



## 交流图片

图像和视频编码课程

2014 年, 第 133-169 页

## 第5章-图像和视频编码的转换

大卫·R·布尔

显示更多 ▾

大纲 | 共享 引用

<https://doi.org/10.1016/B978-0-12-405906-1.00005-2>

获取权利和内容

## 摘要

本章的目的是介绍装饰变换的原理，并演示这些原理如何为图像压缩提供基础。本章首先介绍了装饰变换的原理和性质，并解释了它们与主成分分析和特征分析的关系。然后，它引入了一些简单但有用的变换，例如离散的沃尔什-哈达马德变换，然后概述了最优的卡尔胡宁-洛伊夫变换（KLT）。在讨论了KLT的局限性后，本章的其余部分侧重于离散余弦变换（DCT）的推导出和表征。描述了一维DCT及其二维扩展。然后对DCT系数进行量化，并在JPEG标准中进行性能比较。本章最后讨论了DCT实现及其复杂性。通篇提供了示例。

 [上](#)[下一](#) 

## 关键词

單元變化 ; 瓦尔什-哈達馬德變化 ; 卡尔胡宁-洛伊夫變 (KLT) ; 离散余弦變 ; 量子化; JPEG

正如我们在[第三章](#)中所看到的，大多数使用常规采样捕获的自然场景将表现出高度的像素间相关性。这意味着图像区域中像素的实际熵可能明显低于一阶熵，换句话说，空间表示是多余的。如果我们的目标是压缩图像，因此可能会提供更有利于减少冗余的替代表示形式。图像转换的目的是通过装饰图像或图像区域中的像素来创建这样的表示形式。这是通过将图像域映射到像素能量

重新分配并集中到少量系数的替代域来实现的。这个过程有时被称为[能量压实](#)。

我们将看到，这种能量的再分配不会导致数据压缩，因为转换本身是一个无损操作。然而，我们所能实现的只是将能量集中在少数高值系数中。这些较大的系数可能比低值系数具有更高的心理视觉意义，并可以相应地编码。

本章首先介绍了正相关变换的原理和性质，并解释了它们与原理分量分析和[特征分析](#)的关系。我们在第5.4.2节中介绍了最优Karhunen-Loeve变换（KLT），并在讨论了其局限性后，将本章的其余部分重点介绍离散余弦变换（DCT）的推定和表征。DCT在第5.5节中引入，在第5.5.3节中将其扩展到2-D。DCT系数的量化见第5.6节，第5.7节提供了性能比较。最后，我们在第5.8节中讨论了DCT的实施和复杂性，在第5.9节中简要概述了JPEG。

## 5.1。装饰变换的原则

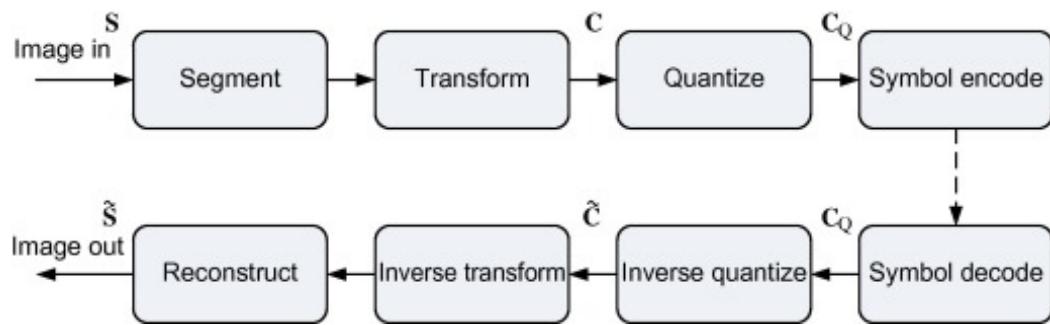
### 5.1.1。基本组成部分

转换为压缩提供了方便的基础，这通过三种机制实现：

1. 它提供数据[装饰](#)，并创建与频率相关的能量分布，允许丢弃低能量系数。
2. 保留系数可以根据其感知重要性，使用标量量化器进行量化。
3. 所有剩余[量化系数](#)的稀疏矩阵都表现出符号冗余，可以使用可变长度编码来利用。

典型的基于变换的图像编码系统中的组件如图5.1所示。正如我们稍后将解释的那样，为了转换编码的目的，输入图像通常被分割成小 $N \times N$ 块，其中值 $N$ 选择在复杂性和装饰性能之间提供妥协。图中所示的组件执行以下功能：

1. **分割**：这将图像分割为 $N \times N$ 街区——通常 $N = 8$ 。这是一个可逆的[一对一映射](#)。
2. **转换（映射或装饰）**：这将原始输入数据转换为更易于压缩的表示形式。同样，这通常是可逆的一对一映射。
3. **量化**：根据保真度和/或比特率标准，这减少了转换输出的动态范围，以减少心理视觉冗余。对于相关空间数据，由此产生的系数块将是稀疏的。这是一个多对一映射，不可逆，因此一旦量化，原始信号就无法完美重建。因此，这是有损压缩的基础。
4. **符号编码（码字分配器）**：量化[系数矩阵](#)的稀疏性被利用（通常通过运行长度编码）来生成一串符号。符号编码器为每个符号分配一个码字（二进制字符串）。该代码旨在减少编码冗余，通常使用可变[长度的代码词](#)。这个操作是可逆的。

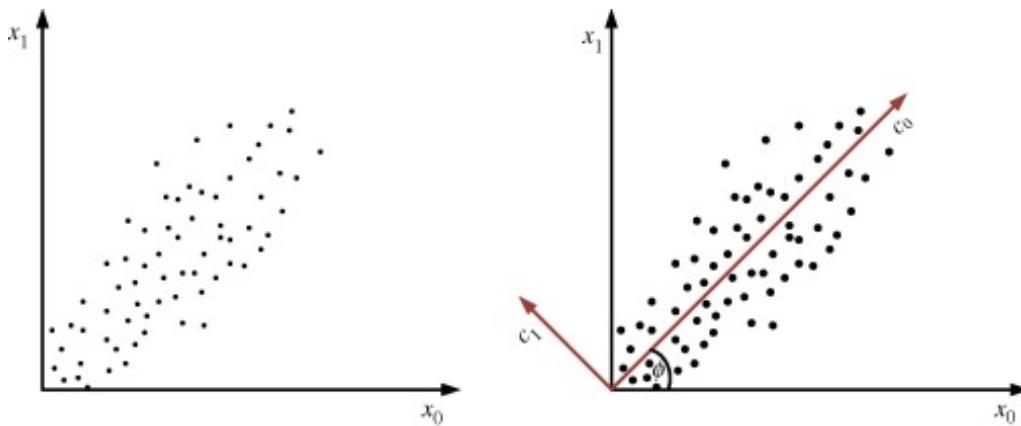


[下载：下载全尺寸图像](#)

图5.1。典型的基于变换的图像压缩架构。

### 5.1.2。主部件和轴旋转

数据压缩背景下的转换目的是**能量压实**。这可以通过识别数据中的趋势，然后修改主轴以优化装饰它来实现。最好通过一个简单的例子来解释。考虑自然图像中任何两个相邻像素之间的相关性，让我们用 $\{x_0, x_1\}$ 。从典型相关图像中绘制的此类相邻像素的散点图可能与图5.2所示。



[下载：下载全尺寸图像](#)

图5.2。通过主轴旋转（右）相关相邻像素数据的图和**装饰关系**。

在图5.2（左）中，我们观察到一种以关系为特征的强相关性 $\hat{x}_1 = x_0$ ，正如从自然图像中预期的那样。我们可以通过引入一组新的轴来利用这种相关性， $c_0$ 和 $c_1$ ，如图5.2所示，该图的定向为 $c_0$ 与数据的主轴对齐。使用图5.3中的注释，我们可以推断：

$$\begin{aligned} A &= x_1 \cos \phi \\ B &= x_0 \sin \phi \\ D &= x_1 \sin \phi \\ E &= x_0 \cos \phi \end{aligned}$$

此外：

$$c_0 = D + E = x_0 \cos \phi + x_1 \sin \phi$$

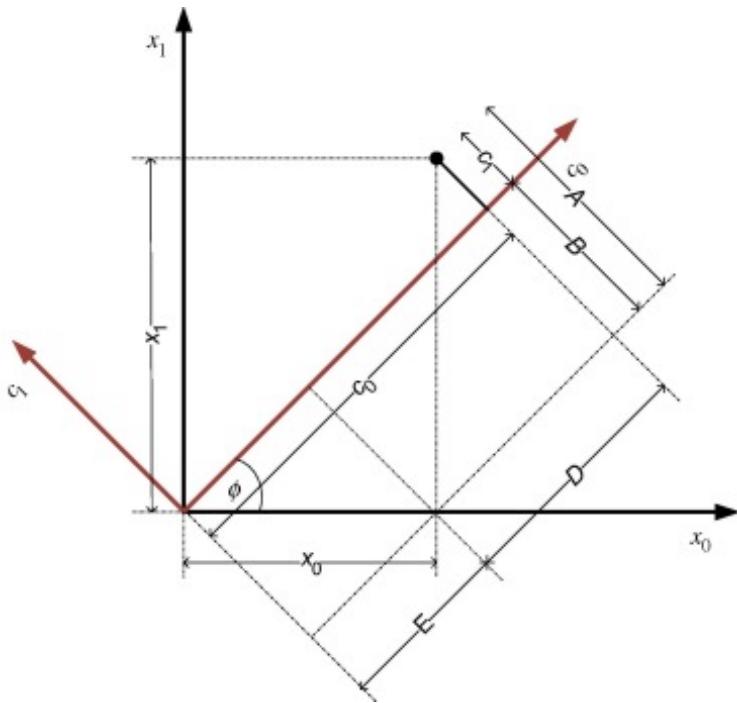
$$c_1 = A - B = -x_0 \sin \phi + x_1 \cos \phi$$

重新安排给出：

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \quad (5.1)$$

或以矩阵向量形式：

$$\mathbf{c} = \mathbf{Ax}$$



[下载：下载全尺寸图像](#)

图5.3。原创数据与转化数据之间的关系。

---

对于相关像素，在 $\phi = 45^\circ$ 冒号：

$$\mathbf{A} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \quad (5.2)$$

正如我们将在第5.3.2节中看到的，这与两点离散沃尔什·哈达马德变换非常相似。我们还将在第5.4.2节中看到，这个过程相当于提取此数据集的特征向量，即 $(1/\sqrt{2}, 1/\sqrt{2})(-1/\sqrt{2}, 1/\sqrt{2})$ 。

从方程 (5.2) 中我们可以看出，这个简单变换的行向量是： $\frac{1}{\sqrt{2}}[1 \ 1]$  和  $\frac{1}{\sqrt{2}}[-1 \ 1]$ 。这些分别是有效的和差运算，可以解释为基本的低通和高通滤波器。例如，在高通情况下：

$$H(z) = \frac{1}{\sqrt{2}}(1 - z^{-1}) \quad (5.3)$$

哪有一根杆子 $z = 0$ 和零在 $z = 1$ 。当我们在第六章中考虑小波和子带滤波器时，这种变换的另一种观点将很好地支持我们。

让我们反思一下我们在本节中取得的成就。首先，我们注意到旋转轴的操作有效地装饰了数据，许多 $c_1$ 价值观很小。在实践中，我们可以将小值设置为零，只留下一组稀疏的系数，这些系数集体现并保留原始数据集的主要属性。然后，这些可以量化，以减少其动态范围，并编码以进行传输或存储。然后，它们可以解码、重新缩放和反向转换，以产生原始数据的近似值，在平均平方误差意义上是最佳的。这就是图5.1中描述的基于转换的数据压缩的本质。

