

西 南 交 通 大 学

毕 业 设 计

# 基于深度图的立体视频编码相关算法研究

年 级： 2010 级

学 号： 20102870

姓 名： 程 柳

专 业： 通信工程

指导教师： 彭 强

二零一四年六月



西南交通大学本科毕业设计

院系 信息科学与技术学院 专 业 通信工程

年 级 2010 级 姓 名 程 柳

题 目 基于深度图的立体视频编码相关算法研究

指导教师

评 语 \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

指导教师 \_\_\_\_\_ (签章)

评 阅 人

评 语 \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

评 阅 人 \_\_\_\_\_ (签章)

成 绩 \_\_\_\_\_

答辩委员会主任 \_\_\_\_\_ (签章)

年 月 日



## 毕业设计（论文）任务书

班 级 通信工程三班 学生姓名 程 柳 学 号 20102870

发题日期： 2013 年 12 月 20 日 完成日期：2014 年 6 月 9 日

题 目 基于深度图的立体视频编码相关算法研究

### 1、本论文的目的、意义

当下，多视点视频（Multi-View Video, MVV）系统中，3DTV（Three Dimensions Television）率先开始逐步应用并且具有相当大的优化空间，而现在开始广泛应用的立体视频主要依靠于双目效应，从而达到立体感官效果。基于深度图的视频（Layered Depth Video, LDV）和多视点视频加深度（Multi-View Video plus Depth, MVD）作为第二代 3DTV 中的重要理论有着广阔的发展空间。LDV 能明显减少多角度视频整体数据量，且观看测试结果显示能够维持比较好的视觉质量，具有更深层的商业意义，包括立体电影、视频会议、卫星成像等对深度信息需求同时要求尽可能低的数据速率。通过深度图的深度信息，观众视点的图像可以利用对深度图的译码和计算来获取其他视点的虚拟纹理图。此外，为了以尽可能少的代价获得更佳的实际效果，深度图的捕捉、虚拟视点的生成、深度图的压缩和合成视点的修复技术也需要引入以解决实际需求中的难题。多视点图像是由处于空间不同位置的摄像机阵列拍摄同一场景得到的一组视频序列信号。随着信息产业的蓬勃发展，多视点视频带有多维度并更具真实感的三维视觉特性以用户需求为基础推广开来。因此，多视点视频成为当前视频领域的研究热点之一。

本次课题的目的在于了解基于深度图的立体视频的相关处理步骤，进行相关算法的编码、解码以及视点合成的模拟，通过修改相关算法软件的配置信息，对获得的一系列输出文件以及软件控制台输出信息进行归纳整理，比较相关算法对视频的压缩效率与信噪比。本次课题意义在于了解当前相关领域的现状与最新进展，并能熟悉已知的立体视频视点合成过程和对压缩算法进行效率比较，为今后深造或就业打下坚实基础。

2、学生应完成的任务

1. 学习和了解立体图像编码的原理和实现方法，掌握立体视频编码、基于深度图立体视频合成算法的基本原理

---

2. 收集查阅现有基于深度图的立体视频预测编码算法的相关资料

---

3. 重点对现有立体视频图像编码算法进行分析研究，熟悉 JMVM 多视点视频编码测试模型等相关开发软件包

---

4. 完成基于深度图的立体视频编码算法实验，重点对视点预测、立体视频编码算法进行实验验证并分析

---

5. 撰写毕业论文

---

3、论文各部分内容及时间分配：（共 15 周）

第一部分 学习和了解立体图像编码的原理和实现方法 (4 周)

第二部分 收集查阅现有基于深度图的立体视频虚拟视点合成算法 (1 周)

第三部分 对现有立体视频图像编码算法进行分析研究，熟悉 JMVM (4 周)

第四部分 完成基于深度图的立体视频编码算法实验验证及分析 (4 周)

第五部分 撰写毕业论文 (2 周)

评阅及答辩 (1 周)

备 注

---

---

---

指导教师： 2013 年 12 月 20 日

审 批 人： 2014 年 月 日

## 摘 要

基于深度图的立体视频是近年立体视频领域的研究热点之一，其主要过程包括了视点深度图的捕捉、虚拟视点的合成、数据的压缩算法和图像的修复。深度图的压缩算法因对降低数据带宽的突出效果而成为了商业前景极高的研究部分。

根据基于深度图的立体视频研究的国内外现状的分析，本文对这四部分进行了整理与综合阐述。视点合成和编解码过程在 3DV-ATM 参考模型中模拟实验。此外，作为实验的研究对象，不同预测算法的核心参数如压缩效率和峰值信噪比（PSNR）被论述。

在绪论中，详细介绍了本课题的研究意义以及当下的国内外现状。相关的基础知识在第二章中引入以供后续分析，与此同时本章节包含了课题的重点——深度图。就视频的编解码方面，三类预测编码算法（视点合成预测 VSP、基于深度的运动估计 DMVP、自适应亮度补偿 ALC）作为本课题的重点研究和实验对象进行了详细的讨论。第三章简要介绍了视点合成与深度层视频特有的冗余层数据。在实验章节中，提出了使用现今流行的 R 语言作为数据分析工具对实验数据进行整理、绘图和分析，同时在 Github 中公开了完整代码和实验数据作为开源素材以供学习和研究。实验结果显示，DMVP、ALC 和 VSP 三类算法分别能减少 27.9%、6.1% 和 0.9% 的比特率，而 PSNR 值方面，三种算法分别增加了 0.278dB、0.307dB 和 0.003dB。结果比较中可以看出 DMVP 在效率上的优势和 ALC 在比特率上的优势，同时得出 VSP 算法实践上并没有突出效果，进一步验证了算法研究的意义。除此之外，实验验证了纹理量化参数每增加 2 能够减少 15.9% 的比特率但同时也减少了 1.134dB 的 PSNR 值。最后，论文总结了实验结论和实验中遇到的困难。

**关键词：**视频编码； 视点合成； 数据分析； 深度图视频

## Abstract

3DTV with corresponding depth information has raised attention of multi-view video research in recent years. Main processes for video consist of getting depth map, virtual view synthesis, data compress and image inpainting. Since depth information can be compressed to reduce data rate, algorithms of compressing take an important commercial role nowadays.

Based on conditions of research, four parts of procedure are investigated in this paper. View synthesis, encoding and decoding processes are tested in reference model (3DV-ATM). Furthermore, key parameters, such as bit rate and PSNR (Peak Signal to Noise Ratio), are compared by varying specific variables for experimental purpose.

I address the significance of study and topics related to them. Basic knowledge of video as well as depth map is introduced. In addition, video encoding theory, including three prediction algorithms: View Synthesis Prediction (VSP), Depth Based Motion Vector Prediction (DMVP) and Adaptive Luminance Compensation (ALC), are discussed separately for experiment. Residual layer as an extension video data format is involved as well as key points of View Synthesis. Moreover, Image inpainting technique is briefly introduced as part of Criminisi's work. R language is a promising data analysis tool and popular worldwide; thus, I propose an advanced method to analyze data based on it and post complete project with code and data as open source in Github. In my experiment, analyzing result denotes 27.9%, 6.1% and 0.9% of bit rate reducing which are influenced by DMVP, ALC and VSP respectively. In terms of PSNR value, 0.278dB, 0.307dB, and 0.003dB can be obtained after applying different algorithms. Obviously, DMVP algorithm has the advantage in reducing data while ALC algorithm does in increasing PSNR and VSP has nearly no improvement practically, which illustrates the importance of research of compressing algorithm. In addition, bit rate will decrease 15.9% with increasing texture quantization parameter by 2, and yet decreasing PSNR by 1.134dB. At last, conclusion and difficulties are summarized.

**Key words:** Video Coding, View Synthesis, Data Analysis, Video plus Depth



## 目 录

摘 要 .....	III
Abstract .....	IV
第 1 章 绪 论 .....	1
1.1 课题背景与意义 .....	1
1.2 国内外发展（应用）现状 .....	1
1.3 论文所做工作及思路 .....	3
1.4 论文结构安排 .....	4
第 2 章 基于深度图的立体视频编码相关 .....	6
2.1 立体视频数据格式 .....	6
2.2 深度数据感知 .....	7
2.3 坐标系变换 .....	9
2.4 视频编解码 .....	10
2.4.1 编码基本概念 .....	12
2.4.2 视点合成预测 .....	12
2.4.3 基于深度的运动估计 .....	12
2.4.4 自适应亮度补偿 .....	12
第 3 章 视点合成与图像修复 .....	16
3.1 视点获取与合成 .....	16
3.1.1 3D 扭曲 .....	16
3.1.2 空洞与堵塞 .....	16
3.1.3 视点的合成 .....	17
3.1.4 质量增强 .....	12
3.2 冗余层数据 .....	19
3.3 图像修复 .....	20
第 4 章 实验数据与结果 .....	21
4.1 实验概要 .....	21
4.1.1 实验目的 .....	12
4.1.2 3DV-ATM 模型 .....	21
4.1.3 量化参数 .....	22
4.2 实验方案 .....	22

4.2.1 视频配置参数修改 .....	22
4.2.2 实验方案 .....	23
4.3 实验数据分析 .....	26
4.3.1 数据分析的意义 .....	25
4.3.2 R 语言与 RStudio .....	25
4.3.3 观测值数据整理 .....	26
4.3.4 变量提取 .....	27
4.3.5 数值获取与绘图 .....	29
4.3.6 实验结果分析 .....	31
结 论 .....	38
致 谢 .....	39
参考文献 .....	40
附 录 1 数据分析平台搭建 .....	42
5.1 下载 .....	42
5.1.1 R 语言基础包下载 .....	42
5.1.2 RStudio 下载 .....	42
5.1.3 Git Bash 下载 .....	43
5.2 安装 .....	44
5.2.1 R 语言 CRAN 基础包安装 .....	44
5.2.2 RStudio 安装 .....	45
5.2.3 Git Bash 安装 .....	45
5.3 创建与配置 .....	45
5.3.1 克隆 R 语言脚本 .....	45
5.3.2 创建工程 .....	46
5.3.3 修改配置文件 .....	47
附 录 2 脚本代码 .....	48

# 第 1 章 绪 论

## 1.1 课题背景与意义

当下,多视点视频(Multi-View Video, MVV)系统中,3DTV(Three Dimensions Television)率先开始逐步应用并且具有相当大的优化空间,而现在开始广泛应用的立体视频主要依靠于双目效应,从而达到立体感官效果。基于深度图的视频(Layered Depth Video, LDV)和多视点视频加深度(Multi-View Video plus Depth, MVD)作为第二代 3DTV 中的重要理论有着广阔的发展空间。LDV 能明显减少多角度视频整体数据量,且观看测试结果显示能够维持比较好的视觉质量,具有更深层的商业意义,包括立体电影、视频会议、卫星成像等对深度信息需求同时要求尽可能低的数据速率。通过深度图的深度信息,观众视点的图像可以利用对深度图的译码和计算来获取其他视点的虚拟纹理图,而近几年达成的一致观点是:只有在用户观看的 3D 图像的图像质量以及用户感受到的视觉映像达到传统的 2D 电视的水准,3D 图像才有可能被接受。为了以尽可能少的代价获得更佳的实际效果,深度图的捕捉、虚拟视点的生成、深度图的压缩和合成视点的修复技术也需要引入以解决实际需求中的难题。多视点图像是由处于空间不同位置的摄像机阵列拍摄同一场景得到的一组视频序列信号。随着信息产业的蓬勃发展,多视点视频带有多维度并更具真实感的三维视觉特性以用户需求为基础推广开来。因此,多视点视频成为当前视频领域的研究热点之一。

本次课题的目的在于了解基于深度图的立体视频的相关处理步骤,进行相关算法的编码、解码以及视点合成的模拟,通过修改相关算法软件的配置信息,对获得的一系列输出文件以及软件控制台输出信息进行归纳整理,比较相关算法对视频的压缩效率与信噪比。本次课题意义在于了解当前相关领域的现状与最新进展,并能熟悉已知的立体视频视点合成过程和对压缩算法进行效率比较,为今后深造或就业打下坚实基础。

## 1.2 国内外发展(应用)现状

带有深度图的立体视频已成为众多学者的研究对象,目前的主流立体视频格式包括传统的立体视频,多视点视频(Multi-view Video),视频加深度(Video plus Depth),多视点视频加深度(Multi-view Video plus Depth)和带深度层的视频(Layered Depth Video)<sup>[1]</sup>。基于深度图的 3DTV(即 3D 视频)系统的关键技术包含信息的生成、数据的压缩与传输、3D 视觉化与视频质量的评估四个过程<sup>[2]</sup>,是立体视频领域的研究重点。深度图的获取可以通过 TOF(Time of Flight, 光束飞行时间)摄像机捕捉深度信息<sup>[3]</sup>,而置放多个纹理摄像机和 TOF 摄像机捕捉多路信息可以同时输入的多路数据流与

多视点视频获取与 LDV 格式有关的冗余层信息<sup>[4]</sup>。

获取深度图的核心价值在于用更少的数据量来传输视频，并且所需的更多视点的数据依靠的是结合深度图的视点合成方法。多数情况下，3 个视点的视频信号会被传输，利用视点合成技术生成中间视点，即虚拟视点<sup>[5]</sup>。然而，多个视点的立体视频序列会大大增加数据处理带宽，数据量是目前存储介质以及带宽容量所无法达到的。为此而发展的视点合成技术在基于深度图像上，估计并重建其他视点视频，从而显著减少数据带宽<sup>[6]</sup>。视点合成时使用的纹理和深度信息对用户来说实际上都是不可见的，多数情况下并不需要两者质量都是最优的。研究表明，两者在一个合适的比例范围内一样可以给用户比较好的视觉感。因而，结合深度数据的压缩编码也成了热点研究对象之一<sup>[7]</sup>。

3DV-ATM 是针对多视点视频加深度的推荐参考模型。对于 MVD 格式而言，3DV-ATM 不仅提供了 3D-AVC，同时也提供了 MVC+D（即 MVD）的格式支持<sup>[8]</sup>。该模型提供了编解码所需的配置信息，并能根据实验需求修改配置文件，为测试者提供了方便，国内外许多学者通过 3DV-ATM 对视频的编解码做了效率方面的测试。

Merkle P 对不同的多视点视频编码的短暂和帧间预测做了实验分析。利用统计学的角度针对这两类测试了运动补偿预测<sup>[9]</sup>。结果表明了短暂性的参考图像预测具有很高的效率，并且从相邻视点做参考而做出的预测会高出 20% 的，因而效率更高。当使用关键帧帧间编码预测时的平均增益约为 1.4dB 的峰值信噪比，而非关键帧的结果具有 1.6dB 的峰值信噪比。由此得出增益每增加 3dB 比特率将会节省 50% 的结论。此外，Yamamoto 提出了两种方法来改进压缩效率，一种是先插值后根据插值结果作为参考图像，即通过几何方法先将参考视点精度变高再进行预测编码。而另一种方法通过遍历搜索的方式改进了其他视点的亮度和色度。实验中作者验证了这些做法的有效性和可行性，在不减少峰值信噪比的情况下，比特率减少了近 15%<sup>[10]</sup>。

在视点合成预测（View Synthesis Prediction, VSP）编码方面。Yea 提出了改进性的方案，针对比特率失真量的框架来提升多视点视频编码的预测效果，深度数据同时在编码端和解码端都用来生成视点合成预测的数据。实验中使用了两个组别，第一组使用了 CABAC 的编码次序来基于宏块和深度数据的多视点视频编码，致力于提升效率，实验结果显示视点合成预测在使用了校验补偿预测时增进了编码效率。第二组实验里通过分离编码多视点深度图而减少了数据速率。由于现有的视点合成预测方法利用的是其他视点或者是几何关系来预测，这些传统的方法无法补偿帧间差异较大的图像<sup>[11]</sup>。Shimizu 提出适应性的外观补偿视点合成预测：使用了自适应滤波器来补偿包括纹理、亮度和空间差异的图像，优化同样基于块。实验结果表明了所使用的方法能够减少在 H.264/AVC 格式下多视点视频编码约 40% 的比特率，相比传统视点合成预测的 26% 减少量多了额外的 14%<sup>[12]</sup>。

在基于深度的运动估计（Depth Based Motion Vector Prediction, DMVP）算法方面。Su 根据现有的模型环境，使用了传统的三路多视点视频加深度的立体视频数据格式，结果显示输出的视频流提升了近 7.5% 的压缩效率<sup>[13]</sup>。

