

武汉理工大学

硕士学位论文

基于SSE4指令集的H. 264编码标准的运动估计优化

姓名：范亚琼

申请学位级别：硕士

专业：信号与信息处理

指导教师：杨杰

20100501

摘 要

H.264, 同时也是 MPEG-4 第十部分, 是由联合视频组(JVT, Joint Video Team)提出的高度压缩数字视频编解码器标准。与之前的视频编码标准相比, H.264 具有低码率、高质量的压缩画面和优秀网络适应性等优点。目前已在视频压缩、数字电视广播、流媒体等领域得到广泛的应用。但是, H.264 优越的编码性能是以其高复杂度为代价的。因此, 如何在保证视频质量的同时, 降低算法复杂度, 提高编码速度是目前对 H.264 研究的热点。

首先, 本文阐述了课题研究背景、视频编码简史、相关的视频编码标准及研究现状等。分析了 H.264 的编解码框架及关键技术。研究了快速运动估计中, 宏块匹配的准则和模式选择原理, 深入分析了四种典型的运动估计算法: 全搜索算法、EPZS 算法、UMHexagonS 算法及简化 UMHexagonS 算法的原理和算法特征, 在 JVT 的 JM10.2 平台上分别实现了这四种算法, 比较了其性能优劣。全搜索算法最精确、算法思想简单, 实验结果表明 EPZS 算法、UMHexagonS 算法及简化 UMHexagonS 算法在全搜索算法基础上能够显著提高编码速度, 同等条件下, 简化 UMHexagonS 算法性能最优。

接着, 阐述了 SSE 指令集发展历史, 重点研究 SSE4 指令集中可快速提升视频编码速度的两条指令, MPSADBW 指令和 PHMINPOSUW 指令, 详细分析了这两条指令的参数, 计算原理, 返回结果等。分析使用 SSE4 指令实现 H.264 快速运动估计算法优化的可行性, 并在 JVT 的 JM8.6 平台上实现了使用 SSE4 指令集实现快速计算 SAD。

然后, 使用 SSE4 指令集在 JVT 的 JM8.6 平台上实现了快速运动估计算法优化, 通过软件仿真, 与 JM8.6 平台上的全搜索算法进行性能比较, 详细分析了峰值信噪比增量、码率增量、编码时间节省因子三个衡量算法优劣的性能指标。最后得出结论, 本论文提出的基于 SSE4 指令集的 H.264 编码标准的运动估计优化算法, 在保证较小图像质量损失、较低码率增量同时, 能显著减少编码时间, 节省编码时间 18~30%。该算法更适用于运动强度较小的视频序列。

最后, 对本文工作作出了总结, 并对未来的研究方向作出展望。将把对运动估计搜索算法的优化及 SSE4 指令集优化结合, 以及对 T264 平台的研究等作为今后的研究方向。

关键词: H.264, 快速运动估计, SSE4, 搜索算法, JM 平台

Abstract

H.264/MPEG-4 part 10 is an international video coding standard proposed by Joint Video Team (JVT), which is a joint group of ITU-T VCEG and ISO/IEC MPEG. Compared with existing video coding standards, H.264 can provide lower bit rates, higher image quality and better adaptability on net transmission. H.264 has been widely applied in a range of fields, such as video compression, digital television broadcasting, and stream media and so on. However, on one hand H.264 provides predominant compression efficiency, on the other hand, it increases coding complexity at the same time. Therefore, how to reduce the arithmetic complexity and accelerate the coding process without decreasing the video quality is a hot spot in the video compression field.

Firstly, in this dissertation the background of research in video compression, the development of video coding technology, the existing video standards and the state-of-the-art of H.264 video coding standard are expounded. The framework of encoding and decoding and the key technologies of H.264 are discussed in the dissertation. The fast motion estimation algorithms, the principles of macro block matching as well as the mode decision principle are discussed. Then this dissertation makes an in-depth research on four classic motion estimation algorithms, which are Full Search Algorithm, EPZS Algorithm, UMHexagonS Algorithm and Simplified UMHexagonS Algorithm. On platform of JM10.2 of JVT, the four classic motion estimation algorithms are used separately to encode one YUV file. According to the experiment data, these four motion estimation algorithms are compared. The result shows that Full Search Algorithm has a most accurate image state and a simplest estimation algorithm, while the other three motion estimation algorithms can increase the coding speed compared with the Full Search Algorithm. Simplified UMHexagonS Algorithm has an optimal performance in these four algorithms.

Secondly, this dissertation makes a brief introduce about the development history of SSE instruction set. Two instructions, MPSADBW instruction and PHMINPOSUW instruction, are in-depth discussed. These two instructions can be used together to improve video encode. The parameters, calculate methods and results

in return of these two instructions are analyzed. On platform of JM8.6 of JVT, MPSADBW instruction and PHMINPOSUW instruction in SSE4 instruction set are used together to calculate SAD with higher speed, which shows that SSE4 instruction set can improve fast motion estimate optimization of H.264.

Thirdly, on platform of JM8.6, SSE4 instruction set is used to improve the fast motion estimate optimization. Compared with the Full Search Algorithm on platform of JM8.6, the parameters such as: Δ PSNR, Δ Bits, Δ Time are calculated. The experimental results show that, the optimization algorithm on motion estimation of H.264 based on SSE4 instruction set proposed in this dissertation can save 18%-30% total encoding time without significant image degradation or bit rate increasing. The proposed algorithm can get better performance in encoding video sequences with less motion intensity.

Last but not the least, summary of this dissertation is concluded, and vista is depicted. Combination of optimization algorithm on motion estimation and SSE4 instruction set, as well as the platform of T264 are taken into consideration for future research.

Keywords: H.264, fast motion estimation, SSE4, search algorithm, JM platform

独 创 性 声 明

本人声明，所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得武汉理工大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签 名： 董亚琼 日 期： 2010.5.10

学位论文使用授权书

本人完全了解武汉理工大学有关保留、使用学位论文的规定，即学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权武汉理工大学可以将本学位论文的全部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或汇编本学位论文。同时授权经武汉理工大学认可的国家有关机构或论文数据库使用或收录本学位论文，并向社会公众提供信息服务。

(保密的论文在解密后应遵守此规定)

研究生（签名）： 董亚琼 导师（签名）： 杨 日期： 2010.5.10

第 1 章 绪论

1.1 研究背景

随着数字通信、集成电路技术、计算机网络及多媒体技术等信息技术的飞速发展,人类对数字领域的多媒体通信业务的需求日益增加。多媒体信息包括文字、声音、图像和视频等信息,其中视频通信是多媒体通信的最重要组成部分。视频信息直观、准确、信息容量大,极易被人们接受,据统计人类接收的信息大约有 70%来自视觉^[1]。因此,视频通信作为主要的业务方式,已融入了人们的生产生活之中并日益紧密。

视频媒体信息,经数字化以后,具有易于加密、抗干扰以及可再生中继等优点。但是视频媒体信息包含巨大信息容量的同时,其压缩前的数据量也无比巨大,不易传输、存储。例如:一部 45 分钟左右的 SIF 格式($352 \times 288 \times 25\text{FPS}$)视频数据,压缩前的数据量约为 10GB,而一张容量为 650MB 的 VCD 只能存放约 4.46 分钟的视频数据;一部 120 分钟左右的高清视频数据($720 \times 480 \times 30\text{FPS}$),压缩前的数据量约为 100GB,需要 15 张容量为 6.67GB 的 DVD 只能存储该节目;广播级的高清视频节目($1920 \times 1080 \times 30\text{FPS}$),需要经过 30:1 或者 60:1 的压缩才能在通信带宽为 24Mbps 上进行传播;而手机视频聊天($320 \times 240 \times 15\text{FPS}$),需要大于 100:1 的压缩,才能在单向带宽约为 128Kbps 的信道上进行传输。由以上数据可知,未经压缩的视频数据不可能在如今的数字带宽下传输,也不适宜存储,这是阻碍人们有效进行视频通信的瓶颈之一。因此,必须对海量的视频媒体数据进行高效的压缩编码,并以相应的压缩格式进行存储和传输。对视频编码技术的研究成为当今多媒体技术研究的热点之一,是解决多媒体通信和信息存储问题的关键技术之一。

在此背景下,国际标准化组织(ISO, International Standardization Organization)、国际电工联合会(IEC, International Electrotechnical Commission)及国际电信联合会(ITU-T, International Telecommunication Union Telecommunication Standardization Sector)陆续制定了一系列图像、音频及视频的国际压缩标准,其中包括 JPEG 系列、MPEG 系列及 H.26X 系列标准^[2],推动了音视频技术在各个领域的迅速普及与发展。

H.264, 同时也是 MPEG-4 第十部分, 是由 ISO/IEC 动态图像专家组(MPEG, Moving Picture Experts Group) 和 ITU-T 视频编码专家组(VCEG, Video Coding Experts Group)联合组成的联合视频组(JVT, Joint Video Team)提出的高度压缩国际视频编解码标准。在 ITU-T 和 ISO/IEC 两大国际组织的积极努力、合作与推动下, H.264 吸纳了近几年来视频编码方面的先进技术, 并将它们很好地结合在一起, 以良好的视频图像质量、较高的编码压缩效率和友好的网络适应性成为了新一代国际视频编码标准。与同类的其它视频编码标准相比, H.264 具有技术层面的先进性、研究层面的开放性及应用层面的国际性等优势, 引起了业界与学术界的强烈关注和积极参与, 但其高效的编码率的取得是以高计算复杂度为代价的。H.264 新引进了多种编码技术, 如: 多模式的空间域帧内预测技术、多种块划分的帧间预测技术、多参考帧运动搜索和运动补偿技术、自适应的熵编码技术以及去块效应的环路滤波技术等, 这些先进技术虽然极大的提高了编码效率, 但相应的计算复杂度也随之提高。相关人士对其编码效率和计算复杂度进行了全面的评估比较, 评估实验结果表明, 与以往的视频编码标准 H.263 相比, H.264 的编码效率提高了 50%, 但其编码器的计算复杂度提高了 4-5 倍, 而解码器的计算复杂度也提高了 2 倍左右。

保持原有算法的视频编码质量和高压缩率的同时, 并能显著降低算法的计算复杂度是 H.264 推广和扩展应用的关键所在。此外, 由于其出色的编码性能和优良的网络适应性, 它将会取代以往的视频编码标准并成为未来一段时间内各种视频处理、存储、传输应用的主要标准, 其发展前景被业内人士看好, 快速算法的研究也引起越来越多的关注。因此, 在保持算法原有性能的基础上, 降低计算复杂度, 减少编码时间这方面开展研究工作具有十分重要的理论意义和有实际应用的必要性。

1.2 视频编码标准简介

1.2.1 视频编码发展简史

1984 年国际电报电话咨询委员会(CCITT, Consultative Committee of International Telegraph and Telephone)第 15 研究组发布了数字基群电视会议编码标准 H.120 建议。1988 年 CCITT 通过了“ $p \times 64\text{Kbps}$ ($p=1, 2, 3, 4, 5, \dots, 30$)”视像编码标准 H.261 建议, 被称为视频压缩编码的一个里程碑。之后, ITU-T、ISO

等陆续公布了一系列基于波形的视频编码标准，这些标准的编码思想都是基于 H.261 的混合编码框架。

1986 年，ISO 和 CCITT 成立了联合图像专家组(JPEG, Joint Photographic Experts Group)，该组织专门对具有连续色调的静止图像压缩算法开展研究，并于 1992 年 7 月通过了 JPEG 国际图像压缩标准。

1988 年 ISO/IEC 信息技术联合委员会成立了活动图像专家组(MPEG, Moving Picture Expert Group)，对动态图像压缩算法开展研究。1991 年公布了 MPEG-1 视频编码标准，码率为 1.5Mbps，主要用于家用的 VCD 影碟视频压缩处理；1994 年 11 月，公布了 MPEG-2 标准，码率从 4Mbps、15Mbps 直至 100 Mbps 分别用于不同档次和不同级别的视频压缩处理，主要用于数字视频广播(DVB)、家用 DVD 视频压缩存储及高清数字电视(HDTV)等。

1995 年，ITU-T 推出 H.263 标准，用于低码率视频传输，传输码率低于 64 Kbps。公用开关电话网络(PSTN, Public Switched Telephone Network)信道中可视会议、多媒体通信等均使用的 H.263 编码标准。1998 年和 2000 年 ITU-T 又分别公布了 H.263+、H.263++等标准。

1999 年 12 月份，ISO/IEC 通过了“视听对象的编码标准”——MPEG-4，它不仅定义了音视频的压缩标准，而且还强调了多媒体通信的交互性和灵活性。

2003 年 3 月，ITU-T 和 ISO/IEC 正式公布了 H.264 视频压缩标准，它具有良好的视频图像压缩质量、高压缩比、良好网络适应性等优点，并加强了对 IP 网、移动网的误码和丢包的处理。

1.2.2 MPEG 标准

MPEG-1^[3]标准由三个部分组成，它包括 MPEG-1 视频、MPEG-1 音频和 MPEG-1 系统。MPEG-1 的传输效率大约为 1.5Mbps，其压缩算法主要采用了 DCT 变换编码技术和运动补偿技术，DCT 变换技术可以减少帧内的空间冗余，而运动补偿技术可以减少相邻帧之间的运动冗余，压缩比最大可达到 200:1。MPEG-1 视频可以将格式为 352×288×25(图像帧速为每秒 25 帧，每帧图像的扫描行数为 288 行，每行像素为 352 像素) 或 352×240×30 的视频信息和音频信息进行有效压缩。MPEG-1 音频定义了三个层次，每个层次代表了不同的编解码方式和音频质量。其中，第三层去除了音频信号之中人耳听觉阈值以外的所有信号，并且去除了大信号掩盖下的小信号，人耳具有掩盖效应，因此，基本察觉不出小

信号被去除,这样,实际记录的信息量就比原始的音频信息数据量小很多,其压缩比可达到 101~961。原本一张只能容量十多首歌曲的光盘就可以记录 150 首以上第三层编码的歌曲,这就是我们熟知并且普遍使用的 MP3 音乐。

由于计算机技术、数字图像技术、多媒体通信以及交互式电视技术等相关技术的告诉发展,MPEG-1 在音视频分辨率和传输率方面已不能满足人们日益增长的需求,所以 1994 年,ISO/IEC 又推出了 MPEG-2 运动图像及伴音通用压缩编码标准。

MPEG-2^[4]标准分为系统、视频、音频三大部分,也称为“动态图像和伴音的通用标准”标准,可以对格式为 720×576×25 或 720×460×30 的广播级视频图像进行压缩,其码率约为 3~15Mbps。目前使用的高清数字电视 HDTV、DVD 和数字广播电视等数字视频压缩技术即采用 MPEG-2 标准。

MPEG-4^[5]也称为“视听对象的编码标准”,由 ISO/IEC1999 年 12 月提出,可以支持低 比特率多媒体通信与访问的处理方式,不仅定义了视频音频的压缩处理标准,更具有交互性、灵活性和扩展性等优点。MPEG-4 采用人工智能和图像合成技术,可以以极高的压缩比实现精确清晰的视频画面。

1.2.3 H.26X 标准

H.261^[6]编码标准是最早的运动图像压缩标准,是 ITU-T 为在综合业务数字网(ISDN, Integrated Services Digital Network)上开展双向声像业务而制定的,时速为“ $p \times 64\text{Kbps}(p=1, 2, 3, 4, 5, \dots, 30)$ ”,主要应用于可视电话、视频会议等。H.261 只对 CIF 和 QCIF 两种图像格式进行处理,每帧图像分成图像层、宏块组(GOB)层、宏块(MB)层、块(Block)层来处理。它详细制定了视频编码的各个部分,包括运动补偿的帧间预测技术、DCT 变换及反变换、量化及反量化、熵编码,以及与固定速率的信道相适配的速率控制技术。

H.263^[7]信源编码方法与 H.261 类似,与后者不同的是,H.263 支持多种输入格式,编码后输出为 H.263 码流。传输码率最初定为低于 64Kbps,但实际应用中,范围已远远超出低码率图像编码范围,例如:16QCIF 已是高清晰度电视水平。因此,可以说 H.263 也适于高速率图像编码。为了进一步提高编码图像质量,并适应低码率传输要求,ITU-T 相继通过了 H.263+、H.263++编码标准,在适应低码率传输和提高图像质量方面做了不少改进。

H.264^[8]是由 MPEG 和 VCEG 联合组成的 JVT 开发的一个比早期研发的

MPEG 系列和 H.26X 系列性能更好的视频压缩编码标准,被命名为先进视频编码标准 H.264/AVC(Advanced Video Coding),也被称为 MPEG-4 的第 10 部分,在本论文中统一简称为 H.264。新一代国际视频编码标准 H.264 已于 2003 年 3 月被 ITU-T 正式通过并在国际上正式颁布,这些年来已在多媒体通信方面取得越来越多的普及与应用。

1.2.4 AVS 编码标准

面对 MPEG 系列及 H.26X 系列等编码标准高昂的专利费,我国数字视频产业面临着严峻挑战,为了提高国内数字音视频产业的核心竞争力,2006 年 6 月国家信息产业部科学技术司批准成立了“数字音视频编解码技术标准工作组”,联合国内从事数字音频视频编解码技术研发的科研机构和企业等,针对我国国情,根据具体国内音视频产业的现状与需求,提出了具有我国自主知识产权的信源编码标准——《信息技术 先进音视频编码》系列标准,简称 AVS(Audio Video coding Standard)^[9]。AVS 标准包括系统、视频、音频、数字版权管理等四个主要技术标准和一致性测试等支撑标准。