## 5.2。单一变换

### 5.2.1. 基函数和线性组合

我们从第3章对傅里叶分析的介绍中知道，我们可以使用基函数的线性组合来近似信号，例如谐波相关的正弦和余弦或复指数。让我们假设我们有 $N$ 此类基础功能，并注明每个基础功能的贡献 $\mathbf{b}_k$ 需要用一个值来近似数据块 $c(k)$ 在哪里 $k$ 表示基函数的索引。考虑一维 $N \times 1$ 数据块 $\mathbf{x}$ ，我们可以用矩阵-向量的形式写这个关系如下：

$$\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix} = \begin{bmatrix} b_{0,0} & b_{1,0} & \dots & b_{N-1,0} \\ b_{0,1} & b_{1,1} & \dots & b_{N-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{0,N-1} & b_{1,N-1} & \dots & b_{N-1,N-1} \end{bmatrix} \begin{bmatrix} c(0) \\ c(1) \\ \vdots \\ c(N-1) \end{bmatrix} \quad (5.4)$$

但是，如果我们得到一套适当的基函数和一些输入数据，那么我们通常想计算的是系数值或权重 $c(k)$ 正是这些，我们将在图像压缩的背景下处理、存储和传输。因此，让我们将方程 (5.4) 重写如下：

$$\begin{bmatrix} c(0) \\ c(1) \\ \vdots \\ c(N-1) \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1,0} & a_{N-1,1} & \dots & a_{N-1,N-1} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

$$c(k) = \sum_{i=0}^{N-1} x[i] a_{k,i}; \quad k = 0 \dots N-1$$

或者

$$\mathbf{c} = \mathbf{Ax} \quad (5.5)$$

我们已经根据我们的系数将原始信号定义为：

$$\mathbf{x} = \mathbf{Bc}$$

但从方程 (5.5) 中我们也可以看到，可以使用逆变换重建原始信号：

$$\mathbf{x} = \mathbf{A}^{-1} \mathbf{c} \quad (5.6)$$

因此  $\mathbf{B} = \mathbf{A}^{-1}$ 。  $\mathbf{A}$  是一个  $N \times N$  矩阵，简称变换矩阵和  $\mathbf{c}$  是一个  $N \times 1$  变换系数的矢量。矩阵的列  $\mathbf{A}^{-1}$  是变换的基本函数。因此可以观察到，信号  $\mathbf{x}$  由加权基函数的线性组合表示，其中权重由列向量中的系数给出  $\mathbf{c}$ 。

### 5.2.2。正交性和归一化

**正交性** 确保变换的基函数是独立的，因此它们提供了输入向量的装饰关系。基础函数  $\mathbf{a}_i$  和  $\mathbf{a}_j$  如果它们的内积为零，则定义为正交。也就是说：

$$\mathbf{a}_i^H \mathbf{a}_j = 0; \quad \forall i, j \ (i \neq j) \quad (5.7)$$

**正交性** 对基向量的范数施加了进一步的限制，因为它们必须有单位长度，即  $\|\mathbf{a}_i\| = 1$ 。正交性在我们希望使用相同的正向和逆变换操作时非常有用，因为它在域之间保存能量，即：

$$\sum_{k=0}^{N-1} c^2(k) = \sum_{i=0}^{N-1} x^2[i] \quad (5.8)$$

此外，如上所述，它通过换位确保正向和逆基函数矩阵是相互关联的：

$$\mathbf{a}_i^H \mathbf{a}_j = \begin{cases} 1; & i = j \\ 0; & i \neq j \end{cases} \quad (5.9)$$

如果变换是正交（或 ?? 的），那么它有一个唯一和可计算的逆向，由以下给出：

$$\mathbf{A}^{-1} = \mathbf{A}^H \quad (5.10)$$

因此  $\mathbf{A}^H \mathbf{A} = \mathbf{I}$ 。显然，如果  $\mathbf{A}$  矩阵是实值的，那么：

$$\mathbf{A}^{-1} = \mathbf{A}^T \quad (5.11)$$

然后逆变换变为：

$$\mathbf{x} = \mathbf{A}^T \mathbf{c} \quad (5.12)$$

在这种特殊情况下，变换矩阵的行是基函数。单元变换有一些有用的性质：

1. 信号域和变换域中的自相关矩阵（或非零均值的协方差）矩阵通过以下方式相关：

$$\mathbf{R}_c = \mathbf{A} \mathbf{R}_x \mathbf{A}^H \quad (5.13)$$

2. 原始矢量和变换向量的总能量相等（见第 5.4.1 节）。

3. 如果我们选择一个子集  $k$  变换系数，那么如果我们选择  $k$  最大系数。

### 5.2.3。扩展至二维

在二维信号的情况下，让我们假设变换应用于  $N \times N$  真实数据块  $\mathbf{X}$ 。假设可分离变换，如第 3 章和

第5.5.3节所述：

$$\mathbf{C} = \mathbf{A} \mathbf{X} \mathbf{A}^T \quad (5.14)$$

和

$$\mathbf{X} = \mathbf{A}^T \mathbf{C} \mathbf{A} \quad (5.15)$$

请注意，方程 (5.14) 需要两个大小的矩阵乘法  $N \times N$  而不是（在不可分割的情况下）一个大小矩阵的乘法  $N^2 \times N^2$ 。正如我们稍后将看到的，[可分离性](#)的存在和利用对于降低正向变换和逆变换的复杂性很重要。

由此，二维变换的基函数本身就是二维函数，二维变换可以解释为从单个一维基函数的外乘积中获得的矩阵展开式， $\mathbf{a}_j$ 。因此，如果二维基函数表示为  $\alpha_{i,j}$ ，并且一维基函数是实值列向量，则二维函数由  $\mathbf{a}_i$  和  $\mathbf{a}_j$  冒号：

$$\alpha_{i,j} = \mathbf{a}_i \mathbf{a}_j^T \quad (5.16)$$

$$\alpha_{i,j} = \begin{bmatrix} a_{i,0}a_{j,0} & a_{i,0}a_{j,1} & \cdots & a_{i,0}a_{j,N-1} \\ a_{i,1}a_{j,0} & a_{i,1}a_{j,1} & \cdots & a_{i,1}a_{j,N-1} \\ \vdots & \vdots & \ddots & \dots \\ a_{i,N-1}a_{j,0} & a_{i,N-1}a_{j,1} & \cdots & a_{i,N-1}a_{j,N-1} \end{bmatrix} \quad (5.17)$$

### 例5.1基函数的正交性

考虑第5.1.2节的简单变换矩阵：

$$\mathbf{A} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

表明这是一个 ?? 变换。

### 解决方案

基函数是正交的，因为：

$$\frac{1}{2} [1 \ 1] \begin{bmatrix} -1 \\ 1 \end{bmatrix} = 0$$

所有基函数都有单位长度，因为：

$$\|\mathbf{a}_i\| = \frac{1}{\sqrt{2}} \sqrt{(1)^2 + (1)^2} = 1$$

## 5.3。基本变换

### 5.3.1. 哈尔变换

考虑简单的哈尔变换：

$$\mathbf{H}_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix} \quad (5.18)$$

第一个基函数创建输入数据的运行和，第二个基函数创建前两个和后两个数据样本之间的差异，第三个基函数创建前两个数据点之间的差异，同样，底部行的基函数对后两个数据点也是如此。

### 例5.2 能量压实与哈尔变换

考虑输入向量：

$$\mathbf{x} = [1.0 \ 0.5 \ -0.5 \ -1.0]^T$$

表明哈尔变换的应用提供了能量压实。

### 解决方案

哈尔变换减少了非零系数，再加上适量的能量压实，因此：

$$\mathbf{c} = \frac{1}{2} \begin{bmatrix} 0 & 3 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}^T$$

正如我们稍后将看到的，存在其他变换，可以提供更大的能量压实和[装饰关系](#)。

### 5.3.2。沃尔什-哈达马德变换

沃尔什变换和沃尔什-哈达马德变换是压缩数据的简单而有效的方法。它们具有显著的优势，即基本变换不需要乘法，只需要和和差。虽然它们的编码增益低于我们不久将考虑的DCT等变换，但它们确实在现代视频[压缩系统中](#)找到了应用，如H.264/AVC，在那里它们用于帧内编码。

离散沃尔什-哈达马变换（DWHT）是通过离散哈达马矩阵的简单重新排列获得的。哈达马矩阵是一个 $N \times N$ 具有属性的矩阵 $\mathbf{HH}^T = \mathbf{NI}$ 。通过迭代应用以下操作可以找到高阶矩阵：

$$\mathbf{H}_{2N} = \begin{bmatrix} \mathbf{H}_N & \mathbf{H}_N \\ \mathbf{H}_N & -\mathbf{H}_N \end{bmatrix} \quad (5.19)$$

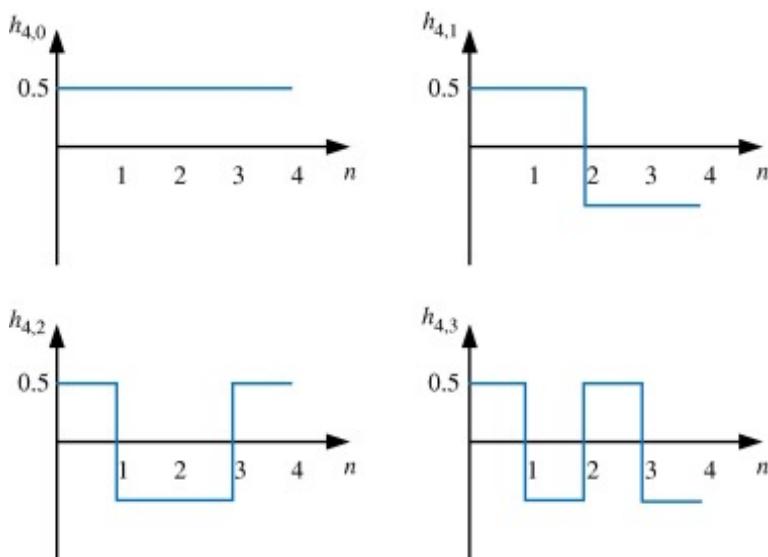
例如：

$$\mathbf{H}_1 = 1; \quad \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \quad \mathbf{H}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \quad (5.20)$$

DWHT只是通过规范化和按顺序重新排列行（即符号变化的数量）从相应的哈达玛矩阵中获得的。因此，四点DWHT由以下方式给出：

$$\mathbf{H}_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \quad (5.21)$$

一维DWHT的基本函数如图5.4所示。



[下载：下载全尺寸图像](#)

图5.4。DWHT基础功能 $N = 4$ 。

在二维变换的情况下，基函数再次由单个一维基函数的外积形成， $\mathbf{h}_j$ 。二维的基础功能 $4 \times 4$  DWHT如图5.5所示。

### 示例5.3一维DWHT

为数据向量计算一维DWHT  $\mathbf{x} = [5 \ 6 \ 4 \ 8]^T$  冒号：

$$\begin{bmatrix} c(0) \\ c(1) \\ c(2) \\ c(3) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \\ 4 \\ 8 \end{bmatrix}$$

### 解决方案

$$\mathbf{c} = [11.5 \ -0.5 \ 1.5 \ -2.5]^T$$

### 示例5.4二维DWHT

计算以下输入图像块 $\mathbf{S}$ 的二维DWHT：

$$\mathbf{S} = \begin{bmatrix} 5 & 6 & 8 & 10 \\ 6 & 6 & 5 & 7 \\ 4 & 5 & 3 & 6 \\ 8 & 7 & 5 & 5 \end{bmatrix}$$

