

Image Processing with Wreath Product Groups

Will Chang

Michael Orrison, Advisor

Weiqing Gu, Reader

May, 2004

HARVEY MUDD
COLLEGE

Department of Mathematics

Contents

1	Introduction	1
2	Representation Theory	3
2.1	Representations of Groups	3
2.2	An Algebraic Interpretation of the DFT	7
3	Wreath Product Groups and the Quad Tree Scan	11
3.1	Wreath Product Groups	11
3.2	Quad Tree Decompositions of Images	12
3.3	A Decomposition	14
3.4	The Quad Tree Scan	15
	Bibliography	17

Chapter 1

Introduction

Image processing refers to the various operations performed on pictures that are digitally stored as an aggregate of pixels. One can enhance or degrade the quality of an image, artistically transform the image, or even find or recognize objects within the image.

This paper is concerned with image processing, but in a very mathematical perspective, involving representation theory. The approach traces back to Cooley and Tukey's seminal paper on the Fast Fourier Transform (FFT) algorithm (1965). Recently, there has been a resurgence in investigating algebraic generalizations of this original algorithm with respect to different symmetry groups.

My approach in the following chapters is as follows. First, I will give necessary tools from representation theory to explain how to generalize the Discrete Fourier Transform (DFT). Second, I will introduce wreath products and their application to images. Third, I will show some results from applying some elementary filters and compression methods to spectrums of images. Fourth, I will attempt to generalize my method to non-cyclic wreath product transforms and apply it to images and three-dimensional geometries.

Chapter 2

Representation Theory

In this chapter, we will briefly cover the basic tools of representation theory. A more complete treatment can be found in the later chapters of (6). Much of the material of the section on the algebraic interpretation of the DFT is found (concisely presented) in (8), (15), and (19).

2.1 Representations of Groups

Definition 2.1. Let G be a finite group, let F be a field and let V be a vector space over F .

1. A *linear representation* of G is any homomorphism φ from G into $GL(V)$.
2. Let $n \in \mathbb{Z}^+$. A *matrix representation* of G is any homomorphism from G into $GL_n(F)$.

A representation for a group is just a map from group elements in G to linear transformations in $GL(V)$. When V is a finite vector space, we can fix a basis for V and obtain an isomorphism from $GL(V)$ to $GL_n(V)$. This paper only concerns finite dimensional vector spaces, so the word “representation” really means either a linear or matrix representation.

Definition 2.2. The *group ring* of G over F is the set of all formal sums of the form

$$\sum_{g \in G} \alpha_g g, \quad \alpha_g \in F$$

where addition and multiplication are performed componentwise.

We call the above group ring FG . We can identify a representation with an FG -module.

Definition 2.3. Let R be a ring (not necessarily commutative nor with 1). A *left R -module* or a *left module over R* is a set M together with

1. a binary operation $+$ on M under which M is an abelian group, and
2. an action of R on M (that is, a map $R \times M \rightarrow M$) denoted by rm , for all $r, s \in R$ and for all $m, n \in M$ which satisfies
 - (a) $(r + s)m = rm + sm$,
 - (b) $(rs)m = r(sm)$,
 - (c) $r(m + n) = rm + rn$,
 - (d) and if the ring R has a 1, then we require $1m = m$ as well.

An FG -module refers to a module over the group ring FG , where the set involved is the vector space V over F . These modules are important because we can make a direct correspondence between representations and FG -modules.

Suppose $\varphi : G \rightarrow GL(V)$ is a representation of G on the vector space V over F . Then, we can make V into an FG -module by using the identification

$$\left(\sum_{g \in G} \alpha_g g \right) \cdot v = \sum_{g \in G} \alpha_g \varphi(g)(v) \quad \text{for all } \sum_{g \in G} \alpha_g g \in FG, v \in V.$$

Conversely, given an FG -module V , we obtain an associated vector space over F and representation by defining

$$\varphi(g)(v) = g \cdot v \quad \text{for all } v \in V,$$

where $g \cdot v$ is the action of the group element g in the element v of V .

Thus, giving a representation $\varphi : G \rightarrow GL(V)$ on a vector space V over F is therefore equivalent to giving an FG -module V , and we say that V *affords* the representation of G .