1.3 本课题研究现状

从 1984 年 CCITT 公布第一个视频编码国际标准 H.261 以来,至今已有 26 年历史。MPEG 系列和 H.26X 系列视频编码大大推动了视频通信和数字电视广播的发展。2003 年 3 月 ISO/ITU-T 正式公布 H.264 视频编码标准,至今已有七年。H.264 是新一代的视频编码标准,但其在实际当中的应用仍然存在着一些不足,其原因在于 H.264 引进了许多先进的算法,如:多模式的空间域帧内预测、多种块划分的帧间预测、多参考帧运动搜索和运动补偿、自适应熵编码及去块效应的环路滤波技术等,这些先进算法虽然极大的提高了编码效率,但是必然会引起编码复杂度的提高。相对于之前的 H.263 视频编码标准,对视频的编码复杂度约提高了 3 倍,解码复杂度约提高 2 倍^[10]。编码复杂度的提高必然导致了 H.264 整个系统的计算量,增加整个编码系统的运算时间,这样,在一些实时性要求较高的场合,如:视频会议,数字视频直播等应用中,其优秀的压缩性能就无法显现出来。因此,如何在不影响视频图像质量的同时,有效的降低编码复杂度,更好的提升 H.264 在实际应用场合的推广,是这几年来业界的研究热点。

在 H.264 视频编码标准正式推出之后,国内外科研人员针对其非常高的计算

复杂度开展了大量研究。在该视频编码标准推出的初期,对其研究主要是介绍性、综述性和验证性的简单工作。之后,人们更加关注对具体算法复杂度优化方面的研究。在国际上,ISO/IEC 与 ITU-T 推出了多个版本的 H.264 参考模型,从早期的 TML 系列到现在的 JM 系列,都在不断的采用新算法对进行改进。目前国内外有较多的参考模型,如:JM 平台、T264、X264 等,科研人员在各种平台上进行了大量的优化,在具体算法方面也提出了很多改进的快速估计算法,其中部分先进的快速估计算法已被具体平台采纳吸收。

2003 年,Michael Horowitz^[11]等人研究了 H.264 解码器的复杂度,并探讨了解码器的实现原理。2007 年 Hanli Wang^[12]等人,2008 年 B.G Kim^[13]等人对多帧预测选择技术进行深入研究,提出改进算法。2008 年 Lai-Man Po^[14]等人研究了帧内预测模式选择技术,提出改进算法。E.Q.Li^[15]等人对使用处理器实现 H.264 编解码优化进行研究。郑琳俊^[16]等人对多线程实现 H.264 编解码进行研究。

国内外研究工作主要集中在两个方面^[17]:一方面是对 H.264 的参考模型,如:JM 模型,进行算法级的改进,另一方面是对参考模型进行代码或者处理器指令级的改进。这两方面的研究工作,最终目的均旨在不降低视频编码质量的前提下,减少算法的计算复杂度,提升编码效率,为多媒体视频的压缩处理应用的推广作出贡献。

1.4 本文结构安排

本论文对 H.264 编解码原理和关键算法进行了研究,重点对其快速运动估计的块匹配准则和模式选择原理开展深入研究,并通过软件仿真,在 JM 平台上实现了国内外研究人员提出的四种典型的运动估计算法,进行仿真比较。之后,学习研究 Intel 公司提出的 SSE 指令,并采用 SSE4 指令集在 JM8.6 平台上对快速运动估计算法部分采用代码和处理器指令集的优化改进,在没有显著降低视频图像质量和增加传输码率的基础上,显著减少了编码时间,提高了编码效率。具体内容安排如下:

第 1 章综合论述了本课题研究背景,视频编码标准简史,介绍了相关视频编码标准,引出 H.264 视频编码标准,并论述了本课题的研究现状。

第 2 章首先阐述了 H.264 的视频编码框架和关键技术,接着研究快速运动估计中宏块匹配准则和模式选择原理,然后研究国内外四种典型的快速运动估计算法:全搜索算法(Full Search)、EPZS 算法、UMHexagonS 算法及简化

UMHexagonS 算法,最后在 JVT 的 JM10.2 平台上进行仿真,对这四个典型运动估计算法进行全面的比较分析。

第 3 章研究 SSE 指令集,首先阐述了 SSE 指令集的发展历史,接着重点研究 SSE4 指令集中新引进的适合快速计算 SAD 的指令 MPSADBW 和 PHMINPOSUW,分析使用 SSE4 指令进行快速运动估计优化的可行性,并给出部分代码,论述了具体实现快速计算 SAD 的方法。

第 4 章运用 SSE4 多媒体指令集进行运动估计优化,首先分析了使用 SSE4 指令优化算法的原理及实现步骤,在 JM8.6 平台上进行优化设计,并通过软件仿真,实现快速运动估计优化,接着分析了本文实验环境,最后对实验结果进行了全面具体的分析研究,给出结论。

第 5 章对本文的工作进行了总结,对未来的学习进行了展望。

第 2 章 H.264 快速运动估计算法研究与实验仿真

H.264 基于混合编码系统，这与之前的 MPEG 系列、H.26X 系列视频编码标准类似。不同的是，H.264 还具有更高编码效率、高质量视频画面、良好网络适应性、多模式运动估计等多选择的编码方式、更强错误恢复功能及较高复杂度等特点。

2.1 H.264 视频编解码框架及关键技术

H.264 并没有明确的规定编解码器具体如何去实现，而是对编码后的视频比特流的句法给出限定，并规定了编码后的视频流的解码方法。限定了编解码器的框架后，各个厂商的产品可以在此框架下兼容胡同，不仅在实现上更具灵活性，而且更有利于竞争优化。

H.264 采用的是变换和预测的混合编码方法，其编码器的框架图及功能组成如图 2-1^[18]所示。

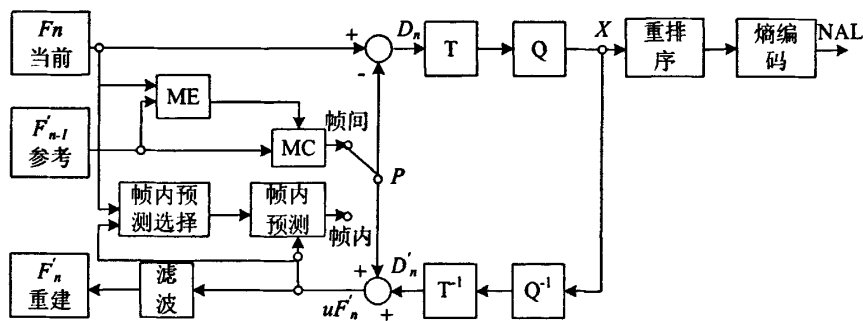


图 2-1 H.264 编码器

图 2-1 中， F_n 表示输入的场或帧， F_n 是以宏块为单位的。编码器首先选择帧间预测或者帧内预测的编码方式进行处理。

如果选择帧内预测编码，其预测值 P 是由当前帧中前面已经编码的参考图像经过运动补偿(MC)之后得出，参考图像如图 2-1 用 F'_{n-1} 表示，实际的参考图像可以在上一帧或者下一帧已编解码重建和滤波的帧中选择，这样可以提高压缩率。

D_n 表示残差帧，是预测帧 P 和当前帧相减后的差值，经过快变换 T 、量化 Q 之后产生 X 序列，再进行重排序、熵编码，与解码所需的一些其它信息，如：量化参数、运动矢量、预测模式等信息，组成一组压缩后的码流，经过网络自适应层 NAL 传输出去，或者存储下来。

上述中提到，参考图像可以是重建滤波后的帧图像，因此，编码器还需具备重建图像的功能。则如图 2-1 所示，编码器将量化后的残差图像经过反量化、反变换后得到 D'_n 与预测值 P 相加，这样就得到 uF'_n ，经过环路滤波器去除编码解码环路中产生的噪声，从而得到高质量的参考帧 F'_n 。

H.264 解码器的框架图及功能组成如图 2-2 所示。

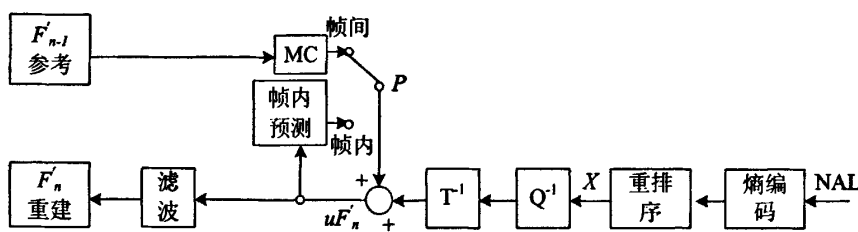


图 2-2 H.264 解码器

由图 2-1 可知，编码器的 NAL 最后输出一个压缩后的 H.264 码流，图 2-2 中，将从 NAL 获得的码流经过熵编码、重排序后得到一组序列 X ，再经过反量化、反变换得到残差帧 D'_n 。将 P 与残差块 D'_n 相加，产生信息 uF'_n ，在经过环路滤波器滤波，去除噪声，就得到最后的解码帧 F'_n 。

H.264 与之前的编码标准相比，引入了许多先进编码技术，其中，几个主要的关键技术有^[19]：

(1) 帧内预测编码

帧内编码即在同一帧图像中预测编码，该技术可以减少图像的空间冗余度。H.264 编码器选择帧内预测编码后，在给定的当前帧内充分利用相邻宏块的空间相关性，这是由于一幅图像中，相邻的像素相邻的宏块通常相似，这样，在对给定的宏块编码时，可以根据周围的宏块进行预测。通常选用左上角的宏块进行预测，因为左上角宏块一般已经被编码处理。在获取相邻宏块作为预测值后，再与实际值相减，得到残差值，再对残差值进行压缩编码。由于残差值相对于当前帧的实际值很小，因此，可以大大减少编码器的计算量，也极大的减少了编码码率。

预测块 P 是基于以编码重建块和当前块形成的。 P 块可用于亮度子块和色度

子块的相关操作。其中亮度块包括 4×4 子块和 16×16 宏块, 4×4 亮度子块有 9 种可选预测模式, 独立预测每一个 4×4 亮度子块, 16×16 亮度块有 4 种预测模式, 预测 16×16 亮度块。 4×4 亮度子块区域小, 适用于图像纹理复杂的图像编码, 而 16×16 亮度块相对区域大, 适用于图像细节较少的图像编码。色度块的预测模式类似于 16×16 亮度块, 有 4 种预测模式。编码器在帧内预测时, 会计算 P 块和编码当前块的差值, 差值最小的模式最为预测模式。

(2) 帧间预测编码

帧间预测编码及在不同帧图像之间预测编码, 该技术利用连续图像之间具有的时间冗余性, 对编码帧进行运动估计与补偿。H.264 的运动补偿不仅支持以往的 MPEG 系列、H.26X 系列视频编码标准中的大部分关键特性, 而且添加了更多的功能。除了支持 P 帧、B 帧外, H.264 还支持一种新的流间传送帧——SP 帧。码流中包含 SP 帧后, 能在有类似内容但有不同码率的码流之间快速切换, 同时支持随机接入和快速回放模式。

H.264 帧间预测是利用已编码视频帧/场和基于块的运动补偿的预测模式。与以往标准帧间预测的区别在于块尺寸范围更广(从 16×16 到 4×4)、亚像素运动矢量的使用(亮度采用 $1/4$ 像素精度 MV)及多参考帧的运用等等。

(3) 整数变换

在变换方面, H.264 使用了基于 4×4 像素块的类似于 DCT 的变换, 但使用的是以整数为基础的空间变换, 不存在反变换, 因为取舍存在误差的问题。与浮点运算相比, 整数 DCT 变换会引起一些额外的误差, 但因为 DCT 变换后的量化也存在量化误差, 与之相比, 整数 DCT 变换引起的量化误差影响并不大。此外, 整数 DCT 变换还具有减少运算量和复杂度, 有利于向定点 DSP 移植的优点。

(4) 量化

H.264 中可选 32 种不同的量化步长, 这与 H.263 中有 31 个量化步长很相似, 但是在 H.264 中步长是以 12.5% 的复合率递进的, 而不是一个固定常数。

在 H.264 中, 变换系数的读出方式也有两种: 之字形(Zigzag)扫描和双扫描。大多数情况下使用简单的之字形扫描; 双扫描仅用于使用较小量化级的块内, 有助于提高编码效率。

(5) 熵编码

视频编码处理的最后一步就是熵编码, 在 H.264 中采用了两种不同的熵编码方法: 基于上下文自适应的可变长编码(CAVLC)和基于上下文的自适应二进制算

术熵编码(CABAC)。

2.2 宏块匹配运动估计算法研究

宏块匹配运动估计(BME, Block-Matching Motion Estimation)算法的基本思想是将图像序列的每一帧分成许多互不重叠的宏块,并认为宏块内所有像素的位移量都相同,然后对每个宏块到参考帧某一给定的搜索范围内根据一定的匹配准则找出与当前块最相似的块,即匹配块,匹配块与当前块的相对位移即为运动矢量(MV, Motion Vector),得到运动矢量的过程被称为运动估计(ME, Motion Estimation)。视频压缩的时候,只需保存运动矢量和残差数据就可以完全恢复出当前块^[20]。如图 2-3 所示。

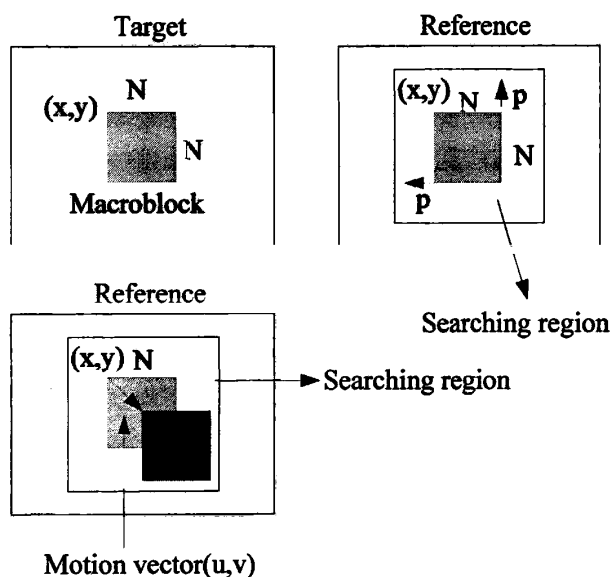


图 2-3 宏块匹配运动估计原理

通过运动估计可以去除帧间冗余度,使得视频传输的比特数大为减少,因此,运动估计是视频编码的帧间预测中的一项关键技术^[21]。目前在 H.264 视频编码中,普遍采用块匹配运动估计技术^[22]。

2.2.1 块匹配准则

运动搜索的目的就是在搜索窗内寻找与当前块最匹配的数据块^[23],这样就存在着如何判断两个块是否匹配的问题,即如何定义一个匹配准则。而匹配准

则的定义与运算复杂度和编码效率都是直接相关的。目前常用的匹配准则函数有：平均绝对误差(MAD, Mean Absolute Difference)函数、绝对误差和(SAD, Sum of Absolute Difference)函数、均方误差(MSE, Mean Square Error)函数与归一化互相关(NCCF, Normalized Cross Correlation)函数^[24-26]。

如下所示，式中 (i, j) 为位移矢量， f_k 和 f_{k-1} 分别为当前帧和前一帧的像素值， $M \times N$ 为宏块大小，若在某一个点 (i_0, j_0) 处使以下函数值达到最小，则该点即为搜索的最佳匹配点。

(1) 平均绝对误差函数(MAD)

$$MAD(i, j) = \frac{1}{MN} \sum_{m=1}^M \sum_{n=1}^N |f_k(m, n) - f_{k-1}(m+i, n+j)| \quad (2-1)$$

MAD 到达最小值的点即为最佳匹配点。

(2) 绝对误差和函数(SAD)

$$SAD(i, j) = \sum_{m=1}^M \sum_{n=1}^N |f_k(m, n) - f_{k-1}(m+i, n+j)| \quad (2-2)$$

SAD 到达最小值的点即为最佳匹配点。

(3) 均方误差函数(MSE)

$$MSE(i, j) = \frac{1}{MN} \sum_{m=1}^M \sum_{n=1}^N [f_k(m, n) - f_{k-1}(m+i, n+j)]^2 \quad (2-3)$$

MSE 到达最小值的点即为最佳匹配点。

(4) 归一化互相关函数(NCCF)

$$NCCF(i, j) = \frac{\sum_{m=1}^M \sum_{n=1}^N f_k(m, n) f_{k-1}(m+i, n+j)}{\left[\sum_{m=1}^M \sum_{n=1}^N f_k^2(m, n) \right]^{1/2} \left[\sum_{m=1}^M \sum_{n=1}^N f_{k-1}^2(m, n) \right]^{1/2}} \quad (2-4)$$

NCCF 到达最小值的点即为最佳匹配点。

由于 MAD 函数没有乘除操作，不需做乘法运算，实现简单方便，所以使用较多。通常使用求和绝对误差(SAD)代替 MAD。

在参考模型 JM(Joint Model)H.264 中定义的匹配误差函数^[27-32]如下：

$$J(MV, \lambda_{MOTION}) = SAD(s, c(MV)) + \lambda_{MOTION} \times R(MV - PMV) \quad (2-5)$$

其中 SAD(绝对差值和)计算公式如下：

$$SAD(s, c(\overline{MV})) = \sum_{x=1, y=1}^{B_x, B_y} |s[x, y] - c[x - MV_x, y - MV_y]|, B_x, B_y = 16, 8 \text{ or } 4 \quad (2-6)$$

