MMAT 5390: Mathematical Imaging

Chapter 2: Image Decomposition

One useful technique in image processing is to decompose an image into linear combination of elementary images. Each elementary image has some specific properties. By carefully choosing appropriate elementary images to decompose an image, a lot of imaging tasks can be done. For example, if elementary images have different frequencies, we can easily smooth out or denoise an image by removing the high-frequency components. Also, by truncating the less important components in the image decomposition, we can compress an image to reduce the image size.

In this chapter, we will discuss some common image decomposition methods using some popular linear image transformations. The applications of these decomposition methods for image compression will be demonstrated.

1 Basic idea of image decomposition

In image processing, it is often desirable to decompose an image $f \in \mathcal{I}$ into a linear combination of elementary images. It turns out a separable linear image transformation can naturally give rise to an image decomposition.

Suppose an image transformation $\mathcal{O}$ is defined as $\mathcal{O}(f) = AfB$. Let g be the transformed image. Then, $g = AfB$ and $f = A^{-1}gB^{-1}$.

Write:

$$A^{-1} \equiv \begin{pmatrix} | & | & & | \\ \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_N \\ | & | & & | \end{pmatrix} \text{ and } B^{-1} \equiv \begin{pmatrix} — & \vec{v}_1^T & — \\ — & \vec{v}_2^T & — \\ & \vdots & \\ — & \vec{v}_N^T & — \end{pmatrix}.$$

We can easily check that:

$$f = \sum_{i=1}^{N}\sum_{j=1}^{N} g_{ij}\vec{u}_i\vec{v}_j^T,$$

which is a linear combination of images $\vec{u}_i\vec{v}_j^T$.

Definition 1.1. Each $\vec{u}_i\vec{v}_j^T$ is called an **elementary image** for all i, j. Also $\vec{u}_i\vec{v}_j^T$ is called the **outer product** of $\vec{u}_i$ and $\vec{v}_j$.

Intuitively, image decomposition aims to decompose an image into a linear combination of a basis, which are called the elementary images. Each elementary image captures some important information. Of course, the decomposition depends on the elementary images, which depend on the choices of A and B. The following crucial question must be addressed.

Question: How do we choose A and B?
Answer: We choose A and B such that:

1. Transformed image requires less storage (g_{ij} contains many zeros);
2. By truncating some terms $g_{ij}\vec{u}_i\vec{v}_j^T$ (such as high-frequency terms), we can obtain better (such as smooth) image;

3. A^{-1} and B^{-1} can be easily computed. For example, A and B are unitary. Recall that a matrix U is <u>unitary</u> if $UU^H = I$ where U^H = conjugate transpose. That is,

$$U^H = \begin{cases} (\overline{U})^T, & \text{if } U \text{ complex} \\ U^T, & \text{if } U \text{ real} \end{cases}.$$

Hence, the inverse of an unitary matrix can be easily obtained.

2 Singular Value Decomposition (SVD)

We will first describe a popular method in linear algebra, called the singular value decomposition, for image decomposition. It can be effectively applied for image compression.

Definition 2.1. For any $g \in M_{m\times n}$, the **singular value decomposition (SVD)** of g is a matrix factorization given by

$$g = U\Sigma V^T$$

with $U \in M_{m\times m}$ and $V \in M_{n\times n}$ both orthogonal, and $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix ($\Sigma_{ij} = 0$ if $i \neq j$) with diagonal elements $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ with $r \leq \min(m, n)$. The diagonal elements are called the **singular values** of g.
[Recall that A is orthogonal if $A^T A = AA^T = I$.]

Theorem 2.2. *The rank of g is given by the number of nonzero singular values.*

Proof. Since both U and V are full rank, rank(g) = rank(Σ) which is the number of nonzero singular values. □

Theorem 2.3. *We have the following relationships between the SVD of g and its fundamental subspaces.*

$$\begin{aligned} range(g) &= span(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_r), \\ null(g) &= span(\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \cdots, \mathbf{v}_N). \end{aligned}$$

Proof. Exercise.

Theorem 2.4. *Every $m \times n$ image g has a singular value decomposition.*

Proof. We will consider the case when $m \leq n$. The proof for $m > n$ is similar. To prove the theorem, let us first recall the following theorem in linear algebra.

Theorem. *Let $B \in M_{n\times n}$ be a real symmetric matrix. Then, there exist n orthonormal eigenvectors $\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_n$ such that*

$$B = \begin{bmatrix} | & & | \\ \vec{v}_1 & \cdots & \vec{v}_n \\ | & & | \end{bmatrix} \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \begin{bmatrix} — & \vec{v}_1^T & — \\ — & \vec{v}_2^T & — \\ & \vdots & \\ — & \vec{v}_n^T & — \end{bmatrix}.$$

Now, note that $gg^T \in M_{m\times m}$ and $g^T g \in M_{n\times n}$ are symmetric. Thus, there exist n pairwise orthonormal eigenvectors $\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_n$ of $g^T g$.

Let $\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_r$ be eigenvectors of $g^T g$ with non-zero eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_r$. Observe that $gg^T g\vec{v}_i = g\lambda_i\vec{v}_i = \lambda_i g\vec{v}_i$. Therefore, $g\vec{v}_i$ is an eigenvector of gg^T with eigenvalue λ_i.

Let $\sigma_i = \sqrt{\lambda_i}$ (why $\lambda_i > 0$?), then $\|g\vec{v}_i\| = \sigma_i$ because

$$\|g\vec{v}_i\|^2 \equiv (g\vec{v}_i)^T(g\vec{v}_i) = \vec{v}_i^T g^T g\vec{v}_i = \lambda_i\vec{v}_i^T\vec{v}_i = \lambda_i.$$

Now, define $\vec{u}_i = \dfrac{g\vec{v}_i}{\sigma_i}$, then $\|\vec{u}_i\| = 1$. Also, $\vec{u}_i$ are orthonormal (for $i = 1, 2, \ldots, r$) because

$$\vec{u}_i^T \vec{u}_j = \frac{(g\vec{v}_i)^T}{\sigma_i} \frac{(g\vec{v}_j)}{\sigma_j} = \frac{\vec{v}_i^T g^T g \vec{v}_j}{\sigma_i \sigma_j} = \frac{\lambda_j \vec{v}_i^T \vec{v}_j}{\sigma_i \sigma_j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}.$$

Besides, $\vec{u}_i^T g \vec{v}_j = \dfrac{\lambda_j \vec{v}_i^T \vec{v}_j}{\sigma_i} = \begin{cases} \sigma_i & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$.

In matrix form,

$$\begin{bmatrix} — & \vec{u}_1^T & — \\ — & \vec{u}_2^T & — \\ & \vdots & \\ — & \vec{u}_r^T & — \end{bmatrix} g \begin{bmatrix} | & & | \\ \vec{v}_1 & \cdots & \vec{v}_r \\ | & & | \end{bmatrix} = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{pmatrix}.$$

We extend $\{\vec{u}_1, \cdots, \vec{u}_r\}$ to an orthonormal basis $\{\vec{u}_1, \cdots, \vec{u}_r, \cdots, \vec{u}_m\}$.
Similarly, $\{\vec{v}_1, \cdots, \vec{v}_r\}$ can be extended to $\{\vec{v}_1, \cdots, \vec{v}_r, \cdots, \vec{v}_n\}$.

Then, we get

$$\underbrace{\begin{bmatrix} — & \vec{u}_1^T & — \\ — & \vec{u}_2^T & — \\ & \vdots & \\ — & \vec{u}_r^T & — \\ & \vdots & \\ — & \vec{u}_m^T & — \end{bmatrix}}_{U^T} g \underbrace{\begin{bmatrix} | & & | & & | \\ \vec{v}_1 & \cdots & \vec{v}_r & \cdots & \vec{v}_n \\ | & & | & & | \end{bmatrix}}_{V} = \underbrace{\begin{pmatrix} \sigma_1 & & & & & & \\ & \sigma_2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_r & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{pmatrix}}_{\Lambda^{1/2}}.$$

[Here, we use the fact that $g\vec{v}_j = \mathbf{0}$ for $j > r$ because $\|g\vec{v}_j\|^2 = \lambda_j = 0$ for $j > r$.]

Then, $U^T U = U U^T = I$, $V^T V = V V^T = I$ and $g = U \Lambda^{1/2} V^T$, where

$$\Lambda = \begin{pmatrix} \lambda_1 & & & & & & \\ & \lambda_2 & & & & & \\ & & \ddots & & & & \\ & & & \lambda_r & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{pmatrix}.$$

$\square$

What are inside U and V ($g = U\Lambda^{1/2} V^T$)?

Note that $gg^T = U\Lambda^{1/2} \underbrace{V^T V}_{I} \Lambda^{1/2} U^T = U \Lambda U^T$.
$\therefore U$ consists of eigenvectors of gg^T.
Similarly, $g^T g = V \Lambda V^T$.
$\therefore V$ consists of eigenvectors of $g^T g$.

Remark.

1. *Note that $gg^T \vec{u} = \lambda \vec{u}$ (where $\lambda =$ eigenvalue, and $\vec{u} =$ eigenvector), then:*

$$\lambda = \frac{\vec{u}^T g g^T \vec{u}}{\vec{u}^T \vec{u}} = \frac{(g^T \vec{u})^T (g^T \vec{u})}{\vec{u}^T \vec{u}} \geq 0;$$

2. $g = U\Lambda^{1/2}V^T = \sum\limits_{i=1}^{r} \lambda_i^{1/2}\vec{u}_i\vec{v}_i^T$.
$\vec{u}_i\vec{v}_i^T$ *are called the* **eigenimages** *of g under SVD.*

For an $N \times N$ image, the required storage after SVD is: $(2N+1)\times r$. Hence, it saves storage when r is small.

Definition 2.5. For any k with $0 \le k \le r$, we define

$$g_k = \sum_{j=1}^{k} \sigma_j \vec{u}_j \vec{v}_j^T$$

where g_k is called a **rank-k approximation** of g.

This low rank matrix approximation can be applied to image compression. For any $M \times N$ image $g \in \mathbb{R}^{M\times N}$, one is required to allocate MN intensity levels in total. The rank-ν approximation A_ν, however, needs only store ν singular values and singular vectors $\vec{u}_j$ and $\vec{v}_j$, which leads to $\nu(1+M+N)$ numbers. In the case when $M = N$ for simplicity, there is a reduction of storage if

$$\nu < \frac{N^2}{2N+1}.$$

We can remove the i-th term of which λ_i is small to further reduce the storage.

Error of the approximation by SVD

Definition 2.6. The **Frobenius norm (F-norm)** given by

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2},$$

where a_{ij} is the i-th row, j-th column entry of A.
Let $\mathbf{a_i}$ be the j-th column of A. We have

$$\|A\|_F = \sqrt{\sum_{j=1}^{n} \|a_j\|_2^2} = \sqrt{\mathrm{tr}(A^*A)} = \sqrt{\mathrm{tr}(AA^*)},$$

where $\mathrm{tr}(\cdot)$ is the trace of the matrix in the argument.

Theorem 2.7. *The F-norm of a matrix is invariant under multiplication by unitary matrices, i.e. for any $A \in \mathbb{R}^{m\times n}$, unitary $U \in \mathbb{R}^{m\times m}$, we have $\|UA\|_F = \|A\|_F$.*

Proof. The proof is very simple, which can be explained in one line:

$$\|UA\|_F = \sqrt{\mathrm{tr}\big((UA)^T(UA)\big)} = \sqrt{\mathrm{tr}(A^TU^TUA)} = \sqrt{\mathrm{tr}(A^TA)} = \|A\|_F.$$

□

Theorem 2.8. *Let $f = \sum\limits_{j=1}^{r} \sigma_j\vec{u}_j\vec{v}_j^T$ be the SVD of an $M \times N$ image f. For any k with $k < r$ and $f_k = \sum\limits_{j=1}^{k} \sigma_j\vec{u}_j\vec{v}_j^T$, we have*

$$\|f - f_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2.$$

Proof. Let $f = \sum\limits_{i=1}^{r} \lambda_i^{1/2}\vec{u}_i\vec{v}_i^T$.

