西安电子科技大学

硕士学位论文

H.264视频编码标准的关键技术研究

姓名：倪伟

申请学位级别：硕士

专业：模式识别与智能系统

指导教师：郭宝龙

20040101

摘　要

H.264/AVC 是由国际电信联盟（ITU）和国际标准化组织（ISO）共同制定的新一代视频编码标准。该标准采用了一系列先进编码技术，在编码效率、网络适应性等诸多方面都超越以往的视频编码标准，代表了未来多媒体数据压缩编码的发展趋势。及时跟踪和掌握 H.264/AVC 的核心技术，并结合实际应用在某些关键方向上有所创新和发展，是一项很有价值的工作。

本文深入剖析了 H.264/AVC 的编码策略和技术特性，并重点研究其中的 3 项关键技术：整数变换，帧内预测和运动估计。首先对 H.264/AVC 所使用的整数变换算法和相关的量化过程进行了系统的推导和论证，并与传统的变换算法对比分析了其优点。随后系统研究了编码算法中的帧内预测技术，针对 H.264/AVC 的频域帧内预测算法，首次揭示了关于预测选项的两条一般性规律，并提出了“相关性因子”的概念用以判断相邻块预测选项；在此基础上提出了一种自适应空域帧内预测算法（简称 ASIP），能够有效降低帧内预测的计算复杂度。最后对视频编码中的运动估计算法进行研究，回顾了经典的运动估计算法，在此基础上提出了一种六边形运动矢量场自适应搜索算法，简称 HMVFAST。实验结果表明，和目前的运动估计算法相比，HMVFAST 在速度和精度方面都具有良好的性能。

关键词：视频编码 H.264/AVC 帧内预测　整数变换 运动估计

Abastract

The emerging H.264/AVC video coding standard was developed collaboratively by the ITU-T and ISO/IEC. By adopting a number of new coding techniques, H.264/AVC has achieved a significant improvement in compression performance and a "network-friendly" video representation relative to existing standards.

In this dissertation, the author mainly research on the new techniques in H.264/AVC, emphatically on 3 core techniques: integer transform, intra prediction and motion estimation.

First the author describes the systematic approach to design the integer transform and quantization in H.264/AVC. Unlike the popular 8×8 DCT used in previous standards, the 4×4 integer transform can be computed without multiplications, just additions and shifts in 16-bit arithmetic, thus minimizing computational complexity and avoiding inverse transform mismatch.

Intra prediction can improve the coding efficiency of I picture effectively. In order to improve the speed of intra prediction in H.264/AVC, we develop a new intra prediction algorithm called Adaptive Spatial-Domain Intra Prediction Algorithm (ASIP). Experimental results show that the proposed algorithm has a better speed performance comparing with the original algorithm.

Finally, the author discusses another key technique in video coding standard: motion estimation. Based on some typical block motion estimation algorithms, a novel algorithm called Hexagon based Motion Estimation Algorithm using Feature Adaptive Search Technique(HMVFAST) was proposed. It is composed of the prediction of initial search point, adaptive search modes between Hexagon Search and Small Diamond Search for video sequence with different motion, and an effective half-stop criteria. Experiments show that HMVFAST could obtain good PSNR performance as well as low computation cost.

Keyword: Video Coding, H.264/AVC, Intra Prediction, Integer Transform, Motion Estimation

第一章 绪论

1.1 问题的提出

21 世纪的人类社会是信息化社会，随着多媒体与网络技术的飞速发展，以前制约图像通信发展的各种因素正在逐步消失，视频、图像、计算机视觉、多媒体数据库和计算机网络技术日益融合，已经渗入到国民经济和社会生活的各个方面。

然而，数字化的视频信息在满足人们需求的同时，对数据传输带宽、数据存储容量提出了更高的要求。例如，一幅中等分辨率（NTSC 制式，24bits/pixel）的彩色数字视频，其传送速率约为 221.1Mbps；而高清晰度电视 HDTV 的传输速率则在 1.2Gbps 以上。庞大的视频数据给信息的存储和传输都造成了较大的困难，成为阻碍人类有效的获取和使用信息的瓶颈问题之一。

早在上个世纪四十年代末期，人们就已开始着手图象压缩编码技术的研究，以期达到有效的数据压缩，至今已经走过了近半个世纪的发展历程。从五、六十年代基本方法的探讨，到七十年代早期可视电话的研究，使得这一领域有了长足的进展，许多基本的思想和方法都相继被提出。到八十年代前后，顺应信息化潮流，面向各种应用的开发研究大力开展起来。进入九十年代以后，国际上致力于标准化的工作，先后制定了一系列视频图象编码标准，如用于视频存储和传输的 MPEG-1[1]、MPEG-2[2]、MPEG-4[3]标准，用于视频会议和可视电话的 H.261[4]、H.263[5]、H.264/AVC[6]等。这些视频压缩编码标准的制定，同时也极大地促进了视频压缩编码技术和多媒体通信技术的发展。

1.2 视频编码发展概况

1.2.1 视频编码技术

视频图象数据具有多种冗余特性，如空间冗余、时间冗余、心理视觉冗余和熵编码冗余等。视频压缩编码就是针对视频数据的一种或几种冗余特性采用相应的方法加以消除，只保留相互独立的信息分量，从而达到令人满意的数据压缩的目的。

视频压缩发展到现在已有几十年的历史。1948 年，Oliver 提出了第一个编码理论——脉冲编码调制（Pulse Coding Modulation，简称 PCM）；同年，Shannon 的经典论文——“通信的数学原理”首次提出了信息率失真函数的概念；1959 年，Shannon 进一步确立了码率失真理论；而 Berger 在 1971 年所著的《信息率失真理

论》一书则对率失真理论理论做了系统地论述和扩展；以上各项工作奠定了信息编码的理论基础。Shannon 的信息论具有高度概括性和综合性，在实践中得到了广泛的应用。可以说，整个压缩编码的历史就是以 Shannon 信息论为出发点，不断克服其缺陷的过程。

“第一代”视频压缩编码技术正是以 Shannon 信息论为基础，主要有预测编码、变换编码和统计编码这三大经典编码方法。它们都是非常优秀的纹理编码方案，能够在中等压缩率的情况下，提供非常好的图像质量。“第一代”编码技术在 20 世纪 80 年代已趋于成熟，这类技术去除客观和视觉冗余信息的能力已接近极限，许多优秀成果已被吸收近年来所制定的图像/视频压缩标准中。“第一代”编码技术只是以信息论和数字信号处理技术为理论基础，其压缩比普遍不高，大约在 10:1 左右。当需要进行低码率的图像数据压缩时，往往无法提供令人满意的质量。究其原因是由于这些技术都没有利用图像的结构特点，因而也就只能以象素或块作为编码的对象；另一方面，这些技术在设计编码器时也没有考虑人类视觉系统的特性。

“第二代”图像压缩编码技术这一术语是在 20 世纪 80 年代中期正式出现的。它突破了原有信息论的框架，充分利用了计算机图形学、计算机视觉、人工智能与模式识别等相关学科的研究成果，如人的视觉生理、心理和图像信源的各种特征，实现从“波形”编码到“模型”编码的转变，为视频图像压缩编码开拓出了广阔的前景。“第二代”编码方法主要有：基于分形的编码、基于模型的编码、基于区域分割的编码和基于神经网络的编码等，其压缩比多在 30:1 至 70:1 之间，有的甚至高达 100:1。但是由于“第二代”编码方法大大增加了实现的复杂度，从当前发展情况来看，“第二代”编码方法仍处于深入研究的阶段。例如，分形法由于图像分割、迭代函数系统代码的获得是非常困难的，因而实现起来时间长，算法非常复杂。模型法则仅限于人头肩像等基本的视频图像上，进一步的发展有赖于新的数学方法和其它相关学科的发展。神经网络的工作机理至今仍不清楚，硬件研制不成功，所以在视频压缩编码中的应用研究进展缓慢，目前多与其他方法结合使用。但由于其巨大的潜力，人们都在致力于这些新方法的研究。

近年来，出现了一类充分利用人类视觉特性的“多分辨率编码”方法，如子带编码、塔形编码和基于小波变换的编码。这类方法使用不同类型的一维或二维线性数字滤波器，对视频图像进行整体的分解，然后根据人类视觉特性对不同频段的数据进行粗细不同的量化处理，以达到更好的压缩效果。这类方法原理上仍属于线性处理，属于“波形”编码，可归入经典编码方法，但它们又充分利用了人类视觉系统的特性，因此可以被看作是“第一代”编码技术向“第二代”编码技术过渡的桥梁。

1.2.2 视频编码的国际标准

标准化是产业化活动的前提，一项技术在能够广泛应用于工业生活之前必须存在全球统一的工业标准。近年来，一系列国际视频编码标准的制定，极大地促进了视频压缩编码技术和多媒体通信技术的发展。视频压缩编码标准的制定工作主要是由国际标准化组织（International Standardization Organization，ISO）和国际电信联盟（International Telecommunication Union，ITU）完成的。由 ITU 组织制定的标准主要是针对实时视频通讯的应用，如视频会议和可视电话等，它们以 H.26x 命名；而由 ISO 和 IEC（International Electrotechnical Commission，国际电工委员会）的共同委员会中的 MPEG 组织（Moving Picture Expert Group）制定的标准主要针对视频数据的存储（如 VCD 和 DVD），广播电视和视频流的网络传输等应用，它们以 MEPG-x 命名。各种视频压缩编码标准都是根据人们在不同领域中对声像数据的要求所制定的，并且随着人们的需求不断地发展。图 1.1 按制定时间的顺序表述了视频编码标准的发展历程。

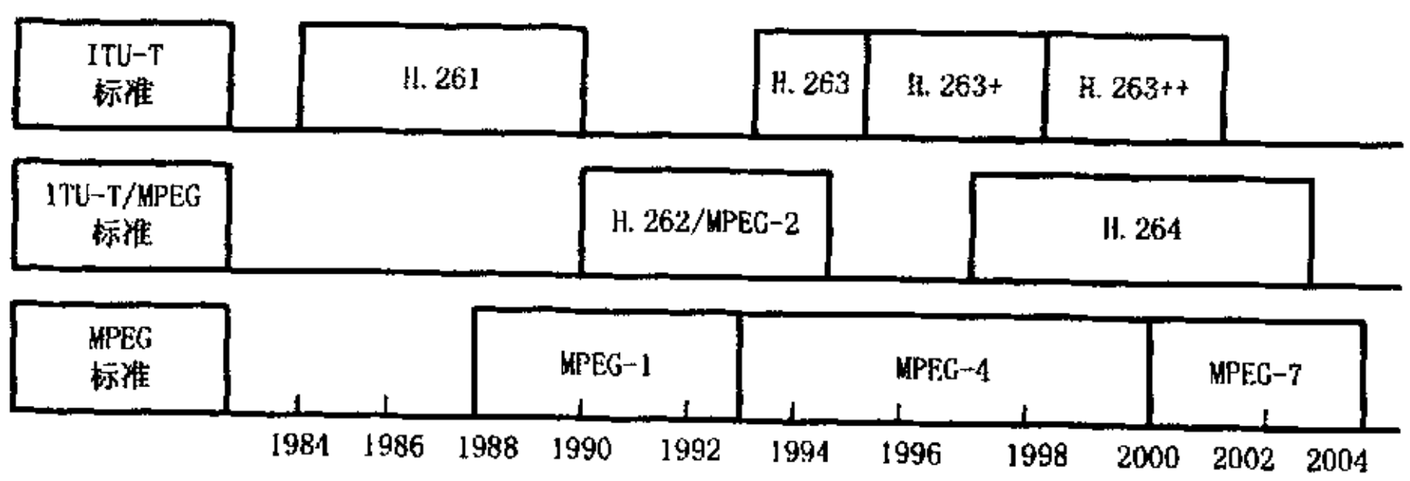


图 1.1 视频编码国际标准的发展

在 ITU 和 ISO/IEC 所制定的这些标准中，并没有对视频编码的具体算法做硬性规定，而仅仅是定义了相应的解码方法和比特流语法，使得对于符合某一标准的压缩码流，所有的解码器都能够得到相同的输出结果，这也为标准的具体应用带来了最大限度的自由度。下面我们就对几种典型的国际视频编码标准做简要介绍。

一、MPEG 系列标准

1. MPEG-1

MPEG-1制订于1993年，是针对数据传输率1.5Mbps以下的数字存储介质图像及其伴音编码的国际标准，共分为图像编码、声音编码和系统（同步和复用）3个部分。该标准主要用于在各种数字存储介质（CD-ROM、DAT、Winchester盘等）上存储同步和彩色运动视频信号，在1.2Mbps速率下的视频质量可与VHS（家用视频系统）所记录的模拟视频质量相媲美。MPEG-1对色差分量采用4:1:1的二次采样率，可优化为中等分辨率，并在优化的模式下采用所谓的标准交换格式(SIF)，其视频压缩率约为26:1。MPEG-1标准采用了运动估计/运动补偿、变换编码等技术，并规

定了编码位流的表示语法和具体解码方法。由于MPEG-1标准是针对数字存储的应用而制定的，因此它的编解码器是不对称的，其编码端的复杂度通常要远远高于解码端。

2. MPEG-2 (H.262)

MPEG-2标准是由ISO的MPEG专家组和ITU-T的第15研究组与1994年共同制定的，全称为“运动图像及其伴音的编码”，在ITU-T的协议中也被称为H.262建议（Recommendation H.262）。MPEG-2的传输速率为3Mbps~10Mbps，主要针对数字电视和高清晰度电视(HDTV)所需要的视频及伴音信号，此外还兼顾了与ATM信元的适配问题。

MPEG-2在MPEG-1的基础上做了相应的扩展，从多方面提高了编码参数的灵活性以及编码性能。它综合采用了运动补偿的帧间预测、空间域离散余弦变换、自适应量化和可变长编码的混合编码。MPEG-2视频编码标准是一个分等级的系列，按编码图像的分辨率分成4个等级(Levels)；按所使用的编码工具的集合分成五个类别(Profiles)。“等级”与“类别”的若干组合构成MPEG-2视频编码标准在某种特定应用下的子集，对某一输入格式的图像，采用特定集合的压缩编码工具，产生规定速率范围内的编码码流。目前MPEG-2标准已经在DVD存储和数字电视广播方面得到了广泛应用。

3. MPEG-4

在MPEG-1和MPEG-2之后，ISO的MPEG工作组于1999 年4月出台了MPEG-4标准（ISO14496），并在1999 年12 月提出了第二版的ISO最终草案(ISO14496-2 FCD)。MPEG-4 提出了音视频对象（Audio Video Object, AVO）的概念，并在此基础上实现了许多新的功能，为各种多媒体应用特别是基于Internet 和移动网络的应用提供了理想的工具，如基于内容的编码、错误掩盖和基于内容的可伸缩性等。与MPEG前两个图像压缩标准相比，MPEG-4 为多媒体数据压缩提供了一个更为广阔的平台，更注重于定义一种格式和框架，而不是具体的算法，其出发点就是希望建立起一个更自由的通信与研发环境，可以在系统中加入许多新的算法，为使用计算机软件实现编码和解码提供更大的方便。它可以将各种各样的多媒体技术充分应用于编码中，除包括压缩本身的一些工具、算法外，还包括图像分析和合成、计算机视觉、计算机图形学、虚拟现实和语音合成技术。

4. MPEG-7和MPEG-21

MPEG-7标准[7]称为“多媒体内容描述接口”（Multimedia Content Description Interface），目的是制定一套描述符标准，用来描述各种类型的多媒体信息及它们之间的关系，以便更快更有效的检索信息。该标准的第4版已于2000年10月发布，它通过标准化一种用来定义描述方案的语言，即描述定义语言(DDL)，使带有与之相关的MPEG-7数据的AV素材，就可以被加上索引，并可进行检索。这些媒体材

料可包括静态图像、图形、3D模型、声音、话音、电视以及在多媒体演示中它们之间的组合关系。

在MPEG-7的基础上，ISO又于1996年开始着手MPEG-21标准[8]的制定工作。MPEG-21是一个支持通过异构网络和设备使用户透明而广泛地使用多媒体资源的标准，其目标是建立一个交互的多媒体框架，该框架能够使遍布全球的各种网络和设备上的数字资源被透明和广泛的使用。

总体来说，MPEG-4和MPEG-21其应用范围业已超出了传统的传输和存储范畴，而是转向多媒体检索、交互式多媒体操作和内容管理等领域，已经不是一种单纯意义上的视频编码算法。

二、H.26x系列标准

1. H.261

H.261 建议是最早出现的视频编码国际标准，由 ITU-T 第 15 研究组为在窄带综合业务数字网（N-ISDN）上开展双向声像业务（可视电话、电视会议）而制定的。该建议于 1990 年通过，其全称为“p×64Kbit/s 视听业务的视频编解码器”，其中 $p=1\sim30$，用以根据传输线路的带宽调整图像质量。H.261 只对 CIF 和 QCIF 两种图像格式进行处理，采用的算法结合了可减少时间冗余的帧间预测和可减少空间冗余的DCT变换的混合编码方法，主要由运动估计/补偿、DCT变换和Huffman编码等部分组成。由于该建议主要针对实时业务，因而希望编解码的延时尽可能小，所以只利用前一帧做参考帧进行前向预测，且编解码器的复杂程度基本对称。

2. H.263

H.263建议是ITU-T提出的关于码率低于64Kbit/s的窄带电信信道视频编码的基本算法，于1996年正式通过。它以H.261为基础，同时吸收了MPEG等其他一些国际标准中有效合理的部分做出改进，如半像素精度的运动估计、不受限运动矢量、高级预测模式、PB帧等，使其性能优于H.261。H.263建议不仅着眼于利用PSTN（Public Switched Telephone Network，公共开关电话网络）传输，而且兼顾GSTN移动通信等无线业务，作为视频编码/解码的核心算法被广泛应用于视频电话终端如ITU-T的H.324(PSTN)、H.320(ISDN)和H.310(B-ISNR)中。

在H.263之后，ITU又相继于1998年和2000年制定了H.263+[9](H.263 v2, H.263第二版)和H.263++(H.263 v3, H.263第三版)。H.263+和H.263++是H.263标准的扩充并与之兼容，主要是在H. 263的4种可选模式的基础上又附加了新的可选模式和其他一些附加特性，目的是拓宽应用领域、提高压缩效率和错误掩盖能力。

3. H.264/AVC

H.264/AVC作为面向电视电话、电视会议的新一代编码方式，最初是由ITU组织的视频编码专家组VCEG于1998年提出的，目标是在同等图像质量条件下，新标

准的压缩效率比任何现有的视频编码标准要提高1倍以上。直到2001年底， MPEG组织也加入了ITU-T的VCEG组织，组成了联合视频专家组（Joint Video Team, JVT）共同完成制定工作。H.264/AVC标准草案于2003年3月正式获得通过。H.264仍基于经典混合编码算法的基本结构，在变换编码、熵编码和运动估计等方面采用了一系列先进技术，是视频编码技术和图像工程的最新研究成果，其性能超越了以往所有的视频编码标准，具有光明的应用前景。

1.3 研究工作概要及章节安排

1.3.1 主要研究工作

新一代视频编码标准 H.264/AVC 正在蓬勃发展，代表着未来多媒体数据压缩编码的发展趋势，及时跟踪和了解 H.264/AVC 的发展动态，掌握其核心技术，并结合实际应用在某些关键方向上有所创新和发展，是一项很有价值的工作。本文以 H.264/AVC 为中心，对视频编码的原理和各项技术进行了深入研究，重点分析了其中的 3 项关键技术：整数变换、帧内预测和运动估计。全部研究工作可以分为如下 6 个阶段：

1. 熟悉视频编码的基本思想和原理，学习编码理论和通信原理，奠定课题开展的初步基础。
2. 对原有视频图象压缩标准如 H.261、H.263、MPEG-1/2/4 等国际标准进行深入学习，对其基本思想、整体框架和关键技术等有了完整的认识。
3. 学习 H.264/AVC 标准，结合其校验模型 JM，深入剖析了 H.264/AVC 编解码的原理和关键算法，并针对 H.264/AVC 部分算法通过软件进行仿真；
4. 针对视频编码中的变换和量化算法进行研究，重点研究了 H.264/AVC 的整数变换算法，并进行了仿真和比较。
5. 针对 H.264/AVC 中的帧内预测算法进行研究，总结出 H.264/AVC 帧内预测的两条一般性规律，在此基础上提出自适应空域帧内预测算法（ASIP）；使用 VC++6.0 编写实验仿真平台加以验证。
6. 针对视频编码的另一项核心技术——运动估计算法进行研究和改进，在传统算法的技术上提出了六边形运动矢量场自适应搜索算法；使用 VC++6.0 编写出实验平台进行验证。

本文的主要贡献：

1. 详细分析了 H.264/AVC 帧内预测算法，首次揭示了关于预测选项的两条一般性规律：相邻块的预测选项具有相关性；预测选项相同的邻块 SAD 值具

有相关性，并提出了“相关性因子”的概念用以判断预测选项的相同与否。在此基础上结合“半步停”技术提出了自适应空域帧内预测算法（ASIP），实验结果表明该算法大大降低了帧内预测的计算量，是一种极为高效的预测算法。

2. 结合起始搜索点预测、运动类型判定和模板组合，提出了一种效率更高的运动估计算法：六边形运动矢量场自适应搜索算法（HMVFAST）。该算法充分利用了运动矢量的时间和空间相关性进行运动类型判定和起点预测，结合使用了效率更高的六边形模板和小菱形模板，同时对于静止块设定阈值直接中止预测，提高了块匹配的速度。与传统的块匹配算法相比，HMVFAST 在速度和精度方面都有着明显提高。
3. 深入研究了视频压缩编码算法和现行的各种视频压缩标准，重点分析 H.264/AVC 中提高编码效率的新技术，归纳总结了其所使用的一系列先进技术，如整数变换、帧内预测、CABAC、高精度、多参考帧运动估计等。并与 H.263 和 MPEG-4 进行了性能比较，证明了 H.264/AVC 是目前编码效率最高、性能最好的编码算法。
4. 分析了视频编码中 DCT 变换的典型快速算法及其优缺点，在此基础上系统推导出 H.26L/H.264 采用的整数变换方案，并对与其相关的量化过程也做了详细探讨和分析。最后通过实验证明 H.264/AVC 中整数变换和量化方案的诸多优点，指出能够完全替代传统的 DCT。

1.3.2 论文章节安排

论文全文共分为 6 章。第一章为绪论，阐述视频编码的研究背景、意义和应用现状。第二章介绍了 H.264/AVC 的编码策略和技术特征。第三章首先分析了视频编码中采用的变换算法，在此基础上针对 H.26L 和 H.264/AVC 所采用的两种整数变换算法做了系统研究。第四章中研究了视频编码标准中的两类帧内预测算法，针对 H.264 帧内预测算计算复杂度高的问题，提出了一种高效的算法：自适应空域帧内预测算法，并做了详细的描述和性能分析。第五章对视频编码中的运动估计算法进行研究探讨，提出一种六边形运动矢量场自适应搜索算法，通过实验与其它算法进行比较，证明了该算法的有效性。第六章是对本文工作的总结和未来工作的一些展望。

第二章 H.264/AVC 视频编码标准

2.1 引言

在 1995 年制定了面向可视电话、视频会议的 H.263 标准后，ITU-T 的视频编码专家组（VCEG，也称为 ITU-T SG16 Q.6）在两个方向上开展进一步的研究：短期目标（short-term goal）是为 H.263 标准增加更多的功能，产生了最终的 H.263++ 建议；长期目标（long-term goal）是建立一套全新的视频编码标准，该标准相比较于以前的编码标准，具有更高的编码效率和更好的图像质量，这就是 H.264 建议的前身。新标准于 1998 年 1 月份开始草案征集，VCEG 将其暂时命名为 H.26L。1999 年 10 月，完成第一个草案和的制定。2001 年 12 月，ISO 的 MPEG 组织和 ITU 的 VCEG 成立了联合视频专家组（Joint Video Team, JVT），共同负责 H.26L 标准的制定和实施，随后在 JVT 的第一次会议上制定了以 H.26L 为基础的 H.264 标准草案和测试模型 TML-9[10]（Test Model Long-Term Number 9）。2003 年 3 月新标准获得正式通过，ITU 将该算法命名为“H.264 建议” (Recommendation H.264)，同时 ISO 将其作为 MPEG-4 的第 10 部分，称之为“ISO/IEC 14496 Part10 高级视频编码算法”(ISO/IEC 14496 10 AVC)。图 2.1 直观的显示出了 H.264/AVC 的发展概况。为了简单起见，在下文中将 H.264/AVC 视频编码标准统称为 H.264 标准。

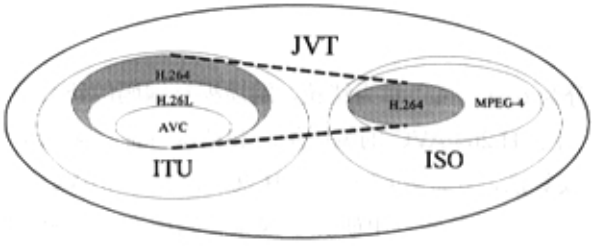


图 2.1 H.264 发展历程

H.264 吸收了以前视频编码标准中一些已经证明行之有效的算法，另外一方面又采纳了视频编码、图像处理领域的最新研究成果，在提高压缩编码效率和增强网络适应能力等方面有了质的飞跃。根据应用场合的不同，H.264 作了制定了不同的算法集和技术限定，共分为 3 个类：基类（Baseline Profile）、主类（Main Profile）和扩展类（Extended Profile），每个类下面又可以划分成不同的等级（Level）。一般来说，H.264 的应用包括但不限于下列领域：

- 基于 Cable、卫星、xDSL 等的广播传输；
- 各种光磁存储设备，如 DVD 等；

- 传统的基于 ISDN、以太网、LAN、DSL 和无线网络的视频传输；
- 各种多媒体信息服务（Multimedia Messaging Services, MMS）。

在本章中将对 H.264 的编码策略和技术特性做简要论述与分析。

2.2 H.264 编码策略

H.264 从 H.263 的基础上发展而来，采用的仍然是经典的运动补偿混合编码算法，具备良好的兼容性和可移植性。编码图像通常被分为 3 种类型：I 帧、P 帧和 B 帧。I 帧为帧内编码帧，其编码不依赖于已经编码的图像数据。P 帧为前向预测帧，B 帧为双向预测帧，编码时都需要根据已编码的帧即参考帧进行运动估计。除此之外，H.264 还定义了新的 SP 帧和 SI 帧，用以实现不同传输速率、不同图像质量码流间的快速转换以及信息丢失的快速恢复等功能。H.264 的编解码基本原理参见图 2.2，(a)为编码器框图，(b)为解码器框图。

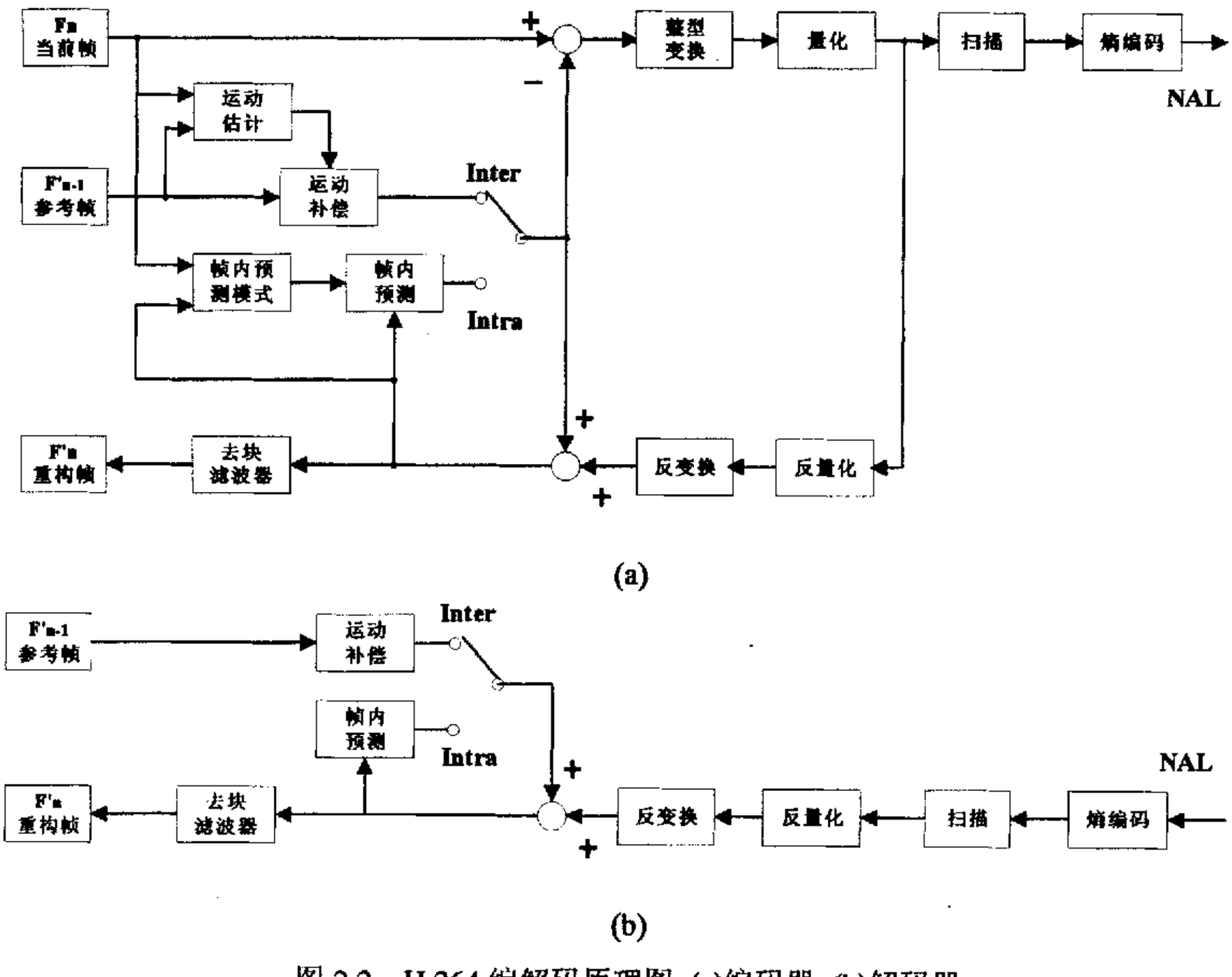


图 2.2 H.264 编解码原理图 (a)编码器 (b)解码器

1. 编码器端

编码过程中，原始数据进入编码器后，当采用帧内编码时，首先选择相应的帧内预测模式进行帧内预测；随后对实际值和预测值之间的差值进行变换、量化和熵编码，同时编码后的码流经过反量化和反变换之后重构预测残差图像，再与预测值相加得出重构帧，得出的结果经过去块滤波器平滑后送入帧存储器。

采用帧间编码时，输入的图像块首先在参考帧中进行运动估计，得到运动矢量。运动估计后的残差图像经整数变换、量化和熵编码后与运动矢量一起送入信道传输。同时另一路码流以相同的方式重构后经去块滤波后送入帧存储器作为下一帧编码的参考图像。

2. 解码器端

当编码后的码流送入解码器时，首先根据语法元素进行判断。如为帧内编码，则直接进行反量化、反变换加以重构；如果是帧间编码，所得到的为重构的残差图像，此时需要根据帧存储器中的参考图像进行运动补偿后与残差图像进行叠加，得出最终的当前帧。

2.3 H.264 提高编码效率的技术特性

H.264 采用了一系列先进的编解码技术，如帧内预测、4×4 整数变换、高精度运动估计、基于上下文的自适应二进制算术编码（CABAC）、多参考帧运动估计等，使得在同样的带宽条件下，H.264 视频图像质量超过了以往的任何编码标准。实验证明，与 H.263+和 MPEG-4 简单类(Simple Profile)相比，在视频图像质量相同的情况下，H.264 最多能够节省 50%的码率。下面就 H.264 提高视频编码效率的技术做进一步分析。

一、整数变换算法

H.264 与以前的编码标准相似，对残差图像采用基于块的变换编码。但变换是以 4×4 像素的图像块为单位，且在变换和反变换过程中只包含整数运算。H.264 在制定的过程中尝试了多种不同的整数变换算法，在最终标准草案中定义的算法是一种“分层”式的变换方案，包括如下 3 个部分：

1. 对亮度分量和色度分量的 AC 系数使用 4×4 的整数变换。
2. 对亮度分量的 DC 系数进行 4×4 的整数变换。
3. 对色度分量的 DC 系数进行 2×2 的整数变换。

H.264 整个变换方案的过程参见图 2.3。

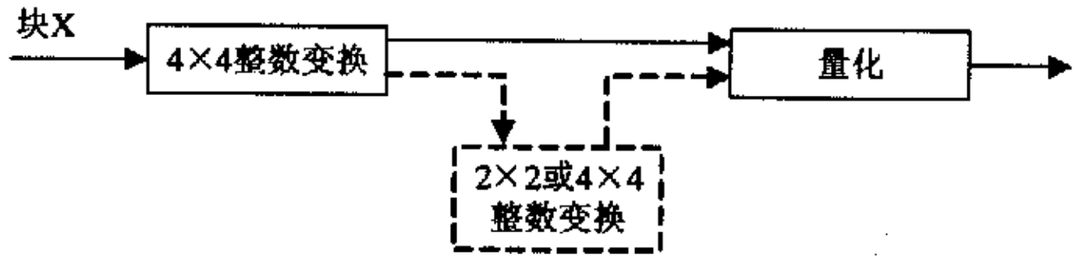


图 2.3 宏块整数变换流程

变换时首先对 4×4 的图像块进行整数变换，使用的变换和反变换核如式(2-1)所示。

$$H=\begin{bmatrix}1 & 1 & 1 & 1\\ 2 & 1 & -1 & -2\\ 1 & -1 & -1 & 1\\ 1 & -2 & 2 & -1\end{bmatrix}\qquad H_{inv}=\begin{bmatrix}1 & 1 & 1 & 1/2\\ 1 & 1/2 & -1 & -1\\ 1 & -1/2 & -1 & 1\\ 1 & -1 & 1 & -1/2\end{bmatrix}\tag{2-1}$$

2 和 1/2 只需要进行移位运算，因此在变换过程中只需要使用整数级别的算术运算即可完成。

由于图像通常是缓慢变化的，在进行了上述的 4×4 整数变换后，各块(0,0)位置的 DC 系数之间往往仍具有相关性。此时可以将亮度/色度分量的 DC 系数集中再做一次变换，进一步解除其相关性。亮度分量的 DC 系数以 4×4 为单位进行整数变换（色度分量则为 2×2），这里使用 Hadamard 变换。一维 4×4 的 Hadamard 变换核为

$$H=\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}\tag{2-2}$$

H.264 通过采用该种“分层”式的变换方案，能够进一步降低变换后 DC 系数之间的相关性。为了便于进行码率控制，变换后量化步长的变化的幅度控制在 12.5%左右，而不是以固定增幅变化。对色度系数采用了较小量化步长，使得色度分量更为逼真。

简而言之，H.264 所采用的整数变换优点在于：

(1) 变换中只使用整数算术运算，不存在反变换的误匹配问题，且降低了计算复杂度；

(2) 变换中只使用加法和移位运算，不使用乘法运算，硬件实现复杂度低；

(3) 变换中的部分运算被结合到量化过程中完成，节省了整体的运算量。

H.264 所采用的整数变换算法是视频编码标准的一大进步，也是新标准区别于其它标准的重要不同之处。在第三章中本文将详细剖析 H.264 的变换和量化过程，并给出对比实验结果以证明整数变换的优异性能。

二、帧内预测技术

在以前的视频编码标准中，大部分情况下帧内编码都采用直接做 DCT 变换、量化和熵编码的方法。在 H.263+和 MPEG-4 在编码 I 帧时采用了基于频域的帧内预测。H.264 中使用了精度更高的帧内预测算法，该算法基于空间的像素值进行预测。对于除了边缘块以外每个 4×4 块，其中的像素都可用若干个最接近的先前已编码的像素的不同加权和来预测。显然，这种帧内预测不是在时间上，而是在空间域上进行的预测编码算法，可以除去相邻块之间的空间冗余度，取得更为有效

的 I 帧压缩。本文将在第四章中对帧内预测的原理和实现方式做详细分析与研究。

三、熵编码算法

H.264 采用了两种熵编码方案，一种是可变长编码方案，包括通用可变长编码（Universal Viable Length Coding, UVLC）和基于上下文的自适应可变长编码（Context-based Viable Length Coding, CAVLC），另一种是基于上下文的自适应二进制算术编码（Context-based Adaptive Binary Arithmetic Coding, CABAC）。