### 解决方案

由于二维DWHT是可分的，我们可以使用方程 (5.14)。放开 $\mathbf{C} = \mathbf{C}' \mathbf{H}^T$  在哪里 $\mathbf{C}' = \mathbf{H} \mathbf{S}$ 冒号：

$$\begin{aligned}\mathbf{C}' &= \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 5 & 6 & 8 & 10 \\ 6 & 6 & 5 & 7 \\ 4 & 5 & 3 & 6 \\ 8 & 7 & 5 & 5 \end{bmatrix} \\ &= \begin{bmatrix} 11.5 & 12 & 10.5 & 14 \\ -0.5 & 0 & 2.5 & 3 \\ 1.5 & 1 & 2.5 & 1 \\ -2.5 & -1 & 0.5 & 2 \end{bmatrix}\end{aligned}$$

然后

$$\mathbf{C} = \begin{bmatrix} 24 & -0.5 & 1.5 & 2 \\ 2.5 & 3 & 0 & -0.5 \\ 3 & -0.5 & -0.5 & 1 \\ -0.5 & -3 & 0 & -1.5 \end{bmatrix}$$

在这个简单的例子中，DWHT的能量压实性和装饰性可以清楚地看到。回想一下，就原始图像块而言，能量在整个块中分布相当均匀。转换后，数据水平和垂直相关，一个显性系数（左上角）现在包含约93%的能量。

### 示例5.5二维DWHT压缩

以[示例5.4](#)的结果，设置所有小值系数（当 $|c_{i,j}| \leq 2$ ）到零，执行逆变换。如果原始信号以6位字长表示，重建的PSNR是多少？

### 解决方案

立即

$$\widetilde{\mathbf{C}} = \begin{bmatrix} 24 & 0 & 0 & 0 \\ 2.5 & 3 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \end{bmatrix}$$

和

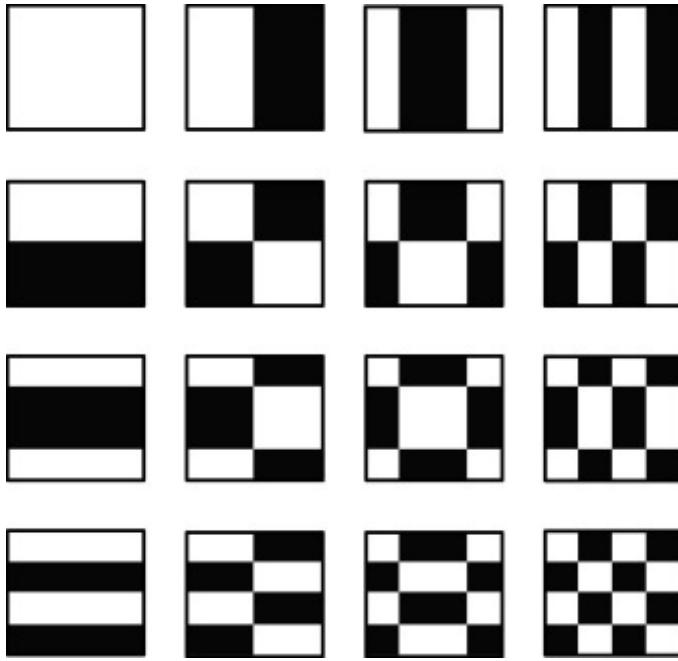
$$\widetilde{\mathbf{s}} = \mathbf{H}^T \widetilde{\mathbf{C}} \mathbf{H}$$

所以

$$\widetilde{\mathbf{s}} = \begin{bmatrix} 7 & 7 & 7 & 7 \\ 7 & 7 & 4 & 4 \\ 3 & 3 & 6 & 6 \\ 6 & 6 & 6 & 6 \end{bmatrix}$$

假设6位字长，MSE = 3，重建的PSNR是：

$$\text{PSNR} = 10 \log_{10} \left[ \frac{16(2^6 - 1)^2}{\sum_{\forall(i,j)} (s[i,j] - \tilde{s}[i,j])^2} \right] = 31.2 \text{ dB}$$



[下载：下载全尺寸图像](#)

图5.5。二维DWHT的基础功能。

### 5.3.3。那么为什么不使用离散傅里叶变换呢？

作为选择与数据相关变换的起点，我们可能会做得比考虑熟悉的DFT更糟。毕竟我们知道DFT非常擅长用很少的系数来表示正弦数据。然而，事情没那么简单。

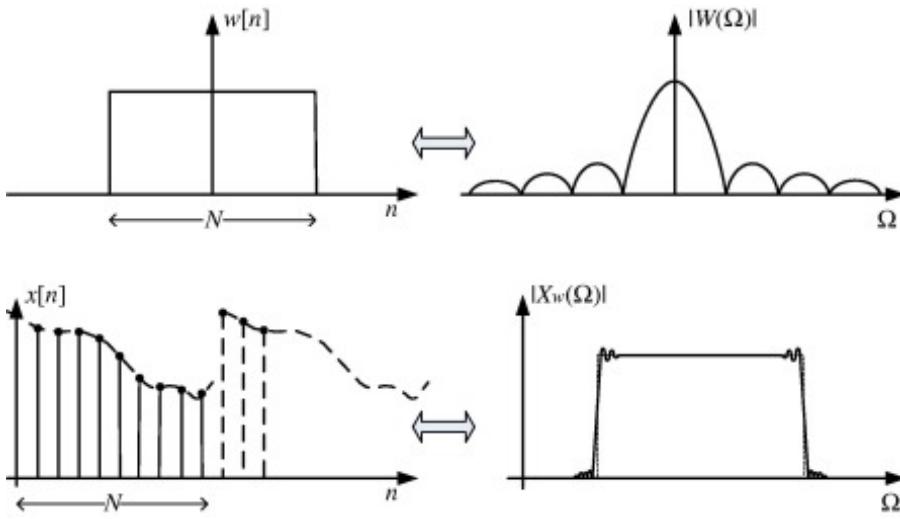
使用DFT，窗口有限长度数据序列在转换前通过周期性扩展自然扩展。将离散时间**傅里叶级数**（DTFS）应用于这个新的周期函数会产生DFT系数。第一个问题是它产生一组复系数。虽然这对于捕获频域中基础信号的大小和相位非常有用，但它对压缩并不特别有用，因为（对于实际输入信号来说）我们正在将信息量增加一倍！

将输出的复杂性暂时放在一边，对于长固定序列或扩展过程不引入不连续的短序列，周期性扩展是可以接受的。然而，对于像压缩中典型的序列这样的较短序列，不连续是常见的，它们在频域中会产生响亮或**光谱泄漏**，可能会对性能产生重大影响。这正是我们在压缩系统中不想要的，因为它在系数空间中引入了更多的能量，需要更多的比特来编码。

矩形窗口对采样和转换过程的影响如图5.6所示。回想一下，输入信号乘以时间（或空间）域中的窗口函数等于其频谱在频域的卷积，即：

$$w[n]x[n] \Leftrightarrow W(\Omega) * \mathcal{X}(\Omega) \quad (5.22)$$

前两个子图形显示长度 $N$ 窗口功能，与光谱一起，而较低的两个子图形在其频谱旁边显示采样和窗口输入信号。由于DFT实际上是通过连接无限个窗口信号样本获得的周期信号的DTFS，因此在窗口边界存在不连续性，如图5.6所示。这些会导致频域的光谱泄漏或涟漪。窗口对由此产生的DFT系数的影响可以清楚地看到。



[下载：下载全尺寸图像](#)

图5.6。由于DFT窗口和周期扩展引起的频谱泄漏：矩形窗口函数及其大小频谱（顶部）；显示周期扩展及其频谱（底部）的输入信号。

在光谱分析应用程序中，环通常通过在扩展前对输入数据应用非矩形窗口函数（如汉明窗口）来解决。与传统的矩形窗户相比，这些窗户的频率域侧叶更小，但中央叶子也更宽。因此，它们减少了光谱环，但涂抹了锐利的频域边缘，扭曲了底层信号的特性。再说一遍，这正是我们不想在压缩系统中想要的！

## 5.4。最佳变换

### 5.4.1。丢弃系数

让我们假设我们希望根据一组将最多能量封装到最少的变换系数中的基础函数定义一个变换。如果我们然后用有限的数字近似我们的信号( $M$ )系数，将剩余系数设置为恒定值， $k_i$ ，首先我们问——应该什么价值 $k$ 是吗？按照克拉克[1]的方法：

$$\tilde{\mathbf{x}}_M = \sum_{i=0}^{M-1} c(i) \mathbf{a}_i + \sum_{i=M}^{N-1} k_i \mathbf{a}_i$$

由此产生的错误可以计算如下：

$$\mathbf{e}_M = \mathbf{x}_M - \tilde{\mathbf{x}}_M = \sum_{i=M}^{N-1} (c(i) - k_i) \mathbf{a}_i$$

我们现在希望计算这个剩余信号中的能量，并尝试将其最小化：

$$\begin{aligned} & E\{e_M^2\} \\ &= E\left\{(c(M) - k_M)^2 \mathbf{a}_M^\top \mathbf{a}_M + (c(M) - k_M)(c(M+1) - k_{M+1}) \mathbf{a}_M^\top \mathbf{a}_{M+1} \right. \\ &\quad \left. + \cdots + (c(N-1) - k_{N-1})(c(N-1) - k_{N-1}) \mathbf{a}_{N-1}^\top \mathbf{a}_{N-1}\right\} \end{aligned}$$

假设变换是正交的（并注意到这里的基函数是列向量）：

$$\mathbf{a}_i^\top \mathbf{a}_j = \begin{cases} 1; & i = j \\ 0; & i \neq j \end{cases}$$

因此，误差能量方程简化为：

$$E[e_M^2] = E\left[\sum_{i=M}^{N-1} (c(i) - k_i)^2\right] \quad (5.23)$$

如果我们通过部分差异来最小化  $k$  并设置结果等于 0，我们可以确定  $k$  的最佳值：

$$\begin{aligned} \frac{\partial}{\partial k_i} E\{e_M^2\} &= \frac{\partial}{\partial k_i} E\left\{\sum_{i=M}^{N-1} (c(i) - k_i)^2\right\} = E\{-2(c(i) - k_i)\} \\ &= 0 \text{ at minimum} \end{aligned}$$

因此，至少：

$$k_i = E\{c(i)\} = \mathbf{a}_i^\top E\{\mathbf{x}\}$$

如果我们假设一个零平均序列，解变成：

$$k_i = 0 \quad (i = M, \dots, N-1)$$

因此，我们最好的近似值是：

$$\tilde{\mathbf{x}}_M = \sum_{i=0}^{M-1} c(i) \mathbf{a}_i \quad (5.24)$$

现在我们可以计算方程 (5.23) 中误差能量最小化的基函数。所以：

$$E\{e_M^2\} = E\left\{\sum_{i=M}^{N-1} (c(i))^2\right\} \quad (5.25)$$

#### 5.4.2. 卡胡宁-洛伊夫变换 (KLT)

也称为 Hotelling 或 Eigenvector 变换，KLT 变换矩阵的基向量， $\mathbf{A}$ ，来自信号自相关矩阵的特征向量（注意与主成分分析 (PCA) 的相似性）。假设信号来自随机过程，由此产生的变换系数将不相关。KLT 通过最小化变换系数方差的几何平均值，为高斯源提供了最佳能量压实，并产生高斯源的统计（均方误差）意义上的最佳性能。为了信号  $\mathbf{x}$ （为了方便起见，有偏见地去掉平均值），基础功能  $\mathbf{a}_k$  可以显示为信号自相关矩阵的特征向量，因此：

$$\mathbf{R}_x \mathbf{a}_k = \lambda_k \mathbf{a}_k \quad (5.26)$$

在哪里  $\lambda_k$  是相应的特征值。考虑一个信号  $\mathbf{x} = \{x[0], x[1], \dots, x[N-1]\}$ 。假设信号是一个静止的随