Approximate f by f_k with $k < r$ where $f_k = \sum\limits_{i=1}^{k} \lambda_i^{1/2}\vec{u}_i\vec{v}_i^T$.

Define the error of the approximation by $D \equiv f - f_k = \sum_{i=k+1}^{r} \lambda_i^{1/2} \vec{u}_i \vec{v}_i^T \in M_{M\times N}$.

The m-th row, n-th column entry of D is

$$D_{mn} = \sum_{i=k+1}^{r} \lambda_i^{1/2} u_{im} v_{in}$$

where $\vec{u}_i = \begin{pmatrix} u_{i1} \\ \vdots \\ u_{iM} \end{pmatrix}$, $\vec{v}_i = \begin{pmatrix} v_{i1} \\ \vdots \\ v_{iN} \end{pmatrix}$. Then,

$$D_{mn}^2 = \left(\sum_{i=k+1}^{r} \lambda_i^{1/2} u_{im} v_{in} \right)^2 = \sum_{i=k+1}^{r} \lambda_i u_{im}^2 v_{in}^2 + \sum_{i=k+1}^{r} \sum_{\substack{j=k+1 \\ j\neq i}}^{r} \lambda_i^{1/2} \lambda_j^{1/2} u_{im} v_{in} u_{jm} v_{jn}.$$

Thus,

$$\begin{aligned}
\|D\|_F^2 &= \sum_m \sum_n D_{mn}^2 \\
&= \sum_m \sum_n \left(\sum_{i=k+1}^{r} \lambda_i u_{im}^2 v_{in}^2 + \sum_{i=k+1}^{r} \sum_{\substack{j=k+1 \\ j\neq i}}^{r} \lambda_i^{1/2} \lambda_j^{1/2} u_{im} v_{in} u_{jm} v_{jn} \right) \\
&= \sum_{i=k+1}^{r} \left(\lambda_i \cancelto{1}{\sum_m u_{im}^2} \cancelto{1}{\sum_n v_{in}^2} + \sum_{\substack{j=k+1 \\ j\neq i}}^{r} \lambda_i^{1/2} \lambda_j^{1/2} \cancelto{0}{\sum_m u_{im} u_{jm}} \cancelto{0}{\sum_m v_{in} v_{jn}} \right) \\
&= \sum_{i=k+1}^{r} \lambda_i = \sum_{i=k+1}^{r} \sigma_i^2.
\end{aligned}$$

Therefore, **Sum of square error of the approximation = Sum of omitted eigenvalues**. □

Remark.

- *To approximate an image using SVD, arrange the eigenvalues λ_i in decreasing order, and remove the last few terms in $\sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^T$.*
- *This is the optimal approximation using k terms in the Frobenius norm.*

Example 2.9. Let

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 2 \\ 2 & 1 \end{pmatrix}.$$

We have

$$A^T A = \begin{pmatrix} 9 & 8 \\ 8 & 9 \end{pmatrix}.$$

Now, eig(A^*A) are 17 and 1, and so $\sigma_1 = \sqrt{17}$, $\sigma_2 = 1$ and

$$\Sigma = \begin{pmatrix} \sqrt{17} & 0 \\ 0 & 1 \end{pmatrix}.$$

Moreover,

$$\vec{v}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \ \vec{v}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

This gives

$$V = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}.$$

Since

$$\sigma_1\vec{u}_1 = A\vec{v}_1,$$

we have

$$\vec{u}_1 = \frac{1}{\sqrt{17}}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 2\\ 2 & 2\\ 2 & 1\end{pmatrix}\begin{pmatrix}1\\ 1\end{pmatrix} = \frac{1}{\sqrt{34}}\begin{pmatrix}3\\ 4\\ 3\end{pmatrix}.$$

Similarly, we have

$$\vec{u}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 2\\ 2 & 2\\ 2 & 1\end{pmatrix}\begin{pmatrix}1\\ -1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\ 0\\ 1\end{pmatrix}.$$

The matrix U is, therefore, given by

$$U = \begin{pmatrix}\frac{3}{\sqrt{34}} & \frac{-1}{\sqrt{2}} & | \\ \frac{4}{\sqrt{34}} & 0 & \vec{u}_3 \\ \frac{3}{\sqrt{34}} & \frac{1}{\sqrt{2}} & | \end{pmatrix}$$

for some vector $\mathbf{u}_3$ orthonormal to both $\mathbf{u}_1$ and $\mathbf{u}_2$. One possibility is

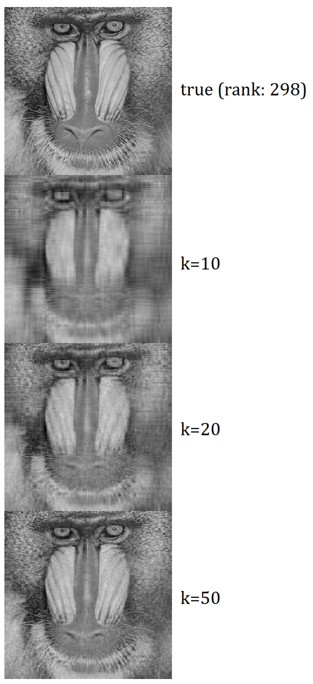
$$\mathbf{u}_3 = \frac{1}{\sqrt{17}}\begin{pmatrix}2\\ -3\\ 2\end{pmatrix}.$$

Finally, the SVD of A is given by

$$\begin{pmatrix}1 & 2\\ 2 & 2\\ 2 & 1\end{pmatrix} = \begin{pmatrix}\frac{3}{\sqrt{34}} & \frac{-1}{\sqrt{2}} & \frac{2}{\sqrt{17}}\\ \frac{4}{\sqrt{34}} & 0 & \frac{-3}{\sqrt{17}}\\ \frac{3}{\sqrt{34}} & \frac{1}{\sqrt{2}} & \frac{2}{\sqrt{17}}\end{pmatrix}\begin{pmatrix}\sqrt{17} & 0\\ 0 & 1\\ 0 & 0\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}}\end{pmatrix}$$

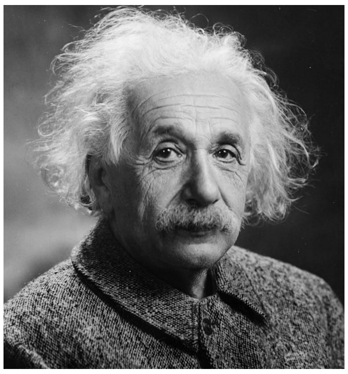
Application of SVD for image compression

SVD can be applied for image compression, by removing terms associated to small singular values. We will show some examples of image compression using SVD. Please refer to Lecture 2 Powerpoint for more illustration.

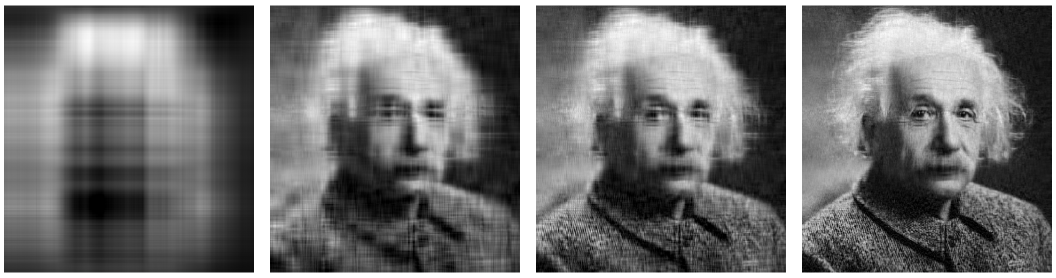


In the above figure, the left shows the rank-k approximation of a baboon image with $k = 10, 20$ and 50 respectively. The original rank of the image is 298. Note that the rank-10 approximation of the image can give a very good approximation of the original image. The right shows the image compression of a fingerprint image, using a rank-40 approximation. SVD is one of the popular methods for image compression.

Next, we test the image compression using SVD on the 'Einstein' image:



The original rank of the 'Einstein' image is 202. Using SVD, we compute some rank-k approximations of the image, which are given below:



$k = 2$ $k = 12$ $k = 22$ $k = 52$

The above figure shows the rank-k approximations of the 'Einstein' image with $k = 2, 12, 22$ and 52 respectively.

3 Haar and Walsh transforms

Note: Walsh transform will not be covered in this course. It is in the lecture note for the sake of completeness.

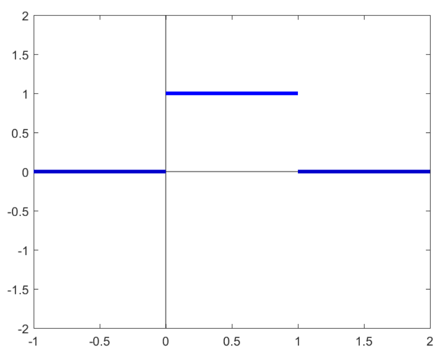
Definition 3.1 (Haar functions)**.** The Haar functions are defined as follows:

$$H_0(t) \equiv \begin{cases} 1 & \text{if } 0 \le t < 1 \\ 0 & \text{elsewhere} \end{cases},$$

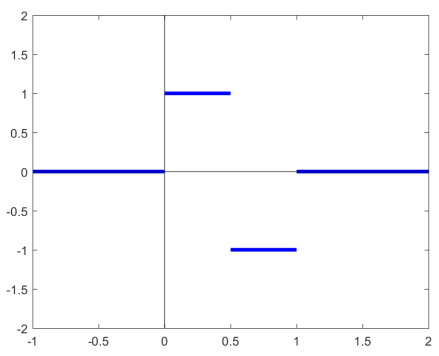
$$H_1(t) \equiv \begin{cases} 1 & \text{if } 0 \le t < 1/2 \\ -1 & \text{if } 1/2 \le t < 1 \\ 0 & \text{elsewhere} \end{cases},$$

$$H_{2^p+n}(t) \equiv \begin{cases} \sqrt{2}^p & \text{if } \frac{n}{2^p} \le t < \frac{n+0.5}{2^p} \\ -\sqrt{2}^p & \text{if } \frac{n+0.5}{2^p} \le t < \frac{n+1}{2^p} \\ 0 & \text{elsewhere} \end{cases}$$

where $p = 1, 2, \cdots; n = 0, 1, 2, \cdots, 2^p - 1$.
($H_{2^p+n}(t)$ is compactly supported by a smaller region if p is bigger)



H_0



H_1

Definition 3.2 (Walsh functions)**.** The Walsh functions are defined recursively as follows:

$$W_{2j+q}(t) \equiv (-1)^{\lfloor \frac{j}{2} \rfloor + q}\{W_j(2t) + (-1)^{j+q} W_j(2t-1)\},$$

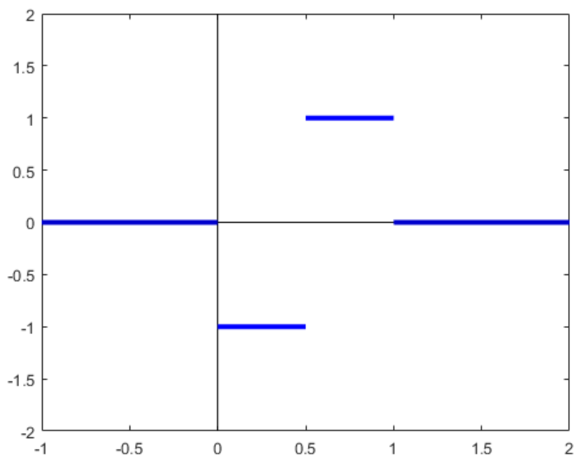
where $\lfloor \frac{j}{2} \rfloor$ = largest integer smaller or equal to $\frac{j}{2}$; $q = 0$ or 1; $j = 0, 1, 2, \cdots$ and

$$W_0(t) \equiv \begin{cases} 1 & \text{if } 0 \le t < 1 \\ 0 & \text{elsewhere} \end{cases}.$$

Example 3.3. Compute $W_1(t)$.
Put $j = 0, q = 1$. Then:

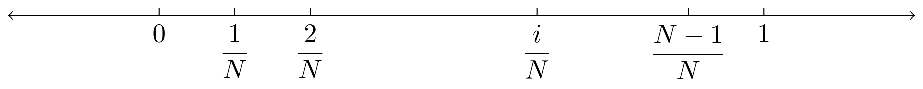
$$\begin{aligned} W_1(t) &\equiv (-1)^{\lfloor 0 \rfloor + 1}\{W_0(2t) + (-1)^{0+1} W_0(2t-1)\} \\ &= -W_0(2t) + W_0(2t-1). \end{aligned}$$

Check that for $0 \le t < \frac{1}{2}$, $W_0(2t) = 1$, $W_0(2t-1) = 0 \implies W_1(t) = -1$.
For $\frac{1}{2} \le t < 1$, $W_0(2t) = 0$, $W_0(2t-1) = 1 \implies W_1(t) = 1$.