1. UVLC/CAVLC

我们首先来看 VLC 的编码思想。对于一个给定的互不相同的事件集合 $e=\{e_1,e_2,...,e_n\}$ 和这些事件的概率分布 P，Shannon 证明了编码一个事件所需的最少比特数是 P 的熵 $H(P)$。

$$H(P)=\sum_{k=1}^{n}-p(e_k)\log_2 p(e_k) \tag{2-3}$$

其中 $p(e_k)$ 是事件 e_k 发生的概率，这样编码一个概率为 p 的事件最短需要 $-\log_2 p$ 位。可变长编码算法（VLC）就是基于该种对数码字长度分布，对出现概率高的符号赋以短码字，概率低的符号赋于长码字，使其编码尽可能的接近信息熵，从而实现有效的数据压缩。

UVLC 是由传统的 VLC 改进而来，它对 H.264 标准中的除变换系数外的所有语法元素，如宏块类型、帧内预测模式、运动矢量等都采用一个统一的码表（Exp-Golomb code）进行编码，编解码简单易实现。编解码过程如下。

编码过程：编码器将要编码的数字 n 按照公式(2-4)和(2-5)计算出其码字长度 L 和码字信息 INFO，再根据 L 和 INFO 得出编码码字形式。

$$L=2\times\left(\log_2\frac{n+1}{2}+1\right)+1 \tag{2-4}$$

$$INFO=n-2\times\frac{n-1}{2}+1 \tag{2-5}$$

解码过程：解码器根据接收到的码字，还原出码字长度 L 和码字信息 INFO，使用公式(2-6)即可还原出原始数据。UVLC 码表结构如表 2.1 和表 2.2 所示。

$$n=2^{\frac{1}{2}}+INFO-1 \tag{2-6}$$

表 2.1 UVLC 码字结构

1
0 x_0
0 x_1 0 x_0
0 x_2 0 x_1 0 x_0
0 x_3 0 x_2 0 x_1 0 x_0 0
…… …… …… …… …… …… …… …… …… ……

表 2.2 UVLC 码字形式

编码符号	码字
0	1
1	001
2	011
3	00001
4	00011
5	01001
6	01011
7	0000001
8	0000011
…	…

H.264 对量化后的残差变换系数使用 CAVLC 进行编码。CAVLC 的特点是：不对 EOB(End Of Block)进行编码，而是对有效系数的数目进行编码；逆序扫描系数；根据已传输的语法元素的出现概率切换到适合的码表。CAVLC 的码表经过统计优化，比单一码表更能提高编码效率。

2. CABAC

UVLC 和 CAVLC 提供了一个简单有效的编码方式，但是在中等位速率和高位速率模式下的压缩效果并不好。因此 H.264 中还定义了另一种编码算法——基于上下文的自适应二进制算术编码[11]（CABAC）。

自适应算术编码（AAC）的基本原理是将被编码的信息表示成区间(0, 1)之间的小数，信息越长，表示这一小数所需的二进制位就越多，信息源中的连续符号能够根据自适应的方法重新调整出现概率。能够对编码信息分配非整数比特使其最大的优点。

CABAC 由自适应算术编码发展而来，其编码步骤是：待编码的符号 n 首先根据符号类型选取合适的上下文模型，此模型用于确定符号 n 根据上下文信息所要编码的码字长度；如果给定的符号是非二进制值，则将其乘以 2 的倍数放大（左移），这个过程称为二进制化（Bin），用以减少表示小数的码字位数；最后使用自适应算术编码器完成熵编码。每个符号编码后，相关模型将用编码的二进制符号更新概率，使编码器模型更接近实际的统计特性。图 2.4 显示了 CABAC 的编码流程。

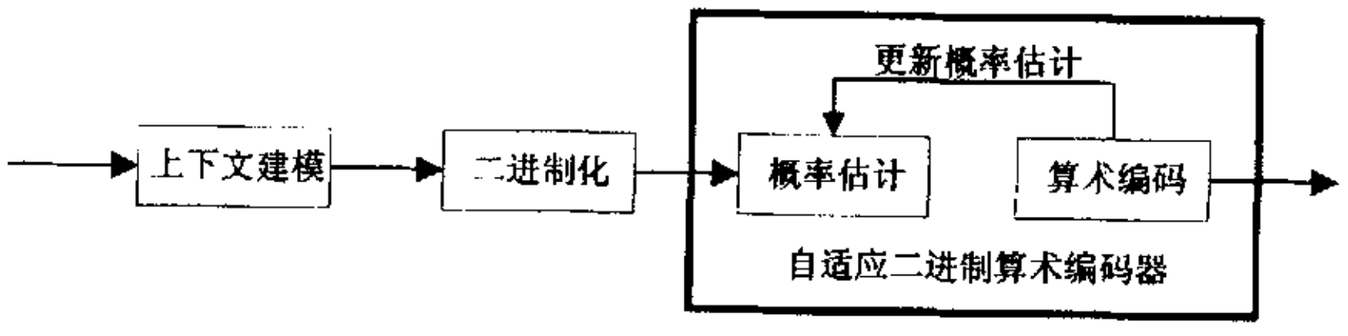


图 2.4 CABAC 编码原理图

对 CABAC 来说，上下文模型的选择可以分为两类：第一类包括语法分量的宏块类型（MB_type）、运动矢量（MVD）、参考帧（Ref_frame）；第二类包括编码块模式的语法分量（CBP）、帧间预测模式（IPRED）、（RUN, LEVEL）等信息。在开始编码前，H.264 的编码器预先给出了 126 种不同的基于上下文的模型和起始概率。编码中随着符号编码的进行，不断根据编码完成的符号更新符号的出现概率，以达到自适应编码的目的。

四、帧间预测编码

帧间预测主要是利用连续图象序列帧与帧之间的相关性，通过运动补偿的预测编码方法来消除视频图象的时间冗余。H.264 除了具有原有标准如 H.263、MPEG-4 中的 P 帧、B 帧预测方法外，还增加了许多新的特点[12]：使用可变块进行运动估计；1/4（亮度分量）像素精度的运动估计算法；采用多参考帧进行帧间预测编码。

1. 可变块运动估计

H.264 标准中对帧间预测时，使用可变块运动估计技术，比以前的编码标准提供了更大的灵活性。每个宏块可根据语法可分为 16×16，8×16，16×8，8×8 的块；当采用 8×8 块时，还可以进一步分为更小的 8×4，4×8，4×4 子块进行运动估计（如图 2.5 所示），预测时每个块拥有单独的运动矢量。也就是说当一个宏块全部使用 8×8 的子块进行运动估计时，该宏块需要传送 16 个运动矢量。

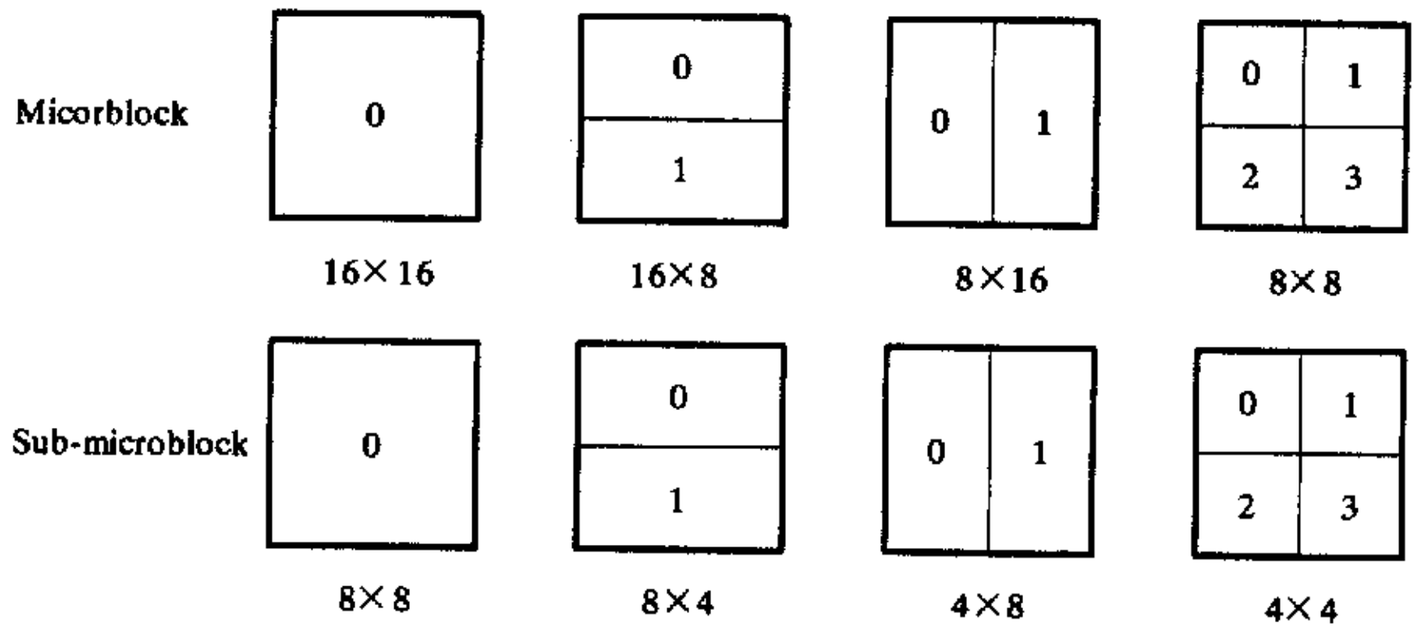


图 2.5 帧间编码中可变块的运动估计扫描顺序

采用可变块进行帧间预测，使得运动估计模型能够更接近物体的实际运动，因此运动补偿精度更高，使用这种方法比单独 16×16 块的预测方法提高大于 15%的编码率。另一方面细化的补偿所得到的图像块效应不明显，具有优良的视觉效果。

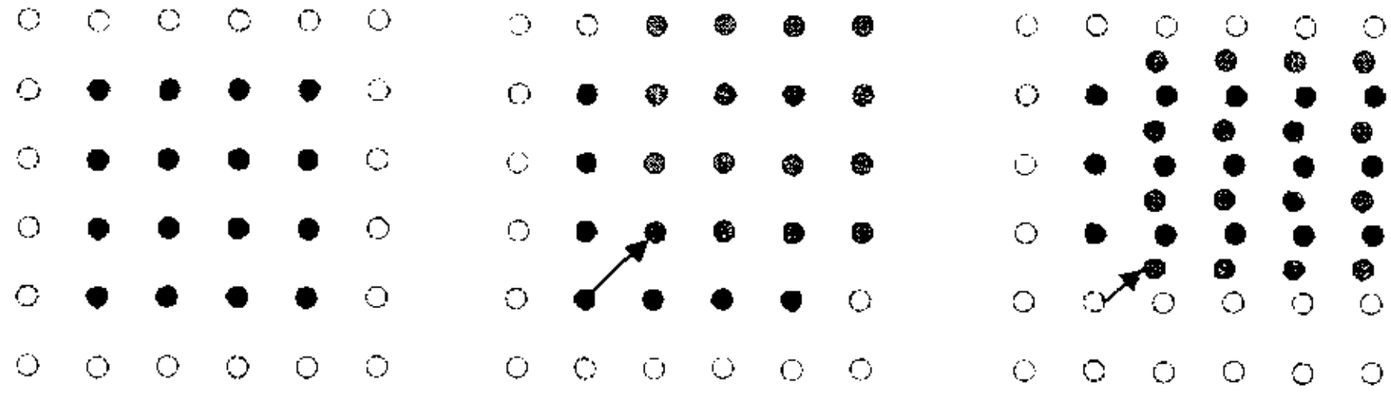
2. 高精度运动估计

H.263 中允许使用半像素精度的运动估计。H.264 进一步提高了运动估计精度，能够实现 1/4 象素精度的运动估计。与整数精度的空间预测相比，使用高精度的运动估计更为精确，PSNR 增益能够提高 1.3dB 以上。

高精度运动估计首先进行整像素级别的运动估计，当搜索到整数级别的最佳匹

配点时，在该点±1 像素范围内处进行插值，得到 1/4 像素精度的采样点。最后在这些 1/4 像素采样点中进行搜索，得出的就是高精度的最优运动矢量。

一般精度的运动估计算法和高精度运动估计算法的对比图示参见图 2.6，(a)中黑色点为原始的 4×4 块，(b)中示意原始块的运动矢量为(1,-1)，(c)中的运动矢量为(0.75,-0.5)。



(a)当前帧 4×4 块　　(b) 整像素运动估计　　(c) 1/4 像素运动估计

图 2.6 亚像素精度运动估计技术比较

H.264 的插值计算分为两步：首先对整数位置的像素点在水平和垂直方向上分别使用式(2-7)所示的 6 阶 FIR 滤波器做插值运算，以得到 1/2 像素位置的采样点。

$$F=[1,-5,20,20,-5,1] \tag{2-7}$$

例如水平方向的像素点分别为 A,B,C,D,E,F，此时点 C 和点 D 之间 1/2 像素位置的插值点为 $a=(A-5B+20C+20D-5E+f)$，再对 a 取整之后即可。

随后对整数位置和 1/2 像素位置的采样点进行普通的均值滤波，即得到了 1/4 像素位置的采样点。

上述的插值算法是针对亮度分量进行的。色度分量的高精度运动估计采用双线性插值获取非整数采样点的色度值，由于色度分量采样率只有亮度分量的 1/2，因而色度分量的运动估计最高可达 1/8 像素精度。

与 MPEG-4 AP(Advanced Profile)相比，H.264 中使用的差值算法运算量更小。在运动估计中可以产生低通滤波或高通滤波的效果，易于在图像的低频信息和高频信息中进行取舍，差值的结果在统计特性上与原数据的相似度差别不大。

3. 多参考帧预测模式

与 H.263 和 MPEG-1/2 不同，H.264 对 P 帧和 B 帧编码时最多可采用 5 个参考帧进行帧间预测，能够进一步提高运动估计的精度。这样比单独参考帧方法可以节省 5~10%的传输码率，并且有利于码流的错误恢复。图象编码顺序不是基于时间的图象显示顺序，而是基于图象之间的依赖关系的。图 2.7 列出了 P 帧编码多参考帧运动估计的示意图，这里使用过去的 3 帧对当前帧进行预测。

多参考帧预测模式同样适用于 B 帧。P 帧和 B 帧图象的参考图象存储在两个参考帧存储器中，分别存储前向预测参考帧和后向预测参考帧。由于每个参考帧存储器中都包含不止一幅图象，因此它包括了原有两个参考帧和多参考帧的更一般的情况，因此可以提高编码效率。但是同时，编码器必须通过参考帧选择过程选

取最佳参考帧进行运动补偿和预测过程，为此，还必须为参考帧提供内存空间和增加索引值，这也增加了系统的处理时间和存储开支。

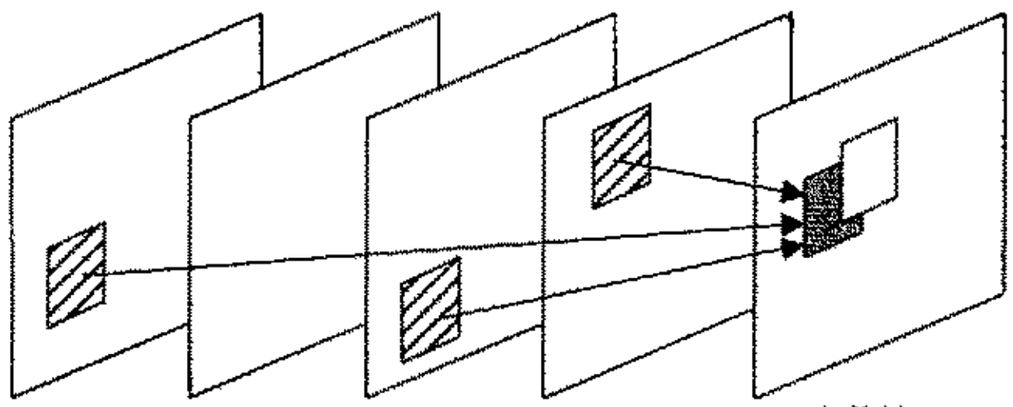


图 2.7 多参考帧编码示意图

五、去块滤波器

在低码率的情况下，由于采用较大的量化步长，基于块的变换编码算法不可避免会产生块效应。在 H.264 中，使用多参考帧在某些情况下也会加剧块效应现象。为此，H.264 中采用了自适应去块滤波器[13]（Adaptive Deblocking Filter）技术来缓解块效应对图像质量的影响。

视频编码中所使用的滤波器一般可以分为两种：后处理滤波器（post filter）和回路滤波器（in-loop filter）。后处理滤波器只对缓冲区内输出的图像进行处理。而去块滤波器位于编码器的运动估计/运动补偿回路中，重构帧必须经过滤波之后才可以存入帧存储器作为下一帧编码的参考帧。H.264 中所采用的自适应去块滤波器属于后者。

滤波器以 16×16 象素的宏块为单位，作用于重建图像的亮度分量和色度分量。除宏块边缘为图象边缘的部分不进行滤波外，其余各宏块都按照先垂直方向、后水平方向的顺序进行滤波。滤波的步骤如图 2.8 所示，其中实线代表亮度分量边界，虚线为色度分量边界。

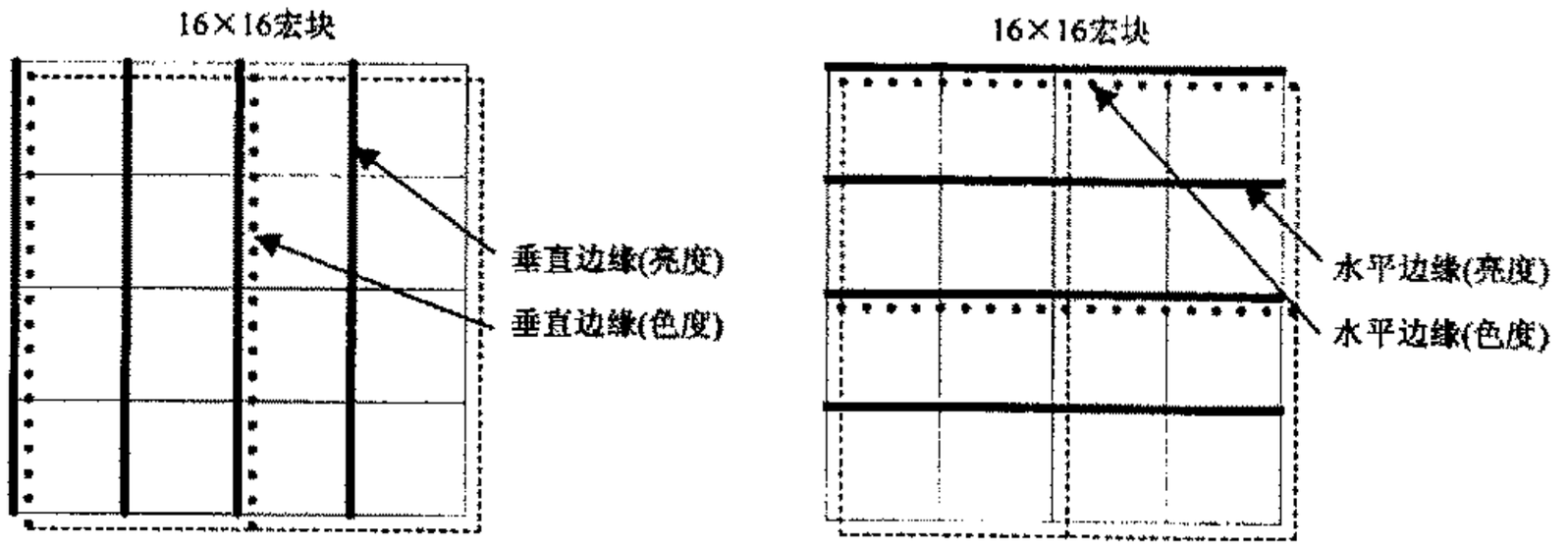


图 2.8 16×16 宏块内图象边缘 (a)垂直边缘 (b)水平边缘

自适应滤波的基本思想是使用边界强度自适应的判定是否需要滤波，如果检测到图像块的边缘采样点之间具有较大的绝对差，则有可能是影响主观视觉效果的人工边缘，需要进行平滑处理；但是如果此绝对差非常大（超过所设定的某一阈值函数），则这些采样点很有可能是原始图像的真实信息特征，此时需要将其保持。

图 2.9 显示了对于两个宏块的垂直边缘滤波的情形，$p_0\ p_1\ p_2\ q_0\ q_1\ q_2$ 代表子块相邻像素点的值，p_0 和 q_0 是宏块边缘的相邻像素。

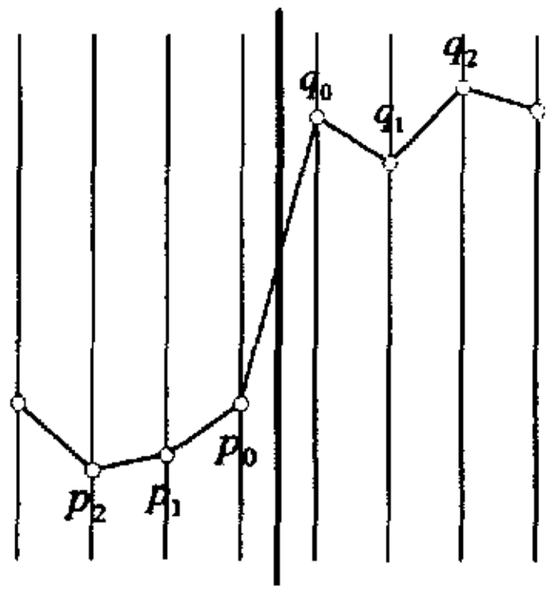


图 2.9 图像块的边缘像素点灰度示意

H.264 定义了两个阈值函数 $\alpha(QP)$ 和 $\beta(QP)$，其中 QP 为量化参数。当满足式(2-5)、(2-6)、(2-7)中的任何一个条件，则对边界相邻的像素值 p_0 和 q_0 进行滤波。

$$|p_0 - q_0| < \alpha(QP) \tag{2-5}$$

$$|p_1 - p_0| < \beta(QP) \tag{2-6}$$

$$|q_1 - q_0| < \beta(QP) \tag{2-7}$$

当满足式(2-8)或式(2-9)时，则还要进一步对 p_1 或 q_1 滤波。

$$|p_2 - p_0| < \beta(QP) \tag{2-8}$$

$$|q_2 - q_0| < \beta(QP) \tag{2-9}$$

实际应用中证明 H.264 的去块滤波方法能够有效的去除预测误差产生的块效应，而对图像原有的边缘信息能够尽可能的保持，所以能够有效提高主观视觉效果。但是采用去块滤波器同时也给系统增加了复杂度。JVT 在实验中发现，即使是经过代码优化的滤波器，其运算量也占到了编码器总运算量的 1/3 左右，因此寻求降低滤波器复杂度的方法也是目前亟待解决的问题。

2.4 H.264 网络适应特性

多媒体视频传输是第三代移动通信（3G）系统的一个重要应用[14]。一般来说，移动终端是小型的手持设备，其功率和存储能力有限。因此，一个移动视频编码器必须尽可能降低复杂度且仍然保持高效性和鲁棒性。由于多径衰落、时延扩展、噪声影响与多址干扰等原因，在无线信道中进行完全没有差错的视频通信是不现实的。对于对时延要求不高的视频应用可以通过重传来实现，但对视频实时会话服务来说不可能进行大量数据的重传。此外，在一个移动蜂窝区域的系统容量有限，且传输的数据量在不断变换。因此，要求视频编解码器能在有限时间里随着

环境的变换来改变编码速率，以适应信道环境的变换。这一切都决定了对用于移动环境的视频编解码器有如下要求：(1)非常高的压缩效率；(2)低功耗、较少的内存和低复杂度；(3)对误码、丢包有较强的鲁棒性；(4)支持快速的码率调整；(5)能产生不同的优先级；(6)能有效地利用特定网络的机制。

为了达到上述要求，H.264 采用全新的编码结构，首次提出了网络适配层（Network Adaption Layer, NAL）的概念，其码流结构对网络的适应性强；还采用了数据分区、数据掩盖和错误恢复等技术使之具备了在高误码率、丢包多发的信道中传输的能力，能够很好的适应 IP 和无线网络如 CDMA2000 和 UMTS 的应用。

一、编码结构的分层处理

H.264 的编码结构在概念上分为两层：视频编码层（Video Coding Layer, VCL）负责高效率的视频压缩能力；网络适配层（Network Adaption Layer, NAL）[15]负责网络的适配，即对不同网络要有不同的适应能力，如以恰当的方式对数据进行打包和传送。H.264 编码器分层结构如图 2.10 所示。在 VCL 和 NAL 之间定义了一个基于分组方式的接口，打包和相应的信令属于 NAL 的一部分。这样，高效率编码和网络适应性的任务分别由 VCL 和 NAL 来完成。

VCL 包括基于块的运动补偿混合编码和一些新特性。NAL 负责针对下层网络的特性对数据进行封装，包括成帧、发信号给逻辑信道、利用同步信息等。NAL 从 VCL 获得数据，包括头信息、段结构信息和实际净荷信息（如果采用数据分割技术，净荷数据可能由几部分组成）。NAL 的任务就是要正确地将它们映射到传输协议上。NAL 下面是各种具体的协议，如 H.323、H.324 等。NAL 层的引入大大提高了 H.264 适应复杂信道的能力。

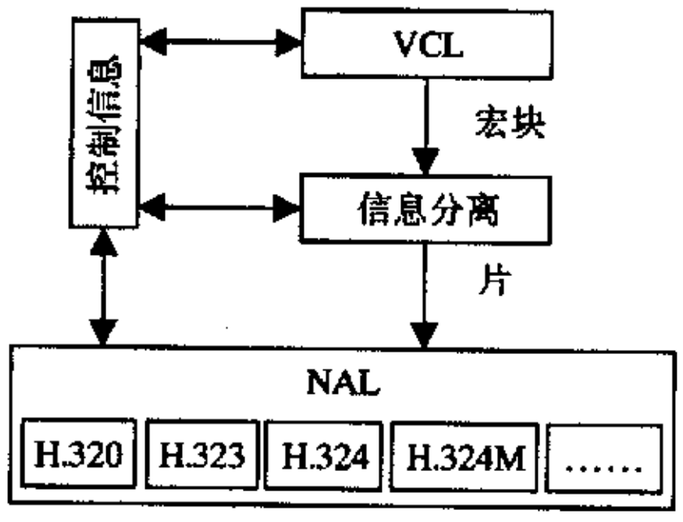


图 2.10　H.264 的分层编码结构

二、差错控制和码率恢复

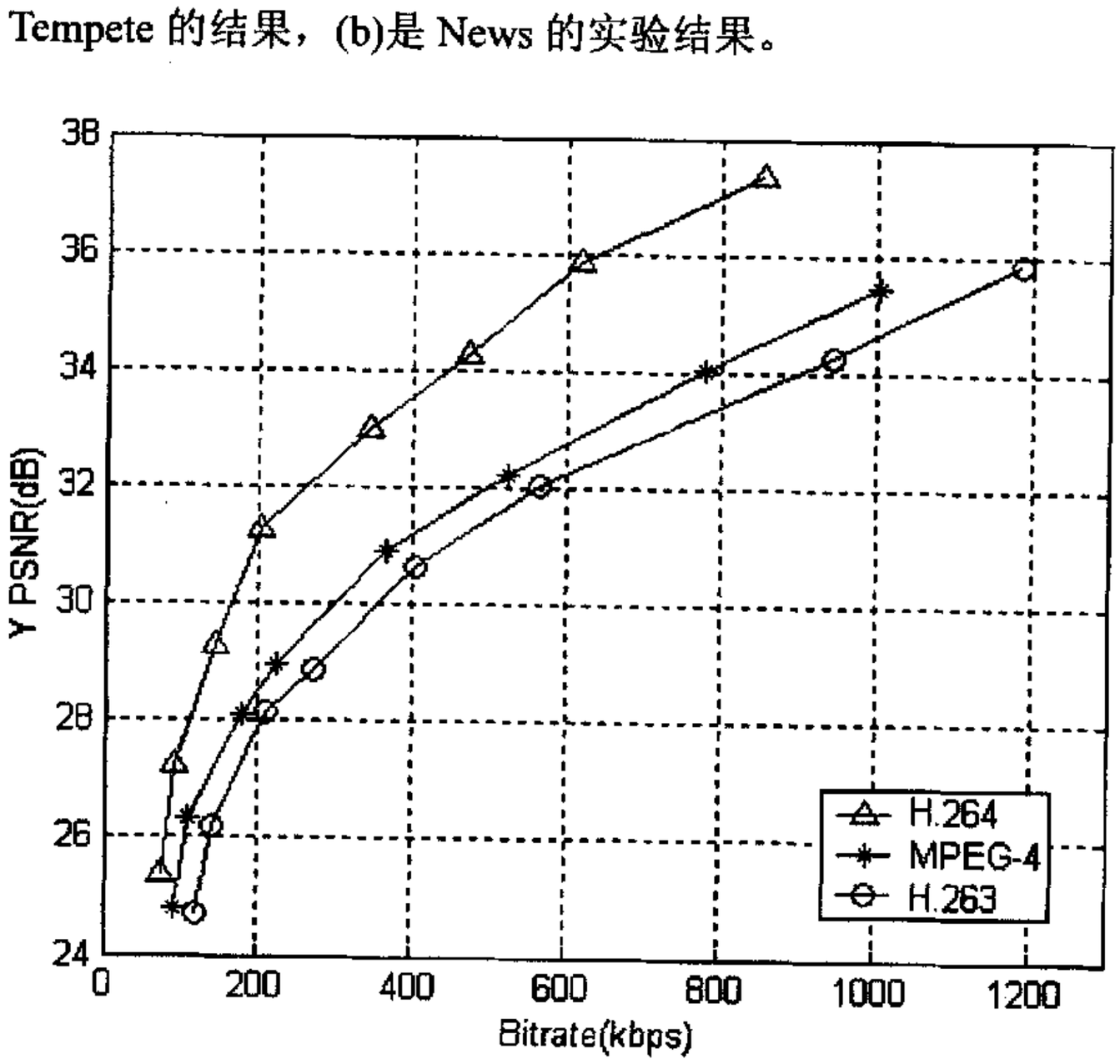
H.264 标准中包含用于差错消除的工具，有利于 H.264 视频流在误码、丢包多发的移动环境中传输，增强了 H.264 视频流的鲁棒性。为了减少传输差错，H.264 视频流中的时间同步可以通过采用帧内图像刷新来完成，空间同步由条结构编码（Slice Structured Coding）来支持。同时为了便于误码以后的再同步，在一帧的视

频数据中还提供了一定的重同步点。另外，帧内宏块刷新和多参考帧模式使编码器在决定宏块模式的时候不仅可以考虑编码效率，还可以考虑传输信道的特性。

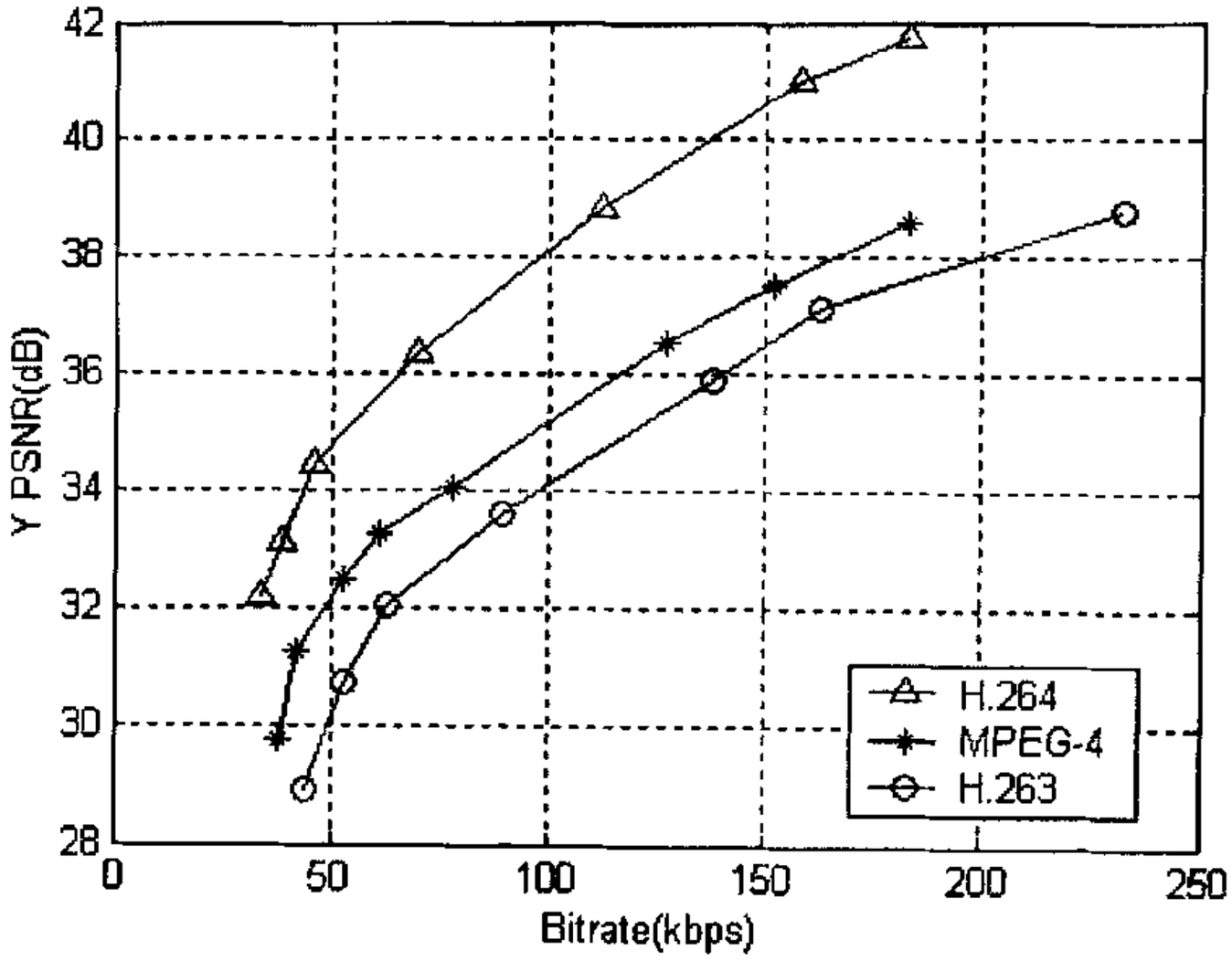
H.264 中还定义了数据分割模式：图像首先进行分段，段内宏块数据划分为宏块头信息、运动矢量和 DCT 系数三部分，且三部分之间由标识符分隔。这样，解码器可较方便地检测出受损数据的类型，减少误码对图像质量造成的损伤。这种数据分割的模式也利于信道编码时进行不等保护，即对重要的数据进行等级较高的保护。快速码率控制可通过在宏块层改变量化精度来实现。在移动通信的应用中，还可以使用空间/时间可伸缩特性来支持移动信道的较大比特率变化。

2.5 H.264 性能分析

本节将通过实验来测试 H.264 的编码性能，并与 H.263 和 MPEG-4 进行对比。我们选取 CIF 格式的图像序列 Tempete 和 QCIF 格式的序列 News，每个序列使用前 60 帧进行编码计算 PSNR 的均值；每隔 30 帧进行一次帧内编码。H.263 使用验证模型 TMN-3.12，可选模式只使用半像素运动估计和不受限运动矢量（Annex D）；H.264 使用 JM-6.0 验证模型，使用 5 参考帧模式，CACBC 编码；MPEG-4 采用校验模型 VM。图 2.11 是这几种编码算法在不同比特率下的 PSNR 性能比较，(a)是 CIF 格式 Tempete 的结果，(b)是 News 的实验结果。



(a) CIF 格式 Tempete 序列实验结果



(b) QCIF 格式 News 序列实验结果

图 2.11 几种视频编码算法的性能比较

从图 2.11 中可以看出，H.264 的压缩性能明显优于 H.263 和 MPEG-4。在相同的码率下，H.264 比 H.263 和 MPEG-4 最多提高了 2dB 左右。在相同图像质量下，H.264 约可节省 50%左右的码率，基本达到了 H.264 当初制定的目标。