参考文献资料以及其他人对基于深度的立体视频编码算法的研究，发现对基于深度图的视频编码算法主要集中在深度信息的捕捉、数据量的控制、数据压缩、以及图像修复等方面。本课题拟研究包括深度图的信息的捕捉、视点合成（View Synthesis）、图像修复技术（Criminisi 提出的修复算法）和包括 3D 扭曲（3D Wrapping）、消除空洞、生成冗余层等相关算法和过程。另外，压缩数据率对带宽的额外节省能有效改善用户的体验质量，分析比较相关算法的效率使本课题的研究十分有必要性，进而本课题研究了包括量化参数（Quantization Parameter）、视点合成预测（VSP）、基于深度的运动估计（DMVP）和自适应亮度补偿（ALC）算法在峰值信噪比与比特率方面的效果。

### 1.3 论文所做工作及思路

高清晰度的电视（High Definition Television, HDTV）已经伴随近年数据带宽的扩大而广泛应用开来，但是，高清晰度电视仍然缺少了人类自然感官中的立体感，即深度信息的感知。另一方面，设备制造商研发出了可以将多个视点通过裸眼或佩戴眼镜方式的 3D 电视。而正如前文所述，3D 视频只有在用户感受到的质量不低于传统 2D 影像时才有可能被应用，减少多视点视频（Multi-view Video, MVV）的传输数据量就显得尤为重要，当下最热门的是视点合成（View Synthesis）技术。就视点合成而言，通常有两类：基于几何的修复方式（Geometry-Based Rendering, GBR）和基于图像的修复方式（Image-Based Rendering, IBR）。由于基于几何的修复方式（GBR）需要诸如了解纹理、对象建模等等，是有着相当高的运算代价，因而没有广泛应用。这里着重说明后者，也就是基于图像的修复方式（IBR）。之前提到的深度图视频（Layered Depth Image, LDV）就是 IBR 的一种应用。事实上，由于图像的组合形成了视频，所以 IBR 最先衍生出深度图像（Layered Depth Image, LDI），而后结合了视频的帧间预测编码以及运动向量预测等方式而衍生出深度图视频（Layered Depth Video, LDV），是在数据传输量上较低的视频编码方式。

LDV 提供了完整的中心视点（Central View）和其他边界视点（Side View）的纹理和深度作为冗余层数据（Residual Layer Data）数据。因而，LDV 是在多视点视频加深度（MVD）基础上扩展的。传统的 MVD 传输了所有视点的纹理与对应的深度图，而 LDV 仅仅传输中心视点的完整信息，而冗余层信息相对有着很少的数据。然而，问题在于中心视点向边界视点映射的过程中并不是每个像素在都会存在对应的，因而造成了空洞（Hole）和堵塞（Occlusion）的出现，为此，视点合成技术通过比较映射过程中重叠和缺少区域而解决了空洞和堵塞问题。

Kalman 滤波的方式被提出,针对运动物体产生的空洞和堵塞。对于通过预处理深度图来减少深度数据的不连续性从而处理堵塞的方式,会造成深度图视频的较多失真。通过增加维度来移除堵塞是可行的,但是因而增加的数据量会对数据带宽要求也随之增加<sup>[14]</sup>。

图像的修复可以依靠缺失像素的邻近像素信息作为预测,来填补缺失的像素。视频中还采用了帧内预测编码、帧间预测编码、运动估计和更复杂的熵编码之类的方式来预测和降低数据量<sup>[15]</sup>。图像和视频的修复有着很广泛的应用领域,例如移除视频中的 Logo 水印和文字水印,恢复老旧或者磨损的图片的原貌等。Criminisi 的工作成果作为图像修复理论将被作为理论基础简要描述<sup>[16]</sup>。这是一个尝试连接图像结构性和纹理性的修复方法:纹理会在某些方向上呈现一致性,额外的深度信息可以区别一个像素属于前景还是背景。

本文将首先介绍相关的基础知识,从提及深度图的信息的捕捉、视点合成中的基于深度的图像转换、图像修复技术、3D 扭曲、消除空洞和生成冗余层几个方面逐步介绍关联深度图的多视点视频的处理。其中,着重分析研究编码过程中的几个重要算法:量化参数(Quantization Parameters, QP),视点合成预测算法(View Synthesis Prediction, VSP),基于深度的运动估计(Depth-Based Motion Vector Prediction, DMVP)和自适应亮度补偿算法(Adaptive Luminance Compensation, ALC)。横向和纵向比较不同算法编码时的压缩效率、峰值信噪比以及对比特率的影响,同时创造性地利用当下流行的用于数据分析的 R 语言,实现在 RStudio 软件平台下对得到的多组数据进行整理、绘图和特性分析。

## 1.4 论文结构安排

论文共分 四 章。

第 1 章 绪论,介绍本课题的课题意义和背景。通过分析国内外文献,叙述有关立体视频和深度图视频的研究现状,据此,明确本文的研究工作和深度视频获取的过程,从而展开了其余章节的详细阐述。

第 2 章 背景,介绍了当下主要的立体视频数据格式,比较各种数据之间的优势和研究意义,同时简要介绍了对深度数据的捕捉过程和坐标系变换。视频编解码作为本文的核心工作在这一章中将会介绍三类算法的具体原理(视点合成预测 VSP、基于深度的运动估计 DMVP 和自适应亮度补偿 ALC)作为基础知识供后续分析所用。

第 3 章 视点合成与图像修复,本章是深度视频数据的拓展,核心为视点合成步骤。其中冗余层作为 LDV 特有的数据层会被讨论,而图像修复主要以 Criminisi 提出的研究算法为基础简要介绍。

第 4 章 实验数据与结果,这一章是基于视频加深度的编解码部分的算法(在第

二章中介绍)所做的实验,对实验所得的数据进行分析比较得出有关结论。

结论是针对本文所做工作的总结。

附录 1 和附录 2 是本文创新点---利用 R 语言分析实验数据,涵盖了包括平台搭建和脚本代码。

## 第 2 章 基于深度图的立体视频编码

### 2.1 立体视频数据格式

根据各界学者们大量的研究分析，立体视频数据的格式主要从以下几个方面讨论其优劣性：

- 使用尽可能多的已有传输设施与媒介
- 改变最少的显示图像所涉及的设备
- 后向兼容性
- 可以支持多种类的显示设备，并且具有拓展性
- 高质量<sup>[3]</sup>

通过使用左眼（左视点）观测的视频和右眼（右视点）观测的视频提供的双视点的视频数据，即立体视频数据。这是立体视频有别于传统 2D 视频的重要特征，2D 视频仅仅包含一个视点的视频数据。而立体视频中左视点加右视点的视频数据包含了两路视频数据，通常称为两路通道。

在立体视频的标准化过程中，多视点视频（MVV）可以被认为是多路通道视频数据，是作为现有立体视频数据格式的拓展。多视点视频同时在某个特定视点区域捕捉和处理多个视点，如图 2-1-1，而获得近似的连续视点。可是，相比传统的 2D 视频，多个视点意味着需要传输的数据会根据视点的数量而线性增加。而在现有条件下，多视点视频（MVV）格式不仅对数据带宽有着极高的要求，同时也对存储介质有巨大的压力。同时，立体显示的需求集中在修复立体视频输出视点的连续性，并且能在解码端输出大量的不同视点。因此，MVV 格式无法满足增加额外输出点和节省带宽的需求。



图 2-1 多视点图像

由此，视频加深度（Video plus Depth）的数据格式被引入，用来提供直接性的视觉响应，并满足节省带宽的要求。在计算机视觉领域中，视频加深度的格式提供了一



个带有深度层的 2D 视频。2D 视频提供了纹理信息，色彩密度以及图像的结构，而深度视频表示的是各个像素的 Z 轴距离，即从摄像头的光学中心到物体空间的立体点，如图 2-2。

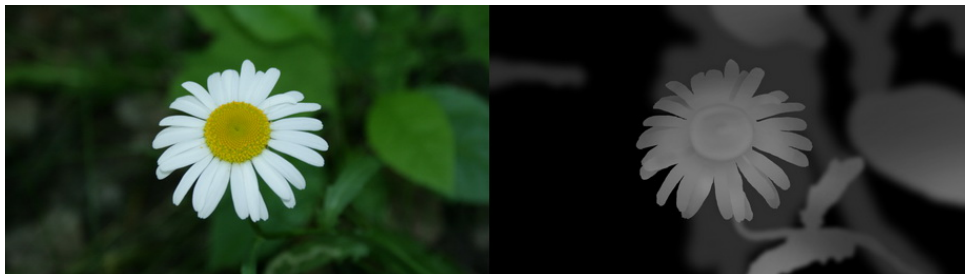


图 2-2 纹理与对应深度图

利用深度图像修复技术（Depth Image Based Rendering, DIBR），可以通过其他视点的深度视频和已知的 2D 视频，对其他视点的纹理进行重构。然而，由于视点合成引起的变化会由于不同视点间的距离而快速变化，视频加深度只能支持在原始视点附近的具有非常有限的连续性的视点。此后，标准化进程加入了多个视频加多个深度的数据格式（即关联了视频加深度与多视点视频），进而形成多视点视频加深度（Multi-view Video plus Depth, MVD）的格式。多视点视频加深度（MVD）包含了多个视点的 2D 视频，而每个视点的 2D 视频都有各自的深度视频。

通过 DIBR 技术和多个视点数据可以得到的需要修复的中间视点，从而保证视频的连续性。从这个角度，所有数据将会被完整地处理和传输，但这又对数据带宽要求，所以深度视频加数据（Layered Depth Video, LDV）发展起来。在 LDV 中，会定义一个中心视点（Main Viewpoint），而其他视点被认为是边视点（Side Viewpoint）。中心视点有完整的纹理图和深度图，而边视点只有深度图。由于不同的边视点的深度图有不同程度的相似性，在合成边视点纹理图的过程中会得到一些近似重复的像素点，从而引入了冗余层（Residual Layer）的概念，针对不同深度视点映射到同一视点引发的堵塞（Occlusion）或空洞（Hole）做出修正，这一部分将会在第三章详细说明。相比多视点视频加深度（MVD），LDV 没有了边视点的纹理图，增加的冗余层也只有很少的数据量，因此有着更低的比特率。

## 2.2 深度数据感知

近年来，半导体技术的发展带来了利用光束飞行时间测量物体距离用的 TOF（Time of Flight）摄像头，来直接捕捉深度视频，也被称作深度摄像头。TOF 摄像头基于光的反射和摄像头的光传感器技术，能够在真实时间里测量摄像头和物体之间的距离。摄像头所发出的红外光线能够被物体反射然后返回摄像头的传感器。因此，可

以得到每一点的像素的光束的往返时间，从而用以计算物体的深度。深度视频可以被认为是只具有单色的没有纹理的视频信号。通常，深度数据会被量化为 8 比特(bits)，最近的点会被量化为 255 而最远的点会被量化为 0。通过这种方式，深度视频被表示为一个平滑的灰度值表示<sup>[3]</sup>,具体计算公式如下:

$$z = f \left[ \frac{d}{2^{BitDepth}-1} \left( \frac{1}{Z_{near}} - \frac{1}{Z_{far}} \right) + \frac{1}{Z_{far}} \right] \quad (2-1)$$

其中， $z$ 表示物体与摄像头光学点的实际距离，焦距长用 $f$ 表示，像素深度值用 $d$ 表示， $Z_{near}$ 和 $Z_{far}$ 分别表示深度的最近点和最远点的Z轴方向距离。由于深度数据会被量化为 8 比特，因而BitDepth的值通常为 8，图 2-3 给出了通用的灰度值。

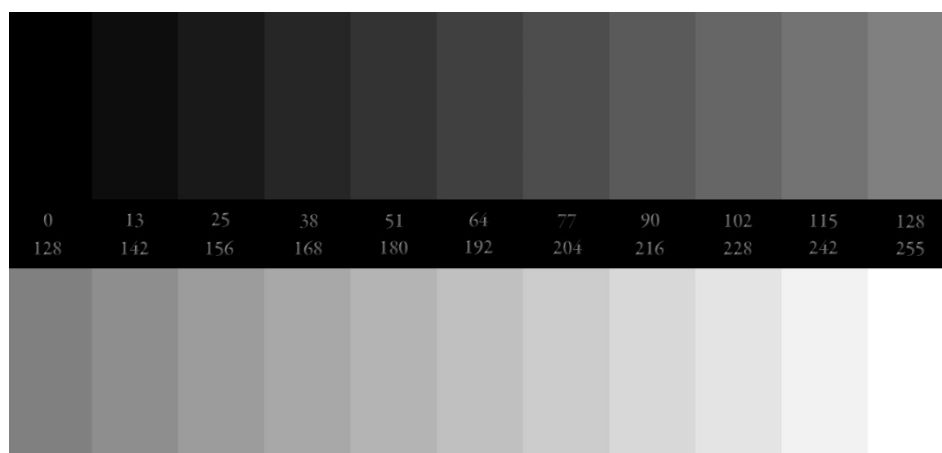


图 2-3 通用灰度值

需要注意的是，传输的基于深度的格式并不是终端的显示格式，深度信息并不会直接给人以立体的感受。3D 数据的显示需要在特定的立体显示设备上视觉化，从而让人感知到深度的信息，为此只需要提供给显示器解码端足够的视点纹理信息，深度的感知归功于双目的深度感知原理。

双目的深度感知基于两个重要因素：眼睛收敛度（视线角度）和双目的视差。对于人眼对于物体图像信息的捕捉，两只眼睛首先观测空间中的固定点，此时左右眼的视线会有不同的角度差。空间点在视网膜中生成的图像会因为人眼视觉系统的双目的混淆而拼接，物体在两只眼睛视网膜上位置的不同反映了物体的深度信息，因而我们的眼睛能够观察到不同物体的远近，如图 2-4。

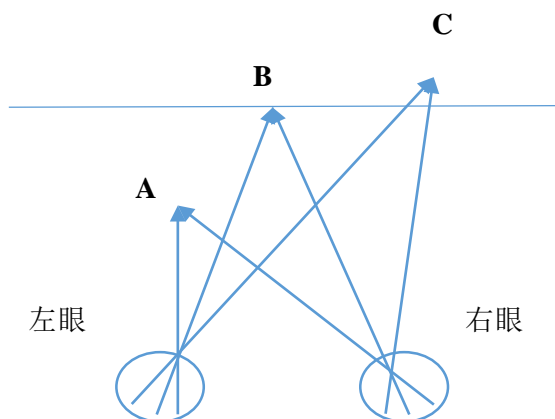


图 2-4 人眼视觉系统的双目深度感知

我们通常把整幅图像中距离摄像头更近的突出物体或运动物体称为前景，与之相应的静止物体就是背景。正是由于双目深度感知，我们才得以区分前景物体与背景物体，而这是单眼无法做到的。

## 2.3 坐标系变换

在计算机视觉领域中，3D 视频在摄像机坐标系（Camera Coordinate System）、图像坐标系（Image Coordinate System）和立体点所在的世界坐标系（World Coordinate System）变换是重要的变换模型。在多视点坐标系统中，任何点都有独一无二的三维世界坐标 $(x_w, y_w, z_w)$ ，独立于所有的摄像机。另一方面，每一个摄像机都有自己的摄像机坐标系和自己的图像坐标系。摄像机坐标系是一个三维的拥有 $X_c$ 、 $Y_c$ 和 $Z_c$ 三轴的坐标系。上文所提到的摄像机的光学中心会落在摄像机平面。而图像坐标系是二维的图像捕捉坐标系 $(u, v)$ 。图像坐标和摄像机坐标是平行的。主点（Principal Point）连接了光学轴和图像坐标系<sup>[16]</sup>，如图 2-5。

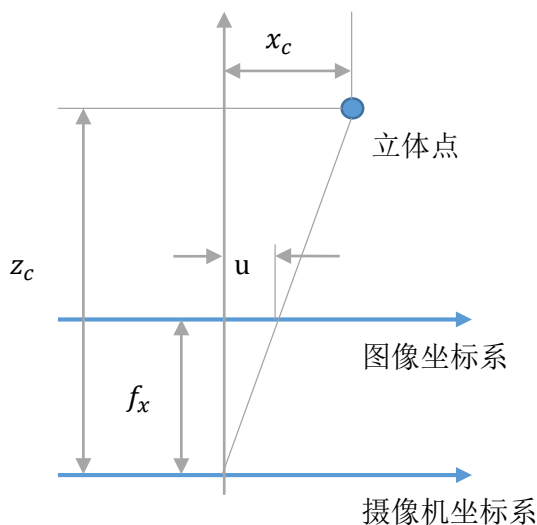


图 2-5 坐标系关系图

为了研究坐标系变换的关系，我们引入两个摄像头。同时，矩阵 $A$ 作为常量矩阵，被用于表示摄像机坐标系和对应图像坐标系的变化：

$$A = \begin{pmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{pmatrix} \quad (2-2)$$

