# 基于多核处理器的
# 多视点视频编解码的并行优化

# The Parallelism of Multi-view Video Coding on The Multi-core Processor

（申请清华大学工学硕士学位论文）


培 养 单 位 ： 计算机科学与技术系

学　　科 ： 计算机科学与技术系

研 究 生 ： 张 凤 妍

指 导 教 师 ： 杨 士 强 教授

联合指导教师 ： 文 铁 华 研 究 员

李　　洁 研 究 员



二〇〇九年五月

基于多核处理器的多视点视频编解码的并行优化

张凤妍

# 关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）根据《中华人民共和国学位条例暂行实施办法》，向国家图书馆报送可以公开的学位论文。

本人保证遵守上述规定。

**（保密的论文在解密后遵守此规定）**

作者签名：_____ 导师签名：_____

日　　期：_____ 日　　期：_____

# 摘　要

多视点视频(Multi-view Video)是近年来多媒体领域出现并迅速发展的一个新的研究方向，也是未来 3D 视频、自由视点视频 FVV(Free View Video)等应用发展的关键，它的提出体现了下一代多媒体应用网络化、交互性和真实感的发展趋势。目前由 MPEG 和 ITU 共同组成的联合视频小组 JVT 负责起草多视点视频编解码的标准---MVC（Multi-view Video Coding）。

多视点视频包含了一个场景的多个视点的视频信息，这使得其数据量非常庞大，要实现高效的压缩，相应的算法复杂度也比较高，对处理器性能提出了更高的要求。目前普通的处理器无法满足多视点视频编解码的处理需求，这将会阻碍多视点视频的发展和未来应用。同时，在体系结构领域，单核处理器的发展出现了瓶颈，而多核处理器成为其主要发展方向。如何利用并行技术充分发挥多核处理器的处理性能，将是解决多视点视频编解码问题的关键。随着多视点视频编解码技术不断成熟，关于它的并行优化已成为多视点视频技术的重要研究方向之一。

本文根据多视点视频编解码的自身特点，构造其编码的有向无环图(DAG)，基于该 DAG 图设计并实现了一种 MVC 可并行性的分析模型。在此基础上文章采用分层有向无环图任务调度模型，在多核处理器平台上设计并实现了针对 MVC 的启发式并行调度算法，来对多视点视频编解码进行了并行加速。实验结果表明，文中设计的启发式调度算法比其他调度算法有更高的加速比，具有更好的并行加速效果；同时实验结果也验证了本文所提的分析模型的正确性。

为了进一步提高加速效果，本文还将实验结果反馈到多视点视频编解码算法中，通过修改 MVC 的预测结构使得其具有更好的可并行性，获得更高的并行加速比。实验结果表明，修改后的预测结构具有更高的可并行性，同时其压缩性能与原来的预测结构基本上保持一致。

**关键词：**多视点视频编解码　　并行调度　　预测结构　　加速比

# Abstract

Multi-view video is one of the most popular research fields of multimedia, which is the key technology of 3DTV and FVV (free view video). The Joint Video Team (JVT), which is composed by MPEG and ITU, is trying to develop the standard of Multi-view Video coding (MVC).

Multi-view video contains many videos of one scene from different view, so the data is very huge, and the computation complexity of the compression algorithm is very high. To process MVC efficiently requires high computing capability of processor. If the processing speed is too low that can't meet the requirement of real-time processing, it will obstruct the development and the application of multi-view video. Multi-core processor provides a new platform for multimedia processing, based on which we can accelerate the processing by parallelism.

This thesis analyses the characteristic of MVC, provides a model, which is based on the DAG (Directed Acyclic Graph) of MVC, to analyze the parallelism of MVC. According to this, two heuristic scheduling algorithms of MVC are proposed and implemented, to parallelize the multi-view video coder and decoder on the multi-core processor platform. The experimental result shows that these two scheduling algorithm have better performance, and demonstrates the correctness and validity of the model.

To improve the parallelism of MVC and accelerate the processing speed further, this thesis modifies the new prediction structure of MVC and presents different ones according to different applications of MVC. The result shows the new prediction structures are more efficient for parallel processing, with almost the same compression performance.

**Keywords:** Multi-view Video Coding (MVC) Parallel Scheduling Prediction Structure Speedup

# 目 录

# 第1章　引言

## 1.1　论文研究背景、目的及其意义

多媒体技术是一种迅速发展的综合性电子信息技术，它利用计算机把文字、声音、图形与图像等多媒体综合一体化，使它们建立起逻辑联系，并能够进行加工处理。它的诞生给传统的计算机系统、音频和视频设备带来了方向性的变革，也给人们的工作、生活和娱乐带来了深刻的革命。

20 世纪末世界信息化步伐不断加快，多媒体技术在人类生产生活中发挥着越来越重要的地位，成为人们关注和研究的热点之一。随着网络的普及，多媒体技术逐渐朝着网络化、真实性、交互式等方向发展。从传统 2D 视频到如今正快速发展的 3D 视频，人们对于多媒体技术不断的提出新的要求，对于数字视频的需求也越来越大，视频的数据量也越来越大，为数据存储和信道传输带来了很大的压力。目前传输通道的发展受到了限制，于是在确保视频质量的基础上，如何用尽量少的比特数来表示视频信息，成为多媒体技术的研究热点。而音视频编码技术就是一种解决这一问题的关键技术，目前国际上有两个负责音视频编码的标准化组织：一是国际电信联盟标准化组(ITU-T)下的视频编码专家组(VCEG：Video Coding Expert Group)，二是国际标准化组织 ISO 和国际电工委员会 IEC 下的运动图像编码专家组(MPEG：Motion Picture Expert Group)。他们制定的视频编解码标准的发展情况如下图 1-1 所示：

ISO/IEC 在 1988 年成立了运动图像编码专家组 MPEG，致力于研究、开发多媒体压缩标准，在保证视频质量的前提下，对视频进行压缩，减低传输码率。其先后制定了 MPEG-1、MPEG-2、MPEG-4、AVC(Advanced Video Coding)、MVC(Multi-view Video Coding)等标准，主要应用于存储媒介(DVD)、广播电视、有线/无线网上的流媒体等。ITU-T 的视频编码专家小组 VCEG 制定的视频编解码标准为 H 系列，包括 H.261、H.262、H.263、H.263+、H.264 等，主要应用于实时视频通信领域，如会议视频等。

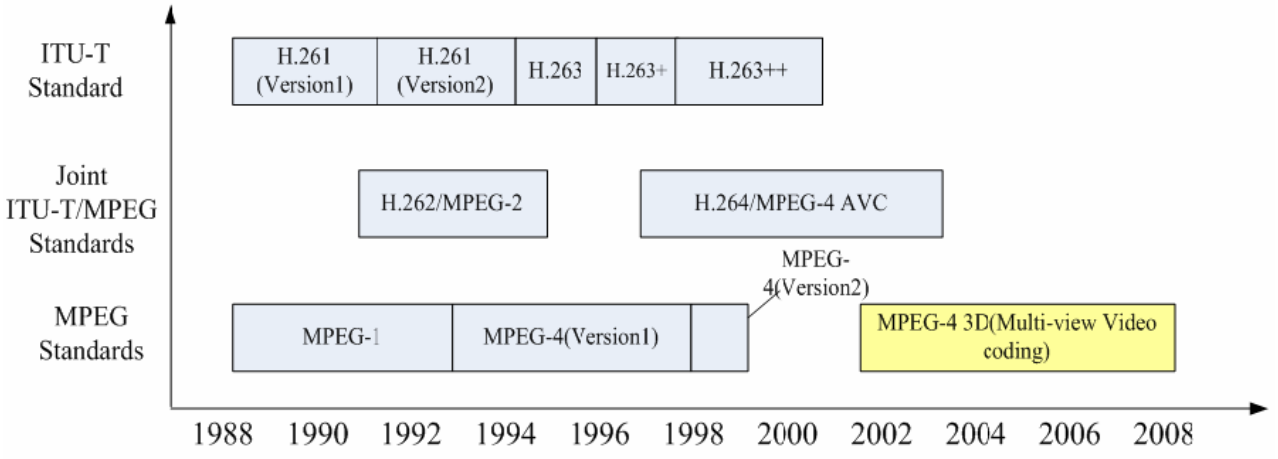



图 1-1 视频编解码标准的发展

2001 年由 MPEG 和 VCEG 共同成立了联合视频小组(JVT：Joint Video Team)，负责制定新一代数字视频编码标准-H.264/AVC，它既是 ITU-T 的 H.264，又是 ISO/IEC 的 MPEG-4 的第 10 部分，于 2003 年制定了最终版本。有关研究表明，与 MPEG2 编码相比，H.264 编码要节省 64%的比特率，与 H.263+或 MPEG4 Simple Profile 编码相比，其平均可节省 39%的比特率，而最多可节省 50%的比特率[1]，这表明 H.264 具有很高的压缩效率；而且 H.264/AVC 码流引入了网络适应机制，增加了差错恢复能力，使得其具有很好的 IP 和无线网络适应性，可以应用于视频电话，视频会议，监控系统，流媒体，手机电视，数字电视以及 IPTV 等，应用范围更加广泛。H.264/AVC 可以说是视频压缩技术的一个里程碑，有人曾经这么评价 H.264，“在可预见的 5 到 10 年内、在传统的编码框架下出现新的压缩协议进一步显著提高压缩效率的可能性很小、除非压缩理论有重大突破”。

除了这两大标准组织所研究的标准外，还有微软目前开发的 VC-1，以及我国自行研发的数字音视频编解码技术标准(AVS：Audio Video coding Standard)等，这些发展的也非常快。

视频编解码算法的压缩效率不断得到提高，但是同时引入了新的问题，其算法复杂度也随之不断提高，当前的处理器难以满足其需求，阻碍了其应用和发展。据估计，H.264/AVC 编码的计算复杂度大约相当于 H.263 的 3 倍，解码复杂度大约相当于 H.263 的 2 倍[2]。所以在 H.264/AVC 标准制定没多久，如何对其进行加速处理便成为研究热点[33-44]，人们从软件级的优化、硬件加速等方

面对其进行加速。

在视频编解码加速的研究过程中，微处理器的性能是其中一个很重要的影响因素。而微处理器一直以每年 1.5~2 倍的速度提高性能，但是由于其设计复杂度不断提高，再加上漏电流等问题，近几年处理器性能提高的速度降低到 1.2 倍左右，单核处理器的发展遇到了难以逾越的瓶颈。而随着科技的发展，各种应用的计算复杂度不断提高，对处理器的性能不断提出更高的要求。为了解决这个矛盾，从 2000 年开始，各大处理器生产厂商开始关注并设计多核处理器。Intel 公司 2004 年 5 月做出策略性决定，对于核心产品 X86 系列微处理器，终止所有单核产品的开发，转向集中开发多核产品。几乎与此同时，AMD 公司也表明，已停止集成单 CPU 核的 X86 系列微处理器的设计。多核微处理器具有结构简单、可扩展性好、兼容性好、能量有效性好、易于快速重构以适应特定计算任务要求等特点，它将取代单核处理器，成为未来处理器的发展方向。

如何利用并行技术充分发挥多核处理器的优势，在软件一级对视频编解码进行并行加速，是一个很崭新的重要课题。

随着计算机网络技术和多媒体技术的快速发展，人们对于多媒体的要求也越来越高，已经不能满足现有多媒体的形式，于是一种新型的数字媒体--多视点视频应运而生。多视点视频通过在场景中放置多台摄像机，获取到整个场景的多个视点的视频信号，将这些视频信号有效的组合在一起，提供给用户，方便他们自由的进行视点选择和场景漫游等，如下图 1-2 所示。多视点视频是针对交互式多媒体提出的，体现了下一代多媒体应用网络化、交互性和真实感的发展方向。

与传统的单视点视频一样，多视点视频面临的第一个问题就是数据量庞大。多视点视频包含一个场景的多个视点的信息，所以它的数据量将是传统视频的好几倍，并随着视点数的增加而成线性增长，能否对这些数据实现有效的压缩和存储将影响到多视点视频的实际应用。不过这些视频都是关于同一个场景的信息，在内容有着一定的相关性，数据上存在大量的冗余，具有很大的数据压缩的潜力。多视点视频编解码(MVC)成为继 H.264/AVC 标准制定后的一个新的视频编解码研究热点。

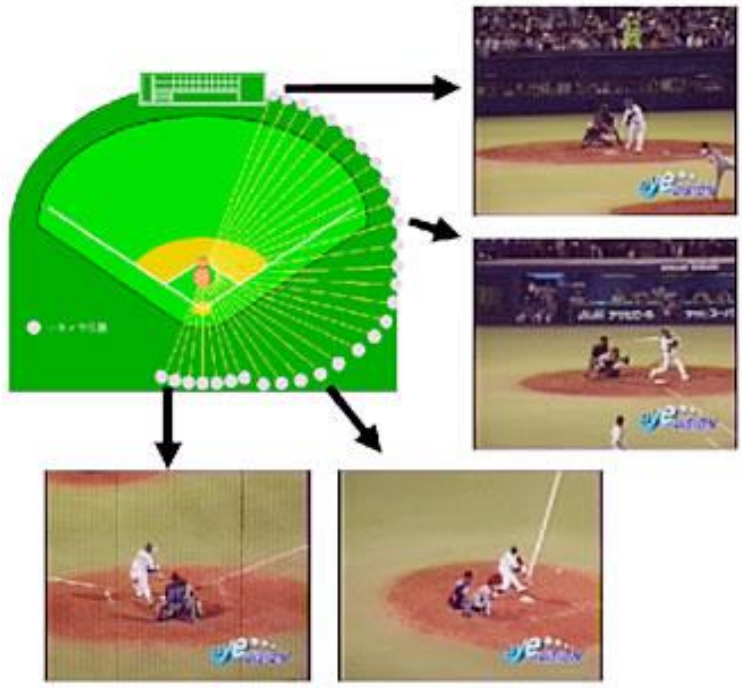


图 1-2 多视点视频

2001 年 12 月在第 58 次 MPEG 会议上表达了对多视点视频标准的强烈必要性[3]，2002 年 5 月在第 60 次 MPEG 会议上正式发出了 3DAV 的需求文档[4]，2005 年 1 月在香港的第 71 次 MPEG 会议上形成了 3DAV 的技术征集文档[5]。2006 年联合视频小组 JVT 开始负责起草 MVC 的标准，并在 MPEG 会议上发布其相关研究成果，包括 MVC 标准的草稿，MVC 的需求文档，MVC 的参考软件等。2008 年 5 月，JVT 颁布了 MVC 的最终修正草案 FDPM（Final Proposed Draft Amendment）。

除了压缩问题外，多视点视频编解码技术和以往的编解码技术一样，也存在着计算复杂度过高的问题。一方面其要处理的数据量比以往的视频要更庞大，另一方面为了保证有效的存储和传输，相比以前的编解码算法，多视点视频的编解码算法的复杂度也将会更高。

本文利用联合视频小组 JVT 提供的多视点视频编解码 MVC 的参考软件[6]对一个分辨率为 320×240，8 个视点(view)的视频进行测试，解码 19 帧需要 5 秒，而编码则需要 18 分钟。这距离实时处理每秒 30 帧的要求还有很大的差距。处理速度过慢将会严重影响 MVC 的应用和发展。由于国内外关于多视点视频的研

究均处于初级阶段，主要围绕如何提高压缩效率以及随机读取能力等方面进行研究，而对于多视点视频编解码的加速处理，目前尚未开始大规模的研究。但是在 MVC 标准化的过程中，就已经提出了并行的需求，我们相信，和其他视频编码标准一样，在其标准制定之后，多视点视频编解码 MVC 并行加速的研究将成为热点。

## 1.2　论文研究现状

随着获取和显示技术的发展，3D 视频逐渐走入现实生活中。2001 年 MPEG 成立了 3DAV 工作组，其首要任务就是定义 3D 音视频领域的范围和应用场景，并为其中的关键技术制定标准。多视点视频就是在 3DAV 框架下近年来迅速发展起来的研究领域，它是针对即将出现的交互式网络多媒体应用而提出的，是从 2D 视频向 3D 视频转变的基础关键技术之一。

多视点视频主要包含以下三个方面的应用：

1. 自由视点视频（Free Viewpoint Video）

   提供用户与场景的交互能力，用户可以自己选择在场景中的视点，如在体育比赛或数字视频游戏中，有多个摄像机从不同视点同步拍摄，通过基于图象/视频的合成技术（Image/Video Based Rendering —— IBR/VBR），用户可以随时改变视点，或者在一定范围内从任意一个角度观看，甚至可以在场景中漫游。例如 CMU 的 Eyevision 原型已在美国 CBS 的橄榄球赛事转播中进行了初步的尝试，日本早稻田大学的 FreeView TV 已建立了 100 个以上摄像机的实验系统，微软亚洲研究院也在搭建真实比赛场馆中的多摄像机环境；

2. 3D 视频（3D Video/3D TV）

   具有深度信息的多视点视频能够给用户提供三维的场景体验和感受，仿佛置身于场景中，达到逼真的效果。其中 3D 数字电视将是未来电视发展的方向，可以带给用户更逼真的体验与感受，随着基于屏幕的立体显示技术的成熟与实用化，多视点视频将会对数字电视的应用产生重大革新，将带动相关产业的发展。国际上正在进行 3DTV 系统的研究与开发工作包括欧盟的 ATTEST 项目,德国 HHI 的 IVVV 系统，美国 MERL 的 3DTV 等；

3. 沉浸视频（Immersive Video）

通过多路视频的传输与合成，为异地用户提供诸如身体语言、目光交流、凝视感知等更多信息交流方式，实现一种沉浸感的环境，支持更为真实、自然的协同工作。这方面的研究在美国 MIT、南加州大学、日本 ATR 通信系统研究实验室、瑞士联邦计算机科学学院、欧洲信息技术研究战略计划(ESPRIT)等一批大学和研究机构中广泛展开。

### 1.2.1　多视点视频技术的发展

多视点视频技术主要包括视频数据采集、视频编解码、实时传输、终端显示等技术。一个完整的多视点视频系统如下图 1-3 所示[7]：

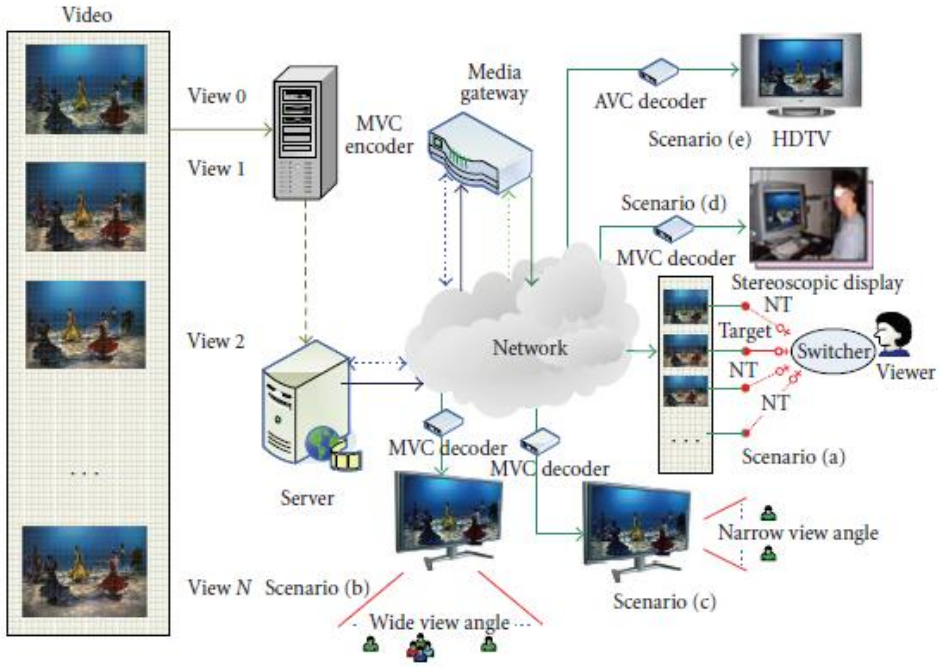



图 1-3 多视点视频系统的框架

多视点视频应用系统一般首先获取到一个场景的多个视点的视频信息，然后利用多视点视频编解码方法对其进行编码，将编码结果保存到服务端。然后通过各种传输通道(有线网、无线网等)将编码后的数据流传输到不同的客户端，并在客户端解码后，根据用户的不同需求，通过不同的显示设备将其显示出来。

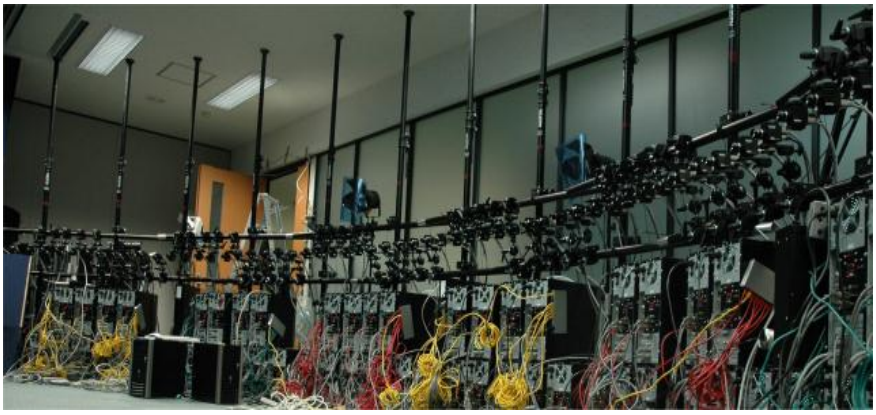
1、 多视点视频采集技术

视频采集最重要的设备就是摄像机了，它将自然界中的客观场景转化为模

拟的电信号输出。与传统视频信号采集技术一样，摄像机的标定是多视点视频信号采集的重要技术，它是多媒体领域的基础技术，通过参考点的空间坐标(x, y, z)和图像坐标(u, v)，确定摄像机内的几何与光学特性（内参数）以及摄像机在三维空间中的坐标关系（外参数）。关于这方面的研究目前已经比较成熟，标定方法也有很多，包括传统的空间参照点方法和自标定方法两大类。而多视点视频数据采集的相关标定方法和参数设置详见参考文献[8]。

视频采集技术对于多视点视频编解码有着重要的影响。这主要是由于视点间相关性是多视点视频进行压缩的关键，而视点间相关性依赖于相机的阵列形式，相机的间距，相机和拍摄对象之间的距离等。所以这些参数的设置对于多视点视频非常重要。目前主要的阵列方式包括以下三种[11]：直线型(图 1-4(a))、矩阵型(图 1-4(b))、弧线型(图 1-4(c))。

( a )　　( b )



( c )

图 1-4 多视点视频摄像机的排列

2、 多视点视频编解码技术

视频编解码技术的目的在于保证视频质量的前提条件下，对视频数据进行压缩，用尽量少的比特数来表示视频信息。多视点视频编解码技术是多视点视频技术的核心，多视点视频能否得到广泛的应用关键就在于其压缩技术的发展。关于这部分内容，将在下一小节详细介绍。

3、 多视点视频传输技术

视频传输包括模拟信号传输和数字信号传输两种。模拟视频传输将摄像头取得的电信号直接进行远距离传输，这种主要是在传统的视频监控系统中应用，它的成本比较高，传输距离也有限制。多视点视频传输主要是将模拟信号转化成数字视频信号，然后通过网络（有线网和无线网等）传输到远端。这方面主要包括传输信道和网络传输协议(例如 RTP 协议等)的研究，目的在于保证视频传输的流畅性、稳定性、可伸缩性和鲁棒性等。由于传输信道带宽的限制，对多视点视频编解码的压缩性能提出了更高的要求，同时要求其压缩后的码流具有良好的网络适应性。

4、 多视点视频显示技术

根据多视点视频应用场景的不同，其在系统终端解码后，通过不同的显示器终端显示技术，可以获得不同的观赏效果。根据多视点视频未来的主要应用，其显示技术主要包括触摸屏技术和 3D 显示技术。

触摸屏技术面向交互性多媒体，目前发展已经比较成熟，包括电阻式、电容式、红外线式和表面声波式等多种方式，在手机、笔记本、电子信息服务等设备上都得到了广泛的应用[14]。由于触摸屏技术面向交互式多媒体，在终端显示时需要速度快，有及时性，根据用户的需要随时完成数据切换，所以其要求多视点视频在显示终端能够快速解码和视点切换。