其中 s 是当前进行编码的原始数据, 而 c 是已经编码重建的用于进行运动补偿的参考帧的数据。 MV 为候选的运动矢量, λ_{MOTION} 为拉格朗日常数, PMV 为中值预测矢量, $R(MV-PMV)$ 代表了运动矢量差分编码可能耗费的比特数, 由于在匹配误差预测方式中匹配误差中的 $\lambda_{MOTION} \times R(MV-PMV)$ 部分通常很接近而抵消, SAD 部分的预测特性基本上可以反映整个匹配函数的预测特性, 因此 $J(MV, \lambda_{MOTION})$ 用 SAD 来表示。

在用块匹配算法进行运动估计的过程中, 利用匹配准则函数进行匹配误差的计算是最主要的计算量, 因此, 可以从这方面进一步减少计算量或计算时间。在第 3 章将详细阐述如何利用 SSE4 指令集减少 SAD 的计算时间。

2.2.2 模式选择原理

H.264 的最佳编码模式选择, 就是在所允许的编码模式中寻找式 2-7 达到最小值的子块划分模式:

$$J(s, c, MODE | QP, \lambda_{MODE}) = SSD(s, c, MODE | QP) + \lambda_{MODE} \cdot R(s, c, MODE | QP) \quad (2-7)$$

式中, QP 是宏块的量化参数, λ_{MODE} 是拉格朗日(Lagrange)系数, SSD 表示源信号 s 与重建信号 c 差的平方和, $MODE$ 表示各种编码模式集合:

I slice: $MODE \in [9 \text{种 Intra_} 4 \times 4 \text{模式}, 4 \text{种 Intra_} 16 \times 16 \text{模式}]$

P slice: $MODE \in [InterMode \text{模式}, SKIP \text{模式}]$

B slice: $MODE \in [InterMode \text{模式}, DIRECT \text{模式}]$

而 $R(s, c, MODE | QP)$ 是与模式和量化参数有关的比特开销, 包括宏块头信息、运动矢量和所有 DCT 块信息的总的二进制位数, 它是通过对块进行实际的编码后获得的, 所以其运算量相当大。

2.3 典型运动估计算法研究

搜索策略是运动估计算法中最为复杂的部分。不同的搜索算法的选择将直接影响编码速度, 运动估计的准确度以及图像质量等。本节就对几种典型的运动估计算法进行研究。

2.3.1 全搜索算法

全搜索算法(FS, Full Search)也成为穷尽搜索法,顾名思义,全搜索算法就是对搜索区域的所有位置进行穷尽搜索,计算当前宏块位置(0, 0)周围 $\pm M$ 个像素范围内每个点的 SAD 值^[33-34],并对参考帧中所有的候选块进行比较,找出具有最小匹配误差的一个,即在搜索窗中找出 SAD 值最小的点,该点就是最佳匹配点,该点与当前帧内编码块之间的位移就是最佳运动矢量(MV)。这样需要计算 $(2M+1)^2$ 个位置的 SAD 值,一般选用 $M=16$,即需要计算 33×33 个位置的 SAD。

全搜索算法一般采用光栅搜索顺序和螺旋搜索顺序。如图 2-4 所示。

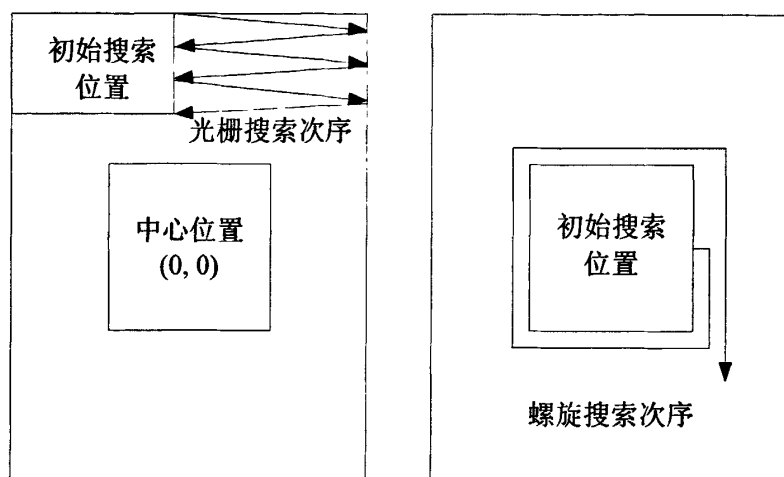


图 2-4 全搜索算法的两种搜索顺序

图 2-4 中左图为光栅搜索顺序,搜索的起始位置在搜索窗的左上角,然后按照光栅搜索的顺序计算搜索窗内的每一个点。右图为螺旋搜索顺序,搜索的起始位置在搜索窗的中心,然后按螺旋逐步向外发散的顺序由内到外逐点计算。实验证明,在视频编码中,运动矢量集中在零矢量点及其附近的概率很大,所以如果与提前退出算法结合,螺旋搜索顺序是全搜索算法中计算量最小,一般采用螺旋搜索。

全局搜索法思想简单,对搜索窗口内所有的点进行搜索,是最优的运动估计算法,也是计算量最大的算法,不适合进行实时编码。为了加快编码的速度,发展出了很多快速块匹配搜索算法。

全局搜索算法精度最高,算法简单,但其计算复杂度最大,难以实时编码。近些年来研究人员提出了很多改进的快速运动估计算法,部分经典算法已被 JVT

采纳。如 EPZS(Enhanced Predictive Zonal Search) 算法、UMHexagonS(Unsymmetrical- cross Multi- Hexagon- grid Search)算法、Simplified UMHexagonS 算法等。

2.3.2 EPZS 算法

EPZS(Enhanced Predictive Zonal Search) 是一种预测搜索算法^[35], 该算法选择更多的预测矢量和多模式搜索路径, 力求从几个非常可能的矢量中预测最佳运动矢量。

(1) 运动矢量选择

预测点的选择对算法的性能非常关键, 精确选择对运动估计性能有重要影响的预测矢量是 EPZS 的主要特征。预测矢量主要根据序列的相关性来选择, 包括空间和时间的相关性。在搜索窗内的一些固定位置, EPZS 定义了一个矢量集 S , 按照重要性将其分成 A 、 B 、 C 等 3 个子集。其中:

$A = \{\text{Median Predictor}\}$ 为中值矢量, 在预测矢量中是最重要的预测值;

$B = \{\text{Collocated Block MV, Left MV, Top MV, Top-left MV, Top-right MV, (0, 0)}\}$, 包括与时域相邻的同位置块的运动矢量和相邻块的矢量以及 $(0, 0)$;

$C = \{\text{Temporal scaled MV, Accelerator MV, Collocated neighbor MV, Reference scaled MV, Inter-mode MV, Sampled pixels}\}$, 考虑了在较大运动的情况下可能的预测矢量。引入附加预测矢量会增加运动估计过程的复杂度, 因此在算法中引入一种称之为“三阶段处理程序”的过程:

Step1: 在检测完子集 A 中的预测块后, 判断当前块的最小绝对误差和(SAD)是否小于或等于阈值 $T1$, 如是则终止搜索, 否则继续;

Step2: 在检测完子集 B 中的预测块后, 如果当前块的最小 SAD 小于或等于阈值 $T2$ 则终止, 否则继续;

Step3: 在检测完子集 C 中的预测块后, 如果当前块的最小 SAD 小于或等于阈值 $T3$ 则终止, 否则用更精细的模式搜索。

(2) 搜索模板

EPZS 算法使用小钻石(DS)、正方形 EPZS($EPZS^2$)和扩展 EPZS(extEPZS)等 3 种搜索模板, 如图 2-5 所示。

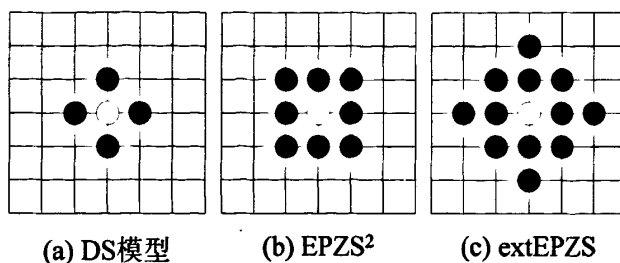


图 2-5 EPZS 的三种搜索模板

由于 DS 在处理相对复杂的图像序列时很容易陷入局部最优，因此 EPZS 以最佳预测矢量为中心按扩展 EPZS(extEPZS)搜索和以中值预测矢量 MVP 为中心按 EPZS² 搜索的双模式搜索。

(3) 算法分析

EPZS 算法由局部开始逼近最佳点，该算法虽然可以较好地避免落入局部最优，但是仍有其局限性，即对于搜索范围比较小，运动比较平缓的情况，效果较好，但在搜索范围较大和运动剧烈的情况下，很容易在搜索初期就落入局部最优。因此如何避免落入局部最优，在较大范围内找到最优的匹配点，仍然是运动估计研究的重点。

2.3.3 UMHexagonS 算法

H.264 于 2003 年 3 月采纳了“非对称十字型多层次六边形格点搜索算法”(UMHexagonS Algorithm)^[36]，该算法多种搜索算法相组合，在很大程度上提高了预测的有效性和健壮性，具有明显优于其他快速算法的率失真性能。

其搜索模板如图 2-6 所示。

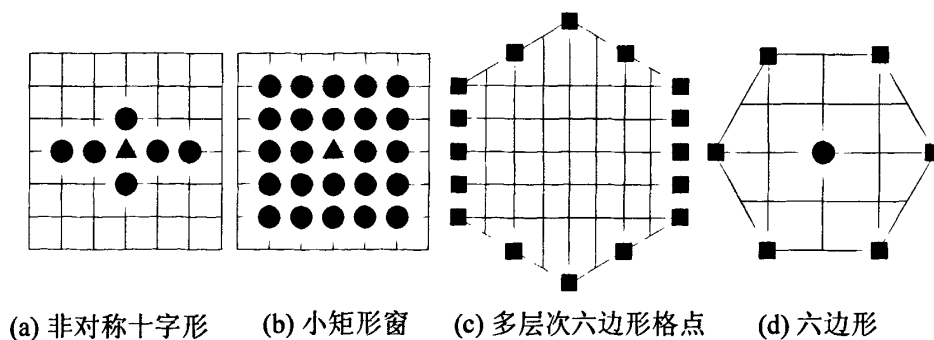


图 2-6 UMHExagonS 算法中的搜索模板

图 2-7 给出了 UMHExagonS 算法的搜索过程，如图可以清楚了解搜索步骤

及各步采用的搜索模板。

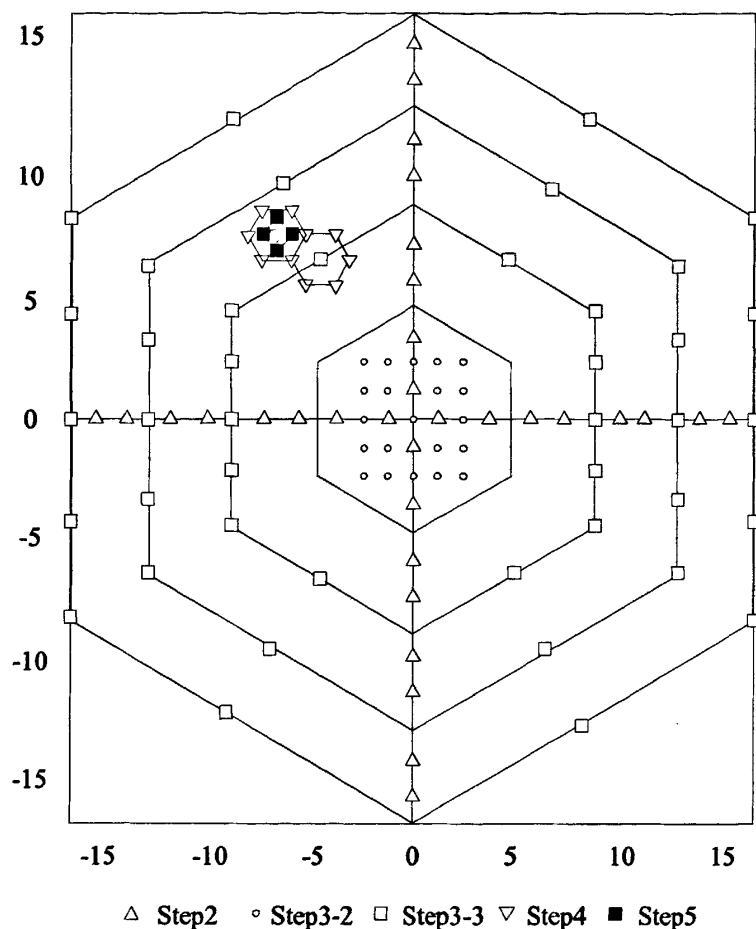


图 2-7 UMHexagonS 算法搜索原理图

(1) 算法实现步骤:

Step 1: 预测矢量选择。将搜索起电设置点在最小 SAD 点。

Step2: 不对称十字形搜索(图 2-6(a)所示)。获得当前的最佳预测起点。随后根据不同起始点的率失真结果使用域值进行判断, 分为不满意区域、满意区域和很满意区域, 分别转入 Step3, Step4, Step5。

Step3: 对于不满意的块使用如下方式进行搜索:

Step3-1: 以目前最优点为中心, 用非对称十字型搜索模板(图 2-6(a) 所示)进行搜索, 获得当前最佳点, 判断此处是否属于满意或很满意区, 跳到相应的

Step3 或 Step4, 或继续搜索;

Step3-2: 以目前最佳点为中心, 在 $(-2, 2)$ 的方形区域中逐点搜索(图 2.6 (b)所示), 获得当前最佳点, 判断此处是否属于满意或很满意区, 跳到相应的 Step3 或 Step4, 或继续搜索;

Step3-3: 用不断扩大一倍直径的大六边形模板(图 2-6 (c)所示)进行搜索, 直至搜索到能符合相应阈值而进入 Step3 和 Step4 的搜索点为止; 或者搜索模板完全超出搜索窗范围, 也结束 Step2 的搜索。

Step4: 以目前最佳点为中心, 使用六边形搜索模型(图 2-6 (d)所示)进行搜索, 直至最佳点在六边形中心为止。

Step5: 对很满意区域的子块进行搜索。以当前最佳点为中心, 用十字型模板进行搜索, 直至最佳点在十字型模板的中心点为止。

(2) 算法分析

UMHexagonS 算法的起点预测准确, 同时该算法采用了内容自适应的搜索方式, 且模板能对不同内容的块进行不同的搜索, 因此具有较高的编码性能。EPZS 算法搜索点较少, 但与 UMHexagonS 算法相比在预测和判断计算上花费了较多的运算时间; UMHexagonS 算法搜索点较多, 计算复杂度相应提高。

2.3.4 简化 UMHexagonS 算法

UMHexagonS 算法需要消耗大量内存用于存储 SAD 和 MV, 为了改进 UMHexagonS 算法, 2005 年 7 月, JVT 的第 16 次会议上又提出了一种“简化 UMHexagonS 算法”(Simplified UMHexagonS Algorithm)^[37], 在保持相同甚至更佳的率失真性能的同时, 该算法的运算量相对于 UMHexagonS 算法可节约 46%~57%。新采纳的简化 UMHexagonS 进行了三大简化和改进^[38]:

(1) 低存储的初始预测点集方案

UMHexagonS 算法的初始预测点集中包括空间预测(中值预测、原点预测和 uplayer 预测)、时间预测(相关块预测、相邻参考帧预测)。其中的时间预测消耗了大量的存储空间。而简化 UMHexagonS 算法使用了一种低存储的初始预测方案, 仅仅使用空间预测点, 因此可以大大节省存储空间。

(2) 简化的提前终止判断

UMHexagonS 算法为了获得代表不同运动情况的判断阈值(为浮点数), 必须根据不同的 SAD 预测值按公式在每一个搜索点进行复杂的运算。而简化

UMHexagonS 算法使用了简化的早期终止阈值判断技术:

由经验得到的整数型常数作为基值;对不同尺寸的块类型,采用对基值进行移位操作,以作为不同的判断阈值。其判别运动类型的条件简化为两类:

①会聚(convergence)条件:若满足会聚条件就停止搜索直接跳到会聚步骤;

②加强搜索(intensive search)条件:若满足加强搜索条件,则进行加强搜索以避免搜索到的是局部最优。因此简化 UMHexagonS 算法可更进一步减少搜索时间和存储空间。

(3) 取消局部全搜索

在简化 UMHexagonS 算法里,不再采用局部全搜索,因此可节省大量搜索时间。

2.4 几种常见的快速运动估计算法仿真与性能比较

为了比较以上四种快匹配运动估计算法的性能,对标准视频序列进行了对比试验。实验的硬件平台是 CPU: Intel 双核 T6670, 2.2GHz, 内存: 1GB。实验软件平台 MS Win7 系统, MS VS2008 SP1, JVT 的 JM10.2。实验选用的视频序列为 foreman_part_qcif.yuv, 实验设定参数为: 30 帧/秒, 视频序列为 IBP 格式, 量化系数为 QPI: 28, QPP: 28, QPB30, Hadamard 变换, 搜索范围为 16 个像素点, 5 个参考帧, 采用 CABAC 熵编码。具体实验参数如表 2-1 所示。

表 2-1 实验参数

encoding frames	3	Format	174×144
R-D optimization	enabled	Coding	CABAC
QPI/QPP/QPB	28/28/30	Search range of MV	±16
GOP structure	IPB	No. of Ref. Frames	5
Profile	Main	Hadamard Transform	Enabled
Intra upd	OFF	Frequence	30Hz

