图清单

表清单

注释表

VCL	Video Coding Layer (视频编码层)
UVCL	Universal Video Coding Layer (统一的视频编码层)
NAL	Network Abstraction Layer (网络提取层)
DCT	Discrete Cosine Transform (离散余弦变换)
MSE	Mean Square Error (均方误差)
PSNR	Peak Signal-to-Noise Rate (峰值信噪比)
CAVLC	Context Adaptive Variable Length Coding (基于上下文自适应的可变长编码)
CABAC	Context Adaptive Binary Arithmetic Coding (内容自适应二进制算术编码)
RDO	Rate-Distortion Optimization (率失真优化)
SA(T)D	Sum of Absolute Transform Difference (绝对变换差值和)
DC	Direct Current (直流)

承诺书

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

本人授权南京航空航天大学可以有权保留送交论文的复印件，允许论文被查阅和借阅,可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文。

作者签名：_____

日　　期：_____

摘　要

H.264 是国际电信联盟 ITU-T 的视频编码专家组 VCEG 和国际标准化组织 ISO/IEC 的活动图像专家组 MPEG 联合制定的视频编码新标准，其目的是为了获得更好的图像压缩效果和更好地适应不同的网络环境。但是 H.264 视频编码的高压缩率是建立在其算法的高复杂度基础上的，H.264 视频编码的巨大运算量成为其广泛应用的瓶颈。

为了降低 H.264 视频编码算法的计算复杂度、减少运算量、提高其实时性，本文首先阐述了 H.264 视频编码的主要流程，着重对 H.264 视频编码的帧内预测算法做了研究，并在分析其计算复杂度的基础上提出两种快速帧内预测模式选择算法：基于宏块内部相邻像素差的帧内宏块类型预判算法和改进的基于边缘方向直方图 Pan 算法。本文完成的主要工作如下：

1、在大量实验与深入研究的基础上，分析了宏块内部相邻像素差的特点和宏块预测类型的相关性，提出一种基于宏块内部相邻像素差的帧内宏块类型预判算法。在 H.264 测试模型 JM90 上实现该算法，其结果表明提出的预判算法在编码质量和码率性能基本不变的前提下，编码时间平均节省约 23.47%。

2、对具体帧内宏块类型下的预测模式快速选择算法进行研究，提出一种改进的基于边缘方向直方图 Pan 算法，并在 H.264 测试模型 JM90 上实现，其结果表明该算法在编码质量基本不变、码率稍有减小的情况下，编码时间平均节省约 62.32%。；把提出的帧内宏块类型预判算法与改进的 Pan 算法相结合进行实验，其结果表明该结合算法在保证失真率和码率性能基本不变的前提下，编码时间平均减少了 72.49%，大大提高了编码效率，对 H.264 的实时应用具有很大的意义。

关键词：H.264　帧内预测　模式选择　宏块预测类型　边缘矢量

ABSTRACT

H.264 is the lastest standard for video compression issued jointly by Video Coding Experts Group of the International Telecommunication Standardisation Sector and Moving Picture Experts Group of the International Organisation for Standardization and the Internation Electrotechnical Commission. It aims to provide enhanced coding efficiency as well as ensuring its suitability for transmission over various kinds of networks. However, the high efficiency of H.264 is at the expanse of its complexity, because of its high operations, it's hard to be used widely.

In order to reduce the complexity of H.264 video coding algorithm, the main process of H.264 is introduced firstly, and then researches in this paper focus on intra-prediction algorithms for H.264. Based on the analysis of the algorithm complexity, two fast intra-prediction mode decision algorithms for H.264 are present in this paper, in which maily complete the following work:

1. On the basis of large number of experiments and in-depth study, the correlation of the macroblock and the characteristics of the differences between the neighbouring pixcels of the macroblock is analyzed. Then macroblock type pre-decision algorithm for intra-prediction is proposed and the algorithm is implemented on H.264 test model JM90. Experimental results show that the pre-decision algorithm save about 23.47% of the encoding time with negligible loss of the quality.

2. An improved Pan algorithm based on edge direction histogram is present. Experimental results show that the improved Pan algorithm save about 62.32% of the encoding time with acceptable loss of the quality. Then we implement both improved Pan algorithm and the pre-decision algorithm on H.264 test model JM90. Experimental results show the combinative algorithm save about 72.49% of the encoding time when ensuring the capability of the rate distortion and the Bits rate.

Key Words: H.264, Intra-prediction, Mode decision, Macroblock type, edge vector

第一章 绪论

视觉是人类获取信息的重要途径，外部世界的信息大部分是通过视觉感知的。据统计，人类从外部获得的信息约有 60-75%来自视觉系统，也就是图像，这里的图像是广义的，例如照片、绘图等等。随着计算机、数字通信、多媒体和网络技术的发展，信息在急剧膨胀。图像与视频作为最重要的载体之一，已经深入人们的日常生活。人们不再仅仅满足于文字及声音，还希望看到生动的画面。绝大多数的娱乐活动，如电影、电视、VCD、DVD、VOD、电子游戏、卡拉 OK 等，工作学习所需的如可视电话、电视会议、多媒体邮件与工业实时监控等，都需要处理图像与视频信号。

1.1 视频压缩标准的发展

随着多媒体技术应用的不断增加，图像和视频编码成为一个广泛研究的课题，现在它的应用涉及到各行各业。不断增长的商业需求也推动着标准的形成，各种国际视频标准相继推出。目前从事视频压缩标准制定的国际组织主要有国际电信联盟 ITU-T 的视频编码专家组 VCEG(Video Coding Expert Group)和国际标准化组织 ISO/IEC 的运动图像专家组 MPEG(Motion Picture Expert Group)。两个标准化组织基于不同的应用需求，采用近似的压缩编码技术，分别制定了 H.26x 和 MPEG-x 系列视频压缩标准。其中 ITU-T 针对可视电话和视频会议等应用分别制定了 H.261、H.262、H.263、H.263+、H.263++、H.264/AVC；ISO/IEC 相继制定了 MPEG-1、MPEG-2、MPEG-4 等。以上国际压缩标准尽管应用领域不同，但是均采用了预测编码结合变换量化的混合编码模式。其中两大视频标准化组织于 1992 年联合提出的 MPEG-2/H.262 是现有最成功的国际视频压缩标准，目前又再次联手提出了 H.264/AVC 即 MPEG-4 第 10 部分。这些图像标准是图像编码技术走向实用的重要一步，也是图像编码技术的结晶。

1.1.1 H.26x 标准系列

(1)H.261

H.261[1]是最早出现的视频编码标准，是 ITU-T 的前身 CCITT 针对可视电话、会议电视和窄带 ISDN 等要求实时编解码和低延时应用提出的一个编码标准。它输出的码率是 P×64Kbit/s，其中 P 为 0 到 31 的整数。H.261 采用的算法主要是帧间预测和二维 DCT 变换的混合编码，该标准同时支持帧间编码和帧内编码，当帧间预测效率较低时，则直接采用 DCT 变换。

(2)H.263

H.263[2]是为低码率视频压缩提供的新标准，目的是支持码率小于 64Kbit/s 的应用。在 H.261 建议的基础上，H.263 进行了重要改进，采用了半像素精度的运动矢量搜索，增加了非限制运动矢量，提出了基于语法的算术编码、先进预测模式和 PB 帧编码等多个高级选项，从而达到了进一步降低码率和提高编码质量的目的。

1.1.2 MPEG 标准系列

(1)MPEG-1

MPEG-1[3][4]标准制定目标码率是 1.2Mbit/s，对于 CIF(352×288)格式图像可以达到实时播放，是为只读 CD-ROM 光盘的视频存储和播放所制定的。类似于 H.261 标准，MPEG-1 也采用运动补偿和二维 DCT 变换，量化后的 DCT 系数进行变长编码，同时每个数据块的直流分量 DC 进行差分编码。

(2)MPEG-2

MPEG-2[5][6]的视频编码部分就是 H.262，该标准主要是针对数字视频广播 DVB(Digital Video Broadcast)、高清晰度电视 HDTV(High Digital Television)和数字光盘 DVD(Digital Video Discard)等 4～9Mbit/s 运动图像的编码。MPEG-2 作为一个得到广泛应用的国际标准，成功之处在于提出了通用的压缩编码方法，定义了不同的“档次”(profile)和“等级”(level)，可满足不同图像分辨率及相应的存储成本和处理速度的需要。与 H.261 视频标准相比，MPEG-2/H.262 开始使用半像素精度的运动矢量搜索，引入了“帧”和“场”的编码方法，支持可分级性技术，包括空间分级性、时间可分级性和信噪比可分级性等。

(3)MPEG-4

MPEG-4[7][8]标准既能够支持低码率的视频应用，也能够支持广播级的视频应用。与其他标准相比，MPEG-4 标准中引入了视听对象 AVO(Audio-Visual Obj

ect)的概念，这种编码模式能有效提高视频通信的交互能力和编码效率。MPEG-4 还采用了诸如形状编码和自适应 DCT 技术以支持任意形状视频对象的编码，以及基于内容的可分级操作。其自然视频编码的基本框架和 H.263 标准是接近的，但是由于“基于对象的编码”尚有技术障碍，在技术专利保护问题上迟迟难以找到有效的收费形式，因此该标准目前仍然没有得到普遍应用。

1.1.3 H.264 视频压缩的特点

H.264/AVC[9]是国际电信联盟 ITU-T 的视频编码专家组 VCEG 和国际标准化组织 ISO/IEC 的运动图像专家组 MPEG 联合开发的一个新的数字视频编码标准。

H.264 标准可分为三档：

(1)基本档次(其简单版本，应用面广)；

(2)主要档次(采用了多项提高图像质量和增加压缩比的技术措施，可用于 SDTV、HDTV 和 DVD 等)；

(3)扩展档次(可用于各种网络的视频流传输)。

在同等图像质量的情况下，H.264 不仅比 H.263 和 MPEG-4 节约了 50%的码率，而且对网络传输具有更好的支持功能。它引入了面向 IP 包的编码机制，有利于网络中的分组传输，支持网络中视频的流媒体传输。H.264 具有较强的抗误码特性，可适应丢包率高、干扰严重的无线信道中的视频传输。H.264 支持不同网络资源下的分级编码传输，从而获得平稳的图像质量。

H.264 标准压缩系统由视频编码层(VCL)和网络提取层(Network Abstraction Layer，NAL)两部分组成。VCL 中包括 VCL 编码器与 VCL 解码器，主要功能是视频数据压缩编码和解码，它包括运动补偿、变换编码、熵编码等压缩单元。NAL 则用于为 VCL 提供一个与网络无关的统一接口，它负责对视频数据进行封装打包后使其在网络中传送，它采用统一的数据格式，包括单个字节的包头信息、多个字节的视频数据与组帧、逻辑信道信令、定时信息、序列结束信号等。包头中包含存储标志和类型标志。存储标志用于指示当前数据不属于被参考的帧。类型标志用于指示图像数据的类型。VCL 可以传输按当前的网络情况调整的编码参数。

H.264 和 H.261、H.263 一样，也是采用 DCT 变换编码加 DPCM 的差分编码，即混合编码结构。同时，H.264 在混合编码的框架下引入了新的编码方式，

提高了编码效率，更贴近实际应用。

H.264 没有繁琐的选项，而是力求简洁的“ 回归基本 ”，它具有比 H.263++ 更好的压缩性能，又具有适应多种信道的能力；它的应用目标广泛，可满足各种不同速率、不同场合的视频应用，具有较好的抗误码和抗丢包的处理能力；它的基本系统无需使用版权，具有开放的性质，能很好地适应 IP 和无线网络的使用，这对目前因特网传输多媒体信息、移动网中传输宽带信息等都具有重要意义。

尽管 H.264 编码基本结构与 H.261、H.263 是类似的，但它在很多环节做了改进，现列举如下：

1、多种更好的运动估计

高精度估计：在 H.263 中采用了半像素估计，在 H.264 中则进一步采用 1/4 像素甚至 1/8 像素的运动估计。即真正的运动矢量的位移可能是以 1/4 甚至 1/8 像素为基本单位的。显然，运动矢量位移的精度越高，则帧间残余误差越小，传输码率越低，即压缩比越高。在 H.264 中采用了 6 阶 FIR 滤波器的内插获得 1/2 像素位置的值。当获得 1/2 像素值后，1/4 像素值可通过线性内插获得，对于 4:1:1 的视频格式，亮度信号的 1/4 像素精度对应于色度部分的 1/8 像素的运动矢量，因此需要对色度信号进行 1/8 像素的内插运算。

理论上，如果将运动补偿的精度增加一倍(例如从整像素精度提高到 1/2 像素精度)，可有 0.5bit/Sample 的编码增益，但实际验证发现在运动矢量精度超过 1/8 像素后，系统基本上就没有明显增益了，因此，在 H.264 中，只采用了 1/4 像素精度的运动矢量模式，而不是采用 1/8 像素的精度。

多宏块划分模式估计：在 H.264 的预测模式中，一个宏块(MB)可划分成 7 种不同模式的尺寸，这种多模式的灵活、细微的宏块划分，更切合图像中的实际运动物体的形状，于是，在每个宏块中可包含有 1、2、4、8 或 16 个运动矢量。

在 H.264 中，可采用多个参考帧的运动估计，即在编码器的缓存中存有多个刚刚编码好的参考帧，编码器从其中选择一个给出更好的编码效果的作为参考帧，并指出是哪个帧被用于预测，这样就可获得比只用前一参考帧更好的编码效果。

2、小尺寸 4×4 的整数变换

以往视频编码中常用单位为 8×8 块。在 H.264 中却采用小尺寸的 4×4 块，

由于变换块的尺寸变小了，运动物体的划分就更为精确。这种情况下，图像变换过程中的计算量小了，而且在运动物体边缘的衔接误差也大为减少。

当图像中有较大面积的平滑区域时，为了不产生因小尺寸变换带来的块间灰度差异，H.264 可对帧内宏块亮度数据的 16 个 4×4 块的 DCT 系数进行第二次 4×4 块的变换，对色度数据的 4 个 4×4 块的 DC 系数(每个小块一个，共 4 个 DC 系数)进行 2×2 块的变换。

H.264 不仅使图像变换块尺寸变小，而且这个变换是整数操作，不是实数运算，即编码器和解码器的变换和反变换的精度相同，没有“ 反变换误差 ”。

3、更精确的帧内预测

在 H.264 中，每个 4×4 块中的每个像素都可用 17 个最接近先前已编码的像素的不同加权和来进行帧内预测。

4、统一的 VLC

H.264 中关于熵编码有两种方法。

统一的 VLC(即 UVLC：Universal VLC)。UVLC 使用一个相同的码表进行编码，而解码器很容易识别码字的前缀，UVLC 在发生比特错误时能快速获得帧同步。

内容自适应二进制算术编码(CABAC：Context Adaptive Binary Arithmetic Coding)。其编码性能比 UVLC 稍好，但复杂度较高。

5、性能优势

H.264 与 MPEG-4、H.263++编码性能对比采用了以下 6 个测试速率：32kbit/s、10F/s 和 QCIF；64kbit/s、15F/s 和 QCIF；128kbit/s、15F/s 和 CIF；256kbit/s、15F/s 和 QCIF；512kbit/s、30F/s 和 CIF；1024kbit/s、30F/s 和 CIF。测试结果标明，H.264 具有比 MPEG 和 H.263++更优秀的 PSNR 性能。H.264 的 PSNR 比 MPEG-4 平均要高 2dB，比 H.263++平均要高 3dB。

1.2 视频质量评价

对压缩后的视频进行质量评估是一个重要的问题。一般来说，视频质量评价方法分为主观评价和客观评价两种[10]。

1.2.1 主观评价

主观评价方法是由评价者直接对一段视频进行观察，从感觉上去度量其失真度，给出质量评价级别，对所有评价者给出的分数进行加权平均，所得结果既为主观评价结果。这种评价结果必然符合人的视觉感受。但人的主观感受不能用数学模型描述，无法直接用于视屏压缩编码过程中质量评价与控制；另外，主观评价容易受到个体因素的影响，如年龄、性格、教育程度、背景以及评价是的心情等。

1.2.2 客观评价

客观评价是用重建图像与原始图像的误差来衡量，常用的有均方误差(MSE)和峰值信噪比(PSNR)两种。

均方误差定义为：

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}[f(i,j) - f^{'}(i,j)]^2 \tag{1-1}$$

其中：M、N 表示图像宽和高的像素点数；$f(i,j)$表示原始图像的像素值，$f^{'}(i,j)$表示重建图像的像素值。

峰值信噪比定义为：

$$PSNR = 10 \times \log_{10}\frac{255^2}{MSE} \tag{1-2}$$

可见二者是一一对应的，实际应用中，PSNR 比 MSE 更为常用。

本论文中使用式(1-2)的 PSNR 作为客观质量的评价标准。

1.3 国内外研究现状

目前国内外对 H.264 视频编码标准都有一定的研究。算法上，主要是对 H.264 的关键技术如：帧内模式选择，帧间模式选择，快速的高精度运动估计算法等的研究。其中帧内模式选择技术研究现状如下：

由于多种预测模式的存在，要求在编码时对这些模式进行择优，择优的方法很多，有必要对其进行研究。择优过程中，H.264/AVC 将基于 T.Wiegand[19] 提出的率失真优化策略(RDO)作为重要可选模式，对于帧内预测，快速方法有

Feng Pan 等在 JVT 会议中提出的基于边缘方向直方图的方法和 Bojun Meng 等提出的基于分组象素点的方法等，他们在不同程度上优化了原全搜索方法。

1.4 本文研究方向和章节安排

为了降低 H.264 编码算法的计算复杂度，本论文对 H.264 视频编码算法进行了较为深入的研究。着重对 H.264 帧内预测算法做了研究，并提出一种基于宏块内部相邻像素差的帧内宏块类型预判算法和改进的 Pan 算法。论文写作结构安排如下：

1、第一章，简要回顾了视频编码发展的历史和背景，根据目前国内外研究状况提出了本课题研究的意义和本文研究的内容。

2、第二章，概述了最新视频编码标准 H.264，介绍了其采用的新技术：帧内预测、帧间预测、变换与量化、熵编码和去方块滤波。

3、第三章，详细介绍帧内预测的编码方法和目前测试模型中广泛使用的全搜索帧内预测模式选择算法流程，分析了算法的计算复杂度。然后介绍了几种现有的典型帧内预测快速算法，并阐述其优缺点。这些先验知识对于后面的研究至关重要。

4、第四章，在大量实验与深入研究的基础上，分析了宏块内部相邻像素差的特点和宏块预测类型的相关性，提出一种基于宏块内部相邻像素差的快速帧内宏块类型预判算法。在 H.264 测试模型 JM90 上实现并给出实验结果。

5、第五章，对基于边缘方向直方图的 Pan 算法进行改进，提出一种改进算法，在 H.264 测试模型 JM90 上实现并给出实验结果；并把改进的 Pan 算法和第四章中提出的快速帧内宏块类型预判算法相结合进行实验，结果表明，该算法在保证失真率和码率性能基本不变的前提下，平均减少了 72.49%的编码时间，大大提高了编码效率，对 H.264 的实时应用具有很大的意义。

6、第六章是对全文的总结及展望，指出本文所做的研究工作并展望以后将继续研究的内容。

1.5 本课题研究的意义

H.264 是最新的视频标准，与 H.263，MPEG-4 视频标准相比，H.264 编码器能在保持相同质量的情况下，大约节省 50%的码率。但压缩效率的提高是以算法计算复杂度的增加为代价的，这使得 H.264 很难应用于实时性要求较强的场合。为了降低 H.264 编码器的复杂度，最近国内外主要从帧间模式选择、帧内模式选择、运动估计等方面开展研究。对帧内模式选择算法的研究和改进大都集中在减少候选预测模式个数的方法上[13,27,29]，在预测前先判断宏块预测类型的方法较少。本论文以降低 H.264 编码器算法复杂度为出发点，深入研究了帧内预测算法，提出了一种基于宏块内部相邻像素差的帧内宏块类型预判算法和一种改进的 Pan 算法，大大降低了 H.264 编码器的复杂度，减少了编码的运算量，对 H.264 的实时应用具有很大的意义。

第二章 H.264 视频编码标准

随着社会的信息化，人们对图像业务的需求越来越大，同时对视频图像的质量也提出更高的要求。宽带通信网的急速发展，尤其是移动通信网络的日新月异，大大的促进了视频通信的大面积应用。3G 发展的主要目的就是让移动网不仅能支持语音业务，更主要的用途是承载视频业务。同时随着电子技术的飞跃发展，媒体处理器的计算能力也正以成倍的速度增长，为终端设备支持高质量的视频提供了可能。在这种情况下，发展一种高编码性能和高抗误码性能的视频编码技术已成为趋势。

H.264 是继 H.263 和 MPEG-4 的下一代视频编码标准，在体系上也是 MPEG-4 的第十部分。早在 1997 年，ITU-T 的视频编码专家组(VCEG ,VideoCoding Experts Group)就已经开始了 H.264 前身的研究工作，后来 ISO/IEC 的活动图像专家组(MPEG)看到 H.264 的良好发展也加入进来，与 VCEG 一起成立了联合视频组(JVT, Joint VideoTeam)，共同致力于 H.264 标准也即 MPEG-4 的第 10 部分“高级视频编码算法”的开发工作。H.264[14]优越性能源于它对编码系统主要摸块的技术改进，本章主要介绍它的主要模块。