Definition 2.4. Let R be a ring and M be an R -module. An *R -submodule* of M is a subgroup N of M which is closed under the action of ring elements, i.e., $rn \in N$ for all $r \in R, n \in N$.

Definition 2.5. Let M be a nonzero R -module for a ring R .

1. M is said to be *irreducible* if its only submodules are 0 and M ; otherwise, M is called *reducible*.
2. M is said to be *indecomposable* if M cannot be written as $M_1 \oplus M_2$ for any nonzero submodules M_1 and M_2 ; otherwise, M is called *decomposable*.

3. M is said to be *completely reducible* if it is a direct sum of irreducible submodules.

Similarly, representations are irreducible or indecomposable according to whether their corresponding modules have those qualities.

Now we are ready to present some definitions and theorems that lead up to Wedderburn's Theorem.

Definition 2.6. The *characteristic* of a field F is the smallest positive integer p such that $p \cdot 1_F = 0$ if such a p exists, and 0 otherwise.

Theorem 2.1. (Maschke's Theorem) *Let G be a finite group and let F be a field whose characteristic does not divide $|G|$. If V is any FG -module and U is any submodule of V , then V has a submodule W such that $V = U \oplus W$.*

Corollary. *If G is a finite group and F is a field whose characteristic does not divide $|G|$, then every finitely generated FG -module is completely reducible.*

Note that in particular, for any finite group G and $F = \mathbb{C}$, every $\mathbb{C}G$ -module is finitely generated and thus completely reducible.

Lemma 2.2. (Schur's Lemma) *If M and N are irreducible (or simple) R -module and $\varphi : M \rightarrow N$ is a nonzero R -module homomorphism, then φ is an isomorphism.*

Theorem 2.3. (Wedderburn's Theorem) *Let R be a nonzero ring with 1. If every R -module is completely reducible, then the ring R considered as a left R -module is a direct sum:*

$$R \cong L_1 \oplus L_2 \oplus \cdots \oplus L_n,$$

where each L_i is a simple module with $L_i = Re_i$, for idempotents $e_i \in R$ which satisfy

1. $e_i e_j = 0$ if $i \neq j$
2. $e_i^2 = e_i$ for all i
3. $\sum_{i=1}^n e_i = 1$.

This theorem guarantees us a decomposition of any $\mathbb{C}G$ -module M as a direct sum of irreducible submodules:

$$M \cong a_1 M_1 \oplus a_2 M_2 \oplus \cdots \oplus a_r M_r,$$

where each a_i is a nonnegative integer indicating the multiplicity of the irreducible module M_i , i.e.

$$a_i M_i = \overbrace{M_i \oplus \cdots \oplus M_i}^{a_i \text{ times}}$$

The above decomposition is called an *isotypic* decomposition and each $a_i M_i$ are called *isotypics*.

Now, there remains the question of finding the idempotents, e_i . To explain this, we must delve into a bit of character theory.

Definition 2.7. A *class function* is any function from G to F which is constant on the conjugacy classes of G .

Definition 2.8. If φ is a representation of G afforded by the FG -module V , the *character* of φ is the function

$$\chi : G \rightarrow F \quad \text{defined by} \quad \chi(g) = \text{tr } \varphi(g)$$

where $\text{tr } \varphi(g)$ is the trace of the matrix of $\varphi(g)$ with respect to some basis of V .

By direct computation, $\text{tr } AB = \text{tr } BA$ for two $n \times n$ matrices A and B . If A is invertible, this implies that

$$\text{tr } A^{-1}BA = \text{tr } B.$$

Thus, we have that the character of a representation is independent of the choice of basis of the vector space affording it. We can also show that characters are class functions, since

$$\chi(g^{-1}xg) = \text{tr } (\varphi(g^{-1}xg)) = \text{tr } (\varphi(g^{-1})\varphi(x)\varphi(g)) = \text{tr } \varphi(x) = \chi(x).$$

Note that if ψ is the character of a representation φ afforded by the $\mathbb{C}G$ -module M (as above), then ψ is actually the sum of the characters of the irreducible submodules in the decomposition, i.e.

$$\psi = a_1\chi_1 + a_2\chi_2 + \cdots + a_r\chi_r.$$

We can put a Hermitian inner product structure on the space of class functions as follows:

Definition 2.9. For class functions θ and ψ , define the inner product to be

$$\langle \theta, \psi \rangle = \langle \psi, \theta \rangle = \frac{1}{|G|} \sum_{g \in G} \theta(g) \overline{\psi(g)}$$

where the bar denotes complex conjugation.