设置以上实验参数,分别采用 FS 算法、EPZS 算法、UMHexgonS 算法及简化 UMHexgonS 算法对 foreman_part_qcif.yuv 视频序列进行编码,比较各快速运动估计搜索算法的运动估计时间、总编码时间、编码总比特率以及编码后图像的 Y/U/V 分量的峰值信噪比(PSNR)。记录实验数据如表 2-2 所示。

表 2-2 实验结果(QPI/QPP/QPB 分别为 28/28/30)

快速运动估计 搜索算法	SNRY (dB)	SNRU (dB)	SNRV (dB)	Total Bits (bits)	ME Time (ms)	Total Time (ms)
FS	36.756	41.083	42.563	31824	1410	3070
EPZS	36.774	41.113	42.614	31968	490	2120
UMHexagonS	36.808	41.121	42.595	32056	475	2115
Simplified UMHexagonS	36.805	41.080	42.620	31832	325	1960

为了方便比较以上四种快速运动估计搜索算法的编码时间,将表格 2-2 中的运动估计时间(ME Time)、总编码时间(Total Time)表示成柱状图,如图 2-8 所示。由图 2-8 可知,EPZS 算法、UMHexagonS 算法及简化 UMHexagonS 算法在降低时间复杂度方面有显著的效果,三种算法的运动估计时间(ME Time)、总编码时间(Total Time)明显低于全搜索算法,而码率相对于全搜索算法均有小幅度提高。其中 EPZS 算法和 UMHexagonS 算法性能相当。UMHexagonS 算法的运动估计时间和总编码时间略低于 EPZS 算法,但是 UMHexagonS 算法的码率提高最多。而简化 UMHexagonS 算法由于采用了更有效的初始预测点集,增加早停止判断,适当地减少了 2.3.3 节中的 Step2 和 Step3 的搜索点,因此,简化 UMHexagonS 算法的运动估计时间能在 UMHexagonS 算法基础上进一步大大减少。

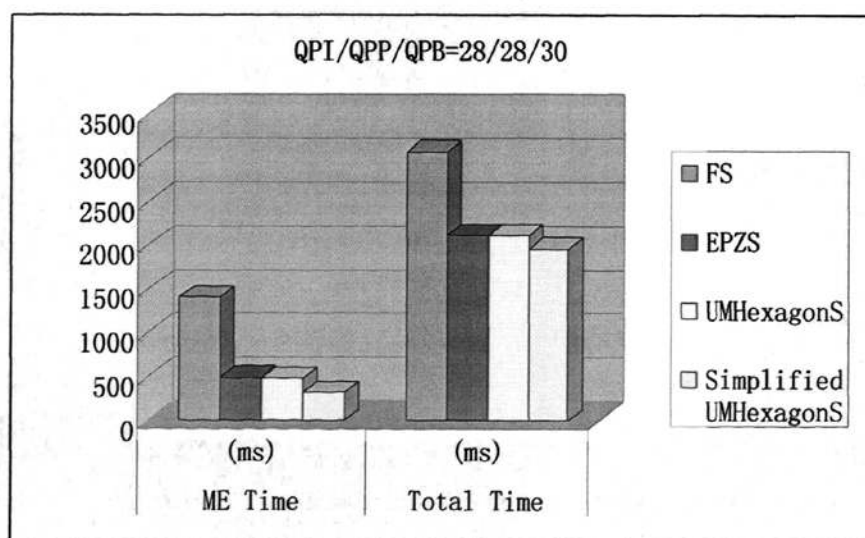


图 2-8 典型算法运动估计时间、编码时间比较

这里引入几个衡量算法性能的指标，分别是峰值信噪比增量（ $\Delta PSNR$ ），码率增量（ $\Delta Bits$ ）以及编码时间节省因子（ ΔT ）。它们的定义如式 2-8 至 2-10 所示。

$$\Delta PSNR = PSNR_{\text{optimized}} - PSNR_{\text{standard}} \quad (2-8)$$

$$\Delta Bits = \frac{Bits_{\text{optimized}} - Bits_{\text{standard}}}{Bits_{\text{standard}}} \times 100\% \quad (2-9)$$

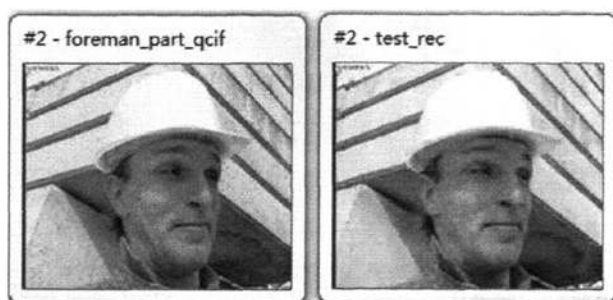
$$\Delta T = \frac{Time_{\text{optimized}} - Time_{\text{standard}}}{Time_{\text{standard}}} \times 100\% \quad (2-10)$$

式 2-8 至 2-10 中，*optimized* 是指优化的算法，*standard* 是只原始算法，这里，将 EPZS 算法、UMHexagonS 算法及简化 UMHexagonS 算法与全搜索算法进行对比，分别算出优化的三种算法相对于全搜索算法的峰值信噪比增量（ $\Delta PSNR$ ），码率增量（ $\Delta Bits$ ）以及编码时间节省因子（ ΔT ）。则由表 2-2 可以得到表 2-3 所示数据。

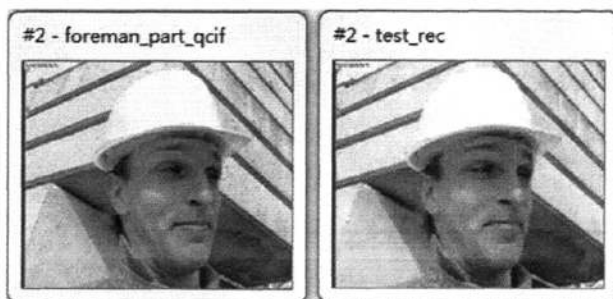
表 2-3 EPZS 算法、UMHexagonS 算法及简化 UMHexagonS 算法性能比较

快速运动估计 搜索算法	$\Delta SNRY$ (dB)	$\Delta SNRU$ (dB)	$\Delta SNRV$ (dB)	$\Delta Bits$ (%)	$\Delta METime$ (%)	$\Delta Total Time$ (%)
EPZS	0.018	0.03	0.051	0.45	-65.25	-30.94
UMHexagonS	0.052	0.038	0.032	0.73	-66.31	-31.11
Simplified UMHexagonS	0.049	-0.003	0.057	0.03	-76.95	-36.16

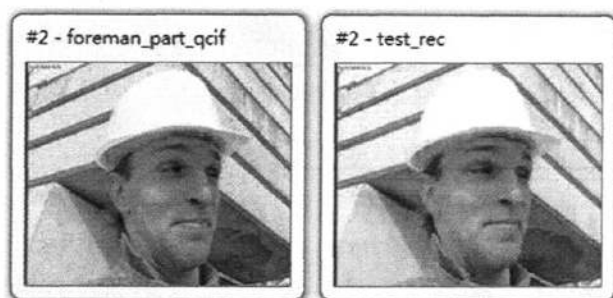
由表 2-3 可以得出，EPZS 算法、UMHexagonS 算法及简化 UMHexagonS 算法运动估计时间节省因子很大，相对于全搜索算法，能够节省运动估计时间 65~77%，而编码时间可以节省约 30~36%。*PSNR* 值几乎没有降低，也就是说图像的清晰度几乎没有下降。三种算法都稍稍增加的码率，但并不明显。图 2-9 所示，每幅图的左侧均为压缩前的 *foreman_part_qcif.yuv* 文件中第二帧图像，每幅图的右侧分别为用全搜索算法、EPZS 算法、UMHexagonS 算法及简化 UMHexagonS 算法对标准视频 *foreman_part_qcif.yuv* 文件进行编码后的第二帧图像。通过对比，可以看到用这四种算法压缩后的图像与原图像相似度很高，用肉眼机会观察不出区别。



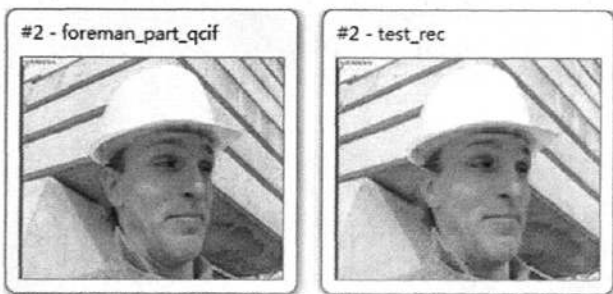
(a) 全搜索算法



(b) EPZS 算法



(c) UMHexasS 算法



(d) 简化 UMHexasS 算法

图 2-9 四种算法压缩前后 foreman 的第二帧图像对比

因此,可以得出结论:全搜索算法、EPZS 算法、UMHexagonS 算法及简化 UMHexagonS 算法四种快速运动估计算法对视频压缩后,图像质量比较高,只会稍稍的增加编码后的码率,影响不大。全搜索算法精度最高,思想简单,但是由于每个点都要计算,时间复杂度比较高。而改进后的三种算法在不显著降低图像质量并且不显著增加码率的条件下,可以显著的提高编码速度。其中 EPZS 算法和 UMHexagonS 算法性能相当,UMHexagonS 算法的运动估计时间和总编码时间略低于 EPZS 算法,码率略高于 EPZS 算法。简化 UMHexagonS 算法由于采用了更有效的初始预测点集,增加早停止判断,适当地减少了搜索点,因此,简化 UMHexagonS 算法的运动估计时间能在 UMHexagonS 算法基础上进一步大大减少,且码率比 UMHexagonS 算法低。

2.5 本章小结

本章首先阐述了 H.264 视频编解码框架及关键技术,然后分析了宏块匹配运动估计算法的原理,块匹配准则及模式选择原理等。接着分析了四种典型的运动估计算法:全搜索算法、EPZS 算法、UMHexagonS 算法及简化 UMHexagonS 算法的搜索原理,搜索模板,步骤等等。最后,在 JVT 的 JM10.2 平台上,通过软件仿真,实现以上四种典型运动估计算法对 foreman_part_qcif.yuv 视频文件的压缩,实验结果表明,四种算法均能够在不显著降低图像质量及增加码率的条件下实现视频的压缩。其中全搜索算法思想最简单,精确度最高。而 EPZS 算法、UMHexagonS 算法及简化 UMHexagonS 算法在全搜索算法基础上能够显著提高编码速度,同等条件下,简化 UMHexagonS 算法性能最优。

第3章 用 SSE4 指令集实现运动估计优化

3.1 SSE 指令集简介

SSE (Streaming SIMD Extensions, 单指令多数据流扩展) 指令集是 Intel 在 Pentium III 处理器中率先推出的。SSE4 指令集并不能完全说是一个独立的新技术, 因为它是集成在英特尔 45 纳米处理器之中, 可以说是处理器众多新技术之一。

首先还是回顾一下英特尔处理器集成指令的历史, 之后再引出 SSE4 指令集的诞生。自英特尔奔腾 MMX(Multi Media Extension, 多媒体增强指令集)处理器开始, 处理器新加入了 SIMD (Single Instruction Multiple Data) 多媒体指令集。该指令集可以把多批次的指令组编辑成为一条单一的指令, 从而达到提升数据处理的能力。集成 MMX 指令的奔腾处理器主要用作提升多媒体数据的处理能力, 共有 57 条指令。

英特尔公司于 1999 年发布了基于 MMX 指令的 SSE 指令集。首颗支持 SSE 产品 Pentium III 处理器, 除新增 70 条指令之外, 还进一步提升了多媒体数据的处理能力, 最重要的是解决了 MMX 指令与浮点指令不能同时处理的问题。

于 2001 年发布了 SSE2 指令集, 又在原来的基础上增加了 144 条新指令。其中除了主要负责 64 位双精度浮点数及整型运算和对 Cache 控制延迟降低两分之外, 更重要的是完全解决了 SSE 指令集需要占用浮点数据暂存器问题。

2004 年, 以 Prescott 为核心的英特尔奔腾 4 处理器加入了 SSE3 指令集, 新增指令仅 13 条, 主要是对水平式暂存器整数的运算, 可对多笔数值同时进行加法或减法运算, 令处理器能大量执行 DSP 及 3D 性质的运算。此外, SSE3 更针对多线程应用进行最佳化, 使处理器原有的 Hyper-Threading 功能获得更佳发挥。

作为 SSE3 指令集的补充版本, 2006 年 SSSE3 出现在人们已经相对比较熟悉的酷睿微架构处理器中, 新增有 32 条指令, 进一步增强 CPU 在多媒体、图形图像和 Internet 等方面的处理能力。而英特尔方面本来是计划将该 32 条指令收录在后来的 SSE4 指令集中, 但考虑到当时硬件升级速度的大幅提升, 最终决定提早加入至酷睿微架构产品中。

2007年11月,随着penryn(酷睿双核)处理器到来的SSE4,被视为继2001年以来最重要的多媒体指令集架构的改进,除扩展Intel 64指令集架构外,还加入有关图形、视频编码及处理、三维成像及游戏应用等指令,指令涉及音频、图像和数据压缩算法等多方面性能大幅度提升。SSE4分为4.1版本及4.2版本,4.1版本共新增47条指令,主要针对向量绘图运算、3D游戏加速、视频编码加速及协同处理加速动作。2008年1月45nm的Nehalem处理器追加了SSE4.2版本,新增7条指令,两个版本合共54条指令。如图3-1所示,为SSE指令集的发展简史^[39-41]。

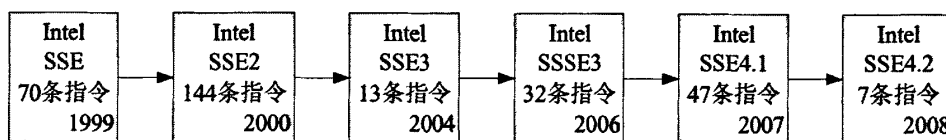


图 3-1 SSE 指令发展简史

Intel指出,加入的SSE4指令集让45nm Penryn处理器增加了2个不同的32Bit向量整数乘法运算单元,并加入8位无符号(Unsigned)最小值及最大值运算,以及16Bit及32Bit有符号(Signed)运算。在面对支持SSE4指令集的软件时,可以有效的改善编译器效率及提高向量化整数及单精度代码的运算能力。同时,SSE4改良插入、提取、寻找、离散、跨步负载及存储等动作,令向量运算进一步专门化。

SSE4还计入了六条浮点运算指令,支持单精度、双精度浮点运算及浮点产生操作,可立即转换其路径模式,大大减少延误,这些支持将会在3D游戏及对浮点运算能力非常敏感的领域起到积极的效果。

SSE4指令集还加入了串流式负载指令,能够提升帧缓冲区的读取数据频宽,理论上可获取完整的快取缓存行,即每次读取64Bit而非8Bit,并可以将其保存在临时缓冲区内,让支持SSE4指令集的读取频宽效能提升最高至8倍。

SSE4指令集进一步强化编码效果,例如可同时处理8个4-byte宽度的SAD(Sums of Absolute Differences)运算,常用于新一代高清影像编码如VC.1及H.264等规格中,令视频编码速度进一步提升。这是Intel宣称双核处理器软解高清视频,同样可以获得流畅、高质量播放效果的原因。

3.2 SSE4 指令集编码优化的可行性分析

SSE4.1新增了47条指令,其中有两条指令可以有效的提高视频编码速度,

分别为 MPSADBW(Improved Sums of Absolute Differences for 4-Byte Blocks)、PHMINPOSUW(Horizontal Search)^[42-45]。

3.2.1 MPSADBW 指令

指令 MPSADBW 可以执行 8 个 4-byte 宽的 SAD(Sums of Absolute Differences)运算。与 PSADBW(8-byte)指令相比, MPSADBW 指令处理更小的模块(4-byte)。这使得 MPSADBW 指令更加适合处理视频编码,如新一代视频编码标准 VC.1 和 H.264 等。在计算 SAD 时, MPSADBW 指令计算速度是 PSADBW 的四倍,即计算时间是 PSADBW 的 1/4。这使得 MPSADBW 可以显著提高密集运动搜索操作的速度。MPSADBW 使用 128-bit 源操作数 4-byte 宽的定义域,该 4-byte 宽的定义域偏移量由两位立即数来控制。MPSADBW 会产生 8 个 16-bit 的 SAD 结果。

图 3-2 给出了 MPSADBW 指令的操作原理图。MPSADBW 可以通过输入原操作数和目的操作数的偏移量来巧妙减少运动估计(ME, motion estimation)时间,可以提高 SAD 计算的吞吐量,并且可以处理更小块的宏块。如 H.264 中的 4×4 宏块。

