method for Invertible Integer DCT and Integer Wavelet Algorithms", in *Applied Computational and Harmonic Analysis*, vol. 16, pp.90-110, March 2004.

[7] P. Roos, M. A. Viergever, M. C. A. van Dijke and J. H. Peters, "Reversible Intraframe Compression of Medical Images", in *IEEE Trans. Medical Imaging*, vol 7, pp. 328-336, Dec 1988.

[8] A. W. Paeth, "A Fast Algorithm for General Raster Rotation", in *Proc Graphics Interface*, May 1986, pp. 77-81.

[9] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity Transform and Quantization in H.264/AVC", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 598–603, July 2003.

[10] G. A. Jones, and J. M. Jones, *Elementary Number Theory*, Springer, 1998.

[11] N. J. Vilenkin, "Special Functions and the Theory of Group Representations", in *Translations of Mathematical Monographs*, vol. 22, American Math. Soc., Providence, 1968.

[12] C. Loeffler, A. Lightenberg, and G. Moschvtz, "Practical Fast 1D DCT Algorithm with 11 Multiplications", *Proc. ICASSP*, vol. 2, pp. 988-991, Feb. 1989.

[13] W. Chen, C. Harrison, and S. Fralick, "A fast Computational Algorithm for the Discrete Cosine Transform", in *IEEE Trans. Communications*, vol. 25, pp. 1004–1011, 1977.

[14] P. Hao and Q. Shi, "Matrix Factorizations for Reversible Integer Mappings", in *IEEE Trans. Signal Proc.*, vol. 49, Oct 2001, pp. 2314-2324.

[15] J. Li, "Reversible FFT and MDCT via Matrix Lifting", in *Proc. IEEE ICASSP*, vol. 4, pp. 173-176, May 2004.

[16] G. H. Golub and C. F. van Loan, *Matrix Computations*, 2nd edn., The John Hopkins University Press, 1989.

[17] J. Hong and M. Vetterli, "Hartley Transforms Over Finite Fields", in *IEEE Trans. Info. Theory*, vol. 39, Sept 1993, pp. 1628-1638.

[18] K. Komatsu and K. Sezaki, "Reversible Discrete Cosine Transform," in *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, Seattle, 1998, pp. 1769-1772.

[19] X. Li, B. Tao, and M. T. Orchard, "On Implementing Transforms from Integers to Integers", in *Proc. IEEE Intl. Conf. on Image Processing*, Chicago, 1998, vol.3, pp. 881-885.

[20] A. Tanaka, M. Kameyama, S. Kazama, and O. Watanabe, "A Rotation Method for Raster Image Using Skew Transformation," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, June 1986, pp. 272-277.

The author gratefully acknowledges the support and encouragement provided by Henrique Malvar and colleagues at Microsoft Corporation, in the preparation of this paper.

[21] T. D. Tran, "The binDCT: Fast multiplierless approximation of the DCT," *IEEE Signal Processing Letters*, vol. 7, pp. 141-144, June 2000.

[22] J. Liang and T. D. Tran, "Fast Multiplierless Approximation of the DCT with the Lifting Scheme", *IEEE Trans. Signal Processing*, vol. 49, pp. 3032-3044, Dec. 2001.

[23] R. Alverson, "Integer Division using Reciprocals", in *Proc 10th Symposium on Computer Arithmetic*, pp. 186-190, Grenoble, June 1991.

[24] T. Granlund and P. L. Montgomery, "Division by Invariant Integers using Multiplication", in *Proc. ACM SIGPLAN '94 Conf. on Programming Language Design and Implementation*, pp. 61-72, Orlando, 1994.

[25] S. F. Oberman and M. J. Flynn, "Division Algorithms and Implementations", in *IEEE Trans. Computers*, vol. 46, pp. 833-854, August 1997.

FIGURES

(a) (b)

Figure 15: Lena rotated by π/8 – (a) critically quantized *R* (12, 5) rotation and (b) lifting with floating point arithmetic.

(a) (b)

(c)

Figure 16: Lena rotated by π/4 – (a) critically quantized *R* (12, 5) rotation performed twice, (b) two stages of lifting, each rotating by π/8, and (c) single stage of lifting with floating point arithmetic.

(a)

(b)

Figure 17: Lena rotated by ≈π/2, and then reoriented – (a) critically quantized *R* (12, 5) rotation applied four times in succession and (b) lifting by π/8 applied four times in succession.