Using the characters, we have the following formula for obtaining idempotents.

Theorem 2.4. Let e_1, \dots, e_r be the orthogonal primitive central idempotents in $\mathbb{C}G$ labelled in such a way that e_i acts as the identity on the irreducible $\mathbb{C}G$ -module M_i , and let χ_i be the character afforded by M_i . Then

$$e_i = \frac{\chi_i(1)}{|G|} \sum_{g \in G} \chi_i(g^{-1})g.$$

Using this formula, we can derive an important orthogonality relation between characters.

Theorem 2.5. Let U and V be non-isomorphic irreducible $\mathbb{C}G$ -modules, with characters χ and ψ , respectively. Then

$$\langle \chi, \chi \rangle = 1, \text{ and } \langle \chi, \psi \rangle = 0.$$

As a consequence of this theorem, we arrive at some key results that will be of use later.

Theorem 2.6. Let V be a $\mathbb{C}G$ -module with character χ . Then V is irreducible if and only if $\langle \chi, \chi \rangle = 1$.

2.2 An Algebraic Interpretation of the DFT

[FIXME: diagram here!] Suppose we have a complex-valued function $f(k)$ that is sampled at (or defined on) n points; i.e. $k \in \{0, \dots, n-1\}$. Then, the classical DFT of this function is given by

Fix me!

$$\widehat{f}(j) = \sum_{k=0}^{n-1} f(k)e^{-2\pi i k \frac{j}{n}}.$$

Similarly, the inverse of this transform, the IDFT, is given by

$$f(k) = \frac{1}{n} \sum_{j=0}^{n-1} \widehat{f}(j)e^{2\pi i k \frac{j}{n}}.$$

For this section, let $G = \mathbb{Z}/n\mathbb{Z}$. Given this function f , we can identify the n points with group elements of $\mathbb{Z}/n\mathbb{Z}$, and therefore each f becomes an element of $\mathbb{C}G$:

$$f = \sum_{g \in G} \alpha_g g.$$

Thus, the space of all functions defined on this domain can be represented as the $\mathbb{C}G$ -module over itself, $\mathbb{C}G$. Let us proceed to decompose this space using Wedderburn's theorem.

Let φ denote the complex representation of $\mathbb{Z}/n\mathbb{Z}$, and let φ_i be the irreducible representations. Let x be the generator for $\mathbb{Z}/n\mathbb{Z}$; thus $\mathbb{Z}/n\mathbb{Z} = \langle n \rangle$. To find a character for our cyclic group, we would like to find a homomorphism between the group elements and the complex numbers. Mapping elements of the cyclic group to the n th roots of unity in \mathbb{C} gives us the appropriate homomorphism. Thus, we define the irreducible characters of $\mathbb{Z}/n\mathbb{Z}$ to be

$$\chi_k(x^j) = e^{2\pi i j \frac{k}{n}}.$$

We know that these characters are irreducible because when we compute the inner product, we get

$$\begin{aligned} \langle \chi_i, \chi_i \rangle &= \frac{1}{|G|} \sum_{g \in G} \chi_i(g) \overline{\chi_i(g)} \\ &= \frac{1}{n} \left(\sum_{j=0}^{n-1} \chi_i(x^j) \overline{\chi_i(x^j)} \right) \\ &= \frac{1}{n} \left(\sum_{j=0}^{n-1} e^{2\pi i j \frac{k}{n}} e^{-2\pi i j \frac{k}{n}} \right) \\ &= 1 \end{aligned}$$

From this information, we can derive the orthogonal primitive central idempotents using our previous formula:

$$\begin{aligned} e_i &= \frac{\chi_i(1)}{|G|} \sum_{g \in G} \chi_i(g^{-1})g \\ &= \frac{1}{n} \sum_{j=0}^{n-1} \chi_i(x^{n-j})x^j \\ &= \frac{1}{n} \sum_{j=0}^{n-1} e^{-2\pi i j \frac{k}{n}} x^j \end{aligned}$$