Definition 3.4 (Discrete Haar Transform)**.**

The Haar Transform of an $N \times N$ image is performed as follows. Divide [0,1] into N partitions. That is,



Let $H(k,i) \equiv H_k\left(\dfrac{i}{N}\right)$ where $k, i = 0, 1, 2, \cdots, N-1$.
We obtain the Haar Transform matrix:

$$\tilde{H} \equiv \frac{1}{\sqrt{N}} H \quad \text{where } H \equiv (H(k,i))_{0 \le k,i \le N-1}.$$

The Haar Transform of $f \in M_{n\times n}$ is defined as:

$$g = \tilde{H} f \tilde{H}^T.$$

Definition 3.5 (Discrete Walsh Transform)**.** The Walsh Transform of an $N \times N$ image is defined similarly as Haar Transform.
Define $W(k,i) \equiv W_k\left(\dfrac{i}{N}\right)$ where $k, i = 0, 1, 2, \cdots, N-1$.
Then, the Walsh Transform matrix is:

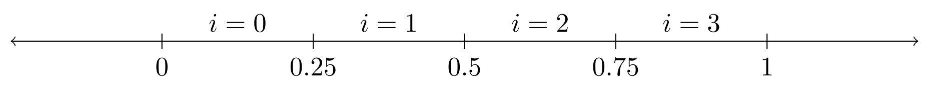
$$\tilde{W} \equiv \frac{1}{\sqrt{N}} H \quad \text{where } H \equiv (W(k,i))_{0 \le k,i \le N-1}.$$

The Walsh Transform of $f \in M_{n\times n}$ is defined as:

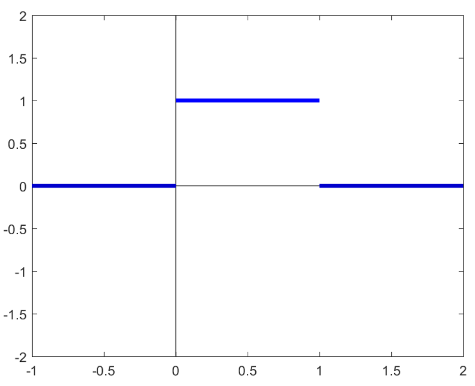
$$g = \tilde{W} f \tilde{W}^T.$$

Example 3.6. Compute the Haar Transform matrix for a 4×4 image.

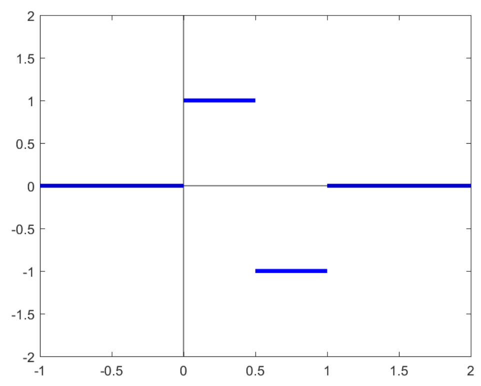
Solution. Divide [0,1] into 4 portions:



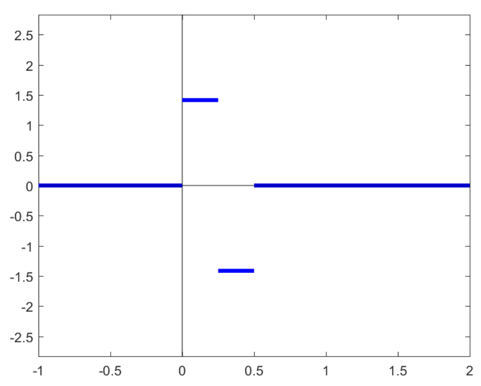
Check that:



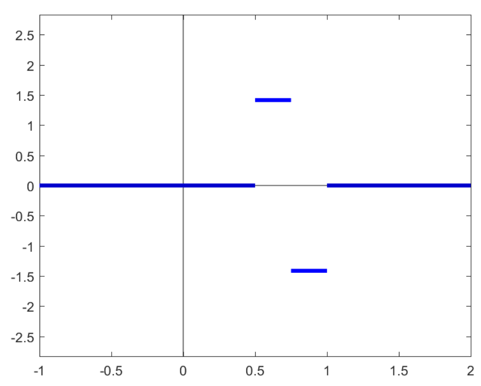
H_0



H_1



H_2



H_3

We get that

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{pmatrix} \text{ and } \tilde{H} = \frac{1}{\sqrt{4}} H = \frac{1}{2} H.$$

Easy to check that $\tilde{H}^T \tilde{H} = I$.

Example 3.7. Compute the Haar Transform of

$$f = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$

Solution.

$$g = \tilde{H} f \tilde{H}^T = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}.$$

Example 3.8. Suppose g in Example 3.7 is changed to:

$$g = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Reconstruct the original image.

Solution.

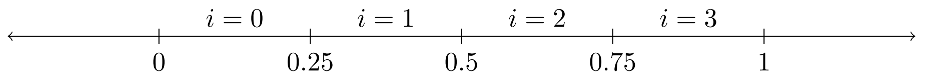
$$f = \tilde{H}^T f \tilde{H} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0.5 & 0.5 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Remark.

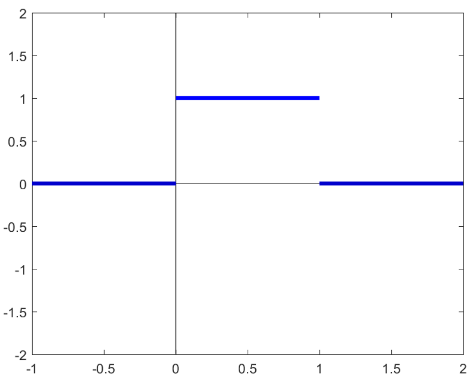
- *Haar Transform usually produces coefficient matrix with more zeroes (compression);*
- *Errors in the coefficient matrix cause localized errors in the reconstructed image (Assign detail of accuracy in compression).*

Example 3.9. Compute the Walsh Transform matrix for a 4×4 image.

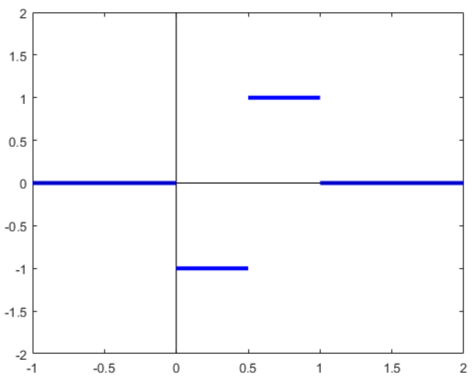
Solution. Again, divide [0,1] into 4 portions:



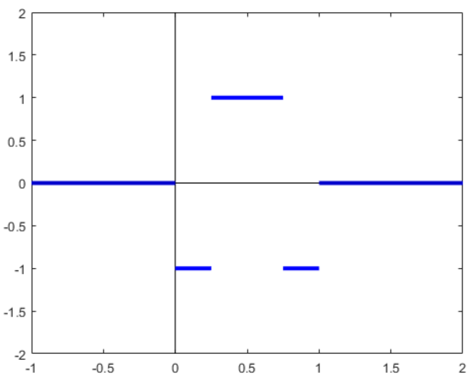
Check that:



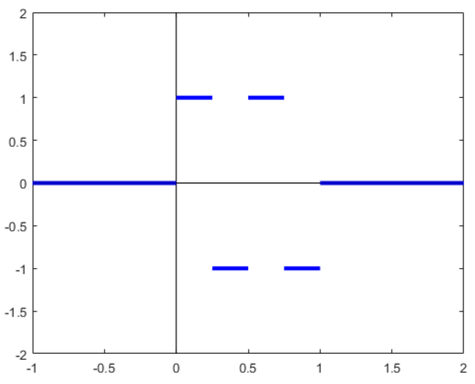
W_0



W_1



W_2



W_3

So,

$$W = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \quad \text{and} \quad \tilde{W} = \frac{1}{\sqrt{4}} W = \frac{1}{2} W.$$

Example 3.10. Compute the Walsh Transform of

$$f = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$

Solution. $g = \tilde{W} f \tilde{W}^T = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$

(Many zeroes $\implies$ compression)

Remark.

- *The idea of Haar / Walsh Transform is to transform an image to a "transformed image" with many more zeroes.*
- *The coefficient in the "transformed" image tells us information of frequency of image intensity changes.*

Another way to define Walsh function

Definition 3.11 (Rademacher functions)**.**

A **Rademacher function** of order n ($n \neq 0$) is defined as:

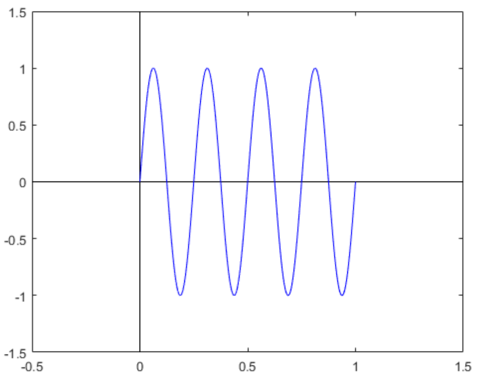
$$R_n(t) \equiv \text{sign}[\sin(2^n \pi t)] \quad \text{for } 0 \le t \le 1$$

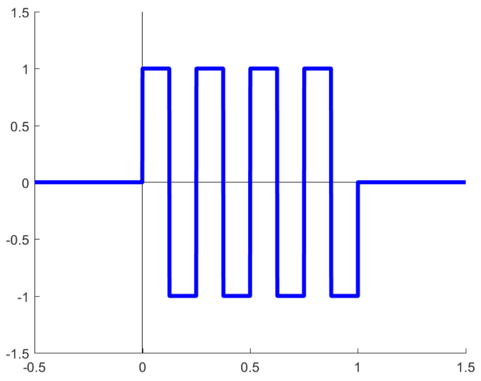
(where $\text{sign}(x) = 1$ if $x > 0$, $\text{sign}(x) = -1$ if $x < 0$ and $\text{sign}(x) = 0$ if $x = 0$). For $n = 0$, $R_0(t) \equiv 1$ for $0 \le t \le 1$.
Let $N = b_{m+1}2^m + b_m 2^{m-1} + \cdots + b_1 2^0$. Then, the R-Walsh function $\tilde{W}_N$ is given by:

$$\tilde{W}_N = \prod_{\substack{i=1 \\ b_i \neq 0}}^{m+1} R_i(t)$$

(where the values at the jumps are defined such that the function is continuous from the right).