2.6 小结

在长达 4 年的制定过程中，VCEG 以及后来的 JVT 通过一系列 H.264 验证模型对各种新算法、新技术进行了广泛的测试。从最初的 TML-1(Test Model Long-Term)到 JM-6(Joint Model)共计 15 个版本的验证模型中，不断有新算法被采纳，同时也有一些算法经过验证不能够满足要求而退出验证模型。例如在 TML-7/8 中纳入的 1/8 像素精度运动估计因为计算复杂度高而被放弃；JM-3 的熵编码方案采用 CAVLC 取代原有单一的 UVLC，进一步提高了编码效率；其它的一些算法，如帧内预测算法、整数变换算法和去块滤波器也先后采用了多种方案。还有一些技术，如自适应块变换（Adaptive Block Transform, ABT）等因为种种原因而未能写入 H.264 标准中。可以说，H.264 的制定就是编码算法不断更新、不断优化的过程。表 2.1 反映了 H.264 和 MPEG 标准的技术特性差异。

表 2.1 几种视频编码标准技术特性比较

技术特性	MPEG-1	MPEG-2	MPEG-4 ASP	H.264
宏块尺寸	16×16	16×16(帧模式) 16×8(场模式)	16×16	16×16
块尺寸	8×8	8×8	16×16, 8×8, 16×8	16×16,16×8, 8×16, 8×8, 8×4, 4×8, 4×4,
变换	DCT	DCT	DCT/DWT	整数变换
量化步长	固定步长	固定步长	矢量量化	步长按 12.5%增加
熵编码	VLC	VLC	VLC	CAVLC, CABAC
帧内预测	否	否	是	是
运动估计	是	是	是	是
估计精度	整数像素, 1/2 像素	整数像素, 1/2 像素	整数像素, 1/2 像素, 1/4 像素	整数像素, 1/2 像素, 1/4 像素
参考帧数目	1	1	1	最多支持 5 参考帧
帧类型	I, P, B	I, P, B	I, P, B	I, P, B, SI, SP
错误隐藏	是	是	是	是
码率	<1.5Mbps	2-15Mbps	64kbps-2Mbps	64kbps-150Mbps
编码复杂度	低	中	中	高
类别	否	5 profile	8 profile	3 profile

作为面向未来 IP 和无线环境的视频编码标准，H.264 的视频压缩效率比目前所有的视频编码标准都要高，而且算法结构上得分层处理使它能适应不同的传输环境，提高传输效率。H.264 的这些优点必将使其在视频通信领域得到广泛应用，如实时视频通信、Internet 视频传输、视频流媒体服务、异构网上的多点通信、压缩视频存储、视频数据库等。在未来的移动视频通信领域，H.264 具有非常广阔的应用前景，发展潜力巨大。

但是 H.264 性能的提高是以复杂度的提高为代价获得的。据估计，H.264 编码的计算复杂度大约相当于 H.263 的 3 倍，解码复杂度相当于 H.263 的 2 倍。这种高复杂度对于 H.264 的应用是一个瓶颈，因此迫切需要寻找 H.264 关键技术的快速算法。在随后的几章中，我们会对整数变换算法、帧内预测算法、和运动估计算法进行详尽探讨和研究。

第三章 变换和量化算法研究

3.1 引言

变换编码是图像视频编码系统的重要组成部分，基于变换的编码方法已经成为图像编码和视频编码的主流，如视频编码标准 MPEG-x 和 H.26x，静止图像压缩标准 JPEG、JPEG2000[16]。变换将在空间域内以像素值形式表示的图像信息变换到频域中，以变换系数的形式加以表示，然后对这些变换系数进行编码处理。

变换能够减少系数之间的相关性，从而有效地消除图像的空间冗余；另一方面视频和图像信号的能量主要集中在低频部分，能量密度随频率的升高而迅速下降，考虑到人眼对于高频信号不敏感的视觉特性，变换后利于针对不同频率的信号进行量化、游程编码和熵编码，从而达到数据压缩的目的。变换编码的性能对于编码系统的整体性能有很大的影响，常用的变换算法有 WHT（沃尔什-哈达马变换），DFT（离散傅立叶变换），DCT（离散余弦变换）和 DWT(离散小波变换)等。H.26L 的验证模型中使用了整数 DCT 取代了传统的 DCT 变换；在最终获得通过的 H.264 标准中，联合视频专家组对新标准所使用的变换和量化算法又做了进一步的改进，采用了新的整数变换核，使得整个整数变换和量化过程可以通过 16 位的算术运算完成，且不使用乘法运算。整数变换算法能够在不影响编码性能的前提下，有效降低计算复杂度，提高编码效率。

本章首先简要描述了传统的 DCT 算法，分析了其不足之处，然后重点研究了 H.264 所采用的整数变换和量化算法，最后通过实验数据的对比证明整数变换算法的高效性。

3.2 离散余弦变换

变换具有将图像能量集中于某些系数中的能力，一个能把最多的能量集中于最少的系数上的变换所产生的重建误差最小。理论上，K-L 变换（Karhunen-Loeve Transform）是所有变换中集中能力最优的变换，它能够完全去处原信号中的相关性。但是 K-L 变换的基函数取决于待变换图像的协方差矩阵，因此其变换基函数是不确定的，而且计算特征值和特征向量的工作量也很大，所以只在特殊需要的场合才使用。DCT 是最接近 K-L 变换的正交变换，可以将图像的大部分能量集中到直流系数中，从而有效去除图像的空间相关性；同时又兼具有效的快速算法，易于进行硬件实现。因此，除了 JPEG2000 和 MPEG-4 的纹理编码采用 DWT 之外，DCT 变换成为目前绝大多数编码标准的变换编码算法。

3.2.1 DCT 数学模型

DCT 是由 N.Ahmed 等人在 1974 年提出的变换算法[17]，共具有四种不同的形式，分别称为 DCT-I~DCT-IV。在信号处理领域中应用最广的是 DCT-II，通常也直接将 DCT-II 称为 DCT。DCT 的实质是通过线性变换 $X = Hx$，将一个 N 维向量 x 变换为变换系数向量 X。DCT 变换核 H 的第 k 行第 n 列的元素定义为

$$H(k,n) = c_k \sqrt{\frac{2}{N}} \cos\frac{(2n+1)k\pi}{2N} \tag{3-1}$$

其中 $k = 0, 1, \ldots, N\text{-}1$，$n = 0, 1, \ldots, N\text{-}1$，$c_0 = \sqrt{2}$，$c_k = 1$。DCT 是线性正交变换，因此 DCT 是完全可逆的，且逆矩阵就是其转置矩阵。

为了降低运算量，图像编码中一般将图像分为相互独立的块，以块为单位做二维 DCT。二维 N×N 点 DCT 公式为

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N} \tag{3-2}$$

IDCT 公式为

$$f(x,y) = \frac{2}{N} C(u)C(v) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N} \tag{3-3}$$

其中 x,y 是空间坐标，$x,y = 0,1,\ldots,N-1$；u,v 是 DCT 空间坐标，$u,v = 0,1,\ldots,N-1$。可变系数 $C(0) = 1/\sqrt{2}$，$C(i) = 1$，$i = 1,2,\cdots,N-1$。

可以看出，DCT 变换只需实数偶函数形式，任何一个在给定区间内满足狄里赫利条件的连续实对称偶函数，均可以展成仅含余弦项的傅立叶级数，即 DCT。

3.2.2 DCT 快速算法

二维 DCT 含有较多的重复运算，计算复杂度高。例如对于 8×8 大小的图像子块直接使用式(3-2)做 DCT 变换，共需要 4096 次乘法运算和 3584 次加法运算。在图像编码尤其是实时视频和多媒体业务的应用中，如此庞大的运算量占用了大量的系统资源，会对系统的整体性能和实时性带来较大的影响，因此需要寻找精确而又快速的算法。1992 年，Feig-Winograd 通过把 DCT 看作是一种循环旋转运算，证明了在有理数域上计算长度为 2^n 的一维 DCT 所需的最小实数乘法次数为 $2^n - n - 2$，对于一维 8 点 DCT，最少需要 11 次乘法[18]针对不同的软硬件应用，人们提出了许多种快速算法和实现技术。目前的常见的快速 DCT 算法可以归纳为如下几种技术：

(1) 利用 DCT 与其他快速变换的关系来达到快速运算的目的，也称为间接 DCT 算法，如使用 FFT[19], WHT[20][21], DHT[22]等。间接算法过程简单，但是由于这些变换的快速算法在 DCT 的应用中存在一定的局限性，所以目前已较少采用。

(2) 利用 DCT 变换矩阵的稀疏分离特性以实现快速运算。比较具有代表性的是 Chen 和 Loeffler 的分解算法。其中使用 Loeffler 的 LLM[23]算法，一维 8 点 DCT 变换只需要 11 次乘法和 29 次加法，乘法的计算复杂度达到了理论下限。

(3) 对于二维 DCT，利用其变换核的可分离性，将 2 维 DCT 分解为串联的 2 次一维 DCT 运算。即对一个 N×N 的二维矩阵，先做 N 行（或列）的一维 DCT，产生中间矩阵，然后对此中间矩阵做 N 列（或行）的一维 DCT，得到最终的二维 DCT 结果。使用此种分离算法可以将计算量减少到原来的 1/4，还可以利用一维 DCT 的快速算法进一步简化运算。这种算法适合硬件实现，可以大大简化系统的设计。

(4) 使用尺度 DCT（scaled DCT）[24]。尺度 DCT 的思想是将 DCT 的部分运算结合到系统的其它部分中计算（如视频编码中的量化部分），从而减少 DCT 部分的运算量。

下面对几种经典的 DCT 快速算法做简要介绍。

一、AAN 快速算法

AAN算法[25]是Y. Arai, T. Agui和M. Nakajima于1988年提出的一种快速算法，它也是将二维DCT分解成行和列的一维变换，一维8点的DCT变换再通过16点离散傅立叶变换（DFT）来实现，而16点DFT又可以通过快速傅立叶变换(FFT)实现。其算法流程参见图3.1。

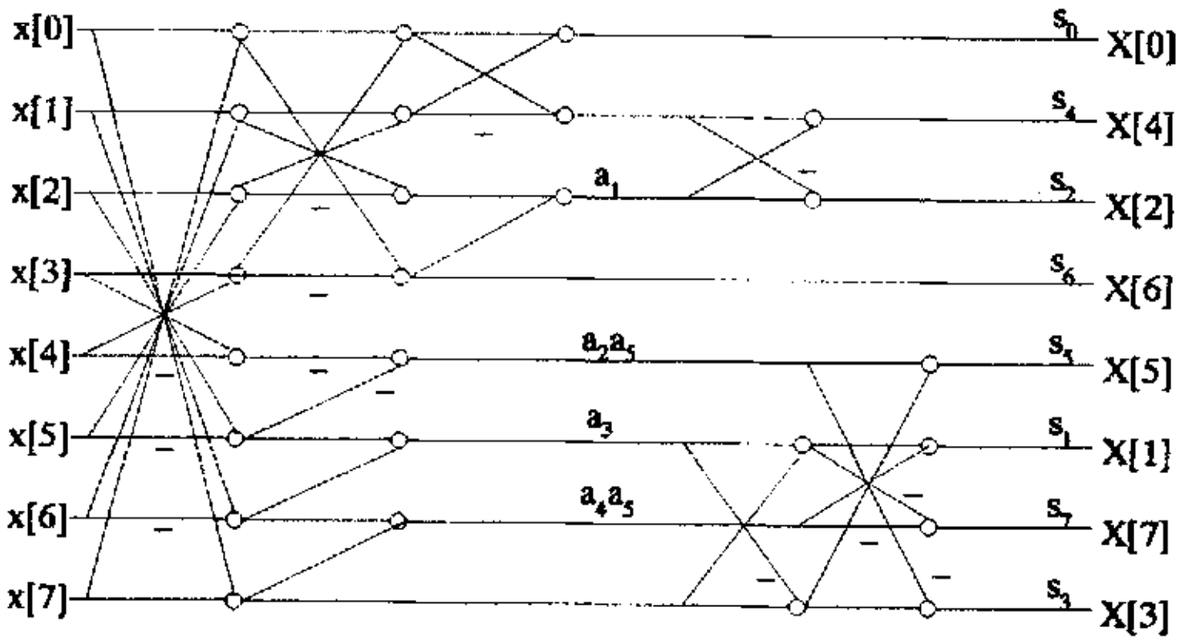


图3.1 一维8点AAN变换的流程图

其中 $a_1 = C_4$，$a_2 = C_2 - C_6$，$a_3 = C_4$，$a_4 = C_2 + C_6$，$S_0 = \frac{1}{2\sqrt{2}}$，$s_k = \frac{1}{4C_k}$，而 $C_k = \cos(k\pi/16)$。

从一维AAN变换的算法流程图可以看出，其一维8点变换只需13次乘法和29次加法，如果将最后的尺度变换和量化结合在一起，则变换部分只需13-8=5次乘法和29次加法。二维8×8点DCT变换采用此法要16×5=80次乘法和16×29=464次加法（不考虑尺度转换）。

二、Chen分解算法

W. Chen等人提出了基于旋转分解的N点（$N=2^m$，$m\geq 2$）DCT的快速递归算法[26]，其蝶形结构图如图3.2所示。使用该种算法计算一维8点变换需要13次乘法运算和29次加法运算。图中为了简明起见，我们将cos简写为C，sin简写为S。

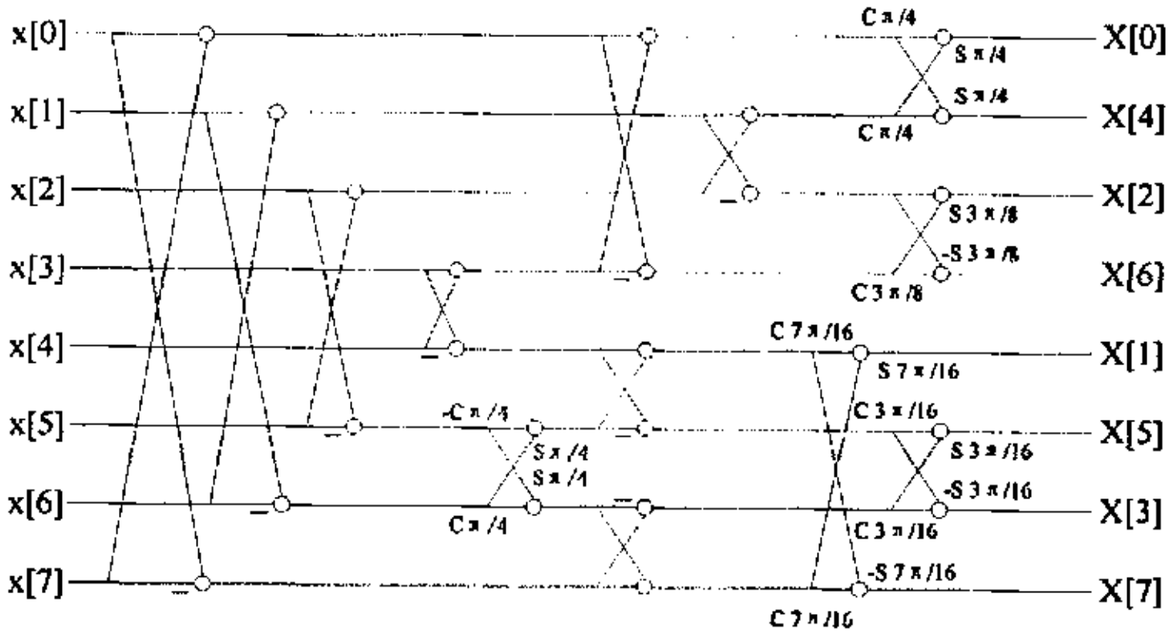


图3.2 一维8点Chen分解流程图

三、LLM 算法

Loeffler, Lightenberg 和 Moschytz 提出了一种更为有效的快速分解算法，称为 LLM 算法。使用 LLM 算法，一维 8 点 DCT 只需要 11 次乘法和 29 次加法。这种算法所使用的乘法运算次数达到了理论下限。JPEG 和 H.263 中的 DCT 运算部分都采用了 LLM 算法。图 3.3 显示了 LLM 快速算法的蝶形结构图。

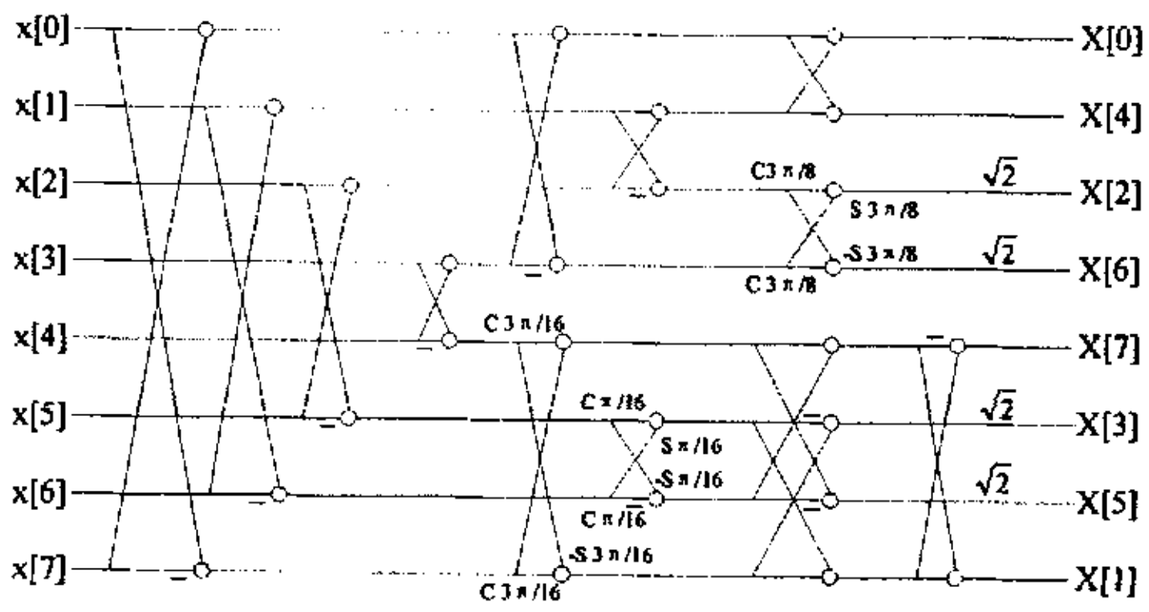


图 3.3 LLM 算法流程图

四、查找表法

查找表法是在 DCT 的硬件实现中被广泛使用的一项技术。它从节省余弦运算的角度出发，将计算中所需的余弦值在 DCT 变换前计算出来，放在表（或数组）中，然后在变换中直接查取该表选取所使用的数值。分析得知，一个 8×8 的矩阵做 DCT 运算只需 45 个余弦值，它们是

$\cos 0\pi/16$ $\cos \pi/16$ $\cos 2\pi/16$ $\cos 3\pi/16$ $\cos 4\pi/16$ $\cos 5\pi/16$
$\cos 6\pi/16$ $\cos 7\pi/16$ $\cos 9\pi/16$ $\cos 10\pi/16$ $\cos 11\pi/16$ $\cos 12\pi/16$
$\cos 13\pi/16$ $\cos 14\pi/16$ $\cos 15\pi/16$ $\cos 18\pi/16$ $\cos 20\pi/16$ $\cos 21\pi/16$
$\cos 22\pi/16$ $\cos 25\pi/16$ $\cos 26\pi/16$ $\cos 27\pi/16$ $\cos 28\pi/16$ $\cos 30\pi/16$
$\cos 33\pi/16$ $\cos 35\pi/16$ $\cos 36\pi/16$ $\cos 39\pi/16$ $\cos 42\pi/16$ $\cos 44\pi/16$
$\cos 45\pi/16$ $\cos 49\pi/16$ $\cos 52\pi/16$ $\cos 54\pi/16$ $\cos 55\pi/16$ $\cos 60\pi/16$
$\cos 63\pi/16$ $\cos 65\pi/16$ $\cos 66\pi/16$ $\cos 75\pi/16$ $\cos 77\pi/16$ $\cos 78\pi/16$
$\cos 90\pi/16$ $\cos 91\pi/16$ $\cos 105\pi/16$

上述查找表还可以进一步精简。这样就可以由查找表来解决大量的重复余弦值运算问题，从计算 4096 个余弦值降低为 45 个甚至更少。因为查表的速度远远快于计算大量的余弦值，从而提高了变换的速度。查找表技术还可以与其它 DCT 快速算法结合使用。

3.2.3 DCT 算法分析

DCT 具有优良的压缩性能，但其缺点也是十分明显的。从式(3-1)中可以看出，DCT 变换核中的元素 $H(k,n)$ 是无理数，这意味着在做 DCT 变换时需要进行浮点运算。对于向量 x，做 DCT 变换 $X = Hx$ 及反变换 $u = round(H^T X)$ 后，并不能够保证对于所有的 $x(n)$，反变换后的结果 $u(n)$ 都能满足 $u(n) = x(n)$。也就是说使用计算机进行 DCT 变换和反变换之后得到的可能是不同于原输入的结果。因此利用有限精度的浮点 DCT 实现变换编码只能是有损的编码。这在一些特定的场合如医学图像、遥感成像等应用中是不允许的。

另外，在硬件实现中不同的处理器对于浮点数和取整的定义不同也会造成不同的结果。在整个视频编码过程中，只有 DCT 变换是浮点运算，其他的编码过程如量化、熵编码和图像重构都是定点运算。如果采用定点 DSP 芯片进行浮点运算时，直接计算的计算量相当大，实时性差。

此时一个较好的解决办法是寻找一个近似于 DCT 的整数变换矩阵，使用定点运算来代替浮点运算。由于是整数到整数的变换，因此对于所有的输入都可以进行精确的运算，而不存在反变换的误匹配问题，同时整数运算必然会带来运算速度的提高。

3.3 H.26L 整数变换方案

近年来，人们在 DCT 运算整数化方面做了大量的研究工作。石青云等人证明了如下结论：线性变换只要是有限维的和可逆的，则必然存在整数实现方式[27]。并给出了诸如 FFT 和 DCT 的快速实现方法。为了获得更快的速度，DCT 系数还

可以利用整数进行尺度缩放和近似，通过该种方法用整数乘法来替代浮点数乘法，实现由整数变换到整数，该种算法也被称为整数 DCT（Integer DCT）[28]，在速度上较以前的 DCT 算法有显著提高。

随着小波提升的深入研究，人们对提升结构（lifting scheme）也有了深刻的认识。如果能将提升结构对传统的 DCT 蝶形运算加以改造，也可以实现可逆的整数到整数的 DCT 变换算法。1999 年以后发展起来的基于提升结构的 DCT 算法就是利用提升结构消去了乘法，而且还能达到无损的压缩效果。Jie Liang, Trac D. Tran, Ying-Jui Chen 等人提出了几种基于提升的整数变换算法[29][30]。

有鉴于 DCT 的缺点，在 H.26L 的制定过程中联合视频专家组决定采用整数变换替代传统的 DCT 变换，这也是整数变换算法在视频国际编码标准中的首次应用。与基于提升的整数 DCT 变换算法不同，H.26L（包括正式的 H.264 标准）发展了全新的整数 DCT 变换方案[31]。H.26L 中采用的整数变换方案包括两种变换算法：一种是类似于 DCT 的 4×4 整数正交变换算法，另一种是对变换的 DC 系数所做的 2×2 整数变换。下面对这两种算法整数变换和量化方案进行详细的推导与分析。

3.3.1 4×4 块整数变换算法

一、整数变换核的构造

对于式(3-1)中的一维DCT，当N= 4时，我们可以将其改写为如下矩阵乘积的形式

$$Y = HX = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \tag{3-4}$$

其中 $a = \frac{1}{2}$，$b = \frac{1}{\sqrt{2}}\cos(\pi/8)$，$c = \frac{1}{\sqrt{2}}\cos(3\pi/8)$。

为了生成一个近似于 DCT，但是只包含整数系数的变换矩阵，根据整数 DCT 的思想，我们可以将式(3-1)的无理数变换核扩大 α 倍并取整，即

$$Q(k,n) = round(\alpha H(k,n)) \tag{3-5}$$

这里 α 是整型尺度因子。经过验证，当 $\alpha < 100$ 时，只有 $\alpha = 2$ 和 $\alpha = 26$ 能够满足变换核的正交性要求。而当 $\alpha = 2$ 时 H 是 Hadamard 阵，因此使用 $\alpha = 26$，我们得到新的变换核 Q，如式(3-6)。

$$Q = round(26H) = \begin{bmatrix} 13 & 13 & 13 & 13 \\ 17 & 7 & -7 & -17 \\ 13 & -13 & -13 & 13 \\ 7 & -17 & 17 & -7 \end{bmatrix} \tag{3-6}$$

新的整数变换核仍是正交阵，且各列范数相同。因此，可以直接使用其逆矩阵构造反变换核，定义为

$$Q_{inv} = Q' = \begin{bmatrix} 13 & 17 & 13 & 7 \\ 13 & 7 & -13 & -17 \\ 13 & -7 & -13 & 17 \\ 13 & -17 & 13 & -7 \end{bmatrix} \tag{3-7}$$

式(3-6)和式(3-7)也正是在最初的 H.26L 草案中所采用的整数 DCT 变换算法。在变换编码中一维整数变换使用简单的一次方程式来完成。假设变换前输入的一行（或列）像素值为 $x[0], x[1], x[2], x[3]$，变换之后的结果为 $X[0], X[1], X[2], X[3]$。一维整数变换可以根据变换核 Q 改写为如下形式：

$$\begin{aligned} X[0] &= 13x[0] + 13x[1] + 13x[2] + 13x[3] \\ X[1] &= 17x[0] + 7x[1] - 7x[2] - 17x[3] \\ X[2] &= 13x[0] - 13x[1] - 13x[2] + 17x[3] \\ X[3] &= 7x[0] - 17x[1] + 17x[2] - 7x[3] \end{aligned} \tag{3-8}$$

一维整数反变换也可以写为一次方程式的形式，$x'[0]\ x'[1]\ x'[2]\ x'[3]$ 是反变换后的恢复结果：

$$\begin{aligned} x_r[0] &= 13X[0] + 17X[1] + 13X[2] + 7X[3] \\ x_r[1] &= 13X[0] + 7X[1] - 13X[2] - 17X[3] \\ x_r[2] &= 13X[0] - 7X[1] - 13X[2] + 17X[3] \\ x_r[3] &= 13X[0] - 17X[1] + 13X[2] - 7X[3] \end{aligned} \tag{3-9}$$

H.26L 中的整数变换算法利用了尺度 DCT（Scaled DCT）的思想，即在变换过程中不进行归一化运算。对一个输入向量 x 执行上述的一维整数变换和反变换，输出的恢复结果扩大了 676 倍，即 $x' = 676x$。随后将尺度缩放部分的运算结合到量化中完成，降低整个变换编码的运算量。

二、整数变换核的分析

对于式(3-1)所示的线性变换 $X = Hx$ 有如下结论

$$\text{如果 } x(n) \le U, \quad \text{则有 } X(k) \le \beta U \tag{3-10}$$

这里 β 是变换核矩阵 H 范数的最大值，定义为 $\beta = \max_k(\sum_n |Q(k,n)|)$。

对于图像来说，变换前输入矩阵的像素值的范围是 0~255（对于 P 帧和 B 帧来说，范围是-255~255），即上限 $U = 255$，因此每个像素需要使用 9 bit 来表示。使用 H.26L 整数 DCT 算法，$\beta = 52$，因此进行二维整数变换之后，信号扩大为原来的 52^2=2704 倍。由于 $\log_2 2704 = 11.4$，因而在变换后至少需要 12+9=21bit 才可以精确的表示变换后的系数，在实际中通常采用 32 位数据格式来存储和表示。

三、量化过程

从某种意义上讲，变换是无损的，只是单纯的将信息从空间域转换到一个变换域中使之能够被有效的编码，真正造成图像信息损失的是量化过程。在图像和视频编码标准中，一般按照如下方式定义量化和反量化过程。

对于量化步长 Q_s，量化公式为

$$X_q(i,j) = sign\{X(i,j)\}\frac{|X(i,j)| + f(Q_s)}{Q_s} \tag{3-11}$$

反量化公式为

$$X_r(i,j) = Q_s X_q(i,j) \tag{3-12}$$

其中 $X(i,j)$ 是输入像素值，$X_q(i,j)$ 是量化后的结果，$X_r(i,j)$ 是反变换后的恢复结果，$f(Q_s)$ 用于量化取整。可以看出在传统的量化过程中必须使用除法运算。而在一些硬件如 DSP 中，除法运算通常是通过调用库函数实现，而这势必会打破流水线的循环操作，影响量化速度。

H.26L 中采用了一种新的量化方法。首先，如果量化表中的量化步长满足

$$A(QP)B(QP)G^2 = 2^{L+N} \tag{3-13}$$

那么便可以在量化和反量化过程中可以分别使用 L 位和 N 位的移位运算来代替除法运算。具体选取 L 和 N 值的时候，考虑到取值较大，会占用较多的存储空间，同时计算量也会有一定的增加；取值较小，会带来较大的误差，不能满足式(3-13)的要求。在 H.26L 的变换算法中，G=676，此时可取 L=N=20。

于是量化公式变为：

$$X_q(i,j) = sign\{X(i,j)\}\left\lfloor \left(|X(i,j)|A(QP) + f \cdot 2^L\right) >> L \right\rfloor \tag{3-14}$$

反量化公式为

$$X_r(i,j) = X_q(i,j)B(QP) \tag{3-15}$$

其中 $|f| = 0 \sim 0.5$，且与 $X(i,j)$ 同符号，用于对量化值进行取整。由于所设定除数为 2 的整数次幂，所以量化中的除法运算可以用 L 位的右移位替代，而无需使用乘法。

量化数组 A(QP)和反量化数组 B(QP)共包含 32 个量化值。QP 越大，量化越粗糙，编码后的图像质量也就越差。A(QP)和 B(QP)定义如下：

A(QP=0,..,31)={620,553,492,439,391,348,310,276,246,219,195,174,155,138,123,110,98,87,78,69,62,55,49,44,39,35,31,27,24,22,19,17}.

B(QP=0,..,31)={3881,4351,4890,5481,6154,6914,7761,8718,9781,10987,12339,13828,15523,17435,19561,21873,24552,27656,30847,34870,38807,43747,49103,54683,61694,68745,77615,89113,100253,109366,126635,141533}.

在量化数组中，相邻的量化值之间保持大约 12.5%的增量，使得 QP 每增加 6，量化步长约变化为原来的 2 倍或 1/2,。即

$$A(QP+6) \approx B(QP)/2, \quad B(QP+6) \approx 2B(QP) \tag{3-16}$$

这样就使得量化参数 QP 和图像的 PSNR 值能够保持一种近似的线性关系，便于编码器在码率和图像质量之间进行调节。

3.3.2 2×2 块整数变换算法

在进行了 2×2 的整数 DCT 变换后，为了进一步降低 DC 系数的相关性，H.26L 提取色度分量的 DC 系数，组成 2×2 大小的 DC 系数块，再单独进行一次变换编码。相比较于 4×4 整数 DCT，2×2 整数变换计算更加简洁。

设提取出的的 2×2DC 系数矩阵为 x_D，变换后的结果为 X_D。则整个计算过程如下。

正变换过程：

$$\begin{aligned} X_{D00} &= (x_{D00} + x_{D01} + x_{D10} + x_{D11})/2 \\ X_{D10} &= (x_{D00} - x_{D01} + x_{D10} - x_{D11})/2 \\ X_{D01} &= (x_{D00} + x_{D01} - x_{D10} - x_{D11})/2 \\ X_{D11} &= (x_{D00} - x_{D01} - x_{D10} + x_{D11})/2 \end{aligned} \tag{3-17}$$

反变换过程：

$$\begin{aligned} x_{D00} &= (X_{D00} + X_{D10} + X_{D01} + X_{D11})/2 \\ x_{D01} &= (X_{D00} - X_{D10} + X_{D01} - X_{D11})/2 \\ x_{D10} &= (X_{D00} + X_{D10} - X_{D01} - X_{D11})/2 \\ x_{D11} &= (X_{D00} - X_{D10} - X_{D01} + X_{D11})/2 \end{aligned} \tag{3-18}$$

3.4 H.264 整数变换方案

H.26L 所采用的整数 DCT 有效的提高了编解码系统的整体性能，但是在变换编码过程中必须使用 32 位运算，变换编码的中间值也必须使用 32 位的数据类型才可以精确的存储和表示。在实际应用中，尤其是一些内存和处理器的计算能力不高的移动设备，32 位整数运算仍然受到一定的限制。在随后的 JVT 提案中，Texas Instrument、Nokia/Microsoft 和 FastVDO 公司分别提出了改进的整数变换算法。在 JM-2 验证模型中采纳了 Nokia 和 Microsoft 联合提出的整数变换方案，这也是在最终的 H.264 建议中所使用的变换方案。新方案采用了全新的变换核和量化公式，在保留整数变换优点的前提下，使变换可以通过 16 位运算来实现，有效的降低对存储器的要求；同时简化了运算步骤，在变换时只使用加法和移位运算，而无须使用乘法运算。使用这种算法做变换和反变换同样是完全可逆的，不存在误匹配问题。

H.264 中新的整数变换方案是一个“分层”式的变换算法，其核心是 4×4 块的亮度/色度分量整数变换；为了提高压缩比，还对亮度分量 DC 系数做 4×4 块整数变换，针对色度分量 DC 系数做 2×2 块整数变换。对图 3.4 显示的宏块，Y,U,V 分别代表亮度和色度分量，变换过程描述如下：首先对块 0~块 15 依次进行 4×4 整数变换；之后将亮度分量块的 DC 系数块（即图中的块-1）进行变换并传输，下来对色度分量的 DC 系数矩阵（块 16、块 17）进行变换。

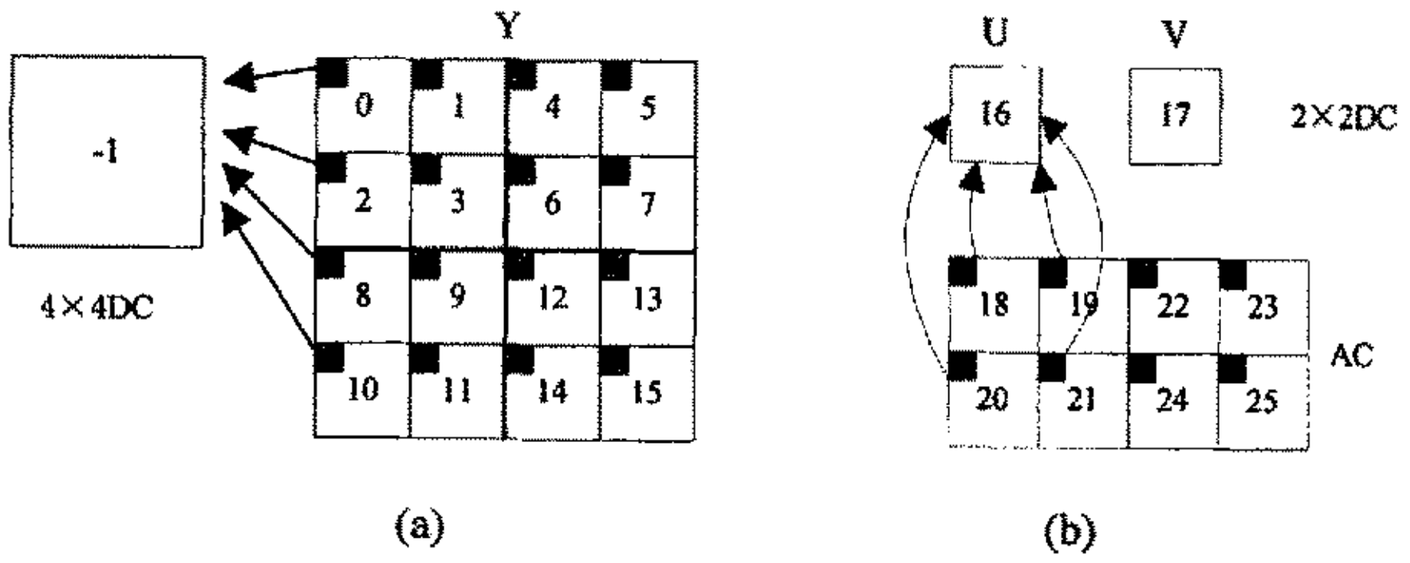


图 3.4 H.264 子块排列顺序 (a)亮度分量 (b)色度分量

3.4.1 4×4 整数变换算法

一、整数变换核的构造

仿照式(3-4)可以将二维 4×4 块 DCT 运算写为如下形式

$$Y = AXA^T = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix} \tag{3-19}$$