机零均值序列，该像素块的自相关矩阵由以下方式给出：

$$\mathbf{R}_x = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) \end{bmatrix} \quad (5.27)$$

在哪里

$$r_{xx}(k) = E\{x[n+k]x[n]\} = r_{xx}(-k)$$

如上所述，KLT是使用方程 (5.26) 特征向量作为基向量的 ?? 变换。替换基函数矩阵  $\mathbf{A}$  在特征向量矩阵的方程 (5.13) 中， $\emptyset$ ，我们获得：

$$\mathbf{R}_c = \emptyset^H \mathbf{R}_x \emptyset \quad (5.28)$$

$$= \begin{bmatrix} \boldsymbol{\phi}_1^H \\ \boldsymbol{\phi}_2^H \\ \vdots \\ \boldsymbol{\phi}_N^H \end{bmatrix} \mathbf{R}_x [\boldsymbol{\phi}_1 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_N]$$

但从方程 (5.26) 中，这可以重写为：

$$\begin{aligned} \mathbf{R}_c &= \begin{bmatrix} \boldsymbol{\phi}_1^H \\ \boldsymbol{\phi}_2^H \\ \vdots \\ \boldsymbol{\phi}_N^H \end{bmatrix} \mathbf{R}_x [\lambda_1 \boldsymbol{\phi}_1 \ \lambda_2 \boldsymbol{\phi}_2 \ \cdots \ \lambda_N \boldsymbol{\phi}_N] \\ &= \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{bmatrix} \end{aligned} \quad (5.29)$$

因此，我们可以看到自相关矩阵被KLT对角化。对于任何具有对角线条目的自相关矩阵，都可以显示它， $\sigma_{c,k}^2$ ，以下不平等成立：

$$\det[\mathbf{R}_c] \leq \prod_{\forall k} \sigma_{c,k}^2 \quad (5.30)$$

但是，从方程 (5.13) 中我们可以看出：

$$\det[\mathbf{R}_c] = \det[\mathbf{R}_x]$$

因此，对于任何 ?? 变换，以下不等式成立：

$$\prod_{\forall k} \sigma_{c,k}^2 \geq \det[\mathbf{R}_x] \quad (5.31)$$

但是，就KLT而言，我们可以看到：

$$\prod_{\forall k} \sigma_{c,k}^2 = \det[\mathbf{R}_x] = \det[\mathbf{R}_c] \quad (5.32)$$

因此，KLT最小化几何平均值，因此（见方程 (5.43)）最大化了高斯源的编码增益。根据方程 (5.25)，如果使用系数子集来表示信号，它还提供了最小近似误差。

### 5.4.3。实践中的KLT

KLT的主要缺点是变换依赖于信号，因此需要在信号重建之前计算变换矩阵并将其传输到解码器。在小块大小和信号不平稳的情况下，这可以造成重大开销。此外，对于相关数据（如图像），其他变换，如离散余弦变换（DCT）是KLT的近似值。出于这些原因，DCT是大多数图像和视频编码应用程序的首选转换。

有关 KLT 的更多详细信息，请参阅 Clarke [1] 的优秀文本。

## 5.5。离散余弦变换（DCT）

### 5.5.1。DCT的推导

离散余弦变换最早由Ahmed等人引入[2]，是图像和视频编码应用中最广泛使用的??变换。与离散傅里叶变换一样，DCT提供有关频域信号的信息。然而，与DFT不同，实值信号的DCT本身是实值信号，重要的是，由于输入数据的周期性扩展，它也不会引入工件。

使用DFT，有限长度的数据序列通过周期性扩展自然扩展。因此，时间（或空间）域的不连续性在频域中产生环形或光谱泄漏。如果数据序列在应用DFT之前对称（而不是周期性）扩展，则可以避免这种情况。这会产生一个偶数序列，该序列具有产生实值系数的额外好处。DCT在表示纯正弦波时缺乏功能，因此在频域信号分析方面不如DFT有用。然而，在信号压缩的主要作用中，它的表现非常出色。

我们将看到，DCT具有良好的能量压实性能，其性能接近KLT的相关图像数据。此外，与KLT不同，其基础函数独立于信号。最受欢迎的一维DTC由以下方式给出：

$$c(k) = \sqrt{\frac{2}{N}} \varepsilon_k \sum_{m=0}^{N-1} x[m] \cos\left(\frac{\pi k}{N}(m + \frac{1}{2})\right) \quad (5.33)$$

在二维中，通过：

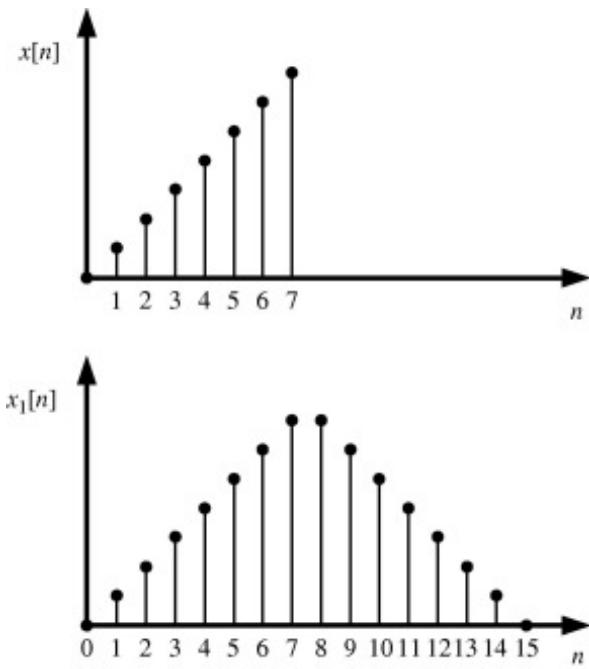
$$c(k,l) = 2 \frac{\varepsilon_k \varepsilon_l}{\sqrt{NM}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x[m,n] \cos\left(\frac{\pi k}{N}(m + \frac{1}{2})\right) \cos\left(\frac{\pi l}{M}(n + \frac{1}{2})\right) \quad (5.34)$$

$$\varepsilon_k = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ 1 & \text{otherwise} \end{cases}$$

### DCT推导

考虑图5.7所示的一维信号。下部子图说明了一维信号如何对称地扩展（或镜像）以形成长度序列

$2N$ 。这有明显的优势是没有引入不连续，因此，当我们应用DFT创建频域表示时，没有响铃元件。



[下载：下载全尺寸图像](#)

图5.7。DCT的对称信号扩展。

所有 $N$ 原始信号的元素， $x[n]$ ，被复制以给出一个新的序列 $x_1[n]$ 冒号：

$$x_1[n] = \begin{cases} x[n]; & 0 \leq n \leq N-1 \\ x[2N-1-n]; & N \leq n \leq 2N-1 \end{cases}$$

DFT $x_1[n]$ 然后由以下方式给出：

$$\begin{aligned} \mathcal{X}_1(k) &= \sum_{n=0}^{N-1} x[n] W_{2N}^{nk} + \sum_{n=N}^{2N-1} x[2N-1-n] W_{2N}^{nk} \\ &= \sum_{n=0}^{N-1} x[n] \left( W_{2N}^{nk} + W_{2N}^{-(n+1)k} \right); \quad 0 \leq k \leq 2N-1 \end{aligned}$$

将分子和分母乘以 $W_{2N}^{0.5k}$ 冒号：

$$\begin{aligned} \mathcal{X}_1(k) &= W_{2N}^{-k/2} \sum_{n=0}^{N-1} x[n] \left( W_{2N}^{(n+0.5)k} + W_{2N}^{-(n+0.5)k} \right) \\ &= 2W_{2N}^{-k/2} \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi(n+0.5)k}{N}\right) \end{aligned}$$

除了旋转因子使结果复杂且不对称外，作为压缩的基础，这开始具有吸引力。由于该术语在压缩性能方面没有任何贡献，因此DCT通常没有它的定义如下：

$$c(k) = W_{2N}^{k/2} \mathcal{X}_1(k)$$

然后规范化为：

$$c(k) = \sqrt{\frac{2}{N}} \varepsilon_k \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi k}{N}(n + 0.5)\right) \quad (5.35)$$

在哪里

$$\varepsilon_k = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ 1 & \text{otherwise} \end{cases}$$

同样，逆DCT的定义是：

$$x[n] = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \varepsilon_k c(k) \cos\left(\frac{\pi k}{N}(n + 0.5)\right) \quad (5.36)$$

### 5.5.2。DCT基函数

基函数是逆变换矩阵的列向量。正是这些由变换系数加权，并线性组合形成信号近似。然而，在实值正交变换的情况下，我们已经看到这些与正向变换矩阵中的行向量相同。

使用方程 (5.35) 我们可以看出DCT基函数由以下方式给出：

$$a(k,n) = \sqrt{\frac{2}{N}} \varepsilon_k \cos\left(\frac{\pi k}{N}(n + 0.5)\right); \quad 0 \leq k, n \leq N - 1 \quad (5.37)$$

#### 示例5.6 四点DCT基函数

计算四点DCT的基函数和变换矩阵。

#### 解决方案

应用方程 (5.37) 我们有：

$$\begin{aligned} \mathbf{A} &= \sqrt{\frac{2}{N}} \varepsilon_k \begin{bmatrix} \cos(0) & \cos(0) & \cos(0) & \cos(0) \\ \cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{3\pi}{8}\right) & \cos\left(\frac{5\pi}{8}\right) & \cos\left(\frac{7\pi}{8}\right) \\ \cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{3\pi}{4}\right) & \cos\left(\frac{5\pi}{4}\right) & \cos\left(\frac{7\pi}{4}\right) \\ \cos\left(\frac{3\pi}{8}\right) & \cos\left(\frac{9\pi}{8}\right) & \cos\left(\frac{15\pi}{8}\right) & \cos\left(\frac{21\pi}{8}\right) \end{bmatrix} \quad (5.38) \\ &= \sqrt{\frac{1}{2}} \varepsilon_k \begin{bmatrix} \cos(0) & \cos(0) & \cos(0) & \cos(0) \\ \cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{\pi}{8}\right) \\ \cos\left(\frac{\pi}{4}\right) & -\cos\left(\frac{\pi}{4}\right) & -\cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{4}\right) \\ \cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{\pi}{8}\right) & -\cos\left(\frac{3\pi}{8}\right) \end{bmatrix} \end{aligned}$$

回到四点DCCT，让我们说服自己这个变换是正交的，并确认 $\varepsilon_k$ 。例如，考虑第一个基函数。我们可以将此重写为矢量 $\mathbf{a}_0$ 如下：

$$\mathbf{a}_0 = k_0 [1 \ 1 \ 1 \ 1]$$

现在我们知道，对于正交性，基函数必须具有单位范数，因此：

$$\|\mathbf{a}_0\| = k_0 \sqrt{1^2 + 1^2 + 1^2 + 1^2} = 2k_0 = 1$$

因此：

$$k_0 = \frac{1}{2}$$

同样，对于第二个基函数：

$$\begin{aligned}\|\mathbf{a}_1\| &= k_1 \sqrt{\cos^2\left(\frac{\pi}{8}\right) + \cos^2\left(\frac{3\pi}{8}\right) + \cos^2\left(\frac{5\pi}{8}\right) + \cos^2\left(\frac{7\pi}{8}\right)} = \sqrt{2}k_1 \\ &= 1\end{aligned}$$