Example 3.12. Compute R-Walsh functions $\tilde{W}_3$ and $\tilde{W}_4$ using Rademacher functions.
Consider $\sin(8\pi t)$: Therefore, $R_3(t) =$





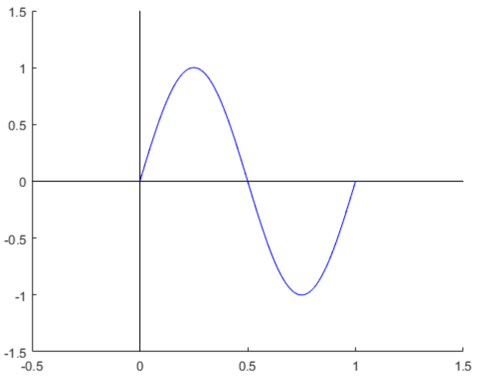
As $4 = \underbrace{1}_{b_3} \cdot 2^2 + \underbrace{0}_{b_2} \cdot 2^1 + \underbrace{0}_{b_1} \cdot 2^0$, we have

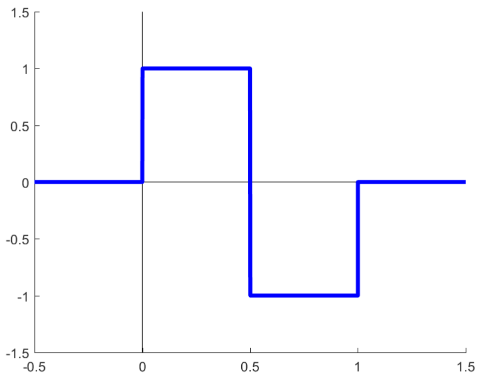
$$\tilde{W}_4 = \prod_{\substack{i=1 \\ b_i \neq 0}}^{3} R_i(t) = R_3(t)$$

($W_{2j+q}(t) \equiv (-1)^{\lfloor j/2 \rfloor + q}\{W_j(2t) + (-1)^{j+q} W_j(2t-1)\}$).
For $W_3(t)$: As $3 = \underbrace{1}_{b_2} \cdot 2^1 + \underbrace{1}_{b_1} \cdot 2^0$, we have

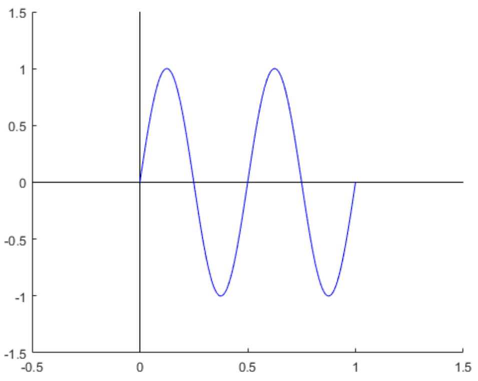
$$\tilde{W}_3(t) = \prod_{\substack{i=1 \\ b_i \neq 0}}^{2} R_i(t) = R_1(t) R_2(t).$$

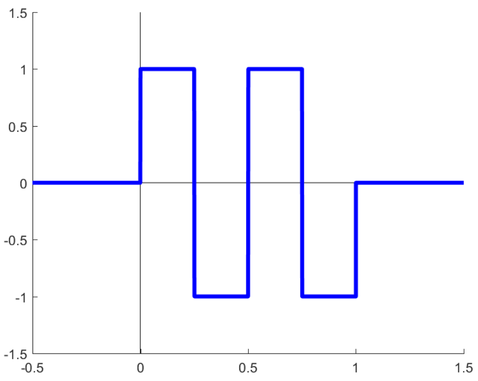
$R_1(t)$:



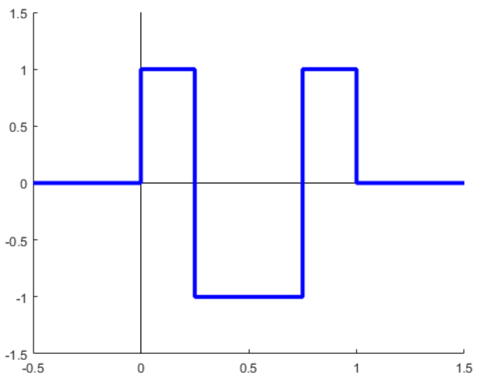


$R_2(t)$:





Therefore, $\tilde{W}_3(t)$:



Relationship between Walsh functions and R-Walsh functions

$W_0(t) = \tilde{W}_0(t)$, $W_1(t) = -\tilde{W}_1(t)$, $W_2(t) = -\tilde{W}_3(t)$, $W_3(t) = \tilde{W}_2(t)$,
$W_4(t) = \tilde{W}_6(t)$, $W_5(t) = -\tilde{W}_7(t)$, $W_6(t) = -\tilde{W}_5(t)$, $W_7(t) = \tilde{W}_4(t)$.

How to determine R-Walsh $\tilde{W}_i(t)$ associated to Walsh $W_j(t)$

Idea: Write j as

$$j = b_{m+1}2^m + b_m 2^{m-1} + \cdots + b_1 2^0.$$

The binary representation of j is:

$$b_{m+1}b_m \cdots b_1.$$

Then, i is given by:

$$c_{m+1}c_m \cdots c_1$$

where

$$c_{m+1} \equiv b_{m+1} \pmod 2, \quad c_k \equiv (b_{k+1} + b_k) \pmod 2.$$

The sign is determined from $W_j(0)$.

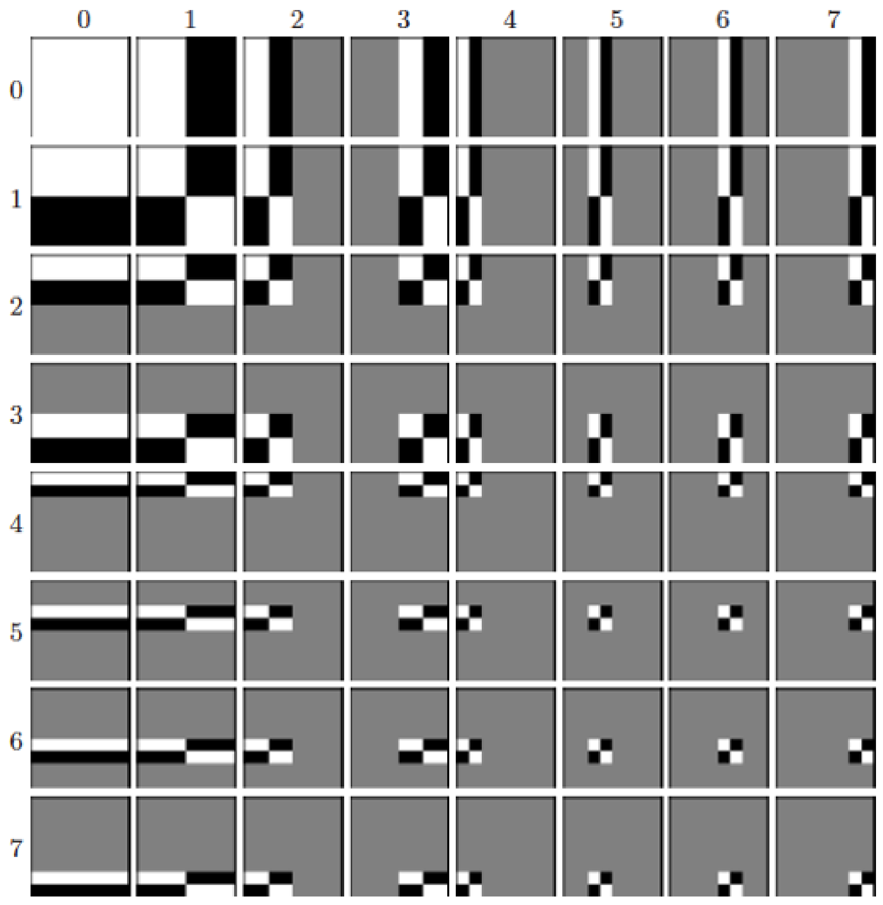
Example 3.13. Consider $W_7(t)$.
Check that $W_7(t) > 0$.
Now, $7 = 2^2 + 2^1 + 2^0$ so the binary representation of 7 is 111.
Therefore, $i = 100$ (binary) $= 4$.
Thus, $W_7(t) = \tilde{W}_4(t)$.

What do the elementary images under the Haar and Walsh Transform look like?

(Please refer to the Lecture 3 and 4 Powerpoint for clearer figures)
Using Haar transform, the 8×8 image f can be written as:

$$f = \tilde{H}^T g \tilde{H} = \sum_{1 \le i,j \le N} g_{ij} I_{ij}^H$$

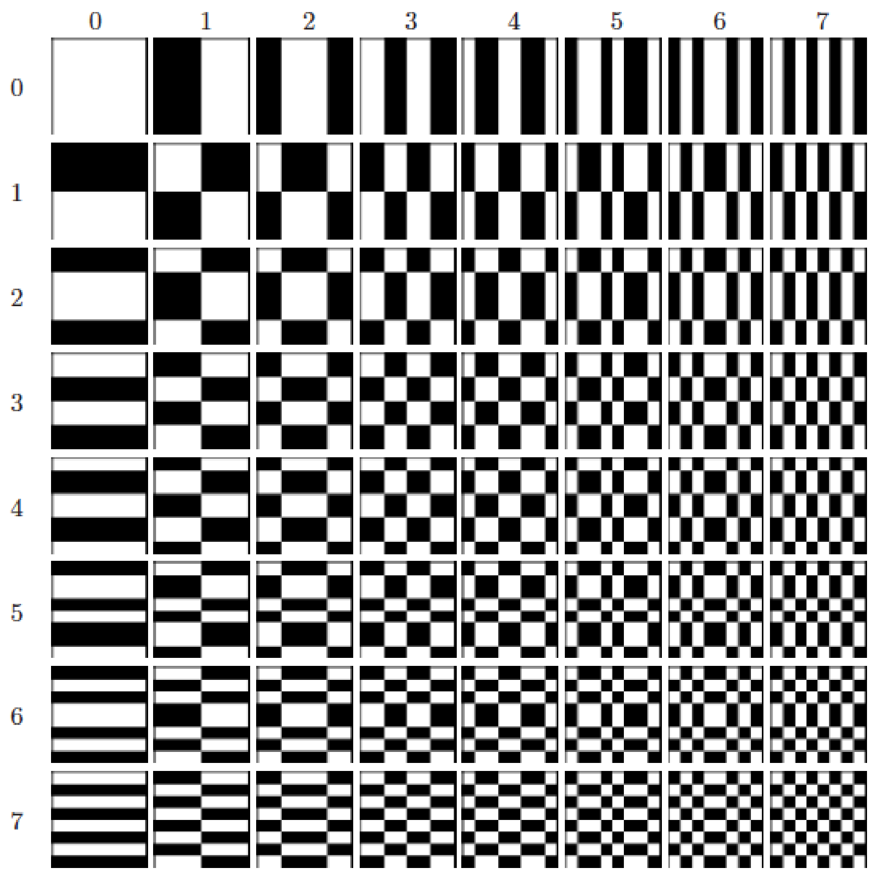
where I_{ij}^H is the elementary image given by taking the outer product of the i-th row and the j-th row of $\tilde{H}$.
The elementary images under the Haar transform (of an 8×8 image) look like the following:



Here, the i-th row j-th column image represents the elementary image I_{ij}^H.
[White = positive; Black = negative; Grey = 0]
Note that some elementary images are locally supported (i.e. non-zero at some locally small region). Therefore, they capture both spatial and frequency information.
Similarly, under the Walsh transform, the 8×8 image f can be written as:

$$f = \tilde{W}^T g \tilde{W} = \sum_{1 \le i,j \le N} g_{ij} I_{ij}^W$$

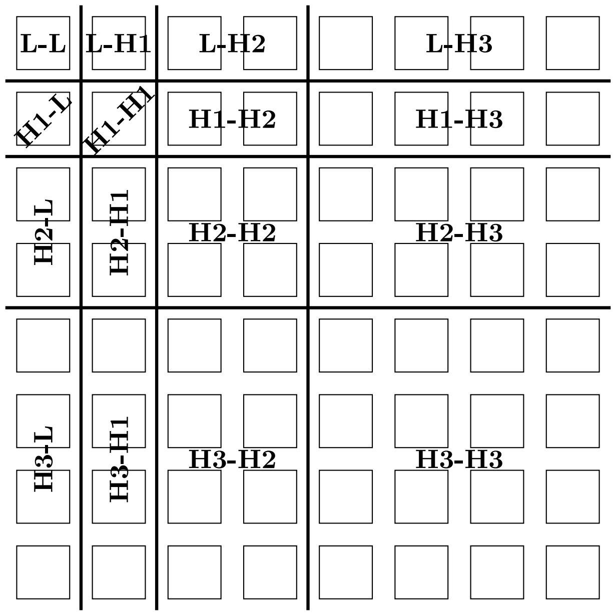
where I_{ij}^W is the elementary image given by taking the outer product of the i-th row and j-th row of W.
The elementary images under the Walsh transform (of an 8×8 image) look like the following:



Here, the i-th row j-th column image represents the elementary image I_{ij}^W.
[White = positive; Black = negative; Grey = 0]

Properties of each elementary image under the Haar Transform

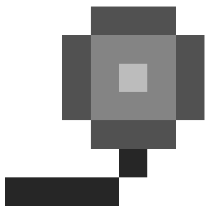
Each elementary image captures different levels of details with different levels of resolution in the horizontal and vertical directions. Please refer to the Lecture 3 and 4 Powerpoint for a clearer figure.



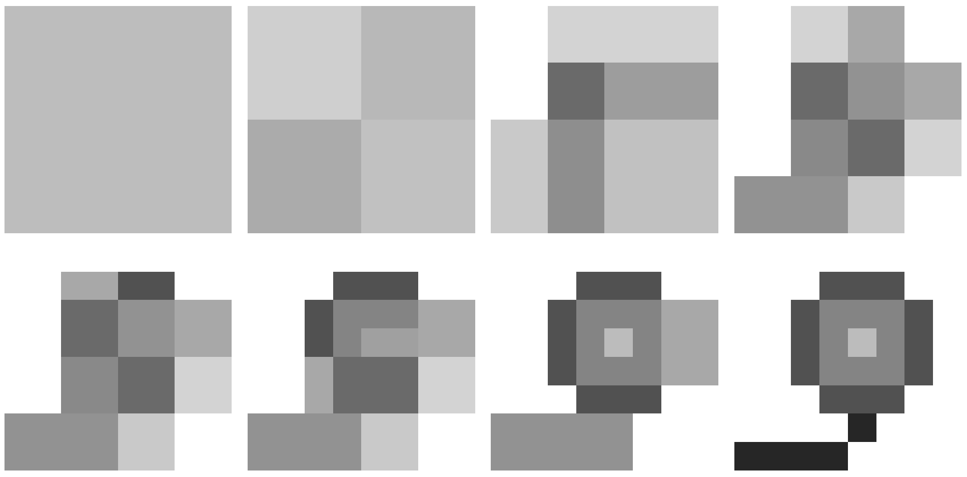
In the above figure, the thick lines divide them into sets of elementary images of the same resolution. Letter L and H are used to indicate low and high resolution, respectively. The numbers next to letter H indicates which level of high resolution. The pairs of letters used indicate which resolution we have along the vertical and horizontal axes. For example, pair L-H2 indicates that the corresponding panels have low resolution along the vertical axis, but second order high resolution along the horizontal axis.

Application of the Haar and Walsh transforms for image compression

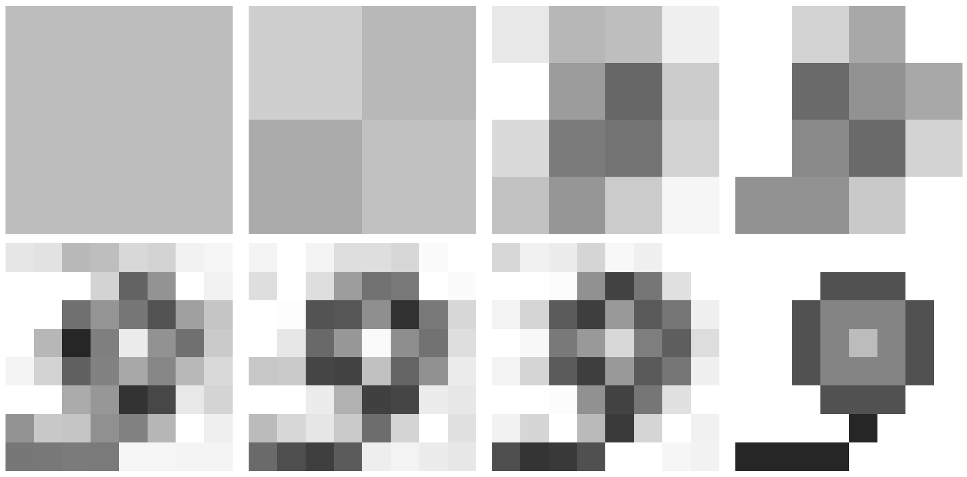
The Haar and Walsh transforms can be used for image compression, by truncating high-frequency terms. Below we will illustrate the idea on a simple image:



The approximations of the image using the Haar transform are shown below. More precisely, the following figure shows reconstructed images when the basis images used are those created from the first one, two, three, ..., eight Haar functions, from top left to bottom right, respectively.



The approximations of the image using the Walsh transform are shown below. Similarly, the following figure shows reconstructed images when the basis images used are those created from the first one, two, three, ..., eight Walsh functions, from top left to bottom right, respectively.



4 Discrete Fourier Transform

Definition 4.1. The 1D **discrete Fourier transform (DFT)** of a function $f(k)$, defined at discrete points $k = 0, 1, \cdots, N-1$, is defined as

$$F(m) = \frac{1}{N}\sum_{k=0}^{N-1} f(k)e^{-2\pi j\frac{mk}{N}} \qquad \begin{pmatrix} j = \sqrt{-1} \\ e^{j\theta} = \cos\theta + j\sin\theta \end{pmatrix}.$$

The 2D **discrete Fourier transform (DFT)** of an $M \times N$ image $g = (g(k,l))_{k,l}$, where $k = 0, 1, \cdots, M-1$ and $l = 0, 1, \cdots, N-1$, is defined as:

$$\hat{g}(m,n) = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} g(k,l)e^{-2\pi j(\frac{km}{M}+\frac{ln}{N})}. \tag{*}$$

<u>What is the inverse of DFT?</u>

Multiply both sides of (*) by $e^{2\pi j(\frac{pm}{M}+\frac{qn}{N})}$ (with p chosen from $0, 1, \cdots, M-1$ and q chosen from $0, 1, \cdots, N-1$) and sum m over 0 to $M-1$, and n over 0 to $N-1$:

$$\begin{aligned}
&\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \hat{g}(m,n)e^{2\pi j(\frac{pm}{M}+\frac{qn}{N})} \\
&= \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} g(k,l)e^{2\pi j(\frac{m(p-k)}{M}+\frac{n(q-l)}{N})} \\
&= \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} g(k,l)\sum_{m=0}^{M-1} e^{2\pi j\frac{m(p-k)}{M}}\sum_{n=0}^{N-1} e^{2\pi j\frac{n(q-l)}{N}}.
\end{aligned} \tag{**}$$

We can show that for $s \in \mathbb{Z}\setminus\{0\}$ and $t \in \mathbb{Z}$,

$$\sum_{m=0}^{s-1} e^{2\pi j\frac{tm}{s}} = s\mathbf{1}_{s\mathbb{Z}}(t) = \begin{cases} s & \text{if } t \in s\mathbb{Z} \\ 0 & \text{otherwise} \end{cases}.$$

Therefore, R.H.S. of (**) becomes:

$$\begin{aligned}
&\frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} g(k,l)\cdot M\mathbf{1}_{M\mathbb{Z}}(p-k)\cdot N\mathbf{1}_{N\mathbb{Z}}(q-l) \\
&= \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} g(k,l)\delta(p-k)\delta(q-l) = g(p,q).
\end{aligned}$$

(since $k, p \in [0, M-1] \implies p-k \in [1-M, M-1]$, whose only intersection with $M\mathbb{Z}$ is $\{0\}$; and $l, q \in \cap[0, N-1] \implies q-l \in [1-N, N-1]$, whose only intersection with $N\mathbb{Z}$ is $\{0\}$)

Inverse discrete Fourier transform:

$$\boxed{g(p,q) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \hat{g}(m,n) e^{2\pi j(\frac{pm}{M}+\frac{qn}{N})}}$$

How is DFT in matrix form?

Recall:

$$g = \begin{pmatrix} | & | & & | \\ \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n \\ | & | & & | \end{pmatrix} f \begin{pmatrix} — & \vec{v}_1^T & — \\ — & \vec{v}_2^T & — \\ & \vdots & \\ — & \vec{v}_n^T & — \end{pmatrix} = UfV^T \text{ if and only if}$$

$$g = \sum_{i=1}^{M}\sum_{j=1}^{N} f_{ij}\vec{u}_i\vec{v}_j^T$$

We need to express DFT in terms of matrix multiplication.
For an $N \times N$ image g, the DFT of g is given by:

$$\hat{g}(m,n) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} g(k,l)e^{-2\pi j\frac{km+ln}{N}}.$$

Let $U_{x\alpha} = \dfrac{1}{N}e^{-2\pi j\frac{x\alpha}{N}}$ where $0 \le x, \alpha \le N-1$, and $U = (U_{x\alpha})_{0\le x,\alpha\le N-1} \in M_{N\times N}(\mathbb{C})$. Then, U is symmetric and

$$\hat{g} = UgU.$$

Question: Can you write the DFT of an $M \times N$ image in matrix form?

Example 4.2. Find the DFT of the following 4×4 image

$$g = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Solution. The matrix U is given by:

$$U = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix}.$$

$$\therefore \text{DFT of } g = \hat{g} = UgU = \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Property of U:

Note that given $N \in \mathbb{N}\setminus\{0\}$ and $x_1, x_2 \in \mathbb{Z}\cap[0, N-1]$,

$$\begin{aligned} \frac{1}{N^2}\sum_{\alpha=0}^{N-1} e^{-2\pi j\frac{x_1\alpha}{N}}e^{2\pi j\frac{x_2\alpha}{N}} &= \frac{1}{N^2}\sum_{\alpha=0}^{N-1} e^{2\pi j\frac{(x_2-x_1)\alpha}{N}} \\ &= \frac{1}{N^2}N\delta(x_2-x_1). \end{aligned}$$

Therefore, rows of U are mutually orthogonal but not orthonormal.
We can conclude that $UU^* = \dfrac{1}{N}I$ (so U is NOT unitary).
If we define $\tilde{U} \equiv \sqrt{N}U$, then $\tilde{U}$ is unitary.

Remark.

