文章编号：1001-2486（2002）02-0069-04

整数 DCT（BinDCT）快速算法*

朱桂华

（常德师范学院计算机系，湖南 常德 415000）

摘 要：设计和实现了基于提升结构的无乘法二进制 DCT（BinDCT）的快速算法。该算法是在基于旋转变换的递归算法基础上设计的，并将传统的旋转变换的 3 个提升矩阵乘积减少至 2 个提升矩阵乘积，从而使算法的运算量比现有算法大为减少。

关键词：离散余弦变换；二进制离散余弦变换；提升结构

中图分类号：TN957.52 文献标识码：A

Fast Algorithm for Integer DCT（BinDCT）

ZHU Gui-hua

（Department of Computer，Changde Teacher's College，Changde 415000，China）

Abstract：In this paper，we design and implement fast multiplierless approximations of the discrete cosine transform（DCT）with the lifting scheme，named the BinDCT，which avoids floating-point operations. This algorithm is derived from the plane rotation－based algorithm，which reduces 3 lifting steps to 2 lifting steps via the plane rotation. So the arithmetic operation cost is reduced greatly.

Key words：DCT；BinDCT；lifting scheme

离散余弦变换（DCT）具有非常好的能量压缩性能，因而在数字语音及图像处理等领域有着广泛的应用，现已成为 JPEG、H.26X 和 MPEG 上许多国际标准的核心。理论上已经证明，一维 8 点 DCT 所需的乘法与加法数量至少是 11 和 29 个。然而，在图像和视频处理中，量化阶段通常需要压缩数据，所以可以将 DCT 的某些运算放到量化阶段处理，从而能降低算法的运算量，这就产生了所谓的尺度 DCT（Scaled DCT）。此时所有的快速 DCT 算法仍需要用到浮点数的乘法，一定程度上降低了计算机软、硬件实现浮点 DCT 的速度。为了获得更快的速度，许多算法都利用整数进行近似，用整数乘法来替代浮点乘法，从而建立速度较浮点算法大大提高的整数 DCT 算法。1999 年后发展起来的二进制 DCT 利用提升结构消去了乘法[4]，只用移位和加法来实现，得到基于提升结构的无乘法 DCT。提升结构不仅能实现灵活的正交变换，而且还能达到无损压缩效果，因此在变换编码中是一个非常有用的工具。本文在文献［4］的基础上加以改进，提出了一种新型 BinDCT 算法。基本思想与 Tran[2] 的 BinDCT 方法一致，但将 8 点 BinDCT 推广为任意长度的 BinDCT。

1 8 点 DCT 算法描述

为简单起见，用图 1 表述 8 点 DCT 的算法过程。从图 1 可以看出 8 点 DCT 中浮点乘法与加法不可避免。下面讨论用移位和加法实现的整数 DCT 算法。为方便起见，不考虑变换因子$\sqrt{\frac{2}{N}}$，因此逆变换时应乘以因子 $2/N$。

为了建立整数 DCT 算法，首先看如下的等式：

$$\begin{bmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{bmatrix}=\begin{bmatrix}1 & \alpha\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ \alpha & 1\end{bmatrix}\begin{bmatrix}1 & b\\ 0 & 1\end{bmatrix} \tag{1}$$