因此：

$$k_1 = \frac{1}{\sqrt{2}}$$

还可以证明：

$$k_2 = k_3 = \frac{1}{\sqrt{2}}$$

因此，正如预期的那样，总的来说：

$$k_i = \frac{1}{\sqrt{2}} \varepsilon_i \quad \varepsilon_i = \begin{cases} 1/\sqrt{2} & i = 0 \\ 1 & i \neq 0 \end{cases}$$

由于我们现在知道DCCT是正交的，我们可以简化逆变换的计算，因为它很容易从方程 (5.11) 中证明：

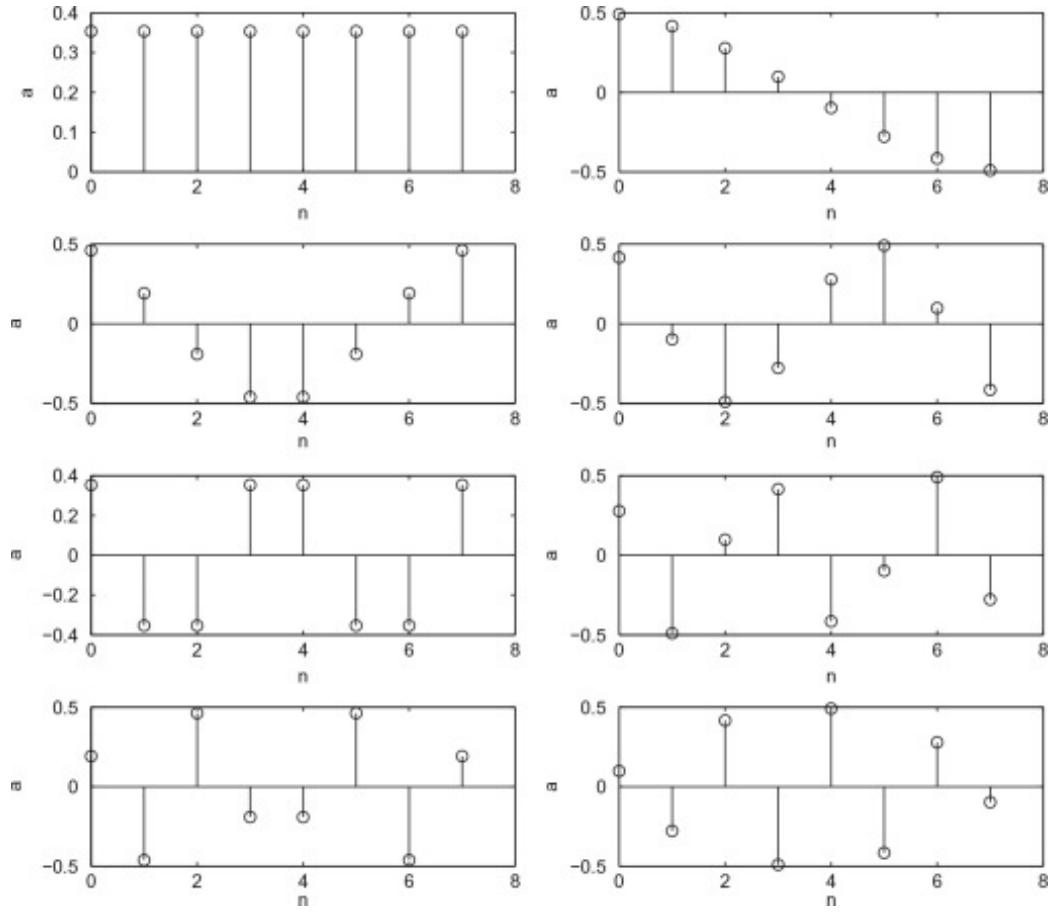
$$\mathbf{A}^{-1} = \mathbf{A}^T = \sqrt{\frac{1}{2}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{3\pi}{8}\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{\pi}{4}\right) & -\cos\left(\frac{\pi}{8}\right) \\ \frac{1}{\sqrt{2}} & -\cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{8}\right) \\ \frac{1}{\sqrt{2}} & -\cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{\pi}{4}\right) & -\cos\left(\frac{3\pi}{8}\right) \end{bmatrix}$$

### 例5.7 八点DCT基函数

我们可以很容易地计算其他长度变换的基础函数。例如，计算八点DCT基函数。

#### 解决方案

八点DCT基函数可以使用方程 (5.37) 计算。这些如图5.8所示。

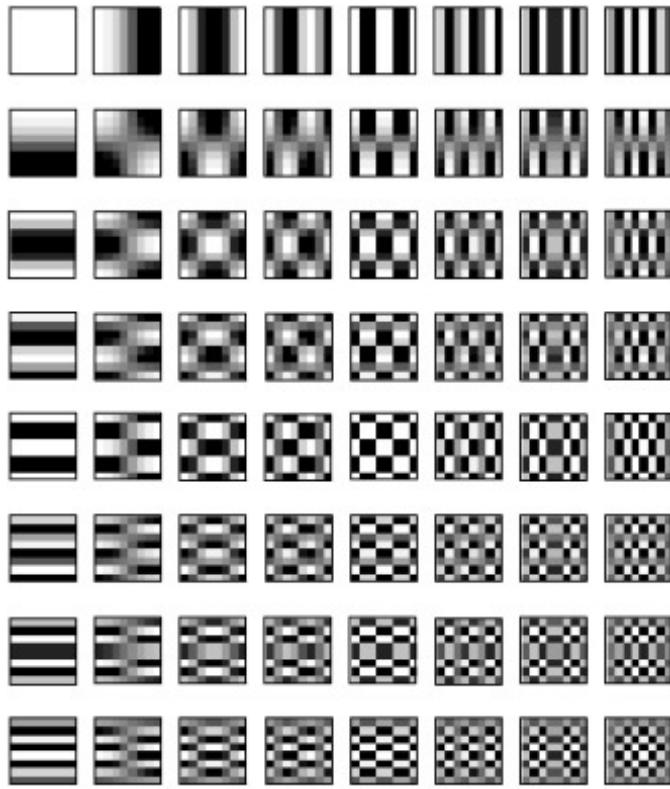


[下载：下载全尺寸图像](#)

图5.8。DCT基础功能 $N = 8$ 。

### 5.5.3。扩展至二维：可分离性

如前所述，二维DCT基函数是从一维基函数的外部乘积中获得的。这些显示在 $8 \times 8$ 图5.9中的DCT。



[下载：下载全尺寸图像](#)

图5.9。二维DCT基础函数 $N = 8$ 。

现在让我们在二维D DCT的背景下考虑可分离性问题。使用 $8 \times 8$ 转换为例：

$$c_2(m,n) = \frac{\epsilon_m \epsilon_n}{4} \sum_{i=0}^7 \sum_{j=0}^7 s[i,j] \cos\left(\frac{(2i+1)m\pi}{16}\right) \cos\left(\frac{(2j+1)n\pi}{16}\right) \quad (5.39)$$

现在，一维DCT由以下方式给出：

$$c_1(m) = \frac{\epsilon_m}{2} \sum_{i=0}^7 s[i,j] \cos\left(\frac{(2i+1)\pi m}{16}\right)$$

因此，方程 (5.39) 可以重新排列如下：

$$c_2(m,n) = \frac{\epsilon_n}{2} \sum_{i=0}^7 c_1(m) \cos\left(\frac{(2j+1)\pi n}{16}\right)$$

换句话说，二维DTC可以通过两次通过形成，首先在 $m$ -方向，然后在 $n$ -方向，因此：

$$c_2(m,n) = \text{DCT}_n^{1-D} (\text{DCT}_m^{1-D}) \quad (5.40)$$

二维DCT（或任何可分离变换）的可分离计算可以可视化，如图5.10所示。



[下载：下载全尺寸图像](#)

图5.10。前向DCT的可分离计算。

可分离性有两个主要优点：

1. 计算次数减少——对于 $8 \times 8$ 案例，每次通过都需要 $8 \times 64$ 乘积运算（MAC），给出总计 $2 \times 8 \times 64 = 1024$  MACs。
2. 一维D DCT可以进一步分解，以产生更大的节省（见第5.8.2节）。

### 示例5.8二维DCT计算

计算以下二维图像块的DCT：

$$\mathbf{S} = \begin{bmatrix} 21 & 19 \\ 15 & 20 \end{bmatrix}$$

### 解决方案

二维DCT变换矩阵可以计算为：

$$\mathbf{A} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

回顾：

$$\mathbf{C}_2 = \mathbf{ASA}^T$$

我们有

$$\begin{aligned} \mathbf{C}_2 &= \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 21 & 19 \\ 15 & 20 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 37.5 & -1.5 \\ 2.5 & 3.5 \end{bmatrix} \end{aligned}$$

现在，如果我们对这个结果执行反向DCT，我们应该恢复原始信号：

$$\tilde{\mathbf{S}} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 37.5 & -1.5 \\ 2.5 & 3.5 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 21 & 19 \\ 15 & 20 \end{bmatrix} = \mathbf{S}$$

### 5.6。DCT系数的量化

正如我们之前讨论过的，[能量压缩](#)过程本身并不提供任何数据压缩。例如，如果我们有一个 $8 \times 8$ 动态范围为8位的空间域像素值块，我们将这些值转换为 $8 \times 8$ 块变换域系数，每个系数也需要8

位，那么我们得到的很少。然而，我们实现的只是将能量集中在少数高值系数中。这些较大的系数可能比低值系数具有更高的心理视觉意义，并可以相应地编码。

**量化是有损压缩的重要一步；然而，它是不可逆转的，会导致信号退化。**正如我们在[第三章](#)中所看到的，量化器包括一组决策级别和一组重建级别。挑战之一是以尽量减少其心理视觉影响的方式进行量化。

变换编码的主要优点是它既提供能量压缩，也提供了[装饰关系](#)。这意味着变换系数可以近似为来自[无记忆源](#)（例如i.i.d）。拉普拉斯）存在简单量化器。因此，帧内变换系数通常使用均匀量化器进行量化，系数预先加权以反映HVS的频率依赖灵敏度。方程 [\(5.41\)](#) 给出了一个捕捉这一点的一般表达式，其中 $Q$ 是量化器步长， $k$ 是一个常数和 $\mathbf{W}$ 是从心理视觉实验中获得的系数相关加权矩阵：

$$c_Q(i,j) = \left\lfloor \frac{kc(i,j)}{Qw_{i,j}} \right\rfloor \quad (5.41)$$

传输或存储后，我们必须在逆变换之前重新缩放量化的变换系数，因此：

$$\tilde{c}(i,j) = \frac{c_Q(i,j)Qw_{i,j}}{k} \quad (5.42)$$

首先，让我们考虑一个简单的例子（[示例5.9](#)），然后再进入更现实的 $8 \times 8$ 案例（[例如5.10](#)）。

### 示例5.9

## 用量化重新计算[示例5.8](#)

使用以下量化矩阵量化[示例5.8](#)中的DCT结果（假设 $k = 1$ 和 $Q = 1$ ）：

$$\mathbf{W} = \begin{bmatrix} 4 & 8 \\ 8 & 8 \end{bmatrix}$$

### 解决方案

立即

$$\mathbf{C}_Q = [\mathbf{C}_2 / W]$$

所以

$$\mathbf{C}_Q = \begin{bmatrix} 9 & 0 \\ 0 & 0 \end{bmatrix}$$

现在计算缩放后的反向二维DCT：

$$\tilde{\mathbf{s}} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 36 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 18 & 18 \\ 18 & 18 \end{bmatrix} \neq \mathbf{s}$$

这留下了以下错误信号：

$$\mathbf{E} = \begin{bmatrix} 3 & 1 \\ -3 & 2 \end{bmatrix}$$

假设6位信号——什么是PSNR?