其中 $f_x$ 和 $f_y$ 分别为 X 轴与 Y 轴方向上的焦距长度。主点（Principal Point）坐标偏移量为 $(o_x, o_y)$ 。由于在普遍的图像坐标系中，原点位于左上角，所以图像坐标系映射到摄像头坐标系需要一致的坐标偏移量。从图 2-3 相似性可知：

$$\frac{u}{x_c} = \frac{f_x}{z_c} \quad (2-3)$$

所以对于非零的 $o_x$ 值有：

$$u = \frac{f_x \cdot x_c}{z_c} + o_x \quad (2-4)$$

综上可得：

$$z_c \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = A \cdot \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = \begin{pmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} \quad (2-5)$$

类似于图像坐标系与摄像机坐标系的变化，世界坐标系与摄像机坐标系的变换系数矩阵为 $R$ （三行三列），此外还包括了转换矩阵 $t$ （三行一列），具体变换公式如下<sup>[17]</sup>：

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = R \cdot \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + t \quad (2-6)$$

通过式 2-5 和 2-6 得到了世界坐标系与摄像机坐标系、图像坐标系与摄像机坐标系之间的变换。

## 2.4 视频编解码

视频信号的编码通常使用基于已知帧的相关特征对未知的帧进行预测。帧内的预测往往基于的是几何像素区域的特征性，被认为是单一的图像修复。在通过比较当前帧和预测帧的差值时，多个区域的像素将会被选中<sup>[12]</sup>。然而，帧与帧之间的像素值往往不会呈现大范围的变化，那么基于像素中间值来处理几何区域将会是比较理想的方式。这样，大部分像素的值仅有微弱的变化，存储差值的数据量明显小于存储像素的完整信息（如 ARGB 编码标准），从而达到压缩编码的目的。同样地，对于信号的处理，解码过程通常是编码过程的逆过程，拥有类似的处理单元，解码的目的是在解码端恢复完整的视频信息。

基于深度的立体视频数据格式会生成用于产生立体效果的虚拟视点。根据上文叙述可知,通过深度图,中间视点可以利用最小复杂度的深度图像修复技术(DIBR)获得。这类格式的标准化过程正在进行中,尤其以 H.264 和 MPEG-4 AVC 的标准为代表(目前主流的视频格式)。相比原有的纯粹的多视点视频格式(MVV),视频数据格式逐渐整合了深度数据,所以寻找同时对多视点、包含深度数据的效率更高的压缩编码方式显得尤其重要<sup>[18]</sup>。

## 2.4.1 编码基本概念

### 1. 编码块

将整幅图像分割成若干子图像,这样的子图像称为编码块,简称块。常见的块包括 $16 \times 16$ ,  $16 \times 8$ ,  $8 \times 16$ ,  $8 \times 8$ ,  $8 \times 4$ ,  $4 \times 8$ ,  $4 \times 4$ 等。利用基于不同帧之间块的参考,相比基于像素有更高的效率。对于逻辑上合理的部分块还可以再次组合成宏块(Macro Block),以便于预测编码。

### 2. 像素精度

通常对于某个特定像素,该像素仅会有一个色彩值。然而在图像之间,即帧与帧之间的预测过程,往往会引入运动向量的概念,比如前景物体在前一个图片的左下角中移动到第二个图片的中间位置,可以预测推理在下一个图片中会位于右上角的位置。因此物体移动轨迹可以用向量表示,且这种向量的精度可以高于一个像素单位,所以预测物体可能落在非一个像素单位的位置,这时候就会引入像素精度的概念。当物体某些像素在一个图片的像素的非完整区域时,称为子像素,通常可以将子像素划分为 $1/2$ 、 $1/4$ 以及 $1/8$ 等精度,分别表示占有某个像素的 $1/2$ 、 $1/4$ 或 $1/8$ 。

例如,对于 $1/4$ 像素精度的情况,我们通常先使用6阶滤波器得到 $1/2$ 像素的值,再使用整像素和半像素的线性插值预测 $1/4$ 像素值<sup>[15]</sup>。

### 3. 参考帧与预测帧

参考帧称为I帧,能够记录该帧的完整地图像信息,也称为全帧。而预测帧称为P帧,按照一定的算法根据参考帧预测而得来的帧。通常是根据与前一帧或后一帧的图像的差值比较,而去掉与前一帧相似的数据构成的<sup>[19]</sup>。

### 4. 预测编码

#### (1) 帧内预测编码

统计表明,一幅图像的所有像素点会呈现出较强的相关性。相邻像素间发生突变的概念非常小。不仅对于相邻像素,相邻行之间的相关性也非常强,人们利用单幅图像内像素相关性特性进行预测,称为帧内预测编码,这也可以认为是纯粹的图像编码领域<sup>[20]</sup>。

#### (2) 帧间预测编码

相比帧内编码，帧间预测编码会有更高的效率，帧间编码利用了帧与帧之间的相关特性，显然对于视频而言会具有更高的效率。根据毕厚杰所著的《新一代视频压缩编码标准》[15]中写道：“有人测得，对缓慢变化 256 级灰度的黑白图像序列，帧间差超过阈值 3 的像素不到一帧像素的 4%；；对局列表话 256 亮度值的彩色电视序列，帧间差超过阈值 6 的像素平均只占一帧的 7.5%。”

### (3) 运动估计

运动估计是典型的用于帧间预测编码的一种估算手段。在一段连续的视频序列中，对于特定的某一段场景，帧与帧之间的物体在不考虑亮度值上，会呈现非常高的色彩相似程度。由此，我们可以将这一序列图像按照之前说的分块方式，按照物体的大小决定分块方式，并且搜索参考帧中各个块的物体位置，得到物体在这一段视频序列的位移向量，这一过程即称为运动估计<sup>[15]</sup>，如图 2-6，运动向量被表示为 $mv_i$ 。

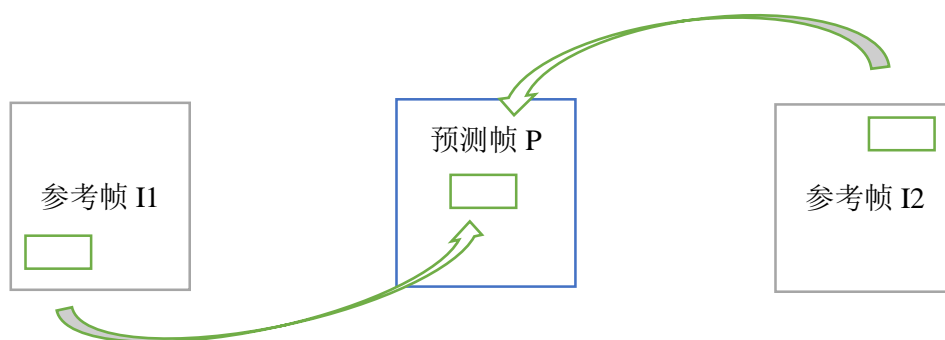


图 2-6 运动估计

## 2.4.2 视点合成预测

视点合成预测（View Synthesis Prediction, VSP）是连接纹理加深度的视频编码算法。通常，视点合成预测将图片从时间邻近的视点中通过扭曲到当前视点（3D 扭曲将在第三章中介绍），而扭曲的图片将作为参考图片来预测当前的图片。扭曲的图片也被称为合成图（Synthetic Picture），是用于作为编码或解码当前帧的完整图片。考虑到只有部分合成图片会用于预测，基于块的处理被用于减少时间复杂度。通过这种方式，只有用于预测的合成的像素会被生成<sup>[8]</sup>。

依赖于用于生成合成图片的视点的深度信息，视点合成预测（VSP）过程可以通过前向扭曲或者后向扭曲过程。在前向扭曲中（即扭曲到未来的帧），参考视点的深度图被用于从参考视点映射到未来视点（虚拟视点）。因此，映射的视点可能会落在当前视点的子像素位置，需要用整数位置的插值弥补。此外，前向扭曲在基于块的处理中表现一般，没能够直接定位用于预测块的参考区域<sup>[8]</sup>。

与前向扭曲相对应的是后向扭曲（利用当前帧扭曲到以往的帧）。当前视点的深度

图被用于作为参考视点提取像素值。当提取的像素值不是整数精度时，会被量化为子像素精度（如半像素精度、四分之一像素精度），然后通过同样地插值法进行运动补偿。通过这种方式，所有的整数像素精度将会被填补一定的值<sup>[8]</sup>。

通常，视点合成预测可以用两种方式实现。一种方式为高等级的修改，合成图片不仅会从参考图片中扭曲得到，还能继续作为新的参考图片来合成其他图片，这种方式可以用基于块的方式，且只需要最少的处理需求。另一种方式为低等级的修改，这种时候合成图片仅仅通过零位移向量得到，即对参考图片的部分复制。

视点合成预测（VSP）与帧间预测对多视点视频的处理上有很大的相似性。两者都引入了运动向量，区别于帧间预测，视点合成预测对物体的运动向量有不同的向量值。普通的帧间预测，运动向量纯粹用于编码的预测目的。而视点合成预测会针对所携有的深度图数据拥有不同的运动向量值，从而支持了更多的数据格式修复目的（即同时拥有对深度数据的优化目的）<sup>[8]</sup>。

### 2.4.3 基于深度的运动估计

基于深度的运动估计（Depth Based Motion Vector Prediction, DMVP）指结合深度信息的一种运动估计预测算法。深度信息可以看作是对运动向量的补偿，现在常用的有两种模式：跳跃模式（Skip Mode）和直接模式（Direct Mode）。深度运动向量的估计过程对不同组有不同的效果<sup>[8]</sup>。

在跳跃或直接模式下，运动信息不会被编码，而是在核心过程中推导出。例如，选择视点内的帧间预测，可以通过深度值找到相邻点的参考块，然后利用这一运动向量作为块的运动估计预测。

核心过程包括使用深度的最大前景深度值来预测视差，从而找到参考块，再使用参考块的运动向量作为当前块的预测向量，如图 2-7 所示流程图<sup>[8]</sup>。

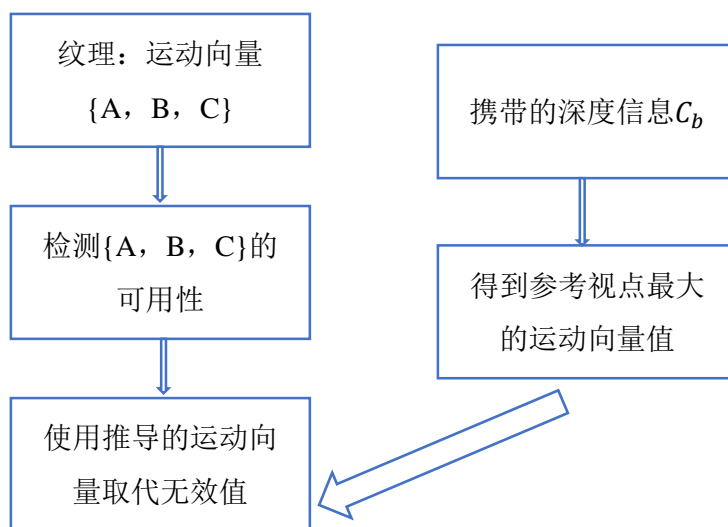


图 2-7 基于深度的运动估计

这里，需要引入绝对差的和（Sum of Absolute Differences, SAD）作为比较的评估标准，如式 2-7：

$$SAD(mv_i) = SAD(d(C_b, mv_i), d(C_b)) \quad (2-7)$$

运动向量 $mv_i$ 为特定方向提供了最小的SAD值：

$$mv_{pdir} = arg \min_{mv_{pdir}} (SAD(mv_i)) \quad (2-8)$$

短暂的方向预测表示为 $mv_{tmp}$ ，而帧间方向预测表示为 $mv_{pinter}$ 。预测值会通过跳跃模式提供最小的SAD值：

$$mv_{popt} = arg \min_{mv_{pdir}} (SAD(mv_{tmp}), SAD(mv_{pinter})) \quad (2-9)$$

从而获得优化后的运动向量值 $mv_{popt}$ <sup>[13]</sup>。

#### 2.4.4 自适应亮度补偿

自适应亮度补偿（Adaptive luminance compensation, ALC）是抑制编码宏块和预测块时光亮度变化的一种编码工具。自适应亮度补偿算法包含了：补偿模块，光亮度差值估计处理，求导运动向量和信号机制<sup>[8]</sup>。

大多数多视点原生视频序列和部分多视点合成视频序列通常表现为帧间的不一致，尤其是物体映射时光亮度不一致性，然而在纹理的几何结构上却能保持高度的一致性。抑制光亮度的变化能够增加块的帧间预测编码质量，同时减少冗余数据速率从而使得编码帧的峰值信噪比增加（PSNR）。

峰值信噪比（Peak Signal to Noise Ratio, PSNR）指的是峰值（最大值）功率与均方差的比值。由于比值的范围较大，同时近似满足人类的敏感程度，通常用对数形式来表示峰值信噪比的值。峰值信噪比常用于评估图像压缩后信号重构的质量，均方差的值用 $MSE$ 表示。例如，两个分辨率为 $x \times y$ 的单色图像 A 和 B，如果 B 为 A 的噪声近似图像，那么 A 与 B 之间的均方差定义为：

$$MSE = \frac{1}{xy} \sum_x \sum_y |A(i, j) - B(i, j)|^2 \quad (2-10)$$

其中，i与j分别为遍历x和y的因子， $MSE$ 表示均方差。

如果单色的 A 图像中，像素的最大值为 $MAX_A$ 。那么，峰值信噪比的表达式为：

$$PSNR = 10 \log_{10} \left( \frac{MAX_A^2}{MSE} \right) \quad (2-11)$$

单色图像中使用 8 比特表示一个像素点的值，那么 $MAX_A = 2^8 - 1 = 255$ 。

ALC 算法在编码中的核心是检测所有的帧间 P 帧宏块，从而通过编码区的失真率优化模式选择（Rate distortion optimized mode decision, RDO）来选择最佳模式。解码



的图片用于 ALC 的参数计算和纠正。1 比特的标志位会为每一个宏块标明是否使用 ALC。

在实际运用中，运动估计需要考虑光亮度变化的问题，因此需要调整 SAD，即分别减去亮度均值。而对于运动的补偿，要对当前块和参考块周边的已编码区域来计算 ALC 的权重。再利用 ALC 的权重对预测块进行校正，再进行补偿，并压缩 ALC 的标志位，如图 2-8。

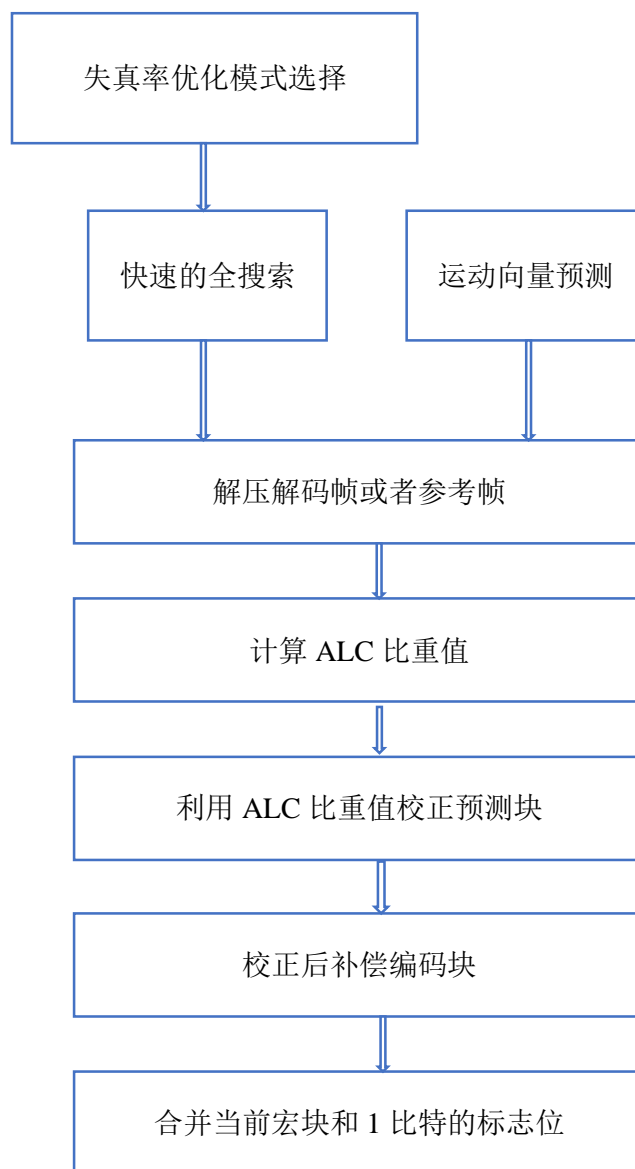


图 2-8 自适应亮度补偿编码模型

## 第 3 章 视点合成与图像修复

### 3.1 视点获取与合成

#### 3.1.1 3D 扭曲

正如第二章立体视频数据格式中介绍的，在 LDV 中，DIBR 技术用于利用边视点的深度数据合成中间视点（Intermediate View）。而视频合成中的核心问题是生成中间视点，也称为虚拟视点（Virtual View），即原本不存在的视点。对于已存在的视点我们称为参考视点（Reference View），因为摄像机的差异性，我们不仅仅需要参考视点的相机参数，同时也需要虚拟视点的相机参数以修正矩阵参数信息。