其中 $a = \frac{1}{2}$，$b = \frac{1}{\sqrt{2}}\cos(\pi/8)$，$c = \frac{1}{\sqrt{2}}\cos(3\pi/8)$。

将式(3-19)的 A 阵做进一步分解，得

$$Y = BCXC^T B =$$

$$\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & b \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & 1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & b \end{bmatrix} \tag{3-20}$$

若将 B 阵合并，即 $Y = (CXC^T) \otimes E =$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & 1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \tag{3-21}$$

上式中，$d = c/b = 0.414213...$，⊗代表两矩阵的点乘。为了构成整数变换核，必须将无理数 d 替换，如 $d = 7/16, 3/8, 1/2$ 等。H.264 中令 $d = 1/2$。为了满足 A 的正交性，即 $A^T A = I$，得到 $b = \sqrt{2/5}$，$a = 1/2$，$c = \sqrt{1/10}$。

经上述的取值，式(3-21)写为

$$Y =$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & 1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \tag{3-22}$$

此时在式(3-22)中的变换核只包含有 1 和 1/2 两个元素。为了获得只包含整数的变换核，同时消除变换和反变换的阶段误差，再对变换核的第一行和第三行同时扩大 2 倍，变为

$$Y = \begin{bmatrix}1 & 1 & 1 & 1\\ 2 & 1 & -1 & 2\\ 1 & -1 & -1 & 1\\ 1 & -2 & 2 & -1\end{bmatrix}\begin{bmatrix}x_{00} & x_{01} & x_{02} & x_{03}\\ x_{10} & x_{11} & x_{12} & x_{13}\\ x_{20} & x_{21} & x_{22} & x_{23}\\ x_{30} & x_{31} & x_{32} & x_{33}\end{bmatrix}\begin{bmatrix}1 & 2 & 1 & 1\\ 1 & 1 & -1 & -2\\ 2 & -1 & -2 & 2\\ 1 & -2 & 1 & -1\end{bmatrix} \otimes \begin{bmatrix}a^2 & ab/2 & a^2 & ab/2\\ ab/2 & b^2/4 & ab/2 & b^2/4\\ a^2 & ab/2 & a^2 & ab/2\\ ab/2 & b^2/4 & ab/2 & b^2/4\end{bmatrix} \tag{3-23}$$

相应的，反变换公式可写为

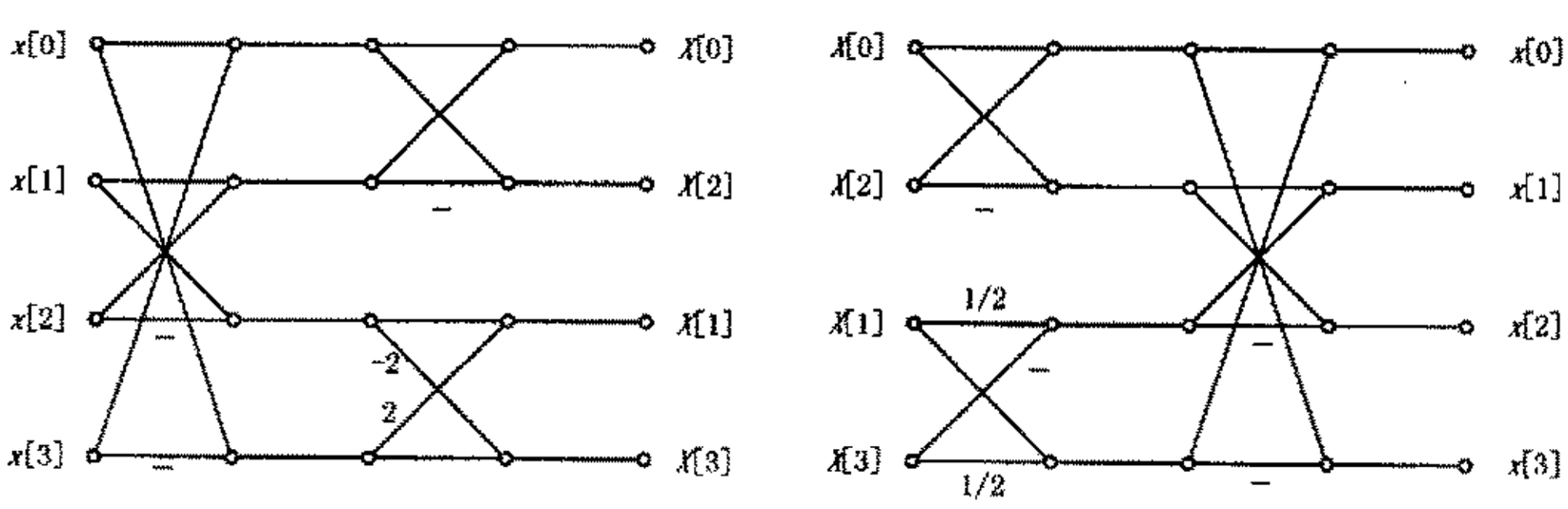
$$X = \begin{bmatrix}1 & 1 & 1 & 1/2\\ 1 & 1/2 & -1 & -1\\ 1 & -1/2 & -1 & 1\\ 1 & -1 & 1 & -1/2\end{bmatrix}\begin{bmatrix}y_{00} & y_{01} & y_{02} & y_{03}\\ y_{10} & y_{11} & y_{12} & y_{13}\\ y_{20} & y_{21} & y_{22} & y_{23}\\ y_{30} & y_{31} & y_{32} & y_{33}\end{bmatrix}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1/2 & -1/2 & -1\\ 1 & -1 & -1 & 1\\ 1/2 & -1 & 1 & -1/2\end{bmatrix} \tag{3-24}$$

利用尺度 DCT 的思想，最后的尺度变换矩阵 E 可以结合到量化过程中，便得到了 H.264 的核心 4×4 整数变换和反变换算法。新的整数变换算法变换核和反变换核为

$$Q_D = \begin{bmatrix}1 & 1 & 1 & 1\\ 2 & 1 & -1 & 2\\ 1 & -1 & -1 & 1\\ 1 & -2 & 2 & -1\end{bmatrix} \qquad Q_I = \begin{bmatrix}1 & 1 & 1 & 1/2\\ 1 & 1/2 & -1 & -1\\ 1 & -1/2 & -1 & 1\\ 1 & -1 & 1 & -1/2\end{bmatrix} \tag{3-25}$$

二、运算步骤

为了简化运算步骤，便于软硬件实现，H.264 中通过蝶形结构进行整数变换和反变换，这样变换仅仅通过简单的加法和移位运算便可以完成。计算的流程图参见图 3.5，(a)是整数变换流程，(b)是整数反变换流程。



(a) 整数变换　　(b)整数反变换

图 3.5　H.264 整数变换蝶形结构图

对于 4×·4 的像素块，作整数变换时定义进行变换的一行或一列的元素为，变换后得到变换系数为 $X[0],X[1],X[2],X[3]$；解码时定义反量化后得到的变换系数矩阵一行（或一列）变换系数为 $X[0],X[1],X[2],X[3]$，反变换后得到的结果为 $x_r[0],x_r[1],x_r[2],x_r[3]$。使用图 3.5 的蝶式做整数变换及其反变换的运算步骤如表 3.1 所示。

表 3.1 16 位整数变换运算过程

类别	输入值	中间步骤	输出结果
正变换	$x[0],x[1]$ $x[2],x[3]$	$M[0]=x[0]+x[3]$ $M[3]=x[0]-x[3]$ $M[1]=x[1]+x[2]$ $M[2]=x[1]-x[2]$	$X[0]=M[0]+M[1]$ $X[2]=M[0]-M[1]$ $X[1]=M[2]+(M[3]<<1)$ $X[3]=M[3]-(M[2]<<1)$
反变换	$X[0],X[1]$ $X[2],X[3]$	$M[0]=x[0]+x[2]$ $M[1]=x[0]-x[2]$ $M[2]=(M[1]>>1)-M[3]$ $M[3]=(M[3]>>1)+M[1]$	$x_r[0]=M[0]+M[3]$ $x_r[3]=M[0]-M[3]$ $x_r[1]=M[1]+M[2]$ $x_r[2]=M[1]-M[2]$

其中 $M[0],M[1],M[2],M[3]$ 为一维变换所生成的中间变量，<<代表左移 1 位，>>代表右移 1 位。反变换后再分别对 $x_r[0],x_r[1],x_r[2],x_r[3]$ 做移位操作，即可得到最终的重构的像素值。

三、整数变换核分析

对于式(3-25)中的变换核和反变换核，Q_D 行和列是正交的，虽然奇数列和偶数列的范数不同，但是仍可在量化过程中加以补偿。对变换核矩阵 Q_D，根据式(3-10)的结论，输入信号的最大动态范围为6倍。做二维变换后，信号增大了 $6^2=36$ 倍。$\log_2 36=5.17$，因此变换后的系数比变换之前要多占用 6 bit。对于 9 bit 的输入像素值，输出结果使用 9+6=15 位即可精确的表示。所以新的整数变换可以通过 16 位算术运算实现。

Q_I 的最大增益为 4 倍，因此二维反变换后信号增大了 $4^2=16$ 倍，共需要 13 bit 来存储，同样可以通过 16 位运算实现而不会溢出。变换核中的系数 2 和 1/2 可以直接通过移位运算加以实现，从而在整个变换过程中避免了使用乘法。

四、量化过程

式(3-14)和式(3-15)的量化虽然简单，但同样需要进行 32 位的运算。为了和 16 位整数变换相适应，我们可以对量化过程做进一步的化简，使量化过程也可以通过 16 位的运算来实现。H.264 量化过程如下。

量化公式为：

$$X_q(i,j) = sign\{X(i,j)\}\left[\left(|X(i,j)|A(Q_M,i,j) + f \cdot 2^{17+Q_E}\right) >> (17+Q_E)\right]$$

$$i,j = 0,1,2,3 \qquad (3\text{-}26)$$

反量化公式为：

$$X_r(i,j) = X_q(i,j)B(Q_M,i,j) << Q_E, \qquad i,j = 0,1,2,3 \qquad (3\text{-}27)$$

其中 $Q_M = QP\%6$， $Q_E = QP/6$， QP 是量化系数， $X(i,j)$ 是整数变换系数， $X_q(i,j)$ 是最终的量化结果， $X_r(i,j)$ 代表反量化后的输出。

在 16 位整数变换中，变换核 Q_I 和反向变换核 Q_D 之间有式(3-31)所示的关系。可以看出奇数列与偶数列扩大的尺度不同。

$$Q_I \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} Q_D = \begin{bmatrix} 20 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 20 \end{bmatrix} \qquad (3\text{-}28)$$

观察式(3-23)中的尺度矩阵 E 可以发现，不同位置的变换系数扩大了不同的倍数，共有 3 种情况，分别是 4^2，5^2 和 4×5＝20 倍。考虑到这一点，H.264 中设计了新的量化表。在量化和反量化中使用 $QP\%6$ 来选择量化步长，一方面可以减小量化表长度，节省了存储空间；另一方面仍然可以保持量化系数 QP 和图像 PSNR 良好的线性关系。量化表 A 反量化表 B 的计算参见式(3-29)和式(3-30)。

$$A(QP\%6,i,j) = M(QP\%6,r), \quad B(QP\%6,i,j) = S(QP\%6,r) \qquad (3\text{-}29)$$

$$M = \begin{bmatrix} 13107 & 5243 & 8066 \\ 11916 & 4660 & 7490 \\ 10082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \\ 7282 & 2893 & 4559 \end{bmatrix}, \quad S = \begin{bmatrix} 10 & 16 & 13 \\ 11 & 18 & 14 \\ 13 & 20 & 16 \\ 14 & 23 & 18 \\ 16 & 25 & 20 \\ 18 & 29 & 23 \end{bmatrix} \qquad (3\text{-}30)$$

为了对不同的变换尺度加以补偿，量化表中定义 r 用于根据系数的位置选择所使用的量化步长。设 4×4 矩阵左上角的位置为(0,0)，则 r 的取值如表 3.2 所示。

表 3.2 r 的取值

r	系数位置
0	(0,0),(0,1),(1,0),(1,1)
1	(0,2),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1)
2	(2,2),(2,3),(3,2),(3,3)

最终像素值经过整数变换、量化、反量化、反变换解码后，增加了 6 位，因此

在重建时需要对结果值在进行 6 位的右移操作。

3.4.2 亮度分量 DC 系数变换

为了进一步提高压缩比，当宏块以 16×16 帧内预测模式进行编码时，随后宏块中的每个 4×4 子块的 DC 系数还被提取出来组成 4×4 的 DC 系数矩阵，再做一次整数变换。由于 DC 系数数值较大且通常较为接近，因而此处的整数变换算法使用 Hadamard 变换。

设亮度 DC 系数矩阵为 x_D，Y_D 为变换后的输出结果，量化后的结果用 Y_{QD} 表示，最终反变换的结果是 W_{QD}，则二维变换为

$$Y_D = CX_DC^T =$$

$$\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}\begin{bmatrix}x_{D00} & x_{D01} & x_{D02}, & x_{D03}\\ x_{D10} & x_{D11} & x_{D12} & x_{D13}\\ x_{D20} & x_{D21} & x_{D22} & x_{D23}\\ x_{D30} & x_{D31} & x_{D32} & x_{D33}\end{bmatrix}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix} >> 1 \tag{3-31}$$

量化使用如下公式：

$$Y_{QD}(i,j) = \left[Y_D(i,j)\cdot Q(Q_M,0,0) + 2\cdot f\right] >> \left(18 + Q_E\right), \quad i,j = 0,...3 \tag{3-32}$$

反量化公式为

$$X_D(i,j) = \left[X_{QD}(i,j)\cdot R(Q_E,0,0)\right] >> 2, \quad i,j = 0,...3 \tag{3-33}$$

反变换使用如下公式：

$$W_{QD} = \begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}\begin{bmatrix}y_{00} & y_{01} & y_{02} & y_{03}\\ y_{10} & y_{11} & y_{12} & y_{13}\\ y_{20} & y_{21} & y_{22} & y_{23}\\ y_{30} & y_{31} & y_{32} & y_{33}\end{bmatrix}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix} \tag{3-34}$$

3.4.3 色度分量 DC 系数变换

编码时每个色度分量子块的 DC 系数也被抽取出来做单独的变换，对于 2×2 的色度分量 DC 系数矩阵变换如下：

$$Y_D = \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}x_{D00} & x_{D01}\\ x_{D10} & x_{D11}\end{bmatrix}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \tag{3-35}$$

其中 W_D 为当前的色度分量 DC 系数矩阵，Y_D 为变换后的结果。

量化使用如下公式

$$Y_{QD}(i,j) = \left[Y_D(i,j)\cdot Q(Q_M,0,0) + 2\cdot f\right] >> \left(18 + Q_E\right), \quad i,j = 0,1 \tag{3-36}$$

反量化公式为

$$X_D(i,j) = \left[X_{QD}(i,j) \cdot R(Q_M,0,0)\right] >> 1, \quad i,j = 0,1 \tag{3-37}$$

反变换为

$$X_{QD} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} Y_{QD00} & Y_{QD01} \\ Y_{QD10} & Y_{QD11} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3-38}$$

3.4.4 整数变换算法分析

通过以上对 DCT 和整数变换的分析，我们可以看出 H.264 中所采用的整数变换方案具有如下优点：

(1) 由于变换采用了更小的像素块进行变换，相比于 8×8 点 DCT 变换编码来说，运动估计精度更高，并且能够降低图像的块效应，从而获得更好的图像质量和主观视觉效果。

(2) 整数变换、量化计算及其逆过程可以通过整数运算实现，比原先的浮点运算能有效提高计算速度，也更利于硬件实现实时系统。

(3) 由于是整数变换，运算结果精确度高，且不存在浮点运算及取整，因而可以有效的避免反变换误匹配问题。

(4) 尺度运算被结合到量化过程中进行，进一步降低了整数变换的复杂度。

(5) 新的 16 位整数变换更加简洁，只需要使用 16 位的算术运算，且无须使用乘法运算；减少了变换后的动态范围，从而降低了对存储器和处理器的要求。

表 3.3 中列出了单个像素使用 4×4 整数变换计算量的对比结果。这里两种整数变换都使用蝶形结构进行计算。从中可以看出，在整个的变换编码中，H.264 中的 16 位整数变换算法大大减少了变换所需的计算步骤，且不使用乘法运算。

表 3.3 单个像素整数变换计算量对比

整数变换	H.26L				H.264			
	加法	乘法	移位	运算位数	加法	乘法	移位	运算位数
变换	4	3	—	32	4	—	1	16
量化	—	1	—	32	—	1	—	32
反量化	—	1	—	32	—	1	—	16
反变换	4	3	—	32	4	—	1	16
尺度变换	1	—	1	32	1	—	1	16

3.4.5 实验结果

一、复杂度比较

本节通过具体实验来比较不同变换算法的运算速度。因为实验只是对 DCT 变换算法进行测试，因此选取了 6 幅 512×512 像素的标准静止图像 Lena、Barbara、Man、Goldhill、Baboon 作为测试对象，分别采用 8×8 的直接公式变换法，H.26L 中的整数 DCT 变换方案以及 H.264 中的整数变换算法做变换，得出每种算法进行变换的平均运算时间，实验结果如表 3.4 所示。

表 3.4 单幅图像变换运算时间（单位：ms）

测试图像	运算时间		
	DCT	H.26L	H.264
Lena	2493	145	119
Barbara	2501	141	115
Man	2491	145	119
Goldhill	2490	142	114
Baboon	2489	146	121
平均值	**2493**	**143**	**118**

从表 3.4 的结果中可以看出，采用整数变换算法的计算时间明显少于公式直接计算的方法，计算时间仅仅是它的 5%左右。新的 16 位整数变换算法在原有的基础上又进一步提高了运算速度，平均只需 118ms 即可以对单幅图像完成变换。因此实验证明了整数变换是一种简单易于实现、预算速度快、占用内存小的快速变换算法。

二、精度比较

下面使用 PSNR 作为图像质量的量度，测试两种整数变换算法对图像质量的影响。PSNR 定义为

$$PSNR = 10 \times \lg \frac{MN \cdot x_{\max}^2}{\sum_{i=1}^{M}\sum_{j=1}^{N}(f(i,j) - f'(i,j))^2} \tag{3-39}$$

上两式中，$x_{\max}$ 为原图像中象素的最大灰度值，MSE 为重建图像与原图像之间象素值的均方误差，M、N 分别为图像的长和宽，$f(i,j)$和 $f'(i,j)$ 分别为原图像和重建图像中对应的象素灰度值。

原始的 JM-1.0 中使用 32 位的整数 DCT 算法，我们将验证模型的整数变换编码部分按照新的 16 位整数变换算法加以改写，保留软件的其他部分不变，对两种

算法进行比较。实验中使用 CIF 格式的图像序列 Mobile、Tempete、Paris 和 QCIF 格式的图像序列 Container、Foreman 和 News。使用两种编码模式：inter 模式是正常的编码模式，第 1 帧使用帧内编码模式，后续帧使用帧间编码模式进行编码；intra 模式指对所有帧都使用帧内模式进行编码。实验中将每个图像序列使用两种整数变换编码算法，量化步长从最小值（QP=0）变化到最大值（QP=31），分别计算 PSNR 值。将 H.264 整数变换算法提高的 PSNR 求平均值，最后的结果如表 3.5 所示。

表 3.5 变换算法 PSNR 对比（单位：dB）

序列格式	图像序列名称	编码模式	$\Delta PSNR$
CIF	Mobile	inter	0.043
		intra	0.027
	Tempete	inter	0.049
		intra	0.012
	Paris	inter	-0.050
		intra	0.057
QCIF	Container	inter	0.007
		intra	0.032
	Foreman	inter	-0.031
		intra	-0.001
	News	inter	0.008
		intra	0.008
—	—	平均	0.013

由表 3.5 可以看出，H.264 中的整数变换编码算法和原有的 32 位整数 DCT 变换编码算法相比，图像质量平均提高了 0.013dB。只有在 Foreman 和 Paris 的帧间编码中，新的整数变换算法会降低图像的 PSNR 值，但是由于改变值较小（$\max(\Delta PSNR) = -0.050$），基本可以忽略不计。总体来说，H.264 中的 16 位整数变换算法在性能上和 H.26L 的整数 DCT 性能相当，在一些图像序列中还略有提高。

图 3.6 中使用 3 个 CIF 格式图像序列 Mobile、Tempete 和 Paris 的 QP-PSNR 曲线图来直观的比较两种整数变换的性能。平滑的直线表示使用原始 H.26L 草案中采用的整数变换算法，由带有符号的曲线所表示的是 H.264 采用的的 16 位整数变换编码的 PSNR 平均值。

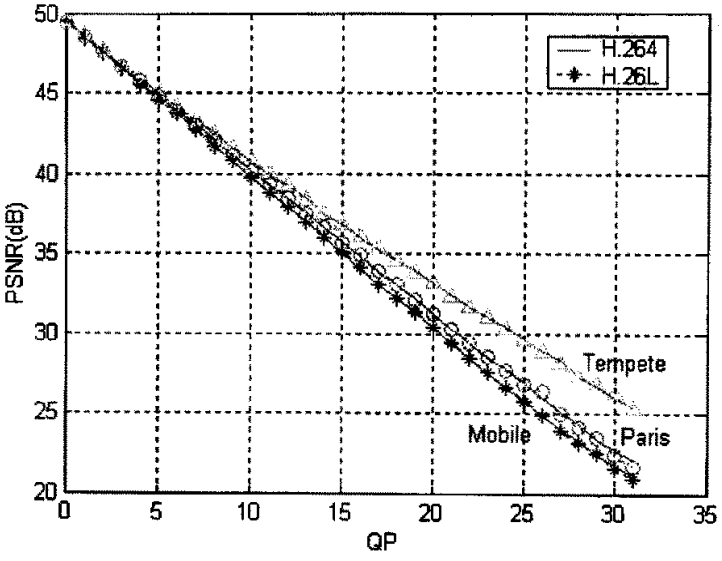


图 3.6 整数变换算法性能比较

从图 3.6 中可以看出，3 个图像序列使用两种整数变换算法在不同的量化系数 QP 下 PSNR 曲线非常接近，说明 16 位整数变换具有良好的性能，完全可以用来代替 H.26L 中的整数 DCT。两种变换算法中 QP 和 PSNR 之间都具有良好的线性关系，有利于使用不同的反馈算法进行码率控制。

三、主观质量比较

图 3.7 显示了 Paris 序列使用两种整数变换算法的重构图像。(a)和(b)是在量化参数 QP=10 分别使用整数 DCT 和 H.264 中的整数变换算法进行编码的恢复图像，(c)和(d)是量化参数 QP=28 时的重构图像。对比同等量化步长的两幅图像可以发现，肉眼几乎观察不到不同算法重构图像的差异，说明 H.264 的 16 位整数变换算法对编码性能几乎没有影响。

QP=10

(a)

(b)

QP=28

(c) (d)

图 3.7 Paris 图像主观质量对比

第四章 帧内预测算法研究

4.1 引言

在以前的视频编码标准如 H.261、H.263、MPEG 1/2 中，帧内编码一般采用的都是先分块再直接进行 DCT 变换、量化和熵编码的方法。由于 I 帧通常包含有大量的信息，且不能利用帧间的相关性，所以编码后仍然需要使用较多的比特数加以表示。事实上，无论是在变换之前还是在变换之后，分割后的子块之间仍然具有一定的相关性。如果能够利用这种相关性，则可以进一步降低 I 帧的比特数，提高编码效率。因此，在 H.263+中的高级帧内编码模式（Advanced Coding Mode）首次引入了帧内预测技术（Intra Prediction），有效提高了帧内编码的压缩比。H.263 系列建议作为多媒体通信领域的一项工业标准被广泛应用于 PSTN 可视电话、电视会议的多媒体终端中。在这些实际应用中，帧内预测被证明是一项简单有效的技术，因此在随后 ISO 和 ITU-T 制定的视频编码标准 MPEG-4 和 H.264 的过程中，对帧内预测技术进行了深入的研究和测试，并将帧内预测技术纳为帧内编码中的一个关键部分。

帧内预测的基本思想是根据相邻块的系数进行预测以得到当前块的预测值，编码时只对实际值和预测值之间的差值进行量化和编码。由于该差值总是小于当前块的原始值，因而预测之后所使用的比特数会得到降低。帧内预测越准确，I 帧压缩比也就越高。根据预测域不同可以将帧内预测分为两种不同类型的算法：频域帧内预测和空域帧内预测。频域帧内预测则利用频域系数子块间的相关性进行预测，H.263+和 MPEG-4 所采用的帧内预测算法都是基于频域的；空域帧内预测使用图像在空域中的原始信息——像素值进行预测，如 H.264 中的帧内预测算法。一般来说，频域帧内预测在频域进行，只针对块内的部分系数（如低频系数）进行预测，运算量小，计算速度快，利于实时编解码，但预测精度有所欠缺；空域帧内预测则相反，使用数目不等的预测选项对子块内部的像素逐点进行预测，预测准确度较高，重构图像质量好，但是庞大的计算量限制了其在实时领域的应用。

本章首先对 H.263+、MPEG-4 和 H.264 的帧内预测算法进行研究，随后针对 H.264 的空域帧内预测算法计算复杂度高的问题，通过分析得出了 H.264 帧内预测中的两条一般性规律：预测选项相关性和 SAD 相关性，在此基础上提出了一种快速自适应空域帧内预测算法（Adaptive Spatial-Domain Intra Prediction Algorithm, ASIP），大幅度提高了预测速度，同时保持了较高的预测精度和主观图像质量。

4.2 频域帧内预测算法

4.2.1 H.263+帧内预测算法

H.263+于 1998 年被 ITU 正式批准，是 H.263 标准的扩充并与之兼容，主要是在 H.263 的可选模式的基础上，以附录的形式增加了新的可选模式和其他一些附加特性，如无限制运动矢量模式（Annex D）、高级帧内编码模式（Annex I）、增强 PB 帧模式（Annex G, Annex M），交替帧间 VLC 编码模式（Annex S）等，其目的是拓宽应用领域、提高压缩效率和错误掩盖能力。

一、基本思想

高级帧内编码模式（Annex I: Advanced Intra Coding Mode）目的是进一步提高 I 帧的压缩比，包含以下 3 项改进的技术：频域帧内预测算法；扩展的扫描和量化方法；针对帧内系数的独立的 VLC 码表。这里所引入的频域帧内预测算法，采用 DCT 系数空间预测以去除帧内冗余信息，提高帧内编码效率。其思想是，在编码中出于对计算复杂度和存储器容量的考虑，一般将图像分隔成为大小固定、相互独立的矩形子块（如 8×8 像素）进行变换。图像的全局性决定了相邻块中各像素值较为相似，也就是说不但块内的各像素点之间具有相关性，子块与子块之间也具有一定的空域相关性。子块的这种空域相关性决定了在 DCT 变换后，相邻块的 DCT 系数也具有相关性，我们称之为子块的频域相关性。频域帧内预测正是利用子块在频域上的这种相关性来对当前块的 DCT 系数进行预测。编码时对预测值和实际值之间的差值进行量化，便可以有效降低 I 帧编码的比特数。

由于 DCT 变换具有将图像的低频信息集中于少数系数中的优良特性，同时随着频率由低向高变化，相邻块的 DCT 系数之间的相关性也随之降低。因此在频域帧内预测中只将 DC 系数、第一行和第一列的 AC 系数看作“重要”系数，帧内预测时只针对该部分系数进行计算，而将其他的 DCT 系数置 0。

帧内预测算法含有 3 种预测选项：DC 预测，垂直 DC 和 AC 系数预测，水平 DC 和 AC 系数预测。当使用 DC 选项时，则只预测当前块的 DC 系数，通常使用上方和左方相邻块的对应 DC 系数进行预测；使用垂直 DC 和 AC 系数预测选项时，DC 和第 1 行的 AC 系数使用上方的相邻块的对应系数进行预测；使用水平 DC 和 AC 系数预测选项时，DC 和第 1 列 AC 系数使用左方的相邻块的对应系数进行预测。

H.263+帧内预测模式的原理如图 4.1 所示。设当前块为块 X，其左方和上方的相邻块分别为块 A 和块 B，则块 X 的 DC 系数可写为 $F_X[0][0]$；同理，块 A、块 B、

块 C 的 DC 系数可以表示为 $F_A[0][0]$，$F_B[0][0]$，预测过程如下：

1. DC 预测

$$F_X[0][0] = Mid(F_A[0][0], F_B[0][0]) \tag{4-1}$$

$$F_X[i][j] = 0, \quad i, j = 1...7 \tag{4-2}$$

2. 水平 DC/AC 预测

$$F_X[i][0] = F_B[i][0], \quad i = 0...7 \tag{4-3}$$

$$F_X[i][j] = 0, \quad i = 0...7,\ j = 1...7 \tag{4-4}$$

3. 垂直 DC/AC 预测

$$F_X[0][j] = F_A[0][j], \quad j = 0...7 \tag{4-5}$$

$$F_X[i][j] = 0, \quad i = 1...7,\ j = 0...7 \tag{4-6}$$

其中 $Mid(\ )$代表求均值运算，在完成上述计算后选择能产生最佳预测的选项作用于当前宏块。

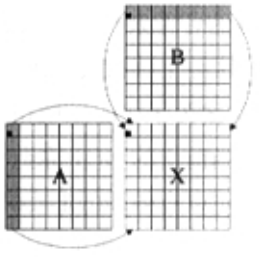


图 4.1 H.263+的帧内预测原理图

根据所选择的预测选项，原始系数减去预测系数后的差值进行不同的量化和扫描。扫描类型共有 3 种：zigzag 扫描、垂直交替扫描、水平交替扫描（见图 4.2）。DC 预测后的扫描使用传统的 zigzag 扫描，垂直 DC/AC 预测后使用水平交替扫描，否则使用垂直交替扫描方式。

进行该种帧内预测编码时，量化电平（LEVEL）较大但零游程（RUN）较短的情况非常普遍，与帧间编码恰恰相反。因此在高级帧内预测模式中定义了新的 VLC 码表，该表通过对帧内宏块进行全局统计后优化获得，能够更好的适应帧内块变换系数分布的特点。在相同的信噪比的情况下，采用此种帧内预测编码可以节省 20%左右的码率。

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	63
36	37	49	50	58	59	62	64

(a) zigzag 扫描方式

1	2	3	4	11	12	13	14
5	6	9	10	18	17	16	15
7	8	20	19	27	28	29	30
21	22	25	26	31	32	33	34
23	24	35	36	43	44	45	46
37	38	41	42	47	48	49	50
39	40	52	52	57	58	59	60
53	54	55	56	61	62	63	64

(b) 水平交替扫描方式

1	5	7	21	23	37	39	53
2	6	8	22	24	38	40	54
3	9	20	25	35	41	51	55
4	10	19	26	36	42	52	56
11	18	35	31	43	47	57	61
12	17	41	32	44	48	58	62
13	16	52	33	45	49	59	63
14	15	55	34	46	50	60	64

(c) 垂直交替扫描方式

图 4.2 H.263+的帧内扫描方式

4.2.2 MPEG-4 帧内预测算法

MPEG-4 中的帧内预测算法称为自适应 DC/AC 帧内预测算法，包括自适应 AC 预测和自适应 DC 预测两部分。帧内预测时首先根据 DC 系数的梯度特性自适应的确定最优预测方向，随后再使用该方向上的相邻块作为预测块进行 DC 预测和 AC 预测。

1. DC/AC 预测方向的选取

自适应 DC/AC 预测通过比较相邻块 DC 系数的梯度以确定最优预测块。图 4.3 是自适应 DC 预测的原理图，其中块 X 代表当前宏块内的待编码块，块 A，块 B，块 C 分别是位于块 X 左方、左上方和上方的相邻块，块 X 和块 Y 位于同一宏块中。

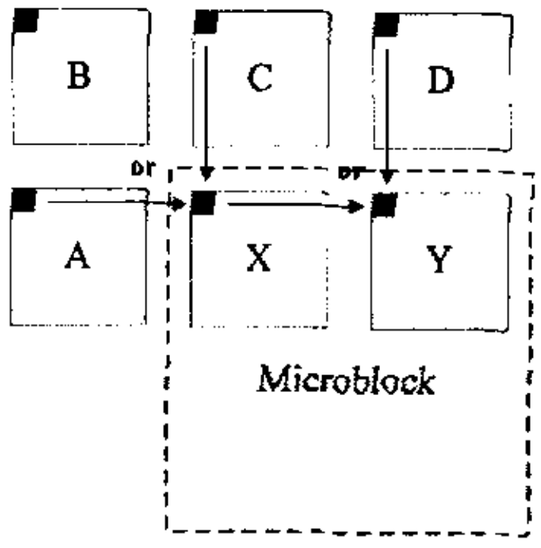


图 4.3 自适应 DC 预测

按照 DCT 系数的位置，设 DCT 变换后的 DC 系数为 $F[0][0]$，则块 X 的 DC 系数可写为 $F_X[0][0]$，块 A、块 B、块 C 的 DC 系数可以表示为 $F_A[0][0]$，$F_B[0][0]$，$F_C[0][0]$。预测方向的判定使用如下准则

$$\textbf{if} \quad |F_A[0][0]-F_B[0][0]|<|F_B[0][0]-F_C[0][0]| \tag{4-7}$$

$$F_X[0][0]=F_C[0][0]$$

else

$$F_X[0][0]=F_A[0][0]$$

如果块 A,B,C 在 VOP 外部或不是帧内编码的宏块，那么预测值为 128。

2. 自适应 DC 预测

自适应 DC 预测只对当前块的 DC 系数进行预测。根据第一步中确定的预测方向，DC 预测的步骤如下：

如果从块 C 进行预测，即预测方向为垂直方向，则

$$QF_X[0][0] = F_C[0][0] / dc\_scaler \tag{4-8}$$

如果从块 A 进行预测，即预测方向为水平方向，则

$$QF_X[0][0] = F_A[0][0] / dc\_scaler \tag{4-9}$$

其中 dc_scaler 是 DC 系数的量化步长。

3. 自适应 AC 预测

自适应 AC 预测根据确定的预测方向有两种预测方式：使用上方块的第一行进行预测；使用左方块的第一列进行预测。DC 预测中的预测方向同样适用与 AC 预测。在宏块内部每个块的预测方向可以互不相同，具有较大的灵活性。自适应 AC 预测的原理图如图 4.4 所示。

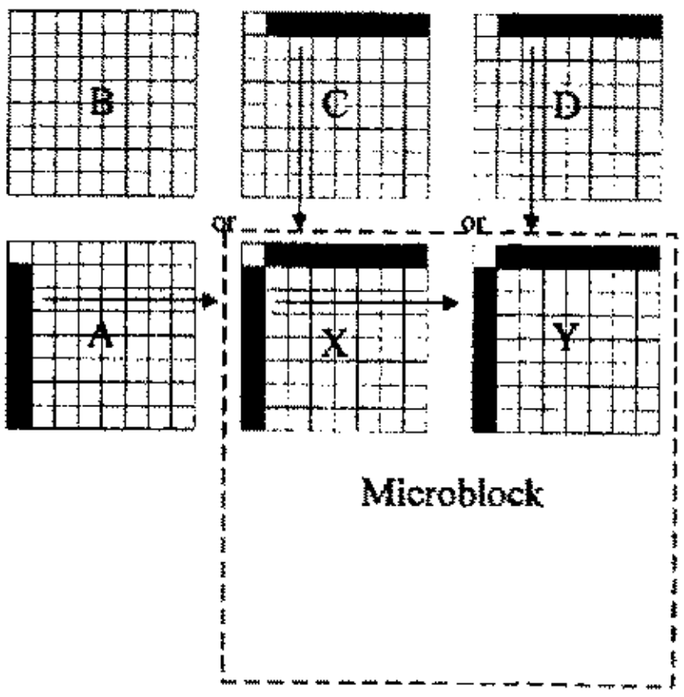


图 4.4 自适应 AC 预测

由于 AC 系数的相关性不如 DC 系数高，因而 AC 预测在某些时候会产生较大的预测误差，这样并没有达到提高压缩比的目的，所以在进行 AC 预测之前需要对预测的有效性进行判断，以便来确定该宏块的 AC 系数是否适合用预测方法得到。这里我们用宏块的预测偏差来确定是否进行 AC 预测。