2.1 H.264 编码器及结构

2.1.1 H.264 编解码器的特点

H.264 并没有明确地规定一个编解码器如何实现，而是规定了一个已编码的视频比特流的句法和该比特流的解码方法，各个厂商的编码器和解码器在此框架下应能够互通，在实现上有较大的灵活性，而且有利于相互竞争。H.264[15]编码器和解码器的功能组成分别如图 2.1 和图 2.2 所示。

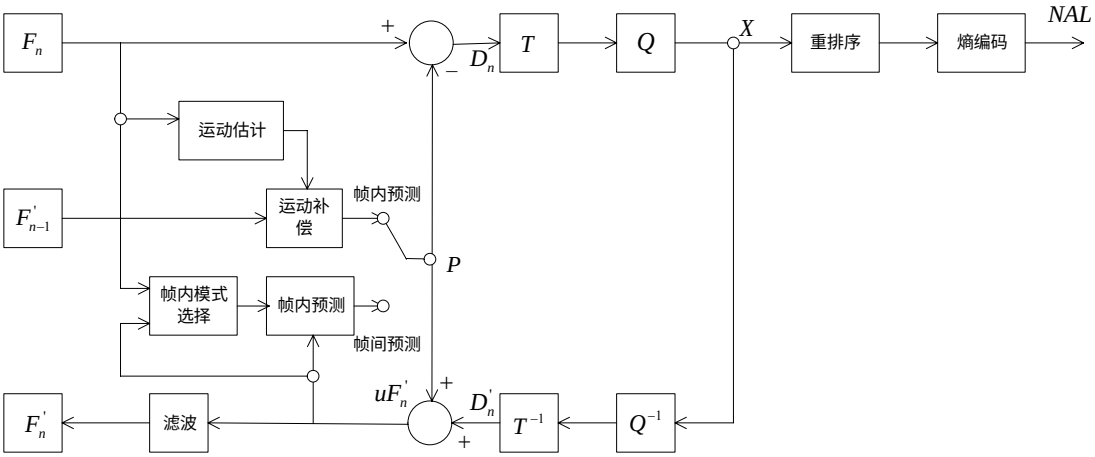


图 2.1 H.264 编码器

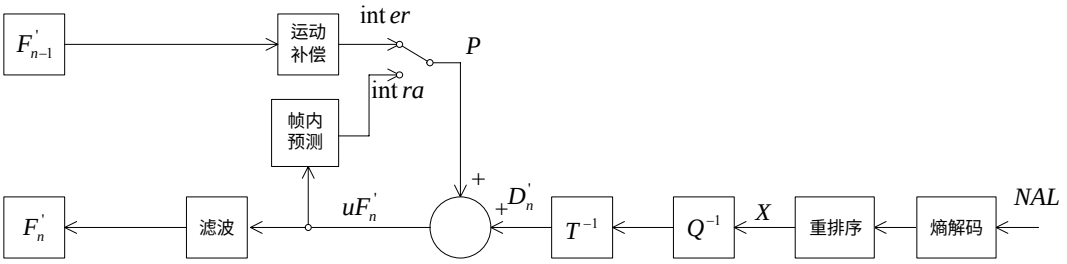


图 2.2 H.264 解码器

图 2.1 和图 2.2 中，T 代表整数变换，Q 代表量化，T^{-1}代表整数反变换，Q^{-1}代表反量化，P 代表预测值。从图 2.1 和图 2.2 中可见，H.264 和基于以前标准(如 H.261、H.263、MPEG-1、MPEG-4)中的编解码器功能块的组成并没有太大的区别，主要的不同在于各功能块的细节。由于视频的内容时刻在变化，有时空间细节很多，有时大面积的平坦，这时内容的多变性就必须采用相应的自适应的技术措施；由于信道在环境恶劣下也是多变的，例如互联网，有时畅通，有时不畅，有时阻塞；又如无线网络，有时发生严重衰落，有时衰耗很小，这就要求采取相应的自适应方法来对抗这种信道畸变带来的不良影响。这两方面的多变带来了自适应技术的复杂性。H.264 视频编码就是利用实现的复杂性获得压缩性能的明显改善。由于大规模集成电路技术和工艺的迅猛进步，今天已完全具备了实现的可能性。

2.1.2 H.264 编码器

编码器采用的仍是变换和预测的混合编码方法。

在图 2.1 中，输入的帧 F_n 以宏块为单位被编码器处理。首先，按帧内或帧间预测编码的方法进行处理。

如果采用帧间预测编码，其预测值 PRED(图中用 P 表示)是由当前帧中已编码的参考图像经运动补偿(MC)后得到的，其中参考图像用式 F'_{n-1} 表示。为了提高预测精度，从而提高压缩比，实际的参考图像可以在过去或未来(指显示次序上)已编码解码重建的帧中进行选择，帧间预测时的参考帧不经过滤波。

预测值 PRED 和当前块相减后，产生一个残差块 D_n，经块变换、量化后产生一组量化后的变换系数 X，再经熵编码，与解码所需的边信息(如预测模式量化参数、运动矢量等)一起组成压缩后的码流，经 NAL(网络适应层)供传输和存储用。

正如上述，为了提供进一步预测用的参考图像，编码器必须有重建图像的功能。因此必须使残差图像经反量化、反变换后得到的 D'_n 与参考值 P 相加，得到 uF'_n 式(未经滤波的帧)。为了去除编码解码环路中产生的噪声，提高参考帧的图像质量，从而提高压缩图像性能，设置了一个环路滤波器，滤波后的输出 F'_n 即为重建图像，可用作参考图像。

2.1.3 H.264 解码器

由图 2.1 可知，由编码器的 NAL 输出一个压缩后的 H.264 压缩比特流。在图 2.2 中，经熵解码得到量化后的一组变换系数 X，再经反量化、反变换，得到残差 D'_n。利用从该比特流解码出的头信息，解码器就产生一个预测块 PRED，它和编码器中的原始 PRED 是相同的。当该解码器产生的 PRED 与残差 D'_n 相加后，就产生 uF'_n，再经滤波后，最后就得到重建的 F'_n，这个 F'_n 就是最后的解码输出图像。

2.2 视频格式和宏块编码类型

1、视频格式

H.264 采用 $Y-C_b-C_r$ 对图像进行描述，即一个亮度分量 Y 和两个色差分量

C_b和C_r，H.264 支持取样格式为 4: 2: 0 的连续或隔行视频的编码和解码，见图 2.3。

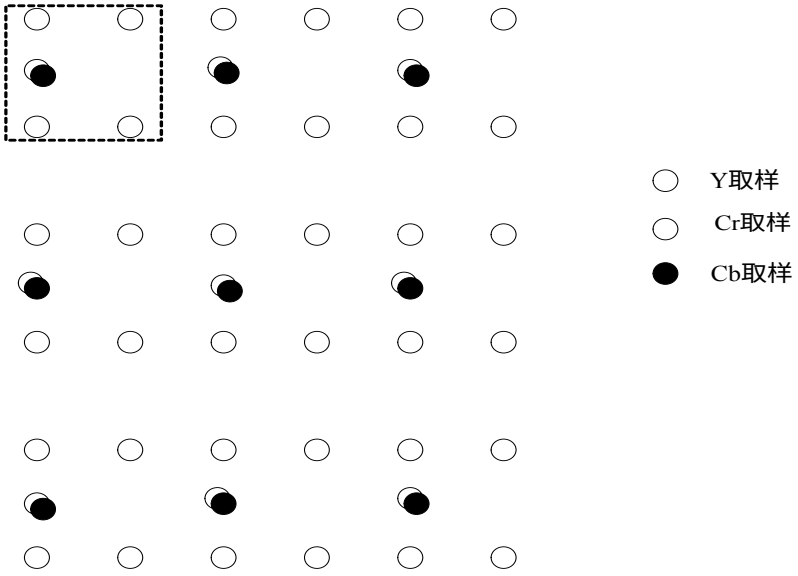


图 2.3 视频取样格式

2、宏块编码类型

如前所述，编码器首先将一幅完整图像分为多个宏块(MB)，然后对这些小块分别进行预测、量化、编码等处理。这样做的目的是为了加快处理速度，提高编码效率。因此，对图像的压缩实际上是在宏块级(MB)进行的。

宏块编码类型(MB_Type)字段用来传送本宏块的编码类型，帧内预测编码(Intra)有两类：分别是以 4×4 小块(block)为单位的 9 种帧内预测模式以及以整个宏块(16×16)为单位的 4 种帧内预测模式。一共是 13 种帧内预测模式。

帧间预测编码((Inter)有三类：第一种是跳过(SKIP)，表示本宏块与上一帧相应宏块完全相同，没有附加信息需要传送，在解码端只需要将上一帧解码后的图像中相应宏块拷贝一份进行图像重建即可。第二种代表本宏块采用帧间预测的方法，而且进行运动搜索的预测块的大小为 N×M，根据 N 和 M 的不同，一共有 7 种搜索模式，每种模式所需要传的运动矢量的个数也由 1 个到 16 个不等。最后一种是帧内预测方式，当帧内预测比帧间预测的压缩效果好时采用这种预测。

2.3 帧内预测编码

为了提高编码效率，在帧内编码[16]中，H.264 采用帧内预测模式，这样能够更好的消除图像的空间冗余。预测分为 4×4 子块和 16×16 子块的两种模式。帧内预测的原理主要依据图像中相邻宏块具有相似性的特点，因此通过已编码的宏块(特别是在当前宏块左边和上方的相邻宏块)来预测当前宏块，然后对当前宏块与预测值的差值进行变换编码。

4×4 亮度块帧内预测[17]共有 9 种预测模式，16×16 亮度宏块有 4 种预测模式。16×16 适用与在变化很小而面积较大的区域。

2.3.1 4×4 亮度预测模式

图 2.4 是 4×4 亮度预测模式图。图中的大写字母 A~Q 表示来自邻近块，已解码重构但尚未进行滤波的像素(当这些像素在图象外部或编码次序上滞后于被预测像素时称为不可得)，小写字母 a~p 表示将要被预测的 16 个 4×4 亮度块像素。图 2.5 为 4×4 块 9 种预测模式的预测模式图。下面给出 4×4 亮度块帧内预测各预测模式的具体计算方法。在这里以像素 a 的位置作为坐标原点，用 $\mathrm{pred4x4}_{\mathrm{L}}[x,y]$，$x,y=0\mathrm{K}\ 3$ 表示被预测的像素 a~p。用 $\mathrm{p}[x,-1]$，$\mathrm{x}=0\mathrm{K}\ 7$ 表示 A-H，用 $\mathrm{p}[-1,y]$，$y=0\mathrm{K}\ 7$ 表示 I-P，用 $p[-1,-1]$ 表示 Q。

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				
M								
N								
O								
P								

图 2.4 4×4 亮度预测模式图

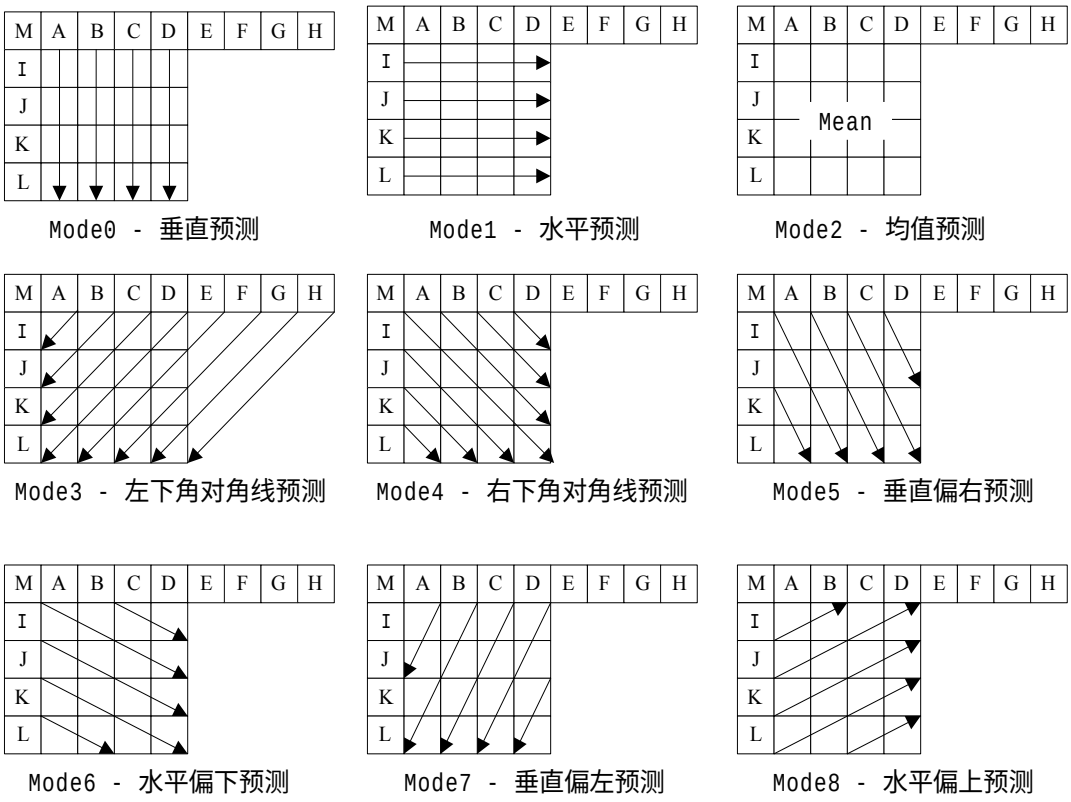


图 2.5 4×4 亮度块预测模式

- 模式 0：垂直预测

当像素 A、B、C、D 可得时可以使用。

$$pred4\times4_L[x,y]=p[x,-1] \tag{2-1}$$

- 模式 1：水平预测

当像素 I，J，K，L 可得时可以使用。

$$pred4\times4_L[x,y]=p[-1,y] \tag{2-2}$$

- 模式 2：均值预测

当 A、B、C、D、I、J、K、L 都可得时：

$$\begin{aligned} pred4\times4_L[x,y]=(&p[0,-1]+p[1,-1]+p[2,-1]+p[3,-1]+p[-1,0]\\ &+p[-1,1]+p[-1,2]+p[-1,3]+4)>>3 \end{aligned} \tag{2-3}$$

当仅 A、B、C、D 可得：

$$pred4\times4_L[x,y]=(p[0,-1]+p[1,-1]+p[2,-1]+p[3,-1]+2)>>2 \tag{2-4}$$

当仅 I，J，K，L 可得：

$$pred4\times4_L[x,y]=(p[-1,0]+p[-1,1]+p[-1,2]+p[-1,3]+2)>>2 \tag{2-5}$$

当都不可得：

$$pred4\times4_L[x,y]=128 \tag{2－6}$$

- 模式 3：左下角对角线预测

当 A、B、C、D、E、F、G、H 都可以得时可以使用。

当 $x=3$ 且 $y=3$ 时：

$$pred4\times4_L[x,y]=(p[6,-1]+3\cdot p[7,-1]+2)>>2 \tag{2－7}$$

否则：

$$\begin{aligned}pred4\times4_L[x,y]=(&p[x+y,-1]+2\cdot p[x+y+1,-1]\\&+p[x+y+2,-1]+2)>>2\end{aligned} \tag{2－8}$$

- 模式 4：右下角对角线预测

只有 A、B、C、D、I、J、K、L、Q 都可得时可以使用。

当 x 小于 y 时：

$$\begin{aligned}pred4\times4_L[x,y]=(&p[-1,y-x-2]+2\cdot p[-1,y-x-1]\\&+p[-1,y-x]+2)>>2\end{aligned} \tag{2－9}$$

当 x 大于 y 时：

$$\begin{aligned}pred4\times4_L[x,y]=(&p[x-y-2,-1]+2\cdot p[x-y-1,-1]\\&+p[x-y,-1]+2)>>2\end{aligned} \tag{2－10}$$

否则：

$$pred4\times4_L[x,y]=(p[0,-1]+2\cdot p[-1,-1]+p[-1,0]+2)>>2 \tag{2－11}$$

- 模式 5：垂直偏右预测

只有 A、B、C、D、I、J、K、L、Q 都可得时可以使用。

令变量 $zVR=2\cdot x-y$。

当 zVR 等于 0，2，4，6 时：

$$pred4\times4_L[x,y]=(p[x-(y>>1)-1,-1]+p[x-(y>>1),-1]+1)>>1 \tag{2－12}$$

当 zVR 等于 1，3，5 时：

$$\begin{aligned}pred4\times4_L[x,y]=(&p[x-(y>>1)-2,-1]+2\cdot p[x-(y>>1)-1,-1]\\&+p[x-(y>>1),-1]+2)>>2\end{aligned} \tag{2－13}$$

当 zVR 等于-1 时：

$$pred4\times4_L[x,y]=(p[-1,0]+2\cdot p[-1,-1]+p[0,-1]+2)>>2 \tag{2－14}$$

否则(zVR 等于-2,-3):

$$pred4\times4_L[x,y]=(p[-1,y-1]+2\cdot p[-1,y-2]+p[-1,y-3]+2)>>2 \tag{2－15}$$

➢ 模式 6：水平偏下预测

只有 A、B、C、D、I、J、K、L、Q 都可得时可以使用。

令变量 $zVR = 2 \cdot y - x$。

当 *zVR* 等于 0，2，4，6 时：

$$pred4\times4_L[x,y] = (p[-1, y-(x>>1)-1] + p[-1, y-(x>>1)] + 1) >> 1 \quad (2-16)$$

当 *zVR* 等于 1，3，5 时：

$$\begin{aligned} pred4\times4_L[x,y] = (&p[-1, y-(x>>1)-2] + 2\cdot p[-1, y-(x>>1)-1] \\ &+ p[-1, y-(x>>1)] + 2) >> 2 \end{aligned} \quad (2-17)$$

当 *zVR* 等于-1 时：

$$pred\,4\times4_L[x,y] = (p[-1,0] + 2\cdot p[-1,-1] + p[0,-1] + 2) >> 2 \quad (2-18)$$

否则(*zVR* 等于-2,-3):

$$pred\,4\times4_L[x,y] = (p[x-1,-1] + 2\cdot p[x-2,-1] + p[x-3,-1] + 2) >> 2$$

➢ 模式 7：垂直偏左预测

当 A、B、C、D、E、F、G、H 都可以得时可以使用。

当 y 等于 0，2 时：

$$pred4\times4_L[x,y] = (p[x+(y>>1),-1] + p[x+(y>>1)+1,-1] + 1) >> 1 \quad (2-19)$$

否则：

$$\begin{aligned} pred4\times4_L[x,y] = (&p[x+(y>>1),-1] + 2\cdot p[x+(y>>1)+1,-1] \\ &+ p[x+(y>>1)+2,-1] + 2) >> 2 \end{aligned} \quad (2-20)$$

➢ 模式 8：水平偏上预测

当像素 I，J，K，L 可得时可以使用。

令变量 $zHU = x + 2\cdot y$

当 *zHU* 等于 0，2，4 时：

$$pred4\times4_L[x,y] = (p[-1, y+(x>>1)] + p[-1, y+(x>>1)+1] + 1) >> 1 \quad (2-21)$$

当 *zHU* 等于 1，3 时：

$$\begin{aligned} pred4\times4_L[x,y] = (&p[-1, y+(x>>1)] + 2\cdot p[-1, y+(x>>1)+1] \\ &+ p[-1, y+(x>>1)+2] + 2) >> 1 \end{aligned} \quad (2-22)$$

当 *zHU* 等于 5 时：

$$pred4\times4_L[x,y] = (p[-1,2] + 3\cdot p[-1,3] + 2) >> 2 \quad (2-23)$$

否则

$$pred4\times4_L[x,y] = p[-1,3] \quad (2-24)$$

2.3.2 16×16 亮度预测模式

对 16×16 亮度宏块帧内预测，共有 256 个被预测像素，用 $pred(x,y)$ $x,y=0\text{K}\ 15$ 表示。邻近块已解码重构的像素用 $P(x,-1)\quad x=-1\text{K}\ 15$ 和 $p(-1,y)$ $y=0\text{K}\ 15$ 表示。图 2.6 是 16×16 的预测模式图：

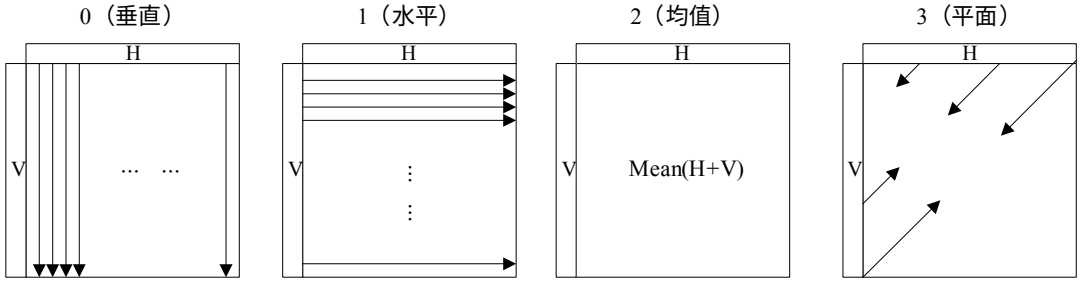


图 2.6 16×16 亮度块预测模式

- 模式 0：垂直预测

当所有邻近得像素 $p(-1,y),\ x=0\text{K}\ 15$ 都可以使用。

$$pred(x,y)=p(x,-1),x,y=0\text{K}\ 15 \tag{2－25}$$

- 模式 1：水平预测

当所有邻近得像素 $p(-1,y)$，$y=0\text{K}\ 15$ 可得时可以使用

$$pred(x,y)=p(-1,y),x,y=0\text{K}\ 15 \tag{2－26}$$

- 模式 2：均值预测

当所有邻近的像素 $p(x,-1)$，$p(-1,y)$，$x,y=0\text{K}\ 15$ 可得时：

$$pred(x,y)=\left[\sum_{x'=0}^{15}p(x',-1)+\sum_{y'=0}^{15}p(-1,y')+16\right]>>5 \tag{2－27}$$