3D 显示是目前的一个研究热点[15]。3D 显示技术也已经发展了很多年，早在上个世纪人们就开始研究 3D 显示技术，随之而产生了 3D 电影等。但是当时的 3D 显示设备，不论是显示器还是电影院中的大屏幕，人们都需要佩戴特制的眼镜才能看到 3D 立体效果。随着人类需求的提高和技术发展，裸眼 3D 技术诞生。所谓裸眼 3D 即人们不需要佩戴眼镜就可以看出 3D 立体效果。目前已经有包括三星在内的多家显示器厂商都推出了这种类型的 3D 显示设备。但是该技术也存在着一定的缺陷[16]，即人们在观看屏幕时，必须位于一定的范围内才能观察到立体画面，若距离屏幕位置太远，或观察角度太大的时候，3D 效果并不明

显；此外，若离屏幕距离太紧，人会有明显的头晕现象，暂时还不适合在小尺寸显示器上使用。

对于 3D 显示技术来说，在终端显示之前需要先进行 3D 图像的合成，这就需要提前获取到当前时刻的所有视点的信息，所以其同样要求在终端能够快速解码。

### 1.2.2　多视点视频编解码技术

多视点视频技术的核心即为多视点视频编解码技术(MVC：Multi-view Video Coding)，这是由于多视点视频自身数据量庞大等特点决定的。由于用来表示图像和视频信息的大量数据彼此之间存在高度的相关性，这些相关性会引起信息的冗余，通过一些方法去除这些冗余信息从而可以实现了对视频数据的压缩，这就是视频压缩的基本原理。多视点视频中的信息都是关于同一个场景的，所以其视点之间将存在着大量的信息冗余，能够充分消除这些冗余信息是 MVC 压缩效率的关键所在。

对于视频编解码来说，目前有多种编码方式。最传统的方法为/基于波形的编码，伴随着视频编码相关学科及新兴学科的迅速发展，也出现了对象编码、分布式编码等新的压缩工具。

传统的基于波形的编码一般采用预测编码和变换编码相结合的混合编码框架，其主要思想是将视频的每一帧划分成多个块，通过基于块的预测和变换技术来充分减少视频序列中的冗余信息。H.261、MPEG-2、H.263、H.264、AVS 等都采用这种方法。

对象编码（Object-Based Video Coding）[17,18]，即基于图像中的内容进行编码，它将场景中所有对象进行分割，然后对这些对象进行编码、存储、传输和组合。目前 MPEG-4 标准采用此种编码方式[19]。这种编码方法压缩效率高，具有交互性，适于交互式视频服务以及远程监控。然而目前对象编码的发展出现了瓶颈。由于对象分割本身就是一个多义的问题，涉及对视频内容的分析和理解，计算机还不具有观察、识别、理解图像的能力，特别是一些复杂物体(例如人的肢体、头发等)很难识别。因此，尽管 MPEG-4 框架已经制定，但至今仍没有通用的有效方法根本解决视频对象分割问题，视频对象分割被认为是一个具有挑战性的难题。

分布式视频编码 DVC[20,21,22]（Distributed Video Coding），基于

Slepian-Wolf 理论和 Wyner-Ziv 理论，对两个或多个独立同分布的信源进行独立编码，然后由单一解码器利用信源之间的相关性对所有编码的信源进行联合解码。它与传统的视频编码技术的区别在于：传统编码方式通常都在编码端挖掘视频中的冗余信息，因此编码复杂度比较高；而分布式视频编码具有编码器复杂度低、编码端耗电量低、容错性好等特点。

对于当前的研究热点---多视点视频来说，其编解码可以采用传统波形编码方法，也可以采用上述其他新的编码工具。无论采用哪种方法，目前其都主要围绕着如何提高压缩效率以及随机读取能力进行研究。

基于传统波形编码的研究一般采用多级预测结构的方式来进行编码压缩，文献[63]即采用这种方法，同时为了提高压缩效率，其将帧内编码的图像和几何模型联合用来预测中间帧。另外也有基于子图的编码方式和基于模型的编码方式。文献[64]中提出针对同心圆拼图的基于子图的压缩，子图由所有图像中心的狭长片段粘连生成，实际上可粗略看作是柱状全景图，子图采用帧内编码，所有采集到的图像基于子图进行预测编码，这种方案具有良好的随机存取性能，同时具有良好的压缩性能。在文献[65]中，图像被反向映射到几何模型作为视点相关的纹理图，然后通过四维小波编解码器进行编码。基于模型的多视点视频压缩方法常常具有非常高的压缩比，然而其性能依赖于几何模型的好坏。

三维小波编码是另一类多视点视频编码方法[66]。文献[67]在同心圆拼图上应用了三维小波编码技术，由于横向帧滤波效率低下，其压缩比仍然不令人满意，这与三维小波视频编码方面发表的成果相一致。文献[68]提出一种高级帧对齐算法(smart-rebinning)，把同心圆拼图数据重组为一组全景图，这种对齐处理极大地增加了横向帧之间的相关性，并且有助于达到非常高的压缩比。

对于多视点视频编码的研究目前存在的一个很主要的问题就是缺少进行编码效率评价的理论模型。对于解码的多路视频合成的中间视点或虚拟视点来说，由于没有该视点的真实视频数据而无法用诸如 PSNR 等方法进行客观比较，这就需要将传统视频编码的率失真理论和模型扩展到多视点视频编码，建立多路视频编码的码率分配、附加的三维信息（摄像机参数、视差等）对解码质量以及合成质量的数学模型，研究多路视频传输中的差错与丢失对解码视频质量的影响。这方面的研究也是目前多视点视频编码亟待解决的问题。

为了能够对多视点视频编码方法进行规范，目前由 MPEG 和 ITU-T 组成的联合视频小组 JVT 致力于起草多视点视频的压缩标准。JVT 采用多级预测结构

的传统波形编码方式，在具有高压缩性能的 H.264/AVC 标准的基础上，根据多视点视频本身的特点，引入了一些新的预测工具和技术进一步提高压缩效率，同时降低解码的复杂度，为多视点的操作增加了一些功能。

由于多视点视频编解码的未来应用比较广泛，除了要实现较高的压缩效率外，还涉及到可扩展性、低延时编解码等问题，为此，JVT 根据多视点视频编解码应用的实际需求，对多视点视频编解码算法提出了如下 15 项需求[10]：

1. 压缩效率(Compression efficiency)
2. 已有视点的可扩展性(View scalability)
3. 生成视点的可扩展性(Free viewpoint scalability)
4. 空间/时间/质量 SNR(Signal to Noise Ratio) 的可扩展性(Spatial/Temporal/SNR scalability)
5. 向后兼容性(Backward compatibility)
6. 低延迟(Low delay)
7. 鲁棒性(Robustness)
8. 分辨率，比特深度，色度采样格式(Resolution, bit depth, chroma sampling format)
9. 视点间的图片质量(Picture quality among views)
10. 时间上的随机访问(Temporal random access)
11. 空间上的随机访问(Spatial random access)
12. 视点上的随机访问(View random access)
13. 资源管理(Resource management)
14. 资源消耗(Resource consumption)
15. 并行处理(Parallel processing)

高效的视频压缩是多视点视频存储与网络传输的前提，编码端多路视点的数据必须利用多摄像机之间的空间冗余大幅度降低用于传输和存储的码流。

为了能够在不同的终端和不同网络条件下，对同样的多视点视频内容进行显示交互，多视点视频编码需要支持不同类型的可扩展性，包括各种视点扩展性、质量（SNR）可扩展性、空间可扩展性、时间可扩展性等。比如视点扩展性，在编码端需要编码框架和算法能够随视点数量增加自适应扩展，在解码端计算能力不足或网络带宽有限的情况下可以只解码显示其中一路视频的内容。

多视点视频编解码还需要支持低的编解码延迟，用户视点切换延迟，端到端

延迟等。

为支持在网络尤其是无线网络上传输分发多视点视频内容，多视点视频编码需要提供容错或错误隐藏的机制，在传统的视频容错编码技术基础上引入空域关联的视点差错控制与恢复算法；

多视点视频编解码需要支持多种空间分辨率，包括 QCIF(176×144)，CIF(352×288)以及高清 HD(High Definition)等。多视点视频除了能够支持每个像素用 8bit 来表示的方法外，还应该支持更多比特来表示的方法。

由于压缩有损失，多视点视频编解码要求在解码端同一时刻恢复的不同视点视频应当具有可接受的相似视觉质量，不能因为编码端消除相邻视点间的空间冗余而导致在解码端，不同视点间存在明显的质量差异。

由于用户具有切换视点、在场景中漫游的交互能力，所以就需要多视点视频编码除了支持时间上的随机访问外，还要支持对于视点的随机访问，以及某一视点内的部分解码与绘制合成。这是多视点视频编码不同于传统视频编码的一个特点。

多视点编解码要充分考虑资源消耗和计算复杂度，编码框架一方面要适应多路视频具有空间相关性的特点，在视点增加时能够有效利用系统资源并具有并行性，同时编码方法还要考虑解码端基于视点合成的解码需求，能够在有限增加解码系统复杂度的前提下支持用户进行交互选择视点和观察方向。

多视点视频编解码需要能够支持对不同的视点进行并行处理，从而有利于其编解码器的实现，提高其处理速度。

以上这 15 项编码压缩需求（也即目标）将是未来 MVC 发展的趋势与方向，并依赖于新型的编码工具，新型的预测工具以及预测结构的探索。

### 1.2.3　多视点视频编解码并行优化的研究现状

多视点视频编解码 MVC 的数据量庞大，算法复杂度高，其计算量是以往编解码算法的好几倍，如图 1-5 所示。所以在 MVC 的需求文档中并行处理作为其目标之一，以便于 MVC 标准制定后，能够具有可接受的处理速度，使其在实际中尽快得到广泛应用。

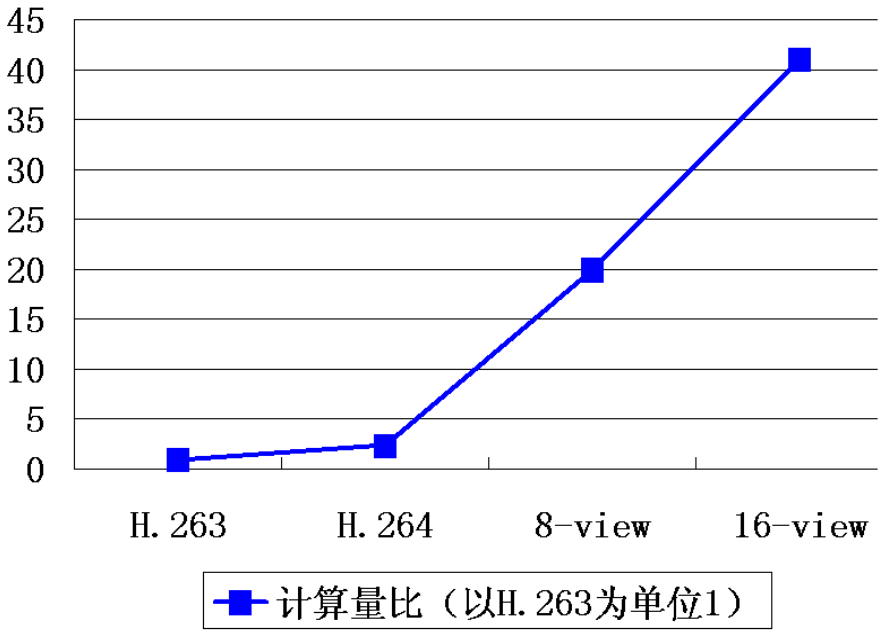



图 1-5 编解码标准的计算量

对于视频编解码并行加速的研究，随着视频编解码标准的出现，就已经开始。加州伯克利分校、普渡大学、香港大学等对早期的 MPEG-1 和 H.261 的加速系统进行过相关研究[24-27]，对于 MPEG-2 后续也有所研究[27-29]。随着视频数据的增多，视频压缩需求不断增加，视频编解码算法的复杂度也越来越高，相应的处理速度不断下降，无法满足应用需求，阻碍了视频编解码技术的发展和应用，于是视频编解码加速成为人们的研究热点，目前有很多人都在研究关于 MPEG-4[30-32,44]和 H.264 [33-43]的加速问题。

而多视点视频编解码是最近新兴起的领域，目前主要围绕如何提高压缩效率以及随机读取能力进行研究，所以关于它的并行加速的研究还比较少[45-48]。但是从以往视频编解码标准的发展状况可以看出，随着 MVC 标准的逐步制定，对其并行加速将成为未来的研究焦点。关于视频编解码并行加速的研究情况参见表 1-1：

表 1-1 视频编解码算法并行加速的研究状况

| | MVC | H.264 | MPEG4 | MPEG2 | H.261 | MPEG1 |
|---|---|---|---|---|---|---|
| Berkeley Purdue | | | | | | [24] |
| 香港大学 | | | | | [25] | |
| GaTech | | | | [26] | | |
| Philips | | [33,34] | [30] | [29] | | |
| CSELT | | | [31] | | | |
| NTU | | [32,34,35] | [32] | | | |
| 汉诺威大学 | | | [44] | | | |
| TUD | | [36] | | | | |
| NCTU | | [37] | | | | |
| 三星 | | [38,39] | | | | |
| NEC | | [40] | | | | |
| Intel | | [41] | | | | |
| IBM | | [42] | | | | |
| Nokia | [45] | [43] | | | | |
| 中科院 | [44] | | | | | |
| 清华大学 | [47,48] | | | | | |

虽然关于多视点视频编解码并行加速的研究比较少，但是无论是对哪一种编解码算法进行并行加速，其并行方法都是通用的。在并行处理方法中，根据并行粒度(粗粒度和细粒度)和并行方式(数据并行和控制并行)的不同[40]，可以将其主要分为如表 1-2 所示的四大类：

表 1-2 视频编解码算法的并行加速方法

| 并行粒度 / 并行方式 | 粗粒度 | 细粒度 |
|---|---|---|
| 数据并行 | 以宏块(MB)、片(Slice)、帧(Frame)、图片组(GOP)作为并行粒度 | 单指令多数据流(SIMD) |
| 任务并行 | 编码的步骤流(DCT, ME/MC 等)，专用加速模块 | 超长指令字(VLIW)，超标量技术(Superscalar) |

数据并行指的是对数据进行分解，将相互之间不存在依赖关系的数据分配到不同处理器上并行处理。

粗粒度的数据并行方法主要以宏块(MB)、片(Slice)、帧(Frame)或图片组(GOP)作为数据并行的粒度。在针对 MPEG-4 和 H.264 的加速系统中，大多以 MB 为并行粒度[26,30,37,42]，因为这样可以提供比较多的可并行执行的线程。但以 MB 作为粒度比较细，增加了调度开销，同时还引入了同步问题，所以也有一些以 slice 为并行粒度的[36]。以帧和图片组作为并行粒度的方案比较少，因为对于当前的大多数视频编解码算法来说，由此得到可并行性比较低，可扩展性差。

细粒度的数据并行目前主要包括单指令多数据流(SIMD：Single Instruction Multiple Data)技术。它采用扩展指令集的方法，把媒体处理指令扩充至通用处理器的指令集中。这种方法在细粒度并行方面效果显著，对于视频处理算法的加速比可以达到 24 倍[41]。

任务并行指的是将任务进行分解，在不同的处理器上完成不同的任务。

粗粒度的任务并行主要指的是对编码中的 DCT、运动预测、运动补偿等步骤采用流水线技术进行并行处理。均衡各步骤地运算量，以 H.264 为例，绝大多数的计算量集中在运动预测和运动补偿部分，所以针对这部分集中进行并行优化处理，可以得到较好的性能提升[26,30,31,33,34,35,37]。

细粒度的任务并行目前主要包括超长指令字(VLIW：Very Long Instruction Word)技术和超标量技术(Superscalar)。这两个技术的目的在于提高处理器性，并不是专门针对视频编解码并行处理提出的，其能够实现并行加速的作用，提高的比较有限，一般与其他方法混合使用[26,34,37,50]。

上述四种方法可以单独使用，也可相互结合使用来进一步提高加速比。例如可以将数据并行和任务并行相结合，在粗粒度数据并行的基础上，当 CPU 空

闲时，对数据单元内部采用进一步的任务并行；也可以采用粗粒度和细粒度数据并行相结合的方法，在指令级利用 SIMD 优化之后，采用宏块级的数据并行。

## 1.3　论文各部分的主要内容

本文的目的主要是采用相应的并行技术充分发挥多核处理器的并行处理能力，来提高多视点视频编解码的处理速度。为此文中首先提出了一种多视点视频编解码的可并行性分析模型，通过该模型设计分析得到适合多视点视频的启发式并行调度算法，根据这些算法分别对多视点视频的编码器和解码器进行了并行优化，提高其处理速度。同时为了充分发挥多核优势，进一步提高多视点视频编解码的加速比，从 MVC 自身的预测结构出发，根据不同的应用设计了不同的预测结构，以提高其可并行性。

整篇文章包含五章内容。第一章主要对多视点视频技术、多视点视频编解码技术及其并行加速技术进行了调研，介绍了相关领域的发展情况；第二章首先对多视点视频编解码中的相关理论知识和主要技术进行了分析介绍，并根据 MVC 的特点，建立了一种 MVC 可并行性的分析模型，用来分析 MVC 算法的可并行性以及 MVC 并行调度算法的加速性能；第三章基于分析模型，针对 MVC 的特点设计了两种启发式调度算法，并在多核处理器平台上利用这两种算法分别对多视点视频编码器和解码器进行并行加速；在文章的第四章中，分析了当前多视点视频编解码的可并行性，从提高可并行性的角度出发，根据其未来的不同应用方向，提出了两种改进的预测结构；第五章对全文进行了总结。

# 第2章　MVC 可并行性的分析模型

## 2.1　多视点视频编码框架

多视点视频的压缩原理和其他视频一样，通过消除视频在空间、时间、结构、视觉等上的冗余信息来减低码流。通常用于表示图片和视频的数据在图像内部存在着空间相关性，相邻图像（帧）之间也存在时间相关性，这些相关性造成信息的冗余，视频编码的目的主要就在于通过一些方法去除这些冗余信息，从而实现对图片和视频数据的压缩。

传统的视频编解码方法多采用预测编码和变换编码相结合的混合编码框架。在该框架下，首先对视频数据进行预测编码，消除视频在时间域的冗余；然后对预测编码后的结果进行变换编码、量化和熵编码，消除视频在空间域的冗余。

多视点编解码同样采取上述混合的编码框架，同时为了进一步提高压缩比，利用多视点视频自身的特点，在预测编码中提出了一些新的技术。

### 2.1.1　变换编码(Transform Coding)

变换编码对信号进行一种函数变换（通常采用正交变换），将其从一个信号域(例如：空间域)变换到另一个信号域（例如：频域），使得在该信号域下信息的相关性减低，减少了冗余信息，从而实现压缩。由于采用正交变换，所以对得到的新信号进行逆变换，可以对原信号进行还原。

常见的变换有 K-L 变换、离散傅里叶变换(DFT)、离散余弦变换(DCT)、沃尔什(WALSH)变换等。其中 K-L 变换是最优的正交变换，去相关性最好，但是其计算过程复杂，变换速度比较慢。而离散余弦变换 DCT 具有很强的“能量集中”特性，大多数的自然信号(包括声音和图像)的能量都集中在离散余弦变换后的低频部分，而且当信号具有接近马尔科夫过程(Markov processes)的统计特性时，DCT 的去相关性接近于 K-L 变换的性能。所以目前大多数都采用 DCT 变换。

目前多视点视频采用 H.264/AVC 标准中的基于块的 4*4 整数变换，该变换的过程和 DCT 变换基本相似，但是其采用的是整数操作，而不是实数操作。这种方法一方面减少了反变换时的浮点数计算误差，另一方面可以将变换编码和

量化一起进行，加快了计算速度。

### 2.1.2　熵编码(Entropy Coding)

利用信源的统计特性进行码率压缩的编码方式称为熵编码，也叫统计编码[58]。熵编码是一种无损数据压缩算法，编码过程中不会丢失信息量。预测编码和变换编码用于去除视频数据在内容上的冗余，而熵编码对预测、变换、量化后得到的系数和运动信息，进行统计，来消除空间域的信息冗余。

视频编码中常用的熵编码有两种：变长编码和算术编码。变长编码依据符号在序列中出现的概率的大小来分配相应长度的码字，出现概率越大，码字越短，从而有效的实现数据上压缩；算术编码采用一个浮点数来表示一串输入符号。在多视点视频编解码算法中，其有两种熵编码方法可以供选择。

由于熵编码和变换编码都是用来消除空间域的冗余，其不是专门针对多视点视频的特性来设计的，而且这两种编码的可并行相对比较小，所以不是本文研究的重点，相关内容不在此赘述。

### 2.1.3　预测编码(Prediction Coding)

预测编码用来消除视频时间域上的冗余。离散的信号之间存在着一定相关性，这就是预测编码的压缩基础。它的基本思想是利用前面一个或多个信号来预测下一个信号，对预测值和实际值作差（预测误差），将此差值替代原信号。如果预测的比较准确，那么预测误差就会很小，在同等精度要求的条件下，就可以用比较少的比特来表示该信号，从而能够达到压缩数据的目的。预测编码包括线性预测和非线性预测两种，目前多采用线性预测编码方法。

对于一幅二维静止图像，根据该图以前已出现的像素点的值来对当前像素点的值进行预测，从而消除图像中在空间上的冗余度。这就是视频压缩中的帧内预测（Intra -Prediction）。

对于视频来说，它是由时间上连续的图像组成的序列，一般为每秒钟包含 30 帧(或 25 帧)图片。大多数电视图像相邻帧间的细节变化很小，据统计，相邻两帧只有 10%以内的像素有变化，亮度信号(Y)有 2%的变化，色度信号(UV)有 1%以内的变化。所以视频在时间上比在空间上具有更大的相关性，对视频的帧与帧之间采用预测编码，能获得比帧内预测更高的压缩比，于是出现了帧间预测（Inter -Prediction）的方法。帧间预测的性能是影响视频压缩性能的重要因素。

直接预测对于静止图片的压缩效果比较好，但对于视频来说，更多的是运动的物体，直接预测就达不到很好的效果。于是在帧间预测的基础上，引入了运动补偿(Motion Compensation)技术。运动补偿技术通过跟踪画面内的运动情况，计算相应的运动向量，对其加以补偿之后再进行帧间预测，这样可以更加准确的找到其预测值，降低预测误差。

目前预测编码已经发展的比较成熟，一般包括以下几个步骤：首先将图像分解成相对静止的背景和若干运动的物体，各个物体可能有不同的位移，但构成每个物体的所有像素的位移相同，通过运动估值得到每个物体的位移矢量；然后利用位移矢量计算经运动补偿后的预测值；最后对预测误差进行量化、编码、传输，同时将位移矢量和图像分解方式等信息送到接收端。

由于多视点视频信号是由多个视点相机采集获得，它们之间因为相机位置的不同而存在差异，但拍摄的视频内容都是同一个场景的，所以存在着一定的相关性，其相关性的大小与相机陈列的形式、相机与拍摄对象的距离以及相机间距有着很大的关联。预测编码可以很有效的减少这种相关性，于是预测编码成为多视点视频编解码的重要研究方向。

为了消除视点间的信息冗余，多视点视频编解码在预测编码中引入了视点间预测（Inter-View Prediction）的方法。关于视点间预测的技术很早就已经提出，在过去传统立体视频（stereoscopic video，左右两视点的立体视频）已采用视点间预测和时间预测相结合的方法，其相关的标准规范已经在文档[51][52]中给出。但当时是基于 H.262/MPEG-2，利用其分层编码的方法，将传统立体视频的左右通道视频放入到基本层和增强层进行编码，增强层通过基本层来预测，只针对有两个视点的立体视频的编码，应用范围受到限制。

随着 3DTV 的兴起，多视点视频的预测编码也成为研究热点。目前多视点视频的预测编码的研究主要在于预测结构和预测工具的研究[9]。

#### 2.1.3.1　预测工具

预测工具是用来提高多视点视频各个视点之间预测的准确性的工具，它是 MVC 的压缩效率能否显著提高的关键。目前提出的工具包括亮度补偿(Illumination Compensation)、视差/运动补偿(Disparity/Motion Vector Compensation)、视点插值(View Interpolation)、2D 直接预测模式等[9]。