如果使用块 A 进行预测，定义 S 为

$$S = \sum_{i=1}^{7} |QF_X[i][0]| - \sum_{i=1}^{7} |(QF_C[i][0] \times QP_C)/QP_X - QF_X[i][0]| \tag{4-10}$$

如果使用块 C 进行预测

$$S = \sum_{i=1}^{7} |QF_X[0][i]| - \sum_{i=1}^{7} |(QF_A[0][i] \times QP_A)/QP_X - QF_X[0][i]| \tag{4-11}$$

接下来计算宏块内所有子块的偏差和$\sum S$以便来确定使用 AC 预测：

if $\sum S \geq 0$

ACpred_flag=1　使用 AC 预测

else

ACpred_flag=0　不使用 AC 预测　(4-12)

当满足 ACpred_flag=1 的条件时，进行 AC 预测。由于当前块与相邻块量化步长有可能不同，在 AC 预测中还必须对所使用的 AC 系数加以转换，使之符合当前块的量化参数。

如果从块 C 进行预测，即预测方向为垂直方向，则

$$QF_X[i][0] = (QF_C[i][0] \times QP_C)/QP_X, \quad i=1\ldots7 \tag{4-13}$$

如果从块 A 进行预测，即预测方向为水平方向，则

$$QF_X[0][i] = (QF_A[0][i] \times QP_A)/QP_X, \quad i=1\ldots7 \tag{4-14}$$

其中 QP 代表各块的量化系数。QF_X为自适应 AC 预测结果。当块 A 或块 C 位于 VOP 外部，此时将块内系数全部置零。

当块 A,B,C 在 VOP 外或当不是帧内编码块时，它们相应的预测值设置为 0，并且色差块和亮度块的预测方法相似，都采用和其 DC 预测方向相同的预测。

4.2.3 频域帧内预测算法分析

H.263+和 MPEG-4 帧内预测算法都基于 DCT 系数进行预测。由于 DC 系数代表了图像的低频信息，即图像的主要信息，因而在两种算法 DC 预测选项是不可缺少的。AC 系数是图像的高频信息，代表了图像细节。H.263+还使用了水平 DC/AC 预测和垂直 DC/AC 预测，选取 3 种预测选项中最接近实际值的选项进行编码，以求提高预测精度；MPEG-4 采用了自适应的思想，直接选取最优的预测方向。可以看出，频域帧内预测算法简单可靠、易于硬件实现、运算量小，能够在一定程度上降低 I 帧所占用的比特数。

对一些图像的细节纹理，其 AC 系数尤其时频率较高的 AC 系数相关性很差，使用相邻块的 AC 系数进行预测不可避免地会带来较大的误差；另外频域帧内预测算法使用的第一行或第一列的 AC 系数，分别对应于空域图像的水平高频信息和垂直高频信息，对含有倾斜对角纹理的图像具有先天的局限性。

因而对于视频序列，尤其是一些细节丰富的图像，频域帧内算法不能有效的逼近其实际值，预测误差大，需要寻找更精确的预测算法。

4.3 空域帧内预测算法

在将图像分隔成为相互独立的块后，图像中的物体通常位于相邻区域的数个或数十个子块中。正是由于物体对象的这种全局性，造成了当前块与其邻块的纹理方向往往是高度一致的，且各像素值间相差不大。即不但子块内部的像素具有空间冗余性，子块与子块间也存在空间冗余（如图 4.5 所示）。另一方面，自然场景图像中的前景和背景通常具有一定的纹理特性，按其方向性可以划分为水平纹理、垂直纹理和倾斜纹理等。这两个特性就为空域的帧内预测创造了条件。

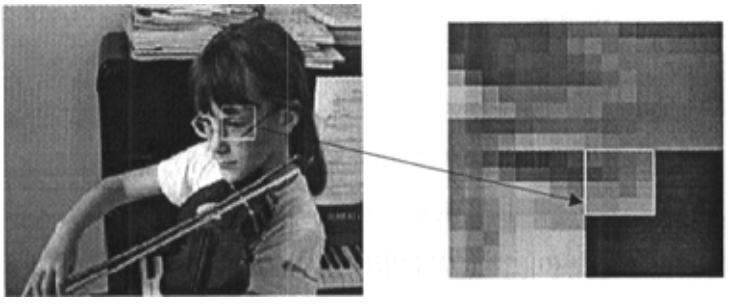


图 4.5 图像空间相关性示意

H.264 的帧内预测算法正是利用图像在空间域上的方向特性及子块像素间的相关性进行预测，去除子块间的空间冗余。编码时，首先使用已经编码的块(尤其是当前块上方和左方的块) 的像素值来预测当前块亮度分量和色度分量值，随后对实际值和预测值的残差图像进行整数变换、量化及熵编码。

H.264 对亮度分量和色度分量设定了不同的预测方案，独立实施预测。亮度分量的帧内预测包含两种预测模式：基于 4×4 像素块的帧内预测模式和基于 16×16 像素块的帧内预测模式；对色度分量只有一种预测模式。在每种预测模式设定了不同方向的预测选项以尽可能的高精度的预测不同纹理特性的图像子块，下面对这 3 种预测模式加以详细分析。

4.3.1 4×4 块亮度分量预测模式

当图像区域中包含细节丰富时，相邻像素点差异往往较大，即空间上的相关性比较小，因此按照以往的算法将图像分割为 8×8 大小的块进行预测误差较大，此时将图像分为更小的块进行预测是一种好的选择。H.264 支持小至 4×4 块的预测，具有更高的预测精度。在该模式中，将图像分割为 4×4 大小的子块，以子块为最小单位进行帧内预测。4×4 亮度分量预测模式含有 9 种预测选项：Mode2 为 DC 预测，其余 8 种预测选项分别代表了 8 种不同方向（参见图 4.6），块内同一方向上的像素点具有相同的预测值，以此来近似的逼近不同方向纹理特性的图像。

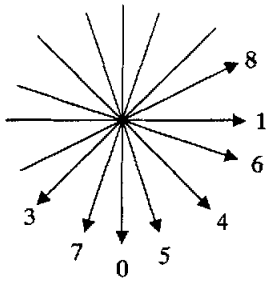


图 4.6 4×4 帧内预测模式

Q	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				

图 4.7 4×4 块各像素点定义

对于图 4.7 所示的 4×4 图像块，a~p 代表子块内部的 16 个像素点，A~L 分别代表上方和左方与子块相邻的 12 个像素点，Q 代表当前块左上方的邻块。9 种预测选项如表 4.1 所示，其中预测角 $\alpha = tg^{-1}1/2$。

表 4.1 4×4 块亮度分量帧内预测算法

预测选项	预测选项名称	预测算法
Mode 0	垂直预测	通过当前预测块正上方的相邻系数进行预测
Mode 1	水平预测	通过当前预测块正左方的相邻系数进行预测
Mode 2	DC 预测	(A+B+C+D+E+F+G+H)/8
Mode 3	左对角预测	以 45°角的方向预测当前块
Mode 4	右对角预测	以-45°角的方向预测当前块
Mode 5	垂直/右对角预测	以 270°+α 的方向预测当前块
Mode 6	水平/右对角预测	以-α 的方向预测当前块
Mode 7	垂直/左对角预测	以 270°-α 的方向预测当前块
Mode 8	水平/左对角预测	以α 的方向预测当前块

令图 4.7 中子块上方的 8 个像素点为 $p(x,-1), x = 0...7$，左方的相邻像素点为 $p(-1,y), y = 0...3$，Q 为 $p(-1,-1)$，当前块的预测结果为 $pred(x,y)$，$x,y = 0...3$。预测算法如下：

1. Mode 0(垂直预测)：该预测模式通过当前预测块正上方的相邻系数进行预测，只有当点 A,B,C,D 都在图像内部时有效。块内各像素点计算值如下：

$$pred(x,y) = p(x,-1),\ x,y = 0...3 \tag{4-15}$$

2. Mode 1(水平预测)：通过当前预测块正左方的相邻系数进行预测，只有当点 I,J,K,L 都在图像内部时有效。计算公式如下：

$$pred(x,y) = p(-1,y),\ x,y = 0...3 \tag{4-16}$$

3. Mode 2(DC 预测)：Mode0 可以被用在所有块的帧内预测中。一般情况下块内所有的像素的预测值均为

$$pred(x,y)=\big(p(0,-1)+p(1,-1)+p(2,-1)+p(3,-1)+p(-1,0)+p(-1,1)+p(-1,2)+p(-1,3)+4\big)>>3 \quad (4\text{-}16)$$

如果 ABCD 或 EFGH 个像素不在图像内部，则使用另外的 4 个像素进行预测，即

$$pred(x,y)=\big(p(0,-1)+p(1,-1)+p(2,-1)+p(3,-1)+2\big)>>2 \quad (4\text{-}17)$$

或

$$pred(x,y)=\big(p(-1,0)+p(-1,1)+p(-1,2)+p(-1,3)+2\big)>>2 \quad (4\text{-}18)$$

如果 A~H 8 个像素都不在图像内部，即当前 4×4 块位于图像的最左上角，这时块内的所有像素 a~p 被置为 128。

4. Mode 3(左对角预测)：以左下 45° 角的方向预测当前块，只有当 A~H8 个像素点都位于图像内部时使用。

   当 $x=3,y=3$ 时

   $$pred(x,y)=\big(p(6,-1)+3\times p(7,-1)+2\big)>>2 \quad (4\text{-}19)$$

   对其它位置的像素点

   $$pred(x,y)=\big(p(x+y,-1)+2\times p(x+y+1,-1)+p(x+y+2,-1)+2\big)>>2$$

   $$x,y=0,1,2 \quad (4\text{-}20)$$

5. Mode 4(右对角预测)：以右下 45° 角的方向预测当前块。当 A~D,I~L8 个像素位于图像内部时本选项可用。

   当 $x>y$ 时

   $$pred(x,y)=\big(p(x-y-2,-1)+2\times p(x-y-1,-1)+p(x-y,-1)+2\big)>>2 \quad (4\text{-}21)$$

   当 $x<y$ 时

   $$pred(x,y)=\big(p(-1,y-x-2)+2\times p(-1,y-x-1)+p(-1,y-x)+2\big)>>2 \quad (4\text{-}22)$$

   当 $x=y$ 时

   $$pred(x,y)=\big(p(0,-1)+2\times p(-1,-1)+p(-1,0)+2\big)>>2 \quad (4\text{-}23)$$

6. Mode 5(垂直/右对角预测)：以垂直偏右 α 角的方向预测当前块。当 A~D,I~L 和 Q 点都位于图像内部，此时本选项有效。首先定义位置标识 $zVR=2x-y$。

   当 $zVR=0,2,4$ 时

   $$pred(x,y)=\big(p(x-(y>>1)-1,-1)+p(x-(y>>1),-1)+1\big)>>1 \quad (4\text{-}24)$$

   当 $zVR=1,3,5$ 时

$$pred(x,y)=\big(p(x-(y>>1)-2,-1)+2\times p(x-(y>>1),-1)+p(x-(y>>1),-1)+2\big)>>2 \quad (4\text{-}25)$$

当 $zVR=-1$ 时

$$pred(x,y)=\big(p(-1,0)+2\times p(-1,-1)+p(0,-1)+2\big)>>2 \quad (4\text{-}26)$$

当 $zVR=-2,-3$ 时

$$pred(x,y)=\big(p(-1,y-1)+2\times p(-1,y-2)+p(-1,y-3)+2\big)>>2 \quad (4\text{-}27)$$

7. Mode 6(水平/右对角预测)：以垂直偏下 α 角的方向预测当前块。A~D,I~L 和 Q 各点都位于图像内部，此时本选项有效。令位置标志 $zHD=2y-x$。

当 $zHD=0,2,4,6$

$$pred(x,y)=\big(p(-1,y-(x>>1)-1)+p(-1,y-(x>>1))+1\big)>>1 \quad (4\text{-}28)$$

当 $zHD=1,3,5$

$$pred(x,y)=\big(p(-1,y-(x>>1)-2)+2\times p(-1,y-(x>>1)-1)+p(-1,y-(x>>1)+2\big)>>2 \quad (4\text{-}29)$$

当 $zHD=-1$

$$pred(x,y)=\big(p(-1,0)+2\times p(-1,-1)+p(0,-1)+2\big)>>2 \quad (4\text{-}30)$$

对于其它情况，如 $zHD=-2$ 或 -3 时，

$$pred(x,y)=\big(p(x-1,-1)+2\times p(x-2,-1)+p(x-3,-1)+2\big)>>2 \quad (4\text{-}31)$$

8. Mode 7(水平/左对角预测)：以水平偏上约 α 角的方向预测当前块。当 A~H 各点都位于图像内部时，该选项有效。

当 $y=0,2$ 时

$$pred(x,y)=\big(p(x+(y>>1),-1)+p(x+(y>>1)+1,-1)+1\big)>>1 \quad (4\text{-}32)$$

当 $y=1,3$ 时

$$pred(x,y)=\big(p(x+(y>>1),-1)+2\times p(x+(y>>1)+1,-1)+p(x+(y>>1)+2,-1)+2\big)>>2 \quad (4\text{-}33)$$

9. Mode 8(垂直/左对角预测)：以垂直偏左约 α 角的方向预测当前块。位置标识变量 $zHU=x+2y$。

当 $zHU=0,2,4$ 时，

$$pred(x,y)=\big(p(-1,y+(x>>1)+p(-1,y+(x>>1)+1)+1\big)>>1 \quad (4\text{-}34)$$

当 $zHU = 1,3$ 时

$$pred(x,y) = \left(p(-1,y+(x>>1)) + 2\times p(-1,y+(x>>1)+1) + p(-1,y+(x>>1)+2) + 2\right) >> 2 \quad (4\text{-}35)$$

当 $zHU = 5$ 时

$$pred(x,y) = \left(p(-1,2) + 3\times p(-1,3) + 2\right) >> 2 \quad (4\text{-}36)$$

以上的 9 种预测选项可以由图 4.8 直观的加以表述。

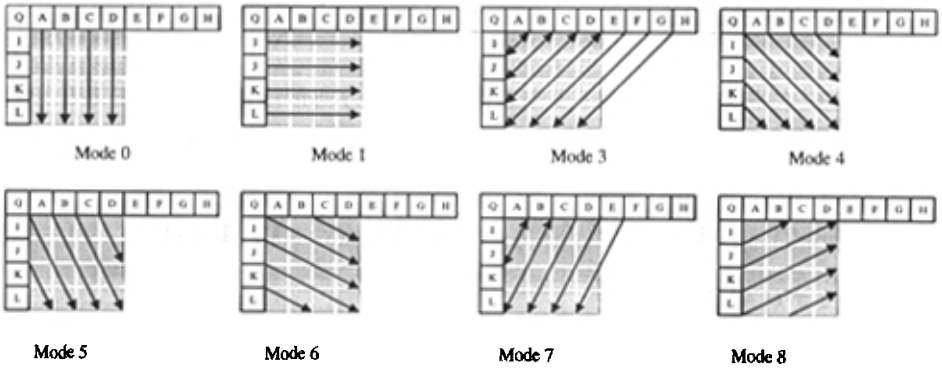


图 4.8 4×4 帧内预测选项方向示意

当采用 4×4 块亮度分量帧内预测模式时，依次使用上述的 9 种预测选项，选取相应的相邻像素点进行预测。对于这 9 种预测选项，使用绝对误差和准则（Sum of Absolute Difference,SAD）作为判决函数来比较预测选项的优劣。编码时选取最接近当前子块实际值的预测结果，并对残差和预测选项进行编码。

4.3.2 16×16 亮度分量预测模式

在图像区域内部较为平坦、包含的图像细节不多的情况下，可以对 16×16 的像素块直接进行帧内预测，而无需进一步分块，从而可以减少帧内预测的计算量。针对亮度分量的 16×16 帧内预测模式，H.264 设定了 3 个预测选项，分别为：垂直预测，水平预测，DC 预测和平面预测（Plane Prediction），如图 4.9。

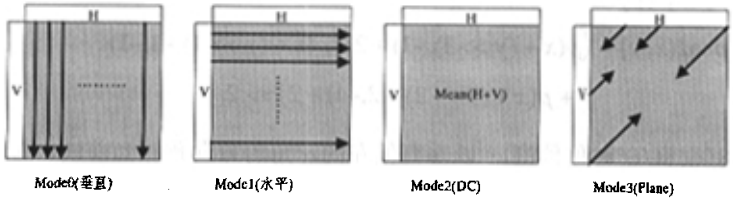


图 4.9 亮度分量的 16×16 块预测模式

设当前块为 $p(x,y)$，$x,y = 0...15$，当前块上方的16个相邻像素点可以表示为 $p(i,-1)$，$x = 0...15$，左方相邻像素点表示为 $p(-1,y)$，$y = 0...15$。$pred(x,y)$，

$x, y = 0...15$，代表使用亮度分量的16×16帧内预测模式得出的预测值。预测的计算公式为：

1. Mode 0（垂直预测）

$$pred(x,y) = p(x,-1), \qquad x,y = 0...15 \tag{4-37}$$

2. Mode 1（水平预测）

$$pred(x,y) = p(-1,y), \qquad x,y = 0...15 \tag{4-38}$$

3. Mode 2（DC 预测）：

$$pred(x,y) = ((\sum_{i=0}^{15}(p(-1,y) + p(x,-1))) + 16) >> 5$$

$$x,y = 0...15 \tag{4-39}$$

如果上方或左方的相邻点位于图像外部，则只使用图像内的相邻像素点进行预测，如果都不在图像内部，此时将块内所有像素点全部置为 128。

4. Mode 3（平面预测）

对上方和左方的相邻像素点，定义

$$H = \sum_{x=0}^{7}(x+1)(p(8+x,-1) - p(6-x,-1)) \tag{4-40}$$

$$V = \sum_{y=0}^{7}(y+1)(p(-1,8+y) - p(-1,8-y)) \tag{4-41}$$

$$a = 16 \times (p(-1,15) + p(15,-1)) \tag{4-42}$$

$$b = (5 \times H + 32)/64 \tag{4-43}$$

$$c = (5 \times V + 32)/64 \tag{4-44}$$

则预测结果为

$$pred(i,j) = (a + b(i-7) + c(j-7) + 16) >> 5, \quad i,j = 0...15 \tag{4-45}$$

4.3.3 色度分量预测模式

除了上面所讨论的针对亮度分量所进行的帧内预测外，H.264 对于色度分量也进行单独的帧内预测。人眼系统的视觉特性（HVS）决定了人眼对色度信号不如亮度信号敏感，在编码时色度分量一般采用较为粗糙的采样和量化方式，因此对色度分量也只需进行低精度的帧内预测即能够满足需求。其帧内预测模式是以 8×8 块为单位进行的，如图 4.10 所示，一个 8×8 像素的色度块（Cb 或 Cr）被分

为 4 个 4×4 像素的色度块 A,B,C,D。色度分量的预测过程与 16×16 亮度分量预测模式非常类似，分为如下 4 种预测选项。

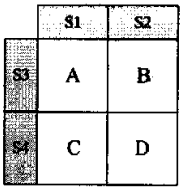


图 4.10 色度块的划分方式

设 8×8 块上方和左方邻块分别为 $p(x,-1)$ 和 $p(-1,y)$，$x,y=0,...,7$。

1. Mode 0（DC 预测）：使用 $p(x,-1)$ 和 $p(-1,y)$，$x,y=0,...,7$ 等像素的均值作为当前块的预测结果。

2. Mode 1（水平预测）：

$$pred_C(x,y)=p(-1,y),\quad x,y=0...7 \tag{4-46}$$

3. Mode 2（垂直预测）：

$$pred_C(x,y)=p(x,-1),\quad x,y=0...7 \tag{4-47}$$

4. Mode 3（平面预测）

令
$$H=\sum_{x=0}^{3}(x+1)(p(4+x,-1)-p(2-x,-1)) \tag{4-48}$$

$$V=\sum_{y=0}^{3}(y+1)(p(-1,4+y)-p(-1,2-y)) \tag{4-49}$$

$$a=16\times(p(-1,7)+p(7,-1)) \tag{4-50}$$

$$b=(17\times H+16)>>5 \tag{4-51}$$

$$c=(17\times V+16)>>5 \tag{4-52}$$

则预测结果为

$$pred_C(i,j)=(a+b(i-7)+c(j-7)+16)>>5,\quad i,j=0,...7 \tag{4-53}$$

4.3.4 空域帧内预测算法分析

H.264 设定了基于 4×4 像素块、基于 16×16 像素块的亮度分量预测算法和独立的色度分量帧内预测算法。为了保证对不同纹理方向图像的预测精度，各预测模式中又详细定义了多种预测选项，4×4 块的帧内预测模式中更是多达 9 种预测选项。在预测时每个 4×4 亮度块都要依次使用 9 种预测选项进行预测，即必须计算 4×4×9=144 种预测选项，得出相应的 SAD 值，再根据 SAD 值确定最优预测

选项。由于详细设定了多种方向的预测选项，对于不同类型的图像此种帧内预测方式都能有效的逼近其真实值，预测精度较频域帧内预测算法有明显提高，图像的主观质量也能够令人满意。实验证明在中高编码率下，H.264 的帧内编码图像质量可以与静止图像压缩标准 JPEG2000 相媲美；在低码率的情况下其质量甚至超过了 JEPG2000。这里面先进的空域帧内预测算法起到了至关重要的作用。

但是 H.264 中过多的预测选项也不可避免的使得帧内编码的运算量大大增加，在实际应用尤其是一些实时编码的场合中仍不能满足要求。为此，必须寻找一种新的帧内预测算法，降低预测所需的运算量。

4.4 自适应空域帧内预测算法

为了解决 H.264 中 4×4 块帧内预测选项计算量大，不利于实时应用的缺点，我们针对 JM 中的 5 方向帧内预测算法进行研究，详细分析了帧内预测选项的空间分布，首次提出预测选项相关性和帧内预测 SAD 空间相关性规律，在此基础上提出了一种适用于 H.264 的快速自适应空域帧内预测算法（Adaptive Spatial-Domain Intra Prediction Algorithm, ASIP）。该算法在预测过程中还采用了“半步停”（half-way stop）技术，设定阈值灵活中止计算，能够进一步提高帧内预测的运算速度。

4.4.1 5 方向帧内预测算法

事实上，JVT 也注意到空域帧内预测计算复杂度高、不利于实时应用的问题。在 ITU 给出的 H.264 验证模型 JM 中，其帧内预测算法在 4×4 块亮度分量帧内预测模式中并没有使用标准草案所规定的全部 9 种预测选项，而是选取了在预测中使用频率最高的 5 种预测选项，组成一种简化的帧内预测模式，我们称之为 5 方向帧内预测算法。该预测算法包含如下 5 种预测选项：DC 预测，垂直/对角预测，垂直预测，对角预测，水平预测和水平/对角预测，各选项预测方向如图 4.11 所示。JVC 在报告中证明，使用简化的预测算法能够将运算量减小 30%左右，同时预测的质量下降不多。对于图 4.12 的 4×4 子块，预测计算公式如下

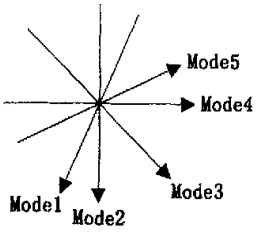


图 4.11 JM 中采用的预测选项

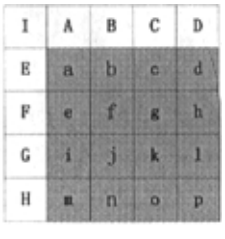


图 4.12 4×4 子块

1. Mode 0：块内的像素点全部置为(A+B+C+D+E+F+G+H)/8 (4-54)

2. Mode 1： a=(A+B)/2; e=B; b, i=(B+C)/2; f, m=C;
c, j=(C+D)/2; d, g, h, k, l, n, o, p=D; (4-55)

3. Mode 2： a, e, i, m=A; b, f, j, n=B;
c, g, k, o=C; d, h, l, p=D; (4-56)

4. Mode 3： m=(H+2G+F)/4; i, n=(G+2F+E)/4;
e, j, o=(F+2E+I)/4; a, f, k, p=(E+2I+A)/4; (4-57)

5. Mode 4： a, b, c, d=A; e, f, g, h=B;
i, j, k, l=C; m, n, o, p=D; (4-58)

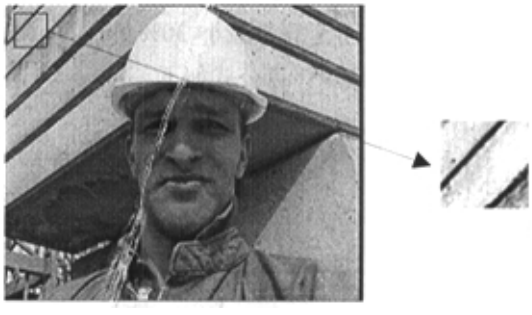
6. Mode 5： a=(E+F)/2; b=F; c, e=(F+G)/2; f, d=G;
i, g=(G+H)/2; h, j, k, l, m, n, o, p=H; (4-59)

5 方向帧内预测算法从减少帧内预测选项的数目的角度出发，选取使用频率最高的 5 种预测选项进行预测，同时简化了计算公式，在一定程度上降低了预测的运算量，是一种较好的折中方案。事实上，无论是原有的帧内预测算法还是 5 方向帧内预测算法，都仅仅从预测选项的角度入手，没用利用帧内预测选项的空间相关性，因而存在进一步改进的余地。

4.4.2 预测选项的空间相关性

前面提到了图像的子块各像素间具有空间相关性，这种相关性使得相互接近的子块的预测选项也十分接近，即：图像的相邻子块之间具有一定的相关性，由此造成进行帧内预测相邻图像子块之间的预测模式通常也具有相关性，

图 4.13(a)是 CIF 格式的标准图像序列 Foreman 中的一帧图像，图 5(b)截取了(a)背景中 48×48 大小的矩形子块；我们对(b)中的子块进行 5 方向帧内预测，按照子块的顺序将各预测选项排列后如(c)所示，0~5 代表 Mode0~Mode5。可以看出绝大部分子块的预测选项与周围的一个或几个邻块是相同的。



2	1	0	4	5	1	1	1	5	1	5	0
2	2	0	0	1	1	5	1	5	5	0	1
1	1	2	1	1	5	1	5	0	2	0	4
2	2	4	1	1	5	2	1	2	2	4	5
2	0	1	1	1	1	1	1	1	5	1	1
5	1	1	0	0	5	0	0	5	5	1	1
4	5	1	1	1	0	2	5	1	1	1	2
1	1	5	1	0	1	1	0	1	4	1	2
0	4	1	2	2	2	0	5	5	2	2	1
2	1	1	4	5	1	1	0	1	0	1	0
0	1	1	5	0	5	1	4	2	2	2	1
5	5	5	1	1	1	5	0	5	0	2	1

(a) Foreman 序列的一帧 (b) 截取的 96×96 子块 (c) 子块的预测选项

图 4.13 帧内预测选项的空间相关性

为了更清楚地说明二者之间的关系，下面选取 6 个典型的 CIF 格式标准图像序

列 Alex, Foreman, Miss America, Tennis, Claire, Mobile，使用 5 方向帧内预测算法对以上序列的前 100 帧进行预测。Alex、Claire 和 Miss America 是典型的头肩序列，图像运动微小平缓；Foreman 背景复杂，且包含镜头的平移和拉伸；Tennis 序列中包含物体的快速运动和场景切换；Mobile 序列中既含有镜头的平移也包括物体的移动，且背景复杂。

表 4.2 中分别列出了预测后子块的预测选项和左方、上方和右上方子块预测选项相同的数目及其所占百分比的平均值。在统计时，表 4.2 将当前块与其上方、左方和右上方的子块预测选项分离，结果中包含了当前块与一个以上的相邻块预测选项相同的情况。当前块与两个或三个相邻块预测选项相同时，各邻块次数分别计算。表 4.3 中列出了忽略各子块间预测选项相同情况下的相关性结果，顺序为首先对上方块进行比较，如果当前块的预测选项与上方块相同，则转向下一子块的比较，否则继续比较左方块和右上方子块。

表 4.2 子块间的相关性（考虑邻块预测选项一致的情况）

图像序列	上方		左方		右上方		不能预测的子块	
	次数	百分比	次数	百分比	次数	百分比	次数	百分比
Alex	3286	51.86	2823	44.55	2787	43.98	1806	28.50
Foreman	1828	30.37	1990	31.41	1792	28.29	2522	39.81
Miss America	2098	33.11	2054	32.43	1951	30.79	2397	37.84
Tennis	1946	30.71	2080	32.94	1786	28.19	2463	38.88
Claire	2023	31.93	1790	29.25	1718	27.11	2514	39.68
Mobile	2063	32.56	1911	30.15	1686	26.61	2460	38.82
平均	**2202**	**34.92**	**2108**	**33.46**	**1953**	**30.83**	**2360**	**37.30**

表 4.3 子块间的相关性（不考虑预测选项一致的情况）

图像序列	上方		左方		右上方		不能预测的子块	
	次数	百分比	次数	百分比	次数	百分比	次数	百分比
Alex	3286	51.86	843	13.31	401	6.33	1806	28.50
Foreman	1828	30.37	1146	19.04	649	10.78	2522	39.81
Miss America	2098	33.11	1169	18.46	668	10.59	2397	37.84
Tennis	1946	30.71	1253	19.78	674	10.63	2463	38.88
Claire	2023	31.93	1158	18.29	640	10.10	2514	39.68
Mobile	2063	32.56	1102	20.39	522	8.23	2460	38.82
平均	**2207**	**35.09**	**1112**	**18.21**	**592**	**9.44**	**2360**	**37.30**

从表 4.2 中可以看出，在一幅图像的 6336 个 4×4 子块中，平均有 34.92%的子

块的预测选项与其上方块一致；33.46%子块的预测选项和左方子块相同；30.83%子块的预测选项等于右上方子块的预测选项；只有 37.8%的子块与相邻块不一致，无法通过相邻块的预测选项得出结果。统计结果证明了前述的结论，即一般的图像序列中子块间预测选项具有较强的相关性。按其邻块相关性从大到小排列分别为：上方子块、左方子块和右上方子块。表 4.3 说明按照上方、左方、右上方的顺序进行预测，平均有 62.17%的子块可以这三步得到其预测选项。

通过对自然图像和 H.264 帧内预测算法结果的分析，我们得出如下规律

规律 1：使用 5 方向帧内预测算法进行计算，当前子块的预测选项和空间相邻块的预测选项具有相关性，且与左方、上方和右上方子块的相关性最强。这种子块预测选项的相关性也是新算法的基础。

4.4.3 SAD 值的空间相关性

设当前块为 X，当前块的预测选项为 $Mode_X$，当前块的某邻块 Y 编码的最优预测选项为 $Mode_Y$。SAD 在某种意义上反映的是块与块之间的相似度。如果当前子块的最优预测模式与其邻块的最优预测模式相同，即 $Mode_X = Mode_Y$，那么使用两子块使用的 SAD 值通常相差不多，即当前子块的 SAD 值与其相邻子块的 SAD 之间具有高阶相关性。为了通过 SAD 有效地对预测选项进行比较，在此我们构造 SAD 相关性因子 β 作为判断块 X 和其邻块 Y 的预测选项是否一致的量度。相关性因子 β 定义为

$$\beta = \frac{SAD - SAD'}{SAD} \tag{4-60}$$

其中 SAD' 是子块 Y 预测所得的 SAD 值，SAD 代表当前块 X 使用选项 $Mode_X$ 预测的 SAD 值。

当 $\beta < 0$ 时，即 $SAD < SAD'$，说明使用该预测选项对当前块进行预测的误差比其邻块还要小，此时就可以认为该邻块的预测选项即为当前块的预测选项。

当 $\beta \geq 0$ 时，其值越接近于 0，说明当前块的 SAD 值与其邻块的 SAD 值越接近，我们就认为预测选项和邻块的预测选项相同的几率就越大。

为了确定相关性因子 β 的值，我们对上述的 6 个典型图像序列的前 100 帧分别使用 5 方向帧内预测算法做预测，计算出当子块与上方、左方和右上方的子块预测选项相同情况下的 β 值，并求出平均值。表 4.4 中列出了各序列 β 值的具体结果。

表 4.4 图像序列的相关性因子

图像序列	上方	左方	右上方	平均
Alex	0.1335	0.2581	0.2977	**0.2298**
Foreman	0.3322	0.3353	0.2685	**0.3120**
Miss America	0.0989	0.0979	0.1235	**0.1068**
Tennis	0.1336	0.1065	0.1727	**0.1376**
Claire	0.3482	0.3986	0.3767	**0.3745**
Mobile	0.1667	0.2174	0.1959	**0.1209**

从表 4.4 的结果中可以看出，相邻子块的 SAD 值与预测选项之间存在数值上的相关性，邻块的预测选项相等意味着两块间的 SAD 相差不多，此时 β 也处于较小范围之内，因此利用 β 可以有效地反映出这种相关性。如在 Alex 序列中，在当前块邻块预测选项相同的情况下，$\beta = 0.2298$ 。其他序列的 β 介于 0.1209 到 0.3120 之间。反之，如果当前子块与其邻块的 β 值满足上述条件，则可以近似认为两邻块的预测选项相同。综上所述，对多个视频序列的统计分析总结出如下规律。

规律 2：使用 H.264 帧内预测算法进行计算，预测选项相同的子块 SAD 值也存在一定的关系，这种关系可以使用相关性因子 β 表达。

通过实验，我们设定阈值 $T \in [0.3,0.5]$。当满足条件 $\beta \le T$ 时，认为当前子块与其邻块的预测选项一致。此时预测的虚警概率和漏报概率能够被控制在合理的范围之内。使用 β 对预测选项进行判断比较，不受视频运动类型的影响，具有较强的自适应性，且能够较为准确的判断出当前块的预测选项，有助于使用较小的代价获得较好的预测效果。

4.4.4 自适应空域帧内预测算法描述

基于子块间预测选项相关性，并考虑预测选项与 SAD 值之间的关系，我们构造了一种新的帧内预测算法，称为自适应空域帧内预测算法（ASIP）。在预测过程中的每一步都使用 SAD 相关性因子 β 对相邻块的预测选项进行判定，满足条件的子块直接跳出预测，有效的提高了预测速度，体现了“半步停”（half-way stop）的思想。对于图 4.14 所示的各子块，当前 4×4 块为 X，块 A,B,C,D 分别为块 X 左方、左上方、上方和右上方的子块，设定阈值为 T。

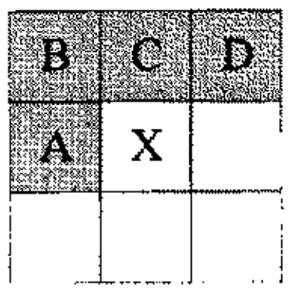


图 4.14 图像内部的子块划分

自适应空域帧内预测算法的具体步骤如下：

Step 1：令当前块的预测选项等于上方子块的预测选项，即 $Mode_X = Mode_C$。计算出预测结果 $f'(x,y)$ 和 SAE_X，使用公式(4-60)得出块 X 和块 C 之间的相关性因子 β。当满足 $\beta \le T$，则认为 $Mode_C$ 就是当前块的最优预测模式，$f'(x,y)$ 是最终的预测结果，结束本块的预测；如不满足上述条件，则转入 Step 2。

Step 2：首先判断块 A 的预测选项是否与块 C 一致，如果 $Mode_A = Mode_C$，说明 *ModeA* 也非最优选项，此时跳转到 Step 3。如果 $Mode_A \neq Mode_C$，令当前块的预测选项等于左方子块的预测选项，即 $Mode_X = Mode_A$。计算出预测结果 $f'(x,y)$ 和 SAE_X，得出此时的相关性因子 β。当满足 $\beta \le T$，则认为 $Mode_A$ 就是当前块的最优预测模式，结束本块的预测；否则转入 Step 3。

Step 3：如果满足 $Mode_D \neq Mode_C$ 且 $Mode_D \neq Mode_A$，令当前块的预测选项等于左方子块的预测选项，即 $Mode_X = Mode_D$。计算出预测结果 $f'(x,y)$ 和 SAE_X，得出此时的相关性因子 β。当满足 $\beta \le T$，则认为 $Mode_D$ 是最优预测模式，结束本块的预测；否则转入 Step 4。

Step 4：使用除 $Mode_A, Mode_B, Mode_D$ 之外剩余的预测选项进行预测，在 5 种预测选项种选择 SAD 值最小的选项，即为最优预测选项。