Let $\omega = e^{\frac{2\pi i}{n}}$. Then, the individual idempotents look like

$$\begin{aligned}
 e_0 &= \frac{1}{n} (1 + x + x^2 + x^3 + \dots + x^{n-1}) \\
 e_1 &= \frac{1}{n} (1 + \omega^{-1}x + \omega^{-2}x^2 + \omega^{-3}x^3 + \dots + \omega^{-(n-1)}x^{n-1}) \\
 e_2 &= \frac{1}{n} (1 + \omega^{-2}x + \omega^{-4}x^2 + \omega^{-6}x^3 + \dots + \omega^{-2(n-1)}x^{n-1}) \\
 e_3 &= \frac{1}{n} (1 + \omega^{-3}x + \omega^{-6}x^2 + \omega^{-9}x^3 + \dots + \omega^{-3(n-1)}x^{n-1}) \\
 &\vdots \\
 e_{n-1} &= \frac{1}{n} (1 + \omega^{-(n-1)}x + \omega^{-2(n-1)}x^2 + \omega^{-3(n-1)}x^3 + \dots + \omega^{-(n-1)(n-1)}x^{n-1})
 \end{aligned}$$

Now, the decomposition into the isotypic submodules of $\mathbb{C}G$ are accomplished by multiplying the idempotents with the desired element from $\mathbb{C}G$. Thus, for a function $f = \sum_{k=0}^{n-1} \alpha_k x^k \in \mathbb{C}G$,

where each $\alpha_k \in \mathbb{C}$. As an example, one of the isotypic components of f is $e_1 \cdot f$: [FIXME: is this wrong?] Fix me!

$$\begin{aligned}
 e_1 \cdot f &= \frac{1}{n} (1 + \omega^{-1}x + \omega^{-2}x^2 + \dots + \omega^{-(n-1)}x^{n-1}) \cdot f \\
 &= \frac{1}{n} (1 + \omega^{-1}x + \omega^{-2}x^2 + \dots + \omega^{-(n-1)}x^{n-1}) \cdot \sum_{k=0}^{n-1} \alpha_k x^k \\
 &= \frac{1}{n} (1 + \omega^{-1}x + \omega^{-2}x^2 + \dots + \omega^{-(n-1)}x^{n-1}) \\
 &\quad \cdot (\alpha_0 + \alpha_1 x + \alpha_2 x^2 + \dots + \alpha_{n-1} x^{n-1}) \\
 &= \frac{1}{n} (\alpha_0 + \omega^{-1}\alpha_1 + \omega^{-2}\alpha_2 + \dots + \omega^{-(n-1)}\alpha_{n-1}) \\
 &= \frac{1}{n} \sum_{k=0}^{n-1} \alpha_k \omega^{-k}.
 \end{aligned}$$

These components look exactly like the Fourier coefficients we saw earlier! We can express the isotypic components more succinctly in matrix form:

$$\frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \dots & \omega^{-(n-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \dots & \omega^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \dots & \omega^{-(n-1)^2} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n-1} \end{pmatrix}$$

This matrix is precisely the DFT matrix.

Chapter 3

Wreath Product Groups and the Quad Tree Scan

We can use wreath product groups to describe automorphisms of trees. In this section, we will describe the wreath product in terms of these automorphisms and see how we can apply the representation theory of these groups to images. This approach was studied closely by (8) on iterated wreath products of cyclic groups. More information on the representation theory of wreath products is in (7), (11) and (12).

3.1 Wreath Product Groups

We begin with a definition of a wreath product.

Definition 3.1. Let G be a finite group, and H a permutation group on n elements (a subgroup of S_n). Let G^n defined by

$$G^n = \overbrace{G \times G \times \cdots \times G}^{n \text{ times}}$$

be the set of ordered n -tuples of elements of G . Then the *wreath product* of G with H , denoted $G \wr H$, is the set $G^n \times H$ with multiplication defined as

$$(g, \sigma)(h, \pi) = \left((g_1 h_{\sigma^{-1}(1)}, \dots, g_n h_{\sigma^{-1}(n)}), \sigma \pi \right)$$

where $g, h \in G^n$ and $\sigma, \pi \in H$.