1. 亮度补偿

由于多视点视频的各个视点视频之间存在着亮度的差异，所以需要在预测之前进行亮度补偿，来提高预测的准确性。目前有基于帧和基于块的两种亮度补偿方法。

2. 视差/运动补偿

多视点视频中当摄像机的位置固定时，在一个短的时间间隔内如一个 GOP，视差关系趋于相似的，因而可以利用这一近似，在同一 GOP 中使用已编码的视差矢量作为该 GOP 中待编码图象的预测矢量。同样当摄像机位置靠近时，相邻视点同一 GOP 内的运动信息也具有相似性，可以用其它视点中已编码的运动矢量作为当前帧中同一时刻待编码图象的预测矢量。这种复用可以减少运动矢量和视差矢量的编码开销。

3. 视点插值(View Interpolation)

一种针对多视点视频提出的新的预测工具。其主要思想是利用相邻两个视点进行插值，将插值预测的结构作为中间视点的信息。例如我们使用传统的编码标准(H.264/AVC)对多视点视频序列中的 1,3,5…序列进行编码，而序列 2,4,6…的预测图象通过序列 1,3,5….相邻两个之间进行插值得到。

4. 2D 直接预测模式

传统视频编码的帧间预测，只在时间上进行参考，其只有沿时间轴的一维预测方向模式，而多视点视频由于具有相邻视点的空间关联性，因而其帧间预测的参考帧可以是不同视点的视频帧，即二维方向模式。

#### 2.1.3.2　预测结构

预测结构指的是视频编码中帧与帧之间的参考关系，包括时间上的以及视点间的。预测结构指出了视频序列中哪些帧进行帧内预测，哪些帧进行时间和视点上的帧间预测，且其分别参考哪些帧。预测结构的设计不仅会影响压缩性能、还将影响随机读取性能、快速解码性能、网络传输代价等，因而在 MVC 研究中受到广泛关注。

关于多视点视频编解码的预测结构，也有多种提案。目前在 JVT 提出的多视点视频编解码标准 MVC 中，其在 H.264/AVC 编码中具有高压缩性能的分级 B 帧预测结构(hierarchical B pictures)的基础上，加入视点间的参考关系。MVC 中建议的预测结构图如下图 2-1 所示[6]：

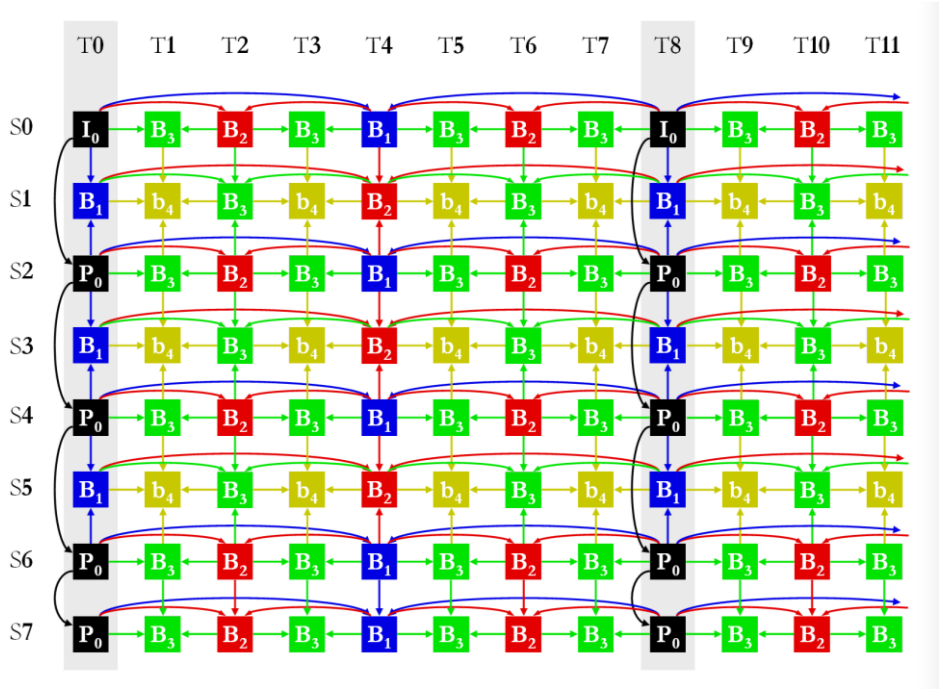



图 2-1 时间和视点间预测的预测结构(采用分级 B 帧的结构)

图中以 8 个视点为例，每一行代表一个视点的视频序列，各个视点按照摄像机的排列，依次排列，每一列代表同一时间上不同视点的视频信息，箭头表示参考关系，箭头出发的帧被箭头指向的帧作为参考。

一个预测结构即为多视点视频编解码的一个处理周期，称为 GOP 组(Go-GOP：Group Of -Group Of Pictures)，大小为视点数×单个视点的 GOP 大小。关于 GOP 的大小，MVC 中可以根据用户需求自己设定，实际中常用值为 8、12 和 15。（本文中不作说明均采用 GOP 大小为 8）

多视点视频编解码中的帧类型和 H.264/AVC 标准中的一样，包含 I、P 和 B 三种帧类型。

I 帧（图 2-1 中的 I0 帧）采用帧内预测的方法进行编码。其包含了视频的详细信息，数据量相对比较大些，在解码时只需要本身的数据即可重构得到原始图像。由于 I 帧包含了视频的具体信息，并为其他类型的帧所参考，所以 I 帧也被称为关键帧(Key Frame)。

P 帧为单向预测帧，对于多视点视频来说，其主要利用 I 帧或 P 帧，通过视点间预测进行编码，只保留预测插值，其压缩率相对 I 帧要高；而 B 帧为双向预测帧，其在编码时可参考前后两个方向上的帧，对于多视点视频来说，其还

可以参考同一时刻相邻视点上的帧，相比于 P 帧，其预测值更加准确，压缩效率更高。在 MVC 中，根据 B 帧在预测中的分级关系，又分为 B1，B2，B3…等帧，其下标号表示通过 I 帧和 P 帧，预测到该帧所需的级数。

对于多视点视频编解码来说，其任务主要包括预测编码、变换编码、量化、熵编码以及文件读取等其他额外开销。图 2-2 为 H.264 的参考软件中各任务所占用的处理时间的分布情况，在传统编码框架下，预测编码共占用了 55%的时间，是运算量最大的任务；而熵编码只占 9%，变换编码占 12%。所以要提高编码算法的处理速度，关键是提高预测编码的处理速度。

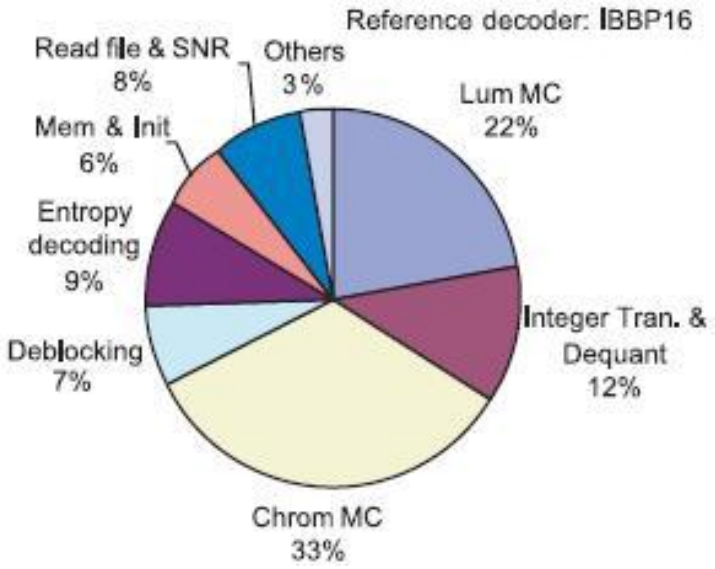



图 2-2 H.264 参考软件中各任务所占用的处理时间

## 2.2　基于数据并行的 MVC 任务调度模型

### 2.2.1　数据并行模型

多视点视频编解码的计算量非常的庞大，编码和解码速度都非常的慢，而普通处理器的发展出现了瓶颈，很难满足其计算需求，多核处理器成为体系结构的发展趋势。要想对 MVC 进行加速，可以引入并行，充分利用多核处理器的优势。而影响并行性能的主要是并行调度算法的研究。

所谓并行调度即根据某一并行粒度，将一个任务分解为若干个子任务，根据这些子任务之间的依赖关系，确定可同时并行处理的任务，按照某种策略将这些任务映射到相应的处理器上。其包括任务分解和任务调度策略两部分。

任务分解即依据数据依赖关系或者任务依赖关系，根据并行粒度，将整个任务划分为可并行执行的子任务，并给出子任务间的相互依赖关系。其中对于并行粒度的选择，一方面要使得任务划分后同一时刻有尽量多的子任务可以执行，另一方面也要考虑到尽量减少任务与任务之间的交互，降低调度开销。

调度策略指的是如何为可并行执行的子任务分配系统资源，从而满足某一调度目标。调度目标主要指的是负载平衡，尽量让所有的处理器都不空闲。

目前常用的并行调度算法的设计模型主要有数据并行模型、分治策略模型、工作池模型、主/从（Mater-Slave）模型、流水线模型等[69][70]，由于多视点视频其数据量比较庞大，数据并行是最简单的一种并行模型。该模型根据数据将整个任务分解为若干个相同的子任务，这些子任务被分配到每个处理器上，对不同的数据进行处理。这种方法设计起来比较简单，但是需要用户将应用程序分配到不同的处理器上。

### 2.2.2　MVC 的分层有向无环图(DAG)任务调度模型

我们利用数据并行的并行模型设计了一种 MVC 的任务调度模型。

首先在并行粒度的选择上，本文采用粗粒度的方法。以往的视频编解码加速算法多以宏块 MB 和片 slice 为并行粒度，而对于多视点视频来说，其数据量庞大，采用宏块 MB 等比较小的并行粒度，会使得调度算法过于复杂，增加开销。由于多视点视频本身数据量比较大，采用帧一级并行可得到比较充足的并行度[47]，同时还简化了调度算法，所以本文采用帧一级的数据并行方法，将一帧的处理作为一个任务调度的对象，依据帧与帧之间的数据依赖关系，对 MVC 的一个处理单元 GOP 中的所有帧进行优化调度。

构造任务的有向无环图 DAG（Directed Acyclic Graph）可以我们选择如何发掘串行程序中存在的并行性,一个好的方法就是构造其对应的并行任务(DAG)图,对并行任务图的分析,调度和最终实现并行计算的工作已经有很多人做过,

图 2-1 中 MVC 的预测结构图即为表示 MVC 数据间关系的有向无环图 DAG（Directed Acyclic Graph）。为了能够更清晰的表示在 Go-GOP 中各个帧之间的数据依赖关系，我们对 MVC 的预测结构图进行分层处理。

假设 MVC 包含 n 个视点，GOP 大小为 m。MVC 的分层 DAG 图为 G=(V，E)，其中节点集 V 和边集 E 的定义如下公式 2-1 所示：

$$V=\{v_{ij}\,|\,1\le i\le n,1\le j\le m\}$$
$$E=\{e(u,v)\,|\,u,v\in V\} \qquad (2\text{-}1)$$

其中 V 为图 G 的节点集合，每个节点代表 MVC 预测结构中的一帧；E 为图 G 的边集合，对于 MVC 中任意两帧 $v,u\in V$，如果帧 u 的处理依赖于帧 v，则 $e(v,u)\in E$。

我们通过公式 2-2 的定义将图 G 的节点集合 V 分为若干个子集。其中 $D(v)$ 为 $v_i\in V$ 的父节点集合，它表示帧 $v_i$ 所依赖的帧的集合。

$$V_0=\{v\,|\,v\ is\ an\ I\ frame, v\in V\}$$
$$V_i=\{v\,|\,D(v)\subset\bigcup_{j=0}^{i-1}V_j, v\in V\},\ i=1,2,3.....L-1 \qquad (2\text{-}2)$$

这样节点集合就被划分为 L 个子集。本文将 L 作为图 G 的层数，子集的下标号作为层号，将子集中的节点放到对应的层上，这样 MVC 中的帧就被分到不同的层中，从而得到分层的 DAG 图。以 GOP 大小为 8，有 8 个视点的多视点视频编解码为例，其分层 DAG 图如下图 2-3 所示：

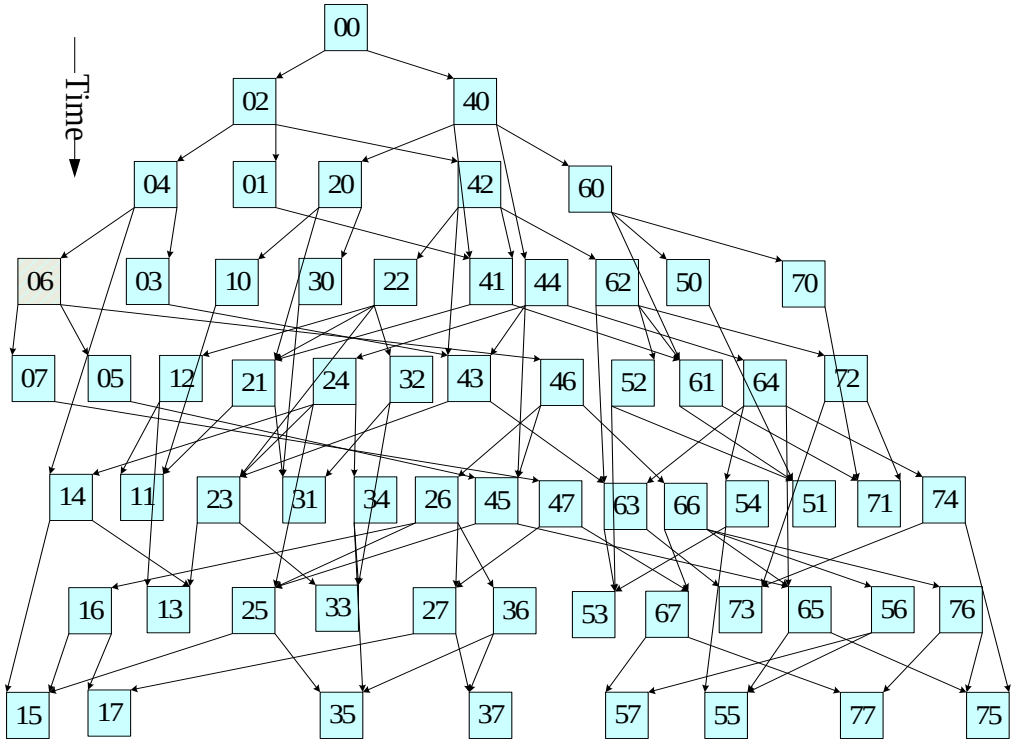



图 2-3 8*8 MVC 的分层 DAG

其中节点上的标号表示该帧在 MVC 预测结构中的位置。例如“02”表示其为预测结构中第 0 视点的第 2 时刻的帧。

通过 2-2 公式可以看出，对于每个子集中的帧来说，由于其所依赖的帧都在其上层集合中，所以子集内的帧之间不存在相互依赖关系，那么就可以作为并行调度的对象。

### 2.2.3　基于拓扑排序的分层 DAG 自动构造

我们利用拓扑排序的方法来实现多视点视频编解码 DAG 图的分层构造。拓扑排序将一个有向无环图 G 中的所有节点排成一个线性序列，使得图中任意一对节点$u,v \in V$，若$e(u,v) \in E$，则 u 在线性序列中出现在 v 之前。拓扑排序后的结果即对该 DAG 图进行了分层。

拓扑排序的方法有很多，我们主要采用基于入度的排序方法[71]。其算法思想为，首先在 DAG 图中选择一个入度为 0 的节点，该节点至少存在一个，否则表明该图存在回路。将该节点放入到线性序列中，并在图中删除该节点以及相应的边，如果图中还存在入度为 0 的节点，则重复上述操作，直到没有入度为 0 的节点。经过上述操作后，DAG 图最终变为空，完成拓扑排序。如果此时该图还不为空，则表明该图存在回路。

该算法的伪代码如下表 2-1 所示：

表 2-1　DAG 图拓扑排序算法的伪代码

```
for 每个节点 u∈V          //统计每个节点的入度
    for 每个节点 v∈V
    begin
        if e(v,u)∈E
            d(u) ← d(u)+1;
    end
CreateStack(s);           //创建一个堆栈 s，用来存放 DAG 中入度为 0 的节点
for 每个节点 u∈V          //对堆栈 s 进行初始化，将初始入度为 0 的放入 s 中
   if d(u) = 0
      将 u 放入到 s 中
//拓扑排序
```

```
while(s 非空)
begin
    从栈顶取出一个节点 u，并将其从栈中删除;
    将 u 放入到排序结果序列 R 中;
     for 每个节点 v∈V
          if e(v,u)∈E
          begin
                d(v) ← d(v)-1
                if d(v) =0
                   将 v 放入到 s 中;
          end
end
```

## 2.3　MVC 可并行性的分析模型

多视点视频编解码本身的预测结构等是否具有可并行性，其并行的性能如何，哪种调度方法可以充分利用其可并行性，在多核处理器上有比较优的加速效果…这些都是研究 MVC 并行优化所面临的问题。但是一方面并行调度的测试需要多核硬件平台的支持，而目前尚无法提供这么多的硬件平台；另一方面，MVC 编解码算法复杂度比较高，处理时间和消耗比较大，不方便进行测试。为此我们专门设计了一个分析模型，抽象不同的硬件平台，对 MVC 在不同规模核数的硬件平台上的运行进行模拟，从而方便我们对 MVC 本身可并行进行分析，同时了解 MVC 的各种并行调度算法的加速性能。

该模型基于 MVC 的分层 DAG 图来实现的，其包括任务调度模块、编码模拟器模块和多核处理模拟器模块三部分，这三部分依据 DAG 图来相互协调工作。

### 2.3.1　任务调度模块

任务调度模块启动一个任务调度线程，主要负责管理一个就绪队列变量 canEncodings，依据某种调度准则在该队列中选择一帧作为调度对象，并将这一帧的信息传送到多核处理模拟器模块中。在该模块中，其包含一个函数

GetNextProcessing()，用户可以在该函数中设计实现不同的调度算法。

### 2.3.2　MVC 编解码模拟器模块

编解码模拟器用来记录和更新MVC编码的相关信息，而不用具体实现MVC的编解码算法，其主要包括如下几个变量来表示分层 DAG 图的信息，以及其相应节点和边的权重：

1、 数组 depends 和 deps：分别用来记录每个节点的父节点集合和父节点数量
2、 数组 effects 和 effs：分别用来记录每个节点的子节点结合和子节点数量
3、 数组 runningTime：记录 DAG 图中每个节点的权重，即 MVC 处理时间该帧的数学期望值。
4、 数据 frameId： 标记 DAG 图中每个节点在 MVC 预测结构中的位置信息
5、 数组 frameTypes：标记 DAG 图中每个节点在 MVC 预测结构中的帧类型

通过前三个变量来表示MVC的DAG图，通过后两个变量将DAG图和MVC的预测结构中进行对应。在每次完成一个帧的处理后，通过修改对应 DAG 图中节点和边的信息，对 DAG 图进行更新。当 DAG 图中存在一个节点的父节点数为 0 时，将其对应的 MVC 中帧放入到就绪队列 canEncodings 中。

### 2.3.3　多核处理模拟器模块

多核处理模拟器模块可以在普通的硬件平台上，抽象出不同规模核数的硬件平台，而来模拟多核处理器的工作情况。在模型中通过变量 CoreNum 来设置多核处理器的核数；通过变量 totalTime 来模拟当前的系统时钟；通过一个 CoreStatus 结构的数组表示每个核的工作状态，该结构的定义如下：

```
struct CoreStatus
{
        double remainingTime;   //记录处理完当前的帧还需要多长时间
        bool isRunning;         //是否正在工作
        int FrameId;            //当前该核正在处理的帧所在的时间号
```

```
        int StreamId;              //当前该核正在处理的帧所在的视点号
};
```

利用这些变量来实现对各种硬件平台进行模拟，并实现对核的模拟调度。

### 2.3.4　模型的处理流程

MVC 可并行性分析模型的处理流程如下图 2-4 所示：

其中红色框为编解码模拟器模块，绿色为任务调度模块，蓝色框为多核处理模拟器模块。

首先对整个系统初始化，将编解码模拟器的 DAG 图的信息进行初始化，然后搜索是否有 deps=0 的帧，如果有则表明这些帧可以被处理，将这些帧的位置信息 FrameId（包括 TimeId 和 ViewId）和处理时间反馈给任务调度模块。

任务调度模块将这些帧的信息加入到就绪队列 canEncodings 中。判断就绪队列是否为空，如果为空则反馈给编解码模拟器，获取可执行的帧；若不为空，则根据某种调度准则在就绪队列中选择一帧进行调度，并将该帧的位置信息和处理时间传给多核处理模拟器模块中。

多核处理器模拟模块在得到这个信息后，对所有核进行搜索，如果存在核空闲，那么就将该帧分配到该核上，把该帧的相关信息更新到核的工作状态上，并将 isRunning 设为 true，表示其正在被用；如果没有核空闲，则在所有核中找一个 remainingTime 最小的核，把该帧分配到该核上，并将当前时刻 totalTime 向前推移 remainingTime 个单位，把该核上原来正在处理的帧信息(TimeId，ViewId)反馈给编解码模拟器模块。编解码模拟器模块收到该信息后对 DAG 图进行更新，然后重复上述步骤。

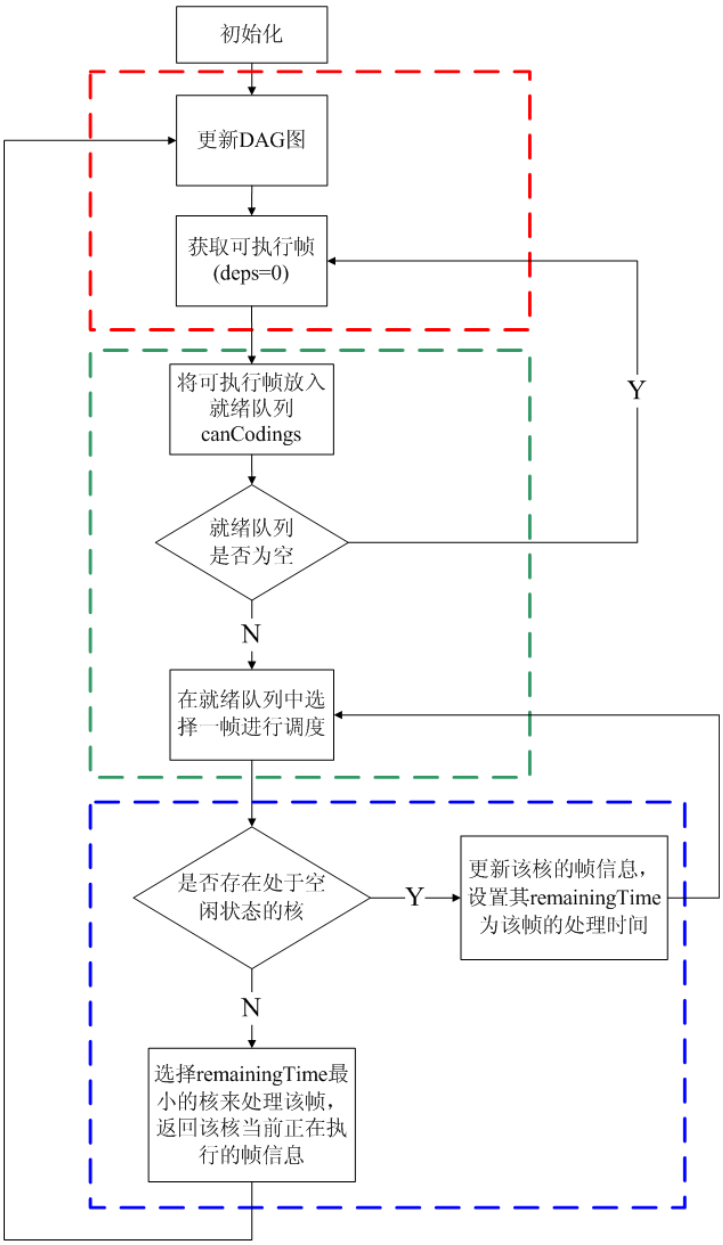



图 2-4 可并行性分析模型的处理流程图

## 2.4　实验结果

在 2.3 节中的分析模型上，我们实现了一种传统的并行调度算法，即时间优先的调度算法，来对我们的模型进行测试。时间优先调度算法是一种最简单的调度算法，每次调度时在就绪队列中选择 TimeId 最小(时间最早)的帧来进行调度。