- *Sometimes, people like to define DFT as*
  (1D) $\hat{f}(m) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f(k) e^{-2\pi j \frac{mk}{N}}$
  (2D) $\hat{f}(m,n) = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} f(k,l) e^{-2\pi j \frac{mk+nl}{N}}$ *so that unitary matrix* $\tilde{U}$ *may be used.*
- *If* $\tilde{U}$ *is used, we must be careful about the formula we derive (as they usually differ by a scaling factor) (e.g. the inverse DFT).*

Elementary images under DFT decomposition

Given an image g, we can compute its DFT $\hat{g}$. Then:

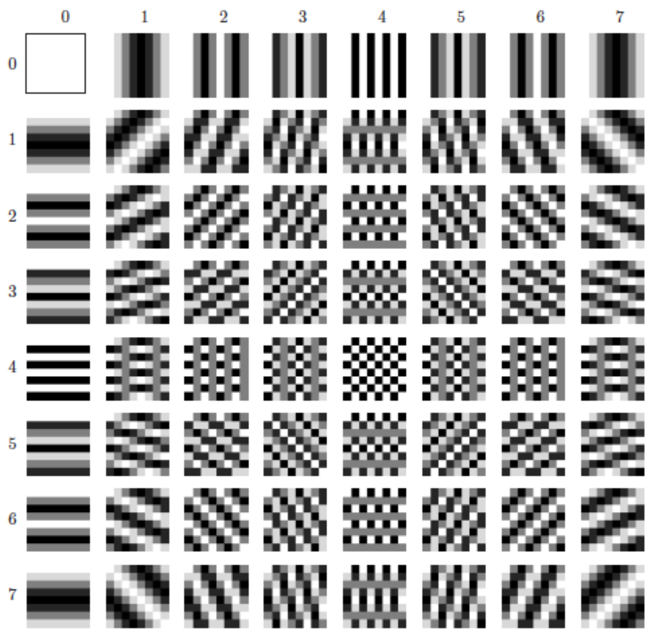
$$\hat{g} = UgU.$$

Hence,

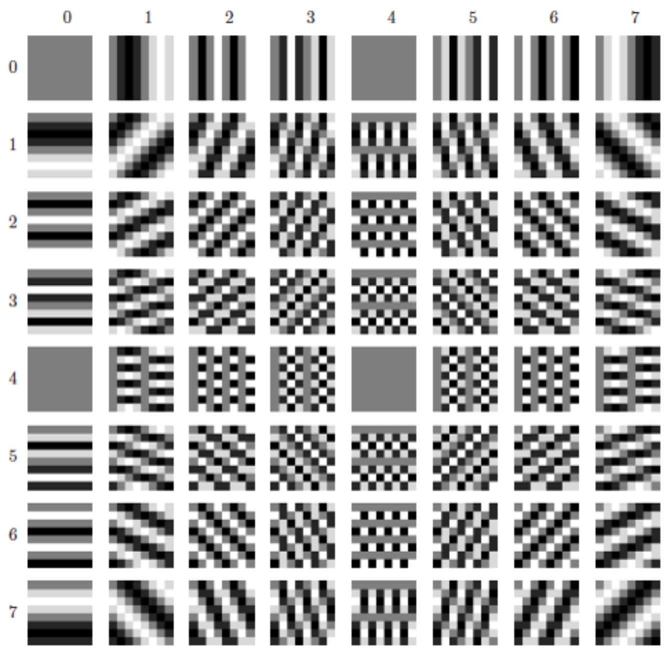
$$g = (NU)^* \hat{g} (NU)^* = \sum_k \sum_l \hat{g}_{kl} \vec{w}_k \vec{w}_l^T$$

where $\vec{w}_k$ is the k-th column of $(NU)^*$. Then $\vec{w}_i \vec{w}_j^T$ are the elementary images of DFT decomposition.
For a visualization of DFT elementary images, please refer to Lecture 4 Powerpoint.

The real part of elementary images under the DFT (of an 8×8 image) look like the following:



The imaginary part of elementary images under the DFT look like the following:



Further properties of DFT

1. DFT of convolution

Let g and w be two $N \times M$ images. Assume that g and w are periodically extended. The convolution of them is

$$v(n,m) = \sum_{n'=0}^{N-1} \sum_{m'=0}^{M-1} g(n-n', m-m') w(n', m').$$

We compute the DFT of v:

$$\begin{aligned}
&\frac{1}{NM} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} v(n,m) e^{-2\pi j(\frac{pn}{N}+\frac{qm}{M})} \\
&= \frac{1}{NM} \sum_{n'=0}^{N-1} \sum_{m'=0}^{M-1} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} g(\underbrace{n-n'}_{n''}, \underbrace{m-m'}_{m''}) w(n',m') e^{-2\pi j(\frac{pn}{N}+\frac{qm}{M})} \\
&= \underbrace{\frac{1}{NM} \sum_{n'=0}^{N-1} \sum_{m'=0}^{M-1} w(n',m') e^{-2\pi j(\frac{pn'}{N}+\frac{qm'}{M})}}_{\hat{w}(p,q)} \underbrace{\sum_{n''=-n'}^{N-1-n'} \sum_{m''=-m'}^{M-1-m'} g(n'',m'') e^{-2\pi j(\frac{pn''}{N}+\frac{qm''}{M})}}_{T}.
\end{aligned}$$

Note that g and w are periodically extended. Hence, we have:

$$g(n-N,m) = g(n,m) \quad \text{and} \quad g(n,m-M) = g(n,m).$$

Then:

$$\begin{aligned}
T \equiv &\sum_{m''=-m'}^{M-1-m'} e^{-2\pi j \frac{qm''}{M}} \sum_{n''=-n'}^{-1} g(n'',m'') e^{-2\pi j \frac{pn''}{N}} \\
&+ \sum_{m''=-m'}^{M-1-m'} e^{-2\pi j \frac{qm''}{M}} \sum_{n''=0}^{N-1-n'} g(n'',m'') e^{-2\pi j \frac{pn''}{N}}.
\end{aligned}$$

Consider

$$\sum_{n''=-n'}^{-1} g(n'',m'')e^{-2\pi j\frac{pn''}{N}}$$

$$\overset{n'''=N+n''}{=} \sum_{n'''=N-n'}^{N-1} \underbrace{g(n'''-N,m'')}_{g(n''',m'')} e^{-2\pi j\frac{pn'''}{N}} \underbrace{e^{2\pi jp}}_{1} \quad (\because p = \text{integer}).$$

(Similarly, we can do the same thing for the index m''.)

Therefore,

$$T = \sum_{m''=0}^{M-1}\sum_{n''=0}^{N-1} g(n'',m'')e^{-2\pi j(\frac{pn''}{N}+\frac{qm''}{M})} = MN\hat{g}(p,q).$$

$$\boxed{\hat{v}(p,q) = MN\hat{g}(p,q)\hat{w}(p,q)}$$

Remark. *Conversely, if* $x(n,m) = g(n,m)\cdot w(n,m)$,
then $\hat{x}(k,l) = \sum_{p=0}^{N-1}\sum_{q=0}^{M-1} \hat{g}(p,q)\hat{w}(k-p,l-q)$ *(convolution of* $\hat{g}$ *and* $\hat{w}$*).*

2. Average value of image v.s. DFT

Average value of image g:

$$\bar{g} = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} g(k,l) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} g(k,l)\underbrace{e^{-2\pi j(0)}}_{1} = \hat{g}(0,0).$$

3. DFT of a rotated image

Let g be an $N\times N$ image.

Consider:

$$\hat{g}(m,n) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} g(k,l)e^{-2\pi j\frac{km+ln}{N}}.$$

Let $k \equiv r\cos\theta$, $l \equiv r\sin\theta$ (Polar coordinates of (k,l)).
Similarly, let $m \equiv w\cos\phi$, $n \equiv w\sin\phi$ (Polar coordinates of (m,n)).
Note that $km+ln = rw(\cos\theta\cos\phi + \sin\theta\sin\phi) = rw\cos(\theta-\phi)$.

Denote $\mathcal{P}(g) = \{(r,\theta) : (r\cos\theta, r\sin\theta) \text{ is a pixel of the image } g\}$, which is called the polar coordinate set of g.
Then:

$$\hat{g}(w,\phi) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} g(r,\theta)e^{-2\pi j\frac{rw\cos(\theta-\phi)}{N}}.$$

Here, we identify $g(r\cos\theta, r\sin\theta)$ with $g(r,\theta)$ and $\hat{g}(w\cos\phi, w\sin\phi)$ with $\hat{g}(w,\phi)$.

Consider the rotated image $\tilde{g}(r,\theta) = g(r,\theta-\theta_0)$. Here, θ is assumed to be defined between θ_0 to $\pi/2+\theta_0$. Hence, one rotates image g counterclockwise by θ_0 to get $\tilde{g}$.

DFT of $\tilde{g}$ can be computed by:

$$\begin{aligned}\hat{\tilde{g}}(w,\phi) &= \frac{1}{N^2}\sum_{(r,\theta)\in\mathcal{P}(\tilde{g})} g(r,\underbrace{\theta-\theta_0}_{\tilde{\theta}})e^{-2\pi j\frac{rw\cos(\theta-\phi)}{N}} \\ &= \frac{1}{N^2}\sum_{(r,\tilde{\theta})\in\mathcal{P}(g)} g(r,\tilde{\theta})e^{-2\pi j\frac{rw\cos(\tilde{\theta}+\theta_0-\phi)}{N}}.\end{aligned}$$

Therefore,

$$\boxed{\hat{\tilde{g}}(w,\phi) = \hat{g}(w,\phi-\theta_0)}$$

Again, ϕ is assumed to be defined between θ_0 to $\pi/2+\theta_0$.

Remark. *DFT of an image rotated by θ_0 = DFT of the original image rotated by θ_0.*

Example 4.3. Let $g = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$, then $\hat{g} = \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$.

Rotate g by 90° clockwise: $\tilde{g} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$.

Note that the indices of $\tilde{g}$ are taken as follows (after the rotation): $-3 \leq l \leq 0$ and $0 \leq k \leq 3$. Then,

DFT of $\tilde{g} = \hat{\tilde{g}} = \begin{pmatrix} 0 & 0 & 0 & \frac{1}{4} \\ 0 & 0 & 0 & -\frac{1}{4} \\ 0 & 0 & 0 & \frac{1}{4} \\ 0 & 0 & 0 & -\frac{1}{4} \end{pmatrix}$ (with indices taken as follows: $-3 \leq l \leq 0$ and $0 \leq k \leq 3$).

(Note that the DFT of $\tilde{g}$ is defined as: $\sum_{k=0}^{3} \sum_{l=-3}^{0} \tilde{g}(k,l) e^{-2\pi j \frac{km+ln}{4}}$.)

4. DFT of a shifted image

Let $g(k', l')$ be an $N \times N$ image.
Assume the indices of the image are taken as $-k_0 \leq k' \leq N-1-k_0$ and $-l_0 \leq l' \leq N-1-l_0$.
Let $\tilde{g}$ be the shifted image whose indices are taken as $0 \leq k, l \leq N-1$, then:

$$\begin{aligned} \tilde{g}(k,l) &= g(k-k_0, l-l_0) \\ \hat{\tilde{g}}(m,n) &= \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} g(k-k_0, l-l_0) e^{-2\pi j \frac{km+ln}{N}} \\ &= \underbrace{\frac{1}{N^2} \sum_{k'=-k_0}^{N-1-k_0} \sum_{l'=-l_0}^{N-1-l_0} g(k',l') e^{-2\pi j \frac{k'm+l'n}{N}}}_{\hat{g}(m,n)} e^{-2\pi j \frac{k_0 m + l_0 n}{N}}. \end{aligned}$$

Therefore,

$$\boxed{\hat{\tilde{g}}(m,n) = \hat{g}(m,n) e^{-2\pi j \frac{k_0 m + l_0 n}{N}}}$$

DFT of the shifted image = DFT of the original image $\times\, e^{-2\pi j \frac{k_0 m + l_0 n}{N}}$.