视点合成的第一步是通过参考视点携带的深度数据，将参考视点中的像素映射到虚拟视点中，这一步称为三维扭曲（3D warping）。

对于二维图像坐标系坐标 $(u, v)$ ，可以假设参考视点和虚拟视点的图像坐标系坐标分别为 $(u_r, v_r)$ 和 $(u_v, v_v)$ 。先将参考视点中的图像坐标映射到世界坐标系中，利用世界坐标系的唯一性，再依靠虚拟视点的相关参数从像素的世界坐标系变换到其图像坐标系中（下标 r 表示参考视点参数），联立式 3-1 与 3-2：

$$\begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} = R_r^{-1} \cdot \left( z_{c,r} \cdot A_r^{-1} \begin{pmatrix} u_r \\ v_r \\ 1 \end{pmatrix} - t_r \right) \quad (3-1)$$

其中 $z_{c,r}$ 表示通过 TOF 摄像机捕捉的深度距离（可以从式 2-1 中的比特值换算得到）。得到了世界坐标系的坐标值后，再次通过联立式 2-5 与 2-6 可以得到虚拟视点的图像坐标系的坐标值（下标 v 表示虚拟视点参数）<sup>[21]</sup>：

$$z_{c,r} \cdot \begin{pmatrix} u_v \\ v_v \\ 1 \end{pmatrix} = A_v \cdot \left( R_v \cdot \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + t_v \right) \quad (3-2)$$

#### 3.1.2 空洞与堵塞

在视点合成的过程中，多个参考视点会同时映射到同一个虚拟视点。然而虚拟视点的每个像素并不是都会被映射到，也同样会有一些像素点会被多个参考视点同时映射。前者出现的情况我们称之为空洞（Hole），后者出现的情况会产生映射的竞争现象，称为堵塞（Occlusion）。空洞的出现在图像中的反映是部分区域丢失信息，而呈现出一个缺失区域。

对映射的竞争，即堵塞，的解决方法是增加一个缓冲器。对于虚拟视点中出现同一像素多点映射时，缓冲器会根据深度值所表现的像素的近似度，比较各个映射过来的像素点的优劣，从而进行最优选择。通常在此情况下，前景的像素相比背景像素更能在竞争中取胜。

空洞的出现解决方法是，利用邻近的背景像素作为已知信息，结合图像修复的相关算法知识填补空洞，这一部分的算法会在图像修复一节中介绍基于 Criminisi 工作成果的算法简要。同时，缓冲器也能同样服务于空洞的弥补，通过存储的信息选择深处图像（即背景），用来填补，大部分实验显示背景信息的填补效果是相对更理想<sup>[11]</sup>，如图 3-1，左右各为两个参考视点映射而得的虚拟视点。



图 3-1 映射过程空洞的出现

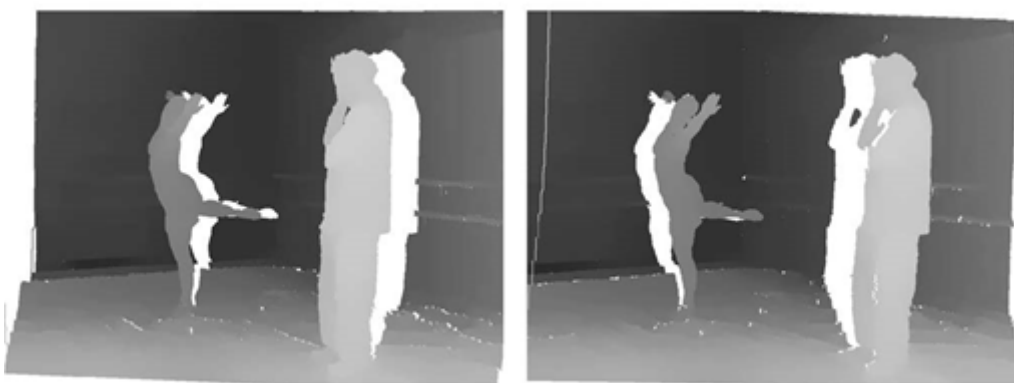


图 3-2 对应的深度图信息

### 3.1.3 视点的合成

由于虚拟视点通常位于已知参考视点的中间，可以认为空洞的填补是一种插值，插值参数为参考视点相关像素信息。从图中可以观察到在左右两个参考视点的 3D 扭曲中，左视点出现的空洞位置在右边而右视点出现的空洞位置在左边。据此，虚拟视点合成通常表示为线性比重：

$$V_i = \frac{|t_{x,l-v}|R_i + |t_{x,r-v}|L_i}{|t_{x,l-v}| + |t_{x,r-v}|} \quad (3-3)$$

这里引入基线（Baseline）的概念。在典型的从左右两个参考视点合成中间的虚拟

视点中，基线指的分别是左右两个参考视点摄像头与中间虚拟视点摄像头的距离。就式 3-3 而言， $V_i$ 、 $R_i$ 和 $L_i$ 分别指的是第  $i$  个像素（对于固定分辨率的视频而言，像素数为分辨率的乘积），以此遍历所有像素。 $|t_{x,l-v}|$ 与 $|t_{x,r-v}|$ 分别指左视点与虚拟视点间的基线距离和右视点与虚拟视点间的基线距离。从而距离虚拟视点较近的参考视点获得更多的比重值。

视点合成中，如果虚拟视点的一个像素只能从一个参考视点中获得，那么比重的算法将会无效，而此时的合成效果也会很差<sup>[21]</sup>。

### 3.1.4 质量增强

由于以上三步的视点合成并没有考虑到突发的失真情况，为此，引入质量增强技术以消除视觉上的失真。合成过程中的严重失真情况主要由两个因素引起：深度值错误和视点合成的参考视点限制。深度值错误如果超出了预设阈值，将会扭曲纹理的特征。正如前文介绍的，缓冲器将会在竞争情况中选择深度最近的参考视点的像素，可想而知，深度值的错误会造成选择的错误，从而造成失真。

使用深度图的预滤波是一个有效的措施。根据 Wan-Yu Chen 的工作成果，边缘依赖性的深度图滤波与插值能够有效解决 DIBR 技术中的空洞填补问题。同时在工作结果发现，峰值信噪比（Peak Signal to Noise Ratio, PSNR）能够相比提升 6dB 或以上<sup>[22]</sup>，如图 3-3。

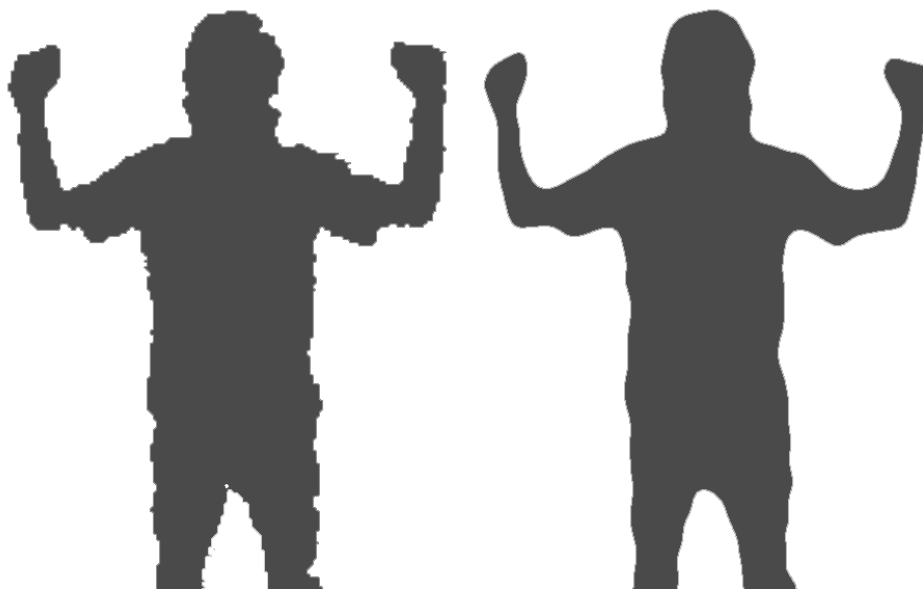


图 3-3 深度图滤波

视点合成的参考视点限制会导致不自然的合成纹理。首先，对一个参考视点来说是不会包含需要扭曲的虚拟视点的全部信息，并且信息量决定于两个视点之间的距离

远近。而对于多个参考视点，插值是相对容易的，所以参考视点的多少很大程度决定了失真程度。高级的空洞填补技术被提出，根据 Daribo 的工作成果，对于大片的堵塞与空洞问题，深度图滤波仍然有限制，而纹理和结构性的蔓延处理能够有效通过区分场景中的前景与后景来估计深度信息，实验的结构也表明了这种方法的有效性<sup>[13]</sup>。其次，对于单物体平面的物体只会关联一个深度值，而有些像素可能会携带多个物体的深度值，如半透明的玻璃杯，它的颜色会是玻璃和背景的混合。这种情况下，视点合成就难以得到令人满意的结果<sup>[22]</sup>。

### 3.2 冗余层数据

比较中心参考视点和从中心参考视点 3D 扭曲(3D warping)得到的边视点会发现，图片中存在相同的部分，同时也会出现空洞，而这些空洞可以从不同的虚拟视点中比较得出，并且集中在前景物体的不连续区域。边视点信息因此被减少（合成视点减去参考视点），称为冗余层，如图 3-4 和图 3-5。这一做法相比传输参考视点和多个合成视点（MVD）来说，可以大大减少数据比特率。

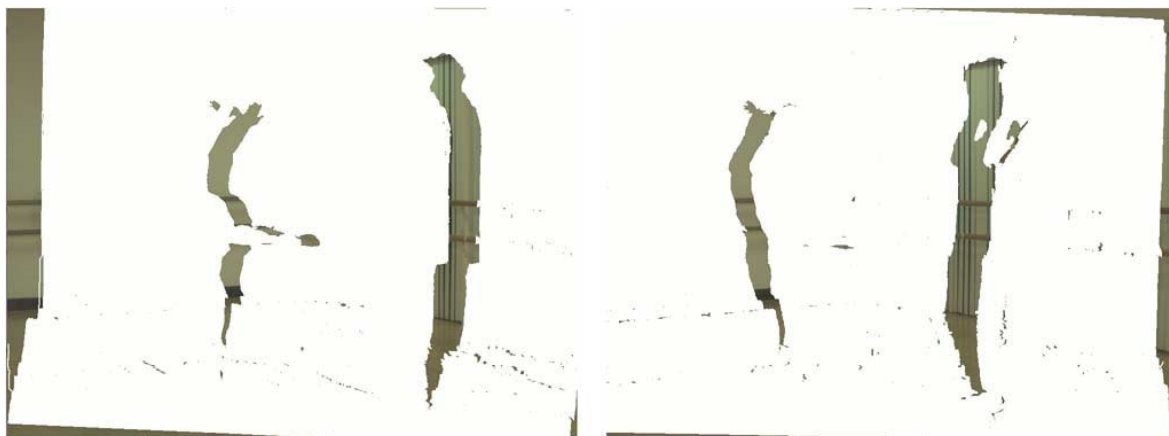


图 3-4 纹理图的冗余层数据

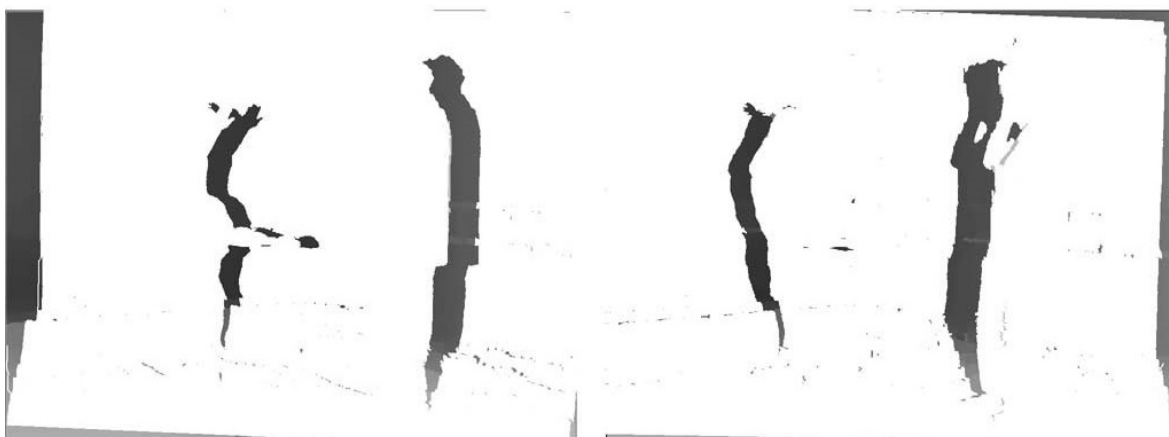


图 3-5 深度图的冗余层数据

在解码端，中心视点和冗余层数据会被解压缩，用以重构边视点，从而在减少压缩的同时同样能够保持高质量的用户体验<sup>[3]</sup>。

类似于视点合成过程中的缓冲器，冗余层数据生成过程也可以引入缓冲器，对于同一物体在不同视点上体现的不同深度值，可能会被认为是不同景深的物体。这时，可以用缓冲器生成噪声背景层并携有冗余和错误的前景信息，如图 3-6。但是这样的做法同样也会使压缩率降低。



图 3-6 去除携有部分背景信息的冗余层数据

被提出的更好的方法是检测帧间深度的不一致性。首先从中心视点 3D 扭曲，得到合成视点（边视点），能够发现合成视点中前景物体的空洞。之后，合成视点能够反向合成到中心视点，发现中心视点缺少的区域，从而避免在背景层中包含不必要的前景信息。除此之外，合成视点中的空洞可以根据附近的结构性进行重组。增强冗余层的表示方式，即减少冗余数据，还包括利用空间相关性（Spatial Correlation），通过将冗余层数据减去图像修复为填补空洞生成的图层，进一步减少冗余层数据量。

### 3.3 图像修复

根据 Criminisi 的研究成果<sup>[3]</sup>，输出图像的合成会被修复过程的顺序影响。对于输入的图像  $I$  和缺失区域  $Q$ ，那么有效的区域  $E$  被表示为  $E = I - Q$ 。Criminisi 的算法通过最优填补的方法，利用缺失区域的边界信息来填补。对于给定一个用于填补的图像  $F$  位于需要修复的店附近， $F$  位于边界可选区域内，那么该图像的优先度  $P(p)$  可以被两个参数决定： $C(p)$  和  $D(p)$ <sup>[23]</sup>。

$C(p)$  为当前用于修复图像的可信度值，而  $D(p)$  特指缺失区域的方向优先度。对于每一个缺失的像素均采用这两者进行评估，可以得到最优的选择方案，进而填补整个缺失区域。这对单帧内的图像修复有着重要意义。

在结合深度图的视频中，通过关联对应的深度值可以改善优先性的选择方案，即增加深度值  $Z$  对优先度的影响  $Z(p)$ 。从而根据  $C(p)$ 、 $D(p)$  和  $Z(p)$  决定缺失像素用来填补的像素。

## 第 4 章 实验数据与结果

### 4.1 实验概要

#### 4.1.1 实验目的

本实验使用的是 3DV-ATM 模型，对微软官方提供的“kendo”等测试序列进行实验，比较 VSP、DMVP、ALC 三种预测算法及 QP 的不同纹理与深度值对包含深度图的 3D 视频序列的压缩效率。