当仅有 $p(-1,y)$ 时可得：

$$pred(x,y)=\left[\sum_{y'=0}^{15}p(-1,y')+8\right]>>4\qquad x,y=0\text{K}\ 15 \tag{2－28}$$

当仅有 $p(x,-1)$ 可得时：

$$pred(x,y)=\left[\sum_{x'=0}^{15}p(x'-1)+8\right]>>4\qquad x,y=0\text{K}\ 15 \tag{2－29}$$

否则：

$$pred(x,y)=128\qquad\qquad x,y=0\text{K}\ 15 \tag{2－30}$$

➢ 模式 3：平面预测

当所有邻近的像素 $p(x,-1)$， $p(-1,y)$， $x,y=0\text{K}\ 15$ 可得可以使用。

$$pred(x,y)=Clip1((a+b\cdot(x-7)+c\cdot(y-7)+16)>>5) \qquad (2-31)$$

$a=16\cdot(p(-1,15)+p(15,-1)$、 $b=(5\cdot H+32)>>6$、 $c=(5\cdot V+32)>>6$

$$clip1(x)=\begin{cases}0 & ; \quad x<0\\ 255 & ; \quad x>255\\ x & ; \quad otherwise\end{cases}$$

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

$$V=\sum_{y'=0}^{7}(y'+1)\cdot(p(-1,8+y')-p(-1,6-y'))$$

2.3.3 8×8 色度预测模式

色度宏块包含 U、V 两个色度分量宏块。在进行 8×8 色度宏块帧内预测时，对两个宏块使用相同的预测模式。与 16×16 亮度宏块帧内预测的各个预测模式相比，除 DC 预测外，其余预测模式十分类似。对于一个色度宏块，共有 64 个被预测像素，用 $\Pr ed_C(x,y)\quad x,y=0\Lambda\ 7$ 表示。邻近块已解码重构的像素用 $P_C(x,-1)\quad x=-1\Lambda\ 7$ 和 $P_C(-1,y)\quad y=0\Lambda\ 7$ 表示

➢ 模式 0：DC 预测

对于 $\Pr ed_C(x,y)\quad x,y=0\Lambda\ 3$，按如下方式计算。

当所有邻近的像素 $P_C(x,-1)$、 $P_C(-1,y),x,y=0\Lambda\ 3$ 可得时：

$$\Pr ed_C(x,y)=\left(\sum_{x'=0}^{3}P[x',-1]+\sum_{y'=0}^{3}p[-1,y']+4\right)>>3,x,y=0\Lambda\ 3 \qquad (2-32)$$

当所有邻近得像素 $P_C(x,-1)\quad x=0\Lambda\ 3$ 可得时：

$$\Pr ed_C(x,y)=\left(\sum_{x'=0}^{3}P[x',-1]+2\right)>>2,x,y=0\Lambda\ 3 \qquad (2-33)$$

当所有邻近得像素 $P_C(-1,y),y=0\Lambda\ 3$ 可得时：

$$\Pr ed_C(x,y)=\left(\sum_{y'=0}^{3}P[-1,y']+2\right)>>2,x,y=0\Lambda\ 3 \qquad (2-34)$$

否则：

$$\Pr ed_C(x,y)=128,x,y=0\Lambda\ 3 \qquad (2-35)$$

对于 $\Pr ed_C(x,y)\quad x=4\Lambda\ 7, y=0\Lambda\ 3$，按如下方式计算。

当所有邻近得像素 $P_C(x,-1)\quad x=4\Lambda\ 7$ 可得时：

$$\Pr ed_C(x,y)=\left(\sum_{x'=4}^{7}P[x',-1]+2\right)>>2, x=4\Lambda\ 7, y=0\Lambda\ 3 \quad (2-36)$$

当所有邻近得像素 $P_C(-1,y)$，$y=0\Lambda\ 3$ 可得时：

$$\Pr ed_C(x,y)=\left(\sum_{y'=0}^{3}P[-1,y']+2\right)>>2, x=4\Lambda\ 7, y=0\Lambda\ 3 \quad (2-37)$$

否则：

$$\Pr ed_C(x,y)=128, x=4\Lambda\ 7, y=0\Lambda\ 3 \quad (2-38)$$

对于 $\Pr ed_C(x,y)\quad x=0\Lambda\ 3, y=4\Lambda\ 7$，按如下方式计算。

当所有邻近得像素 $P_C(-1,y)$，$y=4\Lambda\ 7$ 可得时：

$$\Pr ed_C(x,y)=\left(\sum_{y'=4}^{7}P[-1,y']+2\right)>>2, x=0\Lambda\ 3, y=4\Lambda\ 7 \quad (2-39)$$

当所有邻近的像素 $P_C(x,-1), x=0\Lambda\ 3$ 可得时：

$$\Pr ed_C(x,y)=\left(\sum_{x'=0}^{3}P[x',-1]+2\right)>>2, x=0\Lambda\ 3, y=4\Lambda\ 7 \quad (2-40)$$

对于 $\Pr ed_C(x,y)\quad x,y=4\Lambda\ 7$，按如下方式计算。

当所有邻近的像素 $P_C(x,-1)$、$P_C(-1,y), x,y=0\Lambda\ 3$ 可得时：

$$\Pr ed_C(x,y)=\left(\sum_{x'=4}^{7}P[x',-1]+\sum_{y'=4}^{7}P[-1,y']+4\right)>>3, x,y=4\Lambda\ 7 \quad (2-41)$$

当所有邻近的像素 $P_C(x,-1)$，$x=4\Lambda\ 7$ 可得时：

$$\Pr ed_C(x,y)=\left(\sum_{x'=4}^{7}P[x',-1]+2\right)>>2, x,y=4\Lambda\ 7 \quad (2-42)$$

当所有邻近的像素 $P_C(-1,y)\quad x=4\Lambda\ 7$ 可得时：

$$\Pr ed_C(x,y)=\left(\sum_{y'=4}^{7}P[-1,y']+2\right)>>2, x,y=4\Lambda\ 7 \quad (2-43)$$

否则：

$$\Pr ed_C(x,y)=128, x,y=4\Lambda\ 7 \quad (2-44)$$

- 模式 1：水平预测

当所有邻近的像素 $P_C(-1,y), y=0\Lambda\ 7$ 可得时可以使用

$$\Pr ed_C(x,y)=P_C(-1,y), x,y=0\Lambda\ 7 \quad (2-45)$$

- 模式 2：垂直预测

当所有邻近的像素 $P_C(x,-1), x=0\Lambda\ 7$ 可得时可以使用

$$\Pr ed_C(x,y)=P_C(x,-1), x,y=0\Lambda\ 7 \tag{2－46}$$

- 模式 3：平面预测

当所有邻近的像素 $P_C(x,-1)$、$P_C(-1,y)\quad x,y=-1\Lambda\ 7$ 可得可以使用

$$\begin{aligned}&\Pr ed_C(x,y)=Clip1((a+b\cdot(x-3)+c\cdot(y-3)+16)>>5, x,y=0\Lambda\ 7\\&a=16\cdot(P(-1,7)+P(7,-1))、\ b=(17\cdot H+16)>>5、\ c=(17\cdot V+16)>>5\\&H=\sum_{x'=0}^{3}(x'+1)\cdot(P[4+x',-1]-p[2-x',-1])\\&V=\sum_{y'=0}^{3}(y'+1)\cdot(P[-1,4+y']-P[-1,2-y'])\end{aligned} \tag{2－47}$$

2.4 帧间预测

帧间预测主要是利用连续图像序列之间的相关性，通过运动补偿预测编码方法来消除视频图像的时间冗余。H.264 除了具有以前一些标准中的基本的 P 帧、B 帧预测方法外，还增加了许多新的功能：采用不同大小的预测块进行运动估计；采用 1/4(甚至 1/8)像素精度的运动补偿算法；采用多参考帧进行帧间预测编码。

2.4.1 不同块大小的帧间预测

H.264 标准中对帧间预测[18]时的每个 16×16 宏块又可分为 8×16，16×8，8×8，8×4，4×8，4×4 等更小的块进行变换编码(图 2.7 所示)。色度块大小为亮度块的 1/4。由于采用不同大小的块进行帧间预测，使得运动估计模型更接近物体的实际运动，因此运动补偿的精度更高。

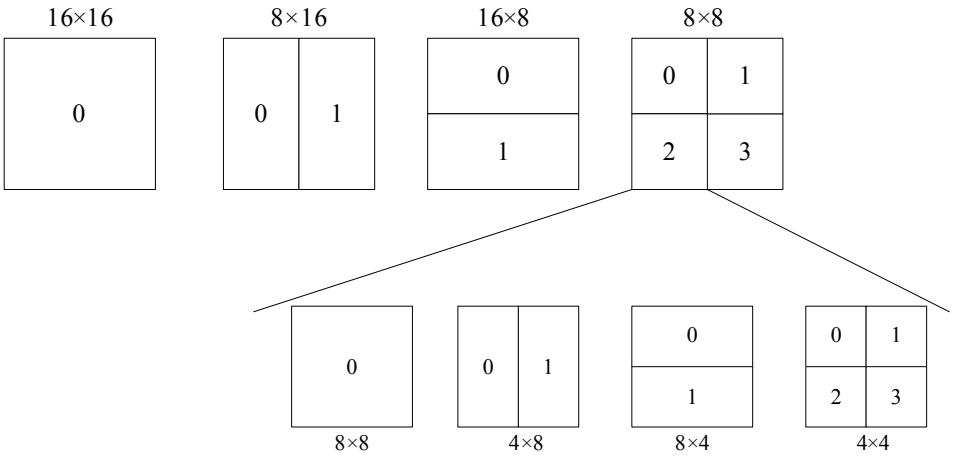


图 2.7 帧间编码的块划分

2.4.2 高精度的运动估计算法

H.264 的运动估计可以达到 1/4，1/8 像素，它主要是通过插值来取得分数像素值。图 2.8 给出了整数像素点和分数像素点的分布示意图，图中大写字母 $A \sim U$ 为整数像素点位置，aa、bb、cc、dd、ee、ff、gg、hh 及 b、h、j、s、m 为半像素点位置，a、c、d、n、f、i、k、q、e、g、p 、r 为像素点位置。其中1/2 像素预测值由一个抽头值(1,-5,20,20,-5,1)的 6 抽头滤波器得到，1/4 像素预测值由相邻的整数像素和半像素值平均得到。

具体采样插值方法如下：

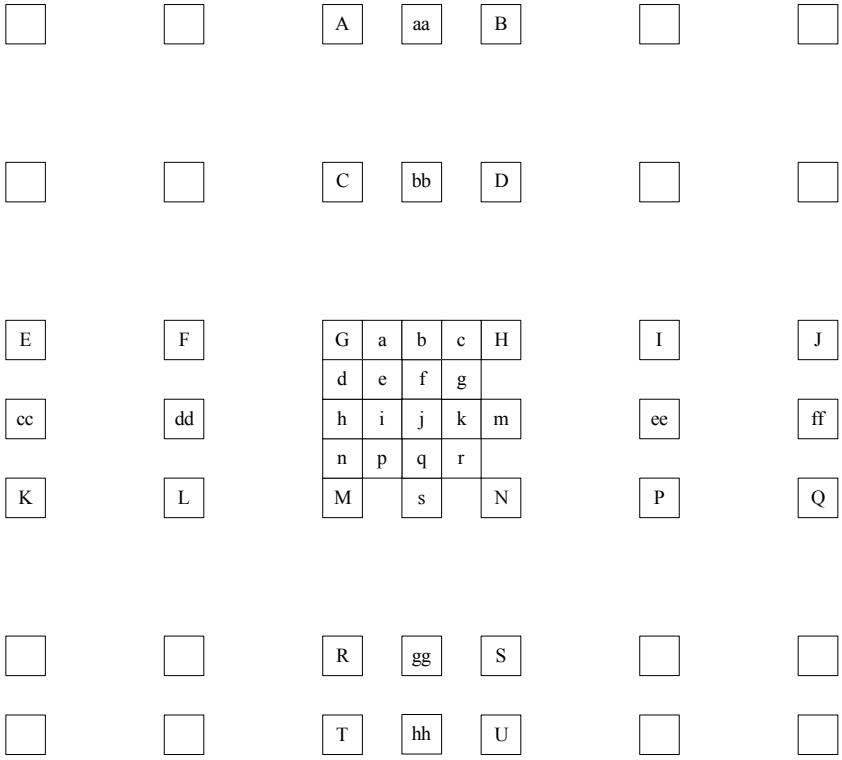


图 2.8 整数、分数像素点位置分布

1、1/4 像素亮度插值方法

半像素点 b 预测值：由水平方向相邻的 6 个整数像素点值利用 6 抽头滤波器得到一个中间临时值 b1，再由 b1 得到 b 的值。h 点则是由垂直方向的 6 个整数像素点值得到中间临时值 hl，再由 hl 得到 h 的值。

具体如下：

$$b_1 = E - 5 \times F + 20 \times G + 20 \times H - 5 \times I + J \qquad (2-48)$$

$$h_1 = A - 5 \times C + 20 \times G + 20 \times M - 5 \times R + T \qquad (2-49)$$

最终 b、h 点值由如下二式获得：

$$b = \max(0, \min(255, (b_1 + 16)/32)) \qquad (2-50)$$

$$h = \max(0, \min(255, (h_1 + 16)/32)) \qquad (2-51)$$

半像素点 j 预测值：首先由水平或垂直方向的 6 个相邻半像素点值由 6 抽头滤波器得到 j1 的值，再由 j1 值得到 j 的值。具体过程如下：

$$j_1 = cc - 5 \times dd + 20 \times h_1 + 20 \times m_1 - 5 \times ee + ff \qquad (2-52)$$

$$\text{或 } j_1 = aa - 5 \times dd + 20 \times b_1 + 20 \times s_1 - 5 \times gg + hh \qquad (2-53)$$

最终 j 点值由下式获得：

$$j = \max(0, \min(255, (j_1 + 512)/1024)) \quad (2-54)$$

半像素点 s、m 点的值同 b、h 类似，先得到中间值 s1、ml 之后再得到。其中 s 点值由公式(2-55)获得，m 与 s 类似可得。

$$s = \max(0, \min(255, (s_1 + 16)/32)) \quad (2-55)$$

四分之一像素点 a 的值为 $a = (G + b + 1)/2$，其它四分之一像素点 c、d、n、f、i、k、q 值分别由其水平或垂直方向上的相邻整像素或半像素点值按与 a 点类似的方法获取。而四分之一像素点 e 值为 $e = (b + h + 1)/2$，其它几个四分之一像素点 g、p、r 的值由其对角方向相邻的半像素点值按类似 e 点的方式获得。

2、色度系数插值方法

图 2.9 给出了色度整像素点 A、B、C、D 及其内部分数像素点 x 的相对位置信息，该色度分数像素点值由公式 得到：

$$\begin{aligned} X = &((8-\Delta x_c)\times(8-\Delta y_c)\times A + \Delta x_c \times (8-\Delta y_c)\times B \\ &+ (8-\Delta x_c)\times \Delta x_c \times C + \Delta x_c \times \Delta y_c \times D + 32)/64 \end{aligned} \quad (2-56)$$

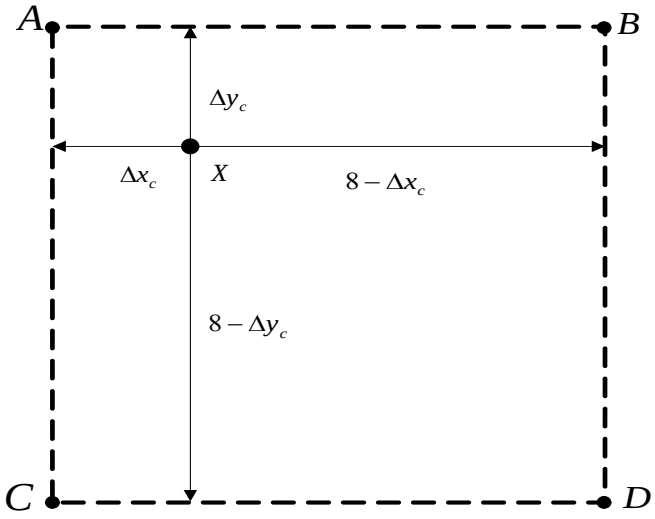


图 2.9 色度分像素差值

2.4.3 多参考帧模式

与 H.263 和 MPEG 标准不同，H.264 采用最多 5 个参考帧进行帧间预测。图像编码顺序不是基于时间的图像显示顺序，而是基于图像之间的依赖关系的。一个具体的编码顺序如图 2.10 所示。

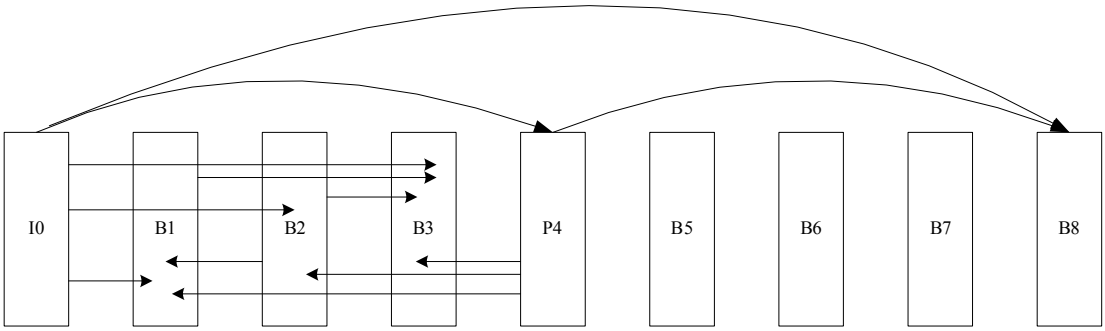


图 2.10 多参考帧编码示意图

图 2.10 中图像序列的编码顺序为 I0、P4、B2、B1、B3、P8、B6、B5、B7。由于 B2 依赖于 I0 和 P4，因此在 I0 和 P4 之后编码，B1 依赖于 I0、P4 和 B2，同理，其它图像依次编码都是在其所依赖的图像之后。解码器得到图像序列后，通过重排，重新按照 I0、B1、B2、B3、P4、B5、B6、B7、P8 的顺序进行显示。

P 帧和 B 帧图像的参考图像存储在两个参考帧序列中，分别存储前向预测参考帧和后向预测参考帧。由于每个参考帧序列中都包含不止一幅图像，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，还必须为增加的参考帧提供更多的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销。通过实际测试表明：通常超过 3 帧以上的参考帧对于提高编码效果没有更大的作用，因此在实时应用中可以根据视频序列的运动剧烈程度来选择适当的参考帧数目，对于运动比较平缓即运动矢量统计均值较小的序列可以适当减少参考帧数目，而对运动相对剧烈即运动矢量统计均值较大的序列可以适当增加参考帧数目。

2.5 整数变换与量化

为了进一步节省图像传输码率，需要对图像信号进行压缩，一般方法为去除图像信号中的相关性以及减小图像编码的动态范围，通常采用变化编码与量化技术。变换编码将图像时域信号转换成频域信号。在频域中，图像信号的大部分能量集中在低频区域，相对于时域信号，码率有较大的下降。H.264 对图像或预测残差采用了 4×4 整数离散变换技术，避免了以往标准中使用的通用

8×8 离散余弦变换、逆变换经常出现的失配问题。量化过程根据图像动态范围的大小来确定量化参数，即保留了图像中必要的细节，又可以减少码流。在图像编码中，变换编码和量化从原理上讲是两个独立的过程，但在 H.264 中，将两个过程的乘法合二为一，并进一步采用整数运算，减少了编解码的运算量，提高了图像压缩的实时性。这些措施对峰值信噪比(PSNR)的影响很小，一般低于 0.02dB，可以忽略不计。H.264 中整数变换及量化的具体过程如图 2.11 所示。其中，如果输入是色度块或帧内 16×16 预测模式的亮度块，则将宏块中的各 4×4 块的整数余弦变换的直流分量组合起来在进行 Hadmard 变换，进一步压缩码率。

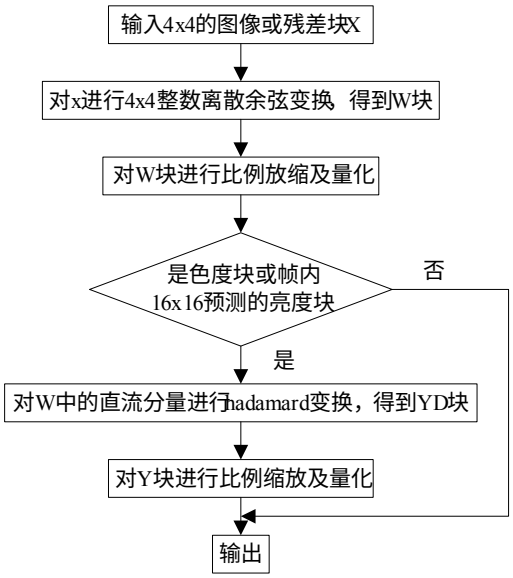


图 2.11 编码器中的变换编码及量化过程

2.5.1 整数变换

对实数的 DCT，由于在解码端的浮点运算精度问题，会造成解码后的数据的失配，进而引起漂移。H.264 较其他图像编码使用了更多的预测过程，甚至内部编码模式也依赖于空间预测。因此，H.264 对预测漂移是十分敏感的。为此，H.264 对 4×4DCT 进行了改造，采用整数 DCT 技术，可有效的减少计算量，同时不损失图像的准确度。式(2－57)给出了 4×4DCT。

$$Y = C_f X C_f \otimes E_f = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} [X] \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right)^T \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix} \quad (2\text{-}57)$$