通过实验我们得到了该调度算法的并行加速比性能，测试结果如下图 2-5 所示：

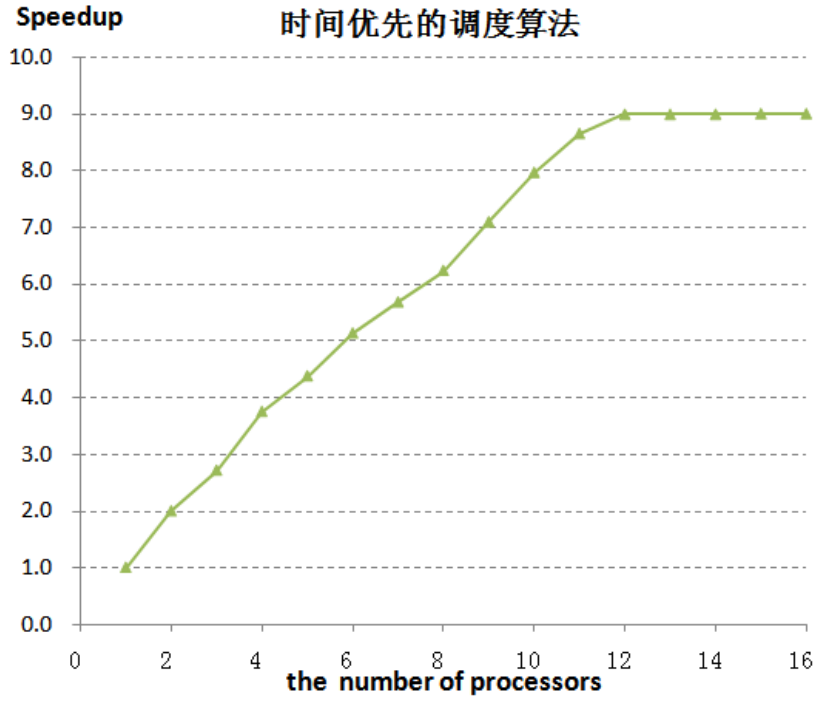



图 2-5 可并行性分析模型的处理流程图

# 第3章　MVC 并行优化的设计与实现

## 3.1　MVC 的启发式并行调度算法的设计

根据调度策略调度方式的不同，并行调度算法又可分为动态调度和静态调度两种。

1、静态调度算法

静态调度指在任务执行前已经为各个子任务确定了执行顺序和静态优先级，并且在整个执行过程中不再变动。静态优先级的分配可以根据应用的属性来进行，比如进程的周期，用户优先级，或者其它的预先确定的属性。

静态调度算法无论是对于单核处理器，还是多核处理器，一般是以单调比率算法 RM（Rate-Monotonic Scheduling）为基础的[53]。RM 算法根据每个任务的执行周期的长短来决定调度的优先级，那些具有小的执行周期的任务具有较高的优先级，周期长的任务优先级低。RM 是单核处理器下的最优静态调度算法。它的一个特点是通过对系统资源利用率的计算来对任务进行调度，算法简单、有效，易于实现。但是由于其优先级是静态的，没有很好的灵活性。

2、动态调度算法

动态调度[54，55]相对于静态调度来说，其所有子任务的优先级在执行前未知，在任务执行过程中根据各种需求动态地分配优先级，优先级随着时间变化而变化，这样使得在资源分配和调度时有更大的灵活性。

最简单的动态调度策略是先来先服务(FCFS：first come first serve)算法，它将所有准备就绪可执行的子任务按照提交的先后顺序放到一个队列中，每次为排在对头的子任务分配系统资源。这种算法自然、简单，系统开销小。但是它以相同的原则来对待所有的子任务，没有考虑到系统对每个子任务的不同需求，无法满足不同的调度目标。

比较流行的动态调度算法为最早时限优先算法(EDF：Earliest-Deadline-First)，其主要思想是对调度列表中的所有任务计算其在所有处理器的最早截止时间，在调度时，任务的优先级根据任务的截止时间动态分配。截止时间越早，被调度的优先级越高。

EDF 是目前使用最多的一种动态调度算法，理论上已被证明在单核处理器

下是动态的最优调度算法，而且是充要条件，处理机利用率最大可达 100% 。但是由于 EDF 在每个调度时刻都要计算任务的 deadline，并根据计算结果改变任务优先级，因此开销大、不易实现，其应用受到一定限制。而且在瞬时过载时，系统行为不可预测，可能发生多米诺骨牌现象，一个任务丢失时会引起一连串的任务接连丢失。

而在多处理器下，如果任务计算时间、截止时间或到达时间不能预先确定，则最优调度算法不存在。所以目前关于多核处理器下的调度算法的研究多为各种启发性算法和包含各类约束条件的近似算法。

对于多视点视频编解码 MVC 在多核处理器上的并行加速，本文根据其本身应用的特点，利用 2.2 中所讲的任务调度模型，为 MVC 设计了两种启发式并行调度算法。

### 3.1.1　基于最长剩余路径的启发式静态调度算法

定义 $w(v)$ 为分层 DAG 图中节点 $v \in V$ 的开销，它表示处理帧 $v$ 所需要的时间；$c(u,v)$ 为图 G 的边 $e(u,v)$ 的长度，它表示处理帧 $v$ 时，系统所需要的其他资源开销，包括等待 CPU 和 IO 开销等；定义每个节点 $v$ 到达根节点的最长路径为 $p(v)$，称为剩余最长路径。

那么对于上述分层的 DAG 图来说，由于同一层的帧之间没有依赖关系，可以并行执行，所以其并行处理的理论最短时间即为该 DAG 的最长路径。为此本文设计了一种启发式静态并行调度算法，其在调度时优先选择当前剩余最长路径最长的帧进行处理，从而加速对于 GOP 的处理，使其处理速度尽量接近其理论最短时间。

由于对于视频编解码算法来说，IO 等其他资源开销相对来说比较小，可以忽略不计，所以可以将 DAG 图中的边的权重 $c(u,v)$ 设置为 0，那么最长路径的计算主要就是帧的处理时间。

通过实验发现，对于同一个视频来说，其 I 帧、P 帧和 B 帧的处理时间基本上稳定在一定的比值上。根据对帧处理时间的统计结果，本文将 MVC 中的帧进行了重新分类，分为如下五种类型：

1、I 帧：帧内预测帧，关键帧；

2、P 帧：通过视点间预测得到的帧；

3、b0 帧：双向参考帧，其参考帧为 I 帧或 P 帧；

4、b1 帧：双向参考帧，其参考帧一个为 I 或 P 帧，另外一个为 B 帧；

5、b2 帧：多向参考帧，其参考帧至少有两个为 B 帧；

这五种类型的帧其处理时间基本上稳定在一个数量级上，本文以 I 帧的处理时间作为单位时间 1，通过统计其他类型帧在各种情形下的相对处理时间，得到其处理时间的期望值。用 golf1 多视点视频序列进行测试，得到如下图 3-1 所示的结果：

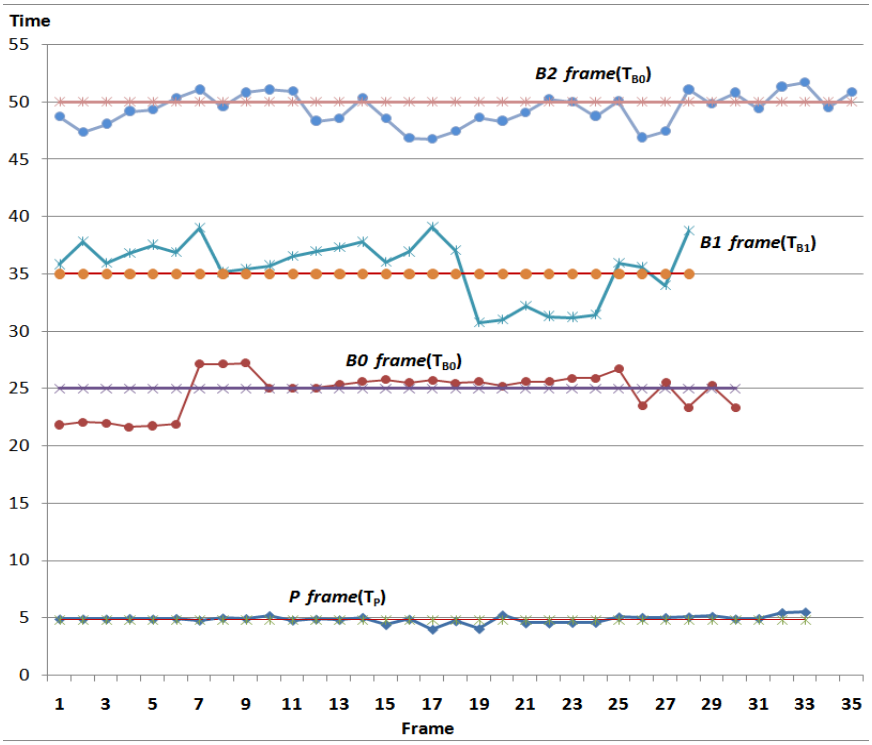



图 3-1　各种类型帧的处理时间

从而求得各种类型帧的处理时间的期望值，$T_I = 1$，$T_P=4.8$，$T_{b0}=25$，$T_{b1}=35$，$T_{b2}= 50$。将这些期望值作为每个节点的开销，那么 DAG 图中每个节点的最长剩余路径就可以得到。

启发式静态调度算法具体描述如下：

1、首先对序列中的多个 GOP 进行单核处理，得到各个帧的执行时间，利用这些结果求得不同类型帧的处理时间的期望值。将这些期望值作为 DAG 的节点的权重 $w(v)$，并根据它计算出每个节点的剩余最长路径 $p(v)$

2、当有 CPU 空闲时，如果就绪队列中有数据，则选取其中剩余最长路径 $p(v)$ 最大的一帧进行处理。

3、每处理完一帧后，若有新的帧可以处理，将其放入到就绪队列中。

由于 $p(v)$ 是在进行视频处理之前就确定了，而且不随着时间的变化而改变，所以该调度算法是静态的调度算法，调度开销相对也会比较少。

### 3.1.2　基于 DAG 出度的启发式动态调度算法

虽然静态调度算法额外开销比较少，但是其调度优先级没有灵活性，不能满足各种应用的需求，所以文中还提出了 MVC 的一种启发式动态调度算法。

从另一个角度来讲，要提高视频编解码的并行处理速度，就要提高多核处理器的利用率，充分发挥多核的优势。要提高处理器的利用率，让处理器尽可能的满负荷工作，那么在编解码时就绪队列中就要有尽可能多的待处理的帧，以防止没有可处理帧，使得处理器进入空闲状态。

基于以上思路，我们设计了一种 MVC 的启发式动态调度算法，该调度算法根据 MVC 的 DAG 中的相关信息来进行调度，并在每次处理完一帧后对 DAG 图进行更新，删除已处理的帧对应的节点，以及相应的边。那么为了能够让更多的帧加入到就绪队列中，在调度时选择当前 DAG 图中其后续节点集中可加入到就绪队列的帧最多的。

定义 $I(u)$ 为当前 DAG 图中节点 $u \in V$ 的入度，即表示帧 $u$ 所依赖的帧数；$S(v)$ 为节点 $v \in V$ 的后继节点集合，它表示依赖于帧 $v$ 的帧的集合；$N(v)$ 为集合 $S(v)$ 中入度为 1 的节点数，即在节点 $v$ 执行完后，这些节点的入度变为 0，可以加入到就绪队列中。

为了能够充分发挥多核的优势，每次在调度时选择就绪队列中 $N(v)$ 最大的节点，因为该帧被处理完后将有最多的帧可加入到就绪队列。具体调度算法如下：

1、 初始化，保存初始 DAG 图的信息，对 DAG 图中所有帧求其初始入度 $I(u)$，将入度为 0 的放入就绪队列中。对就绪队列中的帧计算其 $N(v)$。

2、 当有 CPU 空闲时，从就绪队列中选择 $N(v)$ 最大进行调度。

3、 每处理完一帧，更新 DAG 图，删除其对应节点和边，更新相应帧的入度值 $I(u)$。将入度为 0 的帧放入就绪队列，并计算其 $N(v)$。

通过这种调度算法尽量保证就绪队列非空，从而减少处理器空闲的概率，以提高对 MVC 的处理速度。但是由于其为动态调度算法，每处理完一帧，都需要进行相应的计算，附带了额外的开销。

## 3.2　参考软件平台介绍

由于多视点视频编解码算法比较复杂，独立完成该算法需要很大的工作量，而我们的目标主要是对 MVC 进行并行优化。目前负责起草 MVC 标准的联合专家小组 JVT 在发布 MVC 草案的同时，也提供了 MVC 的参考软件 JMVM（Joint Multi-view Video Model）。所以为了简化工作，本文在 JMVM 的基础上来设计实现 MVC 的并行优化。

### 3.2.1　参考软件的整体框架

JMVM 的设计与实现基于可伸缩编码 SVC（Scalable Video Coding）的参考软件 JSVM，它是 H.264 参考软件的扩展。JVT 在 MVC 标准颁布之前发布了 7 个版本的 JMVM，在标准颁布之后，改名为 JMVC（Joint Multi-view Video Coding），目前已经发布了 3 个版本，并在不断更新中。

作为 MVC 的参考软件[6]，其实现了 MVC 中所需的所有算法。由于其基于 JSVM，整个软件包含很多个工程，但与 MVC 相关的主要包括以下几个工程：

1、 H264AVCCommonLib：公用模块库
2、 H264AVCDecoderLib：解码相关函数库
3、 H264AVCEncoderLib：编码相关函数库
4、 H264AVCVideoIoLib：数据输入输出和管理的函数库
5、 H264AVCDecoderLibTest：MVC 编码算法测试工程
6、 H264AVCEncoderLibTest：MVC 解码算法测试工程
7、 MVCBitStreamAssembler：视频流整合工具
8、 BitStreamExtractor：视频流提取工具

前四个为 MVC 的相关函数库，其中定义了 JMVM 中的各种类型的类，实现编解码的相关算法和进行文件访问和管理的函数。而 H264AVCEncoderLibTest 和 H264AVCDecoderLibTest 分别负责实现 MVC 的编码器和解码器。由于多视点视频包含多个视频信息，所以需要通过 MVCBitStreamAssembler 将多个视频信息整合到一个流中，便于进行传输和存储；而 BitStreamExtractor 负责将多个视频信息提取出来。各个工程之间的相互关系如下图 3-2 所示：

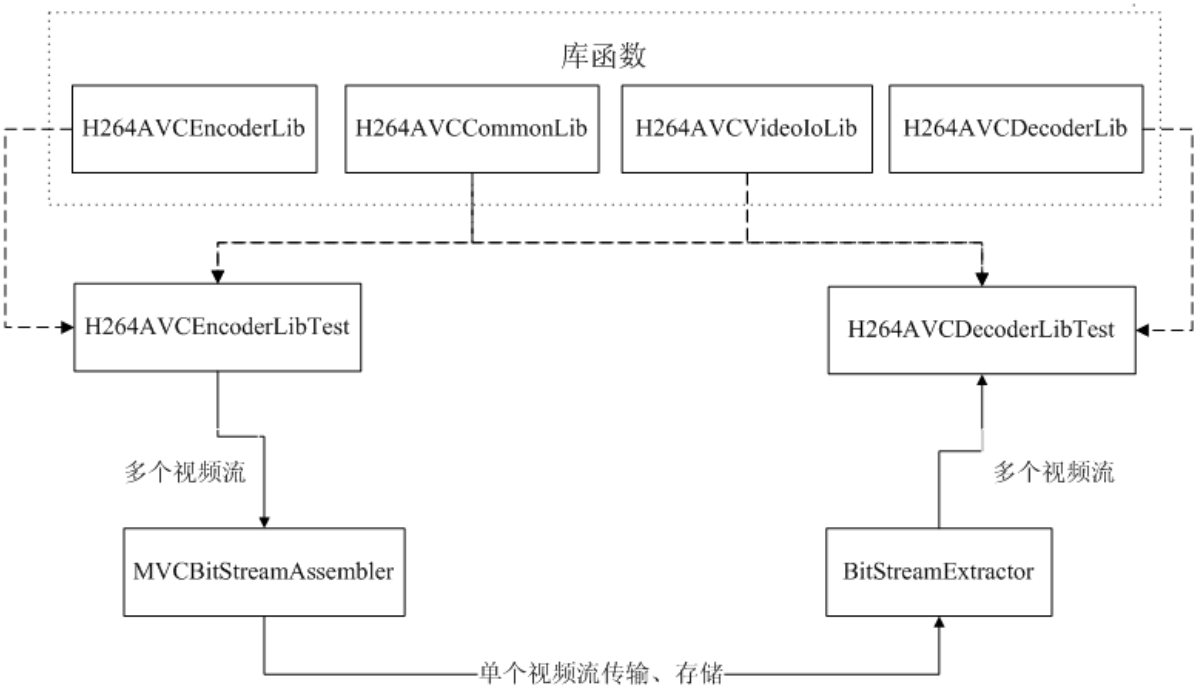



图 3-2 JMVM 的组织框架

JMVM 采用配置文件作为输入的方式，为用户提供了更大的灵活性。用户可以在配置文件对相关参数进行设置，包括视频序列信息，编码预测结构，QP 值，变换编码算法等参数。编码端 H264AVCEncoderLibTest 根据配置文件对用户指定的视点的视频进行编码。用户按照配置文件中指定的视点编码顺序，先后利用 H264AVCEncoderLibTest 对各个视点进行编码，这样就得到多个视点的视频序列。然后通过比特流整合工具 MVCBitStreamAssembler 将所有视点的视频序列整合到一个比特流中，这个比特流可以保存到本地文件中，也可以通过网络进行传输。在用户得到该比特流时，首先利用比特流提取工具 BitStreamExtractor 对整个流进行分离，从而得到各个视点的视频序列，然后再将这些视频信息提供给解码器 H264AVCDecoderLibTest，进行解码。

JMVM 采用传统的编解码处理思想，将整个视频分为多个视频序列(Sequence)，每个视频序列包含多个 GOP 组，每个 GOP 组中包含多个图像(Picture，对于帧编码的场也称为 Frame)。而对于每一帧又可包含一个或者多个片(Slice)，每个片包含整数个宏块 MB(Macroblock)，编解码算法便以宏块为最小的处理单元。MVC 采用预测编码和变换编码相结合的混合编码方式，对于一个宏块进行编码处理的过程如下图 3-3 所示[58][59]：

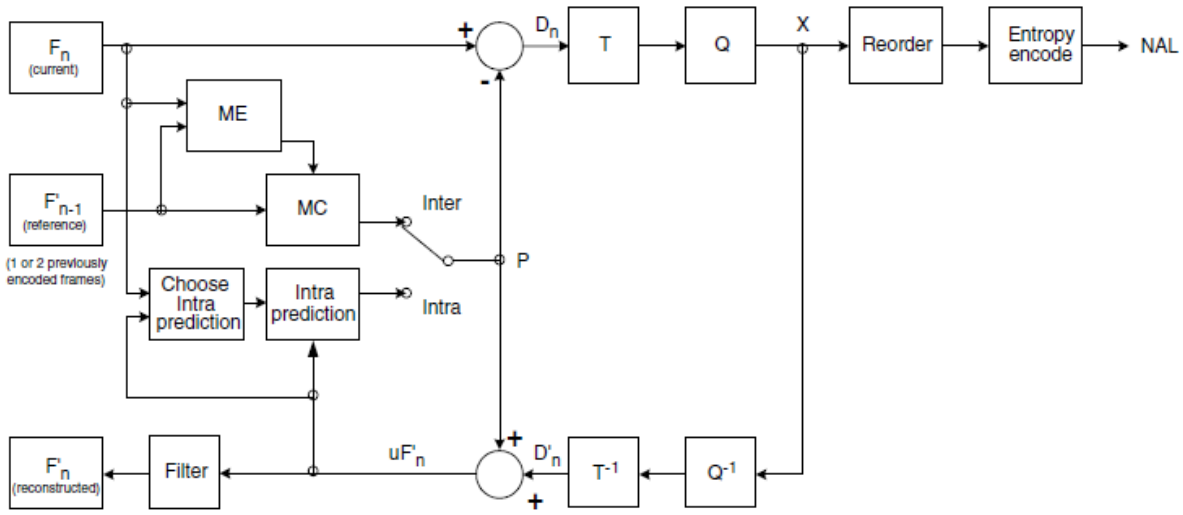



图 3-3 MVC 编码的过程

对应的解码器如下图 3-4 所示：

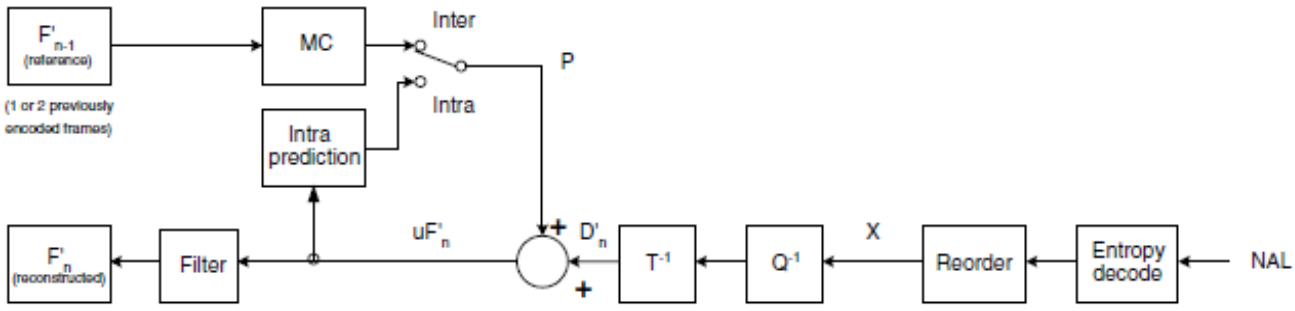



图 3-4 MVC 解码的过程

### 3.2.2　多视点视频编解码的标准码流

多视点视频编解码 MVC 为了具有良好的网络亲和性，可适用于各种传输网络，其保留了 H.264 的分层设计方法。所谓分层设计即对视频流进行多层封装。H.264 包括视频编码层 VCL(Video Coding Layer)和网络提取层 NAL(Network Abstraction Layer)两层[58]。其中视频编码层 VCL 中的数据为编码处理后的输出(即上图 4-2 所示的输出)，它表示被压缩编码后的视频数据序列，这些数据在传输或存储之前先要被封装到相应的 NAL 单元中。MVC 分层包装的结构如下图 3-5 所示：

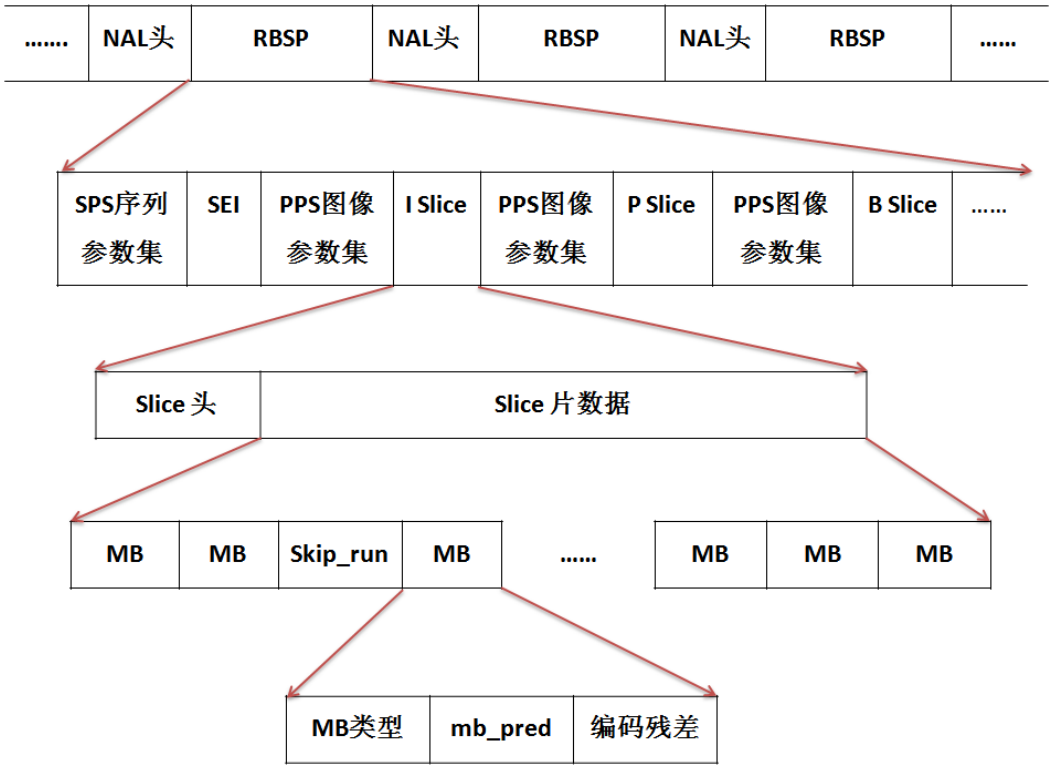



图 3-5 MVC 数据包的封装结构

第一行为 NAL 单元的数据结构，第二行为原始字节序列负荷 RBSP(Raw Byte Sequence Payload)单元，它用来封装一个序列的视频数据，包括一个序列参数集合 SPS(Sequence Parameter Set)，以及该序列中所有图像的信息。第三行为片 Slice 单元，第五行为宏块单元的数据格式，包括了该宏块的类型，预测模式 mb_pred 和编码残差三部分。MVC 的标准比特流便是通过上述分层包装的形式得到的。