### 示例5.10

## 8 × 8二维量子化DCT计算

考虑以下几点8 × 8图像块，**S**冒号：

$$\mathbf{S} = \begin{bmatrix} 168 & 163 & 161 & 150 & 154 & 168 & 164 & 154 \\ 171 & 154 & 161 & 150 & 157 & 171 & 150 & 164 \\ 171 & 168 & 147 & 164 & 164 & 161 & 143 & 154 \\ 164 & 171 & 154 & 161 & 157 & 157 & 147 & 132 \\ 161 & 161 & 157 & 154 & 143 & 161 & 154 & 132 \\ 164 & 161 & 161 & 154 & 150 & 157 & 154 & 140 \\ 161 & 168 & 157 & 154 & 161 & 140 & 140 & 132 \\ 154 & 161 & 157 & 150 & 140 & 132 & 136 & 128 \end{bmatrix}$$

首先计算8 × 8变换系数块，**C**，然后使用量化矩阵量化结果的DCT系数，**W**（这实际上是默认的JPEG量化矩阵）。假设**k = 1**和**Q = 1**冒号：

$$\mathbf{W} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

最后计算了重新缩放的DCT系数矩阵，并进行反向DCT以获得重建的图像块，**S̃**。

### 解决方案

$$\mathbf{C} = \begin{bmatrix} 1239 & 50 & -3 & 20 & -10 & -1 & 0 & -6 \\ 35 & -24 & 11 & 13 & 4 & -3 & 14 & -6 \\ -6 & -3 & 8 & -9 & 2 & -3 & 5 & 10 \\ 9 & -10 & 4 & 4 & -15 & 10 & 5 & 6 \\ -12 & 5 & -1 & -2 & -15 & 9 & -6 & -2 \\ 5 & 10 & -7 & 3 & 4 & -7 & -14 & 2 \\ 2 & -2 & 3 & -1 & 1 & 3 & -3 & -4 \\ -1 & 1 & 0 & 2 & 3 & -2 & -4 & -2 \end{bmatrix}$$

使用量化矩阵量化DCT系数，**W**。假设**k = 1**和**Q = 1**，我们有：

$$\mathbf{C}_Q = \begin{bmatrix} 77 & 5 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & -2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

缩放DCT系数矩阵：

$$\widetilde{\mathbf{C}} = \begin{bmatrix} 1232 & 55 & 0 & 16 & 0 & 0 & 0 & 0 \\ 36 & -24 & 14 & 19 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 14 & -17 & 0 & 0 & 0 & 0 & 0 & 0 \\ -18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

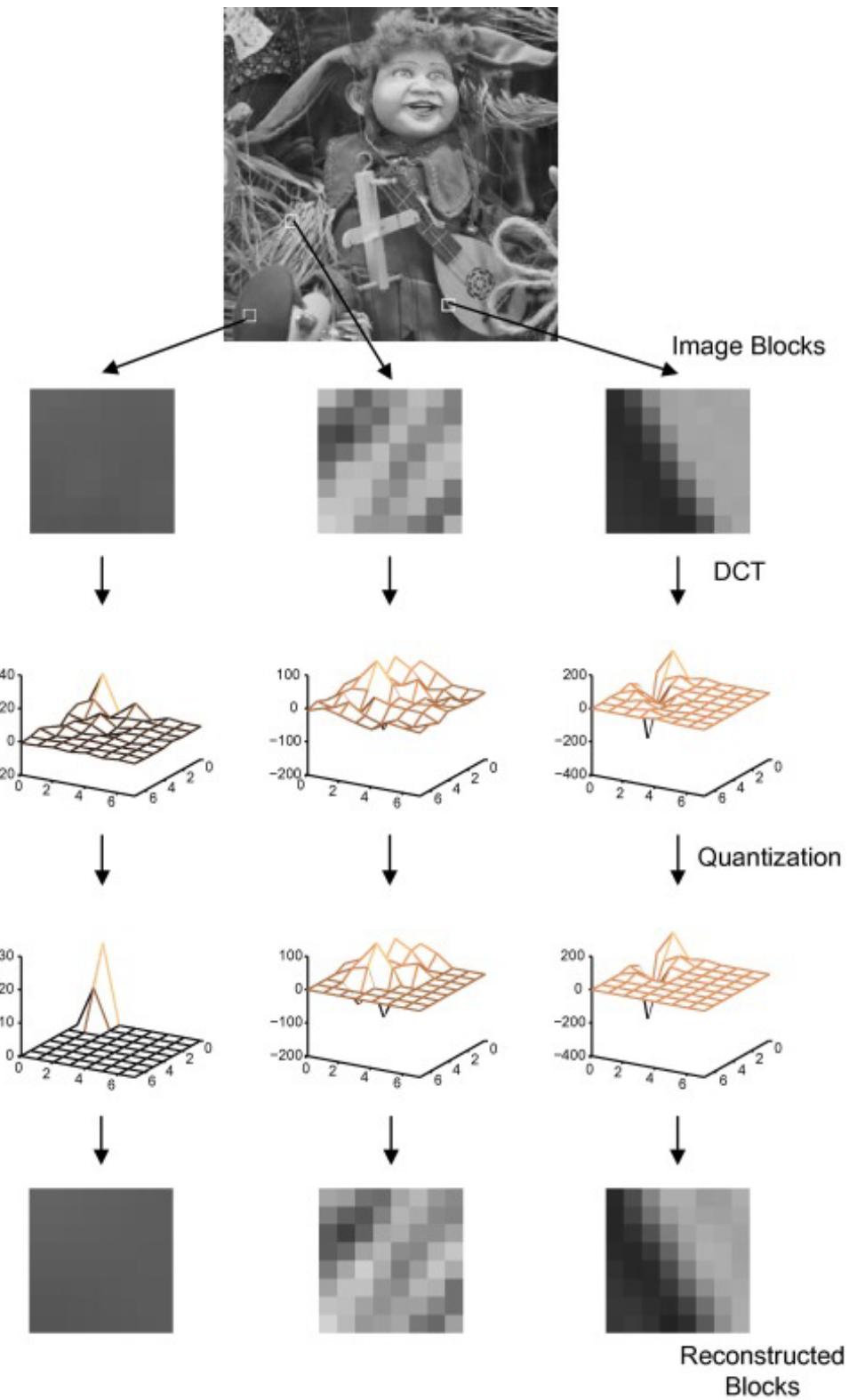
逆变换给出了重建的图像块 $\tilde{\mathbf{S}}$ 冒号：

$$\tilde{\mathbf{S}} = \begin{bmatrix} 173 & 162 & 150 & 149 & 158 & 164 & 164 & 160 \\ 176 & 166 & 156 & 154 & 160 & 163 & 160 & 154 \\ 173 & 165 & 158 & 156 & 160 & 157 & 150 & 143 \\ 163 & 159 & 155 & 154 & 154 & 150 & 142 & 135 \\ 158 & 157 & 156 & 157 & 155 & 151 & 143 & 138 \\ 161 & 161 & 161 & 160 & 157 & 152 & 146 & 143 \\ 163 & 163 & 161 & 156 & 149 & 143 & 139 & 137 \\ 161 & 160 & 156 & 148 & 138 & 131 & 127 & 126 \end{bmatrix}$$

可以看到，量化后得到的矩阵在值上接近原始信号。需要注意的关键是矩阵 $\widetilde{\mathbf{C}}$ 现在非零值稀疏，正如预期的那样（因为输入数据高度相关），DCT系数中的能量在矩阵的左上角压实。

作为额外的练习，计算此近似值的PSNR——假设信号字长为8位。

系数量化对一系列块类型的影响示例见图5.11。正如预期的那样，可以观察到，更多的纹理块需要更多的系数，才能为原始内容创建良好的近似值。对于无纹理块的情况，以更少的系数实现最佳重建，如图中左侧块所示。



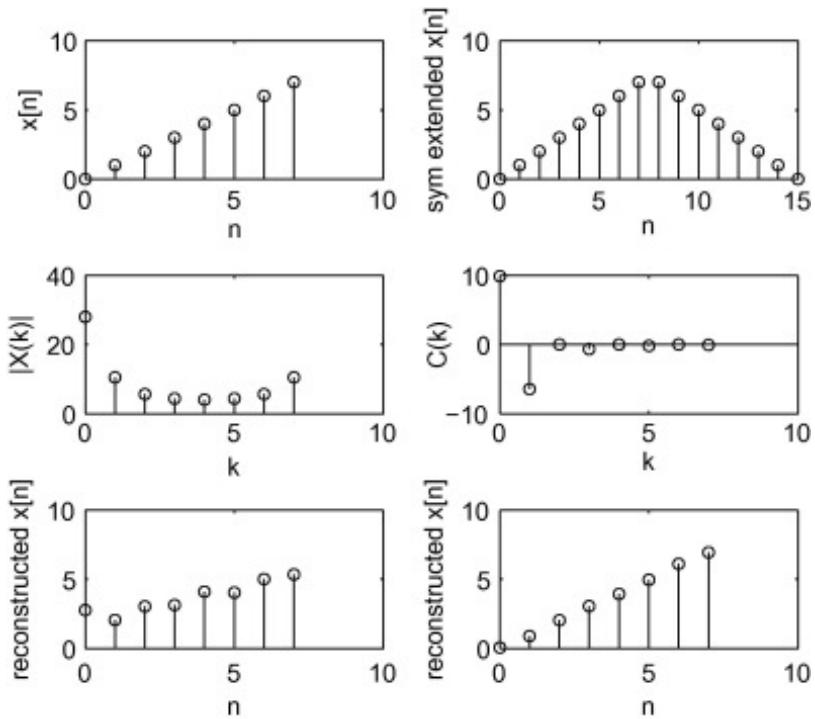
[下载：下载全尺寸图像](#)

图5.11。系数量化对各种数据块的影响。

## 5.7。性能比较

### 5.7.1。DCT vs DFT重访

早些时候，我们解释了为什么DFT不能很好地作为压缩的基础。我们现在可以比较DCT和DFT的性能。[图5.12](#)显示了输入信号 $x[n]$ 被八点DFT和八点DCT压缩。在这两种情况下，压缩都是通过截断变换系数向量只留下前四个系数来实现的，其余的设置为零。可以清楚地看到，DCT的能量压实优于DFT，以及给定压缩策略的重建信号质量。还回顾一下，DFT系数很复杂，因此需要大约两倍的DCT系数带宽。



[下裁：下裁全尺寸图像](#)

[下载](#)

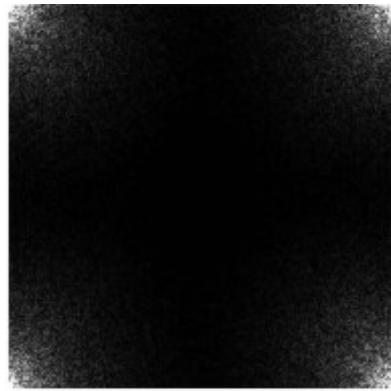
图5.12。用丁压缩的信号的DFT和DCT的对比。顶部：原信号 $x[n]$ 及其对称扩展以及其重建 $x_1[n]$ 。中间：八点DFT $\{X[k]\}$ （仅限震级）和八点DCT $\{x[n]\}$ 。底部：IDFT $\{X(k)\}$ （仅限震级）和IDCT $\{C(k)\}$ 。