#### 4.1.2 3DV-ATM 模型

3DV-ATM 编码可以配置为两种编码模式，称为“MVC+D”和“3D-AVC”，同时也遵循两者各自的特性。“MVC+D”的配置提供了 H.264/MVC 对纹理视图的兼容性并实现了纹理和深度图数据可以独立编码如单一的编码比特流之中。和 H.264/MVC 特性不同的是，“MVC+D”被限制在一个高等级的特征，同样在解码端也有这样的特性。就交错编码而言，不同的帧和域（Field）的编码发生在纹理和深度之间。“3D-AVC”也配置提供了 H.264/AVC 多的兼容性，并且实现了连接纹理和深度图数据，从而在一个比特流之中进行编码。在这种配置下，纹理数据被用于关联自身的深度图数据来进行编码。另一方面，作为增强性视点的深度图数据被用于编码关联的纹理视点。高等级的 3DV-ATM 编码流程图如图 4-1 所示，相似的解码流程图如图 4-2 所示。其中实线表示数据流动方向，而开关上的虚线表示控制信号。虚线 A 使用了高级纹理编码的深度信息，而虚线 B 使用了高级深度编码的纹理信息。除此之外，3DV-ATM 包含了预处理工具（非线性深度表示，Non-linear Depth Representation, NDR）和后处理工具（后处理扩张滤波，Post-processing Dilatational Filtering, PDF），整合在了当前的 3DV 系统设计之中但是不包括“MVC+D”和“3D-AVC”的特性文本之中<sup>[20]</sup>。

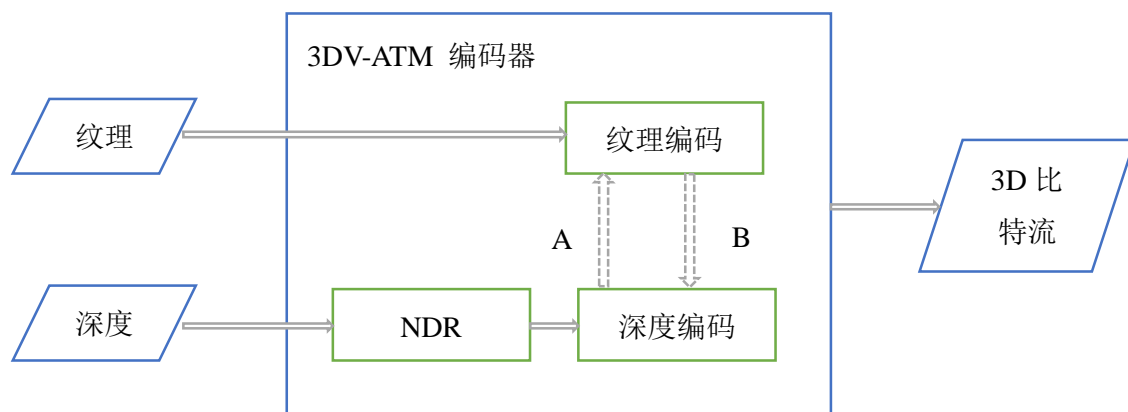


图 4-1 3DV-ATM 编码模型



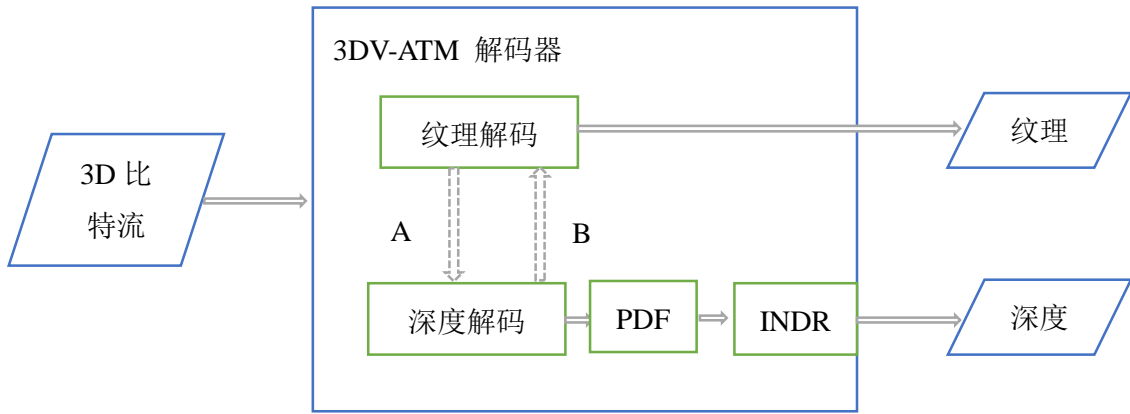


图 4-2 3DV-ATM 编码模型

### 4.1.3 量化参数

本次实验中使用到的参数除了在第二章中介绍的视点合成预测（VSP）、基于深度的运动估计（DMVP）、自适应亮度补偿（ALC）以外，还包括量化参数（Quantization Parameter）。对于以上四个参数，前三者均可控制使能信号，后者可以直接控制参数数值（0-51）。

量化的效果被量化参数 QP 控制，而 QP 事实上只是一个索引值（Index），不同的索引值会指向一个不同的标量矩阵。我们通常使用量化步长（Quantization Step）来推导 QP 的每一个值，随着 QP 的增长，量化步长也会增加。典型地，量化步长每增加一倍时，QP 的值约摸增加 6，属于一种对数性增长方式，即  $20 \log_{10} 2$ ，具体关系如式 4-1。

$$QP = 20 \log_{10} Q_{step} \quad (4-1)$$

式中，量化步长用  $Q_{step}$  表示。

## 4.2 实验方案

### 4.2.1 视频配置参数修改

深度图配置文件中需要修改的参数如表 4-1 所示。本次实验仅仅考虑 I 帧（参考帧）与 P 帧（预测帧）的 QP 参数。

表 4-1 纹理图配置文件部分参数

参数编号	参数属性	参数说明原文
1	LevelIDC	Quant. param for I Slices (0-51)
2	QPPSlice	Quant. param for P Slices (0-51)

类似地，纹理图配置文件中需要修改的参数如表 4-2 所示。



表 4-2 纹理图配置文件部分参数

参数编号	参数名称	参数说明原文
1	FramesToBeEncoded	Number of frames to be coded
2	QPISlice	Quant. param for I Slices (0-51)
3	QPPSlice	Quant. param for P Slices (0-51)
4	DepthBasedMVP	Enable depth-based motion vector prediction for texture(Default:0)
5	AdaptiveLuminanceCompensation	Enable adaptive luminance compensation
6	VSP_Enable	Enabling VSP in dependent texture views

注：对使能信号，“1”表示打开，“0”表示关闭，实验中的算法比较部分对 I 帧和 P 帧的 QP 值相等。

#### 4.2.2 实验方案

实验所采取的方案主要为保持其他参数不变的基础上，更改纹理配置文件和相应深度配置文件的相同参数（三种编码预测算法只在纹理配置文件中修改）。计算编码后的深度加纹理视频比特速率（体现数据量的大小），并通过 PSNR 计算软件计算合成后的 PSNR 值。为了使得到的比特速率和 PSNR 值更准确，所有的实验均使用 100 帧的编码帧数。在测试 QP 的实验中，测试深度图 QP 值的变化对 PSNR 均值以及比特率的影响。而在算法测试实验中，研究的是算法对比特率和 PSNR 值得影响。表 4-3 整理了所需的所有实验组的参数配置，这里用 VSP、DMVP 和 ALC 简称三种预测算法，QP 的取值在 26-40 之间以使实验结果更有合理性，实验编号如“1~11”表示第 1 到第 11 组。

表 4-3 实验参数表

实验编号	编码帧数	DMVP	ALC	VSP	QP（纹理）	QP（深度）
1~11	100	Enable	Enable	Enable	26	28, 29, 30...38
12~22	100	Enable	Enable	Enable	28	28, 29, 30...38
23~33	100	Enable	Enable	Enable	30	28, 29, 30...38
34~44	100	Enable	Enable	Enable	32	28, 29, 30...38
45~55	100	Enable	Enable	Enable	34	28, 29, 30...38

续表

56~66	100	Enable	Enable	Enable	36	28, 29, 30...38
67~77	100	Enable	Enable	Enable	38	28, 29, 30...38
78~88	100	Enable	Enable	Enable	40	28, 29, 30...38
89~96	100	Disable	Enable	Disable	26, 28, 30...40	26, 28, 30...40
97~104	100	Enable	Enable	Disable	26, 28, 30...40	26, 28, 30...40
105~106	100	Enable	Enable	Enable	26, 40	26, 40
107~114	100	Enable	Disable	Enable	26, 28, 30...40	26, 28, 30...40

注：配置文件中算法开关用“1”和“0”表示，实验数据表格用“Enable”和“Disable”以增强可读性。

这里给出原始数据的完整变量名称与含义，如表 4-4：

表 4-4 原始数据变量释义

变量名称	变量描述
Sequence	指测试序列名（本次实验暂时只对“kendo”序列做了测试）
DepthBaseMVP	是否开启 DMVP 算法，包含“Enable”与“Disable”
AdaptiveLuminanceCompensation	是否开启 ALC 算法，包含“Enable”与“Disable”
VSP_Enable	是否开启 VSP 算法，包含“Enable”与“Disable”
Texture_QPISlice	纹理图中 I 帧 QP 参数值
Texture_QPPSlice	纹理图中 P 帧 QP 参数值
FrameToBeEncoded	编码的帧数
Depth_QPISlice	深度图中 I 帧 QP 参数值
Depth_QPPSlice	深度图中 P 帧 QP 参数值
T1_Rate	记录的纹理图视点 1 比特率
T5_Rate	记录的纹理图视点 5 比特率
D1_Rate	记录的深度图视点 1 比特率
D5_Rate	记录的深度图视点 5 比特率
Ratio	纹理图与深度图 I 帧 QP 参数值的比值
T1_PSNR	纹理图视点 1 的 PSNR 值
T5_PSNR	纹理图视点 5 的 PSNR 值
AVE_PSNR	纹理图视点 1 和 5 的 PSNR 均值
SUM_Rate	比特率总和（纹理图视点 1 + 纹理图视点 5 + 深度图视点 1 + 深度图视点 5）

## 4.3 实验数据分析

### 4.3.1 数据分析的意义

为测试实验数据，本人提出基于 R 语言，通过 RStudio 平台来对大量的数据进行整理和分析。通过编程的方式进行数据整理与分析能够大大减少重复性工作，可以将各类数据按照需求进行筛选和整理。此外，本人也将源代码公开（<https://github.com/willowchan/DataAnalysis>），可以针对数据进行个性化的修改和增减，供算法实验结果的数据分析使用。

### 4.3.2 R 语言与 RStudio

#### 1. R 语言

R 语言主要用于数据的整理和分析，同时也在绘图方面有着优异的表现。R 语言来源于上世纪 80 年代的 S 语言，然而 R 语言具有自由和免费特征，因而在数据分析工作者中非常流行。在本人的工作中，使用的 R 语言版本为 3.1.0。

R 语言的源代码可以在 Github 上免费下载，也用编译过的可执行文件，其中以 CRAN 基础包为代表，本人也是在 CRAN 基础包中使用的（官方 CRAN 基础包地址为：<http://cran.r-project.org/>）。R 语言支持多个操作系统，包括 UNIX（也包括 FreeBSD 和 Linux）、Windows 和 Mac OS（下载与安装请见附录 1 关于 R 语言的下载和安装部分）。R 语言使用的是类似 Matlab 的命令行形式，也可以写成大段脚本（Script）形式，有人为此开发了多种图形化界面，其中论文中使用的就是其中一种——RStudio。

R 语言内置了多种统计学及数字分析功能的分析包。通过安装相关拓展包的形式，R 语言因而能够拓展其功能。

#### 2. RStudio

RStudio 是为 R 语言设计开发的一种集成式环境，基于 C++ 制作完成，同样也拥有着免费特性。RStudio 同时有桌面客户端和服务端。论文中使用的是个人桌面客户端（下载与安装详情请见附录 1 关于 RStudio 的下载和安装部分）。

如图 4-3 所示，左上角为脚本文件（运行脚本可选择脚本开始位置，点选 **Run** 按钮逐条执行命令）；左下角为控制台，显示系统提示与输出结果等；右上角为环境中存储的变量信息（可左键点击快速预览变量内容）；右下角继承了包括文件路径（File）、绘图区（Plots）以及帮助信息（Help）等。

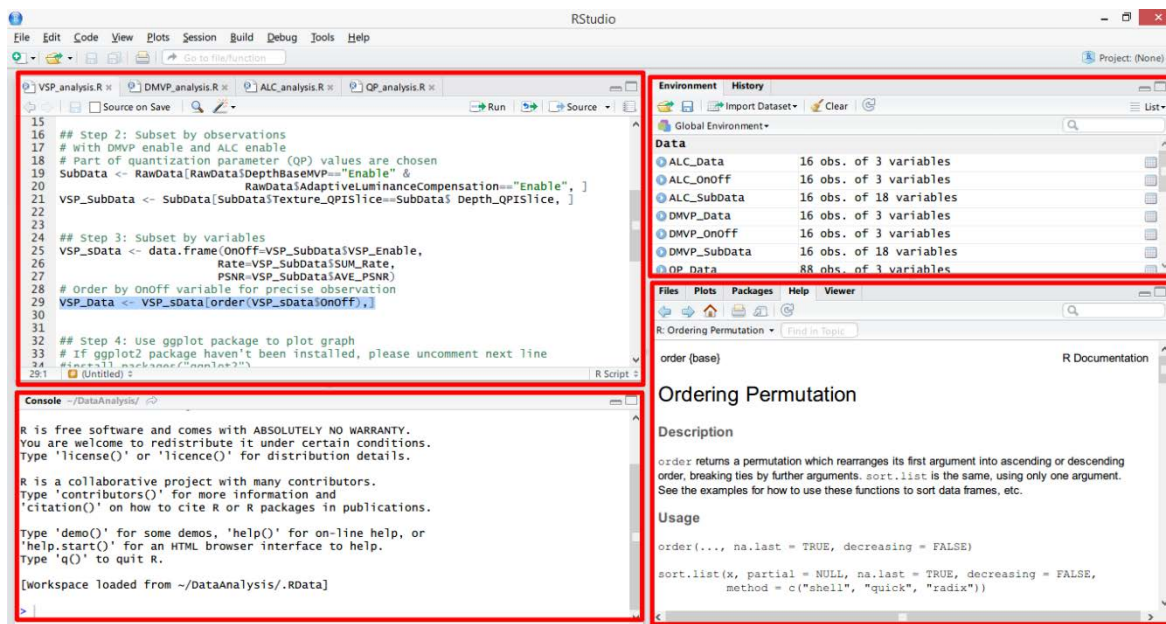


图 4-3 RStudio 界面概览

### 3. Git Bash

参考附录 1 中 Git Bash 的下载与安装可以直接将 Github 源中脚本和数据拷贝至本地。另一个可选的较为简单的方法是使用 Github 桌面客户端直接选择克隆源，这里不做过多的介绍（需要先使用 Fork 按钮添加源到自己的源中）。

#### 4.3.3 观测值数据整理

（部分代码被列举，完整脚本代码见附录 2）

观测值指的是表格中对应的一行数据，根据实验需要测试的组别分类进行。

##### 1. 量化参数测试组

量化参数的测试占了实验序列的很大一部分，为实验的第 1 到第 88 组。根据参数 DMVP、ALC 和 VSP 皆为 Enable 来筛选相应观测值，由于 QP（纹理）与 QP（深度）皆为 26 和 40 的观测值不作为量化参数测试实验部分，故不考虑。核心代码如下：

```
SubData <- RawData[RawData$DepthBaseMVP== "Enable" &
                  RawData$AdaptiveLuminanceCompensation== "Enable"&
                  RawData$VSP_Enable== "Enable", ]
QP_SubData <- SubData[SubData$Depth_QPISlice!=26 &
                     SubData$Depth_QPISlice!=40, ]
```

## 2. 基于深度的运动估计测试组

这一部分仅需要利用 VSP 为“Disable”的观测值，核心代码如下：

```
DMVP_SubData <- RawData[RawData$VSP_Enable == "Disable", ]
```

## 3. 自适应亮度补偿测试组

需要 QP（纹理）与 QP（深度）值相等的组，且 DMVP 与 VSP 均为“Enable”，为增强代码可读性，分两步完成（纹理和深度的 I 帧与 P 帧的 QP 值始终相等）：

```
SubData <- RawData[RawData$VSP_Enable == "Enable" &
  RawData$DepthBaseMVP == "Enable", ]
ALC_SubData <- SubData[
  SubData$Texture_QPISlice == SubData$Depth_QPISlice, ]
```