JMVM 虽然实现了 MVC 的所有功能，但是由于它基于 JSVM，主要用来作为编码性能的分析参考，程序没有进行代码优化，存在着大量的冗余算法和过多的函数嵌套，使得计算复杂度很高，处理速度很慢。本文用分辨率为 320×240 的“golf1”视频序列进行测试，对其中一个视点的 81 帧进行编码，大约需要花费 30 分钟的时间，处理速度过慢，将会影响到 MVC 未来的实际应用。而且用户每运行一次只能处理一个视点的视频，并且需要按照视点编码顺序依次运行该程序，这样缺少对多个视点视频的管理和调度，程序框架不利于并行，不利于进行并行调度。

## 3.3　实验平台和设计框架

本文的实验基于参考软件 JMVM 6.0 版本，采用其中的运动估计、运动补偿、模式选择、变换编码、量化、熵编码等 MVC 所需要的算法，并对其进行代码优化，重新组织程序结构框架，优化内存的组织和管理，从而为实现多视点视频编码和解码的并行加速提供便利。

实验平台采用 intel Xeon 4 核处理器，每个核 1.6G 的处理速度，8G 的内存。同时还利用 Intel 向我们提供的一个 32 位的 16 核处理机，来进行多核加速比实验的测试。具体实现在 visual studio 2005 环境下进行。

本文将整个系统分为三个模块，一个为公用模块，另外两个为编码模块和解码模块。公用模块里面主要包括各种数据结构和数据类型的声明和定义、内存的管理，其为编码模块和解码模块所公用。而编码模块和解码模块分别包含了编解码算法所需的内容。

## 3.4　编码端的并行设计与实现

MVC 的编码功能在编码模块中实现。对于编码端我们采用第 3.1 中所提出的基于 MVC 的启发式静态调度算法，来对其进行并行加速。

考虑到 MVC 未来的应用包括实时系统和分时系统两种，对于分时系统来说，编码端可以对任意多数量的帧进行处理，所以本文引入并发 GOP 的概念，来进一步对 MVC 进行加速。所谓并发 GOP 指的是编码器可以同时处理的 GOP 单元。由于分时系统中所有的视频信息都已存在，没有时间的限制，无需等待，所以可以对多个 GOP 单元同时进行调度，增加了每次可调度的帧数，使得 CPU 尽量满负载工作，从而提高处理器的利用率和处理速度。

### 3.4.1　编码并行优化的框架和流程图

编码器主要包括如下四个模块和一个配置文件：

1、 码流标准化模块：用来负责对多视点视频编解码的码流进行标准化，以及文件的输入输出流、内存的管理

2、 编码模块：用来实现编码中涉及到的预测编码、变换编码和熵编码等具体算法

3、 任务并行调度模块：负责编码时处理器的控制和任务的并行调度。

4、 线程通讯模块：根据帧类型 I/P/B 帧，实现了相应的编码线程，。

5、 配置文件：Configure.h 和 Configure.cpp，用户可以在 Configure.cpp 文件中更改相关变量的参数，包括测试视频的信息，输出视频质量的需求，可供系统使用的处理器核数等。

编码器的主流程图如下图 3-6 所示：

首先用户在 Configure.cpp 文件中设置一些参数，系统通过这些参数来对编码器的相关配置进行初始化；然后系统预先处理该视频序列中的几个 GOP，从而计算得到处理该视频中不同类型的帧所需要的时间比值的期望值，将这些期望值作为其处理时间，从而计算出在 DAG 图中各个节点（即 GOP 中各个帧）的最长剩余路径，用该路径值作为其调度的优先级。

启动调度线程，对处理器各个核的状态进行初始化，利用启发式静态调度算法，根据预处理时得到的优先级，获取下一个可执行的帧的信息；检查各个核的状态，等待空闲的核；根据该帧的类型，启动相应的编码子线程，并将其分配到该空闲的核上。所谓分配到该核上是一种虚拟的分配，只是模拟处理器状态，将某一帧绑定到一个核上，而实际上该帧的处理是由处理器通过自身的调度优化机制来完成核的分配的。

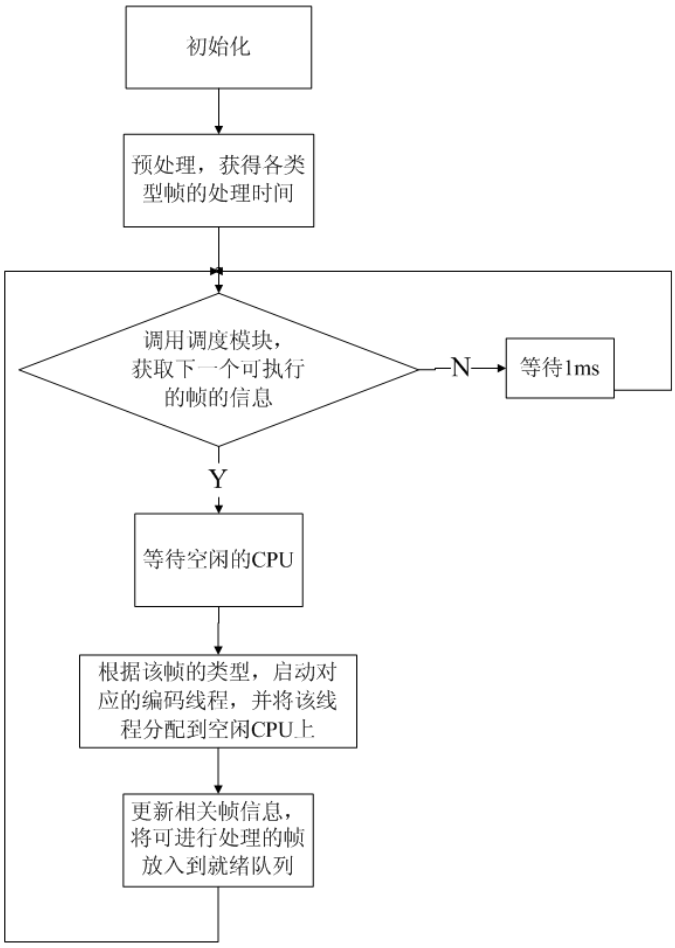



图 3-6 编码端的主处理流程图

### 3.4.2 码流标准化

Buffer 模块负责输入输出流的管理，以及输出流格式的标准化。其中输出流格式标准化是最重要的工作。

为了具有通用性，与国际标准相符，系统根据联合视频小组 JVT 制定的 MVC 标准，即 3.1.1 节中所阐述的 MVC 的流格式，定义了一个 OutStream 类，专门负责对多视点视频编码的输出码流进行标准化。其中包含 NAL 头信息类，序列参数集类 SequenceParametersSet、图像参数集类 PictureParametersSet、片头信息类 SliceParameters 的定义。同时通过以下几个主要函数，分别将编码后视频数据、UVLC 编码信息、序列参数集、图像参数集、以及片头信息等写入到 MVC 的码流中，从而实现对 MVC 的输出流的标准化。

1、 void WriteCode(uint32_t code, uint32_t len)

2、 void WriteUvlc(uint32_t uvlc)

3、 int32_t WriteSequenceParameterSet(SequenceParametersSet &sps)

4、 int32_t WritePictureParameterSet(const PictureParametersSet &pps, int32_t seq_parameter_set_id)

5、 int WriteSliceHeader(const SliceParameters &sp, SlidingWindow &slidingWindow);

### 3.4.3　编码算法的实现

编码模块用来实现编码中涉及到的预测编码、变换编码、熵编码等具体算法。该模块主要基于 JMVM 中的编码函数来实现。由于 JMVM 是在 JSVM 的基础上改进的，所以其存在大量与 MVC 不相关的信息，在编码模块中，我们一方面删除这些冗余代码，另一方面也对其中的函数进行改进和优化，使其满足系统的需要，具有一定的可并行性。编码算法的实现主要包括量化、运动估计，运动补偿、离散余弦变换编码，基于上下文的自适应可变长编码(CAVLC)熵编码等。这一部分不是我们工作的重点，具体实现不在此赘述，相应的可以参考 JMVM 的标准文档。

### 3.4.4　线程间的通讯

线程通讯模块根据不同的帧类型来创建相应的编码子线程，并完成线程间的通讯，其主要作为编码模块与其他模块之间交互的接口。

在该模块中定义了三个线程函数 encodeframe_i、encodeframe_p 和 encodeframe_b，分别用来创建 I 帧、P 帧、B 帧的处理线程，负责子线程的内存分配和管理，以及线程之间的通讯。

由于系统基于多核处理器，相应的线程比较多，所以线程通讯的管理是该模块的工作难点，也是重点。这部分工作主要通过一个 SPUSTATUS 结构的变量和函数 TransferParam 来完成。

在系统设计时，为了降低调度的复杂性，我们在多核处理器进行并行调度时，一个核单独负责处理一个线程，这样线程之间的通讯主要就是主线程与子线程的通讯，即主线程所在处理器与其他处理器之间的通讯。系统为每个核定义了一个 SPUSTATUS 结构的变量，用来描述该核的相关信息，包括其工作状态（忙、空闲、等待）、正在处理的线程的控件，以及该线程的相关图片信息等。

子线程和主线程都可以通过调用函数 TransferParam 访问该变量，进行相关信息的交互。

该模块中同时实现了函数 DoProcessI，DoProcessP 和 DoProcessB，用来负责完成 I 帧、P 帧和 B 帧的处理流程，其具体处理流程图见图 3-3。

### 3.4.5　任务的并行调度

任务并行调度模块是我们工作的重点，它负责编码时对任务的并行调度以及处理器的控制。

对于任务的并行调度，编码器采用 3.1 节所提出的启发式态调度算法，其主要包括两部分内容：MVC 的有向无环图 DAG 的表示和处理，以及调度算法的实现。

系统通过以下几个整数类型的数组变量来表示 DAG 图：

static int _tDepList [][2]：定义了一个 GOP 中时间上帧与帧之间的参考关系。

static int _vDepList [][2]：定义了一个 GOP 中视点间的参考关系。

static int _anchor [GOPSize]：定义了关键帧(I 帧)在 GOP 中的位置。

static int _deps[GOPSize][streamcount][10][2]：记录 GOP 中每个帧所依赖帧的位置信息，通过上述三个变量计算得到。

static const double runningTime[GOPSize][ streamcount]：记录了 GOP 中每个帧的处理时间的期望值。

static const double leftTime[GOPSize][streamcount]：记录 GOP 中每个帧的剩余路径长度。

static int deps_num[GOPSize][streamcount]：记录该帧所依赖的帧的数目

通过对以上数组的处理来对 MVC 的有向无环图进行修改。由于该有向无环图对于 MVC 来说表示的是其预测结构的信息，我们可以通过信息来对 MVC 的预测结构进行修改，为后续的实验提供了方便。

系统定义了一个就绪队列 CanCodings，用来记录可执行的帧的位置信息，包括其在预测结构中的时间位置和视点位置。在处理完一帧后对与该帧相关联的帧进行依赖信息 deps_num 的更新，如果 deps_num 为 0，则表明该帧可进行处理，将其放入到就绪队列中。

启发式静态调度算法在函数 GetNextProcessing 中实现。调度函数被调用

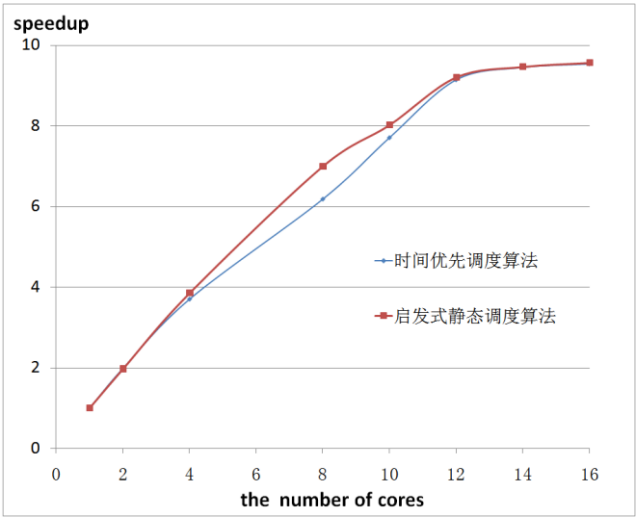
时，通过数组 leftTime 来查找就绪队列中各个帧的最长剩余路径，获得最长剩余路径最长的帧，并将该帧的相关信息返回。这些相关信息包括帧类型、分辨率、时间和视点的位置坐标、量化步长 QP、所参考的帧的位置信息等，这些信息通过 CodingPictInfo 类进行封装。

考虑到分时编码的情况，我们引入了并发 GOP 的概念，在系统中通过变量 GOPLimit 来表示并发 GOP 的数量。当 GOPLimit 不为 1 时，调度算法需要同时对多个 GOP 进行调度，而不仅 leftTime 数组。

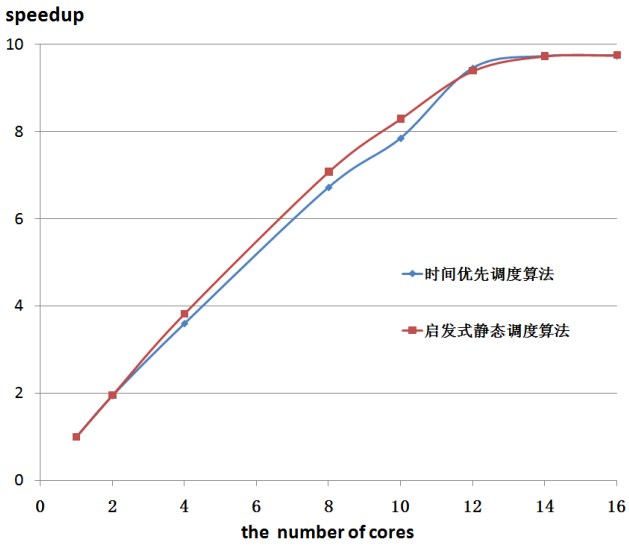
对于多个核的控制和调度通过函数 DispatchTask 来实现。在主线程调用该函数时，其对系统中存在的所有核(用户通过常量 NUM_OF_SPU 进行定义)的状态进行搜索，如果当前核处于空闲状态，则根据帧类型在该核上启动相应的编码子线程，并通过函数 TransferParam 将编码相关的参数传递给该子线程。

### 3.4.6　实验结果与分析

对于并行调度算法以并行加速比作为其主要的衡量指标。本文在 intel 公司提供的 16 核处理器对上述系统进行了加速比的测试。为了了解本文提出的启发式调度算法的性能，实验中同时实现了第二章中所述的传统的基于时间优先调度算法。实验的测试序列为 KDDI 提供的测试序列 golf1[13]和名古屋大学提供的测试序列 rena1[11]。这两个测试序列的帧率都为 30fps，分辨率分别为 320×240 和 640×480。当并发 GOP 数量 GOPLimit 为 1 时，即可同时处理的 GOP 只有一个时，其测试结果如下图 3-7 所示：




(a) golf1




(b) rena1

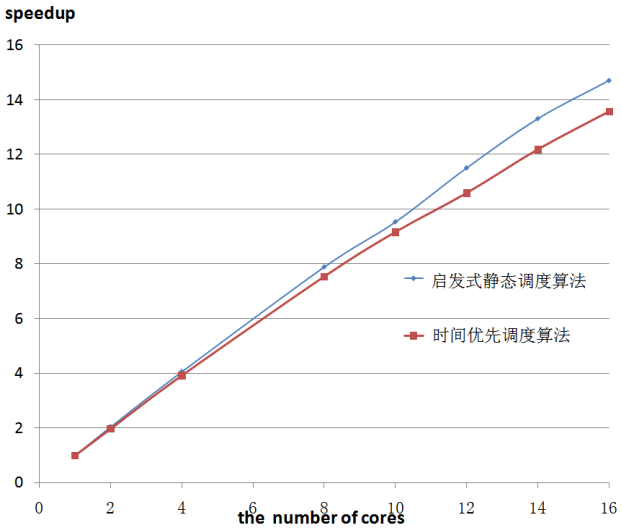
图 3-7 编码器的实验测试结果

从实验结果可以看出，对于并发 GOP 数量为 1 时，随着核数的增加，本文所提的启发式静态调度算法比基于时间优先的调度算法具有更高的加速比，说明它可以对 MVC 的处理单元 GOP 进行更充分的调度，利用多核优势，提高了处理器的使用率。

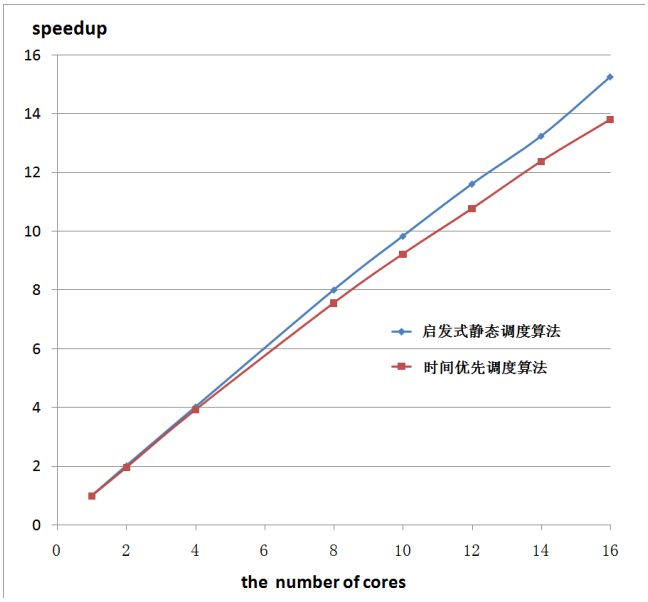
在核数大于 12 后，无论是哪种调度算法，其加速比都达到最大值 9，而且随着核数的增加不再提高。这是由 MVC 的算法本身决定的。由于 MVC 在进行编解码时，数据之间存在着相互依赖关系，所以当核数达到一定数量后，即使核数再增加，但是由于没有可以被处理的帧，只能使得这些核空闲。它表明了当前 MVC 算法在帧一级的可并行的性能。

如果想进一步提高其加速比，一方面可以选择更小粒度的并行方法，以增加可调度任务量，比如说采用宏块级的并行；另一方面也可以利用上述的增加并发 GOP 数量的方法，来提高可并行调度的帧数。但是由于实际应用中用户可获得的 GOP 数量有限，特别是对于实时系统来说，可并行处理的 GOP 数量一般会比较少。

我们对并发 GOP 数量 GOPLimit 为 2 进行了并行加速比性能的测试，其测试结果如下图 3-8 所示：




(a) golf1




(b) rena1

图 3-8　并发 GOP 数为 2 时的编码测试结果

当并发 GOP 数量为 2 时，无论是哪个调度算法，其并行加速的性能都得到了很大的提高，基本上随着核数的增加而呈线性增长的趋势。

下面分析第二章中所提出的分析模型的有效性。当并发 GOP 数量为 1 时，其测试结果如图 3-9 所示，当并发 GOP 数量为 2 时，本文的模型分析出来的结果如下图 3-10 所示。

从实验结果可以看出，实际测试的实验结果和我们的模型分析出来的结果基本上是一致的。这验证了第二章中的分析模型的有效性和正确性。

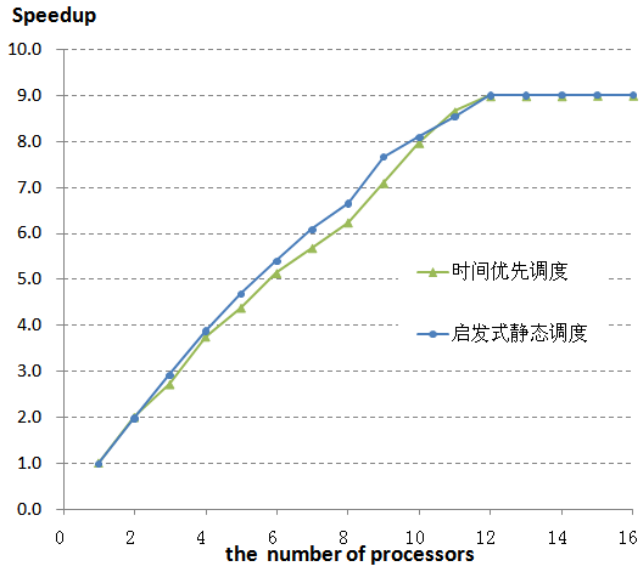



图 3-9 并发 GOP 数量为 1 时模型的测试结果

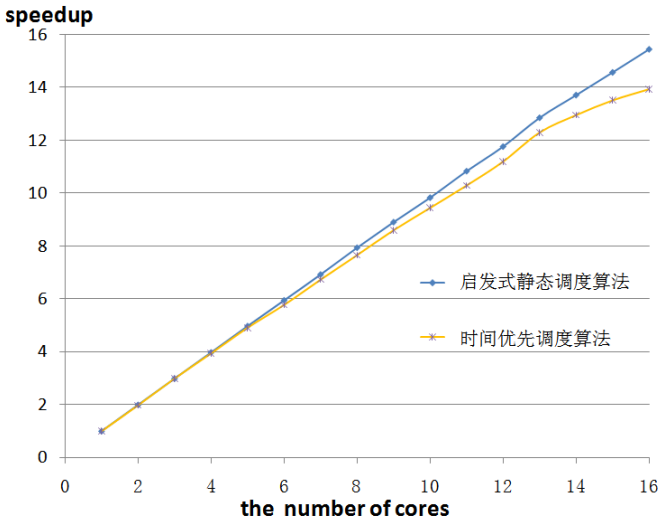



图 3-10 并发 GOP 数量为 2 时模型的测试结果

## 3.5　解码器的并行设计与实现

由于解码器和编码器的原理近似，所以解码器的整体设计框架和主处理流程，与编码器基本一样。

解码器包括了类似于 3.4 节中所述的四个主要模块：码流解析模块、解码模块、线程通讯模块和任务并行调度模块。

码流解析即对编码后标准码流进行分析，获取相应的编码参数和视频信息，它是码流标准化的逆过程，其相关实现和码流标准化模块类似；解码模块同编码模块一样，基于 JMVM 来实现；线程通讯方法与编码器一样；由于解码器终端有着不同的应用需求，对于其调度算法也相应的与编码不同，这是解码器并行优化的工作重点。

### 3.5.1　解码器的并行调度

由于多视点视频未来的应用比较广泛，用户对于解码器的需求会随着应用的不同而不同。比如说对于自由视点视频 FVV 来说，其每次只需要提供给用户当前视点的视频信息即可，所以解码端应该优先将当前视点的视频进行处理；而对于 3D 视频来说，其需要所有视点的视频都处理完后才能对当前时刻进行 3D 合成，所以解码端应该优先处理当前时刻的所有视点的视频。这样对于并行调度算法来说一方面要提高解码器速度，另一方面也需要具有一定的灵活性，方便根据不同的应用需求来优化调度策略，所以在解码端我们采用 3.1 节中提出的具有一定灵活性的基于 DAG 出度的启发式动态调度算法。