其中，运算“ $\otimes$ ”表示对每个矩阵元素只进行一次乘法，同时它将被归纳到量化计算中。这样，$C_f X C_f^T$ 中只剩下整数的加法、减法和移位(乘以 2)运算。H.264 将 DCT 中$\otimes E_f$ 运算的乘法融合到后面的量化过程中，实际的 DCT 输出为：

$$W = CXC^T \quad (2-58)$$

如上所述，可以将式(2－57)的矩阵乘法运算改造成两次一维整数 DCT 变换，例如先对图像或残差块的每一行进行一维整数 DCT 变换，然后对经行变换的块的每一列再应用一维整数 DCT。而每一次的一维整数 DCT 可以采用蝶型快速算法，以节省计算时间，如图 2.12 所示。

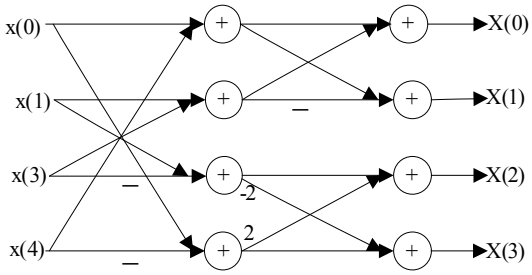


图 2.12 一维变换蝶型图

2.5.2 量化

量化过程在不降低视觉效果的前提下减少图像编码长度，减少视觉恢复中的不必要的信息。H.264 采用了标量量化技术，它将每一个图像样点都映射成较小的数值。一般标量量化器的原理为：

$$FQ = \text{round}\left(\frac{y}{Qstep}\right) \quad (2-59)$$

其中 y 为输入样本点编码，$Qstep$ 是量化步长，FQ 为 y 的量化值，

round() 为取整函数(其输出值为与输入实数最近的整数)。其相反过程，即反量化为：

$$y' = FQ \cdot QP \tag{2－60}$$

在量化和反量化过程中，量化步长 *Qstep* 决定了量化器的编码压缩率和图像精度。如果 *Qstep* 比较大，则量化值 *FQ* 的动态范围较小，其相应的编码长度较小，但反量化时损失较多的图像细节信息；如果 *Qstep* 比较小，则 *FQ* 的动态范围较大，相应的编码长度较大，但图像的细节损失较少。编码器根据图像值实际的动态范围自动改变 QP 的值，在编码长度和图像精度之间折衷，达到整体最佳效果。

在 H.264 中，量化步长 *Qstep* 共有 52 个值。见表 2.1。其中，QP 为量化参数，是量化步长的序号。当 QP 取最小值 0 时，代表最精细的量化，当 QP 取最大值 51 时，代表最粗糙的量化。QP 每增加 6，*Qstep* 增加一倍。应用时可以在这个较宽的量化步长范围内根据实际需要灵活选择。对于色度编码，一般使用与亮度编码同样的量化步长。为了避免在较高量化步长时出现的颜色量化人工效应，现在的 H.264 草案把色度的 QP 最大值限制在亮度 QP 的最大值的 80%范围内，最后的 H.264 草案规定，亮度 QP 的最大值是 51，而色度 QP 的最大值是 39。

表 2.1 H.264 中编解码器的量化步长

Qp	Qstep	Qp	Qstep	Qp	Qstep	Qp	Qstep
0	0.625	1	0.6875	2	0.8125	3	0.875
4	1	5	1.125	6	1.25	7	1.375
…	…	12	2.5	13	2.75	14	3.25
…	…	24	10	25	11	26	13
…	…	36	40	37	44	38	52
…	…	50	208	51	224		

具体的量化过程的运算为：

$$\left|Z_{ij}\right| = (\left|W_{ij}\right| \cdot MF + f) >> qbits \tag{2－61}$$

$$sign(Z_{ij}) = sign(W_{ij}) \tag{2－62}$$

$$MF = \frac{PF}{Qstep} 2^{qbits}$$

$$PF=\begin{cases} a^2 & (0,0),(2,0),(0,2)\text{或}\ (2,2) \\ b^2/4 & (1,1),(1,3),(3,1)\ \text{或}\ (3,3) \\ ab/2 & \text{其他情况} \end{cases} \tag{2－63}$$

$$qbits=15+floor(QP/6)$$

其中，W_{ij} 是矩阵 W 中的转换系数，PF 是矩阵 E_f 中的元素，根据样本点在图像中的位置 (i,j) 取值见式（2－63），“ >> ”为右移运算；sign()为符号函数；f 为偏移量，它的作用是改善恢复图像的视觉效果，例如，对帧内预测图像块 f 取 $2^{qbits}/3$，对帧间预测图像块 f 取 $2^{qbits}/6$。

2.5.3 DCT 直流系数的变换量化

编码过程中，H.264 根据要编码的残差数据类型使用三种变换[19,20]：除了 2.5.2 中对残差信号的整数变换外，如果当前处理的图像宏块是帧内 16×16 预测模式的亮度块或色度块，将其中各图像块的 DCT 变换系数矩阵 W 中的直流分量或直流系数 W_{00} 按对应图像块的顺序排列，组成新的矩阵 W_D，再对 W_D 进行 Hadamard(4×4 和 2×2)变换量化。

1、帧内 16×16 预测模式亮度块的直流系数变换量化

16×16 的图像宏块中有 4×4 个 4×4 图像亮度块，所以亮度块的 W_D 为 4×4 矩阵，其组成元素为各图像块 DCT 的直流系数 W_{00}，这些 W_{00} 在 W_D 中的排列顺序为对应图像块在宏块中的位置。

对亮度块 W_D 的 Hadamard 变换为：

$$Y_D=\left(\begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&-1&1\\1&-1&1&-1\end{bmatrix}W_D\begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&-1&1\\1&-1&1&-1\end{bmatrix}\right)/2 \tag{2－64}$$

其中，Y_D 是 Hadamard 变换结果。接着要对 Y_D 再进行量化输出：

$$\left|Z_{D(i,j)}\right|=(\left|Y_{D(i,j)}\right|\cdot MF_{(0,0)}+2f)>>(qbits+1) \tag{2－65}$$

其中，$MF_{(0,0)}$ 是位置为(0，0)的 MF 系数值。

2、色度块的直流系数变换量化

16×16 的图像宏块中包含图像色度 Cr 和 Cb 块各 2×2 个，所以色度 Cr 或 Cb 块的W_D2×2 矩阵，其组成元素为各图像块色度信号 DCT 的直流系数W_{00}，这些W_{00}在W_D中的排列顺序为对应图像在宏块中的位置。

对色度块W_D的 Hadamard 变换为：

$$Y_D = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} W_D \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2－66}$$

Y_D是 Hadamard 变换的结果，接着要对Y_D再进行量化输出同式(2－65)

2.6 熵编码原理

1．熵编码的基本原理

熵编码[21]是无损压缩编码方法,它生成的码流可以经过解码无失真的恢复出原始数据。熵编码是建立在随机过程的统计特性基础上的。

设信息 x 可发出得信息符号集合为$A = \{a_i \mid i = 1,2,\Lambda\ ,m\}$,并设 x 发出符号$a_i$的概率$p(a_i)$为,则定义符号$a_i$出现的自信息量为：$I(a_i) = -\log p(a_i)$

通常，上式中的对数取 2 为底,这时定义的信息量单位为“比特”(bit)。

如果各符号 a 出现是独立的，那么 x 发出一符号序列的概率等于各符号的概率之积，因而该序列出现的信息量等于相继出现的各符号的自信息量之和。这类信源称为无记忆信源。

对信源 x 的各符号的自信息量取统计平均，可得平均信息量：

$$H(X) = -\sum_{i=1} p(a_i)\log_2 p(a_i) \tag{2－67}$$

$H(X)$称为信息源 x 的熵，单位为 bit/符号。它通常也称为 x 的一阶熵，可以将它理解为信息源 x 发任意一个符号的平均信息量。有信息论的基本概念可以知道，一阶熵是无记忆信息源(在无失真编码时)所需码率的下界。

熵的大小与信源的概率模型有着密切的关系，各个符号出现的概率不同，信源的熵也不同。当信源中各事件是等概率分布时，熵具有极大的价值。信源的熵与其可能达到的最大值之间的差值反映了该信源所含有的冗余度。信源的冗余度越小，即每个符号所独立携带的信息量越大，那么传送相同的信息量所需要的序列长度就越短，符号位就越少。因此，数据压缩的一个基本途径是去除信源的符号之间的相关性，尽可能地使序列成为无记忆的，即前一符号的出

现不影响以后任何一个符号出现的概率。

2．CAVLC 编码的基本原理

在 H.264 的 CAVLC(基于上下文自适应的可变长编码)中，通过根据已编码句法元素的情况，动态调整编码中使用的码表，取得了极高的压缩比。

CAVLC 用于亮度和色度残差数据的编码。残差经过变换量化后的数据表现如下特性：4×4 块数据经过预测、变换、量化后，非零系数主要集中在低频部分，而高频部分系数大部分是零；量化后的数据经过锯齿(zig－zig)扫描，DC 系数附近的非零系数值较大，而高频位置上的非零系数值大部分是＋1 和－1；相邻的 4×4 块的非零系数的数目是相关的。CAVLC 充分利用残差经过整数变换、量化后数据的这些特性进行压缩，可进一步减少数据中的冗余信息，为 H.264 卓越的编码效率奠定了基础。

2.7 去块滤波

去块滤波系统在大幅度改善视频图象质量和压缩效率的同时,也引入了极大的计算复杂度，在解码器中，去块滤波的运算量大约占到了解码器计算总量的 1/3，使之成为实际应用的瓶颈。

1、H.264 标准中的去块滤波算法

标准中的滤波过程[22]是基于4×4块进行的,图2.13为图像中一个宏块(MB)的亮度，色度块以及滤波时用到的相邻的块，数字 0~23 表示宏块中的 4×4 块(0~15 为亮度块，16~23 为色度块)，字母 a~p 表示滤波时用到的相邻的块。一个宏块 MB 的基本滤波顺序是先从左到右对宏块的 4 条垂直块边界进行水平滤波，然后从上到下对宏块的 4 条水平块边界进行垂直滤波。

	e	f	g	h
a	0	1	2	3
b	4	5	6	7
c	8	9	10	11
d	12	13	14	15

	k	l
i	16	17
j	18	19

	o	p
m	20	21
n	22	23

图 2.13 MB 的亮度块和色度块

滤波过程主要分为两步，第一步是边界滤波强度 BS 的计算，第二步是根据不同的滤波强度对边界像素进行不同程度的滤波，下面分别来介绍。

2、边界滤波强度 BS 的计算

对于每一个 4×4 块的边界，边界强度 BS(Boundary Strength)将决定不同的滤波计算方法，使得对不同性质的边界滤波强度不同，这样就在平滑边界像素的同时最大程度的保证图像不会因此而变得模糊。不妨设 P、Q 分别表示两个相邻的 4×4 块，BS 的自适应设定如图 2.14 所示。

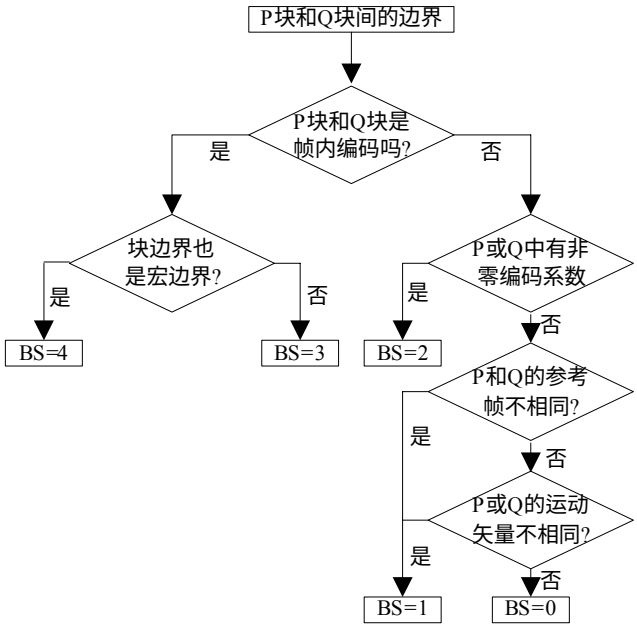


图 2.14 边界强度设定示意

3、滤波过程

滤波中一个关键的问题就是区分图像的真实边界和虚假边界，为此，标准中采用了两个限 α，β，其中 α 表示块间的边界门限，β 表示块内的边界门限，高于门限就判定为真实边界，此时不进行滤波。结合这个判定结果和 BS 的值对块边界进行滤波。图 2.15 为需要滤波的水平和垂直边界，其中 pi，qi(i=0，1，2，3)表示边界两侧的像素点。BS 小于 3 时，滤波系统使用了一个 4 抽头的线性滤波器。输入为 p0，p1，q0，q1，修改边缘相邻的 p0，q0 值。此外，

如果门限 β 判断出块内存在虚假边界，则用相同的滤波器调整 p1 和 q1 的值。BS=3，4 时，使用较强的滤波器，除了上述 4 个像素值外，还要对 p2，q2 进行修改，对不同位置的像素分别使用了 4 抽头和 5 抽头的滤波器。

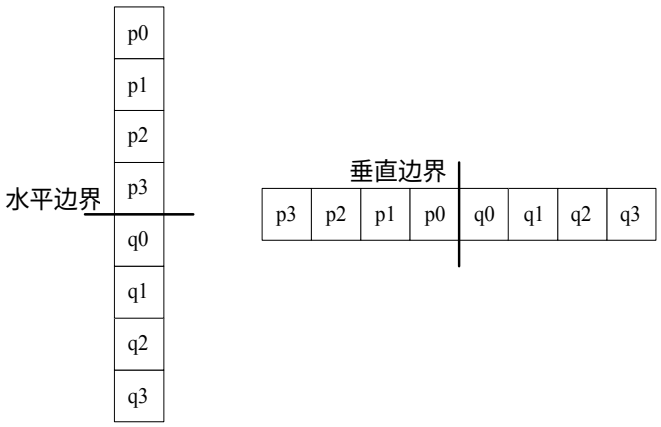


图 2.15 需要滤波的水平边界和垂直边界

第三章 H.264 帧内预测算法

3.1 引言

视频图像序列中存在多种冗余信息，主要为时域冗余和空域冗余。H.264 的帧内预测编码充分利用了图像的空间相关性，使用当前块的相邻块中已解码重建的像素做外推来实现对当前块残差的编码，以消除单帧图像内的空域冗余。H.264 支持三种帧内预测模式：用于 4×4 亮度块编码的 Intra_4×4,有 9 种预测模式，其中包括 8 种方向性预测模式和一种均值预测模式，由于块尺寸较小，适合于图像细节丰富的区域；用于 16×16 亮度宏块编码的 Intra_16×16,有 4 种预测模式，适合于图像中的平坦区域；用于 8×8 色度块编码的 Intra_8×8，也有 4 中预测模式，除预测模式编号不同外，与 Intra_16×16 相同。当使用 RDO[23,24,25](率失真优化)模型时，帧内预测的编码时间将成倍增加。因此，为了满足实时视频通信的要求，必须提高帧内预测编码速率。

减小帧内预测计算复杂度的方法可从两方面考虑：一方面是简化代价函数；另一方面是缩小预测模式选择的范围[26]。对于第一方面，在 SAD(Sum of Absolution Difference)代价函数模型下，可以通过图像亚采样等方法，以代价函数的局部计算为手段达到目的。但在 RDO 代价函数模型下，由于需要在完整编码一块后使用编码比特率完成计算，因此要在简化的同时保证编码效率，不是很容易做到。对于第二方面，可以利用当前块及其相邻像素的某些特征，在空域或变换域中进行预编码，预先排除某些可能性很小的预测模式，或提前终止可能性很小的模式计算，从而达到降低算法复杂度的目的。

3.2 代价函数模型

3.2.1 SAD 代价函数模型

在 H.264 的帧内编码中，使用 SA(T)D 代价函数模型进行模式选择的具体计算步骤如下：

(1) 计算 $SA(T)D_0$

H.264 发现将从大量实验结果中统计得到的经验值 $SA(T)D_0$ 作为偏移量加入编码过程，可以最小化编码完成后需要传送的 SA(T)D 值。

具体对 Intra_4×4 和 Intra_16×16 的模式选择之前计算而言，$SA(T)D_0$ 值分别为 $24\lambda(Q_p)$ 和 0。其中 $\lambda(Q_p)$ 是量化因子 Q_p 的指数函数近似，计算公式为：

$$\lambda(Q_p) = \lambda_{mode} = 0.85 \times 2^{(Q_p-12)/3} \tag{3-1}$$

(2) 计算宏块残差

当前块的原始块与预测块之间的残差值计算如下：

$$diff(i,j) = Orig(i,j) - \Pr ed(i,j) \tag{3-2}$$

其中，$Orig(i,j)$ 表示原始块对应像素值，$\Pr ed(i,j)$ 表示预测块对应像素值。

(3) Hadamard 变换

虽然在最后传输前需要对所有编码后数据进行交换，但是 H.264 为了追求最佳效果，在帧内模式选择的计算过程中先对 $diff(i,j)$ 进行一次二维 Hadamard 变换，得到每个模式的最终结果为：

$$SATD(diff) = (\sum_{i,j} |diffT(i,j)|)/2 \tag{3-3}$$

(4) $SA(T)D_{\min}$

对当前块每个预测模式重复第(3)步，选择其中 SATD 值最小的模式为当前块的最佳模式，并令该模式下的 SATD 值为 $SA(T)D_{\min}$。

3.2.2 RDO 代价函数模型

RDO 算法基本原理是：在可选的宏块类型中，遍历每种可用编码模式，计算出每种模式下的编码比特数和相应的重建图像失真度，再根据公式[27]：

$$\begin{aligned} &J(s,c,MODE \mid QP,\lambda_{MODE}) \\ &= SSD(s,c,MODE \mid QP) + \lambda_{MODE} \cdot R(s,c,MODE \mid QP) \end{aligned} \tag{3-4}$$

计算拉格朗日函数值 J，即率失真开销。经过比较，选取代价值最小的编码模式为宏块的最终编码模式。其中 QP 为宏块的量化参数，λ_{MODE} 表示拉格朗日乘数，MODE 为当前宏块可选的一种编码模式，对 I 帧而言，MODE 即为 Intra_16×16，Intra_4×4 或 Intra_8×8，s, c 分别表示原始图像和重建图像的像素值，$R(s,c,MODE \mid QP)$ 表示在特定 QP 和 MODE 下当前宏块的编码输出比特数，$SSD(s,c,MODE \mid QP)$ 表示图像失真度，计算公式如下：

$$\begin{aligned} SSD(s,c,MODE \mid QP) &= \sum_{x=1,y=1}^{16,16}(s_Y[x,y]-c_y[x,y,MODE \mid QP])^2 \\ &+ \sum_{x=1,y=1}^{8,8}(s_U[x,y]-c_U[x,y,MODE \mid QP])^2 + \sum_{x=1,y=1}^{8,8}(s_V[x,y]-c_V[x,y,MODE \mid QP])^2 \end{aligned} \tag{3-5}$$

在 RDO 代价模型下，比特率 R 值被精确的计算出来，使得其模式选择的准确度大大提高，但 R 值的计算复杂度很高；而基于 SAD 代价函数模型的模式选择由于避免了 R 值的计算，速度大大提高，但模式选择的准确度有所下降。统计表明，SAD 方法的计算复杂度平均为 RDO 方法的 7%，但 PSNR 平均下降 0.47dB[24]。

所以，本论文中讨论的 H.264 帧内预测算法都是在 RDO 代价函数模型下进行的改进和实现。

3.3 H.264 全搜索的帧内预测算法及其复杂度的分析

本文采用 H.264 参考软件 JM90 作为测试模型。全搜索算法(FS, Full Search)检查每一种可能的帧内预测模式来选择最佳模式。FS 帧内模式选择主要步骤如下：

1) 对于 4×4 帧内预测模式 mode4×4 建立相应的帧内预测块；
2) 计算预测 4×4 块和原始 4×4 块之间的绝对差 SAD4×4，以及相应的编码比特率；
3) 计算该模式的率失真开销 Rdcost4×4；
4) 重复以上 a)~c)步，遍历所有的 9 种 4×4 帧内预测模式；