Remark. *We can show that* $\hat{g}(m-m_0, n-n_0) = \text{DFT}\left(\textit{image} \times e^{2\pi j \frac{m_0 k + n_0 l}{N}}\right)$ *(with carefully chosen indices).*

4.1 Fast Fourier Transform (FFT)

Goal: To compute DFT efficiently.

Fast Fourier Transform (FFT)

DFT is separable $\implies$ 2D DFT = Two 1D DFT.
We will discuss how to compute 1D DFT fast.

The 1D DFT is: $\hat{f}(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \omega_N^{ux}$, $\omega_N \equiv e^{-\frac{2\pi}{N} j}$.

Assume $N = 2^n = 2M$, then:

$$\hat{f}(u) = \frac{1}{2M} \sum_{x=0}^{2M-1} f(x) \omega_{2M}^{ux}.$$

Separate it into odd and even parts:

$$\hat{f}(u) = \frac{1}{2} \left\{ \frac{1}{M} \sum_{y=0}^{M-1} f(2y) \underbrace{\omega_{2M}^{u(2y)}}_{\omega_M^{uy}} + \frac{1}{M} \sum_{y=0}^{M-1} f(2y+1) \underbrace{\omega_{2M}^{u(2y+1)}}_{\omega_M^{uy} \omega_{2M}^{u}} \right\}.$$

Then:

$$\hat{f}(u) = \frac{1}{2}\left\{\frac{1}{M}\sum_{y=0}^{M-1} f(2y)\omega_M^{uy} + \frac{1}{M}\sum_{y=0}^{M-1} f(2y+1)\omega_M^{uy}\omega_{2M}^{u}\right\}.$$

For $u < M$,

$$\begin{aligned}\hat{f}(u) &= \frac{1}{2}\left\{\hat{f}_{\text{even}}(u) + \hat{f}_{\text{odd}}(u)\omega_{2M}^{u}\right\} \quad \text{where} \\ \hat{f}_{\text{even}}(u) &\equiv \frac{1}{M}\sum_{y=0}^{M-1} f(2y)\omega_M^{uy} \quad \text{(DFT of even part of } f\text{)} \\ \text{and } \hat{f}_{\text{odd}}(u) &\equiv \frac{1}{M}\sum_{y=0}^{M-1} f(2y+1)\omega_M^{uy} \quad \text{(DFT of odd part of } f\text{).}\end{aligned}$$

For $u \geq M$, we consider:

$$\begin{aligned}\hat{f}(u+M) &= \frac{1}{2}\left\{\frac{1}{M}\sum_{y=0}^{M-1} f(2y)\omega_M^{uy+My} + \frac{1}{M}\sum_{y=0}^{M-1} f(2y+1)\underbrace{\omega_M^{uy+My}}_{\omega_M^{uy}}\underbrace{\omega_{2M}^{u+M}}_{-\omega_{2M}^{u}}\right\} \\ \text{and } \hat{f}(u+M) &= \frac{1}{2}\left\{\hat{f}_{\text{even}}(u) - \hat{f}_{\text{odd}}(u)\omega_{2M}^{u}\right\}.\end{aligned}$$

The FFT algorithm can now be described as follows.

Fast Fourier Transform (FFT) Algorithm

Let $f \in \mathbb{R}^N$ where $N = 2^n = 2M$.

Step 1: Split f into:

$$\begin{aligned}f_{\text{even}} &= [f(0), f(2), \ldots, f(2M-2)]^T \\ \text{and } f_{\text{odd}} &= [f(1), f(3), \ldots, f(2M-1)]^T.\end{aligned}$$

Step 2: Compute $\hat{f}_{\text{even}} = F_M f_{\text{even}}$ and $\hat{f}_{\text{odd}} = F_M f_{\text{odd}}$, where $F_M = (\omega_M^{ux})_{0\leq u,x\leq M-1}$ is an $M\times M$ matrix.
(Recall that the computation of $\hat{f}_{\text{even}}$ is equivalent to the left multiplication of f_{even} by an $M\times M$ matrix. Similarly for $\hat{f}_{\text{odd}}$).

Step 3: Compute $\hat{f}$ using the following formula:
For $u = 0, 1, 2, \ldots, M-1$,

$$\begin{aligned}\hat{f}(u) &= \frac{1}{2}[\hat{f}_{\text{even}}(u) + \hat{f}_{\text{odd}}(u)\omega_{2M}^{u}], \\ \text{and } \hat{f}(u+M) &= \frac{1}{2}[\hat{f}_{\text{even}}(u) - \hat{f}_{\text{odd}}(u)\omega_{2M}^{u}].\end{aligned}$$

Remark. *For Step 2, we can apply the splitting idea to compute $\hat{f}_{even}$ and $\hat{f}_{odd}$.*

Computational cost of FFT

Let C_m be the computational cost of $F_m\mathbf{x}$. Then, $C_1 = 1$.

Obviously,

$$C_N = 2C_M + 3M$$

(2 matrix multiplication, M multiplication, addition, and subtraction)

Hence,

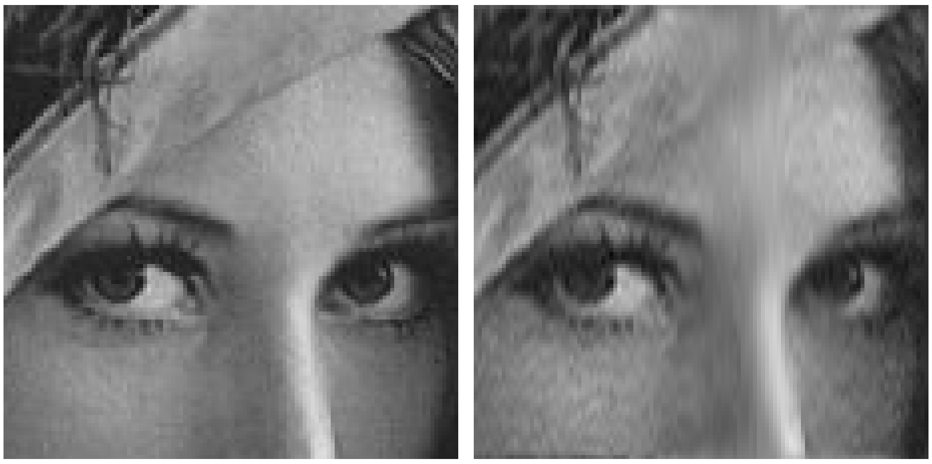
$$\begin{aligned}C_{2^n} &= 2C_{2^{n-1}} + 3M \text{ implies:} \\ 2^{-n}C_{2^n} &= 2^{-(n-1)}C_{2^{n-1}} + 3/2.\end{aligned}$$

The above recursive equation gives $2^{-n}C_{2^n} = C_1 + n(3/2)$. Thus, $C_{2^n} = 2^n + n2^n(3/2)$. We conclude that the computational cost C_N is bounded by $KN \log_2 N$. We denote it by $\mathcal{O}(N \log_2 N)$.

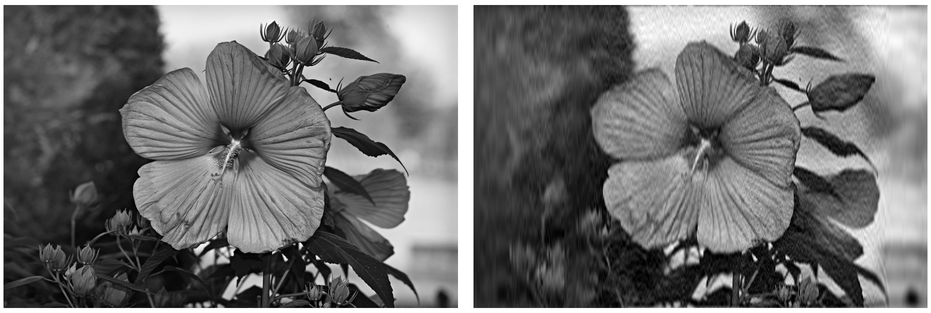
Application of DFT for image compression

DFT is often used for image compression. It is often done by truncating terms associated to small Fourier coefficients. Below we will show some examples of image compression using DFT.

The following shows the image compression of the lena image, by keeping the largest 20% of the Fourier coefficients.



The following shows the image compression of an image of a flower, by keeping the largest 5% of the Fourier coefficients.



DFT is a very popular method for image compression. In fact, DFT is a very important tool for many other image processing tasks, such as image denoising, image deblurring and image sharpening. These will be discussed in the next chapter.

5 Discrete Cosine Transform

Note: Discrete Cosine Transform will not be covered in this course. It is in the lecture note for the sake of completeness.

5.1 Even symmetric discrete cosine transform

Goal: Write an image as a linear combination of cosine functions only. Hence, the elementary images are real-valued.

Consider an $N \times N$ image f. Extend f to a $2M \times 2N$ image $\tilde{f}$, whose indices are taken from $[-M, M-1]$ and $[-N, N-1]$.

Define $f(k,l)$ for $-M \le k \le M-1$ and $-N \le l \le N-1$ such that

$$f(-k-1,-l-1) = f(k,l)\} \text{ Reflection about } (-\frac{1}{2},-\frac{1}{2})$$

$$\left.\begin{aligned} f(-k-1,l) &= f(k,l) \\ f(k,-l-1) &= f(k,l) \end{aligned}\right\} \text{ Reflection about the axes } k=-\frac{1}{2} \text{ and } l=-\frac{1}{2}$$

Next, we make the extension as a reflection about (0,0), the axis $k=0$ and the axis $l=0$. This can be done by shifting the image by $(\frac{1}{2},\frac{1}{2})$.

Here is a simple example illustrating the idea. Let $f = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$. The extended image becomes:

$$\begin{array}{l|ccc:ccc}
\frac{1}{2}-3 \rightarrow & 9 & 8 & 7 & 7 & 8 & 9 \\
\frac{1}{2}-2 \rightarrow & 6 & 5 & 4 & 4 & 5 & 6 \\
\frac{1}{2}-1 \rightarrow & 3 & 2 & 1 & 1 & 2 & 3 \\
\hdashline
\frac{1}{2}+0 \rightarrow & 3 & 2 & 1 & 1 & 2 & 3 \\
\frac{1}{2}+1 \rightarrow & 6 & 5 & 4 & 4 & 5 & 6 \\
\frac{1}{2}+2 \rightarrow & 9 & 8 & 7 & 7 & 8 & 9 \\
 & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
 & \frac{1}{2}-3 & \frac{1}{2}-2 & \frac{1}{2}-1 & \frac{1}{2}+0 & \frac{1}{2}+1 & \frac{1}{2}+2
\end{array}$$

Now, we compute the DFT of (shifted) $\tilde{f}$:

$$\begin{aligned}
F(m,n) &= \frac{1}{(2M)(2N)} \sum_{k=-M}^{M-1} \sum_{l=-N}^{N-1} f(k,l) e^{-2\pi j \frac{m}{2M}(k+\frac{1}{2})} e^{-2\pi j \frac{n}{2N}(l+\frac{1}{2})} \\
&= \frac{1}{4MN} \sum_{k=-M}^{M-1} \sum_{l=-N}^{N-1} f(k,l) e^{-\pi j \frac{m}{M}(k+\frac{1}{2}) - \pi j \frac{n}{N}(l+\frac{1}{2})} \\
&= \frac{1}{4MN} (\underbrace{\sum_{k=-M}^{-1} \sum_{l=-N}^{-1}}_{A_1} + \underbrace{\sum_{k=-M}^{-1} \sum_{l=0}^{N-1}}_{A_2} + \underbrace{\sum_{k=0}^{M-1} \sum_{l=-N}^{-1}}_{A_3} + \underbrace{\sum_{k=0}^{M-1} \sum_{l=0}^{N-1}}_{A_4}) \\
&\quad f(k,l) e^{-\pi j \frac{m}{M}(k+\frac{1}{2}) - \pi j \frac{n}{N}(l+\frac{1}{2})}.
\end{aligned}$$