系统为每一帧定义了一个变量 effs_num，用来记录该帧在 DAG 图中的出度数，即依赖该帧处理的帧的数目。

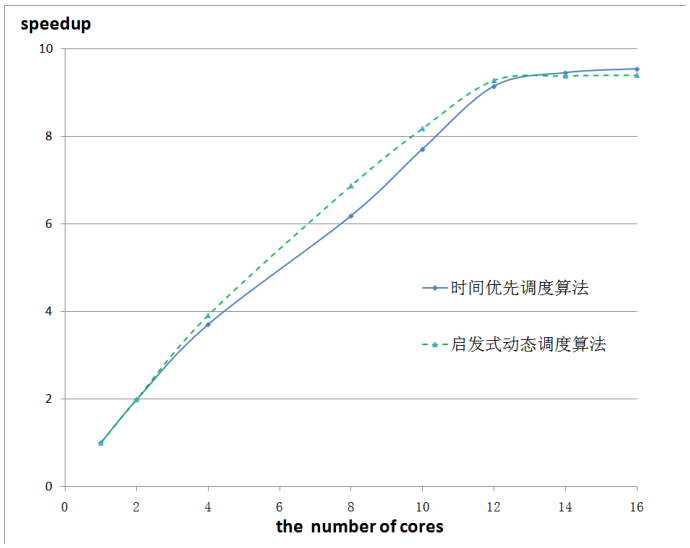
我们的实现以 3D 视频的应用为例。对于 3D 视频来说，其在解码端最后需要进行多个视点的合成，然后提供给用户。考虑到视点合成需要一定的时间，如果等到整个 GOP 解码完后再合成，那么势必会造成很大的延迟，所以我们在解码端的调度算法中加入了时间优先的策略。当通过启发式动态调度算法得到的可调度的帧数大于 1 时，优先选择时间最小的帧来处理。该并行调度算法的伪代码见表 3-1 所示：

表 3-1　基于 DAG 出度的启发式动态调度算法的伪代码

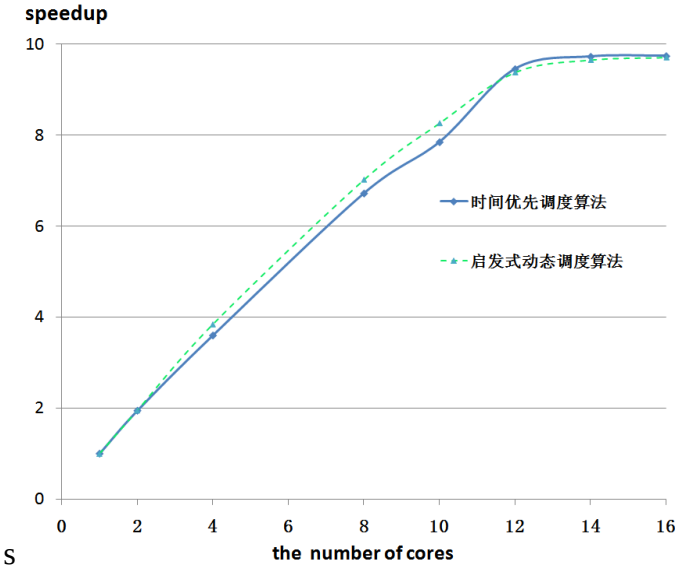
```
If 就绪队列 canCodings 为空
    返回 false;
else
begin
   maxEffsNum = 0;  //记录当前最大的出度值
   TimeID，ViewID = -1;
   for 就绪队列中每一帧 v
   begin
      If   effs_num(v) 大于 maxEffsNum   //选择出度最大的帧
      begin
           maxEffsNum ← effs_num(v);
           TimeID←TimeID(v);
           ViewID←ViewID(v);
      end
       If   effs_num(v)等于 maxEffsNum
          If   TimeID(v) 小于 TimeID   //选择时间最早的帧
          begin
              TimeID←TimeID(v);
              ViewID←ViewID(v);
          end
     end
     返回 ture;
end
```

### 3.5.2　实验结果与分析

采用和上一节同样的实验平台和环境对解码端在多核处理器上的加速比进行测试，得到的结果如下图 3-11 所示：




(a) golf1




S

(b) rena1

图 3-11　解码端的测试结果

从结果中可看出，我们所设计的解码端的启发式调度方法是比时间优先调度算法更优的。

# 第4章　具有高可并行性的 MVC 预测结构的设计

预测结构指的是视频编码中帧与帧之间的参考关系。预测结构的设计不仅会影响压缩性能、还将影响随机读取性能、快速解码性能、网络传输代价等，因而在 MVC 研究中受到广泛关注。对于多视点视频来说，考虑到其未来的应用，除了压缩效率外，还必须追求随机读取能力。但是数据压缩效率和随机读取能力之间存在着矛盾。要保证高的压缩效率，就必须增加数据之间的依赖关系来尽可能地减低信息冗余，而随机读取能力要求数据之间的依赖关系越少越好，从而可以实现快速访问。如何在这两者之间进行折衷是预测结构研究的主要课题。

本章将介绍一下多视点视频编解码 MVC 的预测结构的发展情况，从可并行性的角度出发，对当前的预测结构进行了分析，指出其在可并行性和随机访问上的不足，并根据多视点视频未来的不同应用，对于随机读取能力的不同需求，对其分别提出了改进方案。通过实验结果表明，虽然我们改进的预测结构的方法比较简单，但是很有效，保证了与原始预测结构具有近似视频质量和压缩比的同时，在可并行性和随机读取能力有更优的表现。

## 4.1　MVC 预测结构的发展

多视点视频相当于几个普通视频的组合，所以最简单的编码方式就是利用现有的编码标准对每个视点的视频分别单独进行编码，然后将编码后的数据流再整合在一起。而 H.264/AVC 是目前压缩性能最好的编码标准，所以采用此标准对每个视点来进行压缩。H.264/AVC 中使用 IBBP……结构来对每帧进行编码，其中 B 帧可以参考前后多帧，以便于能够更加准确的进行预测。而在文献[60]中提出了一种更加有效的 H.264 的预测结构—分级 B 帧的预测结构，该结构如下图 4-1 所示：

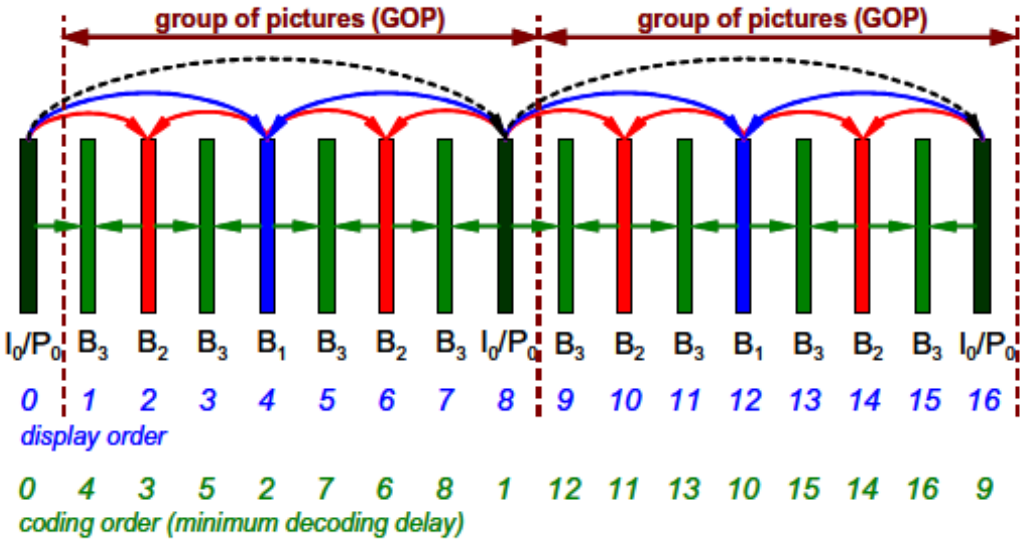



图 4-1 分级 B 帧的预测结构[60]

在该预测结构中，B 帧仍然采取传统的参考两帧的方法，但它对 B 帧在时间上进行了分级，下一级的 B 帧参考与其左右相邻的上一级的帧。这种预测结构一方面不使用多参考帧，减低算法的复杂度，另一方面通过实验结果也证明了该方法与多参考帧的方法相比，也能够很好的减少时间上的冗余。但是由于它进行了向后参考，这意味着有些帧的处理要落后于其后续的帧，这样编码顺序就不能按照时间先后的顺序来进行，在编解码端需要比较大是数据缓存区。MVC 在时间预测上采用了这种分级 B 帧的预测方法，我们以具有 8 个视点的多视点视频为例，若 GOP 大小为 8，则其预测结构如下图 4-2 所示：

其中 Si 表示第 i 个视点的视频序列，Ti 表示第 i 时刻。I 帧采用帧内预测的方法进行编码，被称为关键帧，其所在列(阴影部分)被称为关键帧列。其中 T0 列为第一个 GOP 的关键阵列，而 T8 列为第二个 GOP 的，但它会被第一个 GOP 的帧作为参考。

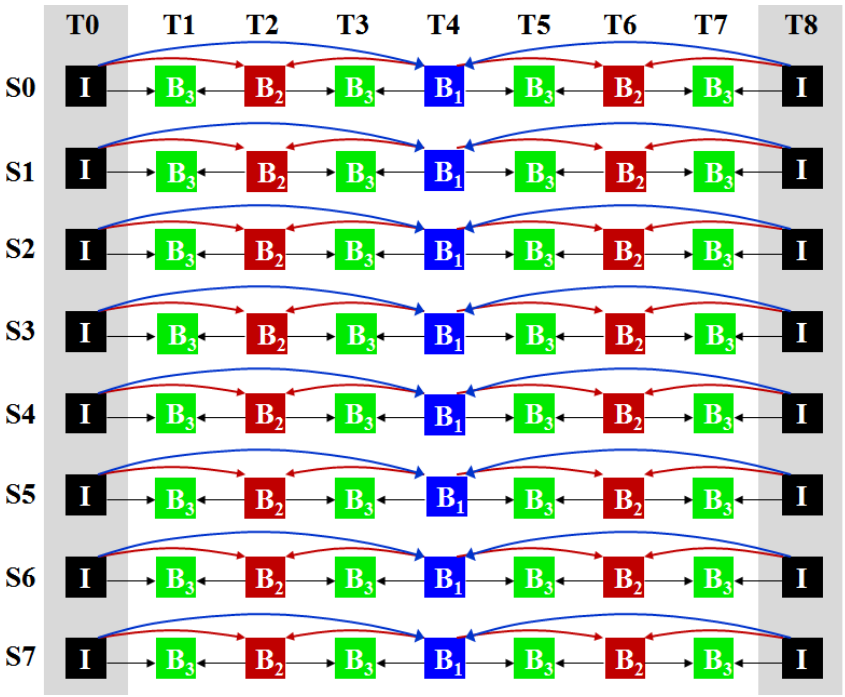



图 4-2 各视点独立编解码的预测结构

这样的预测结构比较简单，视点之间没有什么依赖，可以很快的进行视点间的切换，随机读取能力比较强，但是这种预测结构没有利用多视点视频在视点间的冗余信息，压缩效率比较低。随着视点数的增多，其数据量过于庞大将会影响其实际应用。

由于采用帧间预测(P 帧、B 帧)比帧内预测(I 帧)能够获得更多的压缩增益，而多视点视频各个视点采集的是同一个场景同一时刻的不同角度的信息，所以可以以一个视点为基础，其他视点的关键帧(I 帧)参考已有视点，采取帧间预测的方式进行压缩，以减少视点上的相关性，这就是所谓的视点间预测。从而我们可以得到如下图 4-3 所示的预测结构：

其中 S0 视点序列的预测编码方式不变，而其他视点序列的关键帧则依次参考其上一视点中对应时刻的关键帧。关键帧采用帧内预测编码（I0 帧）的视点序列（S0）成为关键视点。

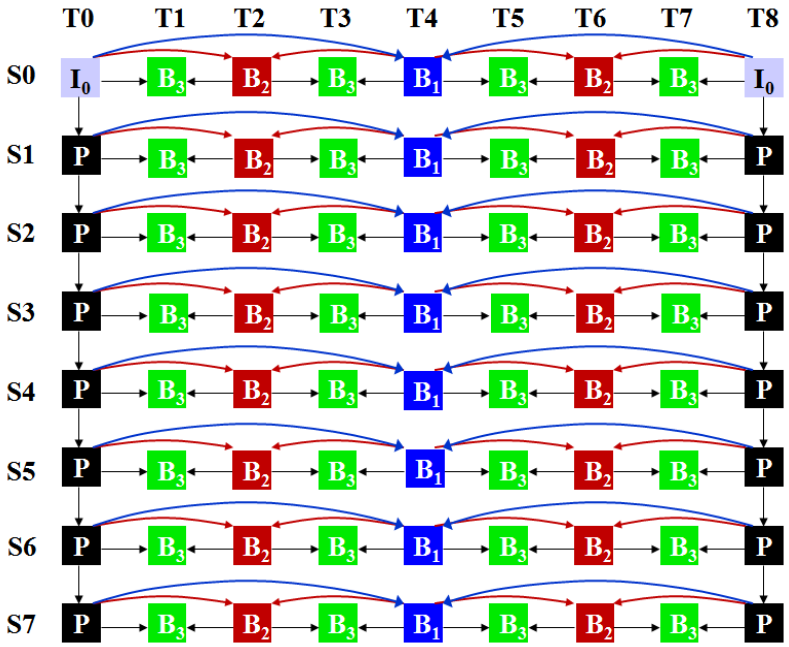



图 4-3 在关键帧列引入视点间预测的预测结构

由于预测编码预测值越准确，压缩效率越高。MVC 为了提高预测编码的准确性，在非关键帧上也引入了视点间预测，使得能够进一步提高其压缩效率。其预测结构如下图 4-4 所示：

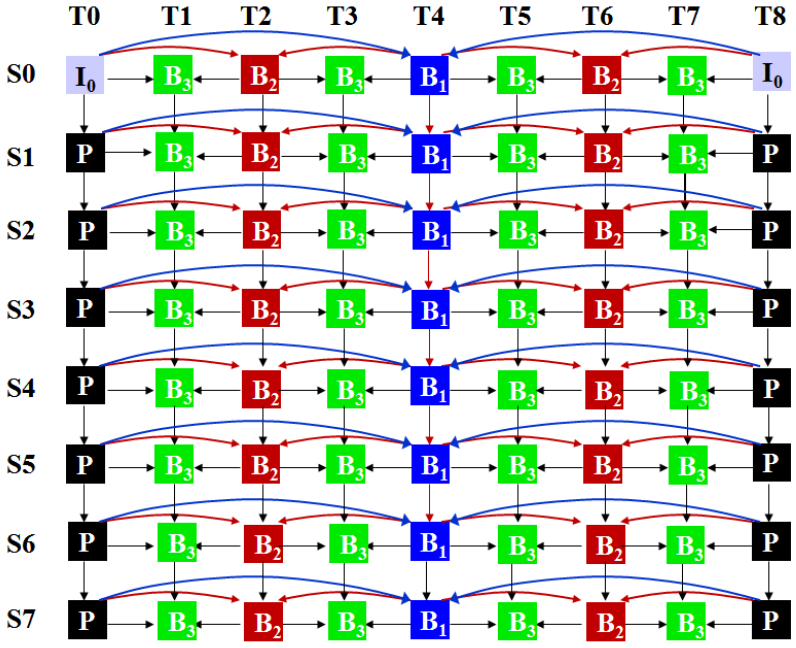



图 4-4 所有帧都采用时间和视点间预测的预测结构

图 4-4 中在关键帧列没有变化，而对于所有的非关键帧，其除了可以参考同一视点上的两帧外，还可以参考相邻视点同一时刻的帧。例如帧 S1T1，其参考帧为 S1T0、S1T2 和 S0T1 三帧。这种预测结构通过牺牲计算复杂度获得了更多的压缩增益，但是它极大的降低了解码时的随机读取能力。例如要访问 S7T7 帧时，其需要将关键阵列中的所有帧都解码，同时还要将 T4、T6 和 T7 时刻上的所有帧都进行编码，共计要处理 32 帧，访问效率太低。这主要是由于各个视点之间层层依赖，依赖关系不断累积造成的。

下图 4-5 所示的预测结构，很好的弥补了上面的不足，降低了依赖关系，在一定程度上提高了随机读取的能力。

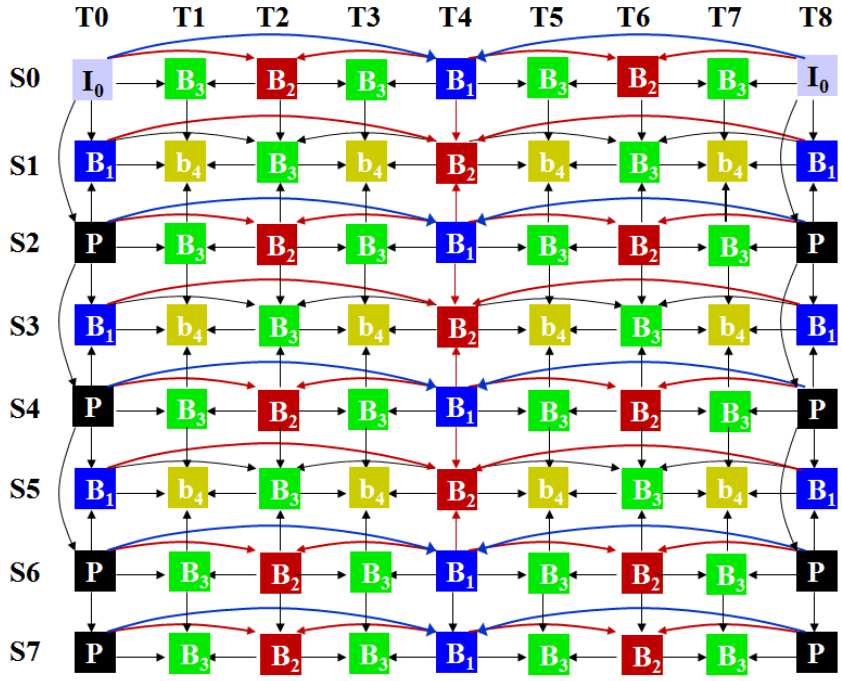



图 4-5 MVC 推荐的预测结构

其采用与时间预测类似的分级 B 帧的预测方式，这样打破了视点间层层依赖的关系，降低了依赖关系。还以访问 S7T7 为例，此时只需要对关键帧列中的所有 P 帧，以及 S6 和 S7 序列中的各三帧(S6T4、S6T6、S6T7、S7T4、S7T6、S7T7)进行解码，共计处理 11 帧，大大加速了随机访问。同时由于其在关键帧上引入了双向预测，在一定程度上还提高压缩效率。这种预测结构就是现在 MVC 标准中推荐使用的预测结构。

除了对于帧与帧之间的参考关系的研究外，预测结构还包括对 GOP 大小的

选择。如果 GOP 大小的值太大，那么在预测结构中时间上的 B 帧的分级数越多，分级数越多意味着要访问级数比较高的 B 帧，其需要解码的帧数越多，那么随机访问的性能就越差。如果 GOP 大小的值太小，那么压缩效率就会被降低。MVC 中推荐的 GOP 大小为 12 或 15，这主要是考虑到不同的电视制式。对于 PAL 制式的，其帧率为 25 帧每秒，而 NTSC 制式的帧率为 30 帧每秒。GOP 大小 12 或 15，表示一个 GOP 中的帧的时间为半秒钟。从人体视觉方面来讲，这样的视频质量不会太受影响，另一方面如果要进行随机访问、视点切换等操作，半秒钟的延迟相对更容易接受。

## 4.2　原始预测结构的分析

虽然图 4-5 所示的预测结构(在此我们称其为原始预测结构)在压缩效率上表现的很好，随机访问的能力有所提高，但是提高幅度有限。而且从可并行性的角度来讲，其数据之间的依赖关系依然比较多，不利于并行。而且多视点视频未来的应用范围比较广泛，主要可分为自由视点视频 FVV、3D 视频和沉浸式视频，而对于不同的应用，其在压缩性能、处理速度、随机访问、容错性等方面的需求是不一样的，所以在 MVC 标准中并没有严格定义其预测结构，而是把上图所示的预测结构作为一种推荐的结构，把帧与帧的参考关系写入码流中，从而提高其可扩展性。本节将对 MVC 推荐使用的预测结构进行了理论分析，然后根据多视点视频未来的不同应用场景提出相应的改进预测结构，以提高 MVC 本身的可并行性。

下面通过第二章所述的基于 MVC 的分层 DAG 图，从理论上来分析 MVC 的可并行性。根据 MVC 分层的 DAG，假设 CPU 资源无限，只要有可处理的帧，就可以分配到 CPU。那么一个 GOP 中所有帧的处理可以被分为几个步骤，每个步骤可执行的帧类型和相应的帧数如下表 4-1 所示：

表 4-1　原始预测结构中每步可执行的帧情况

| 帧类型 | Step1 | Step2 | Step3 | Step4 | Step5 | Step6 | Step7 | Step8 | 每种帧类型的总帧数 |
|---|---|---|---|---|---|---|---|---|---|
| I | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| P | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 4 |
| B0 | 0 | 1 | 2 | 2 | 2 | 1 | 0 | 0 | 8 |
| B1 | 0 | 0 | 2 | 4 | 4 | 4 | 4 | 2 | 20 |
| B2 | 0 | 0 | 0 | 3 | 5 | 9 | 8 | 6 | 31 |
| 可并行执行的帧数 | 1 | 2 | 5 | 10 | 12 | 14 | 12 | 8 | |

假设 CPU 资源无限，对于每帧的处理可以通过其他并行优化方式来做并行，使得每帧的处理时间都能够在同一时刻执行完。那么采用原始预测结构的 MVC，其一个 GOP 的处理需要执行 8 步。

从表中可以看出，可并行执行的帧数在每个步骤中是不均衡的，在处理 GOP 的第一步和第二步中分别只处理了 1 帧和 2 帧，而第 5 步和第 6 步则分别要处理 12 和 14 帧。假设在 10 核处理器环境下，每个核负责处理一帧，那么理论上 CPU 的使用情况如下图 4-6 所示：

| 步骤 | s1 | s2 | s3 | s4 | s5 | | s6 | | s7 | | s8 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 可执行帧数 | 1 | 2 | 5 | 10 | 12 | | 14 | | 12 | | 8 |
| CPU使用数 | 1 | 2 | 5 | 10 | 10 | 2 | 10 | 4 | 10 | 2 | 8 |

图 4-6 在 10 核处理器下执行 MVC 时处理器的使用情况

通过公式 4-1 来计算此时 CPU 的空闲率，可得其 CPU 空闲率为 41.82%，可见 CPU 的利用率不高，没能充分发挥多核的作用。

$$rate = \frac{\sum_{i=1}^{N} |n_i - M|}{M \times N} (M: \text{核数}；\ N = \text{总执行步数}) \qquad (4\text{-}1)$$

为此，需要对预测结构进行改进，来提高每个步骤中可并行处理的帧数，

来减少执行步骤，以便于对 MVC 进行进一步的并行加速。

## 4.3　基于 3D 视频的改进预测结构

3D 视频系统为用户提供了立体感场景的功能，所以在视频终端，其需要将多视点视频所有视点的数据解码后，通过 3D 合成技术最终将 3D 场景提供给用户。所以对于 MVC 在 3D 视频方向应用，一方面要研究其视点同步问题，另一方面编码并行加速也是一个很重要的研究课题。因为它需要得到同一时刻的多个视点的信息后才能够显示，即一个镜头的显示需要处理多个帧，而不像普通视频一样，只需要处理一帧即可。所以对于 3D 视频系统来说，MVC 能够尽快完成一个 GOP 的处理是其最主要的问题。

预测编码是以已编码的帧为参考帧，对其周边的帧进行预测编码。对于 MVC 的分层 DAG 图来说，越上层的帧数越多，那么后续可以处理的帧数也就会更多。所以为了提高 GOP 的可并行性，应尽可能提高 DAG 图的上层的帧数。MVC 的第一步处理的是 I 帧，如果增加它的数量，这将意味着压缩性能的大量损失，因为 I 帧的压缩比相对 P、B 帧来说小很多。所以我们考虑提高第二步可处理的帧数。为此我们提出了如下图 4-7 所示的预测结构：

其中保留了原始预测结构在时间上的预测关系和视点上的预测方法，只是将中间视点作为关键视点，即将 GOP 中 I 帧放在中间视点处，从而可以依据该帧同时向上视点预测和向下视点预测，这样使得 MVC 的第二步可以并行处理的帧数由 2 变为 3 帧（只包含本 GOP 的帧），而由于第二步帧数的增加，相应地第三步的帧数也随之增加，从而提高了每一步可并行执行的数量，有利于减少总的执行步数。