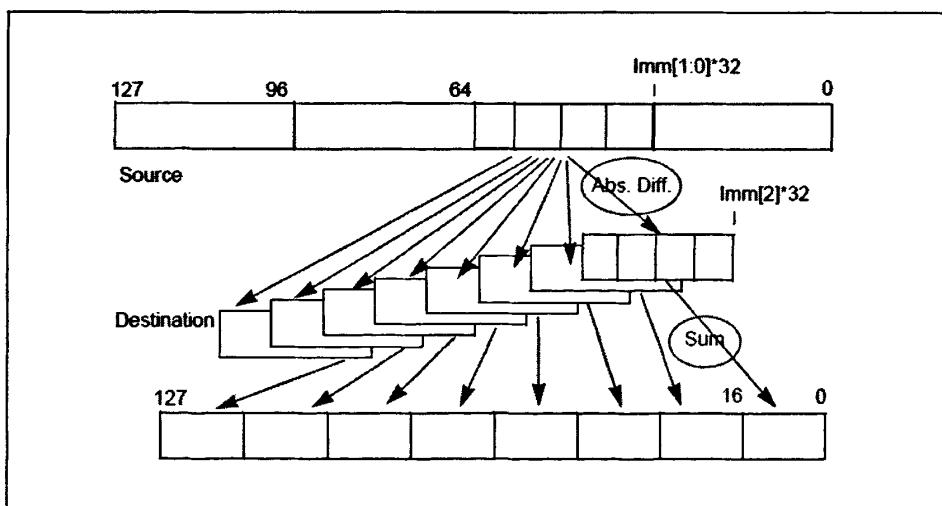


图 3-2 MPSADBW 指令操作原理图

本文就是使用这一条指令来实现 SAD 的快速计算,从而达到节省快速运动估计时间,节省编码时间的。在 Microsoft Visual Studio 2008 中, MPSADBW 对应的指令为 `_m128i_mm_mpsadbw_epu8`, 其定义如下:

```

    _m128i _mm_mpsadbw_epu8(
        _m128i a,
        _m128i b,
        const int mask
    );

```

a 为一个 128-bit 的参数, 其中包含了 16 个 8-bit 的无符号整型数。 b 也是一个 128-bit 的参数, 其中包含了 16 个 8-bit 的无符号整型数。 $mask$ 为整型常量, 用来指定计算偏移量。返回的结果为一个 128-bit 数, 其中包含 8 个 16-bit 的无符号整型数, 返回结果的计算过程等同于以下代码:

```

    i = mask2 * 4;
    j = mask0-1 * 4;
    for (k = 0; k < 8; k = k + 1) {
        t0 = abs(a[i + k + 0] - b[j + 0]);
        t1 = abs(a[i + k + 1] - b[j + 1]);
        t2 = abs(a[i + k + 2] - b[j + 2]);
        t3 = abs(a[i + k + 3] - b[j + 3]);
        r[k] = t0 + t1 + t2 + t3;
    }

```

需要注意的是, 以上 $mask0$, $mask1$, $mask2$ 分别表示 $mask$ 的后三位数字。

在 2.2.1 节快速匹配准则中, 提到了绝对误差和函数(SAD), 式 2-2 中, SAD 到达最小值的点即为最佳匹配点。按照以往的计算方法, 一个 `char` 型只能存放一个像素点, 两个像素点相减后求出绝对值再求和, 需要很多时间周期。从上述代码中可以看到, `_mm_mpsadbw_epu8` 一条指令计算出来的结果, 按照以往的计算方式, 至少需要 8 次循环, 每次循环执行 5 条指令, 一共需要执行 40 条指令, 当需要计算的宏块像素点增加时, 以往的编码方式将更加复杂, 而且容易出错。使用 SSE4 指令集中 `_mm_mpsadbw_epu8` 指令, 128-bit 参数 a , b 存入 16 个像素点的像素值, 可同时处理 8 个 4-byte 宽度的 SAD , 可以很好的简化代码, 极大的节省计算时间。

3.2.2 PHMINPOSUW 指令

PHMINPOSUW 指令是 SSE4.1 新添加的一条横向搜索指令, 如果在一个

`_m128i` 型参数 a 中存放了 8 个 16-bit 的无符号整型, PHMINPOSUW 指令可以搜索出数值最小的值, 并且将最小数值及该数值所处位置(在参数 a 中的偏移量)返回到目的寄存器中。将返回的结果保存在目的寄存器的低双字节中。

快速搜索通常是运动估计的一个重要组成部分, MPSADBW 指令和 PHMINPOSUW 指令可以同时使用, 起到更好的改进视频编码的效果。

本文在使用 MPSADBW 指令计算出宏块搜索范围内所有 SAD 值之后, 马上使用 PHMINPOSUW 指令记录下最小 SAD 值即该点所在位置, 即其运动矢量 MV。在 Microsoft Visual Studio 2008 中, PHMINPOSUW 对应的指令为 `_m128i_mm_minpos_epu16`, 其定义如下:

```
_m128i_mm_minpos_epu16(
    _m128i a
);
```

a 为 128-bit 的参数, 其中包含了 8 个 16-bit 的无符号整型数。返回的结果也是 128-bit 参数, 低 16 位保存参数 a 中最小的双字节数, 次低 16 位返回该最小数值的位置。比如: 若返回结果为 128-bit 参数 b , $a[0] \sim a[7]$ 中最小数值为 $a[3]$, 则返回结果, $b[0] = a[3]$, $b[1] = 3$ 。

综上所述, 使用 SSE4 指令集中 MPSADBW 指令可以快速计算出参考帧与当前帧之间的 SAD 值, 再使用 PHMINPOSUW 指令计算出最小 SAD 值, 及最小 SAD 值的位置。本文在 2.2.1 节中探讨过块匹配准则, 实际应用当中通常使用绝对误差和函数(SAD), 当 SAD 达到最小值时即为最佳匹配点, 则使用 SSE4 指令可以非常方便快捷的计算出最佳匹配点, 从而计算出最佳快匹配模式。

3.3 算法实现

本文在 JVT 的 JM8.6 平台上实现了算法优化, 在其基础上加入了一个 SSE4ME.c 文件, 在该文件中实现了 7 种模式的宏块 SAD 计算。具体宏块的划分将在第 4 章介绍。

下面将简要分析 SSE4ME.c 文件中实现 4×4 子宏块计算式 2-5 中 $J(MV, \lambda MOTION)$ 参数的函数。

```
int blockMatch4x4SSE4(...)
{
    ...
    __m128i s0, s1, s2, s3, s4, s5, s6;    //定义 128-bit 参数
```

```

...
for (k=0; k<blockHeight; k++)    //处理行
{
    s0 = _mm_loadu_si128((__m128i*)pRef); //参考行像素
    s1 = _mm_loadu_si128((__m128i*)pCur); //当前行像素
    s2 = _mm_adds_epu16(s2, _mm_mpsadbw_epu8(s0, s1, 0)); //对应像素 SAD A0-A7:B0-B3
    pCur+=stepBytesCB;
    pRef+=stepBytesRF;
}

s6 = _mm_minpos_epu16(s2); //s2 中 8 点内最小 SAD 位置与 SAD 值
temSum = _mm_extract_epi16(s6,0); // s2 中 8 点最小 SAD 值
k = _mm_extract_epi16(s6,1); // s2 中 8 点内最小 SAD 位置
...
}

```

上述代码中，首先使用 `_mm_mpsadbw_epu8` 计算出参考帧和当前帧 4×4 子宏块第 1 行的 *SAD*，通过循环语句，计算出第 2 行、第 3 行、第 4 行 *SAD*，并使用 `_mm_adds_epu16` 指令累加 *SAD*，结果存放在 `s2` 中，再使用 `_mm_minpos_epu16` 指令将 `s2` 中 8 位双字节数中最小值提取出来，记录最小 *SAD* 值及 *SAD* 位置。

3.4 本章小结

本章首先阐述了 SSE 指令集的发展历史，然后重点分析了 SSE4 指令集中可快速提升视频编码速度的两条指令，MPSADBW 指令和 PHMINPOSUW 指令，详细分析了这两条指令的参数，计算原理，返回结果等。最后给出了使用 SSE4 指令集实现快速计算 *SAD* 的部分代码、原理等。

第 4 章 运动估计优化的实现及实验结果分析

4.1 SSE4 指令优化原理及实现

H.264 支持宏块的 7 种分区模式, 从 16×16 到 4×4 , 分别为 model~7, 分区模式如图 4-1 所示^[46]。图 4-1 中, 上层宏块的分区模式为 model~4, 下层子宏块的分区模式为 mode4~7。

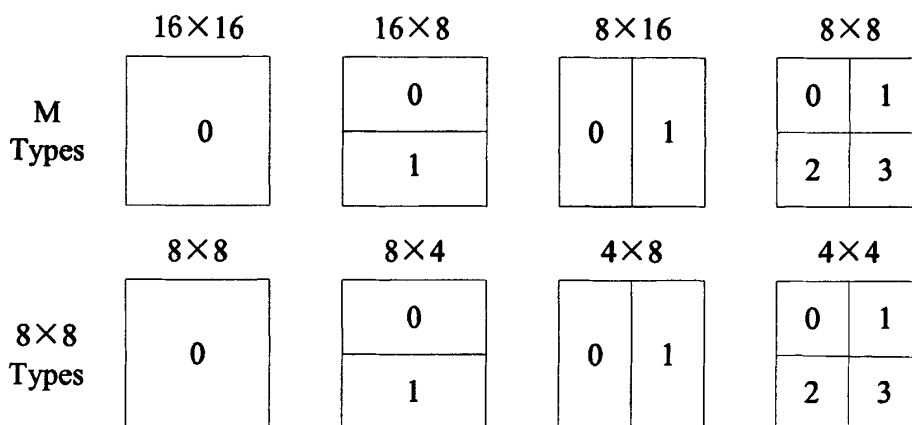


图 4-1 快速运动估计宏块的 7 种分区模式

在本论文 2.2.1 节中, 式 2-5 给出了参考模型 JM(Joint Model)中 H.264 中定义的匹配误差函数。这里重新引入变量 $RDCost$, 定义如式 4-1 所示, 各参数定义同式 2-5^[47-49]。

$$RDCost(MV, \lambda_{MOTION}) = SAD(s, c(MV)) + \lambda_{MOTION} \times R(MV - PMV) \quad (4-1)$$

每一种模式对应一个 $RDCost$, 第 3 章分析了用 SSE4 指令快速计算 SAD 的原理, 这样用 SSE4 指令就可以快速的计算出 $RDCost1 \sim 7$, 通过对比, $RDCost$ 最小的模式对应的即为最佳模式。图 4-2 给出了使用 SSE4 指令对快速运动估计中模式选择优化的流程图。

对优化的快速运动估计模式选择方案步骤简单描述如下:

Step1: 顺序选择帧中未进行宏块划分的空间区域(16×16 像素大小)。

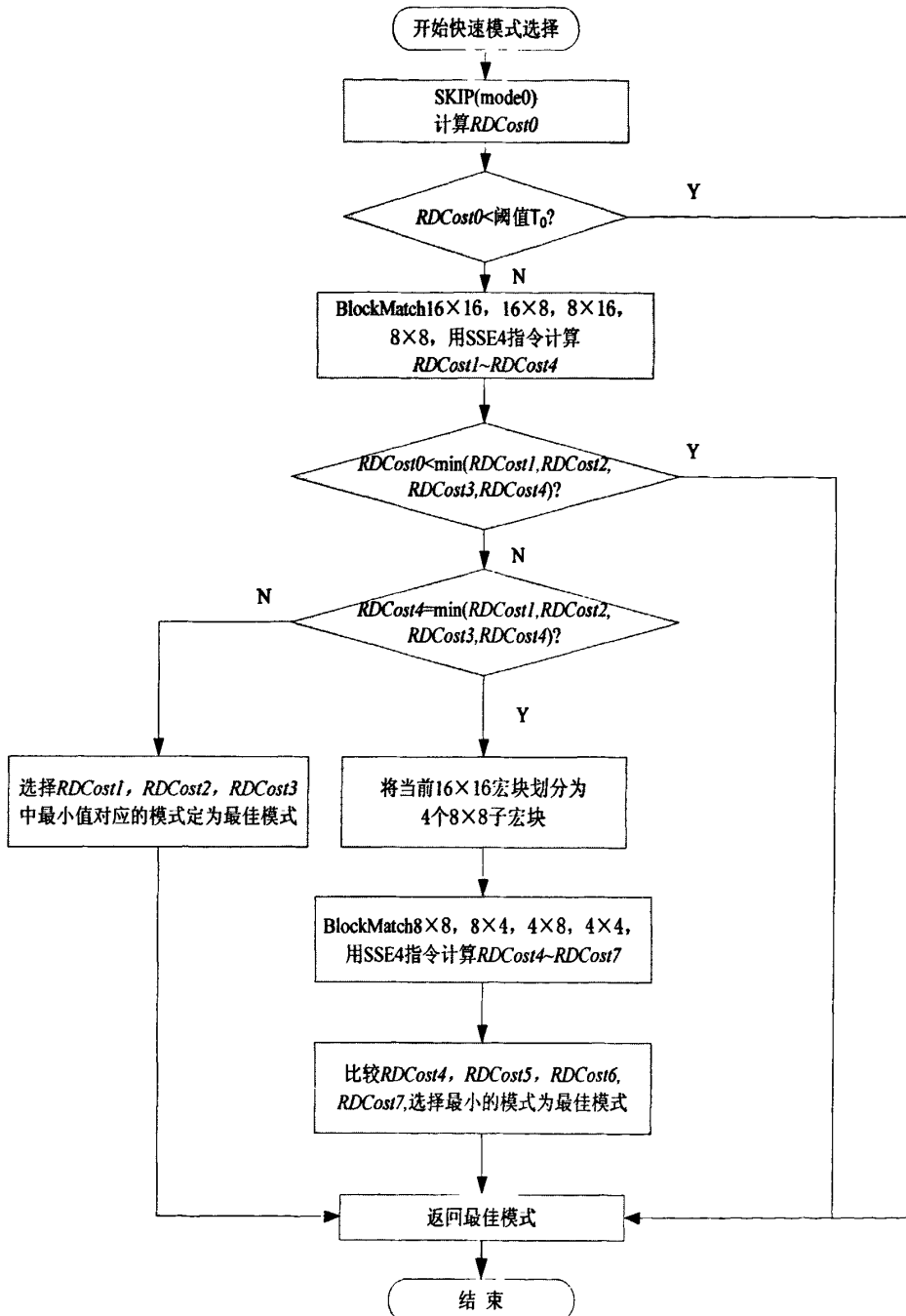


图 4-2 快速运动估计模式选择优化流程图

Step2: 初步定为 SKIP 模式, 即 mode0, 计算 $RDCost0$ 并判断 $RDCost0$ 与阈值 $T0$ 大小, 若 $RDCost0 < \text{阈值 } T0$, 则选择 SKIP 模式为最佳模式, 返回最佳

模式，否则，继续 Step3。

Step3：对宏块 16×16 ， 16×8 ， 8×16 ， 8×8 ，用 SSE4 指令计算 $RDCost1 \sim RDCost4$ ，并找出 $RDCost1 \sim RDCost4$ 中最小值，判断 $RDCost0$ 与 $RDCost1 \sim RDCost4$ 中最小值的大小，若 $RDCost0 < \min(RDCost1, RDCost2, RDCost3, RDCost4)$ ，则选择 SKIP 模式为最佳模式，返回最佳模式，否则，继续 Step4。

Step4：判断 $RDCost1 \sim RDCost4$ 中最小值是否为 $RDCost4$ ，若不是，则 $RDCost1 \sim RDCost3$ 中最小值对应的模式即为最佳模式，返回最佳模式，否则，继续 Step5。

Step5：将当前 16×16 像素大小宏块划分为 4 个 8×8 子宏块，在每个 8×8 子宏块中用 SSE4 指令计算 8×8 ， 8×4 ， 4×8 ， 4×4 ，对应的 $RDCost4 \sim RDCost7$ 值，比较 $RDCost4$ ， $RDCost5$ ， $RDCost6$ ， $RDCost7$ ，选择最小值对应的模式为最佳模式，返回最佳模式。

Step6：再依次按照 Step1~Step5 计算下一个未进行宏块划分的空间区域，直到当前帧的空间区域都被块划分完毕为止。

4.2 试验环境

为了验证基于 SSE4 指令集的快速运动估计优化算法性能的优劣，本章在 JVT 的 JM8.6 平台^[50-52]上，对一系列的 .yuv 标准视频序列进行了测试，分别比较了 JM8.6 平台下使用全搜索算法及本论文提出的优化算法对标准视频压缩的时间复杂度、码率、图像质量等。

4.2.1 JM 平台模型简介

H.264 有三大开源编码器，分别是 X264、T264，JM(Joint Model)平台，JM 是 JVT(joint video team)的官方测试源代码，由德国的 Hhi 研究所负责开发，JM 实现了 H.264 所有的特性，由于是官方的测试源码，所以学术研究的算法都是在 JM 基础上实现并和 JM 进行了比较。Hhi 研究所于 2002 年 2 月推出 JM1.0，至今已经推出多次升级，目前最新版本是 JM17.0。其中于 2004 年 9 月推出的 JM8.6 已经实现了标准 H.264 的基本编码功能，后续版本主要是引入一些先进的算法并针对高保真视频做扩展，比如，在本论文第 2 章，为了研究典型运动估计搜索算法，使用了 JM10.2 平台，该平台在 JM8.6 平台基础上引入了 EPZS 算法，简

化 UMHexagonS 算法等。学术上的算法研究一般采用的是 JM8.6 平台，在该平台上实现，并将优化的算法与标准的 JM8.6 平台进行比较。

本章所使用的平台为 JM8.6 平台，该平台包括编码器、解码器和一个 RTP 包分析工具。其对应的配置文件，分别为 `encode_baseline.cfg`、`encode_main.cfg` 和 `encoder_extended.cfg`，这里使用的是 `encode_baseline.cfg`。JM8.6 的编码器包括文件操作部分、编码控制、B 帧信息、SP 帧信息、环路滤波器、差错控制、输出控制和码率控制等等。下一节将详细分析，实验中 JM8.6 平台各参数的设置情况。解码器相对参数设置比较简单。