A wreath product of G and H is really a semidirect product of G^n and H , where the action of H on G^n is to permute the ordering of the n -tuples. More

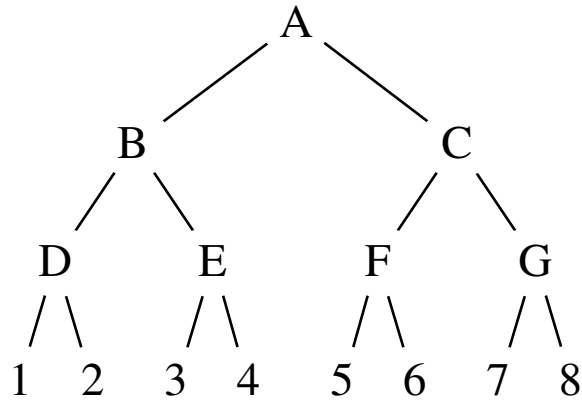


Figure 3.1: A Graph of a Binary Tree of 3 levels.

precisely, $G \wr H$ is a semidirect product of H and G^n with respect to the permutation representation φ from H into $\text{Aut}(G^n)$ (the automorphisms of G^n).

The wreath product turns out to be a convenient way to describe automorphisms of trees. An automorphism of a tree can be thought of as a structured permutations of the leaves of the tree, where subtrees at a given level in the tree are permuted. Consider the figure 3.1 above: a binary tree with height = 3. At each vertex in the tree, we can “swap” the two subtrees connected to it, and this corresponds to permuting “levels” of trees. Notice that this process preserves the overall “hierarchy” of the tree – no matter what the operation, A will always be at the top of the tree, B and C in the second level, D, E, F , and G always in the third level, etc. In addition, we can also create a one-to-one correspondence between the permutations of the leaves and the “swapping” of subtrees. For example, if we want the permutation $(1\ 2)$, then we would simply do a swap operation on the children of vertex D . For $(1\ 3)(2\ 4)$, we can simply perform a swap on vertex B . A more complicated example, $(1\ 6)(2\ 5)(3\ 7)(4\ 8)(5\ 3)(6\ 4)(7\ 1)(8\ 2)$, can be performed by swapping on vertex A , then swapping on vertex C , and finally on vertex D .

3.2 Quad Tree Decompositions of Images

How can automorphisms of trees help us with image processing? We can re-create a tree-like structure from a digital image by recursively breaking the image into pieces, creating a hierarchy within the image. Here’s how it works:

Procedure 3.2. Given a image with dimensions $2^h \times 2^h$,

1. Break the image up into four quadrants, each of dimension $2^{h-1} \times 2^{h-1}$.
2. Order the four quadrants clockwise, starting from the top left quadrant.
3. Continue the process recursively, breaking quadrants into sub-quadrants (of dimension $2^{h-2} \times 2^{h-2}$) and sub-quadrants into sub-sub-quadrants (of dimension $2^{h-3} \times 2^{h-3}$).
4. Stop at the h th recursive step, when quadrants cannot be broken up further since they are made of individual pixels.

Example 3.2.1. Suppose we have an 4×4 image, represented by the matrix

$$\begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix}$$

We first break the image into four quadrants, like so: [FIXME: put another matrix alongside that labels the quadrants] Fix me!

$$\left(\begin{array}{cc|cc} a & b & c & d \\ e & f & g & h \\ \hline i & j & k & l \\ m & n & o & p \end{array} \right)$$

Then, we can break the quadrants recursively:

$$\left(\begin{array}{cc|cc} a & b & c & d \\ e & f & g & h \\ \hline i & j & k & l \\ m & n & o & p \end{array} \right)$$

At this point, we must stop the recursive process since we cannot subdivide the matrix any further. Now, using the ordering scheme, we can order the vertices as follows:

$$\left(\begin{array}{cc|cc} 1 & 2 & 5 & 6 \\ 4 & 3 & 8 & 7 \\ \hline 13 & 14 & 9 & 10 \\ 16 & 15 & 12 & 11 \end{array} \right)$$

This gives rise to the following quadtree: [FIXME: put a diagram here.]