5) 选取具有最小率失真开销 Rdcost4×4 的模式作为最佳 4×4 帧内预测模式；
6) 对宏块中 16 个 4×4 块重复以上 1)~5)步，获得每一个 4×4 块的最佳编码模式和相应的 Rdcost4×4，求和进而获得该宏块的率失真开销 Rdcost；
7) 按类似的方法遍历 4 种 16×16 宏块的帧内预测模式并计算相应的宏块率失真开销 Rdcost16×16，选择宏块率失真开销 Rdcost16×16 最小的模式为最佳 16×16 宏块的帧内预测模式；
8) 根据 6)和 7)中最小的宏块率失真开销,判断亮度宏块采用 4×4 或 16×16 帧内预测模式；
9) 对每一种 8×8 色度宏块的帧内预测模式(两个色度宏块使用相同的模式)计算相应的率失真开销 Rdcost8×8，并重复以上 1)~9)步，获得相应的宏块组合率失真开销 RdcostMB，选择最小的宏块组合率失真开销 RdcostMB 作为该宏块组合的最佳帧内预测模式；

由此完成了一个宏块组合的帧内预测模式选择。显然对一个宏块组合，共有 $M_8 \times (M_4 \times 9 + M_{16})$ (其中 M_8=4, M_4=16, M_{16}=4)种不同的组合，因此共需完成 592 次 RDO 计算，才能获得最优的帧内预测模式编码。并且在最优模式选择的过程中，重复计算了亮度宏块在帧内预测各个模式下的率失真开销。因此，过多的候选模式数和巨大的计算开销使模式选择成为帧内编码的瓶颈所在。

3.4 快速算法的研究现状

目前，不少专家对帧内模式快速选择算法进行研究，他们大部分是根据相邻块与当前块的联系，以相邻块中的信息对当前块进行估计。也有一部分是根据当前块自身的信息来简化计算。总的说来，实现帧内预测模式选择快速算法的途径主要从以下两个方面着手：(1)代价函数的简化；(2)缩小预测模式选择的范围。本文主要是用第二种方法来实现对帧内编码模式进行快速选择，以达到对整个编码器的优化。

研究缩小编码模式搜索范围的主要途径是通过其它一些低计算复杂度的方法，直接排除一些不可能的编码模式，从而缩小模式选择的范围。该类方法几种现有算法介绍如下：

文献[13]最早提出对帧内预测模式的选择进行优化：由于物体边界通常是连续的，沿边缘方向的像素往往具有相似的值。因此，文中采用 Sobel 算子计算当前待编码块的每个像素点的边缘方向矢量，然后利用边缘方向矢量求得每一个块的边缘方向直方图，最后通过该直方图的分布特点，选择出可能性较大的几个模式。这样就可以减少需要计算的帧内预测模式数，提高编码速率。实验结果显示：该方法在提高 20%-30%编码速率时，增加了 2%的码率；提高 50%-60%时，增加了 5%的码率。这是因为利用这种方法时，需要对当前块中所有点的边缘矢量强度进行计算，并统计每一个点的边缘矢量方向所处的预测空间，可见计算复杂度仍然很高。

文献[27]对预测模式进行变换，然后计算变换系数的绝对和得到 SATD(Sum of Absolute Transformed Difference)，由于 SATD 与率失真(RD)性能具有很强的相关性，用 SATD 作为判断准则，预先排除掉一些可能性很小的 Intra4×4 预测模式，缩小模式选择范围，从而降低计算复杂度。SATD 虽然可以作为选择模式的准则，但是并不是说 SATD 最小的预测模式就是 RD 性能最好的模式，该算法的关键是阈值的选取。

文献[28]提出一种基于二维直方图的快速帧内预测判决算法。该算法利用二维直方图的特点，结合双阈值的方法，在进行帧内预测之前先对宏块预判，从两种帧内预测模式中选择一种，从而减小计算的复杂度，提高了编码速率。实验结果表明，该方法在码率少许增加的情况下，编码速度提高 16.26%，PSNR 基本保持不变。虽然该算法编码效率提高不是很多，但其与其他帧内预测模式选择快速算法结合使用可以取得较好效果。

文献[29]针对 H.264 帧内预测中存在的边缘像素预测不精确问题，提出了一种基于递推模式的帧内预测算法。首先，利用相邻像素作为参考进行预测，提高了预测的准确性；其次，利用 4×4 块在空域和频域上的特性，分层次对候选模式进行筛选；最后，充分利用 4×4 块与 16×16 宏块之间的关系，快速确定 16×16 宏块的预测模式。实验结果表明这种基于递推模式的快速算法在很大程度上降低帧内编码的复杂度。

3.5 本章小结

本章首先介绍了 H.264 帧内预测编码技术和当前广泛应用的两类代价函数

模型的相关概念，并给出了测试模型 JM90 中采用的全搜索帧内预测模式选择算法步骤；接着本文对全搜索算法计算复杂度进行了详细的分析，并指出其缺点和改进方向；最后介绍了几种现有的快速帧内预测算法。

从几种现有快速算法的介绍当中可以看出，目前对帧内预测算法的研究和改进大都集中在提高某一种具体宏块类型的编码速度上，在预测前先判断宏块采用帧内 Intra_4×4 预测还是 Intra_16×16 预测的方法较少。在下一章将根据宏块内部相邻像素差值的特点提出一种帧内宏块预判断算法。

第四章 本文提出的基于相邻像素差的帧内宏块类型预判算法

4.1 引言

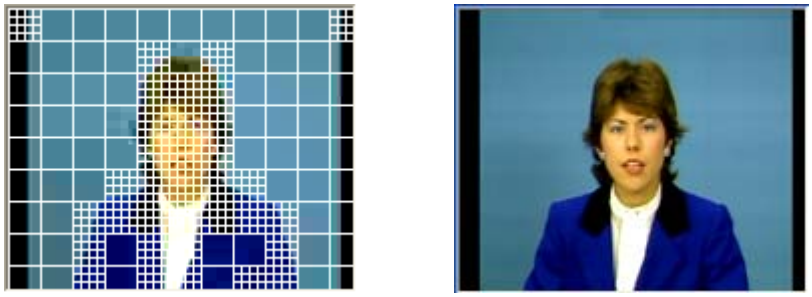
由第三章 3.3 节的分析可知，帧内预测作为 H.264 视频编码过程的一部分，采用了多种预测模式，对于一个宏块，共有 $M_8 \times (M_4 \times 9 + M_{16})$(其中 M_8=4, M_4=16, M_{16}=4)种不同的组合，因此共需完成 592 次 RDO 计算才能获得最优的帧内预测模式编码。并且在最优模式选择的过程中，重复计算了亮度宏块在帧内预测各个模式下的率失真开销。本文在充分考虑图像宏块特征及 H.264 帧内预测算法的设计初衷的基础上，提出一种基于宏块内部相邻像素差值的帧内宏块类型预判算法。该方法对帧内编码进行了快速算法优化，显著地降低了 H.264 编码算法的复杂度。

4.2 帧内预测宏块类型的选取

根据前面帧内预测的介绍，我们知道帧内预测模式中亮度宏块有两种预测类型：Intra_4×4 模式和 Intra_16×16 模式。Intra_16×16 模式是以整个宏块为处理对象，适合用于比较平坦的宏块；而 Intra_4×4 模式则把一个宏块分为 16 个子块，再对子块分别处理，适合用于细节较多的宏块。图 4.1 至图 4.3 分别显示了在 H.264 测试模型 JM90 中计算全 I 帧的 QCIF 序列“ Claire ”、“ Foreman ”和“ Mobile ”的宏块类型示意图和原视频帧，覆盖在图中的白框代表帧内预测中选择的不同的宏块类型。

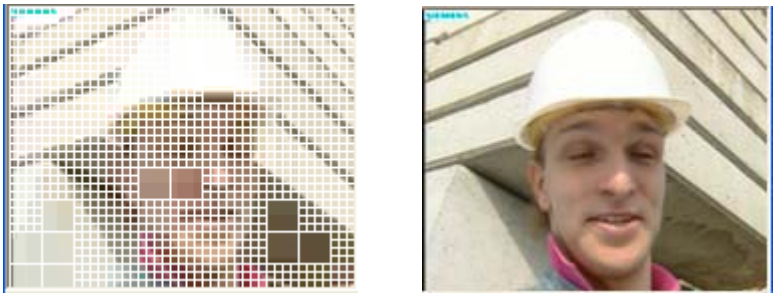
由这些图可以看出：对于纹理平坦的序列“ Claire ”，在变化平缓的背景区域都选用 Intra_16×16 模式，而在脸部、颈部、衣服边缘等细节丰富的区域都选用 Intra_4×4 模式；对于纹理较多的序列“ Foreman ”，由于背景纹理细节较丰富并且人物细节较多，所以多数选择了 Intra_4×4 模式而只有少数平坦区域选择 Intra_16×16 模式；对于纹理细节很多的序列“ Mobile ”，则绝大多数多数

选用 Intra_4×4 模式，只有挂历中白色区域的一个宏块选择了 Intra_16×16 模式。



(a)宏块类型选择　　(b) 原视频帧

图 4.1 Claire 宏块类型选择示意图



(a)宏块类型选择　　(b) 原视频帧

图 4.2 Foreman 宏块类型选择示意图

(a) 宏块类型选择　　(b) 原视频帧

图 4.3 Mobile 宏块类型选择示意图

因此，可以根据图像细节的丰富程度先对宏块进行预判，对于比较平坦的宏块，采用 Intra_16×16 模式；对于细节较多的宏块，采用 Intra_4×4 模式；对于宏块特征不明显，不好判出细节丰富程度的宏块，则对两种预测模式都进行计算，严格保证编码的质量。下面我们将寻找代表宏块细节丰富程度的宏块特征。

4.3 利用相邻像素差值特点提取宏块特征

帧内预测的全搜索算法是一种遍历所有可能的模式然后选取最优模式的方法，该算法实际上并没有用到宏块本身的特征，也没有考虑到 Intra_4×4 模式和 Intra_16×16 模式的设计初衷。所以本文设想如果能够预先判断一个宏块的图像细节是否丰富，就可以提前知道该宏块是适合用 Intra_4×4 模式预测还是 Intra_16×16 模式预测，从而对于一个宏块只要进行一种方式的搜索即可。基于上述观点，就可以利用宏块自身的特征作为帧内预测模式决策的方法，帧内预测模式的决策问题就转化为两个问题：(1)提取宏块的何种特征来表征它的细节丰富程度；(2)如何利用该特征进行计算，来得到宏块是平坦还是细节丰富的定论。

本文选择图像相邻像素差值表征宏块细节丰富程度的特征。像素值是每个像素点的灰度级值。对于 k 比特的图像，一幅图像灰度级范围是[0，2^k]。其中 0 为黑，2^k 在灰度级中为白，所有中间值是从黑到白的各种灰色调[30]。对于细节丰富的图像，像素点之间的灰度级值变化就比较剧烈，而对于比较平坦的图像，像素点之间的灰度级值变化就比较缓慢。显然，图像细节丰富程度与像素值之间的变化有一定的相关性。一幅图像相邻像素差值较大的数量越多则该图像细节就越多，反之，相邻像素差值较小的数量越多则该图像细节就少。

本文通过统计确定了表征图像细节多少的三个阈值 T1、T2、N，由这三个阈值来提前决定最佳预测块类型。理论上讲，一个像素和周围像素点像素差值有 8 个，但为了简化计算，本文只考虑水平和竖直方向的差值，这对判决结果几乎没影响。本算法的设计思想是：对于一个宏块，如果相邻像素差值大于 T1 的差值个数大于 N 则认为该宏块细节较多，采用 Intra_4×4 预测模式；如

果相邻像素差值小于 T2 的个数大于 N 则认为宏块平坦，采用 Intra_16×16 预测模式；不满足以上两种情况时则采用 H.264 的全搜索算法。这样做对于特征明显的宏块直接选择一种预测模式类型进行预测，减小算法复杂度；对于特征不明显的宏块采用标准的全搜索方法进行预测，严格控制了码率的增加。

图 4.4 给出了一个宏块图像分布图，一个宏块共 256 个像素。按照像素的水平和竖直方向差值来考虑，一个宏块中像素差值个数共为 15×16×2=480 个。例如图 4.4 中的像素 f，只计算它与像素 g 和 j 的差值 SUB，以此类推，计算出所有像素之间水平和竖直方向的差值(为了避免差值的重复计算，水平方向从左至右计算竖直方向从上到下计算，边缘行和边缘列单独计算)，然后根据差值大于 T1 的差值个数和差值小于 T2 的差值个数来判断该宏块采用哪一种预测类型。具体的算法流程见图 4.5。

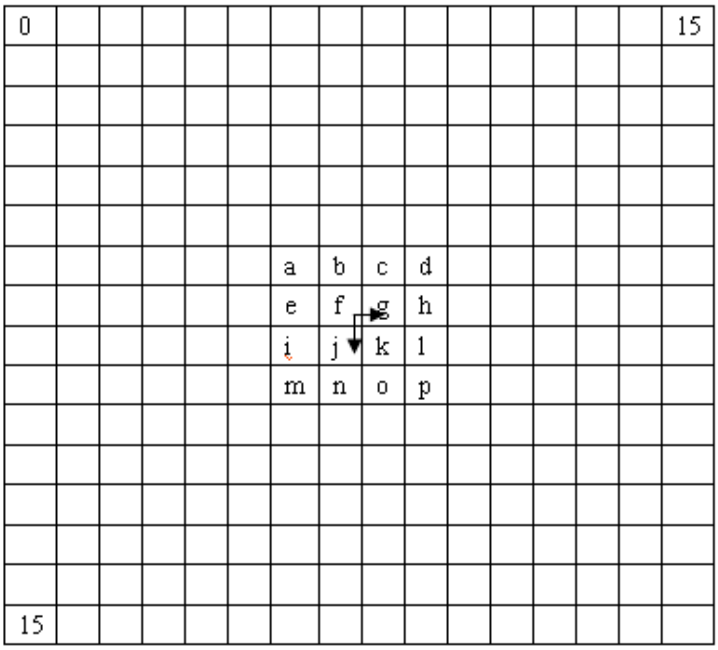


图 4.4 一个宏块图像分布

4.4 基于相邻像素差的宏块类型预判断算法

判断一个宏块选择 4×4 预测模式还是 16×16 预测模式，关键是阈值 T1、T2、N 的确定。本文通过一段嵌在 JM90 模型里的程序来进行统计试验。先给

定 T1、T2 的值，然后统计出采用 4×4 预测的宏块中相邻像素差值大于 T1 的个数 SUM1 和采用 16×16 预测的宏块中相邻像素差值小于 T2 的个数 SUM2。给定不同的 T1、T2 值对几个序列进行大量的实验，统计得出，当阈值 T1=4，T2=2，N=260 时预测效果较好。

通过以上几节的分析，下面给出该预判算法的主要过程：

1） 计算当前宏块相邻像素差值 SUB(水平方向从左至右计算竖直方向从上到下计算，边缘行和边缘列单独计算)；

2） 令 SUM1 和 SUM2 初值为 0。若 SUB>T_1，则 SUM1+1，若 SUB<T_2，则 SUM2+1，否则判断是否为最后一个需计算的像素差值；

3） 如果不是最后一个需计算的像素差值，则跳到 1)继续计算；

4） 如果 SUM1>N，则该宏块类型为 Intra_4×4，采用 Intra_4×4 模式对当前宏块进行预测，跳到 7)；

5） 如果 SUM2>N，则该宏块类型为 Intra_16×16，采用 Intra_16×16 模式对当前宏块进行预测，跳到 7)；

6） 若不满足 3)和 4)两种情况，则采用全搜索算法对当前宏块进行帧内预测，跳到 7)；

7） 根据 4)或 5)或 6)判断出的宏块类型对当前宏块进行预测，找出最佳的帧内预测模式；

8） 判断一帧是否结束，未结束则返回 1)处理下一宏块。

整个算法的流程如图 4.5 所示：

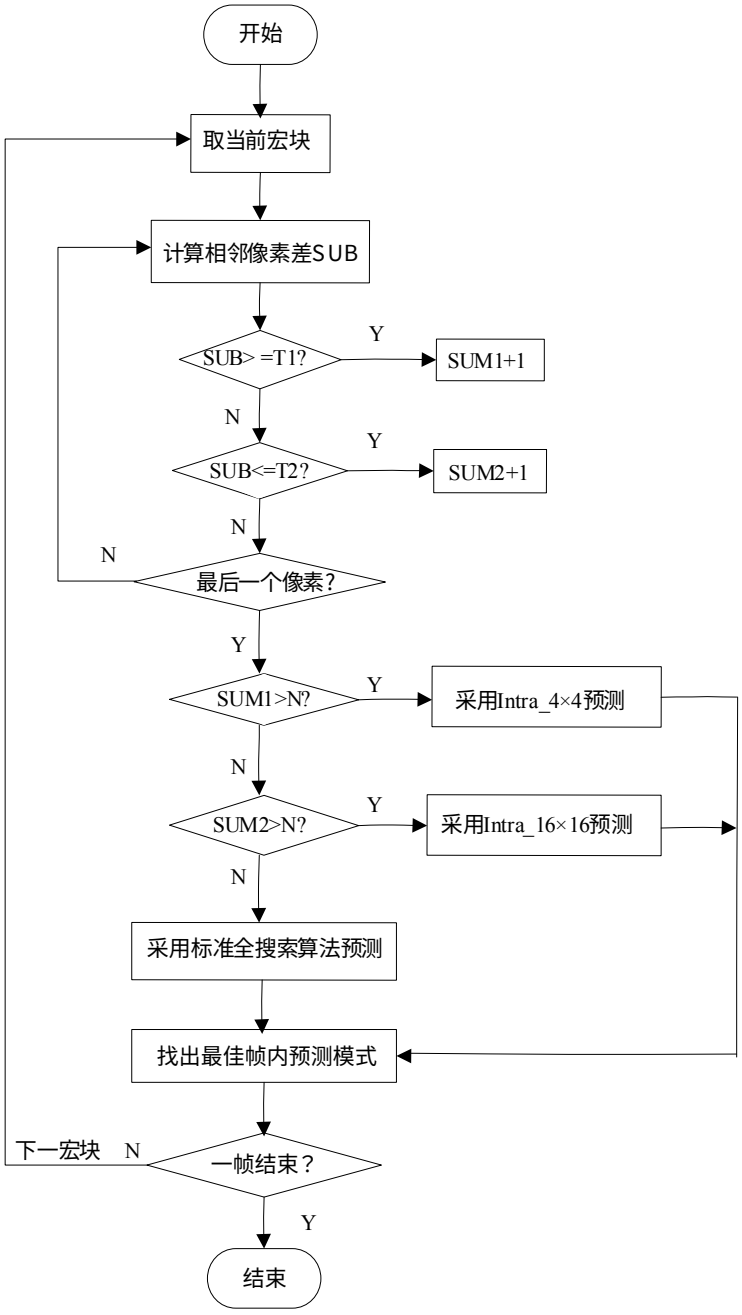


图 4.5 帧内宏块类型预判算法流程图

4.5 实验结果与分析

将提出的算法应用到 JM90 测试模型中并在 VC++6.0 平台上进行实验，实

验平台的配置为：CPU 为 P1.7G，内存为 512M 的计算机。实验中为了减少 XP 多任务的影响，都是在关闭其他应用程序的情况下进行测试得出的结果。实验参数设置如下：

1） 允许率失真优化 RDO；
2） 允许 hardarmard 变换；
3） 熵编码采用 CABAC；
4） Transform8x8Mode 被设置为 0；
5） 编码帧数为 100 帧；
6） 全 I 帧编码。

实验取 6 个具有代表性的 QCIF(176×144)测试序列进行预测。下面是对这 6 个测试序列的简单介绍：

1） Foreman 序列：纹理复杂度一般，运动较剧烈，画面人物和镜头均运动，并涉及到场景切换；
2） Mobile 序列：纹理复杂度极高运动形式丰富，画面有多个运动物体，但各运动物体运动方向规则且平缓。该序列纹理十分丰富，细节较多；
3） Claire 序列：背景静止且平坦，人物运动缓慢变化，该序列细节较少；
4） News 序列：细节较多，活动性低，只有背景人物缓慢的运动；
5） Grandma 序列：人物细节较多，运动较小，左边背景平坦，右边背景细节多；
6） Coastguard 序列：纹理较复杂，细节多，背景和物体都在运动，速度一般。

一、本文预判算法与 JM90 算法的实验数据比较

表 4.1、表 4.2、表 4.3 分别给出 QP=28，32，36 时提出的预判算法相对于JM90的全搜索算法在峰值信噪比(PSNR)、编码时间(Time)和编码比特率(Bits)三个方面的性能比较结果。其中 PSNR 有三个分量：PSNRY(亮度峰值信噪比)、PSNRU(色差 U 峰值信噪比)、PSNRV(色差 V 峰值信噪比)。表中ΔTime、ΔBits、ΔPSNRY、ΔPSNRU、ΔPSNRV 分别计算如下：

$$\Delta Time = \frac{T_{本文} - T_{JM}}{T_{JM}} \times 100\% \qquad (4-1)$$

$$\Delta Bits = \frac{Bits_{本文} - Bits_{JM}}{Bits_{JM}} \times 100\% \qquad (4-2)$$

$$\Delta PSNRY = PSNRY_{本文} - PSNRY_{JM} \quad (4-3)$$

$$\Delta PSNRU = PSNRU_{本文} - PSNRU_{JM} \quad (4-4)$$

$$\Delta PSNRV = PSNRV_{本文} - PSNRV_{JM} \quad (4-5)$$

表 4.1 QP=28 时本文预判算法与全搜索算法结果比较

测试序列	Foreman	Mobile	Claire	News	Grandma	Coastguard
Δtime(%)	-21.16	-17.20	-33.72	-34.80	-25.57	-17.04
ΔPSNRY(dB)	-0.04	-0.07	-0.02	0	-0.01	-0.04
ΔPSNRU(dB)	0	0	+0.01	0	0	0
ΔPSNRV(dB)	0	0	0	0	-0.01	0
ΔBits(%)	+0.33	+0.45	+0.27	+0.56	+0.3	+0.37