## 5.7.2。变换比较

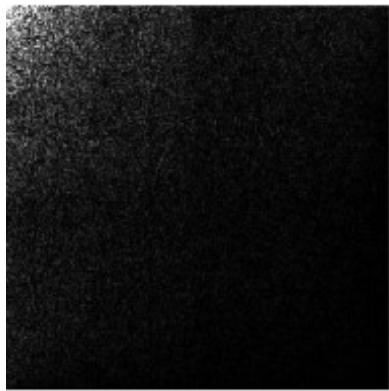
[图5.13](#)显示了转换对自然图像的影响( $256 \times 256$ )使用DFT、DWHT和DCT。正如所料，DFT等光谱表现出多种能量浓度，而DFT的性能提供了显著的能量压实，而DFT没有任何缺点。尽管结构非常简单，但DWHT在这张图像上也提供了良好的性能。



Acer original image



DFT (magnitude)



DWHT



DCT

FEEDBACK

[下载：下载全尺寸图像](#)

图5.13。自然图像上KLT、DFT、DWHT、DST和DCT的比较。

计算一个 $256 \times 256$ 图像不切实际，因此图5.14根据使用 $8 \times 8$ 转换块。这个数字显示了两种情况——第一种情况是每个区块中的64个系数中只有9个保留用于重建，第二种情况是使用25个。可以看出，DCT在质量上接近KLT，但重要的是不需要传输基础功能。这为为什么DCT在图像和视频压缩应用程序中仍然如此受欢迎提供了额外的理由。



DCT first 9 coeffs: PSNR=27.7dB



KLT first 9 coeffs: PSNR=28.4dB



DCT first 25 coeffs: PSNR=33.5dB



KLT first 25 coeffs: PSNR=34.5dB

[下载：下载全尺寸图像](#)

图5.14。KLT和DCT性能比较。

### 5.7.3。DCT的速率失真性能

使用DCT进行基于块变换的编码的好处是，对于相关数据，变换将原始信号中的大部分能量压缩为少量有效系数。块变换的编码增益根据变换块方差的算术和几何平均数之比定义如下：

$$G_{TC} = 10 \log_{10} \left[ \frac{\frac{1}{N} \sum_{i=0}^{N-1} \sigma_i^2}{\left( \prod_{i=0}^{N-1} \sigma_i^2 \right)^{1/N}} \right] \quad (5.43)$$

图5.15显示了DCT压缩性能的示例（ $8 \times 8$ 块）用于 $256 \times 256$ 分辨率图像为0.3 bpp，而图5.16显示了相同图像之间的比较 $256 \times 256$ 图像和等效物 $512 \times 512$ 图像。可以明显看出，对于相同的压缩比，较大的图像失真要小得多。这有两个相当明显的原因：(i)  $512 \times 512$ 尽管图像以相同比例压缩，但比特和(ii)  $512 \times 512$ 图像将显示更高的相关性 $8 \times 8$ 阻止。



[下载：下载全尺寸图像](#)

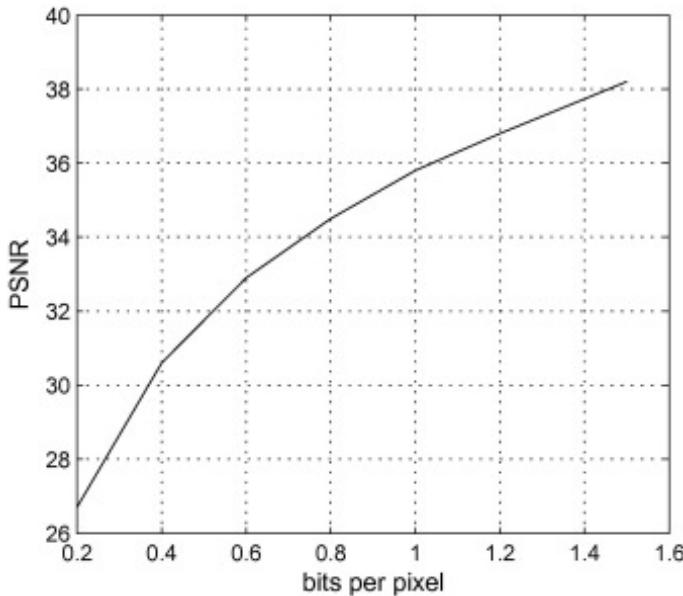
图5.15。DCT性能。左：莉娜原创 $256 \times 256$ 。右：压缩为0.3bpp。



[下载：下载全尺寸图像](#)

图5.16。DCT性能。左：莉娜 $256 \times 256$ 0.3 bpp。右：莉娜 $512 \times 512$ 0.3 bpp。

图5.17显示了基于[Lena图像](#)的DCT编码器在熵编码后速率失真性能的示例（见第7章）。



[下载：下载全尺寸图像](#)

图5.17。DCT速率-失真性能 $256 \times 256$  使用霍夫曼编码的[莉娜图像](#)。

## 5.8。DCT实现

### 5.8.1.转换块大小的选择

转换编码的最佳块大小是多少？这个问题最好通过考虑正在处理的数据和转换本身的复杂性来回答。我们通常寻求[装饰性能](#)和复杂性之间的最佳妥协。如下图所示，典型的尺寸是 $8 \times 8$  pixels，尽管尺寸可以 $4 \times 4$ 用于H.264/AVC和高达块大小的 $64 \times 64$ 在H.265/HEVC中提供。可变块大小现在变得司空见惯，其大小是速率-失真优化（RDO）过程的一部分（见第[10章](#)）。

还值得一提的是，转换的选择可以依赖于数据。例如，在相关性低的情况下，[离散正弦变换](#)（DST）可以优于DCCT。

### DCT复杂性

使用标准二维DCT计算一个 $N \times N$ 块需要 $N^2$ 乘法和 $N^2$ 添加。这体现在顺序的复杂性上 $O(N^4)$ 。这可以简化为 $2N$ 利用[可分性](#)的情况（即 $O(N^3)$ ）。因此，对于一个 $4 \times 4$ 对于一个 $8 \times 8$ 块，每个系数需要16个乘法，对于一个 $16 \times 16$ 块我们需要每个系数32个乘法。

考虑到复杂性和自相关表面特性，在 $4 \times 4$ 和 $16 \times 16$ 通常在实践中使用。

### 5.8.2。DCT复杂性降低

由于其无处不在，有许多快速算法用于计算DTC。早期的快速算法间接操作，利用[快速傅里叶变换](#)（FFT），与 $O(N \log_2 N)$ 复杂性。可以证明[\[1\]](#), [\[2\]](#)一维DCT可以表示为：

$$c(k) = \frac{2}{N} \mathbb{R} \left[ e^{-jk\pi/2^N} \sum_{n=0}^{2N-1} x[n] W^{kn} \right] \quad (5.44)$$

由于需要复杂的乘法和加法，这种方法现在不那么流行。其他方法考虑了 DCT 的 ?? 性质，允许将其分解为相对稀疏矩阵的乘积[4]。递归算法也被提出来 [3]。这些方法的复杂性通常是  $O(N^2 \log_2 N)$ 。

后来的算法直接利用了变换矩阵的性质，要么通过分解、矩阵分区，要么通过在划分和征服方法中利用其对称性。下文介绍了其中一种方法。

## 麦戈文算法

在麦戈文的算法[5]中，矩阵行被分解，使用除法和征服方法利用奇偶对称。该方法的目的是减少每个系数的乘法次数：

$$\begin{aligned} & \begin{bmatrix} c(0) \\ c(1) \\ c(2) \\ c(3) \\ c(4) \\ c(5) \\ c(6) \\ c(7) \end{bmatrix} \\ = & \begin{bmatrix} a_4 & a_4 \\ a_1 & a_3 & a_5 & a_7 & -a_7 & -a_5 & -a_3 & -a_1 \\ a_2 & a_6 & -a_6 & -a_2 & -a_2 & -a_6 & a_2 & a_2 \\ a_3 & -a_7 & -a_1 & -a_5 & a_5 & a_1 & a_7 & -a_3 \\ a_4 & -a_4 & -a_4 & a_4 & a_4 & -a_4 & -a_4 & a_4 \\ a_5 & -a_1 & a_7 & a_3 & -a_3 & -a_7 & a_1 & -a_5 \\ a_6 & -a_2 & a_2 & -a_6 & -a_6 & a_2 & -a_2 & a_6 \\ a_7 & -a_5 & a_3 & -a_1 & a_1 & -a_3 & a_5 & -a_7 \end{bmatrix} \\ & \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix} \end{aligned} \quad (5.45)$$

其中：

$$a_i = \cos(i \frac{\pi}{16})$$

$$\begin{bmatrix} c(0) \\ c(2) \\ c(4) \\ c(6) \end{bmatrix} = \begin{bmatrix} a_4 & a_4 & a_4 & a_4 \\ a_2 & a_6 & -a_6 & -a_2 \\ a_4 & -a_4 & -a_4 & a_4 \\ a_6 & -a_2 & a_2 & -a_6 \end{bmatrix} \begin{bmatrix} x[0] + x[7] \\ x[1] + x[6] \\ x[2] + x[5] \\ x[3] + x[4] \end{bmatrix} \quad (5.46)$$

$$\begin{bmatrix} c(1) \\ c(3) \\ c(5) \\ c(7) \end{bmatrix} = \begin{bmatrix} a_1 & a_3 & a_5 & a_7 \\ a_3 & -a_7 & -a_1 & -a_5 \\ a_5 & -a_1 & a_7 & a_3 \\ a_7 & -a_5 & a_3 & -a_1 \end{bmatrix} \begin{bmatrix} x[0] - x[7] \\ x[1] - x[6] \\ x[2] - x[5] \\ x[3] - x[4] \end{bmatrix} \quad (5.47)$$

方程中的分解 (5.46) 减少了所需的乘法次数  $64 (= N^2)$  在方程 (5.45) 到只有  $32 (= 2(N/2)^2)$ 。请注意，计算偶数组件所需的矩阵  $\mathbf{x}$  与计算四点DTC所需的相同。由此产生的矩阵可以进一步分解为稀疏形式，如下所示：

$$\begin{bmatrix} c(0) \\ c(4) \\ c(2) \\ c(6) \\ c(7) \\ c(5) \\ c(1) \\ c(3) \end{bmatrix} = \begin{bmatrix} a_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_6 & a_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -a_2 & a_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_7 & a_5 & a_1 & a_3 \\ 0 & 0 & 0 & 0 & a_5 & -a_1 & a_3 & a_7 \\ 0 & 0 & 0 & 0 & a_1 & a_3 & -a_7 & a_5 \\ 0 & 0 & 0 & 0 & a_3 & a_7 & -a_5 & -a_1 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \end{bmatrix} \quad (5.48)$$