Fix me!

3.3 A Decomposition

Let the quadtree be denoted by $\mathcal{Q}(h)$, where h is the number of levels in the tree, and thus $\mathcal{Q}(h)$ has 4^h leaves. Let the set of leaves of $\mathcal{Q}(h)$ be denoted X , and let $L(X)$ denote the space of all complex valued functions on X . Let G be the iterated wreath product of $\mathbb{Z}/4\mathbb{Z}$:

$$G = \mathbb{Z}/4\mathbb{Z} \wr \mathbb{Z}/4\mathbb{Z} \wr \cdots \wr \mathbb{Z}/4\mathbb{Z},$$

and let V_i denote the space of all functions that are constant on each block of leaves that descend from a common node at level i . (8) describes a G -invariant decomposition of $L(X)$. The key idea in their paper is decomposing $L(X)$ into the subspaces V_i , which gives a complete filtration of $L(X)$ invariant to the group action of G .

This idea is really based on what is called the *Radon transform*. Under this transform, we take the average of a block of four values (corresponding to the four pixel values) and decompose the space of all values into two components: the average value component, and the difference component. Define the Radon transform as

$$\mathcal{R}_j : L(X) \rightarrow V_j \quad \text{defined by} \quad \mathcal{R}_j(f)(x) = \frac{1}{|A_j(x)|} \sum_{x_n \in A_j(x)} f(x_n),$$

where $A_j(x)$ is the set of all pixels within the $2^{h-j} \times 2^{h-j}$ subframe of the image. Here, \mathcal{R}_j just maps the pixel values of $A_j(x)$ to its average value. Now we arrive at a small theorem:

Theorem 3.1. *The Radon transform gives a G -invariant decomposition of V_{j+1} as*

$$V_{j+1} = V_j \oplus W_j \quad \text{for all } j,$$

where W_j is the nullspace of \mathcal{R}_j .

Notice that this gives rise to a recursive algorithm to decompose $L(X)$. At the lowest level (where the quadrants are individual pixels), we can even use the classical DFT to decompose the pixel values, because the 0th coefficient of the DFT (the very first coefficient) is the average or constant component (i.e. the Radon transform component). Thus, the DFT breaks a space of complex functions on a group of four pixels $L(X_0)$ as

$$L(X_0) = V_0 \oplus W_1 \oplus W_2 \oplus W_3,$$

where each subspace is one-dimensional and G -invariant. The space V_0 corresponds to the 0th Fourier coefficient, and W_k corresponds to the k th coefficient.

Now, we can gather these average values and do another DFT decomposition on those values, as prescribed by our theorem above. [FIXME: this section needs a lot more work.]

Algorithm 3.3. We have a recursive algorithm as follows:

1. Divide the image into quadrants
2. Recursively divide quadrants into sub-quadrants
3. At the bottom-most level, perform the DFT on a vector of the four pixel values, ordered according to our ordering scheme
4. Recursively perform DFTs on the first coefficients and travel back up the recursive stack.

3.4 The Quad Tree Scan

The above recursive algorithm is quite simple and elegant, but its recursive nature tends to waste precious computing and storage resources. We can solve the problem much faster using an iterative method.

This method relies on a quick ordering of the pixels on the image. Recall from section 3.2 that an ordering of the pixel values is

$$\begin{pmatrix} 1 & 2 & 5 & 6 \\ 4 & 3 & 8 & 7 \\ 13 & 14 & 9 & 10 \\ 16 & 15 & 12 & 11 \end{pmatrix}$$

Simply “marching” through the pixel values and performing DFTs along the way is faster than the recursive method; which allocates memory for each recursive call, unnecessarily stores intermediary variables, and needlessly performs the image-dividing procedure for each recursive call.

A quick march is obtained as follows. We start on the upper-left-most pixel and attach ordering numbers to the pixel, starting from 1. In addition to this “counter,” denoted by i , we have three more variables: the row and column position of the pixel (r, c) (initially $(1, 1)$), and the current “level” l of the pixel (we begin at the 0th level).