For A_1, change of variable: $\tilde{k} \equiv -k-1$, $\tilde{l} \equiv -l-1$, and then apply trigonometric identity, and we obtain

$$\begin{aligned}
A_1 = \sum_{\tilde{k}=0}^{M-1} \sum_{\tilde{l}=0}^{N-1} f(\tilde{k},\tilde{l}) \Big\{ &\cos\left[\frac{m\pi}{M}\left(\tilde{k}+\frac{1}{2}\right)\right] \cos\left[\frac{n\pi}{N}\left(\tilde{l}+\frac{1}{2}\right)\right] \\
&- \sin\left[\frac{m\pi}{M}\left(\tilde{k}+\frac{1}{2}\right)\right] \sin\left[\frac{n\pi}{N}\left(\tilde{l}+\frac{1}{2}\right)\right] \\
&+ j \sin\left[\frac{m\pi}{M}\left(\tilde{k}+\frac{1}{2}\right)\right] \cos\left[\frac{n\pi}{N}\left(\tilde{l}+\frac{1}{2}\right)\right] \\
&+ j \cos\left[\frac{m\pi}{M}\left(\tilde{k}+\frac{1}{2}\right)\right] \sin\left[\frac{n\pi}{N}\left(\tilde{l}+\frac{1}{2}\right)\right] \Big\}.
\end{aligned}$$

Similarly, we can perform the same computation to A_2, A_3, A_4. All together, we can obtain

$$A_1 + A_2 + A_3 + A_4 = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f(k,l) \cos\left[\frac{m\pi}{M}\left(k+\frac{1}{2}\right)\right] \cos\left[\frac{n\pi}{N}\left(l+\frac{1}{2}\right)\right].$$

Definition 5.1 (Even symmetric discrete cosine transform).

Let f be a $M \times N$ image, whose indices are taken as $0 \leq k \leq M-1$ and $0 \leq l \leq N-1$. The **even symmetric discrete cosine transform (EDCT)** of f is given by:

$$\hat{f}_{ec}(m,n) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f(k,l) \cos\left[\frac{m\pi}{M}\left(k+\frac{1}{2}\right)\right] \cos\left[\frac{n\pi}{N}\left(l+\frac{1}{2}\right)\right]$$

with $0 \leq m \leq M-1$, $0 \leq n \leq N-1$.

Remark.

- *The inverse of EDCT can be explicitly computed. More specifically, the* **inverse EDCT** *is defined as:*

$$f(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} C(m)C(n)\hat{f}_{ec}(m,n) \cos\frac{\pi m(2k+1)}{2M} \cos\frac{\pi n(2l+1)}{2N} \qquad (***)$$

- *Formula (***) can be expressed as matrix multiplication:*

$$f = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \hat{f}_{ec}(m,n)\vec{T}_m \vec{T}_n'^T$$

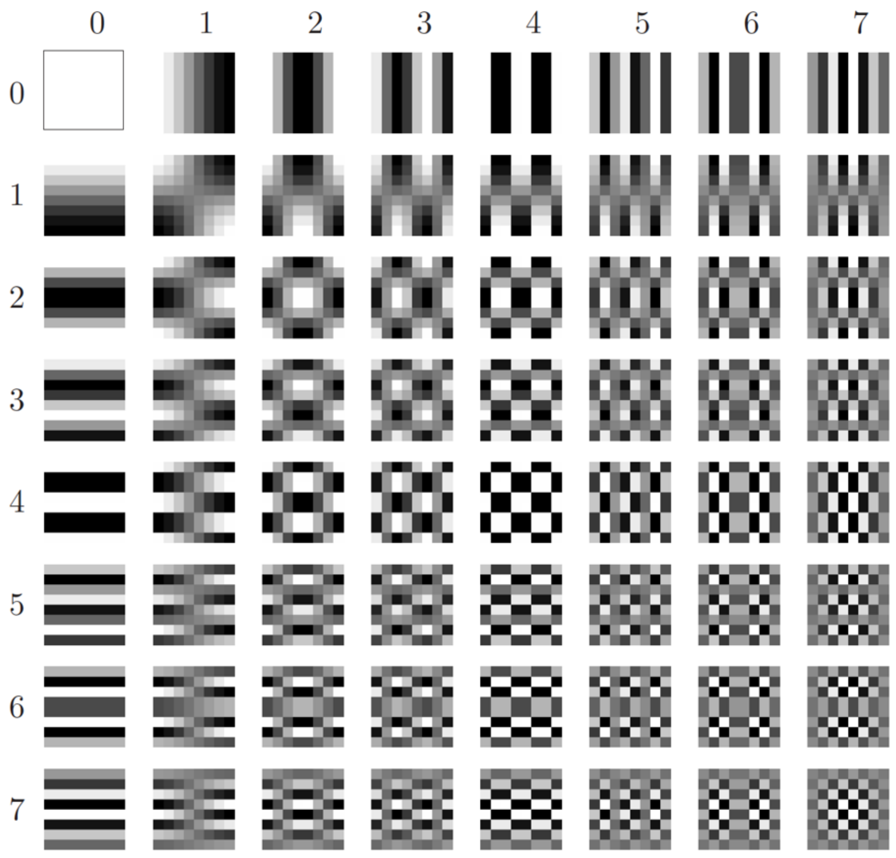
where: $\vec{T}_m = \begin{pmatrix} T_m(0) \\ T_m(1) \\ \vdots \\ T_m(M-1) \end{pmatrix}$, $\vec{T}_n' = \begin{pmatrix} T_n'(0) \\ T_n'(1) \\ \vdots \\ T_n'(N-1) \end{pmatrix}$ *with* $T_m(k) = C(m)\cos\frac{\pi m(2k+1)}{2M}$ *and* $T_n'(k) = C(n)\cos\frac{\pi n(2k+1)}{2N}$.

- $T_m T_n'^T$ *= elementary images of EDCT.*

<u>Image decomposition under EDCT</u>

(Please refer to the Lecture 7 Powerpoint for clearer figures)

The elementary images under the EDCT (of an 8×8 image) look like the following:



Here, the m-th row n-th column image represents the elementary image $T_m T_n'^T$.

Application of EDCT for image compression

EDCT can be applied for image compression by truncating terms associated to small coefficients. **In fact, the famous JPEG compression applies EDCT.**

Below, we will show some results of image compression using EDCT.

In the above figure, (a) shows the original Saturn image. (b) shows the compressed image by keeping the largest 75% of EDCT coefficients. (c) shows the compressed image by keeping the largest 50% of EDCT coefficients. (d) shows the compressed image by keeping the largest 25% of EDCT coefficients.

In the above figure, (a) shows the original 'kid' image. (b) shows the compressed image by keeping the largest 75% of EDCT coefficients. (c) shows the compressed image by keeping the largest 50% of EDCT coefficients. (d) shows the compressed image by keeping the largest 25% of EDCT coefficients.

In the above figure, (a) shows the original 'baboon' image. (b) shows the compressed image by keeping the largest 75% of EDCT coefficients. (c) shows the compressed image by keeping the largest 50% of EDCT coefficients. (d) shows the compressed image by keeping the largest 25% of EDCT coefficients.

5.2 Odd symmetric discrete cosine transform

Other than even symmetric discrete cosine transform, the **odd symmetric discrete cosine transform** can also be defined. We will skip the details but just state the definition.

Definition 5.2 (Odd symmetric discrete cosine transform)**.**

Let f be a $M \times N$ image, whose indices are taken as $0 \leq k \leq M-1$ and $0 \leq l \leq N-1$. The **odd symmetric discrete cosine transform (ODCT)** of f is given by:

$$\hat{f}_{oc}(m,n) = \frac{1}{(2M-1)(2N-1)} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} C(k)C(l)f(k,l) \cos\frac{2\pi mk}{2M-1} \cos\frac{2\pi nl}{2N-1}$$

where $C(0) = 1$ and $C(k) = C(l) = 2$ for $k, l \neq 0$, $0 \leq m \leq M-1$, $0 \leq n \leq N-1$.

The **inverse ODCT** is given by:

$$f(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} C(m)C(n)\hat{f}_{oc}(m,n) \cos\frac{2\pi mk}{2M-1} \cos\frac{2\pi nl}{2N-1}$$

where $C(0) = 1$, $C(m) = C(n) = 2$ if $m, n \neq 0$.

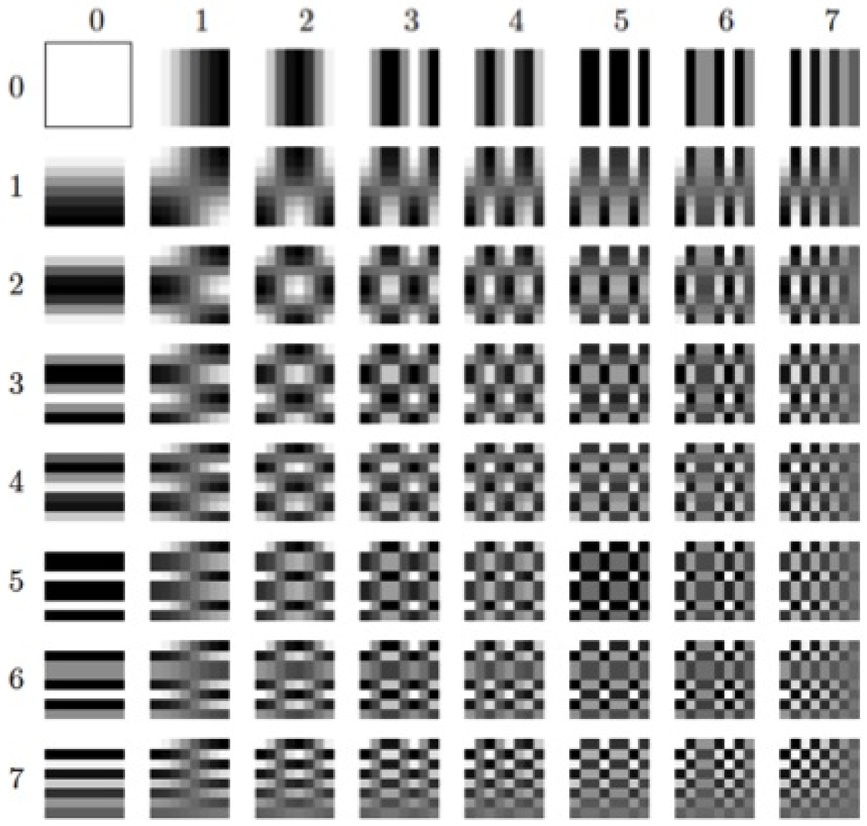
Image decomposition under ODCT

(Please refer to the Lecture 7 Powerpoint for clearer images)

Using ODCT, the image f can be written as:

$$f = \sum_{0 \leq m,n \leq N-1} g_{mn} I_{mn}^{ODCT}.$$

The elementary images under the ODCT (of an 8×8 image) look like the following:



Here, the m-th row n-th column image represents the elementary image I_{mn}^{ODCT}.

6 Conclusion

In this chapter, we have studied different image transformation and decomposition techniques. By image decomposition, we can express an image as a linear combination of elementary images. Each elementary image has different properties. By truncating unnecessary terms, images can be effectively compressed to save storage. Image compression is one of the most important and earliest image processing tasks. We have shown the compression results using SVD, Haar, Walsh, DFT and DCT. In fact, there are other image transformation tools for image compression (such as even/odd antisymmetric discrete sine transform). Please refer to Lecture 7 Powerpoint for details. Amongst different techniques, EDCT usually gives the best compression results. This is the reason why EDCT is applied for the famous JPEG compression.