表 4.2 QP=32 时本文预判算法与全搜索算法结果比较

测试序列	Foreman	Mobile	Claire	News	Grandma	Coastguard
Δtime(%)	-19.35	-17.42	-31.33	-33.22	-23.87	-15.62
ΔPSNRY(dB)	-0.03	-0.02	-0.08	0	-0.03	-0.03
ΔPSNRU(dB)	0	-0.01	+0.01	0	0	0
ΔPSNRV(dB)	0	0	0	0	+0.01	0
ΔBits(%)	+0.29	+0.31	+0.19	+0.41	+0.25	+0.28

表 4.3 QP=36 时本文预判算法与全搜索算法结果比较

测试序列	Foreman	Mobile	Claire	News	Grandma	Coastguard
Δtime(%)	-18.58	-18.16	-32.81	-32.25	-22.24	-13.32
ΔPSNRY(dB)	-0.05	-0.03	-0.13	-0.06	-0.04	-0.05
ΔPSNRU(dB)	-0.01	0	0	0	+0.01	0
ΔPSNRV(dB)	0	0	0	0	0	0
ΔBits(%)	+0.18	+0.27	+0.11	+0.23	+0.16	+0.21

以上表中数据采用大量测试取平均值的数据处理方法。从表 4.1~表 4.3 的试验结果中，我们可以看出，提出的预判算法和 JM90 的全搜索算法相比，编

码时间有很明显的减少，平均减少可达约 23.47%；产生的码率相对于 JM90 略有增加，但增加的范围很小。PSNRY 有很小的变化，由于本算法只对亮度值进行判断，所以色度值基本保持不变，即 PSNRU 和 PSNRV 基本上没有变化。所以提出的算法对测试序列的影响非常小，对于人视觉系统来说几乎是不影响图像质量的。

二、本文预判算法与 JM90 算法的主观质量比较

图 4.6 给出了序列“ Foreman ”的标准全搜索算法和提出预判算法的主观质量的比较图。从图中我们几乎看不出什么差别，也就是说提出的算法不影响图像的主观质量。其他结果也类似。

JM90 全搜索算法　　预判算法

图 4.6 Foreman 全搜索算法和预判算法的主观质量比较图

三、本文预判算法与 JM90 算法的比特率失真比较

图 4.7、图 4.8、图 4.9 给出了三个具有不同细节复杂度的代表序列 Foreman、Mobile、Claire 在不同码率下比特率失真对比结果，另外三个序列也具有类似的结果。

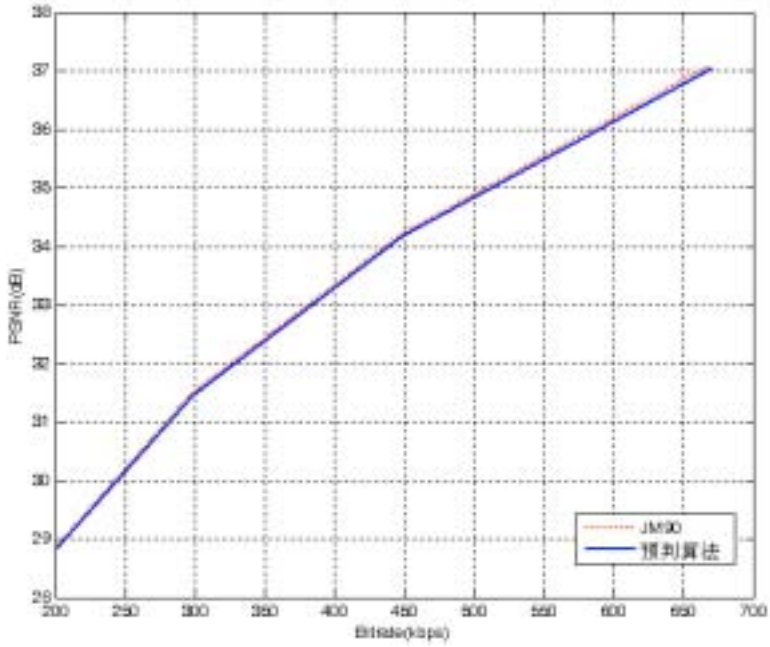


图 4.7 Foreman 比特率失真对比结果

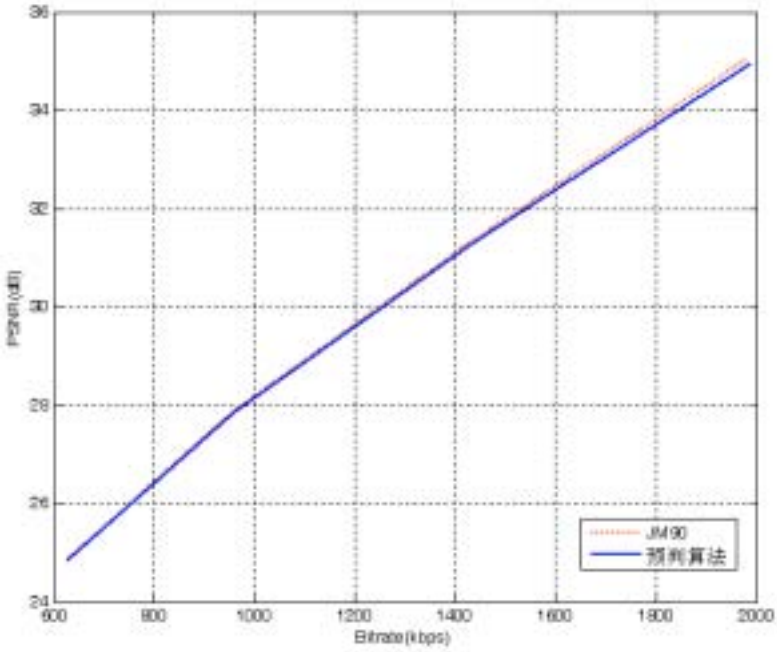


图 4.8 Mobile 比特率失真对比结果

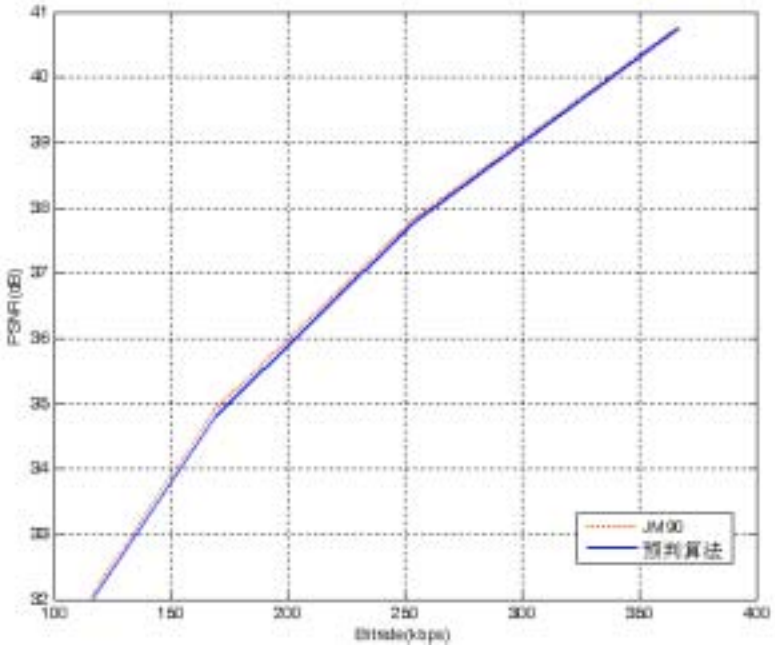


图 4.9 Claire 比特率失真对比结果

从图 4.7、图 4.8、图 4.9 中三种不同测试序列的率失真曲线可以看出，采用预判算法和采用 JM90 标准算法相比，在相同码流条件下产生的图像质量相差很小，几乎无法感知。

4.6 本章小结

本章针对 H.264 帧内预测全搜索算法宏块类型选择的特点，通过考虑宏块本身的特征和标准对平坦的宏块采用 Intra_16×16，细节较多的宏块采用 Intra_4×4 方式的思路，在大量统计观察和实验论证的基础上，提出了一种基于宏块内部相邻像素差的帧内预测类型判决算法。实验表明，利用宏块的相邻像素差特征可以有效地对帧内预测模式进行预判，并且在保证图像质量基本不变、码率变化很小的情况下，编码时间节省约 23.47%。

同时，该算法是一种与其它改进的帧内预测算法思路不同的方法，且该算法几乎无损，没有增加额外的计算复杂度，所以可以将其与其它的快速算法结合使用，更大的提高编码速度。

在下一章中，将继续研究在具体的帧内宏块类型下的预测模式快速选择算法。在分析基于边缘方向直方图的 Pan 算法基础上，提出一种改进的 Pan 算法。

第五章 改进的基于边缘方向直方图的 Pan 算法

在第四章中，通过在帧内预测中采用宏块类型预判算法，我们在 H.264 编码器的编码质量几乎不受任何影响的条件下，节省了约 23.47%的编码时间。但是这距离 H.264 的实时应用还是有一定的距离的。提出的宏块类型预判算法可以和其他任何帧内宏块内部的预测模式快速选择算法相结合，从而更大幅度地节省帧内预测编码时间。在本章中，将继续讨论在具体的帧内宏块类型下的预测模式快速选择算法，在分析基于边缘方向直方图的 Pan 算法基础上，提出一种改进的 Pan 算法。

5.1 基于边缘方向直方图的 Pan 算法

5.1.1 Pan 算法的思想

在 H.264 的帧内预测模式中，大部分的预测模式都是基于方向的预测模式，特别是 4×4 亮度块的 9 中预测模式，其中 8 种预测是方向预测。在编码时，若选择了某一方向的预测模式，也就表示该块的纹理具有所选预测模式对应的方向。基于这样的理论，Feng Pan 等[13,31]提出了基于边缘方向直方图的快速帧内预测模式选择算法(Pan 算法)，其基本思想是：通过观察发现在局部边缘方向上的像素点通常具有相似的值，组成块的各像素都与 8 种方向性模式相关。如果能找到相关性最强的那个方向模式，并使用其进行预测编码，就可以达到最有效果。

Pan 算法首先用 Soble 算子计算每个像素点的边缘方向矢量，然后利用边缘方向矢量求得每一个块的边缘方向直方图，最后通过该直方图的分布特点，选择出可能性较大的几个模式。Pan 算法中定义像素点 $p_{i,j}$ 的边缘矢量 $\vec{D}_{i,j} = \{dx_{i,j}, dy_{i,j}\}$ 为：

$$\begin{aligned} dx_{i,j} &= p_{i-1,j+1} + 2\cdot p_{i,j+1} + p_{i+1,j+1} - p_{i-1,j-1} - 2\cdot p_{i,j-1} - p_{i+1,j-1} \\ dy_{i,j} &= p_{i+1,j-1} + 2\cdot p_{i+1,j} + p_{i+1,j+1} - p_{i-1,j-1} - 2\cdot p_{i-1,j} - p_{i-1,j+1} \end{aligned} \tag{5-1}$$

其中 $dx_{i,j}$ 和 $dy_{i,j}$ 分别表示垂直和水平方向的分量； $p_{i,j+1}$、 $p_{i,j-1}$ 分别为像素点 $p_{i,j}$ 上方和下方相邻像素， $p_{i-1,j+1}$、 $p_{i+1,j+1}$、 $p_{i-1,j-1}$、 $p_{i+1,j-1}$ 分别为像素点 $p_{i,j}$ 左下方、右下方、左上方、右上方的相邻像素； $p_{i+1,j}$、 $p_{i-1,j}$ 分别为像素点 $p_{i,j}$ 右方和左方相邻像素， $p_{i+1,j-1}$、 $p_{i+1,j+1}$、 $p_{i-1,j-1}$、 $p_{i-1,j+1}$ 分别为像素点 $p_{i,j}$ 右上方、右下方、左上方、左下方的相邻像素。边缘矢量的幅值定义如下：

$$Amp(\vec{D}_{i,j}) = \left|dx_{i,j}\right| + \left|dy_{i,j}\right| \tag{5-2}$$

边缘方向角定义如下：

$$Ang(\vec{D}_{i,j}) = \frac{180^{\circ}}{\pi} \times \arctan\left(\frac{dy_{i,j}}{dx_{i,j}}\right), \left|Ang\left(\vec{D}_{i,j}\right)\right| < 90^{\circ} \tag{5-3}$$

对于 Intra_4×4 预测模式，为了确定块中是否包含边缘且该边缘是否剧烈，将 4×4 块中所有相似边缘方向的像素幅度累加，得到对应的边缘方向直方图。令具有最大幅度的直方图单元所对应的预测模式为主要预测模式。另外，由于 H.264 帧内编码实际的 RDO 计算是基于重建图像，而边缘方向直方图的计算因为无法得到重建帧而使用的是原始的无损图像，通过观察发现实际编码中的最佳 RDO 模式如果不是主要预测模式，就是其相邻两模式之一。因此，Pan 算法中也将主要模式的两个相邻模式选为候选模式。最后，为了保证快速算法对平滑方向性不明确的 4×4 块模式预测的准确性，Pan 算法将 DC 模式也作为候选模式。从而，对 4×4 亮度块，只计算了 4 种预测模式的 RDO 而不是 9 种。

对于 Intra_16×16 预测模式，进行和 Intra_4×4 预测模式类似的计算，生成边缘方向直方图。选择直方图中具有最大幅值的单元对应的预测模式为候选预测模式。同时，也将 DC 模式作为候选预测模式。所以，对 16×16 亮度块，只计算了 2 种预测模式的 RDO 而不是 4 种。

另外，对于 Intra_8×8 预测模式，由于 U 和 V 分量各有一个直方图，因此从两个直方图中得到两个方向的预测模式或一个方向的预测模式(当两个直方图中幅度值最大的单元相同时)。同时，也将 DC 模式作为候选预测模式。从

而，对 8×8 色度块，只计算了 2 种或 3 种预测模式的 RDO 而不是 4 种。

文献[4]中总结了该快速算法中要执行 RDO 计算的预测模式数量，如表 5.1 所示。从表中可看出，使用该快速算法，编码器将只执行 132~198 次 RDO 计算，比用 H.264 的全搜索帧内预测算法，编码器所需的 RDO 计算(592 次)少了很多。

表 5.1 Pan 算法候选模式数量

	块大小	预测模式总数	被选择的个数
亮度(Y)	4×4	9	4
亮度(Y)	16×16	4	2
色度(U，V)	8×8	4	3 或 2

5.1.2 Pan 算法的可行性验证

为了验证 Pan 算法的实际改进效果，本文对其进行了实验仿真。实验把 Pan 算法应用在 JM90 测试模型中并在 VC++6.0 平台上进行实验，实验平台的配置和实验参数设置与第四章的 4.4 节中相同。实验取 3 个 QCIF 序列进行测试：低速运动的“Claire”、快速运动的“Foreman”、纹理细节较多的“Mobile”。表 5.2 为 QP=32 时 H.264 标准算法和 Pan 算法的结果比较。

表 5.2 QP=32 时 H.264 标准算法和 Pan 算法的结果比较

序列	性能	H.264 标准算法	Pan 算法
Claire	Time (ms)	124.958	35.87
	PSNR (dB)	37.84	37.84
	Bits (kbit/s)	253.58	285.87
Foreman	Time (ms)	137.637	41.569
	PSNR (dB)	34.22	34.25
	Bits (kbit/s)	447.98	486.17
Mobile	Time (ms)	176.722	61.552
	PSNR (dB)	31.28	31.16
	Bits (kbit/s)	1423.92	1486.25

从实验结果可以看出，Pan 算法相对于 H.264 标准算法取得了较好的改进。在 PSNR 基本不变而 Bits 增加较少的情况下，编码时间有很大的节省。所以，Pan 算法在图像质量几乎不变的条件下，编码速率有明显的提高，对 H.264 的实时应用有一定价值。

5.2 改进的 Pan 算法

从上一节的分析可知，用 Pan 方法进行帧内预测时，需要对当前块中所有点的边缘矢量、边缘方向角及边缘矢量的幅值进行计算，并且还需要利用直方图统计每一个像素点边缘矢量方向所处的预测空间，所以额外复杂度较高，其仍有可改进的空间。Pan 方法给了人们启示：在帧内预测模式中，通过使用一些“先验”知识可以降低编码的计算复杂度，提高编码速率。

文献[32]中曾用一个简单的门限技术根据像素点边缘方向矢量的水平方向之和与垂直方向之和的比$(\sum dy / \sum dx)$确定 4×4 亮度块的预测模式。在该文的启发下，根据需要，本文不仅对 4×4 亮度块，而且对 16×16 亮度块和 8×8 色度块均进行了快速的预测模式选择。又因为文献[32]在预测模式的选择上存在一定的缺陷，所以本文重新给出了预测模式的选择方案。

在本文算法中，我们根据 Soble 算子求得每个像素点的边缘方向矢量，然后计算每个块(4×4 亮度块、16×16 亮度块或 8×8 色度块)中所有像素点边缘方向矢量的水平方向之和$\sum_{i,j} dy_{i,j}$和垂直方向之和$\sum_{i,j} dx_{i,j}$，最后根据$\sum_{i,j} dy_{i,j} / \sum_{i,j} dx_{i,j}$对应的角度的范围确定块的可能预测模式，其中$dx_{i,j}$和$dy_{i,j}$按照式(5-1)计算。这样相对于 Pan 算法，由于数学计算的减少，本文算法可以减少大量的计算量。

本文算法中，定义块的边缘方向角为：

$$\alpha = \frac{180^{\circ}}{\pi} \arctan(\sum dy / \sum dx), |\alpha| < 90^{\circ} \tag{5-4}$$

由于每个帧内预测模式对应一个方向，如图 5.1 所示的 Intra_4×4 的 9 种预测方向模式(16×16 和 8×8 的 4 种预测方向模式与之类似)。所以可以根据每

个块的边缘方向角所在的范围确定该块可能的预测模式。

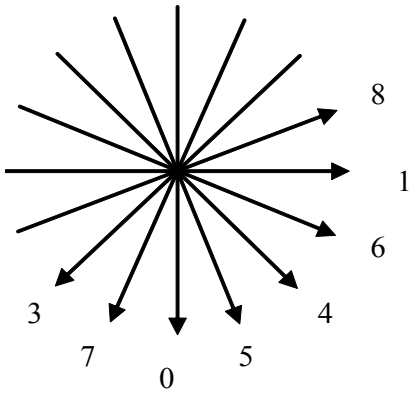


图 5.1　9 种预测方向模式示意图

本文算法中，4×4 亮度块、16×16 亮度块、8×8 色度块根据块的边缘方向角 α 的范围选择的预测模式分别如表 5.3、表 5.4、表 5.5 所示。

表 5.3 本文的 Intra_4×4 预测模式选择

块方向角 α 范围	候选预测模式
$\lvert\alpha\rvert$>76.6°	0、2、5、7
-13.3°< α <13.3°	1、2、6、8
35.8°< α <54.2°	3、2、7、8
-54.2°< α <-35.8°	4、2、5、6
-76.7°< α <-54.2°	5、2、0、4
-35.8°< α <-13.3°	6、2、1、4
54.2°< α < 76.7°	7、2、0、3
13.3°< α <35.8°	8、2、1、3

表 5.4 本文的 Intra_16×16 预测模式选择

块方向角 α 范围	候选预测模式
$\lvert\alpha\rvert$>67.5°	0、2
$\lvert\alpha\rvert$<22.5°	1、2

其他	3、2

表 5.5 本文的色度 8×8 预测模式选择

块方向角 α 范围	候选预测模式
$\lvert\alpha\rvert$>67.5°	2、0
$\lvert\alpha\rvert$<22.5°	1、0
其他	3、0

候选预测模式选择的原则与 Pan 算法类似。如表 5.3 所示，为了保证快速算法对平滑方向性不明确的亮度 4×4 块模式预测的准确性，不论 α 为何值，本方法将 DC 模式作为候选模式；由实验统计得知，利用块的边缘方向角得到的预测模式在很大概率上就是该块的最佳预测模式，因此选择其为主要候选模式；由于主要候选模式与相邻的两个预测模式在预测方向上具有相关性，所以相邻的两个预测模式也被选为候选预测模式。从而，通过 RDO 代价计算，从这 4 个候选模式中选出亮度 4×4 块的最佳预测模式。

同理，对于亮度 16×16 块(如表 5.4)或色度 8×8 块(如表 5.5)，将块方向角 α 所对应的预测模式和 DC 模式作为该块的候选预测模式，进行预测从而选出块的最佳预测模式。

Pan 算法中，每个像素都需要一次除法运算来计算这个像素边缘矢量的边缘方向、一次加法运算来计算这个像素边缘矢量的幅度值和一次加法运算来计算边缘方向直方图，因此，在计算边缘矢量后，对于一个 4×4 亮度块，共需 16 次除法运算和 32 次加法运算；而本文算法中，只需要计算 $\sum_{i,j} ay_{i,j}$ 和 $\sum_{i,j} ax_{i,j}$ 的 32 次加法运算和一次除法运算来决定 4×4 亮度块的边缘方向。16×16 亮度块和 8×8 色度块类似。本文算法和 Pan 算法数学运算量比较见表 5.6。