Procedure 3.4. At each pixel, perform a test on the pixel number:

1. Find the largest non-negative integer k such that 4^k divides i .

2. If $k = 1$, we have “completed” a level and must move on to the next level. Thus, reset $(r, c) = (1, 2^l + 1)$, and increment l by 1.
3. If $k \neq l$, compute $m = (i/4^k) \bmod 4$. Here, k represents a “local level” of sorts, and m tells you the relative position within that local level. First, we reset our row and column positions back to the first pixel in our local level; this can be accomplished by decrementing r by $2^k - 1$ and leaving c fixed. Now, we change the row and column positions based on the value of m , which has three possible values:
 - (a) If $m = 1$, then increment c by 2^k ,
 - (b) If $m = 2$, then increment r by 2^k ,
 - (c) If $m = 3$, then decrement c by 2^k .

Notice that the case where $m = 4$ cannot occur, since k was defined to be the *largest* integer such that 4^k divides i .

Bibliography

- [1] Ulrich Baum and Michael Clausen. Computing irreducible representations of supersolvable groups. *Mathematics of Computation*, 63(207):351–359, July 1994.
- [2] Michael Clausen and Ulrich Baum. *Fast Fourier Transforms*. BI-Wissenschaftsverlag, Mannheim, 1993.
- [3] Michael Clausen and Ulrich Baum. Fast fourier transforms for symmetric groups: Theory and implementation. *Mathematics of Computation*, 61(204):833–847, October 1993.
- [4] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19:297–301, April 1965.
- [5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.
- [6] David S. Dummit and Richard M. Foote. *Abstract Algebra*. Upper Saddle River, N.J., Prentice Hall, second edition, 1999.
- [7] Nathaniel Eldredge. An eigenspace approach to isotypic projections for data on binary trees. Undergraduate Thesis, Harvey Mudd College, 2003.
- [8] R. Foote, G. Mirchandani, D. Rockmore, D. Healy, and T. Olson. A wreath product group approach to signal and image processing. I. Multiresolution analysis. *IEEE Transactions on Signal Processing*, 48(1):102–132, January 2000.
- [9] David M. Goldschmidt. *Group Characters, Symmetric Functions, and the Hecke Algebra*. American Mathematical Society, University Lecture Series, 1993.

- [10] Gordon James and Martin Liebeck. *Representations and Characters of Groups*. Cambridge University Press, second edition, 2001.
- [11] Adalbert Kerber. *Representations of Permutation Groups I*, volume 240 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1971.
- [12] Adalbert Kerber. *Representations of Permutation Groups II*, volume 495 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1975.
- [13] Haeyoung Lee, Mathieu Desbrun, and Peter Schröder. Progressive encoding of complex isosurfaces. In *Proceedings of ACM SIGGRAPH*, volume 22, pages 471–476, July 2003.
- [14] David K. Maslen and Daniel N. Rockmore. Separation of variables and the computation of fourier transforms on finite groups, i. *Journal of the American Mathematical Society*, 10(1):169–214, January 1997.
- [15] David K. Maslen and Daniel N. Rockmore. The cooley-tukey fft and group theory. *Notices of the American Mathematical Society*, 48(10):1151–1160, November 2001.
- [16] G. Mirchandani, R. Foote, D. Rockmore, D. Healy, and T. Olson. A wreath product group approach to signal and image processing. II. Convolution, correlation, and applications. *IEEE Transactions on Signal Processing*, 48(3):749–767, March 2000.
- [17] Elizabeth Norton. Data compression on the symmetric group. Undergraduate Thesis, Harvey Mudd College, 2002.
- [18] Daniel N. Rockmore. Fast fourier transforms for wreath products. *Journal of Applied and Computational Harmonic Analysis*, 2(3):279–292, July 1995.
- [19] Daniel N. Rockmore. Recent progress and applications in group ffts. In *Proceedings of the Conference on Computational Noncommutative Algebra and Applications*. NATO Advanced Study Institute, July 2003.
- [20] R. Tolimieri and M. An. Group filters and image processing. In *Proceedings of the Conference on Computational Noncommutative Algebra and Applications*. NATO Advanced Study Institute, July 2003.
- [21] James S. Walker. Fourier analysis and wavelet analysis. *Notices of the American Mathematical Society*, 44(6):658–670, June/July 1997.