Step5：结束本块预测。

整个算法的实现流程如图 4.15 所示。

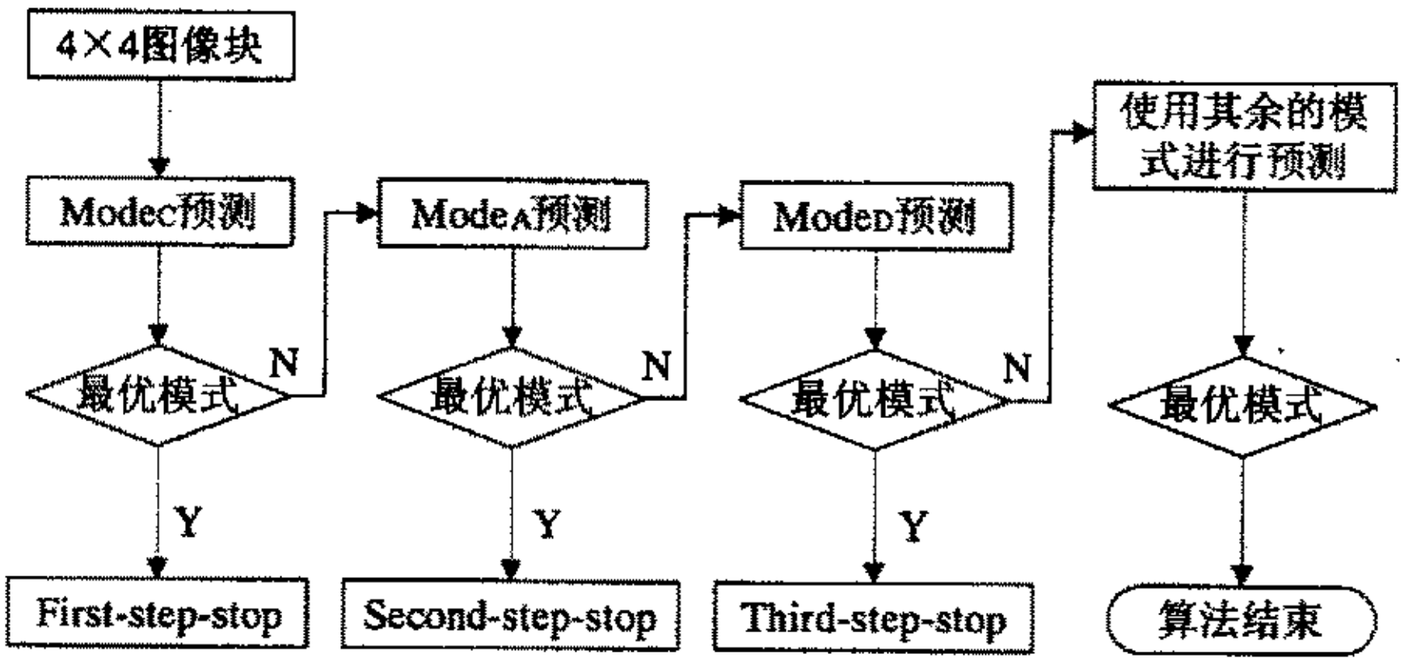


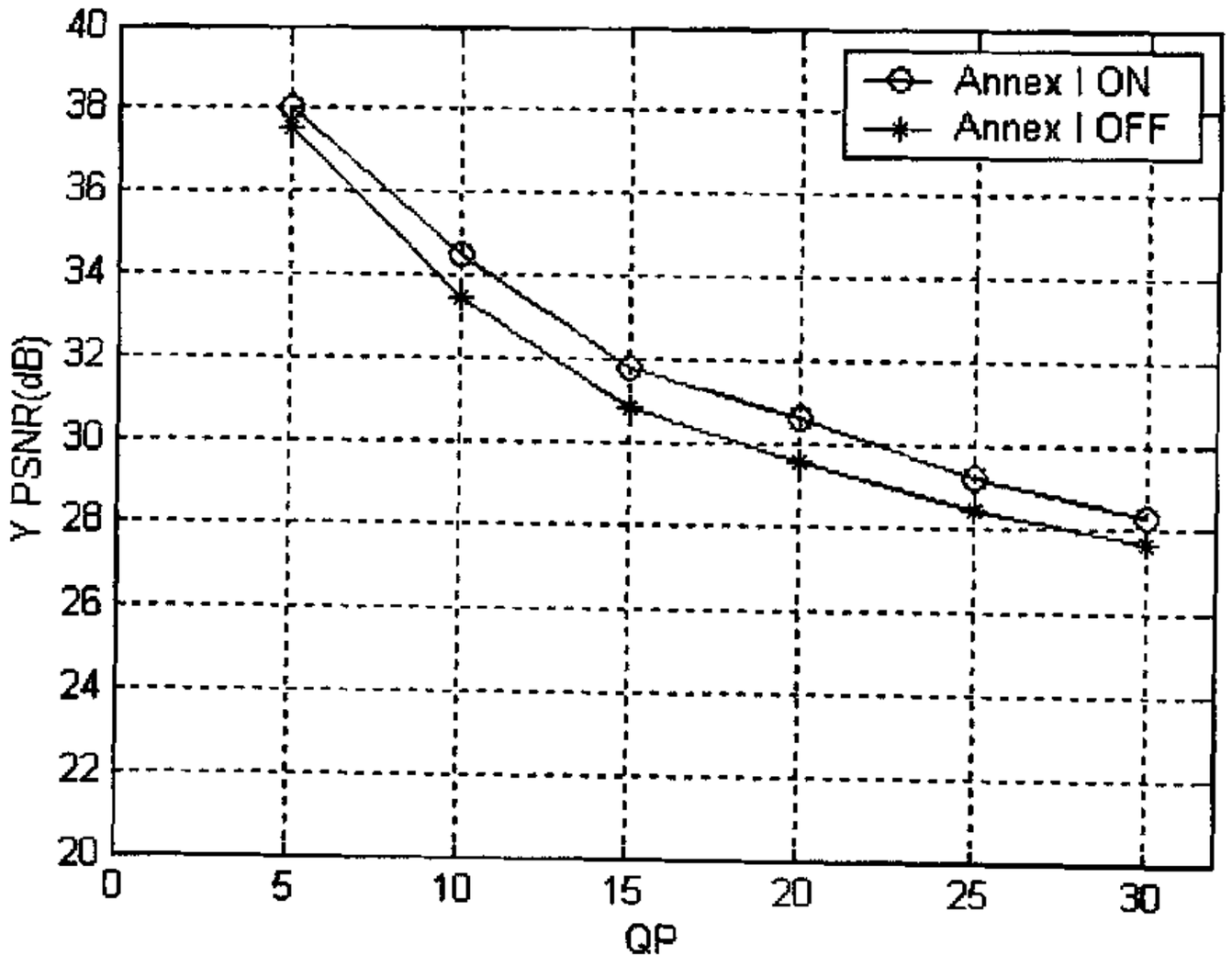
图 4.15 快速帧内预测预测算法流程图

4.5 实验结果与分析

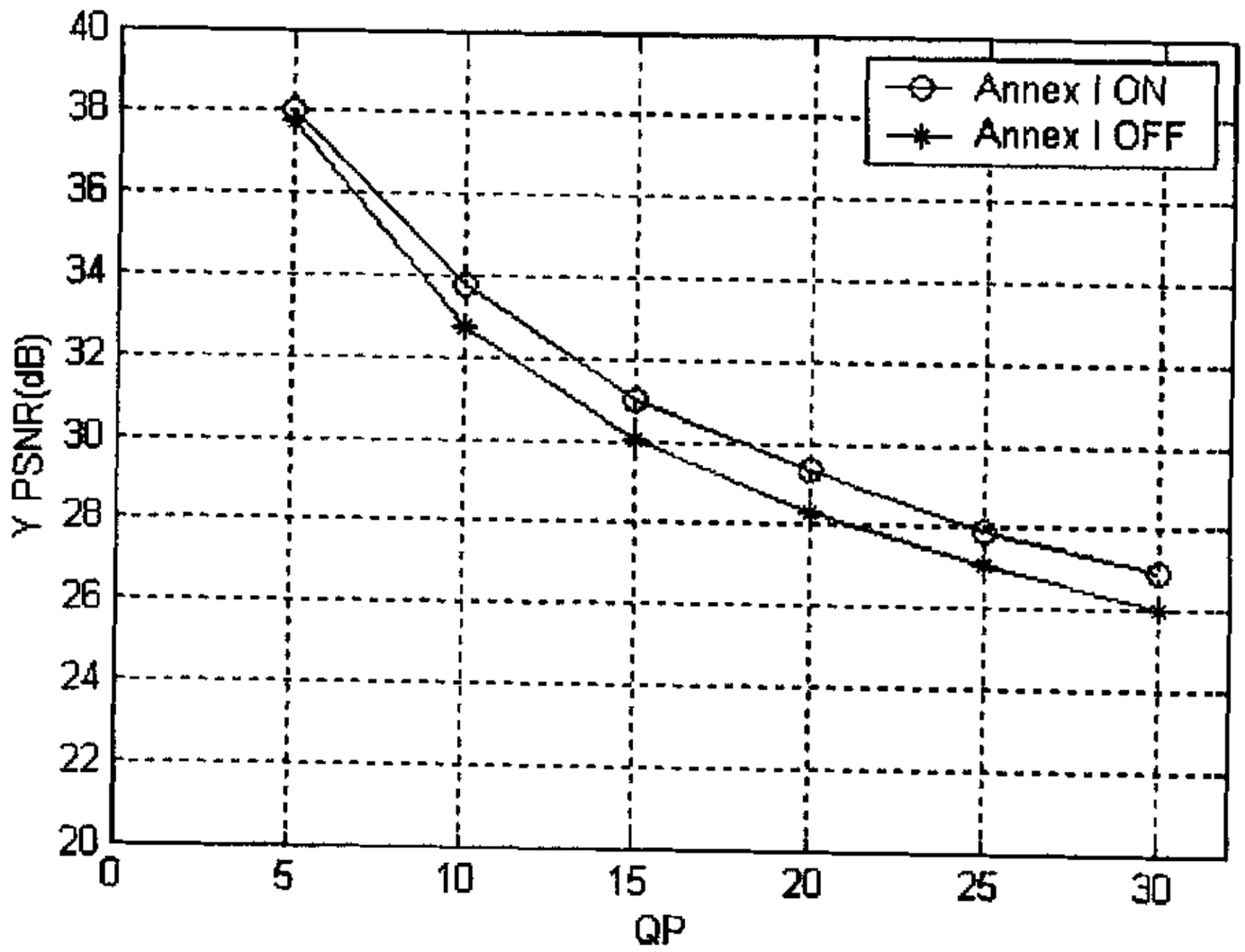
4.5.1 H.263+帧内预测算法性能测试

帧内预测模式是 H.263+的可选项，为了证明帧内预测算法对视频编码的有效性，我们选取 QCIF 格式标准序列 Foreman 和 CIF 格式序列 Paris，使用 H.263+验证模型 TMN-3.0 进行帧内编码。编码时分两种情况：使用帧内预测模式（Annex I ON），不使用帧内预测模式（Annex I OFF）。所有序列的编码速率都为 30 帧/秒，

基于 8×8 象素块的 DCT 变换。其它选项使用 TMN 的默认设置。实验结果对图像的峰值信噪比，在不同量化参数 QP 下比较，如图 4.16 所示。



（a）QIC 序列 Foreman 实验结果



(b) CIF 序列 Paris 实验结果

图 4.16　H.263+帧内预测性能测试

4.5.2 帧内预测算法横向对比测试

在本节中，我们使用 Alex, Foreman, Miss America, Tennis, Claire, Mobile 这 6 个标准图像序列进行分析。实验中选取每个序列的前 60 帧分别使用 H.263+、MPEG-4、H.264 验证模型的 5 方向帧内预测算法和自适应空域帧内预测算法分别进行预测，得出各算法的图像质量和其运算量的情况并加以比较。

1. 帧内预测客观精度比较

我们使用 PSNR 作为图像客观质量的量度比较这几种帧内预测算法的预测精度。PSNR 值越高，说明重构图像与原图像越接近，预测也越为准确。这 4 种预测算法的 PSNR 对比结果如表 4.7 所示。

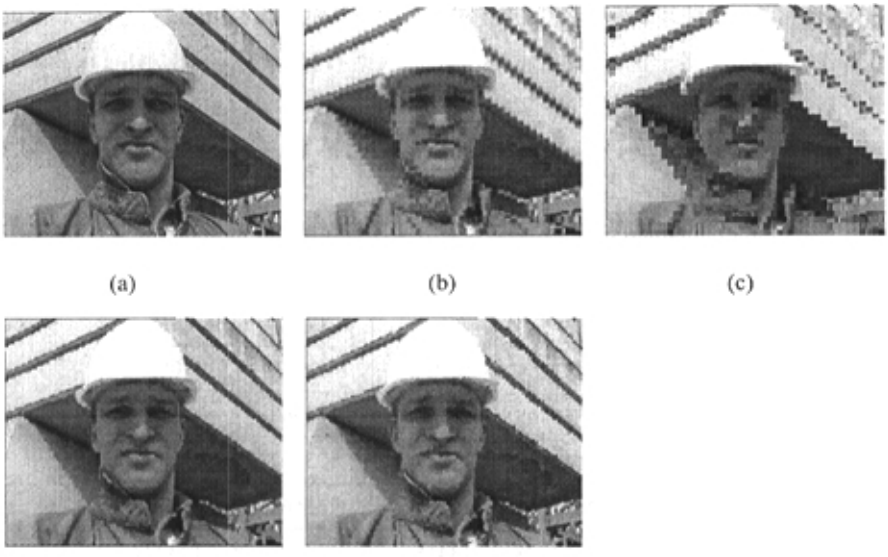
表 4.5 帧内预测算法 PSNR 比较（单位：dB）

图像序列	PSNR			
	H.263+	MPEG-4	JM	ASIP
Alex	25.42	23.65	30.89	29.56
Claire	25.39	23.47	31.89	31.07
Miss America	26.95	24.84	32.97	31.87
Tennis	22.27	21.76	25.04	24.34
Foreman	21.25	20.14	26.04	24.62
Mobile	18.91	15.40	19.21	18.04
平均	**23.37**	**21.54**	**27.67**	**26.58**

从表 4.5 中可以看出，H.263+和 MPEG-4 帧内预测算法精度较低，平均仅为 23.37dB 和 21.54dB。而空域帧内预测算法有显著提高，其 PSNR 值分别达到了 27.67dB。我们所构造的自适应帧内预测算法也达到了 26.58，在 Miss America 序列中达到了 31.87dB，在 Foreman 序列中为 24.62dB，PSNR 平均值较 JM 的算法仅下降了 1.09dB，基本可以满足要求。

2. 重构图像主观质量比较

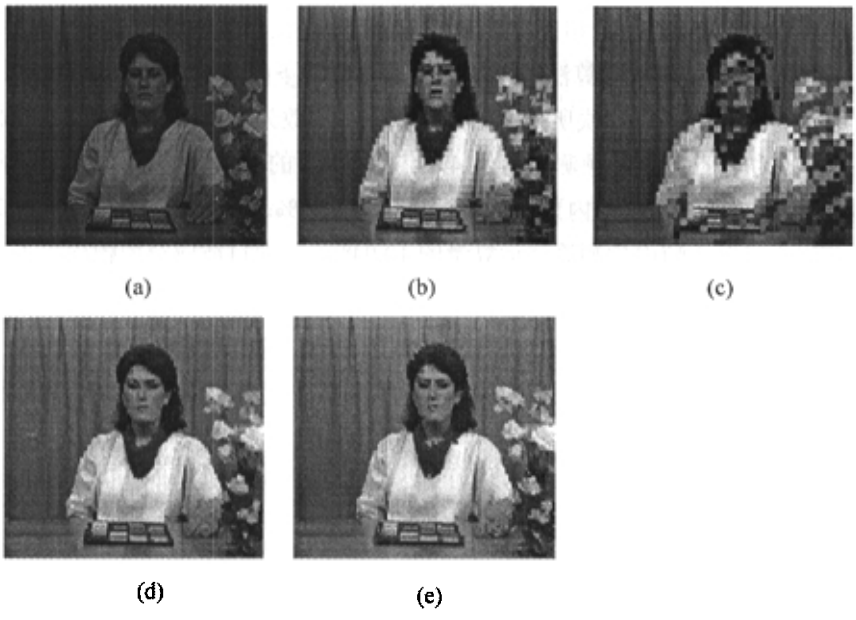
图 4.17 和图 4.18 中显示了 Foreman 和 Alex 两个序列使用不同的帧内预测算法所得到的重构图像。其中(a)是原始图像，(b)~(e)分别是使用 H.263+、MPEG-4、H.264 和 ASIP 预测算法得出的重构图像，对比图像可以看出，H.263+和 MPEG-4 采用的频域预测算法所得出的重构图像块效应明显，图像较为粗糙。空域帧内预测算法如(d)和(e)的重构图像具有较好的主观质量，细节清晰，接近原始图像。而使用 ASIP 算法得出的重构图像总体质量与 JM 算法非常接近，仅有部分边缘细节的锯齿失真有轻微的增大。



(a) (b) (c)

(d) (e)

图 4.17 Foreman 重构图像主观质量比较



(a) (b) (c)

(d) (e)

图 4.18 Alex 重构图像主观质量比较

3. 空域算法运算量比较

运算量是衡量空域帧内预测算法的重要标准。为了准确的比较各算法的运算量，我们使用加乘运算的次数作为衡量帧内预测算法的运算量的量度。在帧内预测中各预测选项的乘法和加法运算都以 2 为单位，实际应用中尤其是使用 DSP 等

硬件实现时，通常使用移位运算来代替乘/除运算以提高运算速度。例如乘 2 运算可以看作将数据左移一位，除 8 可以看作将数据右移 3 位。那么帧内预测的运算量可以更准确的用加法运算和移位运算的数目加以表达。表 4.6 显示了 H.264 帧内预测算法（9 种预测选项）和 ASIP 运算量的对比结果，并给出其每块的平均运算量。

表 4.6 帧内预测算法计算量比较（单位：次）

图像序列	JM		ASIP	
	加法	移位	加法	移位
Alex	2411264	1239803	95589	110039
Foreman	2553408	1317884	136962	149315
Miss America	2610432	1292544	123107	146347
Tennis	2491049	1229178	135561	149593
Claire	2521728	1341856	136283	143227
Mobile	2591424	1279872	126875	144246
平均/块	**399.29**	**202.57**	**19.84**	**22.17**

从表 4.6 中可以看出，相比较于 H.264 中的帧内预测算法和 5 方向帧内预测算法，自适应空域帧内预测算法的计算量有了很大的下降。使用原有的空域帧内预测算法进行预测，平均每块所使用的加法和移位次数为 399.29 次和 202.57 次。使用 ASIP 算法进行预测，预测时每个 4×4 子块需要的平均加法运算的数目降低到 19.84 次，相当于 H.264 帧内预测算法运算量的 4.97%；移位运算数目降低至 22.17 次，相当于 H.264 帧内预测算法运算量的 10.94%，因而自适应空域帧内算（ASIP）法是极为高效的。

第五章 运动估计算法研究

5.1 引言

运动估计（Motion Estimation, ME）是视频编码中所普遍采用的一项核心技术，也是消除视频数据的空间冗余最基本和最重要的方法。运动估计的优劣直接影响到编码的效率和图像恢复质量，各种算法的研究也一直受到学术界和工业界的关注。早在 1997 年，ISO 的 MPEG 组织就公开向全球征集快速有效的运动估计算法。目前的运动估计算法如果按照其匹配域的不同可以分为如下两类：

1. 基于空域的搜索算法
   - 像素递归法
   - 基于对象的匹配算法
   - 块匹配法
2. 基于变换域的搜索算法
   - 相位相关法
   - 小波域匹配算法
   - DCT 域匹配算法
   - 其它变换域匹配算法（如 Harr 变换）

评估运动估计算法的优劣是一项复杂的问题，必须从多方面加以考虑。一般认为图像质量（PSNR）、计算复杂度和存储器要求是其中的 3 个主要因素，一种好的运动估计算法必须在这三个方面做出平衡，以达到最优的综合性能。运动估计的性能可以使用如图 5.1 所示 3 维空间中的点加以表达。

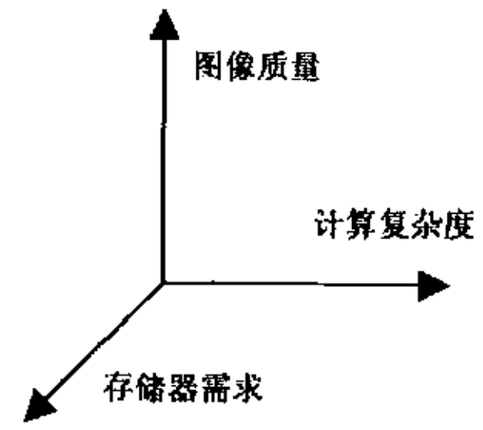


图 5.1 运动估计算法的 3 个考虑因素

像素递归法（Pixel Recursive Algorithm, PRA）[32]的最小搜索单元是像素点，因而对物体的运动有较高的灵敏度，运动估计的精度也高，对复杂运动具有较强的适应性，但是跟踪的范围很小且运算代价大，硬件实现复杂度高，导致其可实时应用性差，不利于一发多收（如 SDTV, HDTV 等）场合的应用。基于对象的匹配算法随着 MPEG-4 的推广而逐渐引起人们的重视，但是由于视频对象（Video

Object)分割算法距离实用尚有一定差距，因而近年来发展缓慢。相位相关法（Phase Correlation Algorithm, PCA）同样具有较高的复杂度，且估计的精确度不高，也没有得到广泛的应用。DCT 域匹配算法使用 DCT 系数中进行搜索匹配，在编码器端只需要做一次 DCT 变换和量化，节省了反变换和反量化的运算过程，使得编解码的运算量基本保持平衡，近年来得到了较广泛的研究。Jarkko Kari 等人提出了一种在 Harr 变换域中进行匹配的搜索算法[33]，也得到了令人满意的效果。

随着小波技术在视频压缩技术中的发展，小波域中的多分辨率运动估计受到越来越多的关注。小波变换是一种全局变换，它最大的特点是能够多分辨地描述图像信号，并将图像信号分解成一组多尺度（Multiscale）的子带图像，分解后的各子带图像相对平稳，易于编码，并且小波变换具有适应人的视觉系统（HVS）的优良特性。早在 1992 年张亚勤和 Zafer S 就在小波域视频编码和多分辨率运动估计方面做出了开创性的研究工作[34]，近年来 Xuguang Yang 和 K Ramchandran 提出一种基于正反双向联合运动估计的小波域运动补偿方案[35]，2002 年向友君提出了一种基于内容的多分辨率运动估计算法（CBMR）[36]，在提高运算效率方面也做了有价值的尝试。

比较而言，块匹配运动估计算法（Block-Matching Algorithm, BMA）由于算法简单高效、额外开销小、易于 VLSI 实现等优点而被包括 H.264 在内的绝大多数视频编码标准所采用。本章首先针对一些典型的块匹配运动估计算法进行研究探讨，在此基础上提出一种新的六边形运动矢量场自适应搜索算法（Hexagon-based Search Algorithm using Motion Vector Field Adaptive Search Technology, HMVFAST)。HMVFAST 搜索速度快，精度高，且能够获得均匀的运动矢量场，利于编码传输。本章在最后通过对比实验证明了该算法的优越性。

5.2 块匹配运动估计算法

块匹配运动估计算法的思想是将当前图像划分为大小相同、互不重叠的子块，并做如下假设：块内的所有像素具有运动一致性，并且只做平移运动，不包含旋转、伸缩；在参考帧的一定范围（称为匹配窗）内按照一定的匹配准则搜索与之最接近的块（称为预测块）。该预测块与到当前块的位移就是运动矢量，预测块和当前块之间的差值称为残差，因而每个原始图像块都可以使用残差块和一个运动矢量表示。预测越准确，意味着残差中的数值越小，编码后所占用的比特数也越少。图 5.2 显示了块匹配算法中宏块、预测块和运动矢量的关系，箭头所指示的即为该块的运动矢量。

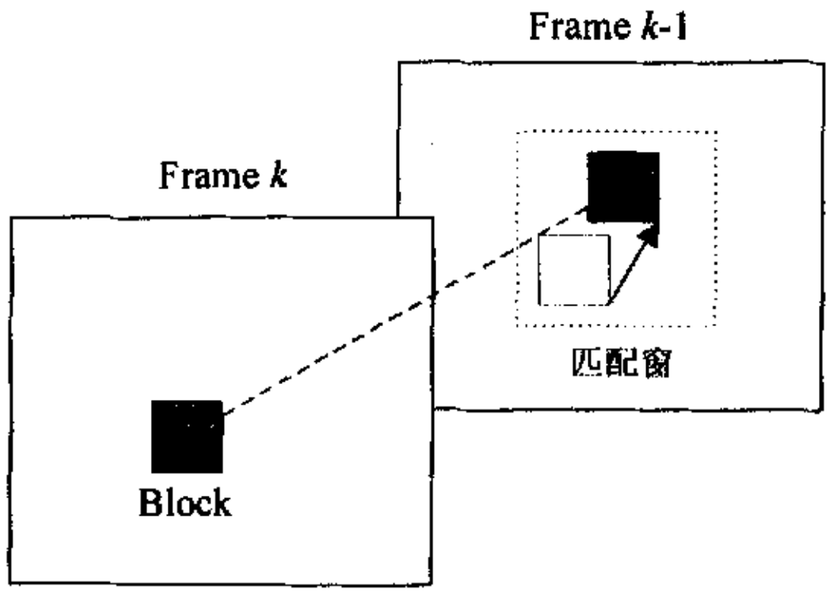


图 5.2 块匹配算法中块与运动矢量的关系

匹配窗的大小通常设定为 $S=(M+2d)\times(N+2d)$，其中 M，N 是宏块的长和宽，d 为垂直和水平方向上的最大位移。例如当宏块的最大偏移矢量介于(-7, 7)时，则搜索范围为 30×30 像素。

在传统的块匹配算法中，块的大小一般取为 16×16，这是一个折衷的选择，因为在宏块大小的选择上存在矛盾：1) 宏块必须足够大，如果太小，很可能发生匹配到有相同像素值但与场景无关的块；并且块小增加运算量，同时也增加了所需传输的运动矢量信息。2) 宏块必须足够小，因为如果在一个块里存在不同运动矢量，匹配块就不能提供准确有效的估计。

5.2.1 块匹配准则

块匹配运动估计算法中依据各种准则函数衡量匹配的准确度，常用的匹配准则有三种，即最小绝对差（MAD）、最小均方误差（MSE）和归一化互相关函数（NCCF）。

1. 最小绝对差

$$MAD(i,j)=\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}|f_k(m,n)-f_{k-1}(m+i,n+j)| \tag{5-1}$$

式中 (i,j) 为位移矢量，f_k 和 f_{k-1} 分别为当前帧和上一帧的灰度值，M×N 为宏块大小，若在某一个点 (i_0,j_0) 处 $MAD(i_0,j_0)$ 达到最小，则该点即为所找的最优匹配点。

2. 最小均方误差

$$MSE(i,j)=\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}[f_k(m,n)-f_{k-1}(m+i,n+j)]^2 \tag{5-2}$$

MSE 值最小的为最优匹配点。

3. 归一化互相关函数

$$NCCF(i,j)=\frac{\sum_{m=1}^{M}\sum_{n=1}^{N}f_k(m,n)f_{k-1}(m+i,n+j)}{[\sum_{m=1}^{M}\sum_{n=1}^{N}f_k^2(m,n)]^{1/2}[\sum_{m=1}^{M}\sum_{n=1}^{N}f_{k-1}^2(m+i,n+j)]^{1/2}} \tag{5-3}$$

这里最优匹配点相对应的是具有 NCCF 的最大值的点。表 5.1 是对一个 M×N 宏块使用三种准则函数的计算复杂度对比结果。

表 5.1 3 种准则函数计算复杂度比较

准则函数	加法	乘法	取绝对值	开方
MAD	$2MN-1$	1	$M\times N$	—
MSE	$3MN-1$	$MN+1$	—	—
NCCF	$3MN-3$	$3MN+2$	—	2

从表 5.1 中可以看出，由于 MSE 准则和 NCCF 准则需要进行乘方运算，在硬件实现中占用资源多，且计算复杂度高。MAD 的所需的加法和乘法次数最少，在上述 3 种匹配函数中计算量最低。在运动估计中，匹配准则对运动估计的精度影响不是很大，为了进一步降低计算复杂度，通常使用绝对误差和（Sum of Absolute Difference, SAD）代替 MAD。使用 SAD 准则不需做乘法运算，实现简单、方便，是软件和 VLSI 实现中最常使用的匹配准则。定义如下：

$$SAD(i,j)=\sum_{m=1}^{M}\sum_{n=1}^{N}|f_k(m,n)-f_{k-1}(m+i,n+j)| \tag{5-4}$$

5.2.2 块匹配搜索算法研究

全搜索法是块匹配运动估计中精度最高的算法，但是其计算复杂度高，不利于实时实现。为此人们提出了一系列快速运动估计算法。本节中我们就几种典型的块匹配运动估计算法进行探讨。

一、全搜索法

1. 算法思想

全搜索法(Full Search Algorithm, FS)也称为穷尽搜索法，是对匹配窗内所有可能的候选位置计算 $SAD(i,j)$ 值，从中找出最小 MAD，其对应偏移量即为所求运动矢量。此算法虽然计算量大，但最简单、可靠，找到的必为全局最优点。

2. 算法描述

Step 1： 从原点出发，按顺时针方向由近及远，在逐个象素处计算 SAD 值，直到遍历搜索范围内所有的点。

Step 2： 在所有搜索点中找到最小块误差 MBD（Mininum Block Distortion）

点（SAD 值最小的点），该点所在位置即对应最优运动矢量。

3. 模板及搜索过程图示

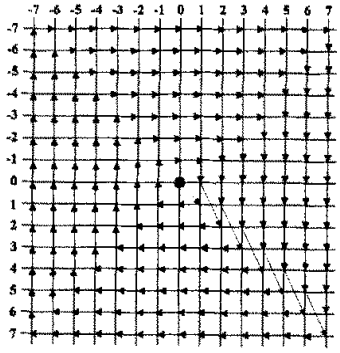


图 5.3 全搜索法图示

4. 算法分析

FS 算法是最简单、最原始的块匹配算法，由于可靠，且能够得到全局最优的结果，通常是其它算法性能比较的标准，但它的计算量非常大，这就限制了在需要实时压缩场合的应用，所以有必要进一步研究其它快速算法。

二、三步搜索法

三步搜索法（Three Step Search，TSS）[37]是 T.KOGA 等人提出的，由于简单、健壮、性能良好的特点，为人们所重视。当最大搜索距离为 7，搜索精度取 1 个像素，则步长为 4，2，1，共需三步即可满足要求，因此而得名三步法。

1. 基本思想

TSS 算法的基本思想是采用一种由粗到细的搜索模式，从原点开始，按一定步长取周围 8 个点构成每次搜索的模板（Search Pattern），进行匹配计算，跟踪最小块误差 MBD 点。

2. 算法描述

Step 1：选取最大搜索长度的一半为步长，在原点周围距离步长的 8 个点处进行块匹配计算并比较。

Step 2：将步长减半，中心点移到上一步的 MBD 点，重新在周围距离步长的 8 个点处进行块匹配计算并比较。

Step 3：在中心及周围 8 个点处找出 MBD 点，若步长为 1，该点所在位置即对应最佳运动矢量，算法结束；否则重复 Step 2。

3. 模板及搜索过程图示

一个可能的 TSS 搜索过程如图 5.4 所示。图中点(4,-4)、(6,-4)分别是第一、第二步的最小块误差点，第三步匹配得到最终运动矢量为(7,-5)。标有数字的点表明了每个阶段中搜索点的位置。

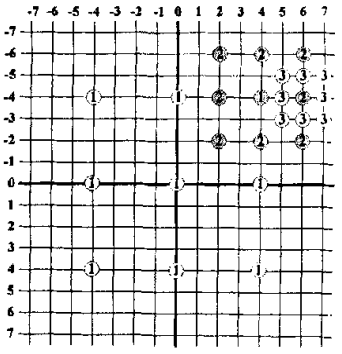


图 5.4 三步法搜索图示

4. 算法分析

使用 TSS 算法搜索时，整个过程采用了统一的 9 点正方形搜索模板。从而使得第一步的步长过大，容易引起误导，从而对小运动效率较低。TSS 最大搜索点数为$1+8\log_2 W$，当搜索范围大于 7 像素时，仅用 3 步是不够的，搜索步数的一般表达式为$\log_2(d+1)$。总体说来，三步法是一种较典型的快速搜索算法，所以得到了较多的研究，后来人们又相继提出了许多三步法的改进算法，提高了它对小运动序列的估计性能。

三、新三步法

新三步搜索法（Novel Three Step Search，NTSS）[38]是 1994 年 R.Li, B.Zeng 和 M.L.Liou 提出的，它是在三步法的基础上所做出的改进算法。

1. 基本思想

NTSS 利用运动矢量的中心偏置分布，采用具有中心倾向的搜索点模式，并应用中止判别技术，减少搜索次数。

2. 算法描述

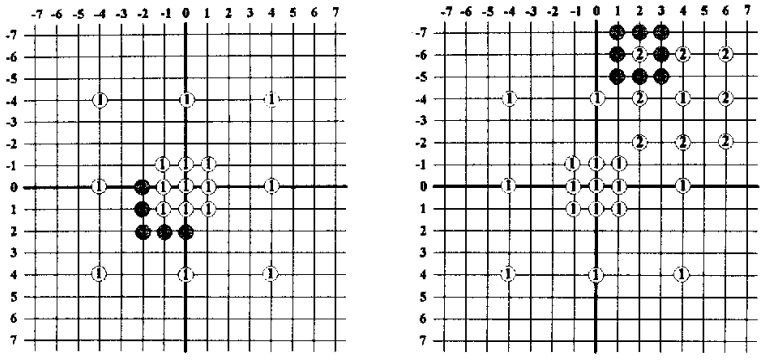
Step 1：搜索模板上的 17 个检测点，如果 MBD 点为搜索窗中心，算法结束；如果 MBD 点位于中心点的 8 个相邻点中，则进行 Step2；否则进行 Step3。

Step 2：以上一步 MBD 点为中心，使用 3×3 搜索窗进行搜索，若 MBD 点在搜索窗中心，则算法结束；否则重复 Step2。

Step 3：执行 TSS 法的 Step2 和 Step3，即将步长减半进行搜索，算法结束。

3. 模板及搜索过程图示

图 5.5 是新三步法的搜索示例。在 (a)中，点(-1, 1)是第一步的 MBD 点，也是第二阶段的 MBD 点，且位于搜索窗中心，故最终运动矢量就是(-1, 1)；在 (b)中，(4, -4)是第一步的 MBD 点，然后以(4,-4)为中心点进行第二步搜索，此时搜索半径已经缩减为 2 像素，最后以当前 MBD 点(2,-6)完成第三步搜索，找到最优匹配点。每个点上的数字表明了不同阶段搜索时的检测点。



(a) (b)

图 5.5 NTSS 搜索示意图

4. 算法分析

运动矢量通常总是高度集中分布在搜索窗的中心位置附近，NTSS 采用中心倾向的搜索点模式不仅提高了匹配速度，而且减少了陷入局部极小的可能性；而采用中止判别技术则大大降低了搜索复杂度，提高了搜索效率。

四、四步搜索法

四步搜索法(Four Step Search, FSS)[39]是 1996 年由 Lai-Man Po 和 Wing-Chung Ma 提出的，该算法类似于三步法，但它基于现实中序列图像的一个特征，即运动矢量都是中心分布的，从而在 5×5 大小的搜索窗口上构造了有 9 个检测点的搜索模板。

1. 基本思想

TSS 算法第一步用了 9×9 的搜索窗，这很容易造成搜索方向的偏离，FSS 算法首先用 5×5 的搜索窗口，每一步将搜索窗的中心移向 MBD 点处，且后两步搜索窗的大小依赖于 MBD 点的位置。

2. 算法描述

Step 1：以搜索区域原点为中心选定 5×5 的搜索窗，然后在 9 个检测点处进行匹配计算，如图 5.6(a)所示，若 MBD 点位于中心点，则跳到 Step 4；否则进行 Step 2。