## 4. 视点合成预测测试组

类似视点合成预测测试组，不同的是需要 DMVP 与 ALC 均为“Enable”，代码如下（同样分两步进行）：

```
SubData <- RawData[RawData$DepthBaseMVP == "Enable" &
  RawData$AdaptiveLuminanceCompensation == "Enable", ]
VSP_SubData <- SubData[
  SubData$Texture_QPISlice == SubData$Depth_QPISlice, ]
```

### 4.3.4 变量提取

（部分代码被列举，详细代码见附录 2）变量提取的目的是为了简化数据表格，对于更大的数据表而言能够有效减少占用的内存空间，从数据的表头信息来看，能给数据分析者一个更清晰可见的数据表格。在 R 语言中，比较使用 `head(RawData, 5)` 和 `head(QP_SubData, 5)` 可以明显发现后者在控制台输出内容可读性更高。使用 `str(QP_SubData)` 同样会有更清晰的数据信息列表（这里以 QP\_Data 为例）。

#### 1. 量化参数测试组

```
QP_Data <- data.frame(PSNR=QP_SubData$AVE_PSNR,
  Rate=QP_SubData$Rate,
  QP_texture=QP_SubData$Texture_QPISlice)
```

使用 data.frame 构造数据表，构造的数据表变量名分别为：PSNR 和 Ratio，再将对应的 SubData 中变量的列向量提取出来，将结果的数据表命名为 QP\_Data。

#### 2. 基于深度的运动估计测试组

```
DMVP_Data <- data.frame(OnOff=DMVP_SubData$DepthBaseMVP,
  Rate=DMVP_SubData$SUM_Rate,
  PSNR=DMVP_SubData$AVE_PSNR)
```

*OnOff* 指是否开关算法（Enable 或 Disable），自适应亮度变化测试组与视点合成预测测试组类似。

### 3. 自适应亮度补偿测试组

```
ALC_Data <- data.frame(OnOff=ALC_SubData$AdaptiveLuminanceCompensation,  
                        Rate=ALC_SubData$SUM_Rate,  
                        PSNR=ALC_SubData$AVE_PSNR)
```

### 4. 视点合成预测测试组

```
VSP_sData <- data.frame(OnOff=VSP_SubData$VSP_Enable,  
                        Rate=VSP_SubData$SUM_Rate,  
                        PSNR=VSP_SubData$AVE_PSNR)
```

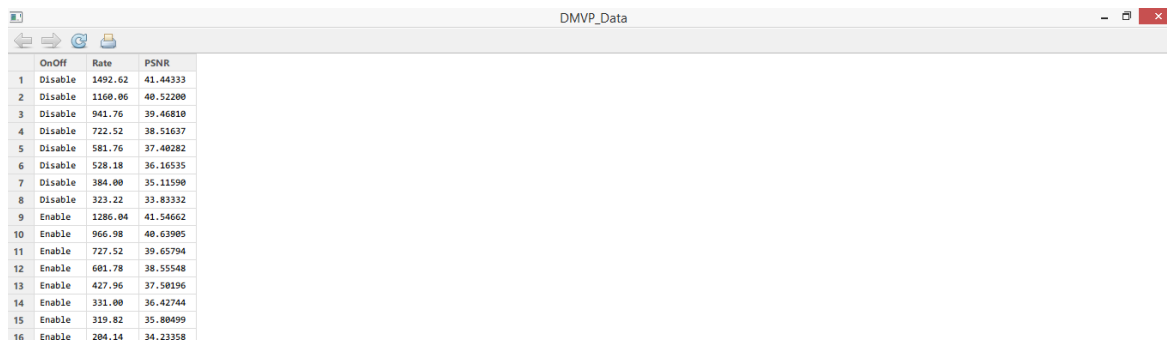
需要指出的是，上述编码整理完成的数据排序上是单纯根据行数来排序的，如果需要额外根据区分变量进行排序，可使用如下代码（此处以 VSP 为例，其他测试组方法类似）：

```
VSP_Data <- VSP_sData[order(VSP_sData$OnOff),]
```

这里给出整理前后的数据结果（以 DMVP 测试组为例），如图 4-4 和 4-5。

Sequence	DepthBaseMVP	AdaptiveLuminanceCompensation	VSP_Enable	Texture_QPISlice	Texture_QPPISlice	FrameToBeEncoded	Depth_QPISlice	Depth_QPPISlice	T1_Rate	TS_Rate	D1_Rate	D5_Rate	Ratio	T
1	kendo	Enable	Enable	26	26	20	28	28	405.58	467.09	143.99	199.48	2.5407459	4
2	kendo	Enable	Enable	26	26	20	29	29	413.72	476.52	126.29	176.86	2.9366320	4
3	kendo	Enable	Enable	26	26	20	30	30	412.95	474.13	110.48	154.22	3.3512656	4
4	kendo	Enable	Enable	26	26	20	31	31	404.30	466.15	98.31	136.96	3.6997917	4
5	kendo	Enable	Enable	26	26	20	32	32	414.82	475.90	85.47	115.77	4.4221825	4
6	kendo	Enable	Enable	26	26	20	33	33	413.42	474.70	76.48	104.95	4.8951111	4
7	kendo	Enable	Enable	26	26	20	34	34	403.71	467.64	65.15	91.96	5.5461142	4
8	kendo	Enable	Enable	26	26	20	35	35	411.87	475.98	57.28	78.68	6.5302295	4
9	kendo	Enable	Enable	26	26	20	36	36	412.31	474.97	48.99	68.82	7.5314489	4
10	kendo	Enable	Enable	26	26	20	37	37	405.54	468.98	43.27	60.13	8.4576402	4
11	kendo	Enable	Enable	26	26	20	38	38	414.96	475.41	36.95	50.99	10.1247441	4
12	kendo	Enable	Enable	28	28	20	28	28	281.94	326.30	136.26	197.94	1.8199880	4
13	kendo	Enable	Enable	28	28	20	29	29	304.21	344.59	126.89	176.21	2.1405477	4
14	kendo	Enable	Enable	28	28	20	30	30	294.67	337.07	110.01	153.77	2.3949503	4
15	kendo	Enable	Enable	28	28	20	31	31	301.43	342.56	97.24	137.83	2.7395669	4
16	kendo	Enable	Enable	28	28	20	32	32	301.93	343.08	85.43	116.42	3.1954917	4
17	kendo	Enable	Enable	28	28	20	33	33	294.25	335.86	76.85	104.31	3.4781961	4
18	kendo	Enable	Enable	28	28	20	34	34	302.65	345.23	65.06	92.42	4.1140462	4
19	kendo	Enable	Enable	28	28	20	35	35	301.20	344.16	56.94	78.89	4.7512332	4
20	kendo	Enable	Enable	28	28	20	36	36	295.21	338.12	49.30	68.81	5.3622047	4
21	kendo	Enable	Enable	28	28	20	37	37	303.11	345.27	43.39	59.75	6.2864068	4
22	kendo	Enable	Enable	28	28	20	38	38	304.38	345.74	37.40	50.97	7.3567953	4
23	kendo	Enable	Enable	30	30	20	28	28	223.98	253.11	145.00	201.63	1.3763667	3
24	kendo	Enable	Enable	30	30	20	29	29	228.87	259.38	126.21	175.82	1.6165613	3
25	kendo	Enable	Enable	30	30	20	30	30	212.68	245.44	103.56	150.98	1.8003615	3
26	kendo	Enable	Enable	30	30	20	31	31	230.30	258.71	98.89	136.42	2.0781522	3
27	kendo	Enable	Enable	30	30	20	32	32	223.06	252.39	84.34	116.76	2.3642466	3
28	kendo	Enable	Enable	30	30	20	33	33	230.41	256.65	76.41	104.66	2.6809080	3
29	kendo	Enable	Enable	30	30	20	34	34	230.48	256.62	64.80	91.17	3.1230365	3
30	kendo	Enable	Enable	30	30	20	35	35	223.27	254.52	56.88	78.42	3.5313378	3
31	kendo	Enable	Enable	30	30	20	36	36	229.36	257.76	49.11	68.14	4.1545416	3
32	kendo	Enable	Enable	30	30	20	37	37	229.93	258.59	43.28	59.83	4.7378528	3

图 4-4 原始数据



	OnOff	Rate	PSNR
1	Disable	1492.62	41.44333
2	Disable	1160.06	40.52200
3	Disable	941.76	39.46810
4	Disable	722.52	38.51637
5	Disable	581.76	37.40282
6	Disable	528.18	36.16535
7	Disable	384.00	35.11590
8	Disable	323.22	33.83332
9	Enable	1286.04	41.54662
10	Enable	966.98	40.63985
11	Enable	727.52	39.65794
12	Enable	601.78	38.55548
13	Enable	427.96	37.50196
14	Enable	331.00	36.42744
15	Enable	319.82	35.80499
16	Enable	204.14	34.23358

图 4-5 量化参数整理结果

### 4.3.5 数值获取与绘图

（完整脚本代码请见附录 2）对于数据的分析，绘图形式是作为最直观的描述表达。在 R 语言中，`ggplot2` 包是一种针对图形语法的绘图包，相比 `Base plot` 以及 `Lattice plot` 等有着简洁、直观且兼具美观的特点。首次读入需要先加载 `ggplot2` 包，使用命令：

```
Install.packages("ggplot2")
```

这里以量化参数为例，其他组别方法类似。

```
QP_value <- aggregate(PSNR ~ Rate + QP_texture, data = QP_Data, FUN = sum)
with(QP_value, qplot(Rate, PSNR, col = QP_texture,
                     geom = "point",
                     xlabs = "Bit Rate (Kbps)",
                     ylabs = "PSNR (dB)",
                     main = "QUANTIZATION PARAMETER"))
```

`QP_value` 存储了所需的有关所有信息值（通过 `aggregate` 函数，`FUN=sum` 代表总和，即整合所需所有数据），提取了 `PSNR`、`Rate` 和 `QP_texture` 三个变量的值，其中 `QP_texture` 用来作为区分变量。再通过 `with` 函数绘制，绘制方法是 `ggplot2` 包中所含有的 `qplot` 绘图函数（所有有关函数用法均可用 `?*` 在 RStudio 控制台中查看，如 `?qplot` 可以查看 `qplot` 的用法）。在 `qplot` 函数中，`Rate` 作为横坐标，`PSNR` 作为纵坐标，以 `QP_texture` 为区分变量（颜色差异），绘图方式为点（`point`）绘图，`xlabs`、`ylabs` 和 `main` 定义了横纵坐标以及标题名。

对于已在屏幕上显示的图形如果需要保存到本地硬盘中，可以键入如下命令（仍然以量化参数为例）：

```
dev.copy(png, file="QP_analysis.png")
dev.off()
```

`dev.copy` 表示从设备（这里设备指屏幕）中拷贝，保存的文件类型为 `png`，文件名为 `QP_analysis.png`。最后关闭设备，使用 `dev.off`。

作为可选择的数据格式，原始和整理后的数据均输出为 `txt` 格式，代码如下：

```
write.table(RawData, file="/RawData.txt")
```

```
write.table(QP_Data, file="/QP_Data.txt")
```

参照附录 1 中平台搭建方法并将数据文件放入工作目录中，运行脚本即可获得相应的输出文件。

对于统计数据的分析，如 `PSNR` 的差值均值、比特率变化率的均值等，以量化参数组为例，可以通过如下代码获取：

```
PSNR_Statistic <- c(
  mean(QP_SubData[QP_SubData$Texture_QPISlice==26,]$AVE_PSNR),
  mean(QP_SubData[QP_SubData$Texture_QPISlice==28,]$AVE_PSNR),
  mean(QP_SubData[QP_SubData$Texture_QPISlice==30,]$AVE_PSNR),
  mean(QP_SubData[QP_SubData$Texture_QPISlice==32,]$AVE_PSNR),
  mean(QP_SubData[QP_SubData$Texture_QPISlice==34,]$AVE_PSNR),
  mean(QP_SubData[QP_SubData$Texture_QPISlice==36,]$AVE_PSNR),
  mean(QP_SubData[QP_SubData$Texture_QPISlice==38,]$AVE_PSNR),
  mean(QP_SubData[QP_SubData$Texture_QPISlice==40,]$AVE_PSNR))
```

即可获得不同 `QP_texture` 的 `PSNR` 的均值的统计表，获得 `PSNR` 均值的差值的命令如下：

```
PSNR_Difference <- vector()
for (i in 1:(length(PSNR_Statistic)-1)) {
  PSNR_Difference <- cbind(PSNR_Difference, PSNR_Statistic[i] -
    PSNR_Statistic[i+1])
}
AVE_QP_Difference_PSNR <- mean(PSNR_Difference)
```

首先构造空的向量表 `PSNR_Difference`，通过循环遍历将不同 `QP_texture` 值的 `PSNR` 差值，最后将差值通过 `mean` 函数得出均值。

使用类似的方法也可以得到不同 `QP_texture` 值所得到的比特率均值，代码如下：

```
QP_Rate_Difference <- vector()
for (i in 1:(length(Rate_Statistic)-1)) {
  QP_Rate_Difference <- cbind(QP_Rate_Difference,
    (Rate_Statistic[i] -
      Rate_Statistic[i+1])/Rate_Statistic[i])
}
```



```
}
```

```
AVE_QP_Rate_Difference <- mean(Rate_Difference)
```

与获得 *PSNR* 均值差值方法类似，对于统计值 *Rate\_Statistic* 的获取部分已省略。

#### 4.3.6 实验结果分析

本节主要以绘图的直接观察和列表的数据列举方式来对实验结果进行分析和总结。其中绘图全部以纹理量化参数为区分变量进行分析。

##### 1. 量化参数测试组

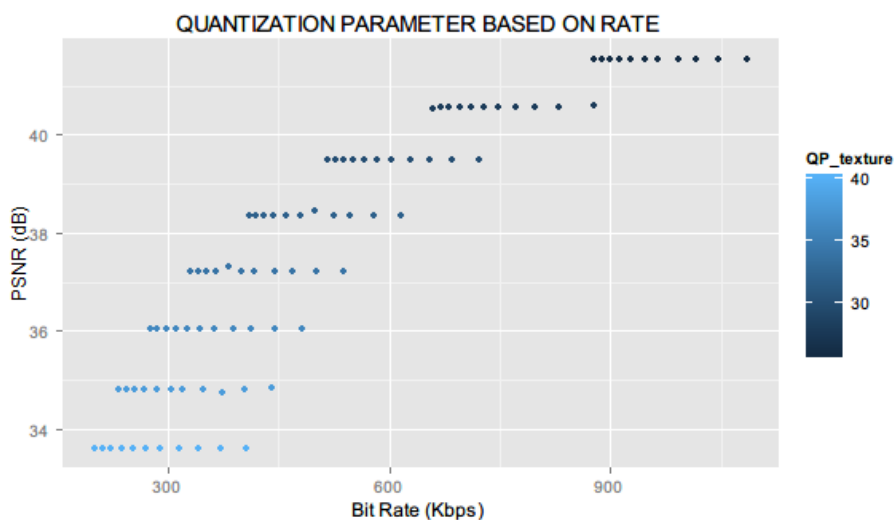


图 4-6 量化参数数据分析

该组实验测试了在多组纹理图量化参数值 (*QP\_texture*) 下不同的深度图量化参数值对最后虚拟视点纹理图的 *PSNR* 值造成的影响。图 4-6 中以纹理图和深度图的比特率之和作为横坐标 (*Bit Rate*)，以 *PSNR* 值为纵坐标 (*PSNR (dB)*)。颜色深度越浅的点具有的量化参数值越高，而量化参数值越高，*PSNR* 均值也就更低。

在 R 语言中通过如下命令可以得到不同纹理量化参数值的 *PSNR* 均值（以量化参数值为 26 为例）：

```
mean(QP_SubData[QP_SubData$Texture_QPISlice==26,]$AVE_PSNR)
```

表 4-5 量化参数 *PSNR* 均值

纹理量化参数值	<i>PSNR</i> 均值
26	41.55
28	40.56
30	39.49
32	38.37
34	37.23
36	36.05
38	34.81
40	33.62

从表 4-5 中可知每组量化参数的 *PSNR* 均值随着量化参数递增(这里递增值为 2), *PSNR* 值会降低, 分别为 0.989dB, 1.063dB, 1.133dB, 1.139dB, 1.178dB, 1.238dB, 1.196dB。统计得到 *PSNR* 值的平均差值为 1.134dB。