表 5.6 改进的 Pan 算法和 Pan 算法数学运算量比较

	4×4 亮度块		16×16 亮度块		8×8 亮度块	
	Pan	改进的 Pan	Pan	改进的 Pan	Pan	改进的 Pan
除法运算量（次）	16	1	96	1	64	1
加法运算量（次）	32	30	192	190	128	126

改进的 Pan 算法流程如图 5.2 所示。

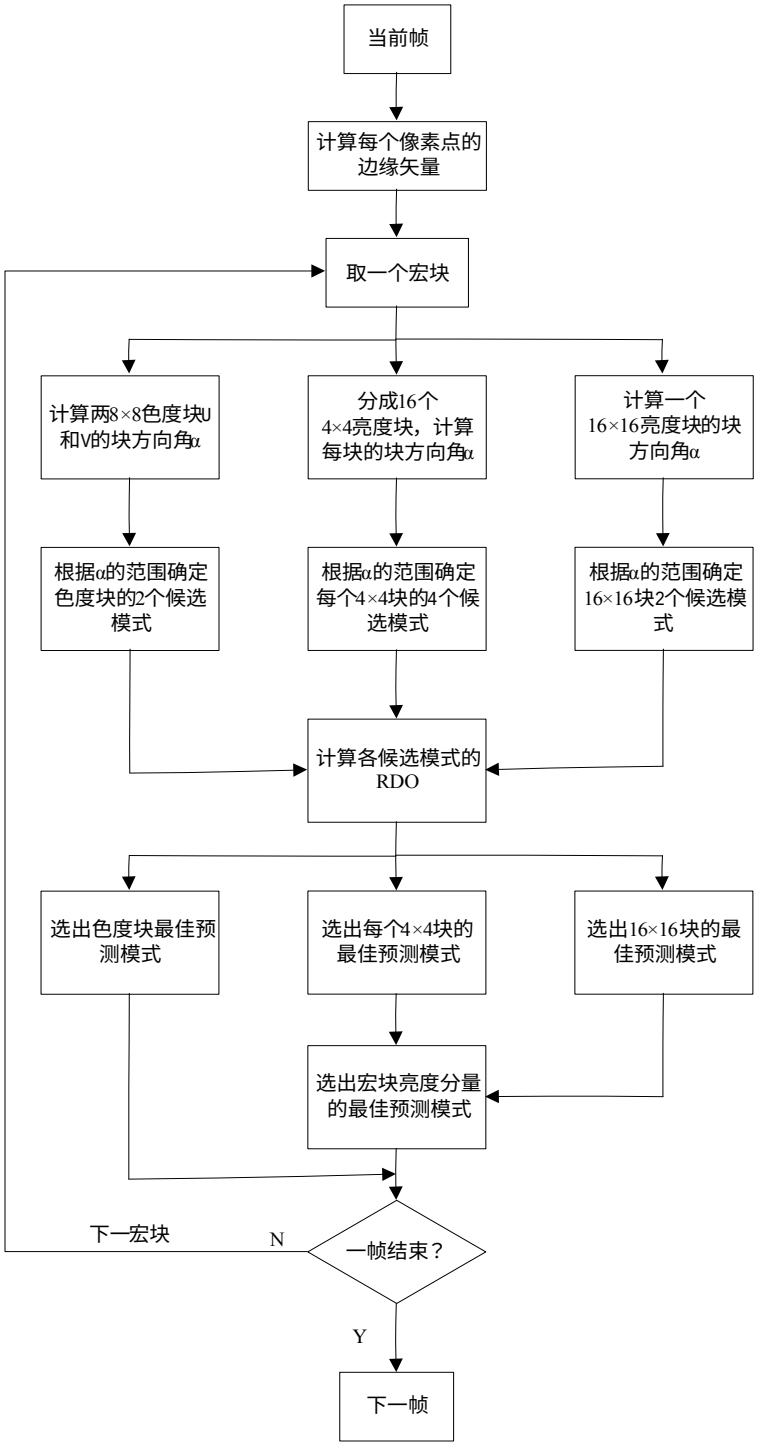


图 5.2 改进的 Pan 算法流程图

5.3 实验结果及分析

实验把改进的 Pan 算法应用在 JM90 测试模型中并在 VC++6.0 平台上进行实验，实验平台的配置和实验参数设置与第四章的 4.4 节中相同。实验取第四章中介绍的 6 个 QCIF 序列进行测试：“Foreman”、“Mobile”、“Claire”、 “News”、“Grandma”、“Coastguard”。

一、3 种算法的实验数据比较

表 5.7、表 5.8 分别给出 QP=28，32，36 时改进的 Pan 算法相对于 JM90 的全搜索算法和 Pan 算法在峰值信噪比(PSNR)、编码时间(Time)和编码比特率(Bits)三个方面的性能比较结果。其中 PSNR 有三个分量：PSNRY(亮度峰值信噪比)、PSNRU(色差 U 峰值信噪比)、PSNRV(色差 V 峰值信噪比)。表 5.7 中 ΔTime、ΔBits、ΔPSNRY、ΔPSNRU、ΔPSNRV 分别计算如下：

$$\Delta Time = \frac{T_{Pan改} - T_{JM}}{T_{JM}} \times 100\% \tag{5-5}$$

$$\Delta Bits = \frac{Bits_{Pan改} - Bits_{JM}}{Bits_{JM}} \times 100\% \tag{5-6}$$

$$\Delta PSNRY = PSNRY_{Pan改} - PSNRY_{JM} \tag{5-7}$$

$$\Delta PSNRU = PSNRU_{Pan改} - PSNRU_{JM} \tag{5-8}$$

$$\Delta PSNRV = PSNRV_{Pan改} - PSNRV_{JM} \tag{5-9}$$

表 5.7 改进的 Pan 算法相对于 JM90 的结果比较

QP	序列	Foreman	Mobile	Claire	News	Grandma	Coastguard
28	ΔTime (%)	-62.13	-63.09	-63.84	-62.15	63.01	-62.57
	ΔPSNRY(dB)	-0.03	-0.01	0.00	-0.01	-0.01	-0.05
	ΔPSNRU(dB)	-0.05	-0.02	+0.02	-0.03	-0.04	-0.07
	ΔPSNRV(dB)	-0.04	-0.07	-0.01	-0.02	+0.01	-0.03
	ΔBits (%)	+5.13	+4.45	+3.92	+5.08	+4.51	+4.81

表 5.7(续)

QP	序列	Foreman	Mobile	Claire	News	Grandma	Coastguard
32	ΔTime (%)	-62.53	-61.38	-63.44	-61.98	-62.90	-61.66
	ΔPSNRY(dB)	-0.04	-0.10	-0.01	-0.06	-0.02	-0.10
	ΔPSNRU(dB)	-0.01	-0.02	+0.01	-0.02	-0.01	-0.03
	ΔPSNRV(dB)	-0.03	-0.05	0.00	0.04	+0.01	-0.09
	ΔBits (%)	+5.10	+4.02	+3.79	+5.15	+3. 31	+4.58
36	ΔTime (%)	-62.34	-61.98	-62.97	-59.89	-61.80	-62.33
	ΔPSNRY(dB)	-0.03	-0.08	-0.02	-0.07	-0.04	-0.07
	ΔPSNRU(dB)	-0.07	-0.10	+0.01	+0.02	+0.02	-0.06
	ΔPSNRV(dB)	-0.06	-0.03	-0.02	-0.03	-0.02	-0.09
	ΔBits (%)	+4.36	+5.10	+3.78	+5.02	+4.01	+5.04

将公式(5-5)~(5-9)中脚注为 JM 的变量换成脚注为 Pan 的变量进行类似计算，可以得出改进的 Pan 算法相对于 Pan 算法的结果比较，如表 5.8 中所示。

表 5.8 改进的 Pan 算法相对于 Pan 算法的结果比较

QP	序列	Foreman	Mobile	Claire	News	Grandma	Coastguard
28	ΔTime (%)	-15.99	-18.70	-16.71	-13.79	-19.18	-16.77
	ΔPSNRY(dB)	-0.01	-0.02	0.00	-0.01	-0.03	-0.01
	ΔPSNRU(dB)	0.00	0.03	+0.01	-0.02	+0.01	-0.02
	ΔPSNRV(dB)	-0.01	0.00	+0.02	-0.01	0.00	-0.01
	ΔBits (%)	+0.97	+1.06	+0.97	+1.08	+0.91	+1.01
32	ΔTime (%)	-16.08	-15.13	-15.96	-14.90	-16.74	-17.05
	ΔPSNRY(dB)	-0.05	-0.02	-0.01	0.00	+0.01	-0.03
	ΔPSNRU(dB)	+0.02	-1.01	+0.03	-0.10	0.00	-0.04
	ΔPSNRV(dB)	-0.05	+0.02	-0.01	-0.03	-0.01	-0.03
	ΔBits (%)	+0.90	+1.00	+1.03	+0.97	+0.86	+0.99
36	ΔTime (%)	-14.55	-14.60	-15.09	-14.58	-16.92	-17.06
	ΔPSNRY(dB)	-0.01	-0.06	0.00	-0.02	-0.01	-0.06
	ΔPSNRU(dB)	-0.02	-0.01	+0.01	-0.02	+0.01	-0.02

表 5.8(续)

QP	序列	Foreman	Mobile	Claire	News	Grandma	Coastguard
36	ΔPSNRV(dB)	-0.01	-0.03	+0.01	-0.04	-0.01	-0.05
	ΔBits (%)	+1.09	+0.79	+0.52	+1.00	+0.71	+1.01

以上表中数据采用大量测试取平均值的数据处理方法。从表 5.7 的实验结果可以看出，改进的 Pan 算法和 JM90 的全搜索算法相比，编码时间有很明显的减少，平均减少约 62.32%；产生的码率相对于 JM90 增加约 4.33%。PSNRY、PSNRU、PSNRV 减小很少，这对于人视觉系统来说几乎是不影响图像质量的；从表 5.8 的实验结果可以看出，改进的 Pan 算法和 Pan 算法相比，编码时间平均减少约 16.1%；产生的码率相对于 Pan 算法增加很少，约 0.96%；而 PSNRY、PSNRU、PSNRV 的值有增加也有减少，但变化很小，基本不影响图像质量。

二、3 种算法的主观质量比较

图 5.3 给出了序列“ Foreman ”的标准全搜索算法、Pan 算法和改进的 Pan

算法的主观质量的比较图。从图中我们几乎看不出什么差别，也就是说提出的算法不影响图像的主观质量。其他结果也类似。

JM90 全搜索算法　　Pan 算法　　改进的 Pan 算法

图 5.3 Foreman 三种算法的主观质量比较图

三、3 种算法的比特率失真比较

图 5.4、图 5.5、图 5.6 分别给出了三个具有不同细节复杂度的代表序列 Foreman、Mobile、Claire 在不同码率下三种算法比特率失真的对比结果，另外三个序列也具有类似的结果。

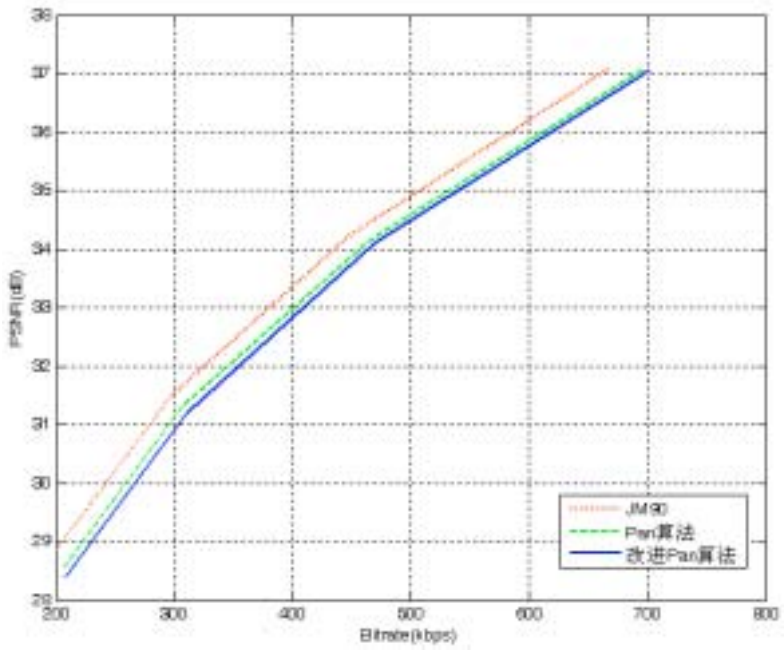


图 5.4 Foreman 比特率失真对比结果

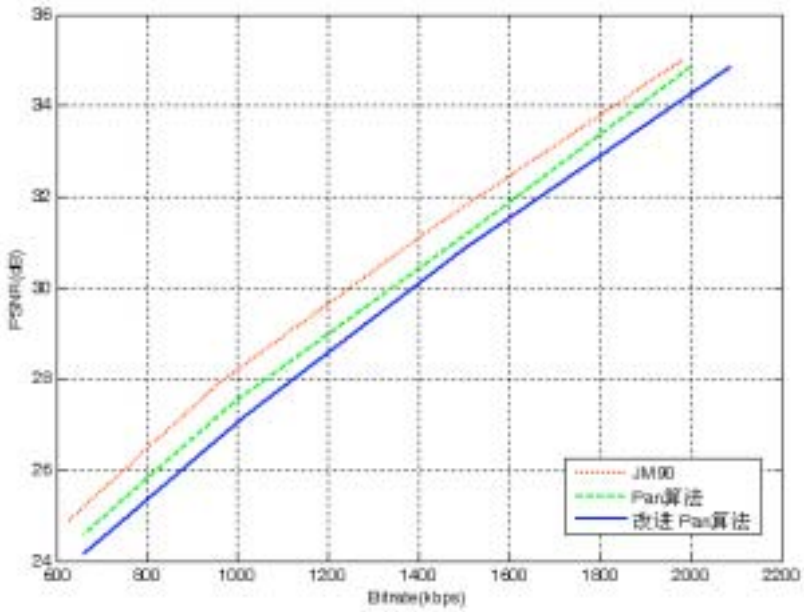


图 5.5 Mobile 比特率失真对比结果

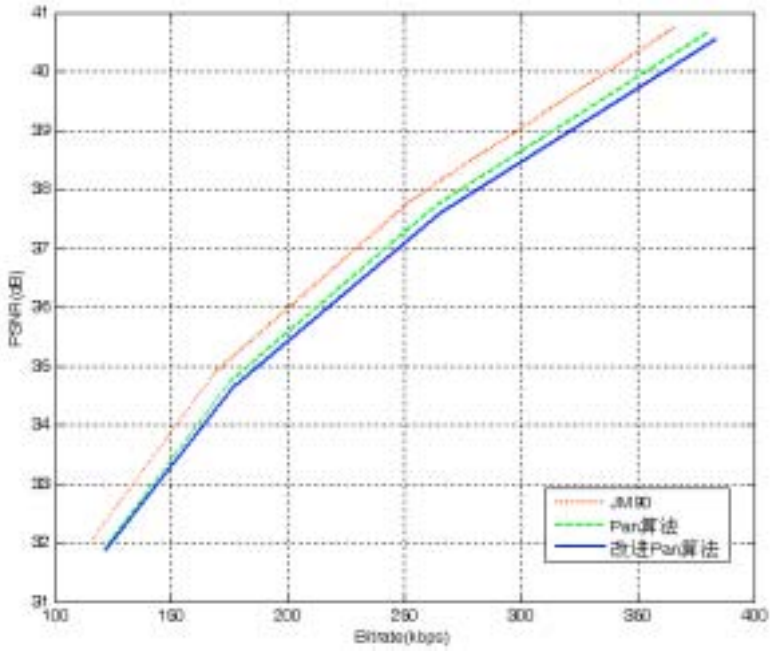


图 5.6 Claire 比特率失真对比结果

从图 5.4、图 5.5、图 5.6 中三种不同测试序列的率失真曲线可以看出，三种算法的率失真性能类似。其中，JM90 标准算法的率失真性能比另外两种算

法稍微好一些；改进的 Pan 算法和 Pan 算法的率失真性能相差更小一些。率失真性能的微小下降是编码速率的提高所需的代价，对于编码时间的大量减少来说，这些微小的代价是可以接受的。

5.4 宏块类型预判算法与改进的 Pan 算法结合实验仿真

第四章中提出了一种帧内预测宏块类型判决算法，在对宏块进行帧内预测之前先根据宏块内部相邻像素差的特征判断宏块类型是 Intra_16×16 还是 Intra_4×4 ，提前终止了一种没有价值的类型预测，这样节省了大量的时间。本章继续研究了具体宏块类型的预测模式快速选择算法，在分析其代表算法 Pan 算法的基础上，提出一种改进算法。现在我们把这两种算法结合对帧内预测模式进行选择，能取得更好的效果。

首先，根据宏块内部相邻像素差的特征判断宏块类型是 Intra_16×16 还是 Intra_4×4，如果已判断出宏块预测类型为亮度 4×4 预测，则根据块方向角 α 选择预测模式见表 5.3；如果已判断出宏块预测类型为亮度 16×16 预测，则根据方向角 α 选择预测模式见表 5.4。对于色度 8×8 预测，根据块的方向角 α 选择候选预测模式见表 5.5。结合算法、JM 全搜索算法和 Pan 算法复杂度比较见表 5.9。

表 5.9 结合算法、JM 全搜索算法和 Pan 算法复杂度比较

结合算法 (次 RDO)	宏块判为 4×4 预测类型	宏块判为 16×16 预测	未判出时
	(16*4)*2 = 128 或(16*4)*3 = 192	2*2 = 4 或 2*3 = 6	(16*4+2)*2 = 132 或(16*4+2)*3 = 198
全搜索(次 RDO)	(16*9+4)*4 = 592		
Pan 算法 (次 RDO)	(16*4+2)*2 = 132 或 (16*4+2)*3 = 198		

实验把结合的算法应用在 JM90 测试模型中并在 VC++6.0 平台上进行实验，实验平台的配置和实验参数设置与第四章的 4.4 节中相同。实验取第四章

中介绍的 6 个 QCIF 序列进行测试：“Foreman”、“Mobile”、“Claire”、 “News”、“Grandma”、“Coastguard”。

一、3 种算法的实验数据比较

表 5.10、表 5.11 分别给出 QP=28，32，36 时结合的算法相对于 JM90 的全搜索算法和 Pan 算法在峰值信噪比(PSNR)、编码时间(Time)和编码比特率(Bits)三个方面的性能比较结果。其中 PSNR 有三个分量：PSNRY(亮度峰值信噪比)、PSNRU(色差 U 峰值信噪比)、PSNRV(色差 V 峰值信噪比)。表 5.10 中 ΔTime、ΔBits、ΔPSNRY、ΔPSNRU、ΔPSNRV 分别计算如下：

$$\Delta Time = \frac{T_{结合} - T_{JM}}{T_{JM}} \times 100\% \tag{5-10}$$

$$\Delta Bits = \frac{Bits_{结合} - Bits_{JM}}{Bits_{JM}} \times 100\% \tag{5-11}$$

$$\Delta PSNRY = PSNRY_{结合} - PSNRY_{JM} \tag{5-12}$$

$$\Delta PSNRU = PSNRU_{结合} - PSNRU_{JM} \tag{5-13}$$

$$\Delta PSNRV = PSNRV_{结合} - PSNRV_{JM} \tag{5-14}$$

表 5.10 结合算法相对于 JM90 的结果比较

QP	序列	Foreman	Mobile	Claire	News	Grandma	Coastguard
28	ΔTime (%)	-72.70	-72.98	-73.37	-72.97	-73.84	-72.42
	ΔPSNRY(dB)	-0.01	-0.20	-0.09	-0.13	-0.18	-0.15
	ΔPSNRU(dB)	-0.05	-0.02	+0.2	-0.03	-0.06	-0.17
	ΔPSNRV(dB)	-0.02	-0.02	-0.03	-0.17	+0.02	-0.21
	ΔBits (%)	+3.88	+4.25	+4.10	+5.32	+4.11	+4.73
32	ΔTime (%)	-72.77	-71.15	-73.44	-72.56	-72.91	-72.15
	ΔPSNRY(dB)	-0.05	-0.17	-0.13	-0.16	-0.17	-0.1
	ΔPSNRU(dB)	-0.04	-2.19	+0.01	-0.13	-0.04	-0.33
	ΔPSNRV(dB)	-0.07	-1.7	-0.04	0.00	+0.06	-0.69
	ΔBits (%)	+3.11	+4.02	+3.53	+3.15	+2.98	+3.58
36	ΔTime (%)	-72.61	-72.09	-73.30	-72.85	-72.88	-73.10
	ΔPSNRY(dB)	-0.17	-0.18	-0.14	-0.11	-0.15	-0.07
	ΔPSNRU(dB)	-0.09	-0.01	+0.32	+0.02	+0.02	-0.69
	ΔPSNRV(dB)	-0.12	-0.03	-0.16	-0.13	-0.02	-0.91
	ΔBits (%)	+4.08	+3.60	+3.01	+2.94	+3.52	+2.81

将公式(5-10)~(5-14)中脚注为 JM 的变量换成脚注为 Pan 的变量进行类似计算，可以得出结合算法相对于 Pan 算法的结果比较，如表 5.11 所示。