4.2.2 测试参数说明

使用 SSE4 指令对快速运动估计的模式选择进行了优化，为了验证算法性能，对一系列不同的标准.yuv 视频序列进行压缩编码，具体视频序列信息下一小节将详细论述。现对实验的相关参数进行详细说明。

实验的硬件平台是 CPU: Intel 双核 T6400，包含 SSE4 指令集，内存: 2G，显卡: HD4570。

实验的软件平台是 VISTA BASIC 操作系统，MS VS2008 编译 JM8.6。

JM8.6 中具体参数设置如下：

- `encoder_baseline.cfg`(基本档次): 视频序列为 IPPP 模式;
- `InputFile`(输入视频序列): 本章中为 7 个 `qcif.yuv` 文件及 4 个 `cif.yuv` 文件;
- `StartFrame`(起始编码帧): 0, 从第 0 帧开始编码;
- `FramesToBeEncoded`(编码帧数): 本章统一设为 100, 对每个视频序列的前 100 帧进行编码;
- `FrameRate`(帧率): 每秒钟传输 30 帧图像;
- `SourceWidth & SourceHeight`(图像像素宽高, 必须是 16 的倍数):
当输入 `qcif.yuv` 文件时, 为 176×144 , 当输入 `cif.yuv` 文件时, 为 352×288 ;
- `OutputFile`(输出文件): *.264;
- `IntraPeriod`(I 帧的周期): 统一设为 10;
- `Usehadamard`(是否使用 hadamard 传输): 使用 Hadamard 传输;
- `SearchRange`(最大搜索范围): 16 个像素点;

- NumberReferenceFrames(参考帧数目): 1;
- NumberBFrames(插入的 B 帧数目): 0, 基本档次不支持 B 帧;
- SPPicturePeriodicity(是否支持 SP 帧): 0, 不支持 SP 帧;
- QPISlice/QPPSlice(量化参数): 本章中选用参数 20、24、28、32、36;
- SymbolMode(熵编码模式): 本章中选择熵编码模式为 CAVLC。

4.2.3 测试标准视频序列说明

为了验证使用 SSE4 指令对快速运动估计的模式选择优化后的算法的性能, 在 JM8.6 平台下进行优化, 对一系列不同的标准.yuv 视频序列进行压缩编码, 并与使用 JM8.6 平台压缩的视频进行比较。实验过程中采用了各种具有不同运动特征的标准视频序列, 如表 4-1, 表 4-2 所示。

表 4-1 *cif.yuv 视频序列信息





标准视频序列	本章简称为	视频序列第一帧图像	序列特点
bridge_close_cif	brid_cif		镜头不变, 背景不变, 远处人小范围变动
waterfall_cif	wfall_cif		镜头由近到远迅速运动, 背景变化较大, 瀑布占画面面积较小
bus_cif	bus_cif		镜头迅速由右向左运动, 背景变化较大, 车速较快,
walk_cif	walk_cif		镜头随着人运动变化剧烈, 背景变化较大, 人动作变化较大

表 4-2 *qcif.yuv 视频序列信息

标准视频序列	本章简称为	视频序列第一帧图像	序列特点
miss_america_qcif	miss_qcif		镜头不动， 背景不动， 表情动作较缓慢
akiyo_qcif	akiy_qcif		镜头不动， 背景不动， 人头部缓慢变化
highway_qcif	hway_qcif		镜头向前推进， 图像上部分变化小， 下半部变化较小
foreman_qcif	fman_qcif		镜头从左至右变动， 背景有变化， 表情动作变化较大，
trevor_qcif	trev_qcif		镜头不动， 背景变化较大， 画面变化较大
walk_qcif	walk_qcif		镜头随着变化剧烈， 背景变化较大， 人动作变化较大
coastguard_qcif	cogd_qcif		镜头由右向左运动快， 背景变化很大， 快艇运动较快

测试序列为 4 个公共中间格式(CIF, Common Intermediate Format)文件, 7 个 1/4 公共中间格式(QCIF, Quarter Common Intermediate Format)文件, 采样格式为 YUV 4: 2: 0。

表 4-1 给出了实验中 4 个*cif.yuv 视频序列的具体信息, 表 4-2 给出了实验

中 7 个*.cif.yuv 视频序列的具体信息。表 4-1, 4-2 中视频基本是按照画面运动强度由弱到强的顺序排列, 这些视频序列中, 每个序列的运动强度, 背景复杂度, 摄像机的运动情况均不相同, 具体信息见表 4-1, 4-2。实验证明, 优化后的算法对具有不同特性的视频序列产生的编码效果不尽相同。

4.3 实验结果

本文在 JM8.6 平台上, 实现了 SSE4 指令优化的运动估计算法, 分别使用该算法及全搜索算法对上述 4 个*.cif.yuv 标准视频文件、7 个*.qcif.yuv 视频文件进行压缩编码, 记录 Y 分量峰值信噪比(PSNR)、码率及编码时间等。统一对上述 11 个.yuv 视频文件前 100 帧进行编码, QP 值分别设为 20、24、28、32、36。

如附录 A 中表 A1、表 A2、表 A3、表 A4 及表 A5 所示, 分别对应 QP 取值 20、24、28、32 及 36 时的实验结果, 记录 JM8.6 平台上全搜索算法及 SSE4 指令优化的运动估计算法对标准视频序列编码的 Y 分量峰值信噪比 PSNR 值、平均码率及编码时间。并根据式 2-8 至 2-10 的定义, 计算出峰值信噪比增量 ($\Delta PSNR$), 码率增量 ($\Delta Bits$) 以及编码时间节省因子 (ΔT)。为了直观的对比两种算法的编码时间, 对应表 A1~A5 中两种算法的编码时间, 生成折线图, 如图 4-3、图 4-4、图 4-5、图 4-6、图 4-7 所示, 分别对应 QP 取值 20、24、28、32、36 时的编码时间对比图, 横、纵坐标分别表示标准.yuv 视频序列、编码时间。

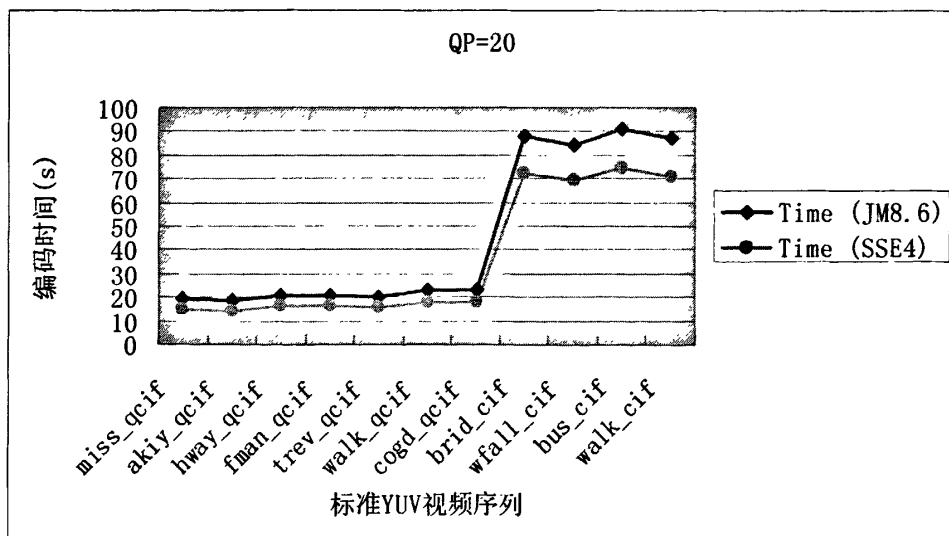


图 4-3 QP=20 时编码时间对比图

图 4-3 所示，为 $QP=20$ 时，在 JM8.6 平台上采用全搜索算法及采用 SSE4 指令集优化后的快速运动估计模式选择算法对 11 个.yuv 视频序列压缩的编码时间对比图。采用 SSE4 优化后的模式选择算法能显著减少编码时间。

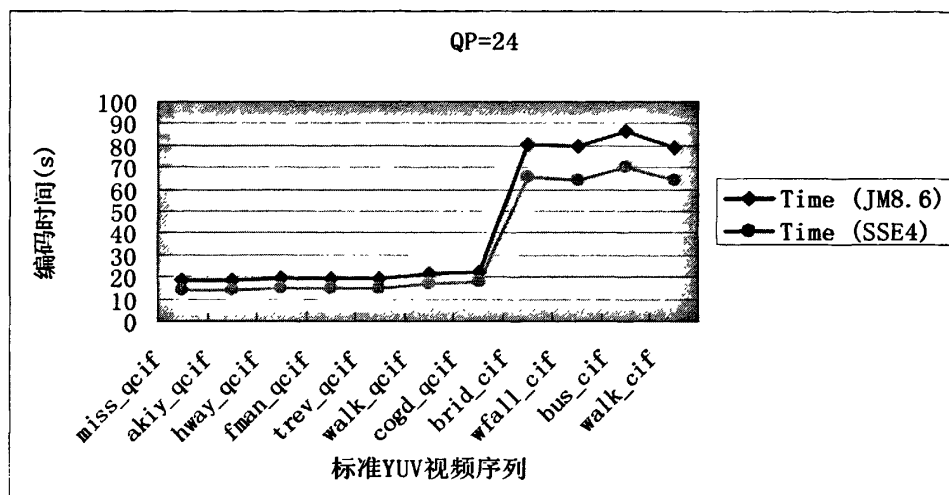


图 4-4 $QP=24$ 时编码时间对比图

图 4-4 所示，为 $QP=24$ 时，分别采用两种算法对 11 个.yuv 视频序列压缩的编码时间对比图。采用 SSE4 优化后的模式选择算法能显著减少编码时间。

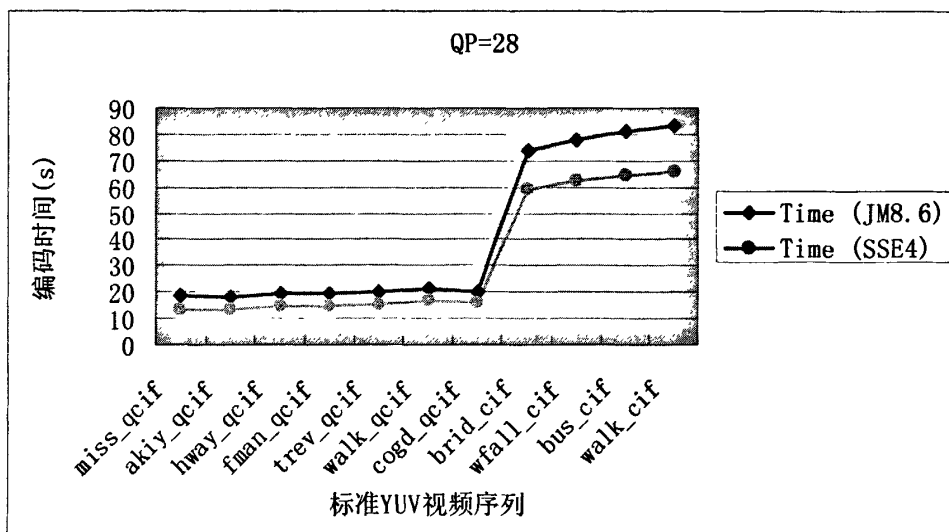


图 4-5 $QP=28$ 时编码时间对比图

图 4-5 所示，为 $QP=28$ 时，分别采用两种算法对 11 个.yuv 视频序列压缩的

编码时间对比图。采用 SSE4 优化后的模式选择算法能显著减少编码时间。

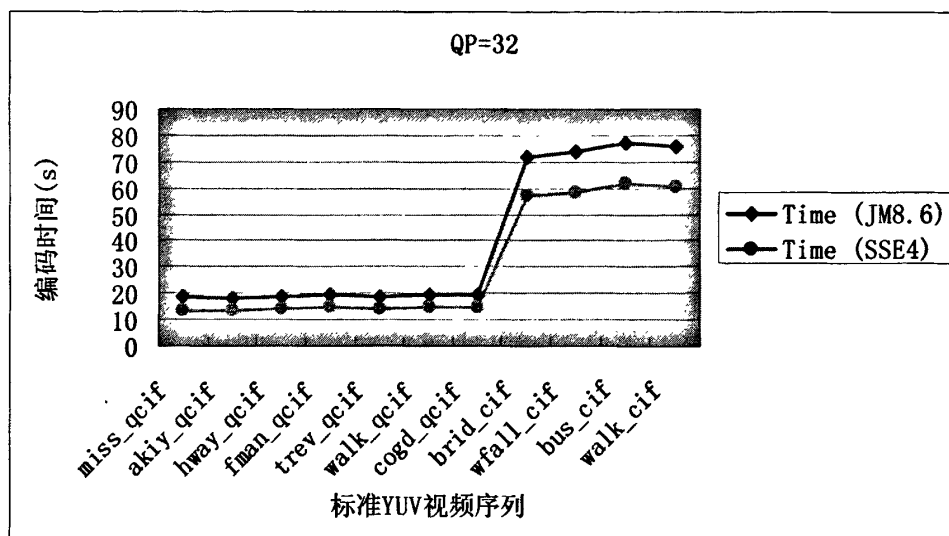


图 4-6 $QP=32$ 时编码时间对比图

图 4-6 所示，为 $QP=32$ 时，分别采用两种算法对 11 个.yuv 视频序列压缩的编码时间对比图。采用 SSE4 优化后的模式选择算法能显著减少编码时间。

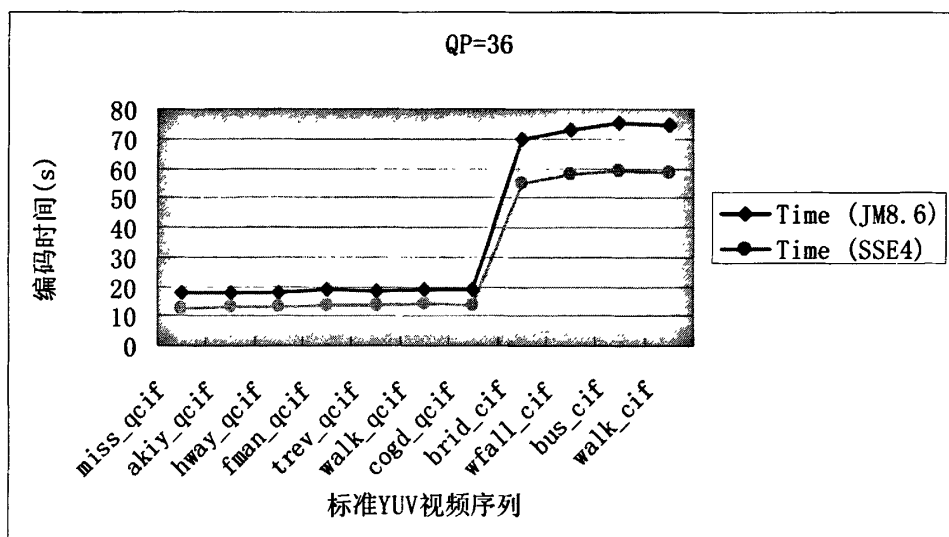


图 4-7 $QP=36$ 时编码时间对比图

图 4-7 所示，为 $QP=36$ 时，分别采用两种算法对 11 个.yuv 视频序列压缩的编码时间对比图。采用 SSE4 优化后的模式选择算法能显著减少编码时间。

由表 A1~A5、图 4-3~4-7 可以看出, 使用 SSE4 指令优化的快速运动估计模式选择算法, 对视频序列的图像质量稍有影响, 峰值信噪比稍有降低, 但是 $\Delta PSNR$ 数值较小, 对图像质量影响不大, 肉眼几乎察觉不出; 码率也稍有增加, 绝大部分码率增量 ($\Delta Bits$) 小于 1%, 影响也不大; 编码时间显著减少, 编码时间节省因子 (ΔT) 在 18~30% 之间。因此, SSE4 指令优化的快速运动估计模式选择算法可以在不显著降低图像质量, 不显著增加码率的前提下, 较大的节省编码时间。接下来, 将从这三个方面对实验数据进行更加全面的比较分析。

4.4 实验数据分析及结论

本文在 2.4 节中, 讨论了三个衡量算法性能的指标, 分别是峰值信噪比增量 ($\Delta PSNR$), 码率增量 ($\Delta Bits$) 以及编码时间节省因子 (ΔT)。一个好的算法应尽可能少的降低图像损失, 即峰值信噪比尽可能小的降低; 码率增量尽可能小; 编码时间节省因子 (ΔT) 尽可能大。下面就分别对峰值信噪比增量 ($\Delta PSNR$), 码率增量 ($\Delta Bits$) 及编码时间节省因子 (ΔT) 这三个指标进行详细比较分析。

4.4.1 峰值信噪比分析

根据表 A1~A5 中峰值信噪比增量 ($\Delta PSNR$) 的数据, 生成折线图, 比较 QP 分别取值 20、24、28、32、36 时, 对各序列编码的峰值信噪比增量, 如图 4-8 所示。