关于求均值的核心代码为(以 *QP* 等于 26 为例):

```
mean(QP_SubData[QP_SubData$Texture_QPISlice==26,$AVE_PSNR)
```

类似地可以计算固定纹理量化参数的比特率均值, 如表 4-6 所示:

表 4-6 VSP 算法不同量化参数的比特率均值

纹理量化参数值	比特率均值
26	959.57
28	742.84
30	597.79
32	491.48
34	412.71
36	357.16
38	315.43
40	283.03

通过统计计算, 纹理量化参数每增加 2, 比特率减少约 15.91%。

对得到的比特率和 *PSNR* 差值向量可在 RStudio 控制台直接键入向量名查看数值:

```
> Rate_Statistic
```

```
> PSNR_Statistic
```

## 2. 基于深度的运动估计测试组

在图 4-7 中, *OnOff* 表示 DMVP 算法的开启或关闭状态, 依然以纹理图加深度图的比特率为横坐标, 虚拟视点的纹理图为纵坐标: 对于相同的比特率, 开启 DMVP 时,

PSNR 会普遍提升。

基于量化参数统计结果如表 4-7:

表 4-7 DMVP 算法不同量化参数的 PSNR 值

纹理量化参数值	DMVP 打开	DMVP 关闭
26	41.55	41.34
28	40.56	40.32
30	39.51	39.25
32	38.37	38.13
34	37.23	36.95
36	36.05	35.75
38	34.79	34.51
40	33.59	33.17

基于量化参数计算时打开 DMVP 时 PSNR 平均增加 0.278dB。然而, 在对视频质量测试时, 通常使用相同的比特率作比较, 因而从这个更具效力的角度看, PSNR 的值差异非常明显, 近似 1.5dB 的差异也体现了 DMVP 算法具有更高的有效性, 根据相同的量化参数值比较比特率的变化, 如表 4-8:

表 4-8 DMVP 算法不同量化参数的比特率值

纹理量化参数值	DMVP 打开	DMVP 关闭
26	1169.96	1386.01
28	867.77	1069.20
30	657.07	842.70
32	500.15	667.40
34	386.03	540.68
36	301.07	451.15
38	236.99	378.99
40	187.63	326.22

从统计的结果看, 对于相同的量化参数, DMVP 打开能够减少约 27.9%的比特率的同时增加了 0.278dB 的峰值信噪比。

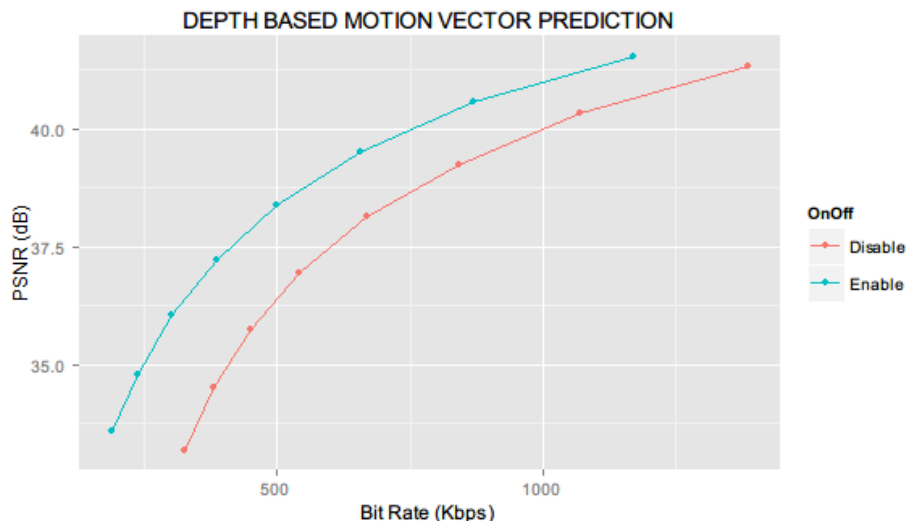


图 4-7 基于深度的运动估计数据分析

### 3. 自适应亮度补偿测试组

对自适应亮度补偿的测试类似于基于深度的运动估计测试，仅 *OnOff* 区分变量变为开启或关闭自适应亮度补偿。

从图 4-8 中可以看到，自适应亮度补偿同样对 *PSNR* 值有提升效果，虽然在比特率的减少方面不如 DMVP 算法，但在 *PSNR* 值的提升方面要更好。针对不同量化参数的统计结果如表 4-9：

表 4-9 ALC 算法在不同量化参数的比特率值

纹理量化参数值	ALC 打开	ALC 关闭
26	1166.85	1254.97
28	876.84	948.77
30	654.1	716.81
32	498.31	535.36
34	382.15	408.36
36	297.19	311.16
38	232.98	242.27
40	181.38	187.49

根据统计结果可得对于相同的量化参数，平均的 *PSNR* 提升值约 0.307dB。而比特率只减少了 6.1%。

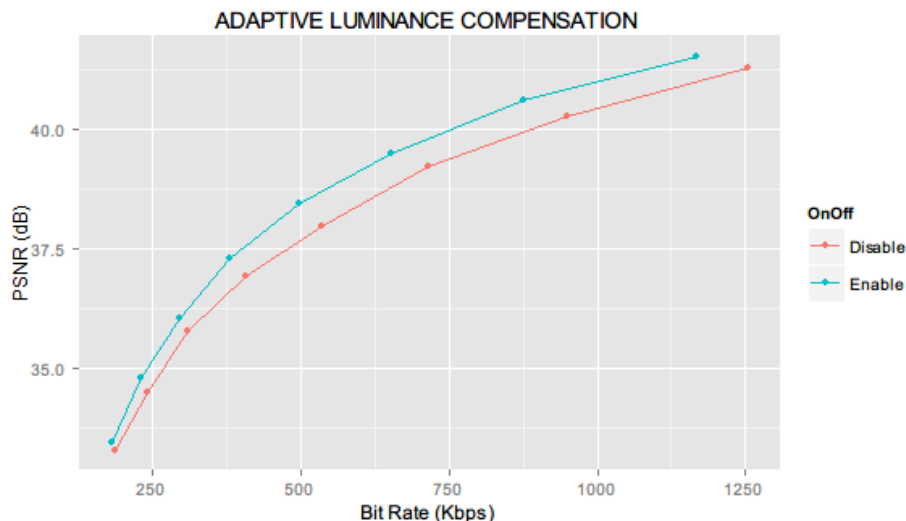


图 4-8 自适应亮度补偿数据分析

参照表 4-10，为 ALC 算法对不同量化参数的  $PSNR$  值得比较

表 4-10 ALC 算法在不同量化参数的  $PSNR$  值

纹理量化参数值	ALC 打开	ALC 关闭
26	41.55	41.29
28	40.61	40.28
30	39.50	39.23
32	38.43	38.97
34	37.30	36.93
36	36.04	35.78
38	34.80	34.47
40	33.43	33.26

#### 4. 视点合成预测测试组

该组实验与基于深度的运动估计测试组类似，仅 *OnOff* 区分变量变为视点合成预测开启或关闭。

如图 4-9 所示的视点合成预测数据分析，可以看到观测的数据结果  $PSNR$  值差别很小，近似忽略。

表 4-11 所示为 VSP 开启和关闭时比特率值的比较。统计结果显示，开启 VSP 后的比特率减少了 0.9%，而  $PSNR$  的值增加了仅 0.003dB，可忽略不计。

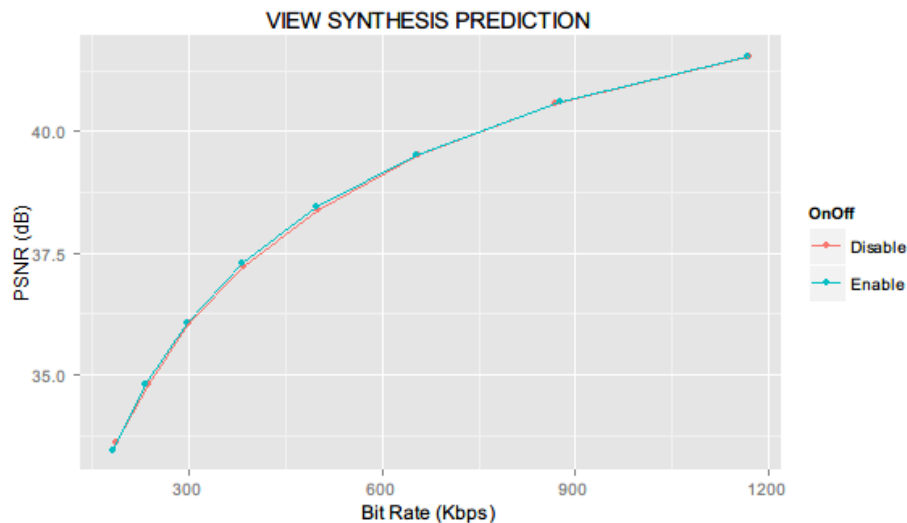


图 4-9 视点合成预测数据分析

表 4-11 VSP 算法在不同量化参数的比特率值

纹理量化参数值	VSP 打开	VSP 关闭
26	1166.85	1169.96
28	876.84	867.77
30	654.10	657.07
32	498.31	500.15
34	382.15	386.03
36	297.19	301.07
38	232.98	236.99
40	181.38	187.63

相比较 DMVP 和 ALC 算法而言，由于 VSP 算法在自身优化度上的问题，在比特率减少和 PSNR 值得降低方面均表现不佳。

表 4-12 给出了相关的 PSNR 统计值。

表 4-12 VSP 算法在不同量化参数的 PSNR 值

纹理量化参数值	VSP 打开	VSP 关闭
26	41.55	41.55
28	40.61	40.56
30	39.50	39.50
32	38.43	38.37
34	37.30	37.23
36	36.04	36.05

续表

38	34.80	34.79
40	33.43	33.59

注：由于 VSP 打开时实验测试配置与 ALC 测试组相同故得出的  $PSNR$  值也相同。

## 5. 总结

综上所述，QP 在纹理图的参数值越低， $PSNR$  值越高，但同时也有较高的比特率，验证了量化步长对两者同时的影响。在此基础上实验做了关于不同算法的测试。其中，DMVP 实验组和 ALC 实验组都获得了理想的结果参数，而 VSP 结果不尽人意。在比特率的减少方面，DMVP、ALC 和 VSP 分别有 27.9%、6.1% 和 0.9% 的减少量，同时  $PSNR$  值方面三个算法分别增加了 0.278dB、0.307dB 和 0.003dB（其中 VSP 实验与理想情况相反）。DMVP 算法在比特率减少减少方面最优，而 ALC 在  $PSNR$  值增加方面最优，综合而言 DMVP 的整体性能提升最高。另外，所做的量化参数部分显示了量化参数每增加 2 在减少 15.9% 比特率的同时也减少了近 1.134dB 的  $PSNR$  值。

## 结 论

本论文就结合深度图的立体视频相关理论做了介绍，尤其集中在立体视频数据格式、视频编解码和视点合成方面。按照发展顺序，对 MVV、LDV、MVD 等立体视频数据格式进行了比较和分析。视点合成是主要针对虚拟视点合成过程中优化方法做了简要介绍，伴随着视点合成的话题，在背景章节中同时引入了关于深度数据的获取和坐标系变换知识。本文重点在视频编解码中算法差别和量化参数的实验数据方面做了分析，并提出了利用 R 语言和 RStudio 平台进行更加智能化的现代数据分析者的分析方式，通过整理数据、绘图分析和数值计算，进行数据的分析。分析结果显示，大部分测试组的结果与理论一致，其中 DMVP、ALC 和 VSP 三类算法分别能减少 27.9%、6.1% 和 0.9% 的比特率，而 PSNR 值方面，三种算法分别改进了 0.278dB、0.307dB 和 0.003dB。可见 DMVP 在三者中的综合性优势以及 VSP 算法在实践上的低下效果，进一步验证了压缩算法研究的意义。此外，实验还验证了关于纹理量化参数值对比特率和 PSNR 值得影响，结果显示纹理量化参数每增加 2，比特率大约减少 15.9% 但同时 PSNR 值也会减少约 1.134dB。

根据实验中遇到的困难，有三点可以改进的地方：

1. 通过实验结果可以发现对于不同算法的 PSNR 值差别并不大，而且部分情况会出现于理论方案相反的结果，主要是由于编码帧数的数量有限，因此增加编码帧数是完善实验的一个方向。例如将算法实验中的 100 帧增加到 300 帧左右。
2. 另一方面，实验仅仅对“kendo”实验序列做了相关的工作，可以改进的一步是对微软公司提供的其他序列，如“bird”等进行相同的实验方案，以减少实验数据的偶然性问题。
3. 最后一点可选是对算法软件代码的改进，通过将软件编解码的部分结果按照观测值结果的格式写入文本文档（.txt）或电子表格（.csv）中。从而大大减少点击工作量，能极大地提高实验效率。例如，预设好需要更改的参数配置表，设置编解码程序按顺序执行编解码，每次的结果按行输出。最后可以统一对获得的数据进行分析处理，从而优化实验数据分析。



## 致 谢

感觉转眼间，大学已经过去了思念。在我快要写完这篇论文的时候，心情也和刚入校的时候截然不同。接下来的路程会让我和社会有更多的接触面，但是发自内心的还是从小学、初中、高中到大学，终于是学到了尽头，我还是很兴奋能尽快把学到的东西用在以后的学习中。感谢指导我的博士生，刘翔凯，在很多问题、文献的阅读引导以及相关软件使用上的帮助。同样感谢指导老师彭强在报告和论文中帮我指出问题所在，更能在每次的修改中给我提出非常有价值的意见。还有一点我必须感谢的是学校电声乐队对我以后成长的启发，我难以想象如果没有音乐，这个除了软件以外的第二特长，该如何发现未来理想的行进方向。

大四这一年我很感谢学校提供的交流项目的机会，来到了加拿大的渥太华大学交流学习，让我感悟了很多也领悟了很多。还有一个让我大四奋发的就是当下热门的MOOC教育，通过在Coursera教育平台里学习，扩宽自己的知识面，用广度来增强自己的学习能力。我更坚定了以后的工作方向，同时也的确让我体会到了工程这一类能应用到实践的魅力，并且改变了以往我对知识深度的追求，而变为对知识广度的追求。论文中所提出的基于R语言的知识要感谢约翰·霍普金斯大学公共卫生学院开设的数据分析系列课程，尤其是Roger Peng老师的用心教学令我收获巨大。

写这篇论文的过程中，我依然发现自己在理解方面存有一定的盲点。但是谁又能说自己学到了尽头呢，我们努力去发现、去寻找，往往就差这一步之遥，而后就是让你兴奋无比的绿洲。

## 参考文献

- [1] Smolic A, Mueller K, Merkle P, et al. An overview of available and emerging 3D video formats and depth enhanced stereo as efficient generic solution[C]//Picture Coding Symposium, 2009. PCS 2009. IEEE, 2009: 1-4.
- [2] Zhao Y, Zhu C, Yu L, et al. An Overview of 3D-TV System Using Depth-Image-Based Rendering[M]//3D-TV System with Depth-Image-Based Rendering. Springer New York, 2013: 3-35.
- [3] Daribo I, Saito H. A novel inpainting-based layered depth video for 3DTV[J]. Broadcasting, IEEE Transactions on, 2011, 57(2): 533-541.
- [4] Bartczak B, Vandewalle P, Grau O, et al. Display-independent 3D-TV production and delivery using the layered depth video format[J]. Broadcasting, IEEE Transactions on, 2011, 57(2): 477-490.
- [5] Smolic A, Muller K, Dix K, et al. Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems[C]//Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on. IEEE, 2008: 2448-2451.
- [6] 郭琪. 三维视频深度图像处理及其 ASIC 设计研究[D]. 山东大学, 2012.
- [7] 季茂胜. 三维视频深度信息压缩技术研究[D]. 中国科学技术大学, 2010.
- [8] Dmytro R, Fang-Chu C, Li Z, et al. 3D AVC Test Model 8. Documento: JCT3V-F1003[J].
- [9] Merkle P, Smolic A, Muller K, et al. Efficient prediction structures for multiview video coding[J]. Circuits and Systems for Video Technology, IEEE Transactions on, 2007, 17(11): 1461-1473.
- [10] Yamamoto K, Kitahara M, Kimata H, et al. Multiview video coding using view interpolation and color correction[J]. Circuits and Systems for Video Technology, IEEE Transactions on, 2007, 17(11): 1436-1449.
- [11] Yea S, Vetro A. View synthesis prediction for multiview video coding[J]. Signal processing: image communication, 2009, 24(1): 89-100.
- [12] Shimizu S, Kimata H, Ohtani Y. Adaptive appearance compensated view synthesis prediction for multiview video coding[C]//Image Processing (ICIP), 2009 16th IEEE International Conference on. IEEE, 2009: 2949-2952.
- [13] Su W, Rusanovskyy D, Hannuksela M M, et al. Depth-based motion vector prediction in 3D video coding[C]//Picture Coding Symposium (PCS), 2012. IEEE, 2012: 37-40.
- [14] Weng S K, Kuo C M, Tu S K. Video object tracking using adaptive Kalman filter[J].