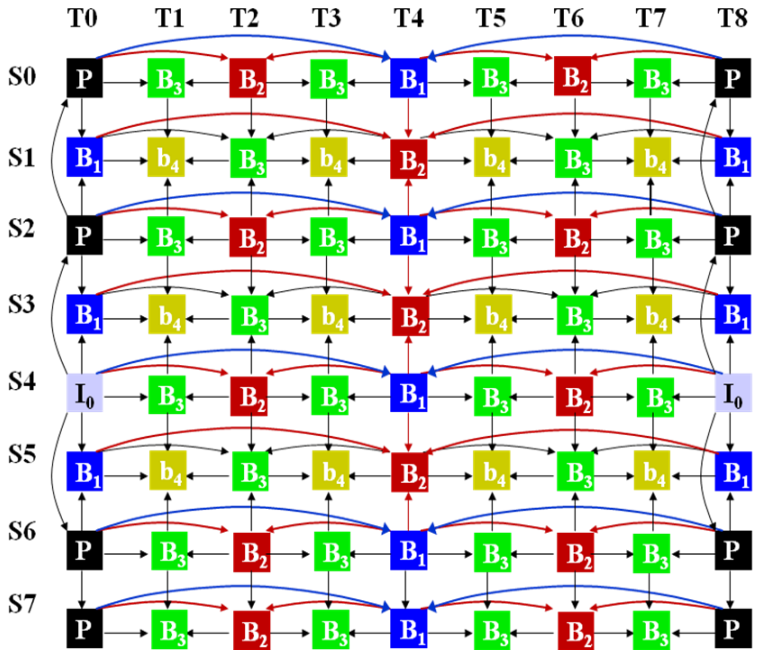



图 4-7 3D 视频的改进预测结构

改进后的预测结构的分层 DAG 图如下图 4-8 所示：

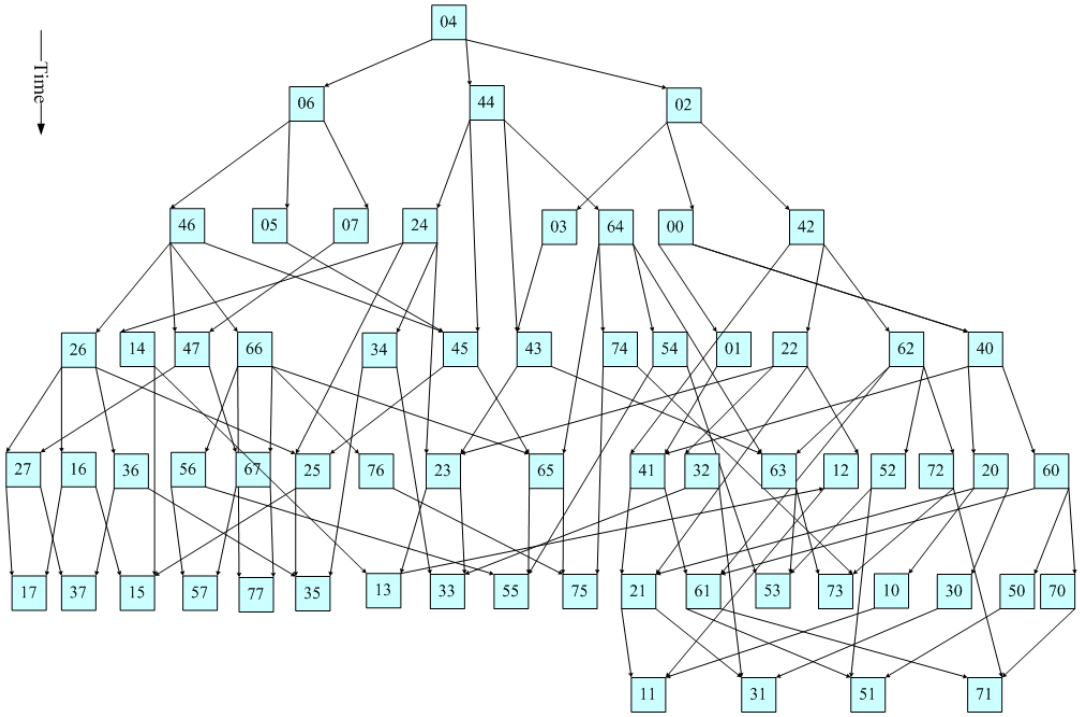



图 4-8 改进后预测结构的分层 DAG

### 4.3.1　改进后的性能分析

改进的预测结构其每步可执行的帧情况如下表 4-2 所示：

表 4-2 改进的预测结构中每步可执行的帧情况

| 帧类型 | Step1 | Step2 | Step3 | Step4 | Step5 | Step6 | Step7 | 每种帧类型的总帧数 |
|---|---|---|---|---|---|---|---|---|
| I | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| P | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 4 |
| B0 | 0 | 1 | 4 | 3 | 0 | 0 | 0 | 8 |
| B1 | 0 | 0 | 2 | 6 | 8 | 4 | 0 | 20 |
| B2 | 0 | 0 | 0 | 4 | 9 | 14 | 4 | 31 |
| 可并行执行的帧数 | 1 | 3 | 8 | 13 | 17 | 18 | 4 | |

从表中可以看出，改进后的预测结构在每一步可执行的帧数都比原来的预测结构有所提高，而且其总执行步骤也由原来的 8 变为 7。我们通过如下方法来计算其理论最短执行时间：

根据表 4-2，假设每一步骤能够求得该步的最长处理时间，那么所有步骤的最长时间加起来即为其处理时间的理论最小极限值。我们利用第二章模型中所使用的方法，通过初步计算，获得每种类型帧的处理时间的期望值，例如 $T_I = 1$，$T_P=4.8$，$T_{b0}=25$，$T_{b1}=35$，$T_{b2}= 50$。每一步所处理的帧中，执行时间最长的帧的处理时间即为该步的执行时间，这样得到的处理时间即为理论最小值。

通过上述方法我们计算原始预测结构的理论最短执行时间，其为 $T_I+T_{B0}+T_{B1}+5T_{B2}=311$，而改进后的为 $T_I+ T_{B0}+T_{B1}+4T_{B2} =260$，比原始预测结构节省了 16.1%的时间。

由于每种类型的帧的数量没有变化，其算法复杂度相对来说也就和原始预测结构的复杂度没有太大变化。另一方面，由于改进后预测结构有 2 个 P 帧直接通过 I 帧预测，这样从一定程度上也有可能提高其视频质量。

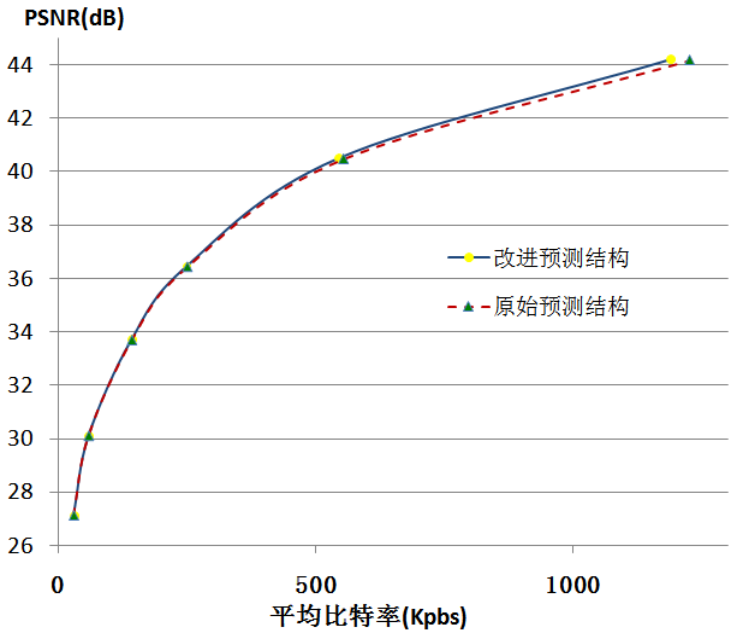
从上述理论分析可以看出，改进后的预测结构在保证视频质量和压缩比的前提下，一定程度上提高了多视点视频编解码的并行度，通过实验结果也证明了，采用同样的调度算法，改进后的预测结构比原来的预测结构的加速比更高。

### 4.3.2　实验结果

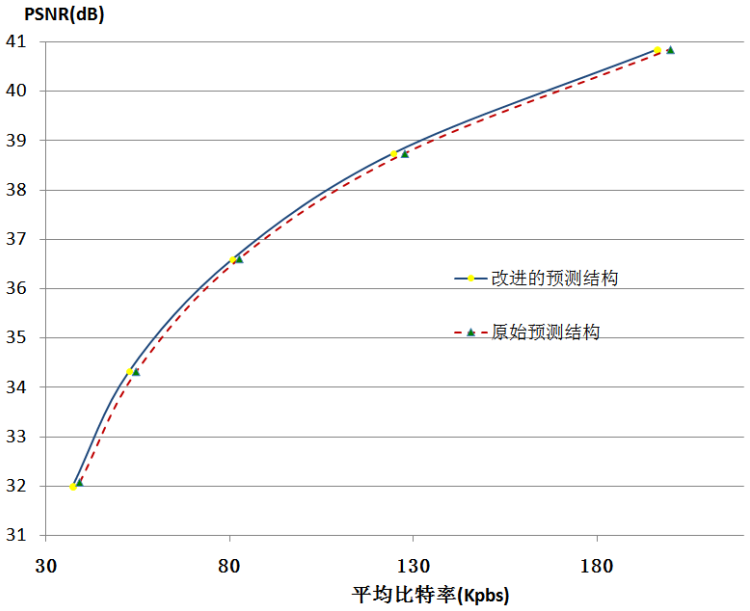
1、PSNR 测试结果

对于视频压缩来说，其前提条件就是保证视频质量。如果视频质量特别差，那么再高的压缩比也没有意义。而对于视频质量的评价一般包括两种方式，一是主观质量评测，由人从视觉上直接进行判定。另外一个是客观质量评测，通常用峰值信噪比 PSNR(率失真曲线)来衡量。

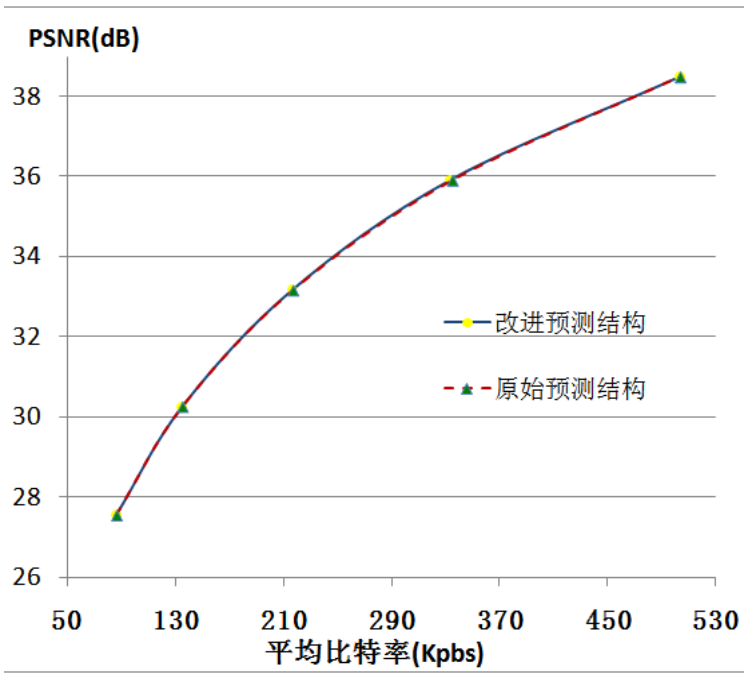
通过对 KDDI 提供的三个具有不同特性的视频进行 PSNR 的测试，这三个视频的名字分别为 golf1、flamenco1 和 objects1。其中 golf1 视频的光线比较亮，拍摄静态物体，但镜头缓慢平稳移动；flamenco1 视频的光线昏暗，有面积比较大物体进行一些慢运动，拍摄的镜头不动；而 object1 视频的背景灯不断闪动，导致光线变化忽明忽暗变化，镜头和物体都移动。实验的结果如下图 4-9 所示：




( a ) golf1




( b ) flamenco1




( c ) objects1

图 4-9　PSNR 测试结果

图中蓝色实线为改进的预测结构的率失真曲线，而红色虚线为原始预测结构的结果。从实验结果可以看出来，改进后的预测结构在视频质量上与原始结

构基本近似，甚至在一些视频上，其质量比原始预测结构表现更优些。这主要由于改进后的预测结构有两个 P 帧以 I 帧参考，提高了视频质量。

2、加速比测试结果

首先用参考软件对两种预测结构进行处理时间上的测试。采用相同的测试环境，测试的视频对象都为“golf1”。利用原始预测结构处理 150 帧所需时间为 14601 秒，而采用改进后的预测结构需要 14733 秒，处理时间基本上差不多。这证明了改进的预测结构没有增加其计算复杂度。

在确定计算复杂度没有提高之后，我们对其加速比进行了测试，来比较两个预测结构的可并行性。测试数据依然采用 golf1 视频，利用第三章的多视点视频编解码的并行编码系统来进行测试，测试结果如下图 4-10：

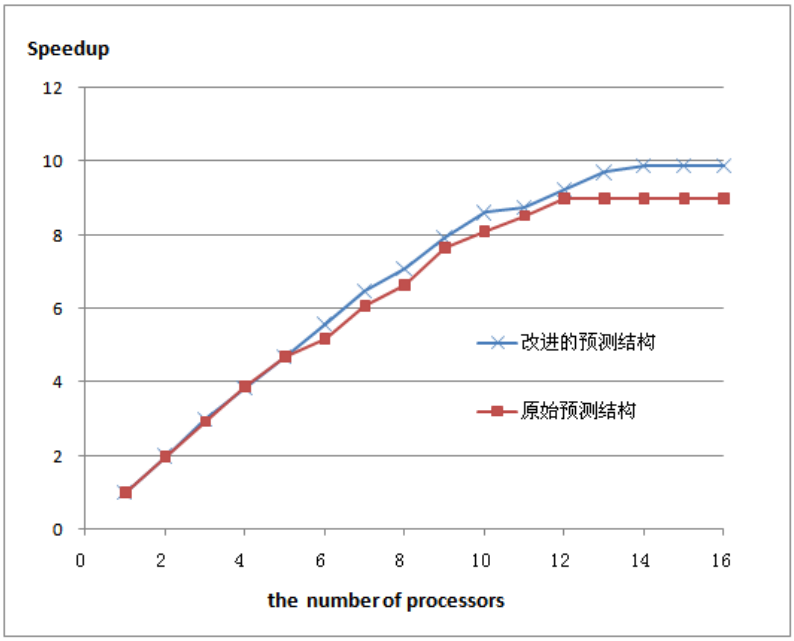



图 4-10　两种预测结构的加速比测试结果

实验结果表明改进后的预测结构在核数比较少时，与原始预测结构的加速比基本一致，但是当核数比较多时，改进的预测结构的加速比性能要比原始预测结构的性能表现更优。

## 4.4　基于自由视点视频的改进预测结构

自由视点视频 FVV 向用户提供了与场景交互的能力，用户可以随时选择场

景中自己想看的视点。由于其每次提供给用户的是单个视点的视频信息，所以解码端对于处理速度的要求不像 3D 视频那样高。自由视点视频的主要问题集中在视点切换上。由于用户可以随时进行视点切换，所以自由视点视频要求比较好的随机读取能力，降低视频切换的延迟，满足实时性的需求。

但是随机读取能力和数据压缩效率是相矛盾的，要实现快速的访问任意一帧，就需要帧与帧之间的依赖性尽量小，这恰好与压缩的需求相矛盾，在研究中需要对两者进行折衷。

为了了解视频序列中各帧之间的依赖关系，文章[62]中对一些多视点视频序列进行了测试，通过实验来分析多视点视频各帧之间时间预测和视点间预测的统计特性。实验中采用 H.264/AVC 编码方式，在对 P 帧进行预测编码时，其运动估计的参考帧有如下图 4-11 所示四种：

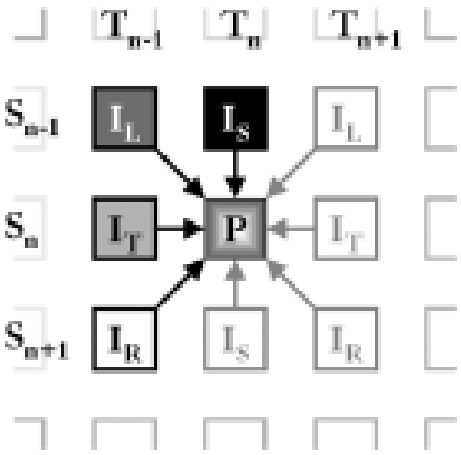



图 4-11 相关性测试实验的参考关系

其中 Tn 表示时间轴，Sn 表示视点序列。对于 P 帧的编码，有 $I_S$、$I_T$、$I_L$、$I_R$ 四种运动估计模式，即可以以这四个帧作为参考。运动估计的匹配原则为选择率失真最小的模式作为最终的运动估计向量。通过这个相关性测试实验，本文对所有帧最终选择的运动估计模式进行了统计，得到如下表 4-3 的结果：

4-3 表 1 相关性测试实验结果[62]

| Sequence Name | T [%] | S [%] | R [%] | L [%] |
|---|---|---|---|---|
| *Ballroom* | 74.98 | 12.12 | 6.86 | 6.04 |
| *Exit* | 76.96 | 8.66 | 7.49 | 6.90 |
| *Uli* | 93.06 | 2.23 | 2.58 | 2.13 |
| *Race1* | 96.64 | 1.35 | 1.06 | 0.96 |
| *Breakdancers* | 57.95 | 19.30 | 12.15 | 10.60 |

实验结果表明，多视点视频序列中大多数帧在预测编码时选择 $I_T$ 帧，即时间上的运动估计更加准确。所以对于多视点视频来说，在统计上其在时间上的相关性比视点间的相关性要强一些，时间上的预测更加准确。

而且对于自由视点视频来说，每次提供给用户的只是当前视点的信息。如果采用原始预测结构，为了获得当前视点视频的信息，可能需要对其相邻的视点也进行解码，而其他视点的信息对于用户来说是不需要的。

因此，对于自由视点视频来说，可以在保证视频质量的前提下，适当减少视点间的依赖关系，以提高其随机访问的性能和可并行性。基于多视点视频发展中提出的一种简单的预测结构，以及上一节中提出的改进预测结构，本文对自由视点视频提出了一种改进的预测结构，其如下图 4-12 所示：

该预测结构一方面取消了原始预测结构中在非关键帧中的视点间参考，只在关键帧上使用视点间预测，从而减少了视点间的依赖关系；另一方面将中间视点的关键帧作为 I 帧，这样由 I 帧到其他视点关键帧的步骤被减少，有利于提高切换速度。

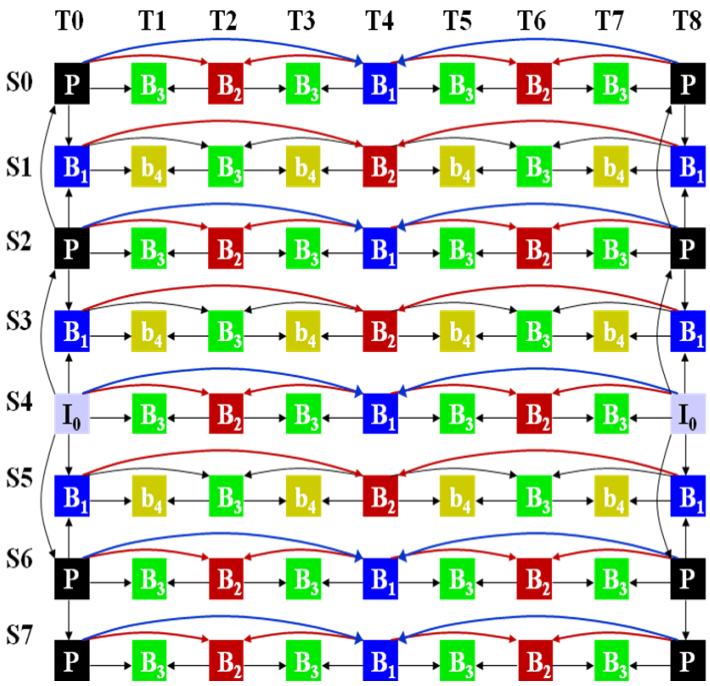
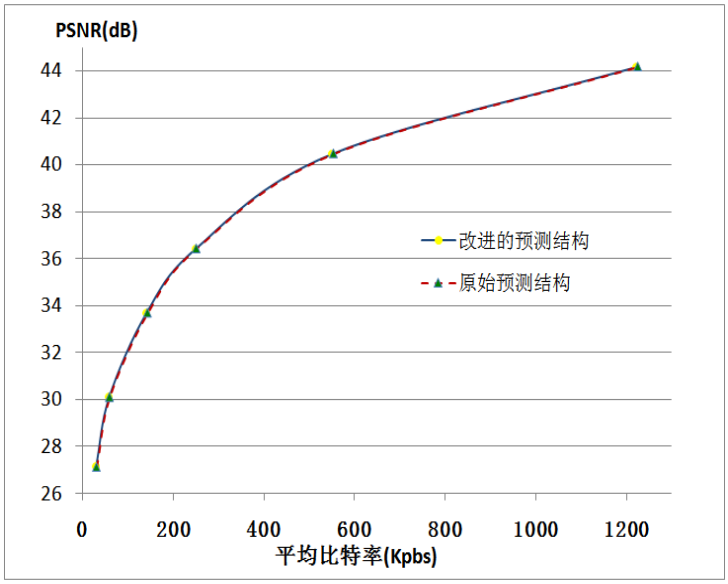



图 4-12 自由视点视频的改进预测结构

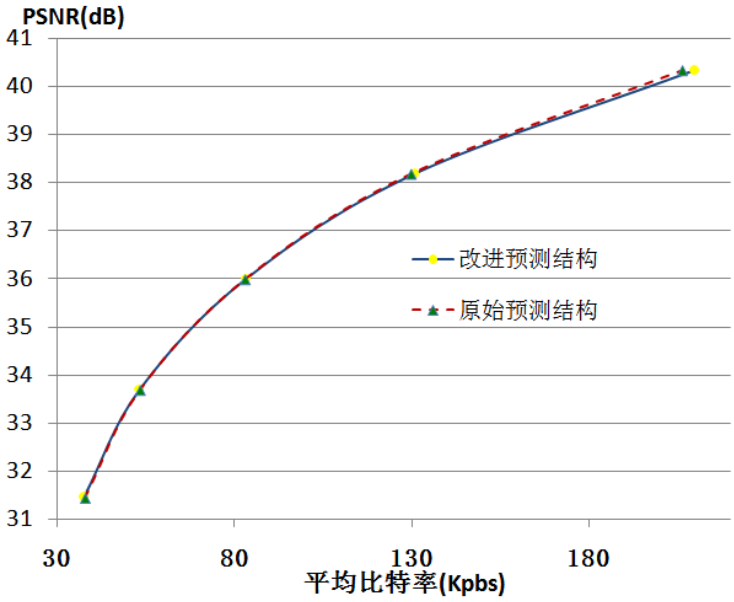
### 4.4.1　改进后的性能分析

通过表 4-3 可以得知，对于多视点视频来说，视点间预测还是有利于其压缩效率的提高的。但在自由视点视频改进的预测结构中取消了非关键帧上视点间预测，所以其肯定对视频压缩性能有一定的影响。但是由于其采用将中间视点作为关键视点的方法，从另一方面获得一些压缩性能的增益，所以其压缩性能不会损失太多，通过实验也证明了这一点。

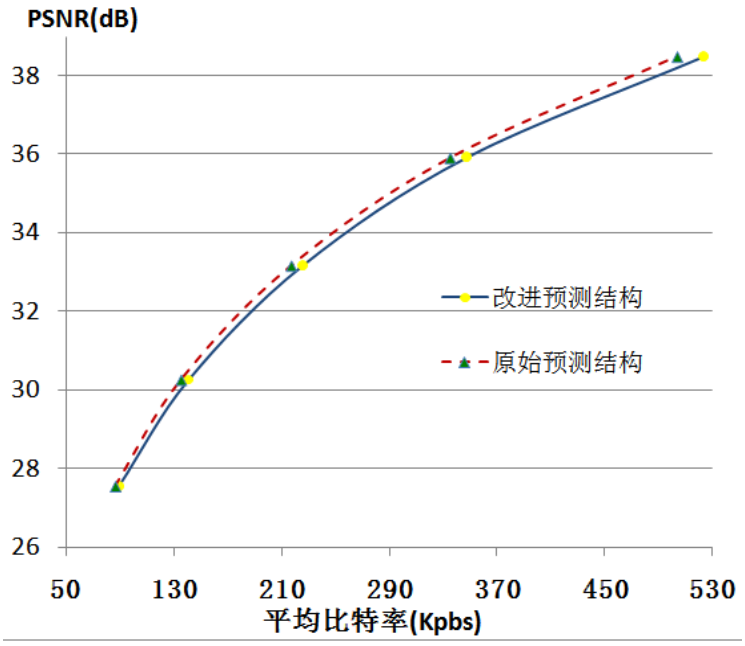
对 Golf1、Flamenco1 和 Objects1 三个多视点视频序列进行 PSNR 的测试，得到如下结果：