Step 2：窗口保持为 5×5，但搜索模式取决于上一步的 MBD 点位置：

a) 上一步 MBD 点位于窗的四个角上，则另外再搜索 5 个检测点，如图 5.6 (b)；

b) 上一步 MBD 点位于窗的四边中点处，则只需再搜索 3 个检测点，如图 5.6(c)和图 5.6(d)。

若这一次 MBD 点在窗口中心，则跳到 Step 4；否则进行 Step 3。

Step 3：搜索模式同 Step 2，但最终要进行 Step 4。

Step 4：将窗口缩小 3×3，这时计算出最小误差点的位置即对应最佳运动矢量，

如图 5.6(e)。

3. 模板及搜索过程图示

图 5.7 是 FSS 搜索的一个具体实例，点(0,-2)、(2,-4)、(2,-4)、(3,-5)是每一步所依次搜索的 MBD 点，最终得到的运动矢量为(3,-5)。

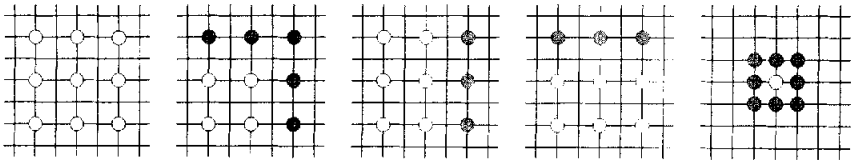


图 5.6 FSS 的搜索模板

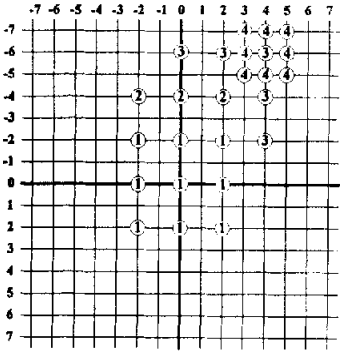


图 5.7 FSS 搜索示意图

4. 算法分析

FSS 是快速搜索算法的又一大进步，它在搜索速度上不一定快于 TSS，搜索范围为±7 时，FSS 最多需要进行 27 次匹配计算。但是 FSS 的计算复杂度比 TSS 低，它的搜索幅度比较平滑，不至于出现方向上的误导，所以获得了较好的搜索效果。而且 TSS 同样适用于像摄像机镜头伸缩、有快速运动物体的图像序列中。因此 FSS 是一种吸引人的运动估计算法。

五、基于块的梯度下降搜索法

基于块的梯度下降搜索法（Block-Based Gradient Descent Search, BBGDS）[40] 是 1996 年由 Lurng-Kuo Liu 和 Ephraim Feig 提出的。

1. 基本思想

BBGDS 算法利用运动矢量中心分布特性，在搜索过程中使用 3×3 搜索窗。视频帧内相邻象素间具有渐变性，每一步的 MBD 点分布具有一定的方向性，也即梯度下降方向。BBGDS 算法使用梯度下降方向来决定下一步的搜索方向。

2. 算法描述

Step 1：使用 3×3 搜索窗对 9 个点进行搜索。

Step 2：若 MBD 点在搜索窗中心，则算法结束；否则以上一步 MBD 点为中心，重复 Step 1。

3. 模板及搜索过程图示

图 5.8 是 BBGDS 搜索的一个具体实例，图中点上的数字表明了该步搜索时所计算的候选点，点(0,-1)、(1,-2)、(2,-3)是四个阶段依次搜索的 MBD 点。最终得到的运动矢量为(2,-3)。

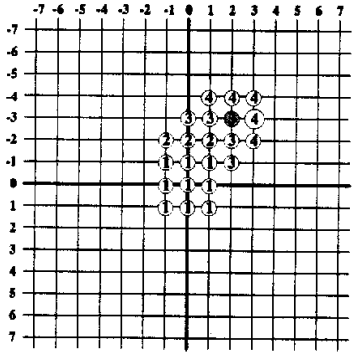


图 5.8 BBGDS 的搜索图示

4. 算法分析

在每一步搜索过程中，BBGDS 算法使用了中心匹配块而不是匹配点，降低了陷入局部最优的可能性。引入梯度下降的概念，用梯度下降方向来指导搜索方向，对该方向进行重点搜索，从而减少和避免了不必要的搜索，降低了算法的复杂度。

六、菱形搜索法

菱形搜索算法（Diamond Search, DS）[41]最早由 Shan Zhu 和 Kai-Kuang Ma 提出，后又经过多次改进，是目前快速块匹配算法中性能较为优异的一种算法。DS 算法曾于 1999 年 10 月被 MPEG-4 国际标准收入验证模型(VM)。

1. 基本思想

搜索模板的形状和大小不但影响整个算法的运行速度，而且也影响它的性能。块匹配的误差实际上是在搜索范围内建立了误差表面函数，全局最小点即对应着最佳运动矢量。由于这个误差表面通常并不是单调的，所以搜索窗口太小，就容易陷入局部最优，例如 BBGDS 算法，其搜索窗口仅为 3×3；而搜索窗口太大，又容易产生错误的搜索路径，象 TSS 算法的第一步。另外，统计数据表明，视频图像中进行运动估计时，最优点通常在零矢量周围。基于这两点事实，DS 算法采用了两种搜索模板，分别是有 9 个检测点的大菱形模板（Large Diamond Search Pattern, LDSP）和有 5 个检测点的小菱形模板（Small Diamond Search Pattern, SDSP），如图 5.9(a)和 5.9(b)所示。搜索时先用大模板 LDSP 计算，当最小块误差 MBD 点出现在中心点处时，将 LDSP 换为 SDSP，再进行匹配计算，这时 5 个点

中的 MBD 即为最优匹配点。

2. 算法描述

Step 1：用 LDSP 在搜索区域中心及周围 8 个点处进行匹配计算，若 MBD 点位于中心点，则进行 Step 3；否则到 Step 2。

Step 2：以上一次找到的 MBD 点作为中心点，用新的 LDSP 来计算，若 MBD 点位于中心点，则进行 Step 3；否则重复 Step 2。

Step 3： 以上一次找到的 MBD 点作为中心点，将 LDSP 换为 SDSP，在 5 个检测点处计算，找出 MBD 点，该点所在位置即对应最佳运动矢量。

3. 模板及搜索过程图示

图 5.10(a)和(b)给出了 LDSP 在水平方向和对角方向进行搜索的图示，图 5.10(c)显示了一个用 DS 算法搜索到运动矢量(3, -2)的例子。整个搜索过程共有 5 步，使用了 3 次 LDSP 和 1 次 SDSP，总共搜索了 24 个点。每个阶段中的 MBD 点分别为(2,0)、(3,-1)、(3,-2) 。

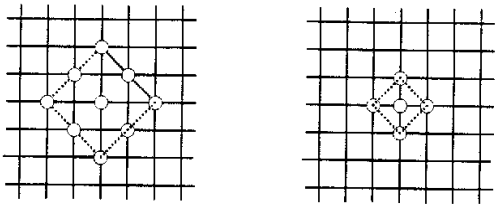


图 5.9 DS 搜索模板 (a) LDSP (b) SDSP

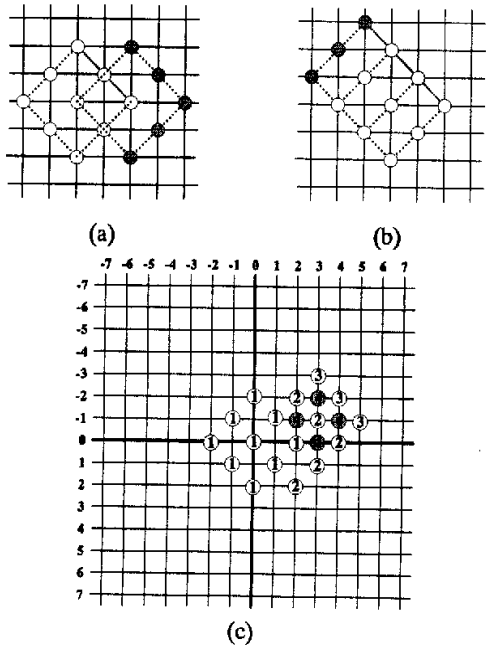


图 5.10 搜索图示 (a)(b) LDSP 搜索 (c) DS 算法

4. 算法分析

DS 算法的特点在于它分析了视频图像中运动矢量的基本规律，选用了大小两种形状的搜索模板：LDSP 和 SDSP。先用 LDSP 搜索，由于步长大，搜索范围广，可以进行粗定位，使搜索过程不会陷于局部最小；当粗定位结束后，可以认为最优点就在 LDSP 周围 8 个点所围的菱形区域中，这时再用 SDSP 来准确定位，使搜索不致于有大的起伏，所以它的性能优于其它算法。另外，DS 搜索时各步骤之间有很强的相关性，模板移动时只需在几个新的检测点处进行匹配计算，所以也提高了搜索速度。

正是由于 DS 算法的这些优良特性，近年来出现了许多基于 DS 的改进算法，如 Chun-Ho Cheung 等人在 2001 年提出的十字形-菱形搜索算法（Cross-Diamond Search, CDS）[42]，Weiguo Zheng 和 Ishfaq Ahmad 等人提出的自适应可伸缩菱形搜索法（Adaptive Motion Search with Elastic Diamond, AMSED）[43]，A.M.Tourapis, O.C.Au 等人提出的高级菱形区域搜索法（Advanced Diamond Zonal Search, ADZS）[44]等，在性能上都获得了不同程度的提高。

七、运动矢量场自适应搜索算法

运动矢量场自适应搜索算法（Motion Vector Field Adaptive Search Technology, MVFAST）[45]是由 Prabhudev Irappa Hosur 和 Kai-Kuang Ma 等人在 1999 年提出的。该算法能够显著提高搜索速度和精度，获得均匀的运动矢量场，而且不需要额外的存储空间来存储搜索点和运动矢量，是目前综合性能最为优异的运动估计算法。在 2000 年 3 月 MPEG Noordwijkerhout 会议上，MVFAST 作为运动估计的核心算法被纳入 MPEG-4 新增的第 7 部分。

1. 基本思想

在视频图像序列中，尤其是视频会议的头肩序列中存在着大量的静止块。对于这些块来说，零矢量(0,0)就是其最优矢量。另外，图像内部的相邻块之间存在着空间相关性。基于这两点考虑，MVFAST 的指导思想是以视频运动的时空相关性为基础，首先通过搜索起点的预测使当前块的初始运动矢量有可能接近其最终运动矢量，其次通过简单有效的视频分类和合适的搜索模式，使其能根据视频运动的类型进行自适应的搜索，最后采用高效的搜索中止准则保证搜索结果在这个预测的起点附近结束时具有足够的精度，从而实现快速、均匀、精度高的运动矢量搜索。

2. 算法描述

MVFAST 结合使用了两种搜索方法：小菱形搜索法（Small Diamond Search, SDS）和普通的菱形搜索法 DS，通过初始点的预测结果选择使用合适的搜索方法完成搜索。如果当前块被判定为小运动或大运动类型，使用 SDS 搜索法；如果当

前块为中等运动类型，此时使用 DS 法。小菱形搜索法（SDS）的搜索过程如下

Step 1：将小菱形模板（SDSP）置于搜索的中心点，检测模板上的 5 个匹配点。如果 SAD 最小值位于模板的中央（没有运动），此时中心点（0, 0）就是当前的运动矢量；否则转入 Step 2

Step 2：将 SDSP 的中心移至上一步的 BDM 点，检测模板上的 5 个匹配点。如果 SAD 最小值位于模板的中央，结束运算；否则继续执行 Step 2。

3. 模板及搜索图示

SDS 搜索模板如图 5.11 所示。图 5.12 显示了使用 SDSP 的搜索过程，共进行了 3 步搜索，每步的 MBD 点分别为 (1,0)、(1,-1)和(2,-1)，最终的最优运动矢量为 (2,-1)。

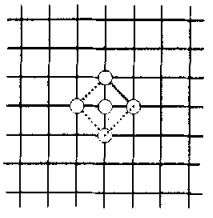


图 5.11 SDS 搜索模板

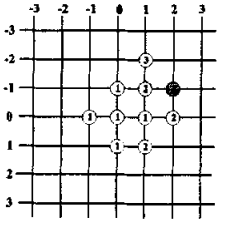


图 5.12 SDS 搜索图示

4. 算法分析

传统算法只考虑了残差块对编码的影响，如何快速找到相似度高的预测块是其唯一目标。事实上，运动估计服务于视频压缩，如何快速找到最有利于视频编码的预测块才是运动估计的根本目标。为此，MVFAST 在“不求最精，务求足够精”的前提下使运动矢量场尽量均匀为目标，同时利用 SAD 的相关性设定阈值，使搜索过程适时中止，对静止块和小运动块的搜索极为高效。实验表明 MVFAST 和其他快速算法相比在速度方面具有明显优势。

八、分层搜索法

1. 基本思想

分层搜索法[46]也可以看作是多分辨率运动估计（MRME）的一种，其思想是首先使用一幅原始图像构造出不同分辨率的图像金字塔（如图 5.13），原始图像为第 0 级（Level 0），相应的最高层图像（即最低分辨率层）为第 N 级（Level N）。进行运动估计是自顶而下实现的，且运动估计越来越精确。搜索时充分利用运动矢量在分层父子块之间的相关性，以低分辨率父层中求出的运动矢量为基础指导下一层的运动估计，缩小运动搜索范围，降低运算复杂度。

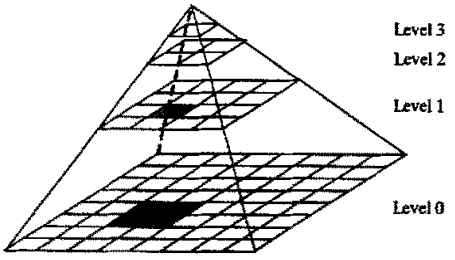


图 5.13 图像金字塔结构

2. 算法描述

从分层块匹配的思想衍生出了多种分层块匹配算法。下面以 Dengler 所提出的算法为例，简要介绍其一般过程。

Step 1：构造 3 级图像金字塔。在构造图像的塔型结构时，一般采用均值滤波。构造公式为

$$f_k^{(l+1)}(i,j)=\frac{1}{4}\sum_{m=0}^{1}\sum_{n=0}^{1}f_k^{(l)}(2i+m,2j+n),\quad l=0,1 \tag{5-5}$$

设原始图像块为 16×16 像素，则经过滤波后 Level 1 和 Level 2 的尺寸分别变为 8×8 和 4×4 像素。

Step 2：在本帧和前一帧 Level 2 中±1 象素范围内进行运动估计，得到尺度较粗的运动矢量 MV_2。

Step 3：在当前帧和参考帧的 Level 1 中进行运动估计，搜索起始点为 $2\times MV_2$ 所对应的坐标点，得出本层运动矢量 MV_1。

Step 4：在原始图像层 Level 0 中进行运动估计，搜索的起始矢量为 $2\times MV_1$，得出最终的运动矢量 MV_0。

3. 算法分析

分层搜索法搜索时首先在最高层进行搜索，由于最高层的宏块尺寸是原始图像的 $2^{-N}\times 2^{-N}$，但却代表了相同的空间域，因此可以较快的搜索到本级较粗尺度的运动矢量。通过上述由粗到细的过程，原始图像中每个象素均可得到精确的运动矢量。由于上层每个象素的运动矢量相当于下一层一个象素块的运动矢量，减少了计算量，且由于是通过滤波产生的金字塔，故具有一定的抗干扰能力。除了上述的多分辨率运动估计算法外，随着小波理论的研究不断深入，基于小波变换的多分辨率特性进行分层搜索法也是人们所关注的热点。

九、 渐进消除算法

渐进消除算法（Successive Elimination Algorithm, SEA）在 1995 年由 W.Li 和 E.Salari 提出的一项无损预测技术，能够在不影响预测精度的前提下减少算法的运算量。且 SEA 所采用的自适应消除思想能够很容易的与其他的块匹配运动估计算

法相结合使用，是一种非常有效的技术。

假设 $f_k(m,n)$ 表示第 k 帧坐标为 (m,n) 点的灰度值，则有：

$$f_k(m,n)-f_{k-1}(m+i,n+j)\le\left|f_k(m,n)-f_{k-1}(m+i,n+j)\right| \tag{5-6}$$

$$f_{k-1}(m+i,n+j)-f_k(m,n)\le\left|f_k(m,n)-f_{k-1}(m+i,n+j)\right| \tag{5-7}$$

对式(5-6)和式(5-7)不等式两边分别累加求和，得到

$$R-M(i,j)\le SAD(i,j) \tag{5-8}$$

$$M(i,j)-R\le SAD(i,j) \tag{5-9}$$

在上两式中，$R=\sum_{m=1}^{M}\sum_{n=1}^{N}\left|f_k(m,n)\right|$，表示当前块内的所有像素灰度值的和；

$M(i,j)=\sum_{m=1}^{M}\sum_{n=1}^{N}\left|f_{k-1}(m+i,n+j)\right|$，表示在前一帧匹配候选块的象素灰度值的和。

假设已经得到了一个运动矢量为 (x,y) 的匹配候选块的 $SAD(x,y)$，那么在搜索过程中，只有当当前块与参考块的绝对差误差和 $SAD(i,j)$ 小于已计算得到的求和绝对差 $SAD(x,y)$ 时才需要进行进一步的匹配计算。由式(5-8)和(5-9)可得，寻求具有更高匹配度的块应该满足式(5-10)所示的条件。

$$R-SAD(x,y)\le M(i,j)\le R+SAD(x,y) \tag{5-10}$$

只有满足此式的候选块才有必要进行匹配计算，否则不予考虑，因而能去除掉相当一部分候选匹配点。参考帧和当前帧的块的象素灰度值的和只需要计算一次，每步只需计算新增的行（或列）的像素值（如图 5.14 所示），而不需要进行重复计算。通过 SEA 搜索判别条件排除了不需要进行匹配计算的块，有效的降低了计算量，且对精度没有任何影响。

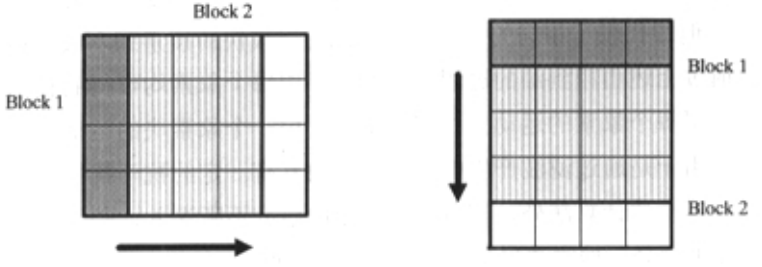


图 5.14 SEA 算法中 SAD 的快速计算方法

由于 SEA 的普适性，近年来人们针对 SEA 算法作了许多研究，陆续提出了一些改进运动估计算法。2000 年 X.Q.Gao 和 C.G. Duanmu 等人提出了多阶渐进消除算法（Multilevel Successive Elimination Algorithm, MSEA），通过在 SAD 计算之前消除分级式不可能的参考运动矢量加速搜索过程，能够进一步降低 SEA 的运算量。

2000 年，Ye-Kui Wang 和 Guo-Fang Tu 将 SEA 的思想与二值块匹配算法相结合，提出了二值渐进消除算法（Binary Successive Elimination Algorithm, BSEA）[47]。Chia-Wen Lin 和 Yao-Jen Chang 等人提出了一种基于金字塔渐进消除算法的分层运算估计算法，结合了多分辨率运动估计、TSS 法、BBGDS 法和渐进消除算法，也收到了良好的效果[48]。Chang-Hsing Lee 和 Ling-Hwei Chen 在多分辨率的渐进消除算法方面也做了有意义的研究[49]。J.J. Francis 和 G.de Jager 提出了一种针对均方误差和的渐进消除算法（Sum Square Error based Successive Elimination Algorithm）[50]，扩展了 SEA 的应用范围，具有一定的研究价值。

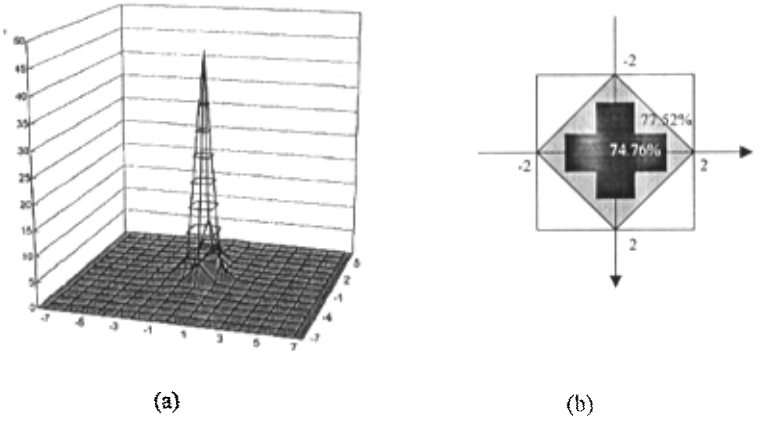
5.3 六边形运动矢量场自适应搜索算法

通过对上述几种经典的运动估计算法的分析，在此基础上，我们提出了一种新的搜索算法——六边形运动矢量场自适应搜索算法（Hexagon-based Search Algorithm using Motion Vector Field Adaptive Search Technology, HMVFAST），具有较高的搜索精度和速度，且能够获得均匀的运动矢量场，利于编码传输。HMVFAST 主要采用了以下几项核心技术：设定阈值对静止块直接中止计算；使用空间相邻块和时间相邻块估计当前块的运动类型；根据运动类型自适应选择搜索起始点和搜索策略；使用两种新的搜索算法——六边形搜索法（HEXBS）和小菱形搜索法（SDS）。

5.3.1 搜索中止准则

BMA 的搜索模板和搜索策略决定了整个算法的计算复杂度和准确度。为了设计合理的搜索模板和搜索策略，研究视频序列中块运动的基本性质是非常重要的。文献[42]详细分析了运动矢量的分布特性，发现基于块运动的搜索具有如下的性质：视频序列的运动矢量具有中心偏置特性，即大部分块是静止的或准静止的。且统计结果显示，运动矢量主要分布在零矢量周围以(0,0)为圆心，±2 像素范围的区域内。图 5.15(a)显示了图像序列平均运动矢量的一般分布情况，从图(b)可以看出，位于半径 1 像素和 2 像素内的区域内包含了大部分的运动矢量。

为此可以构造阈值 T，采用 SAD 准则衡量块失真度对静止块进行判定。在搜索前首先检测(0,0)位置，如果满足 $SAD(0,0) < T$，认为该块属于静止块，其最优运动矢量就是(0,0)，此时直接中止搜索。统计实验证明当 T=512 时，误判概率和虚警概率最小。可以看出对于静止块，HMVFAST 仅仅需要检测 1 个搜索点，即零矢量点即可完成搜索，极大的节省了计算量。



(a) (b)

图 5.15 运动矢量的中心偏置特性

5.3.2 运动类型判定

视频序列中一帧内构成物体对象的块具有相似的运动矢量，即运动矢量具有空间相关性；由于视频序列的连续性，当前帧和其前一帧和后一帧之间的运动矢量具有时间相关性。利用运动矢量的这种空时相关性可以对当前块的运动矢量进行预测，该初始运动矢量在一定程度上反映了当前块的运动情况，利于随后使用最合适的搜索模板和搜索策略实现自适应搜索。一般来说，空间位置的已编码的相邻块和前一帧的图像块都可以被用来预测当前块的运动矢量。实验统计表明，当前块与帧内上方、左方和右上方的子块的相关性最强，而与其他位置的相关性则较弱，本文选取这 3 个空间邻块（如图 5.16 中的块 1,2,3）和参考帧相同位置块运动矢量的对应点（即时间相邻块，如块 4）作为候选点进行预测。

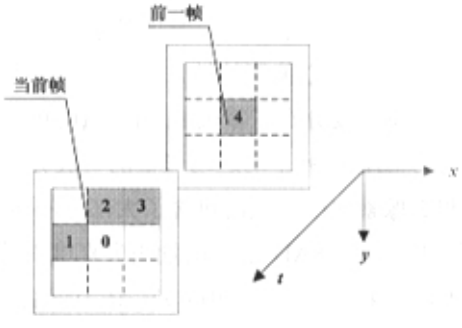


图 5.16 时间邻块和空间邻块的设定

表 5.2 列出了在几个典型的图像序列中，子块与空间和时间邻块的一致的百分比。

表 5.2 邻块运动矢量相等百分比 (单位：%)

测试序列	块 1	块 2	块 3	块 4
Miss America	86.81	85.94	82.14	83.58
Foreman	62.45	56.82	61.76	38.91
Tennis	63.67	60.38	52.60	55.73
Coastguard	72.13	63.36	50.95	46.76
平均	**71.27**	**66.63**	**61.86**	**56.25**

令运动矢量集合$V=\{V_0,V_1,V_2,V_3,V_4\}$，$V_0=(0,0)$，$V_i=(x_i,y_i)$，$i$=1,2,3,4，分别表示块 1、2、3 和块 4 的运动矢量，定义

$$l_i=|x_i|+|y_i| \tag{5-11}$$

则l_i是运动矢量的欧氏距离，在一定程度上反映了该运动矢量的剧烈程度。对于运动矢量集合中的所有候选矢量，定义

$$L=\max\{l_0,l_1,l_2,l_3\} \tag{5-12}$$

设定阈值L_1和L_2，且$L_1<L_2$。当$L\le L_1$时，当前块被认为是小运动块；当满足$L_1<L\le L_2$，当前块为中等运动块；当$L>L_2$时，当前块为大运动块。

5.3.3 起始点预测

搜索起始点根据当前块的运动类型进行设定。如果当前块为小运动块或中等运动块时，意味着子块的最优运动矢量位于(0,0)附近较小的区域内，所以无需进行起始点预测；如果当前块为大运动类型，其最优运动矢量往往偏离中心点(0,0)较大，预测后的搜索起始点更接近物体的真实运动矢量。

确定搜索起点的主要方法有中值法、加权法和 SAD 比较法。中值法使用各预测块运动矢量的均值作为搜索起始点，加权法用各预测块运动矢量按一定的权值求和计算得到，SAD 比较法分别对各预测矢量进行比较，取 SAD 最小者作为搜索起始点。从预测精度的角度考虑，使用 SAD 比较法最好，结合适当的搜索策略，能够最快的寻找到最优矢量。同时使用 SAD 比较法得出的预测起始点必然是某个邻块的运动矢量，使得运动矢量场具有连续性，利于差分编码。HMVFAST 中采用 SAD 比较法选择搜索起始点。

对于上述的运动矢量集合 V 中的每个运动矢量，$V_i=(x_i,y_i)$，i=1,2,3,4，对应的块失真度分别为$\{SAD_0,SAD_1,SAD_2,SAD_3,SAD_4\}$。选取其中的最小值$SAD_n=\min\{SAD_0,SAD_1,SAD_2,SAD_3,SAD_4\}$时的运动矢量 V_n 所对应的点，为搜索起始点。

5.3.4 搜索策略

为了对各种运动类型的块都能快速准确地搜索到最优矢量，HMVFAST 在搜索过程中自适应选择使用两种搜索算法：六边形搜索法（HEXBS）和小菱形搜索法（SDS）。

一、六边形搜索法

1. 基本思想

分析 9 点菱形模板（LDSP）可以发现，LDSP 四周的 8 个匹配点到中心点(0,0)位置距离是不同的：水平和垂直方向的相邻搜索点间距为 2 像素，而中心点和对角方向的相邻搜索点间距为$\sqrt{2}$像素。因此使用 LDSP 进行粗定位时，沿不同方向移动的匹配速度也是不同的，当 LDSP 的顶点为本次匹配的 MBD 点时模板沿水平或垂直方向移动，此时的搜索速度为 2 像素/步；当模板沿对角方向移动时其速度为$\sqrt{2}$像素/步。另一方面，在大模板移动的每一步中，不同的搜索方向需要检测的搜索点数也不同。在水平和垂直方向上需要检测 5 个新搜索点，而在对角方向上只需检测 3 个新的搜索点即可（参见图 5.10(a)和图 5.10(b)）。对于保持静止的图像序列，即运动矢量为零的情况，DS 算法要经历由大模板到小模板的变化过程，要对 13 个搜索点进行搜索，而理想情况是只需搜索 1 个点。从以上几点可以看出大菱形模板并不是最优的搜索模板，需要进一步加以改进。事实上，造成该问题的根源在于块匹配误差实际上是在搜索范围内建立的误差表面函数，全局最小点即对应着最佳运动矢量，而 LDSP 实际上只是一个旋转了 45°的正方形模板，在对角方向上的梯度下降方向不够快，需要较多步才能够搜索到最优点。

圆上的任意一点到圆心的位置都是相同的，如果能利用圆形作为匹配模板，则该模板在各个搜索方向都具有相同的梯度下降速度，搜索速度能够达到最优[51]。考虑到计算复杂度和图像序列的实际情况，我们使用了六边形搜索法（HEXBS），包含图 5.17(a)和(b)中所示的六边形模板（Hexagon Search Pattern, HSP）和小菱形模板（SDSP）。搜索时先用 HSP 进行计算，当 MBD 点出现在中心处时，可以认为最优点位于 HSP 所包围的六边形区域内，此时再将 HSP 换为 SDSP，这 5 个点中的 MBD 就是最优匹配点。

2. 算法描述

六边形搜索法的具体步骤如下：

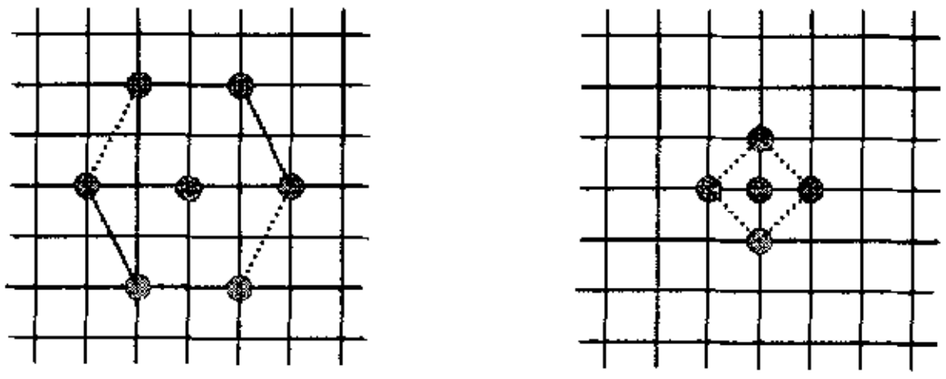
Step 1：用 HSP 在搜索区域中心及周围八个点处进行匹配计算，若 MBD 点位于中心点，则进行 Step 3；否则转入 Step 2。

Step 2：以上一次找到的 MBD 点作为中心点，用 HSP 来计算各匹配点，若 MBD 点位于中心点，则进行 Step 3；否则重复 Step 2。

Step 3：以上一次找到的 MBD 点作为中心点，将 HSP 换为 SDSP，在五个点处计算，找出 MBD 点；当 MBD 点位于 SDSP 的中心点，此点即为最优矢量，转入 Step 5,；否则进行 Step 4。

Step4：搜索距离上一次的 MBD 点距离最近的两个搜索点，这 3 点中的 MBD 点即是最优矢量，转入 Step 5。

Step5：算法结束，得出最优运动矢量。



(a) (b)

图 5.17 HEXBS 搜索模板 (a)HSP (b) SDSP

使用六边形搜索法的搜索过程参见图 5.18，搜索中共使用了 4 次 HSP 和 SDSP 模板，每一步的 MBD 点分别为(1, 2), (2, 4),(4, 4)，最终的最优运动矢量为(4, 3)。

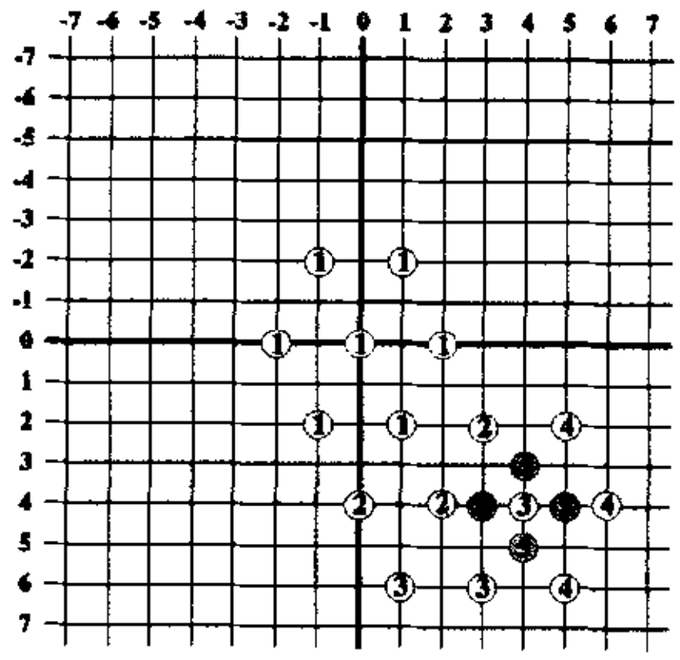


图 5.18 HEXBS 搜索图示

3. 算法分析

六边形模板包含 7 个搜索点，比 LDSP（9 个搜索点）减少了 2 个搜索点，因而在粗定位的过程中比 LDSP 的计算复杂度低。HSP 比 LDSP 更接近于圆，其水平方向的顶点到中心点距离为 2 像素，对角方向的顶点距离中心点 $\sqrt{5}$ 像素。使用 HSP 在匹配窗内移动时，在各个方向上的移动速度也非常接近，沿水平方向模板的梯度下降速度为 2 像素/步，在对角方向为 $\sqrt{5}$ 像素/步，高于 LDSP 的搜索速度，且新增搜索点数与方向无关，无论本次 MBD 点位于模板的何处位置，下次匹配只存在 3 个新匹配点需要计算。当粗定位结束后，这时再用 SDSP 来准确定位，兼顾了速度和精度的要求，综合性能高于 DS 算法。

二、SDS 搜索法

SDS 只采用包含 5 个搜索点的小菱形模板进行搜索，如果本次匹配的 MBD 点位于 SDSP 的中心位置，则结束搜索；如果 MBD 位于周围的 4 个顶点，则以该 MBD 点为中心点，继续使用 SDSP 进行搜索，直到 MBD 位于中心为止。SDS 法的搜索过程如图 5.19 所示，图中的箭头指明了 MBD 点的移动位置；(a)中第一步的 MBD 点位于(0,-1)处，第二步中便以(0,1)为中心进行搜索；(b)中 SDS 第一步的 MBD 点位于(1,0)，因此 SDSP 向右移动匹配。

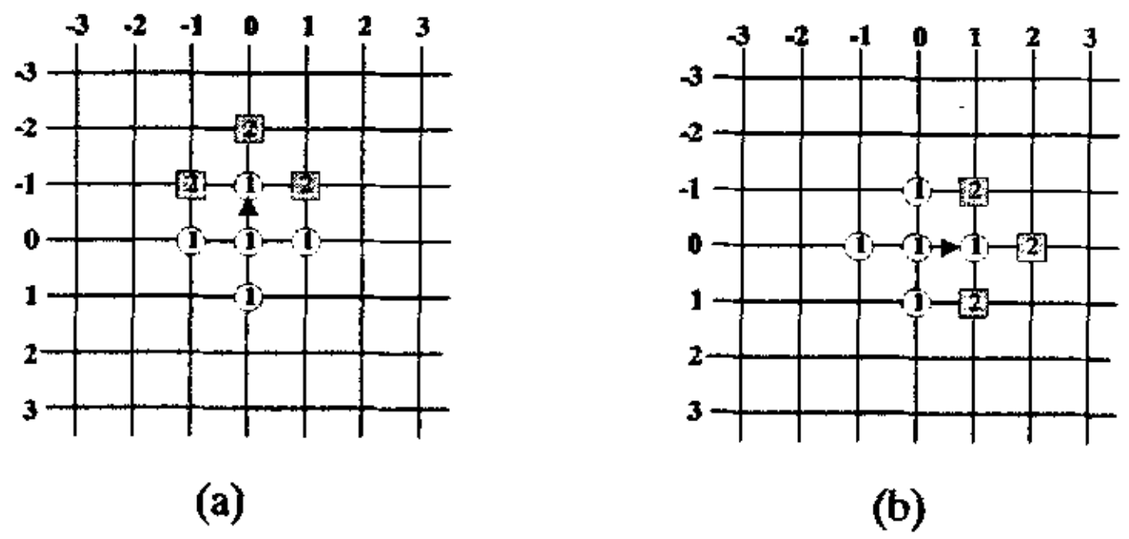


图 5.19 SDS 搜索过程

SDS 法只包含 5 个搜索点，局部搜索速度快，对于静止块和小运动块搜索速度快，但是对于大运动块容易陷入局部最优而终止预测，不能找到全局最优点。

5.3.5 HMVFAST 算法描述

HMVFAST 首先判断当前块是否为静止块，如为静止块则直接中止搜索；对于非静止块则根据运动矢量的时空相关性进行运动类型判定和起始点预测，自适应的选择搜索模板和搜索策略。对于小运动块和中等运动块，最优运动矢量位于中心点附近的位置，使用 SDS 法精度高、速度快；对于大运动块先进行起始点预测，使搜索起始点更接近全局最优点，再使用 HEXBS 搜索，具有良好的搜索性能。HMVFAST 搜索模式如表 5.3 所示，算法流程图参见图 5.20。