解得 $a=\sin\alpha$，$b=-\tan\frac{\alpha}{2}$。因此有下面结论：一个旋转变换可以表示成 3 个提升矩阵的乘积，即

* 收稿日期：2001-10-10
基本项目：国家自然科学基金资助项目（10171109）
作者简介：朱桂华（1964—），女，讲师。

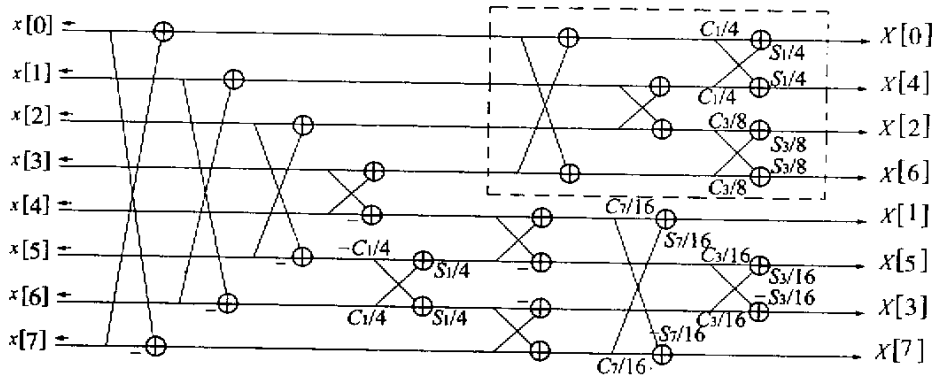


图1 基于旋转变换的8点DCT的算法流程图

Fig.1 Butterfly graph of length 8 DCT via rotations

$$\begin{bmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{bmatrix}=\begin{bmatrix}1 & -\tan\frac{\alpha}{2}\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ \sin\alpha & 1\end{bmatrix}\begin{bmatrix}1 & -\tan\frac{\alpha}{2}\\ 0 & 1\end{bmatrix} \tag{2}$$

由式(2)不难看出提升矩阵有如下特点：

(1)一个提升矩阵的逆矩阵仍然是一个提升矩阵，且逆变换避免了除法；

(2)对提升因子取整或取近似，逆变换得到的值与原始值没有差别。(如果 x **(** i **)为整数)，即对于运算** $y(i)=x(i)+[ax(j)]$，$y(k)=x(k)$，$k\neq i$**(本文中[]表示取整或取近似)，其逆运算为** $x(i)=y(i)-[ay(j)]$，$x(k)=y(k)$，$k\neq i$ **，可以做到逆变换得到的值与原始值没有差别。**

利用提升矩阵的第2条性质，可以对提升因子 a **和** b **取近似为** $\frac{c}{2^k}$ **(** c，$k\in Z$**)，则提升结构就可以实现无乘法运算，完全由移位和加法来实现。例如，** $\frac{3}{16}=\frac{1}{8}+\frac{1}{16}$**，提升因子** $\frac{3}{16}$ **就可用2个移位和1个加法来实现。**

由于1个平面旋转变换可以表示成3个提升矩阵的乘积，为了减少运算量，将旋转矩阵表示为

$$\begin{bmatrix}\cos\alpha & \sin\alpha\\ -\sin\alpha & \cos\alpha\end{bmatrix}=\begin{bmatrix}\cos\alpha & 0\\ 0 & 1/\cos\alpha\end{bmatrix}\begin{bmatrix}1 & 0\\ -\sin\alpha\cos\alpha & 1/\cos\alpha\end{bmatrix}\begin{bmatrix}1 & \tan\alpha\\ 0 & 1\end{bmatrix} \tag{3}$$

2 任意长度的BinDCT算法设计

建立一种只需要移位和加法来实现的任意长度的 BinDCT－Ⅱ **算法(** $N=2^t$，$t\geqslant 3$**)，算法过程描述如下(对应流程图见图2)：**

算法1：BinDCT－Ⅱ **的算法：**

step1：$g(n)=x(n)+x(N-n-1)$，$h(n)=x(n)-x(N-n-1)$，$n=0,1,\cdots,N/2-1$

step2：**对** $g(n)$**计算** *BinDCT*－Ⅱ **的** $C^{\mathrm{II}}_{N/2}$**，得** $G(n)$ **，**$n=0,1,\cdots,N/2-1$**，对** $h(n)$**计算** BinDCT－Ⅳ **的** $C^{\mathrm{IV}}_{N/2}$**，得** $G(n)$ **，**$n=N/2,\cdots,N-1$**；**

step3：**对** $\{G(0),G(1),\cdots,G(N-1)\}$**进行重排，得** $X(k)=G(2k)$，$X(N/2+k)=G(2k+1)$**。其中，2点** BinDCT－Ⅱ **为**

$$\begin{bmatrix}X(0)\\ X(1)\end{bmatrix}=\begin{bmatrix}\cos\frac{\pi}{4} & \sin\frac{\pi}{4}\\ -\sin\frac{\pi}{4} & \cos\frac{\pi}{4}\end{bmatrix}\begin{bmatrix}x(0)\\ x(1)\end{bmatrix}\approx\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ \frac{1}{2} & -1\end{bmatrix}\begin{bmatrix}x(0)\\ x(1)\end{bmatrix} \tag{4}$$