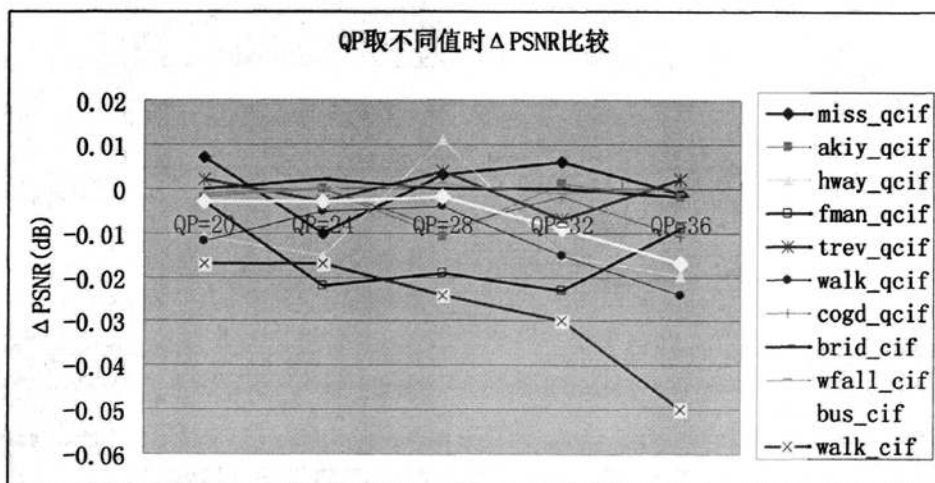


图 4-8 QP 取不同值时 $\Delta PSNR$ 比较

相对于 JM8.6 平台的全搜索算法,使用 SSE4 指令优化的运动估计选择算法,并没有显著降低各序列的 $PSNR$ 值,最坏的点,降低了 0.05dB,而当 $\Delta PSNR$ 小于 0.1dB 时,人眼几乎观察不出图像质量的差别。观察图 4-8,会发现有个别点大于 0dB,也就是说,使用 SSE4 指令集优化的快速运动估计算法, $PSNR$ 值在个别情况下会增加,从某种意义上讲,相对于 JM8.6 平台下的全搜索算法,对图像质量还有所改善。这只是少数点,大部分点的 $PSNR$ 值都小于 0dB,但对图像质量影响不大。图 4-8 中右侧的 11 个视频序列,基本上是按照图像运动程度由平缓到剧烈的顺序排放,可以得出大致规律,视频序列运动越剧烈,使用 SSE4 指令集优化的快速运动估计算法编码后,图像损失将越大。

4.4.2 码率增量分析

根据表 A1~A5 中码率增量 ($\Delta bit\ rates$) 的数据,生成折线图,比较 QP 分别取值 20、24、28、32、36 时,对各序列编码的码率增量,如图 4-9 所示。

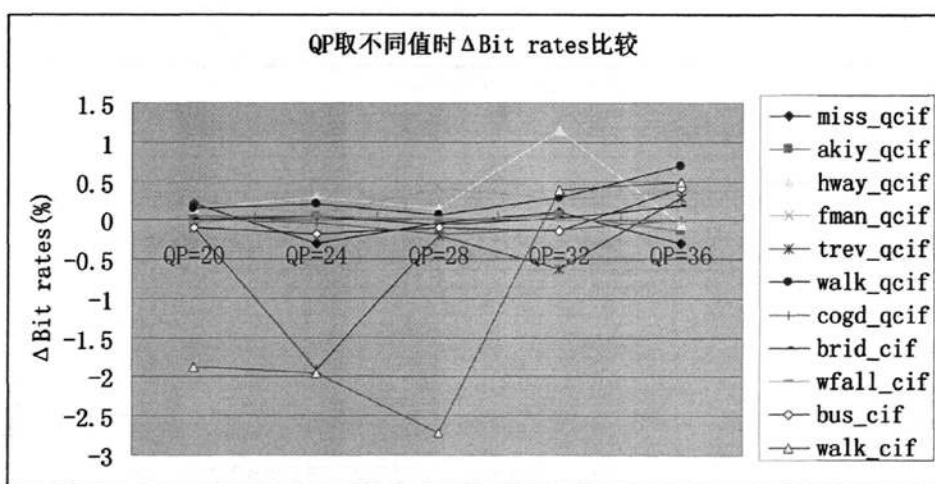


图 4-9 QP 取不同值时 $\Delta Bit\ rates$ 比较

相对于 JM8.6 平台的全搜索算法,使用 SSE4 指令优化的运动估计选择算法,并没有显著增加编码后的码率,最坏的点为 1.16%,绝大部分码率增量都小于 0.5%,也就是说,优化的算法并没有显著增加码率。

相反,会观察到有很多点码率增量小于零,比如: walk_cif 序列、trev_qcif 序列,这两个序列都是画面运动剧烈,图像纹理复杂的视频序列。如图 4-9 中, walk_cif 序列, $QP=20$ 、24、28 时,对应码率分别降低了 1.86%、1.95%、2.71%。对这种运动剧烈、纹理复杂的视频编码时,使用 SSE4 指令优化后的算法相对于

JM8.6 平台的全搜索算法, 不仅不会显著增加码率, 在一定程度上还有控制码率的作用。

4.4.3 编码时间分析

根据表 A1~A5 中编码时间节省因子 (ΔT) 的数据, 生成折线图, 比较 QP 分别取值 20、24、28、32、36 时, 对各序列编码的编码时间节省量。这里为了更清晰的观察到不同运动特征序列对应的编码时间节省因子, 只选取了具有代表性的几个序列:

miss_qcif、fman_qcif、trev_qcif、walk_qcif、brid_cif、bus_cif

以上 4 个 *qcif.yuv 文件, 图像运动由缓慢到剧烈, 两个 *cif.yuv 文件, bus.cif 比 brid.cif 剧烈, 同时 bus.cif 伴随着镜头由右向左快速运动, brid.cif 镜头不动, 具体各视频特性见表 4-1, 4-2。根据以上 6 个视频序列对应的编码时间节省因子 (ΔT), 生成折线图, 如图 4-10 所示。

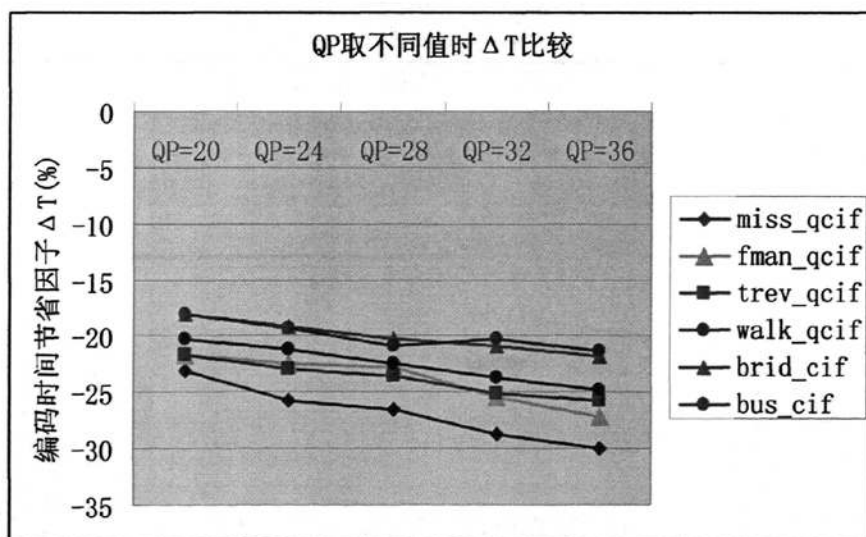


图 4-10 QP 取不同值时 ΔT 比较

由图 4-10, 使用 SSE4 指令集优化的快速运动估计模式选择算法, 相对于 JM8.6 平台的全搜索算法, 可以节省 18~30% 的编码时间。此外, 可以得出几个与编码时间节省因子相关的三个结论:

(1) 对同一个视频序列, 随着 QP 值增大, 对应的编码时间节省因子将增大, 即, QP 值越大, 节省的编码时间越多;

(2)对不同的视频序列,同一个 QP 值下,镜头与图像运动强度越大,图像纹理越复杂,节省的时间编码因子越小,即节省的编码时间越小,则使用 SSE4 指令集优化的快速运动估计模式选择算法,更加适用于运动强度较小的视频序列;

(3)对于*cif.yuv 文件和*qcif.yuv 文件的编码效果不同,*cif.yuv 系列视频的编码时间节省因子为 18~22%,明显小于*qcif.yuv 系列视频的编码时间节省因子 20~30%。这是由于*cif.yuv 一帧图像像素高宽为 352×288 ,而*qcif.yuv 一帧图像像素高宽为 176×144 。*cif.yuv 图像的像素大小是*qcif.yuv 的 4 倍,*cif.yuv 文件的帧内预测编码及熵编码时间占总编码时间的比例上升,快速运动估计的时间所占比例相对下降,所以使用 SSE4 指令集优化的快速运动估计算法效果有所下降。

综上所述,基于 SSE4 指令集的快速运动估计算法,在保证较小图像质量损失、较低码率增量同时,能显著的减少编码时间,节省编码时间 18~30%。该算法更加适用于运动强度较小的视频序列。

4.5 本章小结

本章在 JM8.6 上实现了基于 SSE4 指令集的快速运动估计优化,首先分析研究了该算法的原理、流程图、实现步骤等。然后分别使用该算法及 JM8.6 平台下的全搜索算法,对 miss_america_qcif 等一系列标准视频序列进行编码测试,阐述了实验条件,并分析了实验结果。

分析了峰值信噪比、码率增量、编码时间节省因子三个衡量算法性能的指标,实验证明,本文提出的基于 SSE4 指令集的快速运动估计算法,在保证较小图像质量损失、较低码率增量同时,能显著的减少编码时间,节省编码时间 18~30%。该算法更加适用于运动强度较小的视频序列。

第5章 总结与展望

5.1 总结

H.264 作为新一代国际视频编码标准, 编码性能出色, 具有之前算法无法比拟的优势。由于其出色编码性能, 它将会取代以往的视频编码标准并成为未来一段时期内各种视频应用的主要标准, 其发展前景被人们看好。但是, H.264 优越的编码性能是以高复杂度为代价的, 它的编码复杂度大约是 H.263 的 3 倍。保持原有算法的性能并降低算法的计算复杂度是 H.264 推进和扩展应用的关键所在。

本文的研究重点是使用 Intel 公司新发布的 SSE4 指令集实现 H.264 快速运动估计优化。具体的工作如下:

(1) 了解视频编码系列标准发展历史, 重点对新一代视频编码标准 H.264 进行研究, 分析其编解码框架, 关键技术等。

(2) 深入分析研究块匹配运动估计算法匹配准则和原理, 并对典型的几个运动估计算法: 全搜索算法、EPZS 算法、UMHexagonS 算法及简化 UMHexagonS 算法进行深入研究, 接着在 JVT 的 JM10.2 平台上分别实现这四种典型运动估计算法, 并对实验数据进行分析, 比较这四种算法的性能优劣。

(3) 了解 SSE 指令集发展历史, 重点研究 SSE4 指令集中可快速提升视频编码速度的两条指令, MPSADBW 指令和 PHMINPOSUW 指令, 详细分析了这两条指令的参数, 计算原理, 返回结果等。分析使用 SSE4 指令实现 H.264 快速运动估计算法优化的可行性, 并在 JVT 的 JM8.6 平台上实现了使用 SSE4 指令集实现快速计算 SAD。

(4) 使用 SSE4 指令集在 JVT 的 JM8.6 平台上实现了快速运动估计算法优化, 通过软件仿真, 与 JM8.6 平台上的全搜索算法进行性能比较, 详细分析了峰值信噪比增量、码率增量、编码时间节省因子等三个衡量算法优劣的性能指标。最后得出结论, 本论文提出的基于 SSE4 指令集的 H.264 编码标准的运动估计优化算法, 在保证较小图像质量损失、较低码率增量同时, 能显著的减少编码时间, 节省编码时间 18~30%。该算法更加适用于运动强度较小的视频序列。

5.2 未来工作展望

视频编码设计的领域非常广泛，这些年国内外对 H.264 的研究非常活跃。实际应用当中还有硬件平台的设计及实现。本文只是对编码算法进行了一部分研究，而 H.264 涉及帧内编码、熵编码、码率控制等等，并与实际硬件平台相结合。因此，本课题还有很多有待继续深入研究的问题，主要总结如下：

(1) 本课题使用 SSE4 指令集对 H.264 视频编码标准实现快速运动估计算法优化，主要是指指令级的优化，对具体的运动搜索算法并没有太多改进，如果在优化的运动估计算法的基础上再使用 SSE4 指令集进行改进，将会实现性能更好的运动估计优化，从而进一步在不影响图像质量和码率的前提下，大大节省编码时间。

(2) 对 H.264 算法优化涉及帧内编码、帧间编码、熵编码、码率控制等多方面，今后可以在这些方面进行深入研究，进一步提高编码速度。

(3) H.264 有三大开源编码器，分别是 X264、T264，JM 平台，JM 实现了 H.264 所有的特性，由于是官方的测试源码，所以学术研究的算法都是在 JM 基础上实现并和 JM 进行了比较，但其程序结构冗长，只考虑引入各种新特性以提高编码性能，忽视了编码复杂度，其编码复杂度极高，不宜实用。JM 一直没有做实用化方面的努力，所以其解码速度代表的是 2003 年的水平。而 X264 注重实用，与 JM 相比，在不明显降低编码性能的前提下，努力降低编码的计算复杂度，适用实时编解码传输。T264 是中国视频编码自由组织联合开发的 264 编解码器。今后对 X264、T264 进行相关研究。

致 谢

本论文从选题、相关资料文献查阅到论文实验展开以及最终论文的撰写，得到了各方大力帮助，在此我要向大家表示诚挚的谢意。

首先要真诚的感谢我的导师杨杰教授。杨老师严谨的治学作风、孜孜不倦的创新精神和渊博的专业知识，深深的影响了我，令我终生难忘；杨老师忘我的工作热情 and 无私的博大胸怀使我受益匪浅。将近三年的学习时光里，杨老师对本论文的研究工作提出了很多的宝贵意见，在学习和生活上也给予了无微不至的关怀与帮助。值此论文完成之际，谨向杨老师致以衷心的感谢！

感谢王虹教授、阙大顺教授、杨春金副教授、孟哲副教授、沈维聪副教授、黄晓放副教授、胡君萍副教授、王绪国副教授等，在我学习期间得到了他们无私的帮助、教育、指导。在此致以由衷的谢意！

感谢数字图像处理实验室的众位朝夕相处的兄弟姐妹们，他们是周慧、翁丽娜、侯者非博士，张磊师兄，陈寅、刘宇、金克琼、罗静、李亚南、简雄军、马海、王舸同学。感谢他们在学习生活中给予我的帮助。特别感谢陈寅同学和张磊师兄，他们对本文的实验给予了极大的帮助与支持。感谢信号与信息处理专业的所有同学，感谢他们对我无微不至的关心与支持。

最后，最需要感谢我的父母和亲人们，他们在我将近二十年的求学生涯中，给予了最大的关心、支持与鼓励，他们的关心和爱护是我前进的最大动力。在这里，我要由衷的感谢他们！

参考文献

- [1] 姚庆栋, 毕厚杰. 图像编码基础[M]. 北京: 清华大学出版社, 2006: 24-25.
- [2] 毕厚杰. 新一代视频压缩编码标准——H.264[M]. 北京: 人民邮电出版社, 2005: 55-62.
- [3] ISO/IEC. ISO/IEC 11 172-2. Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s-Part 2: Video(MPEG-1)[S]. Int.: (ISO/IEC)JTC 1, Mar. 1993.
- [4] ITU-T and ISO/IEC JTC 1. Recommendation H.262 and ISO/IEC 13 818-2(MPEG-2 Video). Generic Coding of Moving Pictures and Associated Audio Information-Part2: Video[S]. In.: ITU-T and ISO/IEC JTC 1, Nov.1994.
- [5] ISO/IEC. ISO/IEC 14496-2(MPEG-4 visual version 1). Coding of audio-visual objects-Part 2: Visual[S]. In.: ISO/IEC, 1999~2003.
- [6] ITU-T. ITU-T Recommendation H.261. Video codec for audio-visual services at p×64 kbit/s[S]. In.: ITU-T, Version 1, 1990; version 2, 1993.
- [7] ITU-T. ITU-T Recommendation H.263. Video coding for low bit rate communication[S]. In.: ITU-T, Version 1, 1995; version 2, 1998; version 3, 2000.
- [8] ITU-T and ISO/IEC JTC 1. Recommendation H.264 and ISO/IEC 14496-10(MPEG-4) AVC. Advanced video coding for generic audio-visual services[S]. In.: ITU-T and ISO/IEC JTC 1, 2003.
- [9] 国家数字音视频编解码技术标准工作组. 视频编码标准 AVS 技术介绍[J]. 电子产品世界, 2005(10): 58-63.
- [10] 张一夫. H.264/AVC 残差编码相关算法与应用的研究[D]. 北京: 清华大学, 2008.
- [11] Michael Horowitz, Anthony Joch, Faouzi Kossentini, Antti Hallapuro. H.264/AVC Baseline Profile decoder complexity analysis[J]. IEEE Trans. on Circuits and Systems for Video Technology, July 2003, 13(7): 704-716.
- [12] Hanli Wang; Sam Kwong, Chi-Wah Kok, "An Efficient Mode Decision Algorithm for H.264/AVC Encoding Optimization"[J]. IEEE Trans. Multimedia, vol. 9, no. 4, pp. 882-888, Jun. 2007.
- [13] B.G. Kim. Novel Inter-Mode Decision Algorithm Based on Macroblock (MB) Tracking for the P-Slice in H.264/AVC Video Coding[J]. IEEE Trans. Circuit Syst. Video Technol, vol. 18, pp. 273-279, Feb. 2008.