( a ) Golf1




( b ) Flamenco1




( c ) objects1

图 4-13 PSNR 测试结果

蓝色实线表示的是改进预测结构的率失真曲线，而红色虚线表示原始预测结构的率失真曲线，通过实验表明改进后的预测结构基本上和原始预测结构的视频质量差不多。而视频 Objects1 的改进后的预测结构比原始预测结构要差一些，这主要是由于视频 Objects1 自身的特性导致的。Objects1 视频的背景光线忽明忽暗，和相关性测试实验中的 Breakdancers 视频类似，由于视频中光线变化比较快，或者运动比较剧烈，导致其降低了时间上的相关性，而视点间的相关性得到增强，所以在取消非关键帧上的视点间预测时，导致其压缩性能有明显的变化，但是损失相对来说比较小，平均损失 0.22dB，而一般视频压缩性能都以 0.5dB 为单位，低于 0.5dB 差距的视频对于人眼来说从视觉上没有明显的变化。所以 0.22dB 的损失在可以承受的范围之内。

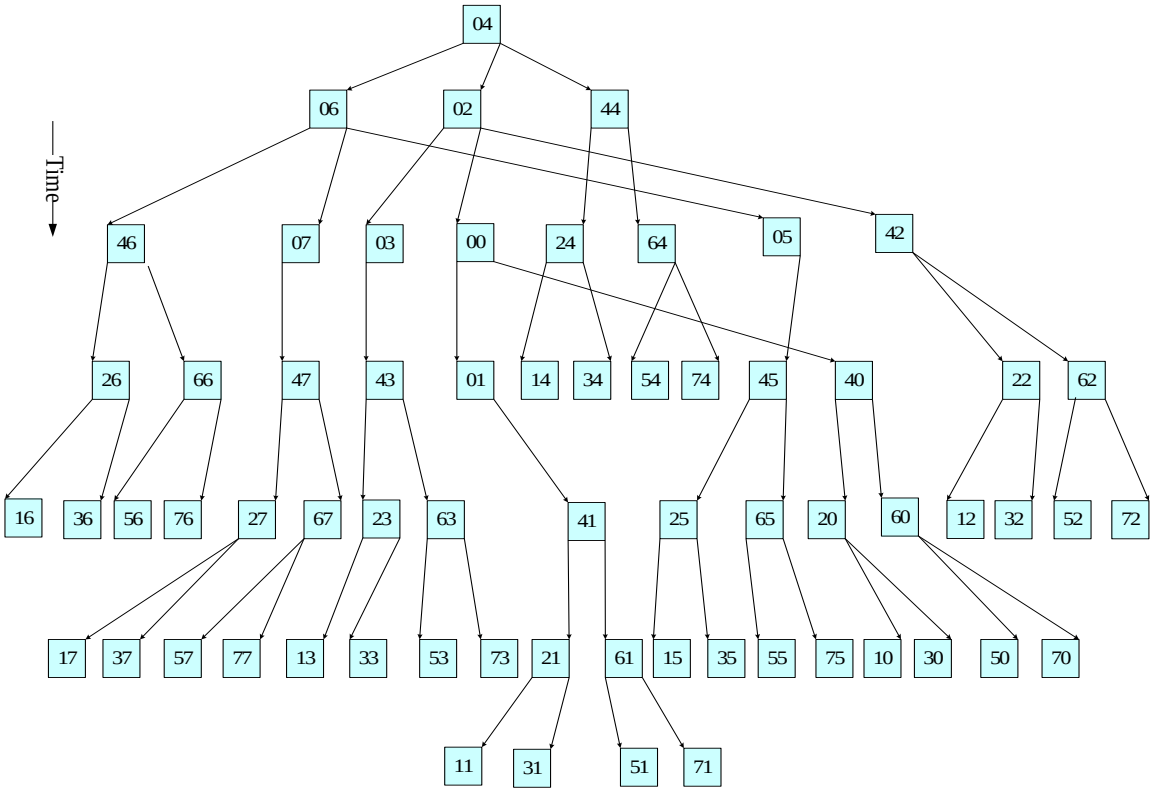
为了对随机访问效率进行分析，本文采用平均切换延迟来对改进后的预测结构和原始预测结构进行性能对比。自由视点视频的随机访问主要是视点切换，而视点切换根据用户进行视点切换时所在的帧的帧类型，包括关键帧切换和非关键帧切换两种方式。无论哪种方式，其最坏的情况就是当前已解码的帧都没法使用，必须从关键视点的关键帧(I 帧)重新开始解码。我们便以从 I 帧切换到其他视点所有帧上所需的平均步骤数来作为平均切换延迟的指标。对于这个值

的计算，可以通过 MVC 的分层 DAG 图来计算，平均切换延迟即 DAG 中从根节点到其他所有节点的平均距离。

假设分层 DAG 中包含 L 层，每层中包含的节点数为 Ni(i=0,1,2….L-1)，那么 MVC 的平均切换延迟 D 的定义如下公式 4-2：

$$D = \frac{\sum_{i=1}^{L-1}(N_i \times i)}{\sum_{i=1}^{L-1} N_i} \qquad (4\text{-}2)$$

原始预测结构和改进预测结构的新的分层 DAG 图分别如下图 4-14 所示：




4-14 自由视点视频的改进预测结构的 DAG

通过计算可得，原始预测结构的平均切换延迟 D=288，而改进预测结构的 D=240，比原始预测结构缩短了 16.67%。

# 第5章 结论

多视点视频作为一种视频技术的发展，越来越受人们的关注。由于其包含了一个场景的多个视点信息，数据量比较庞大，高效的编解码压缩算法是其广泛应用的前提。关于它的编解码压缩标准 MVC 目前正由联合专家小组 JVT 负责起草。高效的压缩算法势必带来很高的算法复杂度，再加上 MVC 本身要处理的数据量很多，目前其处理速度非常慢，这将阻碍多视点视频技术的未来发展和应用。

根据以往的视频编解码发展的规律，可以预见 MVC 标准一旦正式颁布，接下来关于其并行优化的研究就会快速兴起。但是只是依靠修改算法本身来提高 MVC 的处理速度，是远远不能达到实时处理的要求的，这就需要硬件的支持，通过提高硬件的处理速度来满足 MVC 的处理需求。目前单核处理器的发展出现了瓶颈，多核处理器成为体系结构的发展新趋势，于是本文便以多核处理器为平台，利用并行优化技术来充分发挥多核处理器的计算优势，从而实现对多视点视频编解码 MVC 的加速。

文中根据 MVC 自身特点，构造了 MVC 的分层的有向无环图 DAG，基于 DAG 设计了一种 MVC 可并行性的分析模型，用来分析 MVC 的可并行性，以及 MVC 调度算法的并行加速性能。基于 MVC 的分析模型，本文提出了两种启发式并行调度算法，并在 MVC 的参考软件 JMVM 的基础上，设计并实现了 MVC 编码器和解码器的并行优化。实验结果一方面证明了上述启发式调度算法的可行性和优越性，另一方面实际测试结果和可并行性分析模型的理论分析结果是近似的，从而验证了可并行性分析模型的正确性和有效性。

在实验中发现当前 MVC 的编码方法的可并行性相对比较差，不能充分的利用多核优势，为了能够进一步提高 MVC 的处理速度，文章最后从提高 MVC 自身可并行性的角度出发，对 MVC 的预测结构进行了分析，并根据 MVC 未来的不同应用提出了相应的具有高可并行性的改进预测结构。最后的实验结果表明，改进后的预测结构既保证了与原有方法近似的视频质量，甚至更优的视频质量，又提高了 MVC 的可并行性。

# 参考文献

[1] 王春彦,李艺霞，李宏年，刘楷. “H.264/AVC 视频编码新技术研究”，无线通信技术，2005 年第 1 期, pp.55-58

[2] 新标准 H.264 的技术亮点， http://blog.sina.com.cn/s/blog_465bdf0b010001qb.html

[3] R. Yamashita, T. Matsuyama, and K. Hasida, “Standardization of 3D-Video Description,” ISO/IEC JTC1/SC29/WG11, Pattaya, Thailand,Doc. M7779, 2001.

[4] ISO/IEC JTC1/SC29/WG11, “Draft Requirements for 3DAV Coding”, MPEG 2002, N4795, 2002.05

[5] ISO/IEC JTC1/SC29/WG11, “Draft Call for Proposals on Multi-View Video Coding”, MPEG 2006, N6910,Hong Kong, January 2005

[6] ISO/IEC JTC1/SC29/WG 11, “Joint Multiview Video Model (JMVM) 6 Reference Software”, MPEG 2007, N9447, Shenzhen, 2007.10

[7] Ying Chen, Ye-KuiWang, Kemal Ugur. “The Emerging MVC Standard for 3D Video Services”, EURASIP Journal on Advances in Signal Processing, 2008.

[8] D. Cooke, N. O’Connor, and A. Smolic, "Proposal for Specification of 3D Camera Parameters ", ISO/IEC JTC1/SC29/WG11, MPEG04/M11716, Hong Kong, China, January 2005.

[9] ISO/IEC JTC1/SC29/WG11, “Survey of Algorithms used for Multi-view Video Coding (MVC)”, MPEG2005, N6909, Hong Kong, January 2005

[10] ISO/IEC JTC1/SC29/WG 11, “Requirements on Multiview Video Coding,” MPEG 2007/N9543, 2007.

[11] ISO/IEC JTC1/SC29/WG11, “Test Sequences with Different Camera Arrangements for Call for Proposals on Multiview Video Coding”, MPEG 2005 , M 12338, Poznan, July 2005

[12] ISO/IEC JTC1/SC29/WG11, “Fraunhofer HHI test data sets for MVC”, MPEG 2005, m11894, April 2005

[13] ISO/IEC JTC1/SC29/WG11, “KDDI multiview video sequences for MPEG 3DAV use” , MPEG 2004 , m10533, Munich, March 2004

[14] Smolic, A. and Kauff, P., “Interactive 3D Video Representation and Coding Technologies”. Proc. of the IEEE. Special Issue on Advances in Video Coding and Delivery. Vol. 93, No.1. Jan. 2005, pp. 98–110

[15] Philip Benzie, John Watson, Phil Surman, et al. “A Survey of 3DTV Displays: Techniques and Technologies”, IEEE Transactions on Circuits and Systems for Video Technology, Vol.

17, No. 11, Nov. 2007, pp. 1647-1658

[16] 3D 显示技术全面解析, http://www.cgtiger.com/ch/news2.asp?id=407

[17] 王素玉，沈兰荪. “基于对象的视频编码技术”， <<测控技术>>, Vol.25, No.5, 2006, pp.20-23

[18] 翁南钐，蔡德钧. “视频对象分割与两种面向对象的视频编码器”. <<电子学报>>, Vol.28, No.10, Oct. 2000, pp.1-5

[19] ISO/IECJTC1/SC29/WG11, “Overview of the MPEG-4 Standard”, MPEG1998, N2323, Dublin, 1998

[20] Bernd Girod, Anne Aaron, Shantanu Rane, et al. “Distributed Video Coding”, PROC. IEEE, SPECIAL ISSUE ON ADVANCES IN VIDEO CODING AND DELIVERY, January 2005

[21] Rohit Puri, Abhik Majumdar, and Kannan Ramchandran. “PRISM: A Video Coding Paradigm with Motion Estimation at the Decoder”. IEEE Transactions on Image Processing, Vol.16, No.10, OCTOBER 2007, pp. 2436-2448.

[22] 张前进，郭雷. “分布式视频编码关键技术及研究进展”, 《计算机应用研究》, Vol.24, No.8, Aug. 2007, pp.17-21

[23] 徐秋敏，张云等，“多视点视频编码方法研究”,《宁波大学学报》, Vol.19, No.3, 2006, pp.296-301

[24] Shen K., Edward J. Delp. A. “Parallel implementation of an MPEG1 encoder: Faster than real-time”. SPIE Conference on Digital Video Compression: Algorithms and Technologies, San Jose, California, 1995.

[25] Yung N. H.C., Leung K.K. “Spatial and Temporal Data Parallelization of the H.261 Video Coding Algorithm”. IEEE Transactions on Circuits and Systems for Video Technology (CSVT), 2001,11(1): 91-104.

[26] Mattavelli M, Brunetton S, Mlynek D. “A parallel multimedia processor for macroblock based compression standards”. International Conference on Image Processing (ICIP).1997

[27] Bader D A, Patel S. “High performance MPEG-2 software decoder on the cell broadband engine”. IEEE International Symposium on Parallel and Distributed Processing (IPDPS), Miami, Florida USA, 2008:1-10

[28] Chen Yingwei, Zhong Zhun, Lan TseHua, et al. “Regulated Complexity Scalable MPEG-2 Video Decoding for Media Processors”. IEEE Transactions on Circuits and Systems for Video Technology,(CSVT) 2002. 12(8): 678-687.

[29] Martijn J. Rutten, Jos T.J, van Eijndhoven, et al. “A heterogeneous multiprocessor architecture for flexible media processing”. Design & Test of Computers, IEEE, 2002. 19(4):39-50.

[30] Erik B. van der Tol, Jaspers E.G.T. “Mapping of MPEG-4 decoding on a flexible architecture platform”, SPIE Media Processor. 2001.

[31] Franco C., Gianluca D.C., Ronco L. “MPEG-4 Video Decoder Optimization”. IEEE International Conference on Multimedia Computing and Systems (ICMCS). 1999

[32] Chien S.Y., Huang Y.W., Chen C.Y. et al. “Hardware Architecture Design of Video Compression for Multimedia Communication Systems”. IEEE Communications Magazine, 2005:123-131

[33] Erik B. van der Tol, Jaspers E.G.T. Gelderblom R.H. “Mapping of H.264 decoding on a multiprocessor architecture”. Image and Video Communications and Processing SPIE/IS&T. Santa Clara, 2003

[34] Wang Sungwen, Yang Yating, Li Chiaying, et al. “The optimization of H.264/AVC baseline decoder on low-cost TriMedia DSP processor”. Proceedings of SPIE Applications of Digital Image Processing XXVII. Vol.5558, 2004, pp. 524.

[35] Chen T.C., Lian C.J., Chen L.G. “Hardware architecture design of an H.264/AVC video codec”. Proceedings of IEEE International Symposium on Asia and South Pacific Design Automation Conference (ASP-DAC2006), Yokohama, Japan, 2006

[36] Roitzsch M. “Slice-Balancing H.264 Video Encoding for Improved Scalability of Multi-core”.27th IEEE International Real-Time Systems Symposium (RTSS), Brazil, December 2006, pp. 77--80

[37] Wang S.H., Peng W.H., He Y., et al. “A Software-Hardware Co-Implementation of MPEG-4 Advanced Video Coding (AVC) Decoder with Block Level Pipelining”, VLSI Signal Processing, 2005. 41: 93-110.

[38] Baik H., Sihn K.H., Kim Y., et al. “Analysis and Parallelization of H.264 decoder on Cell Broadband Engine Architecture”. IEEE International Symposium on Signal Processing and Information Technology(ISSPIT), Cairo, Egypt, 2007:791-795

[39] Kim Y., Kim J.T., Bae S., et al. “H.264/AVC decoder parallelization and optimization on asymetric multi-core platform using dynamic load balancing”. IEEE International Conference on Multimedia and Expo(ICME), Hannover, Germany, 2008: 1001-1004

[40] Nishihara K, Hatabu A, Moriyoshi T. “Parallelization of H.264 video decoder for embedded multi-core processor”. IEEE International Conference on Multimedia and Expo2008 (ICME 2008), Hannover, Germany, 2008:329-332

[41] Chen Y.K., Li E.Q., Zhou X., et al. “Implementation of H.264 encoder and decoder on personal computers”. Journal of Visual Communications and Image Representation, 2006. 17:509-532.

[42] Nanda A.K., Moulic J.R., Hanson R.E., et al. “Cell/B.E. blades: Building blocks for scalable, real-time, interactive, and digital media servers”. IBM Journal RES & DEV, 2007,51(5): 573 –582

[43] Ville L., Antti H., Hämäläinen T.D. “Complexity of optimized H.26L video decoder

implementation". 2003 IEEE Transactions on Circuits and Systems for Video Technology (CSVT), 2003. 13(7): 717-725.

[44] Berekovic M., Pirsch P., Selinger T., et al. "Architecture of an image redering co-processor for MPEG-4 systems". Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors. 2000.

[45] Yang You, Jiang Gangyi, Yu Mei, et al. "Parallel Process of Hyper-Space-Based Multiview Video Compression". IEEE International Conference on Image Processing(ICIP). 2006

[46] Kemal Ugur1, Hui Liu2, Jani Lainema1 Ugur K., Liu H., Lainema J., et al. "Parallel Encoding: Decoding Operation for Multiview Video Coding with High Coding Efficiency", 3DTV07(1-4). 2007

[47] Pang Yi, Sun Lifeng, Guo Songliu, et al. "Spatial and Temporal Data Parallelization of Multi-view Video Encoding Algorithm". International Workshop on Multimedia Signal Processing (MMSP).2007

[48] Pang Yi, Hu Weidong, Sun Lifeng, Wang Dongsheng, Yang Shiqiang "Parallelized Multi-view Video Coding on Cell Broadband Engine Blade", The Proceedings of the workshop on Cell Systems and Applications (ISCA WCSA) 2008, pp.82-89

[49] Grama A., Karypis G., Kumar V., et al. "Introduction to Parallel Computing". 2003.

[50] Kozyrakis C., Patterson D. "Vector vs. Superscalar and VLIW Architectures for Embedded Multimedia Benchmarks". The 35th International Symposium on Microarchitecture. Instabul, Turkey, 2002.

[51] OHM J R. "Stereo/multiview encoding using the MPEG family of standards". Proceedings of Electronic Imaging'99, vol. 3639, 1999, pp. 242-253.

[52] ITU-T Rec. "the Multi-view Profile", H.262/ISO/IEC 13818-2 MPEG-2 Video

[53] 邢群科，郝红卫，温天江."两种经典实时调度算法的研究与实现"，计算机工程与设计，第 27 卷第 1 期，2006, pp.117-119

[54] C. L. LIU, JAMES W. LAYLAND. "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", Journal of the Association for Computing Machinery, Vo|, 20, No. 1, January 1973, pp. 46-61

[55] Nicholas Mastronarde, Mihaela van der Schaar, "A Queuing-Theoretic Approach to Task Scheduling and Processor Selection for Video Decoding Applications", IEEE Transactions on Multimedia, Vol.9, No.7, Nov. 2007, pp.1493-1507

[56] Yanbing Li, Wayne Wolf. "Scheduling and allocation of single-chip multiprocessors for multimedia", IEEE Workshop on Signal Processing Systems, 2003

[57] P. Kauff, A. Smolic, P. Eisert, et al, "Data Format and Coding For Free View point Video", In Proceedings of International Broadcast Conference, Amsterdam, The Netherlands, Sep. 2005

[58] 毕厚杰，“新一代视频压缩编解码标准 –H.264/AVC”，人民邮电出版社，2007

[59] Iain E.G. Richardson，”H.264 and MPEG-4 Video Compression for Next-generation Multimedia”，国防科技大学出版社，2004

[60] H. Schwarz, D. Marpe, and T. Wiegand, “Analysis of hierarchical B pictures and MCTF,” IEEE International Conference on Multimedia and Expo2006 (ICME 2006), Toronto, Canada, Jul. 2006.

[61] D. Tian, M. M. Hannuksela, and M. Gabbouj, “Sub-sequence video coding for improved temporal scalability,” in Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS ’05), Vol. 6, Kobe, Japan, May 2005, pp. 6074–6077

[62] M. Magnor, P. Eisert, B. Girod, “Model-Aided Coding of Multi-Viewpoint Image Data”, ICIP’00, Vancouver, Canada, 2000, pp. 919-922.

[63] W. H. Leung and T. Chen, “Line-Space Representation and Compression for Image-Based Rendering”, Carnegie Mellon Technical Report: AMP01-02.

[64] M. Magnor and B. Girod, “Model-Based Coding of Multi-Viewpoint Imagery”, VCIP’00, Perth, Australia, June 2000.

[65] D. Taubman and A. Zakhor, “Multirate 3-D subband coding of video”, IEEE Trans. on Image Processing, Vol. 3, No. 5, Sep. 1994, pp. 572-588.

[66] L. Luo, Y. Wu, J. Li, and Y. Zhang, “Compression of concentric mosaic scenery with alignment and 3D wavelet transform”, SPIE: Image and Video Communication. And Processing 2000, San Jose CA, January 2000, pp. 89-100.

[67] Y. Wu, C. Zhang and J. Li, “Smart-Rebinning for Compression of Concentric Mosaic”, IEEE Trans. on Multimedia, Vol. 4, No. 3, Sep. 2002, pp 332-342.

[68] Philipp Merkle, Aljoscha Smolic, Karsten Müller, Thomas Wiegand, “Efficient Prediction Structures for Multiview Video Coding,” IEEE Transactions On Circuits and Systems for Video Technology, Vol.17, 2007, pp.1461-1473.

[69] Barry Wilkinson Michael Allen 著, 陆鑫达(译), “并行程序设计(Parallel Programming: Techniques and Applications Using Networked Workstation and Parallel Computers)”, 机械工业出版社, 2002

[70] 何元清，孙世新，傅彦，“并行编程模式及分析”,《电子科技大学学报》,Vol.31 ，No.2 ,2002, pp.173-17

[71] 图论算法—拓扑排序,http://www.bioisland.com/Algorithm/ShowArticle.asp?ArticleID=57

[72] 钟玉琢，蔡莲红。“多媒体计算机技术基础及应用”，北京:高等教育出版社，1999

[73] ISO/IEC JTC1/SC29/WG11, “JMVM Software Manual”,MPEG 2007, Nov. 2007

# 致　谢

本课题承蒙国家 973 项目和自然科学基金资助，特此致谢。

衷心地感谢我的导师杨士强教授，无论是在科研还是工作上，他都对我严格要求，培养了我踏实勤奋，严谨细致，一丝不苟的作风，使我终生受益；感谢孙立峰老师，从最开始的课题选择，到项目的最终完成，孙老师一直给予我各方面的精心指导，帮助我掌握了正确的科研方法，为我指明了努力的方向。

感谢合作导师文铁华研究员和李洁研究员在生活和科研上为我答疑解惑，给予我很多的帮助；感谢胡事民老师无微不至的关怀和孜孜不倦的教导，激励着我不断前进。

感谢那些陪我一起走过研究生这三年的实验室的师兄弟、师姐妹，以及计研八班的全体同学对我的热情帮助和支持。

最后特别感谢我的家人，感谢他们对我的默默支持，他们永远是我前进的动力。

# 声　明

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。

签　名：　　日　期：

# 个人简历、在学期间发表的学术论文与研究成果

## 个人简历

1983 年 08 月 25 日出生于河南省延津县。

2002 年 9 月考入清华大学计算机科学与技术系与技术系，2006 年 7 月本科毕业并获得工学学士学位。

2006 年 9 月免试进入清华大学计算机科学与技术系攻读硕士学位至今，师从杨士强教授。

## 发表的学术论文

[1] 张凤妍，庞一，孙立峰，杨士强.《多视点视频解码并行处理的设计方案》，第 17 届全国多媒体学术会议，2008， 武汉。被推荐到电子学报。

[2] 庞一，张凤妍，孙立峰，杨士强.《面向多核处理器的视频编码并行加速算法综述》，已被《计算机科学与探索》录用。

[3] Yi Pang, Lifeng Sun, Jiangtao Wen, Fengyan Zhang, Weidong Hu, Wei Feng and Shiqiang Yang， A Framework for Heuristic Scheduling for Parallel Processing on Multi-core Architecture - A Case Study with Multi-view Video Coding (MVC), Submitted to IEEE Transaction on CSVT.

[4] 孙立峰，庞一，胡伟栋，李彦洁，张凤妍，冯威，杨士强.专利《一种用于多核处理器的多视点视频编码的任务调度方法》，申请号：。

## 研究成果

学习期间，参与了自然科学基金--交互多视点视频编码技术研究项目，以及国家 863 项目--自由视点网络视频服务（FreeView Video）关键技术研究。