算法2(BinDCT－Ⅳ **的算法)：**

step1：**按照** $\begin{bmatrix}y(2k-1)\\ y(2k)\end{bmatrix}=\overline{R}_{\frac{\pi}{4}}\begin{bmatrix}x(2k-1)\\ x(2k)\end{bmatrix}$ **(**$k=1$，2，$\cdots$，$N/2-1$**)作提升变换(3个提升步)，**

$y(0)=x(0)$，$y(N-1)=x(N-1)$；

step2 对 $y(k)$作序列重排，得 $\begin{cases} p(k)=y(2k), k=0,1,\cdots,N/2-1 \\ p(N/2+k)=y(N-2k-1), k=0,1,\cdots,N/2-1 \end{cases}$

然后分别计算 $p(k), k=0,1,\cdots,N/2-1$ 与 $p(k), k=N/2,\cdots,N-1$ 的 $N/2$ 的 BinDCT－Ⅲ（即 $N/2$ 的 BinDCT－Ⅱ的逆），设其输出为 $P(k), k=0,1\cdots N/2-1$ 与 $P(k), k=N/2,\cdots N-1$；

step3 对 $P(k)$的后半步作重排，得

$[p(N/2),p(N/2+1),\cdots,p(N-1)]'=[-p(N-1),p(N-2),-p(N-3),\cdots,p(N/2)]'$

step4：$[X(0),\cdots,X(N-1)]'=\overline{T}[P(0),\cdots,P(N-1)]'$作提升变换（若 N 是递归步的 N，则用 3 个提升步；若 N 是初始的 N，则用 2 个提升步）。

BinDCT－Ⅱ，BinDCT－Ⅳ的算法流程如图 2 所示。

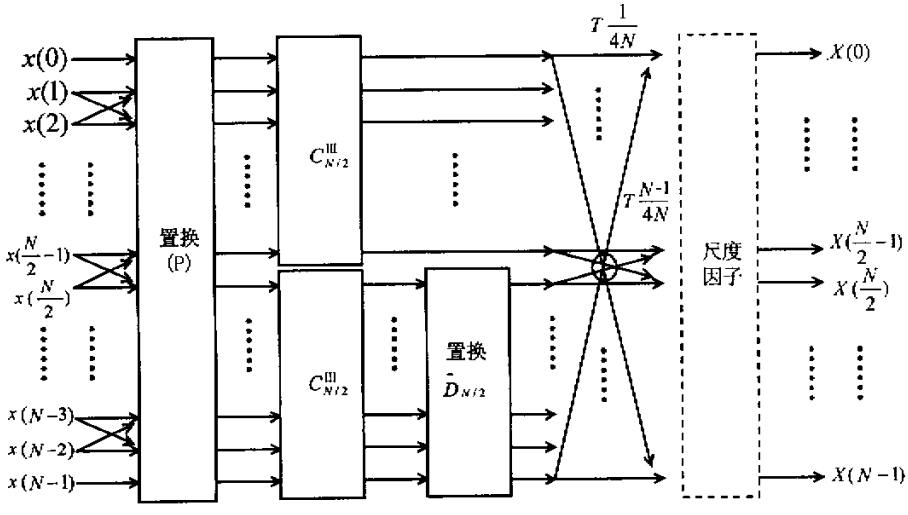


图 2　N 点 BinDCT－Ⅳ算法（$N=2^t$，$t\geqslant 3$）

Fig.2　BinDCT-Ⅳ algorithm of N points（$N=2^t$，t≥3），）

图 3 给出 8 点 BinDCT 的整数实现流程。

图 3 与图 1 相比可以看出，正变换的输出顺序有所改变，X［2］与 X［6］互换，X［1］与X［7］互换。比较一下 X［1］与 X［7］换与不换的频率响应效果的差异，不难看出不换时频率与真实的相差很大，而交换顺序后基本一致。