表 5.3 HMVFAST 的搜索模式

运动类型	搜索初始点	搜索策略
静止块	——	——
小运动块	零矢量位置(0,0)	SDS
中等运动块	零矢量位置(0,0)	HEXBS
大运动块	V 中的最优矢量	SDS

根据以上思想，可以将 HMVFAST 算法表述如下：

Step 1：静止块检测。搜索零矢量位置(0,0)点，得出该点的 SAD 值 SAD_0，如果 $SAD_0 < T$，则判定该块为静止块。(0,0)位置即为最优运动矢量，转入 Step 6；否则转入 Step 2。

Step 2：运动类型检测。根据上方、左方和右上方子块得出当前块的预测矢量范围L，如果满足$L \le L_1$，说明当前块为小运动块，转入Step 4；如果满足$L_1 < L \le L_2$，当前块为中等运动块，转入 Step 5；当$L > L_2$时，当前块属于大运动块，需要进行起始点预测，转入 Step 3；

Step 3：起始点预测。在当前块的时间和空间邻块中选择误差最小的运动矢量作为当前块运动估计的起始点，转入 Step 4。

Step 4：SDS 搜索。以当前 MBD 点为中心，使用 SDSP 进行匹配。如果 MBD 点位于中心位置，则说明当前点即为最优运动矢量，转入 Step 6；否则令此 MBD 点为起始点，转 Step 4，继续使用 SDSP 进行匹配。

Step 5：HEXBS 搜索。以当前点为搜索原点，使用前述的六边形搜索法进行搜索。

Step 6：算法结束，得到最优运动矢量。

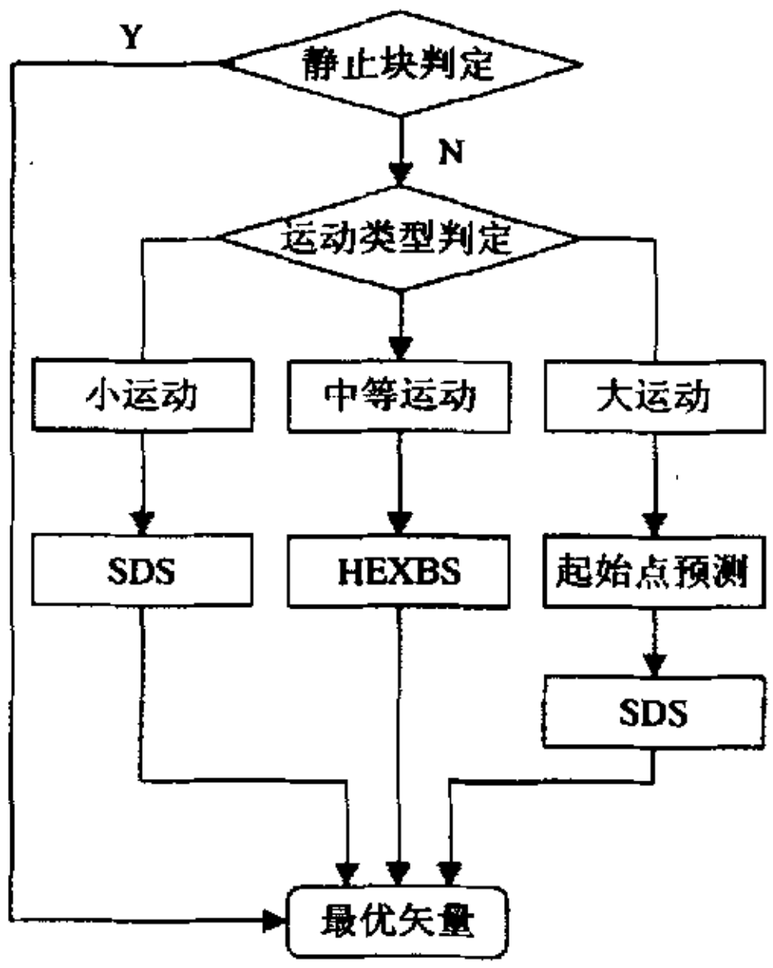


图 5.20 HMVFAST 算法流程图

在 HMVFAST 的搜索过程中可以根据需要调整阈值的大小。对序列的实验统计结果表明，在 L_1=1，L_2=2，T_0=512 的情况下，算法具有最优的估计精度和速度。

5.4 实验结果与分析

为了验证算法的有效性，我们选取了几种最具代表性的块匹配算法：FS、TSS、DS 和 MVFAST，在相同的条件下与 HMVFAST 进行对比实验。使用的匹配块大小为 16×16 像素，搜索窗大小为±15 像素，匹配准则使用 SAD。实验中选取 4 个 CIF 格式图像序列：Claire, Football, Tennis, Mobile。Claire 是典型的头肩序列，图像运动微小平缓；Football 属于大运动序列；Tennis 序列中包含物体的快速运动

和场景切换；Mobile 序列中既含有镜头的平移也包括物体的移动，且背景复杂。对这些序列中的前 100 帧逐帧进行运动估计，计算其 PSNR 和平均搜索点数的平均值。

表 5.4 平均搜索点数比较（单位：次/像素）

算法	Claire		Football		Tennis		Mobile	
	平均搜索点数	加速被数	平均搜索点数	加速被数	平均搜索点数	加速被数	平均搜索点数	加速被数
FS	225	1.00	225	1.00	225	1.00	225	1.00
TSS	25	9.00	25	9.00	25	9.00	25	9.00
FSS	16.63	13.53	19.24	11.69	18.98	11.85	16.98	13.25
DS	12.95	17.37	16.65	13.51	15.71	14.32	13.23	17.01
MVFAST	1.65	136.36	9.42	23.89	6.85	32.85	7.08	31.78
HMVFAST	1.56	144.23	9.02	24.94	6.47	34.76	7.05	31.91

观察表 5.4 中可以发现，对于 Claire 序列，FS 需要搜索 225 点，DS 也需要搜索 12.95 个点，而 MVFAST 和 HMVFAST 平均仅需搜索 1.65 和 1.56 个点即可找到最优矢量。对于大运动序列 HMVFAST 也有效的降低了搜索点数：和 MVFAST 比较，HMVFAST 在 Football 序列中提高了 0.4 个搜索点，在 Tennis 序列中提高了 0.38 点。由此可以看出，采用运动矢量场自适应技术能够明显降低寻找最优矢量的平均搜索点数，HMVFAST 采用了六边形模板进行匹配，收敛速度快，搜索点数更少，是搜索速度最快的算法。

为了比较各运动估计算法的精度，表 5.5 中列出了各算法的重构图像的 PSNR 平均值以及与 FS 的比较结果。

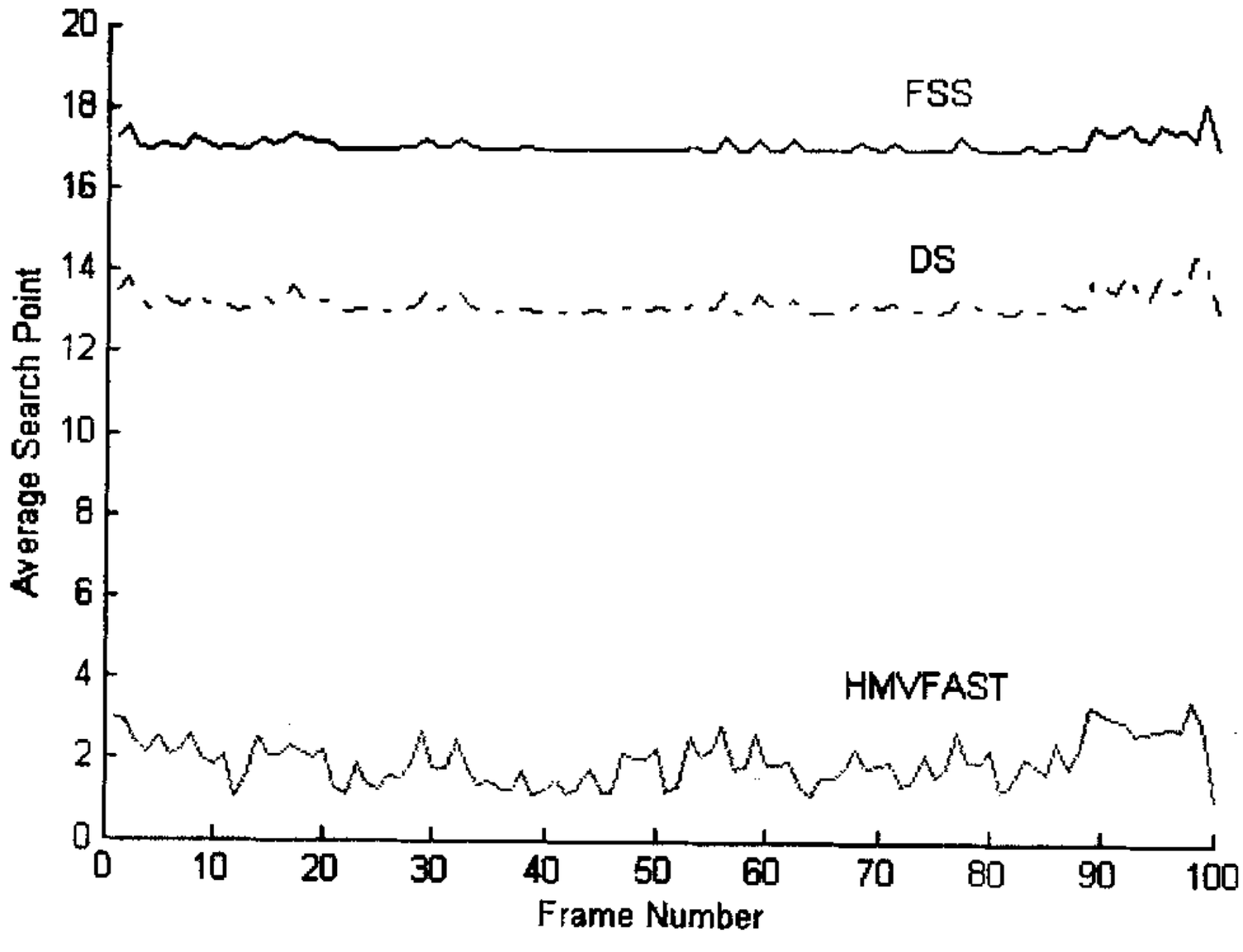
表 5.5 平均 PSNR 值比较（单位：dB）

算法	Claire		Football		Tennis		Mobile	
	PSNR	ΔPSNR	PSNR	ΔPSNR	PSNR	ΔPSNR	PSNR	ΔPSNR
FS	41.00	0.00	22.39	0.00	29.53	0.00	22.76	0.00
TSS	40.67	-0.33	21.22	-1.17	27.37	-2.16	22.44	-0.32
FSS	40.94	-0.06	21.68	-0.71	28.38	-1.15	22.66	-0.10
DS	40.98	-0.02	21.68	-0.71	28.70	-0.83	22.69	-0.07
MVFAST	40.96	-0.04	21.92	-0.47	28.74	-0.79	22.74	-0.02
HMVFAST	40.94	-0.06	21.90	-0.49	28.71	-0.82	22.74	-0.02

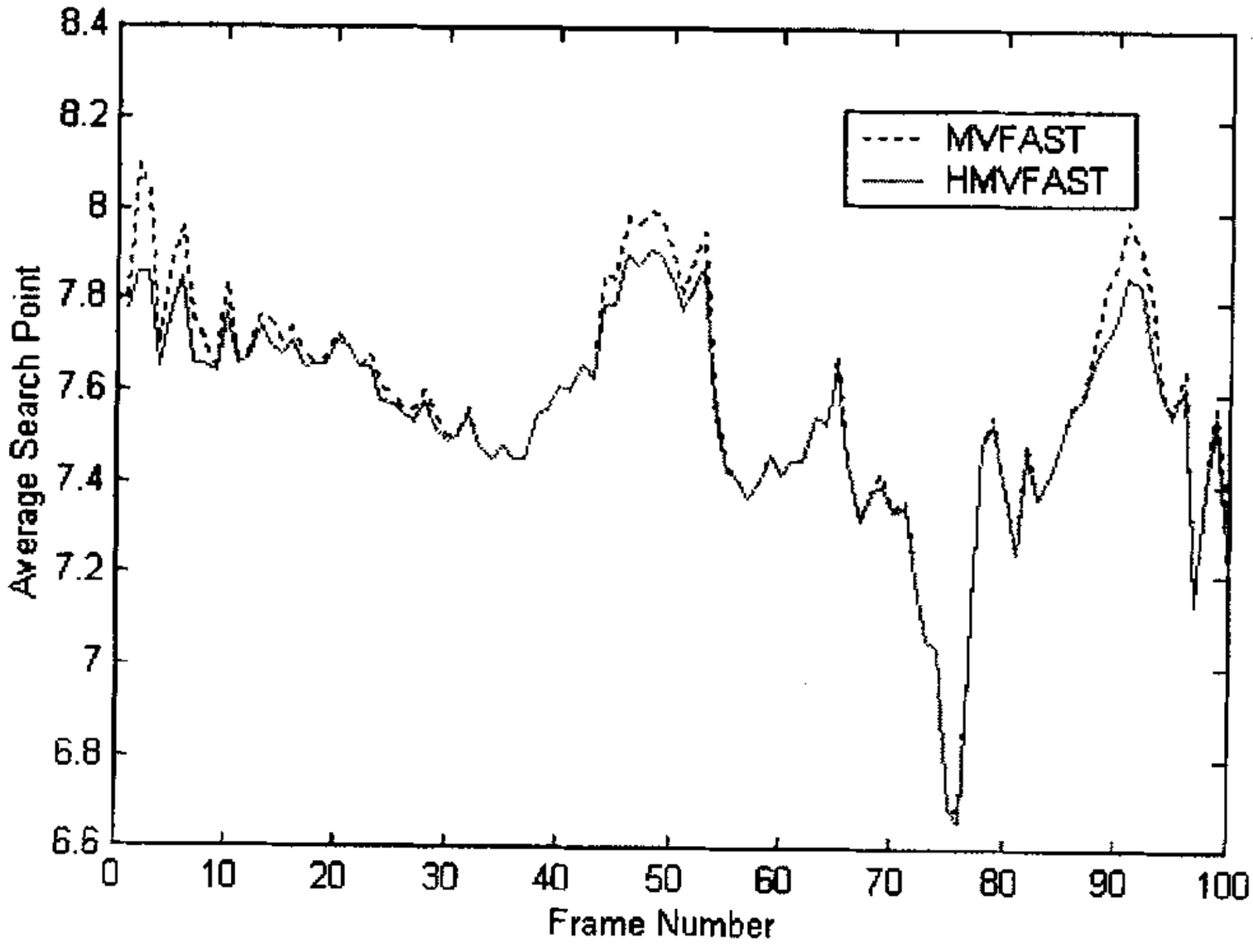
从表 5.5 中可以看出，FS 的平均 PSNR 值最高，是搜索精度最高的块匹配算法。HMVFAST 在大部分序列中的搜索精度都高于 DS 算法，接近于 FS 算法，其中在 Mobile 序列中仅低于 FS 0.02dB。在这 5 个序列中，仅对 Claire 序列，HMVFAST 的 PSNR 值低于 DS 算法，和 MVFAST 效果相当。这说明 HMVFAST 在搜索精度上和 MVFAST 非常接近，与精度最高的 FS 算法相比也相差不多。

图 5.21(a)和(b)直观地显示了使用 HMVFAST 和 TSS、DS、MVFAST 对 Clarie 和 Football 序列的前 100 帧进行运动估计所需的平均搜索点数。对比可以发现，

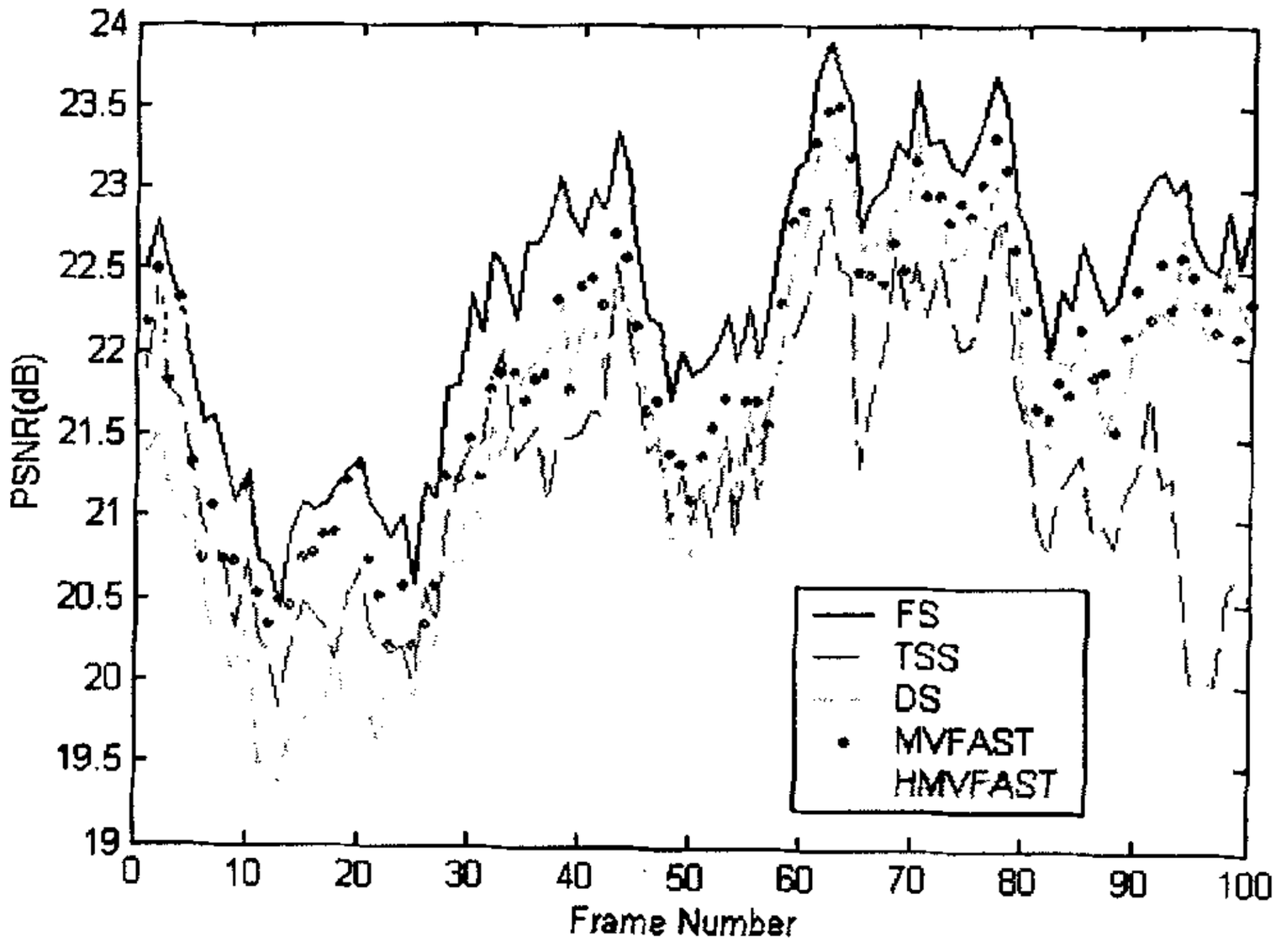
HMVFAST 是搜索点数最少的算法。图 5.21(c)显示了对于 Football 序列不同算法得出的 PSNR 平均值。



(a) Clarie 序列平均搜索点数比较



(b) Football 序列平均搜索点数比较



(c) Football 序列各帧 PSNR 值比较

图 5.21　几种快速运动估计算法性能比较

实验结果表明：可以看出 MVFAST 十分接近于 FS 算法，优于传统的 DS 算法。因此，HMVFAST 在搜索速度和精度上均超过了 DS 算法，其中对于小运动图像序列的搜索速度更是远远超过 DS。该算法的搜索速度也快于 MVFAST，同时搜索精度和 MVFAST 相差不多，接近于 FS 算法，是一种优秀的运动估计算法。

第六章 结束语

6.1 论文完成的主要工作

本文以视频编码技术为基础，针对新一代国际视频编码标准 H.264/AVC 中的核心算法进行了重点研究和分析，在此基础上提出了两种新的改进算法。总结本文的工作，主要包括以下几个方面：

1. 总结了 H.264 的基本原理和新技术，并着重研究了 H.264 中提高编码效率的几项核心技术。
2. 详细的分析了 H.264 标准中首次使用的两种整数变换算法的构造过程并加以推导；对其量化方案也做了深入的研究，并通过实验证明了其高效性和可行性。
3. 针对 H.263、MPEG-4 和 H.264 的帧内预测算法进行研究，分析了几种算法的优缺点；得出了 H.264 帧内预测中的两条一般性规律，提出了相关性因子的概念，以此为基础提出了自适应空域帧内预测算法（ASIP）。
4. 对运动估计算法进行深入研究，结合使用了起始点预测、运动类型判定和不同的搜索模板，提出了六边形运动矢量场自适应搜索算法（HMVFAST）。

6.2 未来研究展望

随着计算机技术的发展，数字视频技术在国民经济中会产生越来越重要的影响。视频编码标准也已经从传统的影音存储、广播电视和视频会议等领域逐渐扩展到网络通信、无线视频通信等应用领域。H.264/AVC 作为 ITU 和 ISO 联合制定的新一代视频编码标准，具有重要的研究和应用价值。作者认为在该领域继续研究，可以从以下几个方面考虑：

1. 精细可伸缩的编码技术

随着新的数字媒体载体特别是互联网的发展，越来越多的视频内容采用数据包广播的方式在互联网上进行传递。在互联网上传输视频流需要解决的一个基本问题是传输带宽的波动。精细可伸缩编码技术（Fine Granularity Scalable, FGS）的出现受到了众多研究单位的关注，它的基本原理是对视频编码产生一个基本层码流和一个嵌入式的增强层码流。基本层码流包含了最重要的、质量较低的视频信号；增强层码流保存着基本层的量化差值，嵌入式的码流保证了视频信号改变是渐进、平滑的，接收到的增强层码流越多，重构的视频效果越好。

H.264 的码流不具有可伸缩性。如何能有效的利用 FGS 编码技术，使 H.264 也

具备可伸缩的优良特性，是以后研究中一个非常有前景的方向。目前国内外的研究人员也陆续提出了一些针对 H.264 可伸缩编码的改进方案。

2. 网络抗误码能力

网络传输所要考虑的另一个问题是无线信道和 Internet 等环境下传输差错和丢包对视频的影响。H.264 目前使用的抗误码技术与具体信道的误码特点和图像序列的误码敏感度针对性不强，在实际误码信道条件下抗误码效果还有进一步改善的必要。目前，国内外针对 H.264 的抗误码性能研究也正在起步阶段，研究有效的错误掩盖和错误恢复技术也是一个很值得研究的方向。

3. 基于小波的视频编码技术

小波变换编码具有分层编码的自然伸缩特性和良好的编码鲁棒性，也是视频编码的重要方法。小波变换的多分辨率特性提供了利用人眼视觉特性的良好机制，从而使小波变换后图象数据能够保持原图象在各种分辨率下的精细结构。静止图象的小波变换方法已经被 JPEG2000 国际标准所采纳，取得了优于 JPEG 的图象压缩效果，且提供了诸如感兴趣区域、渐进传输等传统算法所无法提供的优良特性。将小波变换与分形、矢量量化等技术相结合的视频压缩编码方法除具有高压缩比之外，也能够很容易得到诸如空域可伸缩、PSNR 可伸缩等优良特性，是一项很有发展前景的技术。

致 谢

本论文是在我的导师郭宝龙教授的悉心指导下完成的，从论文的选题、论证、研究到最后完成，自始至终无不凝聚着导师的心血。郭老师创造的宽松民主的学术氛围、团结和谐的工作环境也极大地激发了我的创新意识和进取精神。郭老师严谨的治学态度，高深的学术造诣，对学科前沿的敏锐洞察力，以及不断开拓新领域的精神，令我受益匪浅，终生难忘。在此，我要对我的导师郭宝龙教授表示衷心的感谢！

在此也感谢 1204 教研室的冯宗哲老师和陈生潭老师为我的研究工作提供的热情支持和帮助。感谢丁贵广师兄、郭磊师兄、计文平师兄、费佩燕师姐和向友君师姐，与他们的交流和讨论使我学到了很多知识。感谢朝夕相处的室友张永平和闫允一，以及实验室的同学万旻、聂飞、赵友军、侯振华、孟繁杰和曹蓓，与他们相处，我度过了一段美好的时光。

感谢于培松、吴宪祥、杨志勇、杨志伟、曹昆、侯舒维、潘玉、朱娟娟等师弟师妹，感谢我的好友符为、周峰、王雪以及所有关心帮助我的朋友。

特别感谢我的父母和弟弟，是他们的爱和殷切的期望鼓励我战胜困难，不断前行。

参考文献

[1] ISO/IEC JTC1/SC29/WG11, ISO/IEC, MPEG-1 Committee Draft", CD11172: Information Technology. Dec.1991.

[2] ISO/IEC JTC1/SC29/WG11, ISO/IEC, MPEG-2 Committee Draft", CD13818: Information Technology. Dec.1993.

[3] ISO/IEC/JTC1/SC29/WG11, ISO/IEC, Coding of Audio-Visual Objects-Part 2: Visual, ISO/IEC 14496-2(MPEG-4 Visual Version 1), Apr 1999.

[4] ITU-T, Recommendation H.261: Video Codec for Audiovisual Services at p×64kbit/s, Mar 1993.

[5] ITU-T, Recommendation H.263: Video Coding for Low Bit Rate Communication, ITU-T Recommendation H.263 Draft, July 1995.

[6] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec.H.264/ISO/IEC 14496-10 AVC), JVT-G050, Pattaya, Mar. 2003.

[7] Thomas Sikora, The MPEG-7 Visual Standard for Content Description—An Overview, *IEEE Trans. Circuits Syst. Video Technol.*, Vol.11, No.6, June 2001.

[8] Jan Bormans, Keith Hill, MPEG-21 Overview, ISO/IEC JTC1/SC29/WG11/ Document N40401, Mar. 2001.

[9] ITU-T, "Draft ITU-T Recommendation H.263 version 2: Video Coding for Low Bit Rate Communication, September 1997.

[10] ITU-T, H.26L Test Model Long-Term Number 9 (TML-9) draft0, ITU Document VCEG-N83d1, Dec 2001.

[11] D.Marpe, H.Schwarz, and T.Wiegand, Context-adaptive binary arithmetic coding in the H.264/AVC video compression standard, *IEEE Trans. Circuits Syst. Video Technol.*, Vol.13, No.7, pp.598-603, July 2003.

[12] T.Wedi, Motion comprensation inH.264/AVC, *IEEE Trans.Circuits Syst.Video Technol.*, vol 13, No.7, pp.577-586, July 2003.

[13] Peter List, Anthony Joch, Jani Lainema, etc. Adaptive Deblocking Filter, *IEEE Trans. Circuits Syst.Video Technol.*, vol 13, No.7, pp.614-619, July 2003.

[14] 麻晓园 鄢勇, H.264 视频编码标准及其在移动通讯中的应用, 现代电信科技, 2003.7, pp.5-8.

[15] S.Wenger, H.264/AVC over IP, *IEEE Trans.Cicuits Syst. Video Technol., vol.13, No.7*, pp.645-656, July 2003.

[16] ISO/IEC JTC 1/SC 29/WG 1. "ISO/IEC FDIS 15444-1: Informaiton Technology - JPEG2000 Image Coding Sytem: Core Coding System, 2000.

[17] N.Ahmed, T. Natarajan, and K. R. Rao, Discrete cosine transform, *IEEE Trans. Comput.*, Vol. C-23, pp. 90–93, Jan 1974.

[18] E.Feig and S.Winograd, On the Multiplication Complexity of Discrete Cosine Transforms, *IEEE Trans. Inform. Theory*, Vol 38, No.4:1387-1391,Apr 1992

[19] M. Vetterli and H. Nussbaumer, Simple FFT and DCT algorithms with reduced number of operations, *Signal Processing*, Vol.6, No.4, pp.267-278, Aug 1984.

[20] D.Hein and N.Ahmed, On a real-time Walsh-Hadamard cosine transform image processor, *IEEE Trans. Electromagn. Compat.*, Vol. EMC-20, pp. 453-457, Aug 1978.

[21] S.Venkataraman, V.Kanchan, K.R.Rao, and M.Mohanty, "Discrete transform via the Walsh-Hadamard transform," *Signal Processing*, Vol. 14, No. 4, pp.371-382, June 1988.

[22] H.Malvar, Fast computation of the discrete cosine transform and the discrete Hartley transform, *IEEE Trans. Acoust., Speech and Signal Processing*, Vol. ASSP-35, pp. 1484-1485, Oct 1987.

[23] C.Loeffler, A. Lightenberg, and G. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," *Proc. IEEE ICASSP*, Vol. 2, pp.988–991, Feb 1989.

[24] Y.Arai, T.Agui, and M.Nakajima. Afast dct-sq scheme for images. *Trans. IEICE*, Vol.E-71, No.11, pp.1095-1099, 1988.

[25] Y. Arai, T. Agui, and M. Nakajima, "A Fast DCT-SQ Scheme for Images", *IEEE Transactions of the IEICE*, Vol. E71, No.11, pp.1095～1097, 1988.

[26] W.Chen, C.H.Smith, and S.C.Fralick, A fast computational algorithm for the discrete cosine transform, *IEEE Trans. Commun.*, Vol.COMM-25, pp.1004- 1009, Sep 1977.

[27] 郝鹏威 石青云, 可逆线性变换的整数实现, 中国科学(E 辑), Vol.20, No.2, pp. 132-141, Apr 2000.

[28] Cheng Li Zhi, Xu Hui, and Luo Yong, Integer discrete cosine transform (IntDCT) and its fast algorithm, *IEEE Electronic Letter*, Vol.37, No.1, 2001.

[29] Jie Liang, Trac D.Tran, Fast Mulitipierless Approximations of the DCT with the Lifting Scheme, *IEEE Trans. Signal Processing*, Vol 49, No.12, pp.3032-3044, Dec 2001.

[30] Ying-Jui Chen, Soontorn Oraintara, Trac D.Tran, Multiplierless Approximation of Transforms with Adder Constraint, *IEEE Signal Processing Letter*, Vol.9, No.11, pp 344-347, Nov 2002.

[31] Henrique S.Malvar, Antti Hallapuro, Marta Karczewicz, Low-Complexity Transform and Quantization in H.264/AVC, *IEEE Trans.Circuits Sys. Video Technol.*, Vol.13, No.7, July 2003.

[32] Giles J Nelson, The Block Matching Approach to Motion Estimation, M.Sc. Dissertation, Department of Computer Science, University of Warwick, UK. September 1993.

[33] Jarkko Kari, Gang Liang, Simant Dube, Fast Block Search Using Harr Decomposition, *IEEE Trans. CASVT.*, Vol.8, No.4, pp.399-409, Apr 1998.

[34] Zhang Y Q, Zafer S, Motion-compensated wavelet transform coding for color video compression. *IEEE Trans. CASVT*, Vol.2, No.3, pp.285-296, Feb 1992.

[35] X Yang, K ramchandram. Scalable wavelet video coding using alisasing-reduced hierarchical motion compensation. *IEEE Trans on Image Processing*. 2000,5,9(5): 778-791.

[36] 向友君，视频编码中的运动估计算法研究[硕士论文], 西安:西安电子科技大学, 2002.

[37] Koya T, Iinuma K, Hirano A. Motion-compensated interframe coding for video conferencing. *Proc NTC81*, New Orleans, LA. C9.6.1-9.6.5, Nov 1981.

[38] R.Li, B.Zeng, M.L.Liou. A new three-step search algorithm for block motion estimation. *IEEE Trans. CASVT*, Vol.4, No.8, pp.438-442, Aug 1994.

[39] L. M. Po and W. C. Ma. A novel four-step search algorithm for fast block motion estimation. *IEEE Trans. CASVT*, Vol.6, No.6, pp.313-317, Jun 1996.

[40] Liu L K, Feig E. A block-based gradient descent search algorithm for block motion estimation in video coding. *IEEE Trans. CASVT*, Vol.6, No.4, pp. 419-422, Aug 1996.

[41] S. Zhu and K. K. Ma, A new diamond search algorithm for fast block-matching motion estimation. *IEEE Trans. Image Processing*.Vol.9, No.2, pp.287-290, 2002.

[42] Chun-Ho Cheung and Lai-Man Po, A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation, *IEEE Trans. Circuits Syst.Video Technol.*, Vol.12, No.12, Dec 2002.

[43] Weiguo Zheng, Ishfaq Ahmad and Ming Lei Lion, Adaptive Motion Search with Elastic Diamond for MPEG-4 Video Coding, *ICIP 2001*, pp.377-380.

[44] Alexis M.Tourapis, Oscar C.Au, and Ming L.Liou, Highly Efficient Predictive

Zonal Algorithms for Fast Block-Matching Motion Estimation. *IEEE Trans. Circuit Syst.Video Technol.* Vol.12, No.10, Oct 2002.

[45] Prabhudev Irappa Hosur and Kai-Kuang Ma, Motion Vector Field Adaptive Fast Motion Estimation, *ICICS 1999*, Singapore, pp.7-10, December 1999.

[46] M.Bierling, Displacement estimation by hierarchical block matching. *SPIE Visual Communication and Image Processing*. 1988,5,1001(5): 942-951.

[47] Ye-Kui Wang and Guo-Fang Tu, Successive elimination algorithm for binary block matching motion estimation. *IEEE Electronic Letters*, Vol.36, No.24, pp.2007-2008, Nov 2000.

[48] Chia-Wen Lin, Wao-Jen Chang, Hierarchical Motion Estimation Algorithm Based on Pyramidal Successive Flimination, *International Computer Symposium 1998*, Oct 1998.

[49] Chang-Hsing Lee and Ling-Hwei Chen, A Fast Motion Estimation Algorithm Based on the Block Sum Pyramid, *IEEE Trans. Image Processing*, Vol.6, No.11, Nov 1997.

[50] J.J.Francis and G.de Jager, A Sum Square Error based Successive Elimination Algorithm for Block Motion Estimaiton. *PRASA 2002*, July 2002.

[51] Ce Zhu, Xiao Lin, and Lap-Pui Chau, Hexagon-Based Search Pattern for Fast Blcok Motion Estimation, *IEEE Trans. Circuits Syst. Video Technol.*, Vol.12, No.5, May 2002.

硕士在读期间的研究成果

[1] 倪伟 郭宝龙 等, H.264 变换编码和量化算法的研究, 计算机工程与应用, No. 3, 2004.

[2] 倪伟 郭宝龙 王勇, H.26L 中新的帧内预测算法研究, 计算机工程与应用, No. 4, 2004.

[3] 倪伟 郭宝龙, 基于梯度的自适应频域帧内预测算法, 西安电子科技大学 2003 年研究生学术年会.

[4] 倪伟 郭宝龙, 一种三阶频域帧内预测算法, 计算机应用研究. (已录用)

H.264视频编码标准的关键技术研究

作者：倪伟
学位授予单位：西安电子科技大学
被引用次数：4次

引证文献(4条)

1.廖怡 基于H.264/AVC关键技术的研究[学位论文]硕士 2007

2.陈亚菲 基于视频编码标准的去块效应算法研究[学位论文]硕士 2007

3.胡伟 基于TMS320DM642的视频处理系统硬件设计[学位论文]硕士 2006

4.朱松超 H.264编码算法研究和基于FPGA的设计[学位论文]硕士 2006