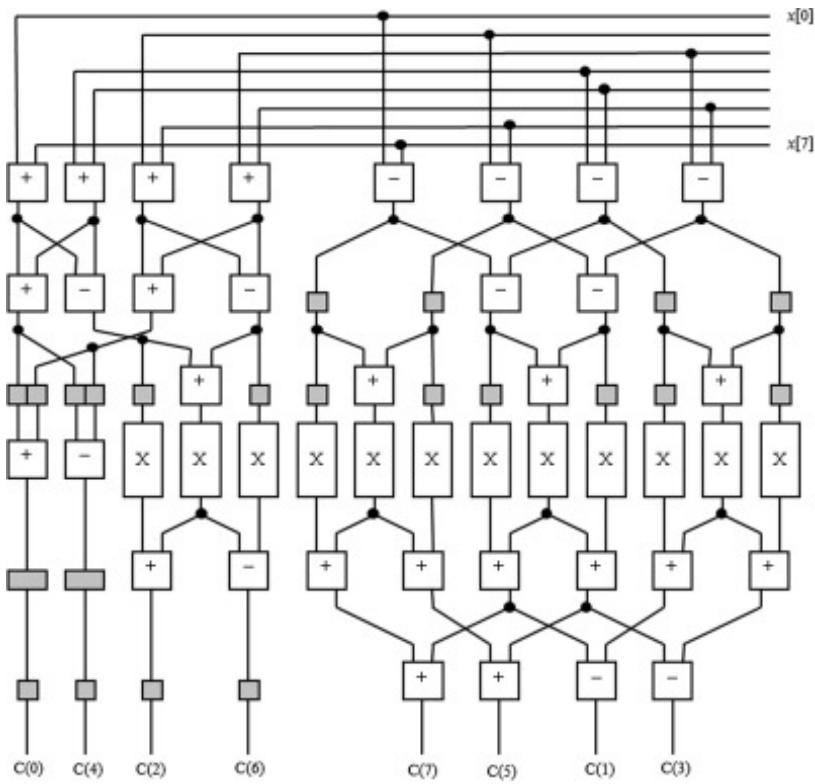
其中：

$$\begin{aligned} g_0 &= x[0] + x[1] + x[2] + x[3] + x[4] + x[5] + x[6] + x[7] \\ g_1 &= x[0] + x[7] + x[3] + x[4] - x[1] - x[6] - x[2] - x[5] \\ g_2 &= x[1] + x[6] - x[2] - x[5] \\ g_3 &= x[0] + x[7] - x[3] - x[4] \\ g_4 &= x[0] - x[7] \\ g_5 &= x[6] - x[1] \\ g_6 &= x[3] - x[4] \\ g_7 &= x[2] - x[5] \end{aligned}$$

这将所需的乘法次数从64减少到22。使用标准旋转器产品，这个数字可以进一步减少到14个：

$$\begin{bmatrix} x & w \\ w & -y \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} (x-w)r_0 \\ w(r_0+r_1) \\ (w+y)r_1 \end{bmatrix}$$

如果我们除以12乘法，还会进一步减少到12乘法  $a_4$ 。该算法的位序列架构如图5.18[5]所示，其中灰色框表示同步延迟运算符。



[下载：下载全尺寸图像](#)

图5.18。高效的DCT实现：麦戈文算法

### 5.8.3。交错序列的字段与帧编码

在交错图像序列的情况下，如果在场扫描之间发生运动，则在高空间频率下可能发生大值虚假DCT系数。这些将在压缩过程中粗量化，从而降低编码序列的质量。自MPEG-2以来，标准支持在每个字段上独立或组合帧执行DCT的选项。对于字段图片，每个宏块中的所有块都来自一个字段，而对于（交错的）帧图片、帧或字段，DCT编码决策可以在一个宏块的基础上自适应地进行。

当隔行帧图像的交错宏块被帧DCT编码时，其四个块中的每个块都有来自两个字段的像素。然而，如果隔行帧图片的交错宏块被字段编码，则每个块仅由两个字段之一的像素组成。

马丁和布尔[6]使用布尔和霍罗克斯的原始运算符方法产生了一种高效的算法[7]。这支持编码任意一个 $8 \times 8$ DCT或两个独立的 $4 \times 4$ DCT块。将结果与使用面积时间指标的替代实现进行了比较，并观察到显著的复杂性节约。

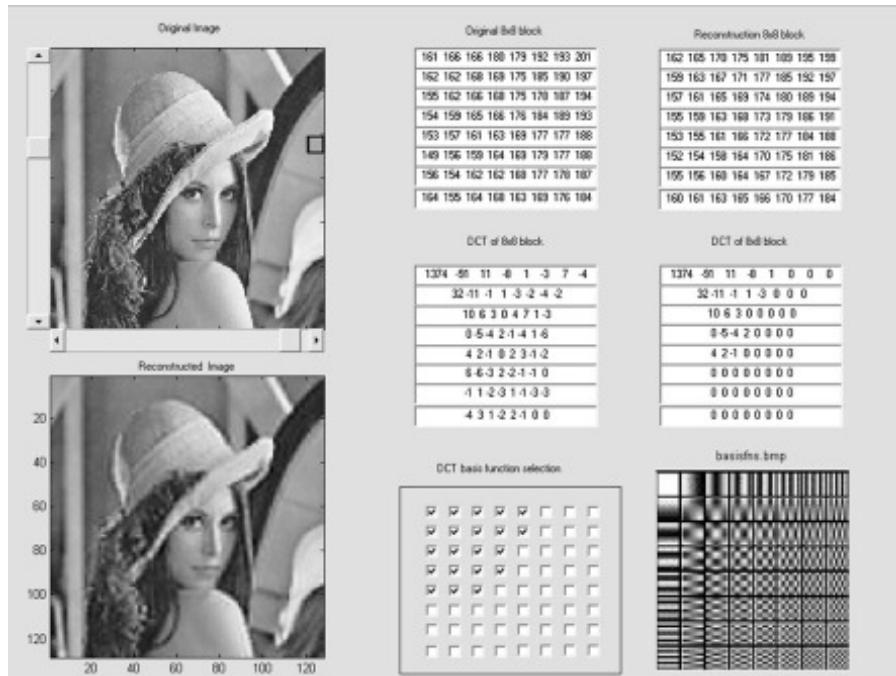
### 5.8.4.整数变换

虽然DCT原则上是无损的，但它可能会因编码器和解码器之间的数值不匹配而导致漂移。这可以通过使用整数变换来克服，其中使用整数算术来确保无漂移实现。这种方法用于基本的H.264/AVC  $4 \times 4$ 整数变换本身来自 $4 \times 4$ DCT。这个变换在方程 (5.49) 中给出，并在第9章中进一步讨论：

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad (5.49)$$

### 5.8.5.DCT演示

图5.19显示了信息丰富的DCT演示的图形用户界面。这个简单的Matlab®软件为用户提供了可视化选择和取消选择DCT系数效果的机会，显示它们对量化DCT矩阵和重建图像的影响。读者可以从<http://www.bristol.ac.uk/vi-lab/demos>下载此软件包，作为综合演示包的一部分。



[下载：下载全尺寸图像](#)

图5.19。DCT演示GUI。

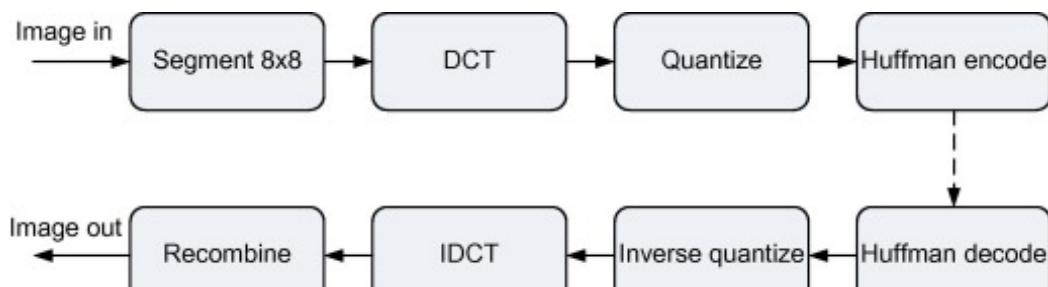
## 5.9。JPEG

JPEG标准最初创建于1992年[8]，现在仍然无处不在。尽管JPEG2000在2000年成为性能卓越的竞争者，但它保持了其作为互联网和消费者使用的主要编解码器的地位。JPEG有四种主要模式：

- 基线（顺序DCT + Huffman编码）。
- 顺序无损（预测编码加霍夫曼或算术编码）。
- 渐进式DCT（位平面渐进式）。
- 层次分层次的。

基线JPEG基于一个简单的基于块的DCCT，具有**标量量化**和霍夫曼编码。适用于每个样本最多8

位的**灰度和彩色图像**, 如图5.20所示。在描述熵编码后, 我们将在**第7章**中更详细地检查完整的JPEG编码过程。



[下载：下载全尺寸图像](#)

图5.20。JPEG基线系统图。

## 5.10。摘要

在本章中, 我们向读者介绍了转换编码, 解释了其基本概念和数学基础。我们已证明, 最佳变换虽然为给定数据集提供了最佳**能量压缩**, 但当数据统计是非平稳的时, 则会受到影响, 因为需要不断更新变换矩阵并将其传输到解码器。替代变换, 如DCT, 为与固定基的相关图像提供了近乎最佳的性能, 是实际图像和视频压缩的首选。我们推导了DCT, 并展示了其出色的性能和低复杂性的实现。

在下一章中, 我们将探索另一种基于小波**滤波库**的数据转换, 在**第7章**中, 我们将描述无损编码, 并展示装饰相关变换如何与熵(符号)编码相结合, 以实现令人印象深刻的压缩比, 同时保持出色的感知图像质量。

[Recommended articles](#)

Citing articles (0)

## 参考文献

- [1] R.克拉克  
**图像的转换编码**  
学术出版社 (1985)  
[谷歌学术](#)
- [2] N.艾哈迈德, T. 纳塔拉根, K. 拉奥  
**离散余弦变换**  
IEEE 计算机交易, 23 (1974), 第90-93  
[CrossRef](#) [在Scopus中查看记录](#) [谷歌学术](#)
- [3] B.李  
**一种计算离散余弦变换的新算法**

[4] W.-H.陈, C. 史密斯, S. 弗拉利克

一种离散余弦变换的快速计算算法

IEEE 通信交易, 25 (1977), 第 1004-1009 页

[在Scopus中查看记录](#) [谷歌学术](#)

[5] F. 麦戈文, R. 伍兹, M. 严

新型超大规模集成电路的实施( $8 \times 8$ )点二维DCT

电子信件, 30 (8) (1994年), p.624-626

[在Scopus中查看记录](#) [谷歌学术](#)

[6] F. 马丁, D. 公牛

自适应离散余弦变换的改进架构

IEEE电路和系统国际研讨会 (1996年), p.742-745

[在Scopus中查看记录](#) [谷歌学术](#)

[7] D.公牛, D. 霍罗克斯

原始操作员数字滤波器

IEE Proceedings G (电路、设备和系统), 138 (3) (1991年), p.401-412

[CrossRef](#) [在Scopus中查看记录](#) [谷歌学术](#)

[8] ISO/IEC国际标准10918-1, 信息技术-连续色调静止图像要求和指南的数字和编码, 1992年。

[谷歌学术](#)

\* 有关图5.14、图5.15、图5.16的彩色版本, 请参阅电子版或网站。

版权所有©2014爱思唯尔有限公司。保留一切权利。



[关于ScienceDirect](#)

[远程访问](#)

[购物车](#)

[广告](#)

[联系和支持](#)

[条款和条件](#)

[隐私政策](#)

RELX™

我们使用cookie来帮助提供和增强我们的服务，并定制内容和广告。继续即表示您同意**使用cookie**。  
版权所有?2021爱思唯尔B.V.或其许可方或贡献者。ScienceDirect ®是爱思唯尔B.V.的注册商标。  
ScienceDirect ®是爱思唯尔B.V.的注册商标。