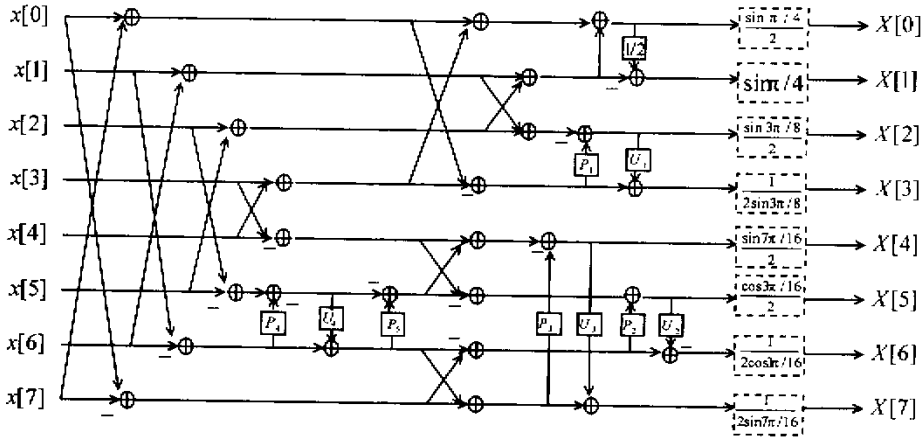
3　结　论

（1）正逆变换都仅通过移位和加法来实现，是一种高效、低成本快速算法；

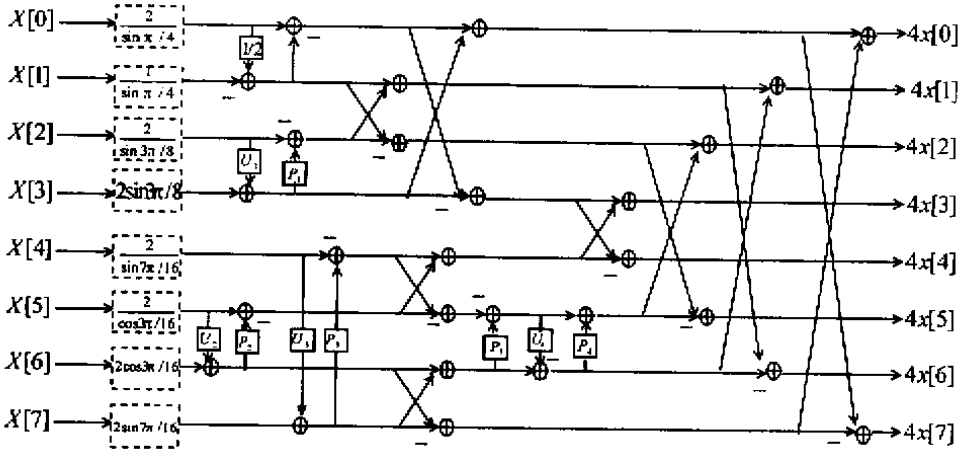
（2）尺度 DCT（scaled DCT）的思想被应用时可以进一步降低 binDCT 的计算复杂度；

（3）BinDCT 继承了所有其它 DCT 快速算法的优点。如较高的编码量（coding gain），对称基础函数，递归结构等；

（4）当长度为 8 时，算法与 Tran 的算法一致[6]，但克服了只局限于 8 点的缺陷。



(a) 正变换



(b) 逆变换

图3 8点 BinDCT－Ⅱ的正逆变换的算法流程图

Fig.3 Total BinDCT－Ⅱ algorithm for 8 points

参考文献：

[1] Ahmed N，Natand T，Rao K R. Discrete cosine transform [J]. IEEE Trans. Comput.，C－23：90－93，1974.

[2] Rao K R，Yip P. Discrete Cosine Transform [M] Academeic Press，New York，1990.

[3] MPEG Software Simulation Group (MSSG) [R]. MPEG2 Encoder/Decoder，Version 1.2 (Online)，Available HTTP：http：//www. mpeg. org/MSSG/，Jun. 1996.

[4] Chen Y J. Integer discrete consine transform (IntDCT) [C]. Invited paper，The second Int. Conf. Inform. Comm. And Signal Processing，1999.

[5] Cheng Li zhi，Xu Hui，Luo Yong. Integer discrete cosine transform (IntDCT) and its fast algorithm [J]. IEE Electron. Lett.，37 (1)，2001.

[6] Tran T D. A fast Multiplierless block transform for image and video comjpression [C]. Proc. of the IEEE ICIP－1999，3：822－826，1999.