- [14] Lai-Man Po, Uddin Y, Kai Guo, Liping Wang. "Compensated Sum of Absolute Difference for Fast H.264 Inter Mode Selection"[C]. in Proc. IEEE ICALIP 2008, pp. 1486-1491.
- [15] E.Q.Li, Y.K.Chen. Implementation of H.264 encoder on General-Purpose processors with Hyper-Threading technology[C]. Proc. of SPIE Conf. on Image and Video Communication and Processing, 2004.
- [16] 郑琳俊, 张颖, 张兆扬. 多线程 H.264 视频传输的动态能量控制[J]. 电视技术, 2008, 32(03): 25-26.
- [17] 张惠民. 基于 H.264 的多平台视频监控系统的研究与实现[D]. 北京: 北京邮电大学, 2009.
- [18] 向东. 基于 H.264 框架的运动估计和变换研究[D]. 武汉: 华中科技大学, 2006.
- [19] 郭群燕. H.264 帧间模式选择与运动估计算法的研究与优化[D]. 武汉: 武汉理工大学, 2008.
- [20] 闫思彤. H.264 视频帧间运动估计快速算法研究[D]. 西安: 西安理工大学, 2008.
- [21] 范瑞霞, 李萍, 方浩. 基于 H.264/AVC 的帧间编码快速算法[J]. 北京理工大学学报, 2008, 28 (6): 510-512.
- [22] Ostermann J, Bormans J, List P, et al. Video Coding with H.264/AVC: Tools, Performance, and Complexity[J]. IEEE Circuits and System Magazine. First Quarter 2004:102-103.
- [23] Wenger S. H.264/AVC Over IP[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2003, 13(7): 645.
- [24] Kordasiewicz R, Shirani S. Electrical and Computer Engineering, University of McMaster, 2004[C]. Hamilton: University of McMaster Press, 2004.
- [25] Tillo T, Grangetto M, Olmo G. Redundant Slice Optimal Allocation for H.264 Multiple Description Coding[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2008, 18(1): 59-70.
- [26] Jing Yang, Xiangzhong Fang, Hongkai Xiong. A joint rate control scheme for H.264 encoding of multiple video sequences[J]. IEEE Transactions on Consumer Electronics, 2005, 51(2): 617-623.
- [27] ITU-T Rec. H.264/ISO/IEC 11496-10. Advanced Video Coding[S]. In: Final Committee Draft, Document JVTE022, September 2002.
- [28] 杨铭, 崔慧娟. H.264 视频编码快速算法预算数编码检错研究[D]. 北京: 清华大学, 2004.5.
- [29] Kate Y, Mukawa N, Okubo S. A Motion Picture Coding Algorithm Using Adaptive DCT Encoding Based on Coefficient Power Distribution Classification[J]. IEEE Journal on Selected Areas in Communications, 1987 5(7): 1090-1099.

- [30] Wiegand T, Sullivan G J, Luthra A. Draft ITU-T Recommendation H.264 and Final Draft International Standard 14496-10 AVC[S]. In: JVT of ISO/IEC JTC1/SC29/WG11 and ITU-T SG16/Q.6, Doc. JVT-G050r1, Geneva, Switzerland, May 2003.
- [31] 沈礼权, 张兆扬. 高效视频编码的算法优化及其扩展研究[D]. 上海: 上海大学, 2008.1
- [32] 陈寅. H.264 帧间编码算法研究与 DSP 移植[D]. 武汉: 武汉理工大学, 2009.12.
- [33] 姜东, 李炜等. 容错视频编码与传输技术研究[J]. 中国图像图形学报, 2006, 11(6): 778-783.
- [34] Wiegand T, Sullivan G J, Bjontegaard G, et al. Overview of the H.264/AVC Video Coding Standard[J]. IEEE Trans. on Circuits and Systems for Video Tech., 2003, 13(7): 560-572.
- [35] Alexis M, Tourapis. Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation. Visual Communications and Image Processing 2002, San Jose, 2002[C], San Jose: SPIE, 2002.
- [36] Zhibo Chen, Peng Zhou, Yun He, et al. Fast Motion Estimation for JVT, JVT-G016, Pattaya, 2003[C]. [S.1.]: Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG(ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6), , March, 2003.
- [37] Yi Xiaoquan, Zhang Jun, Ling Nan, et al. Improved and Simplified Fast Motion Estimation for JM, JVT-P021[C]//Proc. of the 16th Conference on ISO/IEC MPEG & ITU-T VCEG Poznan, Poland: [s. n.], July, 2005.
- [38] 杨育红, 徐烜, 季晓勇. 简化 UMHS 算法的改进方案[J]. 计算机工程与应用, 2006, 43(14): 46-48.
- [39] 百度百科. 计算机上的 SSE[EB/OL]. [2010-03-04]. http://baike.baidu.com/view/65687.htm?fr=ala0_1.
- [40] Varghese G, Sanjeev J, Chao T, et al. Penryn: 45-nm next generation Intel® core™ 2 processor. IEEE Asian Solid-State Circuits Conference, 2007. ASSCC '07, Jeju City, South Korea, 2007[C]. Santa Clara: [s.n.], 2008.
- [41] Wikipedia. SSE4 [EB/OL]. [2010-03-10]. <http://en.wikipedia.org/wiki/SSE4>.
- [42] Tseng S M, Kuo Y C, Ku Y C, et al. Software Viterbi Decoder with SSE4 Parallel Processing Instructions for Software DVB-T Receiver. 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, Chengdu, Aug. 2009[C]. [S.1.]: ISPA, 2009.
- [43] Intel® Software Network. Schema Validation with Intel® Streaming SIMD Extensions 4 (Intel® SSE4) [EB/OL]. [2008-12-15]. <http://software.intel.com/en-us/articles/schema-validation-with-intel-streaming-simd-extensions-4-intel-sse4/>.

- [44] Intel® Software Network. Motion Estimation with Intel® Streaming SIMD Extensions 4 (Intel® SSE4)[EB/OL]. [2008-10-30]. <http://software.intel.com/en-us/articles/motion-estimation-with-intel-streaming-simd-extensions-4-intel-sse4/#F>. Intel® SSE4 - Optimized Function for 16x16 Blocks.
- [45] Intel. D91561-001. Intel® SSE4 Programming Reference[S]. [S.1.]: Intel, April 2007: 7-8.
- [46] 朱向军. 视频运动对象分割与先进运动估计/运动补偿算法之研究[D]. 杭州: 浙江大学, 2006.
- [47] 张鹏. H.264/AVC 中的运动估计算法研究[D]. 西安: 西安电子科技大学, 2006.
- [48] Chang A, Oscar C A, Yeung Y M. A novel approach to fast multi-frame selection for H.264 video coding. Proc[C]. IEEE ICASSP'2003, Hong Kong, Apr. 6-10, 2003, 3:413-416.
- [49] Tourapis A M, Cheong H Y, Topiwala P, et al. Fast ME in the JM reference software[C]. JVT-P026. Poznań, Poland. 2005.
- [50] ISO/IEC and ITU-T. Joint Video Team(JVT). Advanced Video Coding for Generic Audio-visual Services[S], [S.1.]: JVT, 2005.
- [51] Joint Model (JM), H.264/AVC Reference Software. [Online]. [2008-06-21]. <http://iphome.hhi.de/suehring/tml/download/>.
- [52] Hu Feng, Liu Jia, Chen Shuzhen. Improved Diamond Search Algorithm for H.264/AVC Video Coding Standard[J]. Wuhan University Journal of Natural Sciences. 2008, 13 (1): 062-066.

攻读硕士学位发表论文和参加科研情况

一、攻读学位期间发表的论文

[1] 范亚琼, 杨杰, 陈寅. 基于 H.264 的网络视频监控系统的研究与实现. 中国教育部科技论文在线, 2009.09

二、攻读学位期间参加的科研项目

[1] 某海事局网上学校系统

附录 A

表 A1 QP=20 时各视频序列的 Y 分量 PSNR 值、码率、编码时间

序列 (QP=20)	PSNRY (JM8.6) (dB)	PSNRY (SSE4) (dB)	Δ PSNR (dB)	Bit rates (JM8.6) (kbit/s)	Bit rates (SSE4) (kbit/s)	Δ bit rates (%)	Time (JM8.6) (s)	Time (SSE4) (s)	Δ Time (%)
miss qcif	45.044	45.051	0.007	199.8	200.22	0.21	19.233	14.81	-23.00
akiy qcif	44.216	44.214	-0.002	179.88	179.97	0.05	19.000	14.54	-23.47
hway qcif	42.459	42.448	-0.011	582.97	583.84	0.15	20.698	16.102	-22.21
fman qcif	42.049	42.046	-0.003	485.55	486.75	0.25	20.859	16.329	-21.72
trev qcif	42.864	42.866	0.002	439.86	439.47	-0.09	20.156	15.78	-21.71
walk qcif	41.93	41.918	-0.012	930.47	931.75	0.14	22.863	18.25	-20.18
cogd qcif	40.781	40.78	-0.001	903.52	903.49	0.00	22.901	18.106	-20.94
brid cif	41.268	41.268	0.000	3783.6	3783.73	0.00	87.979	72.101	-18.05
wfall cif	41.202	41.201	-0.001	2118.94	2118.25	-0.03	84.286	69.059	-18.07
bus cif	41.122	41.119	-0.003	3767.52	3763.83	-0.10	91.085	74.661	-18.03
walk cif	42.515	42.498	-0.017	3264.14	3203.31	-1.86	87.186	71.265	-18.26

表 A2 QP=24 时各视频序列的 Y 分量 PSNR 值、码率、编码时间

序列 (QP=24)	PSNRY (JM8.6) (dB)	PSNRY (SSE4) (dB)	Δ PSNR (dB)	Bit rates (JM8.6) (kbit/s)	Bit rates (SSE4) (kbit/s)	Δ bit rates (%)	Time (JM8.6) (s)	Time (SSE4) (s)	Δ Time (%)
miss qcif	42.732	42.722	-0.010	99.85	99.55	-0.30	18.868	14.024	-25.67
akiy qcif	41.375	41.375	0.000	116.15	116.2	0.04	18.652	14.035	-24.75
hway qcif	40.15	40.135	-0.015	204.87	205.46	0.29	19.689	15.102	-23.30
fman qcif	39.11	39.088	-0.022	294.62	294.83	0.07	19.568	15.172	-22.47
trev qcif	39.769	39.766	-0.003	282.99	277.58	-1.91	19.687	15.185	-22.87
walk qcif	38.678	38.673	-0.005	604.06	605.35	0.21	21.568	17.001	-21.17
cogd qcif	37.322	37.322	0.000	547.21	546.98	-0.04	22.343	17.705	-20.76
brid cif	37.964	37.966	0.002	1563.1	1563.85	0.05	80.81	65.342	-19.14
wfall cif	37.974	37.974	0.000	1082.35	1082.84	0.05	79.687	64.254	-19.37
bus cif	37.795	37.792	-0.003	2353.98	2349.93	-0.17	86.432	69.783	-19.26
walk cif	39.479	39.462	-0.017	2090.08	2049.28	-1.95	79.249	64.353	-18.80

表 A3 QP=28 时各视频序列的 Y 分量 PSNR 值、码率、编码时间

序列 (QP=32)	PSNRY (JM8.6) (dB)	PSNRY (SSE4) (dB)	Δ PSNR (dB)	Bit rates (JM8.6) (kbit/s)	Bit rates (SSE4) (kbit/s)	Δ bit rates (%)	Time (JM8.6) (s)	Time (SSE4) (s)	Δ Time (%)
miss_qcif	38.095	38.101	0.006	36.18	36.22	0.11	19.128	13.621	-28.79
akiy_qcif	35.716	35.717	0.001	50.89	50.92	0.06	17.997	13.756	-23.57
hway_qcif	35.549	35.534	-0.015	54.15	54.78	1.16	18.982	14.067	-25.89
fman_qcif	33.844	33.821	-0.023	113.63	113.94	0.27	19.691	14.696	-25.37
trev_qcif	34.038	34.031	-0.007	106.26	105.608	-0.61	18.985	14.221	-25.09
walk_qcif	32.534	32.519	-0.015	230.93	231.61	0.29	19.689	15.024	-23.69
cogd_qcif	31.284	31.282	-0.002	145.26	145.102	-0.11	19.687	14.856	-24.54
brid_cif	32.916	32.916	0.000	346.97	347.08	0.03	71.986	57.014	-20.80
wfall_cif	32.308	32.307	-0.001	332.61	332.65	0.01	73.908	58.475	-20.88
bus_cif	31.737	31.728	-0.009	839.41	838.23	-0.14	77.373	61.672	-20.29
walk_cif	33.648	33.618	-0.030	722.76	725.56	0.39	76.163	60.225	-20.93

表 A4 QP=32 时各视频序列的 Y 分量 PSNR 值、码率、编码时间

序列 (QP=28)	PSNRY (JM8.6) (dB)	PSNRY (SSE4) (dB)	Δ PSNR (dB)	Bit rates (JM8.6) (kbit/s)	Bit rates (SSE4) (kbit/s)	Δ bit rates (%)	Time (JM8.6) (s)	Time (SSE4) (s)	Δ Time (%)
miss_qcif	40.468	40.471	0.003	57.33	57.31	-0.03	18.506	13.608	-26.47
akiy_qcif	38.623	38.612	-0.011	75.94	75.94	0.00	18.294	13.743	-24.88
hway_qcif	37.874	37.885	0.011	101.4	101.54	0.14	19.534	14.835	-24.06
fman_qcif	36.518	36.499	-0.019	181.28	182.05	0.42	19.325	14.927	-22.76
trev_qcif	36.823	36.827	0.004	172.45	172.13	-0.19	19.915	15.212	-23.62
walk_qcif	35.669	35.665	-0.004	381.18	381.46	0.07	21.326	16.542	-22.43
cogd_qcif	34.189	34.18	-0.009	300.76	300.22	-0.18	20.278	15.889	-21.64
brid_cif	35.432	35.432	0.000	694.65	694.21	-0.06	73.942	58.915	-20.32
wfall_cif	35.072	35.073	0.001	587.77	587.9	0.02	77.695	62.298	-19.82
bus_cif	34.754	34.752	-0.002	1438.57	1437.25	-0.09	81.311	64.382	-20.82
walk_cif	36.664	36.64	-0.024	1266.7	1232.31	-2.71	83.243	65.825	-20.92

表 A5 QP=36 时各视频序列的 Y 分量 PSNR 值、码率、编码时间

序列 (QP=36)	PSNRY (JM8.6) (dB)	PSNRY (SSE4) (dB)	Δ PSNR (dB)	Bit rates (JM8.6) (kbit/s)	Bit rates (SSE4) (kbit/s)	Δ bit rates (%)	Time (JM8.6) (s)	Time (SSE4) (s)	Δ Time (%)
miss qcif	35.842	35.841	-0.001	24.46	24.39	-0.29	17.869	12.52	-29.93
akiy qcif	33.137	33.135	-0.002	35.03	34.98	-0.14	17.865	13.392	-25.04
hway qcif	33.433	33.413	-0.020	31.58	31.56	-0.06	18.056	13.301	-26.33
fman qcif	31.183	31.174	-0.009	72.58	72.84	0.36	18.956	13.808	-27.16
trev qcif	31.512	31.514	0.002	64.41	64.6	0.29	18.595	13.803	-25.77
walk qcif	29.664	29.64	-0.024	138.56	139.52	0.69	19.029	14.325	-24.72
cogd qcif	28.914	28.903	-0.011	72.83	72.84	0.01	18.986	13.956	-26.49
brid cif	30.642	30.64	-0.002	165.7	166.02	0.19	70.134	54.877	-21.75
wfall cif	29.912	29.912	0.000	190.33	190.31	-0.01	72.989	57.721	-20.92
bus cif	29.02	29.003	-0.017	481.92	483.95	0.42	74.928	58.932	-21.35
walk cif	30.845	30.795	-0.050	423.67	425.73	0.49	74.389	58.278	-21.66

作者: [范亚琼](#)
学位授予单位: [武汉理工大学](#)

本文读者也读过(10条)

1. [曾川豪](#) [基于SSE技术的CCSDS译码复杂度的研究与改进](#)[学位论文]2010
2. [宋麒](#), [罗志宇](#), [丛鹏](#), [SONG Qi](#), [LUO Zhi-yu](#), [CONG Peng](#) [SSE指令集在60Co集装箱CT系统图像重建算法中的应用](#)[期刊论文]-核电子学与探测技术2007, 27(1)
3. [姚莹](#) [基于MMX技术的指纹图像匹配识别算法研究](#)[学位论文]2006
4. [谢红芝](#) [基于MMX技术的指纹图像并行处理算法](#)[学位论文]2005
5. [赵冬晖](#), [潘日华](#), [池凤彬](#), [ZHAO Dong-hui](#), [PAN Ri-hua](#), [CHI Feng-bin](#) [利用SIMD指令加速VLSI设计规则检查](#)[期刊论文]-微电子学与计算机2008, 25(7)
6. [李蕊](#), [陈浩](#), [姜昱明](#) [SSE在多媒体数据处理中的应用](#)[期刊论文]-计算机应用2002, 22(7)
7. [姜伟华](#) [针对实际多媒体程序和多媒体扩展指令集的SIMD编译优化](#)[学位论文]2005
8. [熊蓉](#) [一种H. 264码率控制算法及其在场景切换中的应用与研究](#)[学位论文]2010
9. [李仲林](#) [基于H. 264/AVC的帧内帧间模式选择优化算法研究](#)[学位论文]2010
10. [周冬辉](#) [H. 264快速运动估计算法研究与实现](#)[学位论文]2010

本文链接: http://d.wanfangdata.com.cn/Thesis_Y1680871.aspx