表 5.11 结合算法相对于 Pan 算法的结果比较

QP	序列	Foreman	Mobile	Claire	News	Grandma	Coastguard
28	ΔTime (%)	-26.52	-24.3	-20.11	-23.79	-24.34	-20.79
	ΔPSNRY(dB)	-0.01	-0.13	-0.12	-0.09	-0.11	-0.09
	ΔPSNRU(dB)	0	0	+0.09	-0.03	+0.01	-0.13
	ΔPSNRV(dB)	-0.01	0	+0.13	-0.13	+0.03	-0.07

表 5.11(续)

QP	序列	Foreman	Mobile	Claire	News	Grandma	Coastguard
28	ΔBits (%)	+0.08	+0.11	+0.09	+0.10	+0.07	+0.10
32	ΔTime (%)	-20.73	-22.04	-23.13	-22.56	-21.69	-19.59
	ΔPSNRY(dB)	-0.05	-0.13	-0.1	-0.11	-0.13	-0.03
	ΔPSNRU(dB)	0	-1.01	+0.03	-0.10	0.00	-0.12
	ΔPSNRV(dB)	-0.04	-0.52	+0.02	+0.02	+0.05	-0.17
	ΔBits (%)	+0.07	+0.09	+0.08	+0.11	+0.07	+0.11
36	ΔTime (%)	-21.91	-19.42	-21.35	-20.97	-19.82	-19.14
	ΔPSNRY(dB)	-0.13	-0.11	-0.1	-0.07	-0.13	-0.06
	ΔPSNRU(dB)	-0.03	-0.01	+0.16	+0.04	-0.01	-0.04
	ΔPSNRV(dB)	-0.05	-0.01	+0.14	-0.01	-0.03	-0.01
	ΔBits (%)	+0.07	+0.07	+0.09	+0.07	+0.06	+0.10

以上表中数据采用大量测试取平均值的数据处理方法。从表 5.10 验结果可以看出，结合算法和 JM90 的全搜索算法相比，编码时间有很明显的减少，平均减少约 72.49%；产生的码率相对于 JM90 平均增加约 3.40%；PSNRY、PSNRU、PSNRV 减小很少，这对于人视觉系统来说几乎是不影响图像质量的。从表 5.11 验结果可以看出，结合算法和 Pan 算法相比，编码时间平均减少约 21.62%；产生的码率相对于 Pan 算法码率有很小的增加；而 PSNRY、PSNRU、PSNRV 基本保持不变，即基本上不影响图像质量。

二、3 种算法的主观质量比较

图 5.7 给出了序列 Foreman 的标准全搜索算法、Pan 算法和结合算法的主观质量的比较图。从图中我们几乎看不出什么差别，也就是说提出的算法不影响图像的主观质量。其他结果也类似。

JM90 全搜索算法　　Pan 算法　　结合算法

图 5.7 Foreman 全搜索算法、Pan 算法、结合算法的主观质量比较图

三、3 种算法的比特率失真比较

图 5.8、图 5.9、图 5.10 给出了三个具有不同细节复杂度的代表序列 Foreman、Mobile、Claire 在不同码率下比特率失真对比结果，另外三个序列也具有类似的结果。

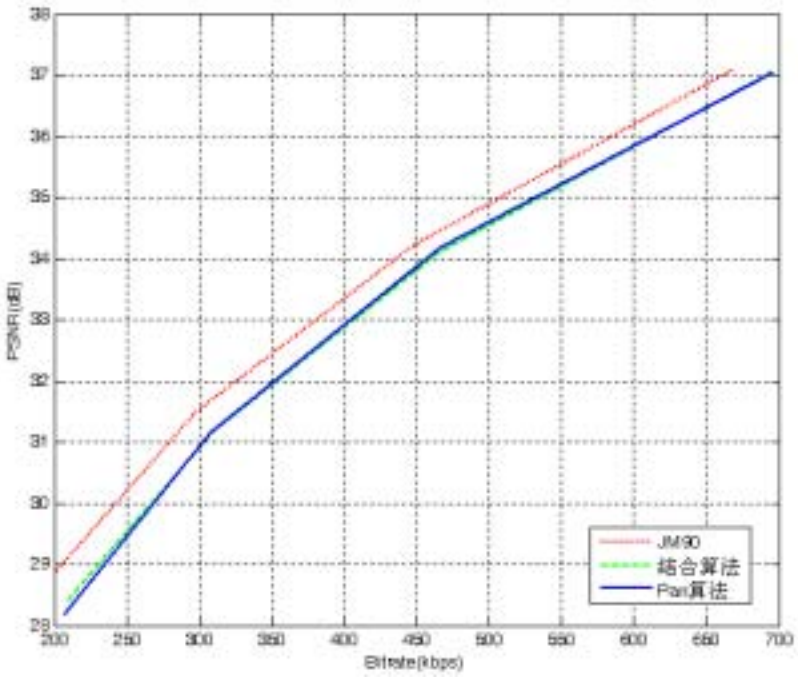


图 5.8 Foreman 比特率失真对比结果

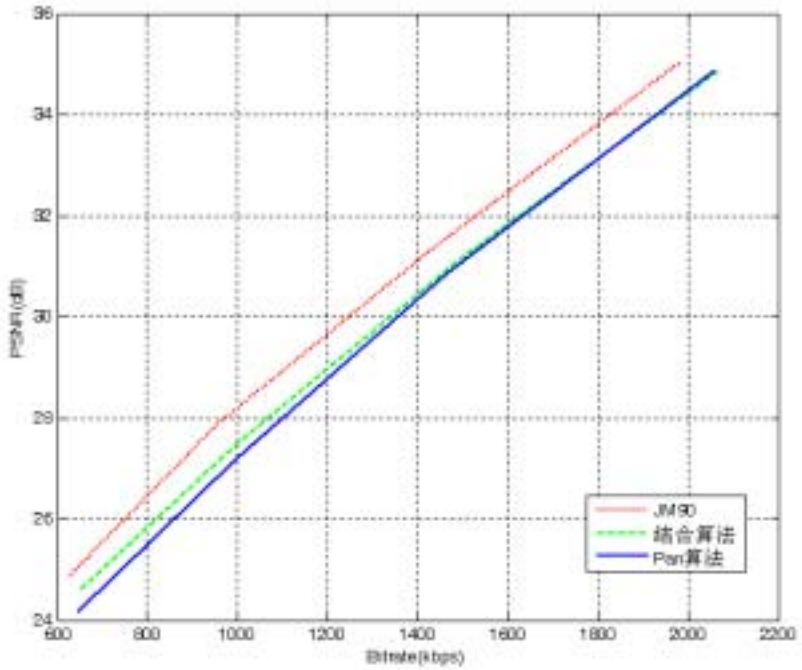


图 5.9 Mobile 比特率失真对比结果

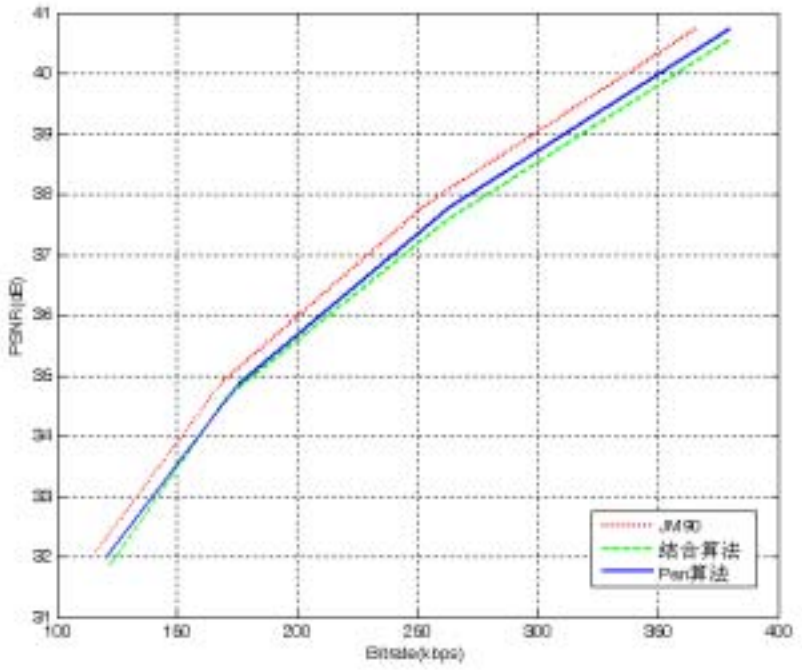


图 5.10 Claire 比特率失真对比结果

从图 5.8、图 5.9、图 5.10 中三种不同测试序列的率失真曲线可以看出，三

种算法的率失真性能类似。其中，JM90 标准算法的率失真性能比另外两种算法稍微好一些；结合算法和 Pan 算法的率失真性能相差很小，几乎没有变化。率失真性能的微小下降是编码速率的提高所付出的代价，对于编码时间的大量减少来说，这些微小的代价是值得的。

5.5 本章小结

本章在分析基于边缘方向直方图的 Pan 算法基础上，提出一种改进的 Pan 算法。该算法是一种在具体的帧内宏块类型下的预测模式快速选择算法。在 VC++6.0 平台上对改进的算法进行实验仿真，试验结果表明，提出的改进算法在 PSNR 和 Bits 变化很小的情况下下，编码时间较 Pan 算法平均节省了 16.1%，较 JM90 标准算法平均节省了 62.32%。在 5.4 节，我们又把第四章提出的帧内预测宏块类型判决算法和改进的 Pan 算法结合对帧内预测模式进行选择，并进行了实验仿真。从实验结果可以看出这种结合的算法编码性能更加优异：相对于 JM90 标准算法，在 PSNR 变化很小、平均码率增加约 3.40%的情况下，编码时间平均节省了 72.49%，；相对于 Pan 算法，在 PSNR 基本不变、码率稍有增加的情况下、编码时间平均节省了 21.62%。这对于 H.264 的实时应用具有很大的意义。

第六章 总结及展望

6.1 本论文研究工作总结

H.264 是 ITU-T 在继 H.261 和 H.263 之后，提出的新一代的视频压缩编码标准。它在视频压缩效率方面比目前所有的视频压缩标准都有显著的提高，但压缩效率的提高是以算法的计算复杂度增加为代价的，这使得 H.264 很难应用于实时性要求较强的场合。如何降低其复杂度，提高编码速度便成了目前研究的重点和难点。

本文通过对视频压缩标准 H.264 的深入研究，发现帧内预测作为 H.264 视频编码过程的一部分，采用了多种预测模式，对于一个宏块，共有 $M_8 \times (M_4 \times 9 + M_{16})$ (其中 M_8=4, M_4=16, M_{16}=4)种不同的组合，因此共需完成 592 次 RDO 计算，才能获得最优的帧内预测模式编码。并且在最优模式选择的过程中，重复计算了亮度宏块在帧内预测各个模式下的率失真开销。因此，对帧内编码进行快速算法优化可以显著地降低整体复杂度。

基于上述观点，在基本保证 H.264 编码性能的条件下，本文通过实验分析提出了 2 种帧内预测模式选择快速算法。

本论文完成的主要工作如下：

(1) 给出了 H.264 编码器的基本框架，概述了最新视频编码标准 H.264，简要介绍了其采用的新技术：帧内预测、帧间预测、变换与量化、熵编码和去方块滤波。

(3) 第三章指出了减小帧内预测计算复杂度的方法可从两方面考虑：1) 简化代价函数；2) 缩小预测模式选择的范围。介绍了目前测试模型 JM 中的代价函数和全搜索帧内预测模式选择算法流程，分析了帧内预测全搜索算法的计算复杂度，并指出其缺点和改进的方向。然后介绍了几种现有的典型帧内预测快速算法，并阐述其优缺点。

(4) 第四章首先通过对 H.264 帧内预测中宏块类型选择特点的分析，验证了 Intra_16×16 模式适合用于比较平坦的宏块；Intra_4×4 模式适合用于细节较

多的宏块。基于此结论，我们选取宏块内部相邻像素差来表征宏块细节丰富程度。在分析宏块内部相邻像素差的特点和宏块预测类型的相关性的基础上，提出一种基于宏块内部相邻像素差的帧内宏块类型预判算法。将提出的算法应用到 JM90 测试模型中并在 VC++6.0 平台上进行实验，实验结果表明该算法对帧内预测模式快速选择有显著的作用。

(5) 为了进一步提高编码效率，本文继续对具体宏块类型下的帧内预测模式快速选择算法进行研究。第五章中，首先介绍 Pan 算法并对其进行实验仿真，给出了实验结果；其次对基于边缘方向直方图的 Pan 算法进行改进，提出一种改进的算法，在 H.264 测试模型 JM90 上实现并给出实验结果，结果表明，提出的改进算法相对于 Pan 算法有一定的改进；最后把第四章中提出的帧内宏块类型预判算法和改进的 Pan 算法相结合进行实验仿真，结果表明，结合算法在失真率和码率性能基本不变的前提下，编码时间平均减少了 72.49%，大大提高了编码效率，对 H.264 的实时应用具有很大的意义。

6.2 进一步工作及展望

基于 H.264 的视频编码技术涉及很多难点。可以从帧内模式选择、帧间模式选择、快速的高精度运动估计算法等几方面进行研究而减少 H.264 编码器的计算复杂度。由于时间有限，本文只对帧内模式选择算法进行了研究。虽然提出的快速算法在保证失真率和码率性能基本不变的前提下，编码速度有很大的提高，对 H.264 实时应用有一定价值，但相对于计算复杂度很高的 H.264 编码器来说，减少的时间还是有限的。

在以后的工作中，本人计划从帧间预测着手研究，继续对帧间模式选择算法进行优化，更大程度地降低 H.264 编码器的复杂度。H.264 的帧间宏块预测包括了 SKIP 模式和 7 种编码模式(16×16，16×8，8×16，8×8，8×4，4×8 和 4×4)，选出最优的帧间模式后，还要进行帧内预测，最后求出最优的预测模式。对于每个宏块，最多需要计算 768 次 RDO[33]，这使得编码器复杂度大大增加，所以对帧间预测模式选择优化算法的研究是有必要的。

参考文献

[1] ITU-T Rec., H.261, Video codec for audio visual services at p×64kbit/s, March 1993.

[2] ITU-T Rec. H.263, Video coding for low bit rate communication, May 1995.

[3] 丁宗豪，多媒体应用的图像压缩及其标准 MPEG-1，电视技术，1994, 8:2~5.

[4] ISO/IEC JTCI, Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbit/s-Part2Video.ISO/IEC 11172(MPEG-1), 1993.

[5] ISO/IEC 13818-2, Information technology-Generic coding of audio-visual objects part2: Visual, Oct.1998.

[6] 李幼林，MPEG-2 数字技术简介，电声技术，1997, 4:41~44.

[7] Rob Koenen, Overview of the MPEG-4 Standard, ISO/IEC JTC1/SC29/WG11 N1730, Stockholm, July 1997.

[8] ISO/IEC, Information Technology-Coding of Audio-Visual Objects: Visual ISO/IEC 14496-2 Committee Draft of 15, May 1998.

[9] 何艳辉，朱珍明，H.264 视频编码技术研究，湖南大学学报，2003, 30(3):196~199.

[10] ISO/IEC JTC1/SC29/WG11 N3908, MPEG-4 video verification model ver.18.0, Pisa, Jan. 2001.

[11] Wiegand T,Gary J,Sullivan G J,et al, Overview of the H.264/AVC video coding standard, IEEE Transaction on Circuits and Systems for Video Technology, 2003,13(17):560~576.

[12] 李世平，蒋刚毅，郁梅，快速帧内预测模式选择新方法，电子学报，2006，34(1):141~146.

[13] Pan Feng,L in Xiao, et al, Fast mode decision for intra prediction, JVT-G013 in ISO / IEC JTC1 /SC29 /WG11 and ITU-T SG16 Q. 6, JVT 7th Meeting. Pattaya, Thailand: IEEE, 2003.

[14] RICHARDSON IEG, Video Codec Design, New York, John Wiley&Sons, 2002.

[15] 周斌，严德聪，杨宗凯，H.264/AVC 先进视频编码研究，计算机工程与设计，2004, 25(9):1523~1532.

[16] Iain E.G.Richardson, H.264/MPEG-4 Part10 White Paper, www.vcodex.com, 2003.

[17] Iain E.G.Richardson, H.264/MPEG-4 Part 10: Intra Prediction, www.vcodex.com, 2003.

[18] Iain E.G.Richardson, H.264/MPEG-4 Part 10: Inter Prediction, www.vcodex.com, 2003.

[19] 魏芳，李学明, H.264 中变换和量化的 SIMD 优化，计算机工程与应用, 2004, 17:24~27.

[20] Iain E.G.Richardson, H.264/MPEG-4 Part 10: Transform & Quantization, www.vcodex.com, 2003.

[21] 毕厚杰，新一代视频压缩编码标准—H.264/AVC，北京，人民邮电出版社，2005:118~125.

[22] Joint Video Team(JVT) of ISO/IEC MPEG and ITU-T VCEG Dratf ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec H.264|ISO/IEC 14496-10AVC)[S],document JVT GOSOd35doc,7th Meeting Pattaya,Thailand, March 2003.

[23] Ortega and K.Ramchandran, Rate-distortion methods for image and video compression, IEEE Signal Processing Mag., 1998, 15(6):23~50.

[24] G. J. Sullivan and T. Wiegand, Rate-distortion optimization for video compression, IEEE Signal Processing Mag., 1998, 15(6):74~90.

[25] Thomas Wiegand, Bernd Girod, Lagrange Multiplier Selection in Hybrid Video Coder Control, IEEE International Conference on Image Processing, 2001, 3:542~545.

[26] 裴世保，李厚强，愈能海，H.264/AVC 帧内预测模式选择算法研究，计算机应用，2005, 25(8):1808~1810.

[27] K. Changsung, S. Hsuan-Huei, K. Chung-Chieh Jay, Multistage mode decision for intra prediction in H.264 codec, Visual Communications and Image Processing 2004, 2004, 5308:355~363.

[28] 谢晶，贾克斌，一种基于二维直方图的 H.264/AVC 快速帧内预测判决算法，电子与信息学报，2005, 27(7):1054~1057.

[29] 尹宝才，孙磊，孔德慧，基于递推模式的帧内预测快速算法，北京工业大学学报，2006, 32(3):252~256.

[30] 冈萨雷斯等著，阮秋琦等译，数字图像处理，北京，电子工业出版社，2003:35~50.

[31] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, D.Wu, S. Wu, Fast Mode Decision Algorithm for Intraprediction in H.264/AVC video coding, IEEE Transaction on Circuits and Systems for Video Technology, 2005,15(7):813~822.

[32] Chern-Loon Lim, Kim-Han Thung, P. Raveendran, EDGE VECTOR BASED MODE DECISION FOR H.264/AVC INTRA PREDICTION, First Asia International Conference on Modelling & Simulation, March 2007:308~312.

[33] JEON B, LEE J. Fast mode decision for H.264. Joint Video Team (JVT) of ISO/IEC

MPEG & ITU-T VCEG. Waikoloa, Hawaii, USA, 2003.

[34] 杜博,方向忠,一种新的 H.264 帧内预测快速算法,电子测量技术,2006, 29(4):111~140.

[35] 田晓冬，田裕鹏，一种改进的快速 H.264/AVC 帧内预测 Pan 算法，计算机应用，2006, 26(10):2383~2385.

[36] J. Kim and J. Jeong, Fast Intra-Mode Decision in H.264 Video Coding Using Simple Directional Marks,Proc. Of SPIE, 2005, 5960:1071~1079.

[37] Abdul H.Sadka, Compressed Video Communications, 北京，科学出版社，2004.

[38] Lee and B.Jeon,Fast Mode Decision Based on Variable Block Size Motion Compression for H.264,Lecture Notes in Computer Science(LNCS), 2003, 2899:410~418.

[39] N. Kamaci, Y. Altunbasak, Performance compassion of the emerging H.264 video coding standard with the existing standards, IEEE Int. Conf. Multimedia and Expo., 2003, 19:345~348.

[40] Y. Zhang, F. Dai, S. Lin, Fast 4×4 intra-prediction mode selection for H.264, IEEE Int. Conf. Multimedia and Expo, 2004, 2:1151~1154.

[41] Iain E.G.Richardson 著，欧阳合，韩军译，H.264 和 MPEG-4 视频压缩，长沙，国防科技大学出版社，2004.

致　谢

在论文收笔之际，我要向所有在我攻读硕士学位期间指导我，帮助我，关心我，鼓励我的老师，同学，亲人和朋友，致以我最诚挚的感谢！

首先，我要感谢我的导师常建平教授，两年半的研究生期间，常老师一直给予我谆谆教诲与殷切关怀。在我的论文完成过程中，常老师悉心指导、严格要求、精益求精，在常老师的鼓舞下，我克服了课题中一个又一个的困难。常老师严谨求实的治学态度，对科学研究敏锐的判断力和深邃的见解让我受益终生！在此，谨向我的恩师表示最诚挚的感谢和最深切的敬意。祝愿您身体健康！工作顺利！

其次，在南航读研期间，有幸认识很多朋友，他们在学习、生活方面都给了我很大的帮助，感谢所有帮助过我的同学和朋友。

最后，向一直默默关心和支持我的父母表示感谢！没有他们的物质和精神的支持，我就不可能取得今天的成绩。祝愿他们永远健康平安。

攻读硕士期间所发表的论文

[1] 王慧，常建平，一种改进的 H.264 快速帧内模式选择算法，计算机工程(已录用)

[2] 王慧，常建平，改进的 H.264 快速帧内预测算法，南京航空航天大学第九届研究生学术会议论文集