- Journal of Visual Communication and Image Representation, 2006, 17(6): 1190-1208.
- [15] 毕厚杰. 新一代视频压缩编码标准[J]. 2004.
- [16] Criminisi A, Pérez P, Toyama K. Region filling and object removal by exemplar-based image inpainting[J]. Image Processing, IEEE Transactions on, 2004, 13(9): 1200-1212.
- [17] Tian D, Lai P L, Lopez P, et al. View synthesis techniques for 3D video[J]. Proceedings of applications of digital image processing XXXII, 2009, 7443: 74430T-1.
- [18] Lee M S. Method and apparatus for encoding/decoding a video signal: U.S. Patent 5,598,216[P]. 1997-1-28.
- [19] Tian D, Zou F, Lee C, et al. Analysis of view synthesis prediction architectures in modern coding standards[C]//SPIE Optical Engineering+ Applications. International Society for Optics and Photonics, 2013: 88560V-88560V-10.
- [20] 宋彬, 常义林, 李春林. H. 264 帧间预测模式的快速选择算法[J]. 电子学报, 2007, 35(4): 697-700.
- [21] Zhao Y, Zhu C, Yu L. Virtual View Synthesis and Artifact Reduction Techniques[M]//3D-TV System with Depth-Image-Based Rendering. Springer New York, 2013: 145-167.
- [22] Chen W Y, Chang Y L, Lin S F, et al. Efficient depth image based rendering with edge dependent depth filter and interpolation[C]//Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on. IEEE, 2005: 1314-1317.
- [23] Daribo I, Pesquet-Popescu B. Depth-aided image inpainting for novel view synthesis[C]//Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on. IEEE, 2010: 167-170.

## 附录 1 数据分析平台搭建

### 5.1 下载

#### 5.1.1 R 语言基础包下载

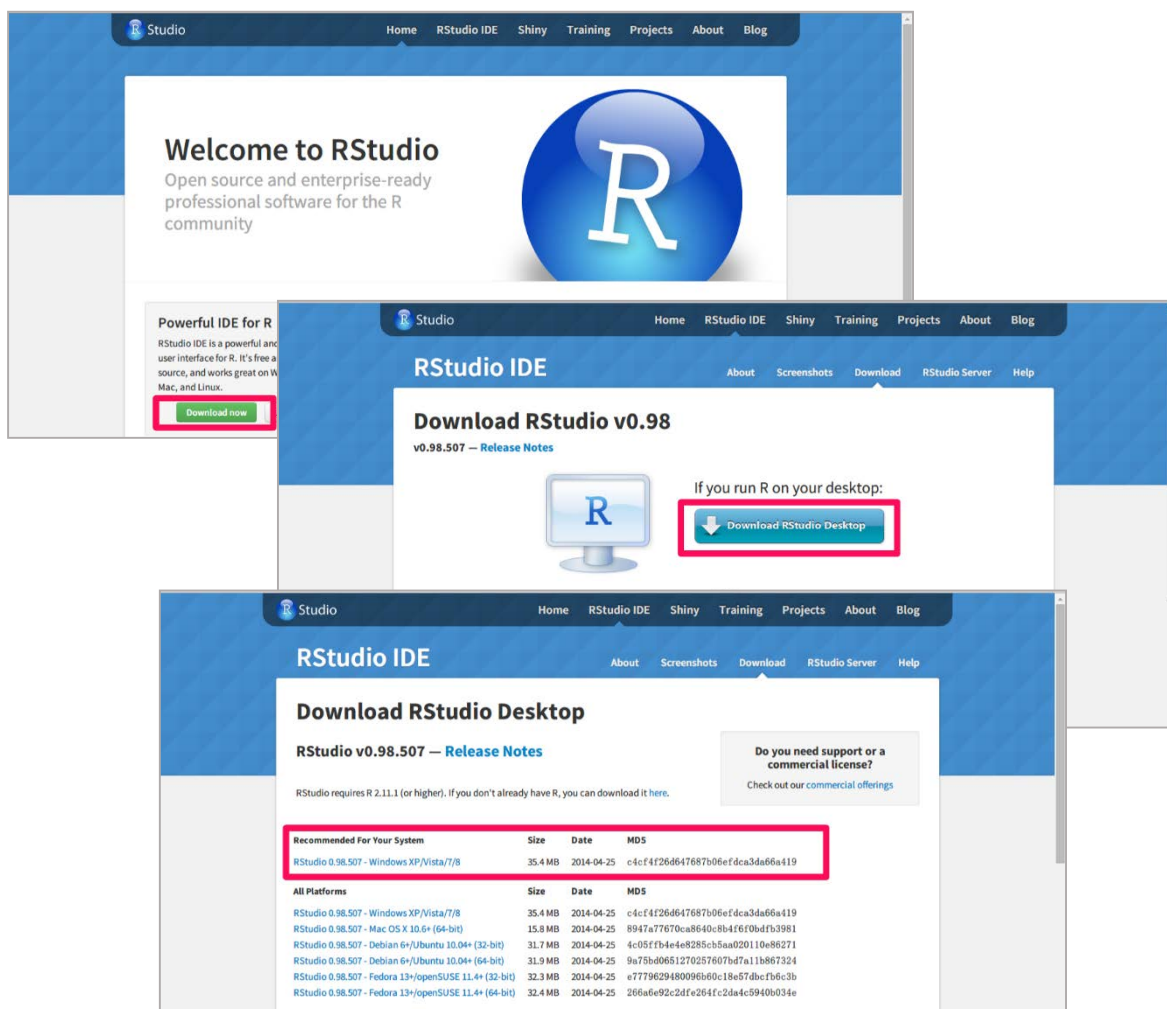
论文中所使用的 R 语言基础包，来自开源项目 CRAN（Comprehensive R Archive Network），官方网址 <http://cran.r-project.org/>（如图）。根据系统环境选择 Linux/Mac/Windows，作者使用环境为 Windows，故以此为例。依次点选 [Download R for Windows](#) - [base](#) - [Download R 3.1.0 for Windows](#) (54 megabytes, 32/64 bit)，即可下载该语言包。



#### 5.1.2 RStudio 下载

论文中所讲述的 R 工作室（RStudio）是 R 语言的集成开发环境，易于 R 语言工作者的使用。RStudio 对个人用户免费，官方网址 <http://www.rstudio.com/>。点选 [Download now](#)，RStudio 不仅提供了桌面版客户端，同时也为数据处理云端工作者提供了基于 Linux 平台的服务器端以提供对多客户端的接入，作者使用基础的桌面版客户端为例，点选 [Download RStudio Desktop](#)。RStudio 支持了包括 Windows、Mac OS

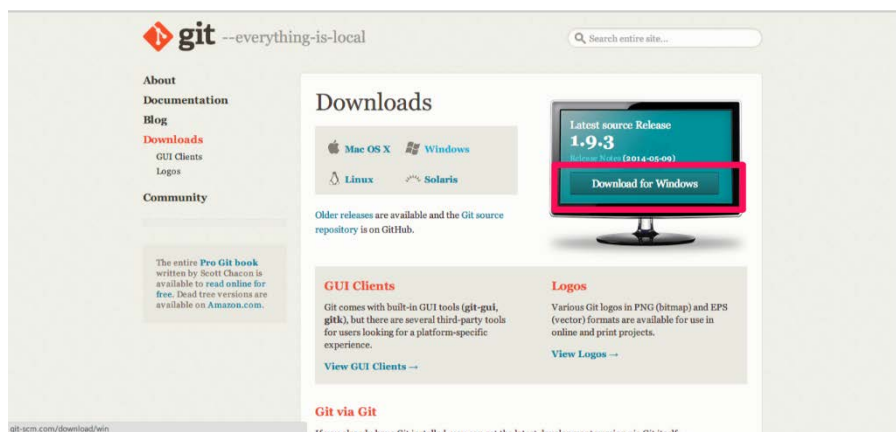
和基于 Linux 的开发套件（如 Fedora、Ubuntu）。网站会根据浏览器的系统使用环境推荐下载的客户端类型，如跟当前操作系统结果不符，请手动选择合适的系统客户端。此处以 Windows 系统为例，点选 [RStudio 0.98.507 - Windows XP/Vista/7/8](#) 即可下载客户端程序。



### 5.1.3 Git Bash 下载

作者已将所需的 R 语言脚本文件作为公开源放在 Github 托管项目中，可使用 Github 桌面客户端或版本控制工具下载，作者这里使用版本控制工具 Git Bash 来实现，官方下载地址 <http://git-scm.com/downloads>。

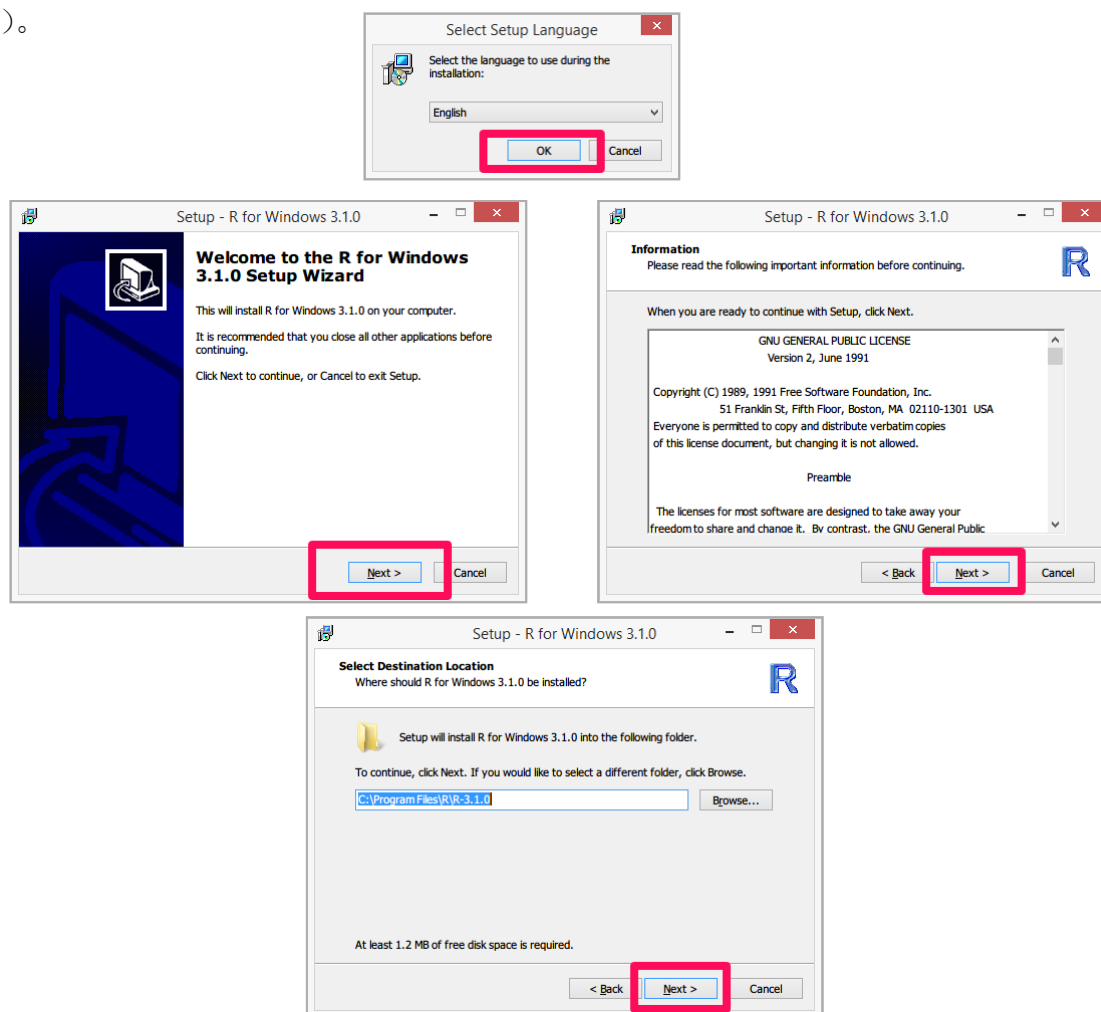
同样地，Git Bash 也支持多个操作系统，本处选择 Windows 客户端，点选 [Download for Windows](#) 下载安装包。



## 5.2 安装

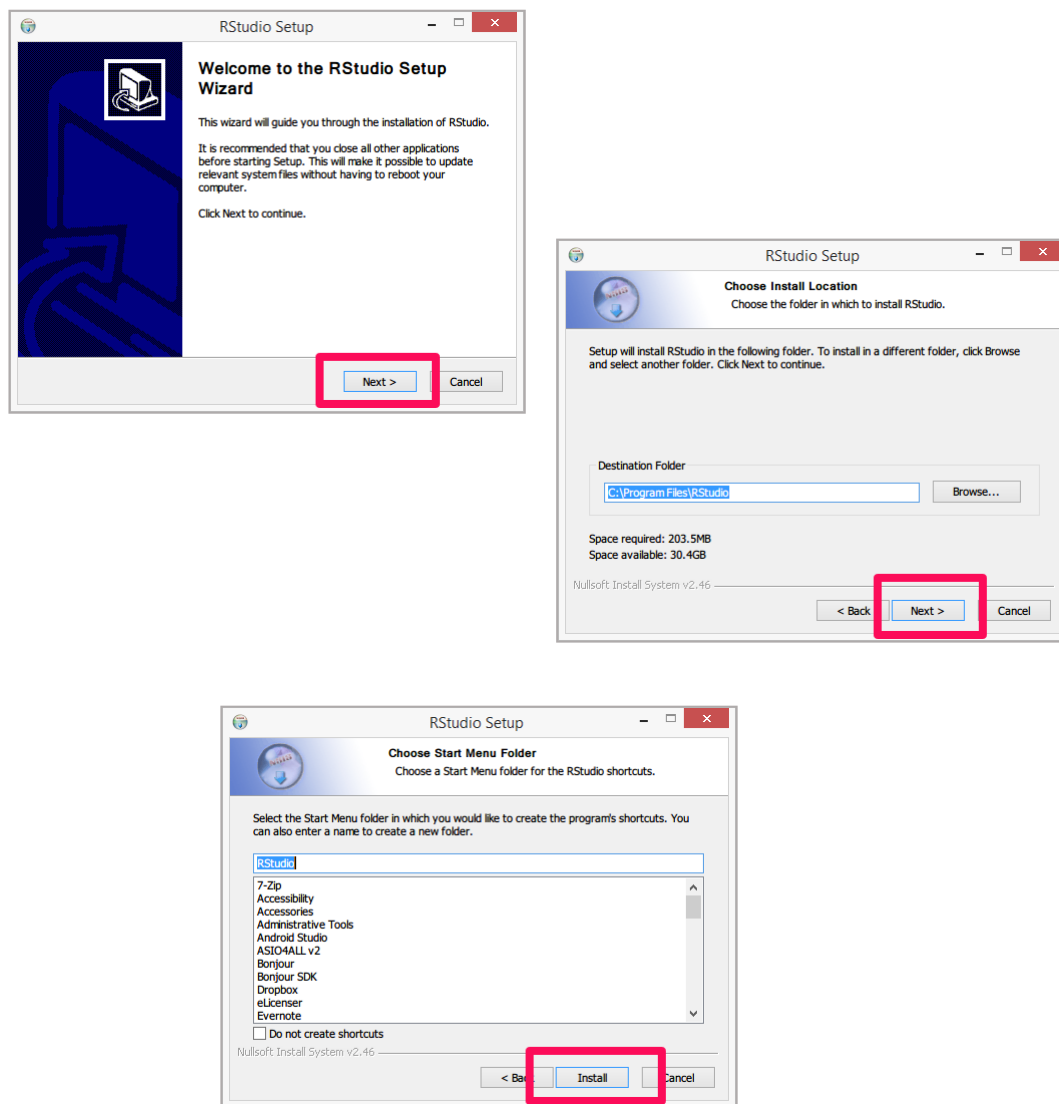
### 5.2.1 R 语言 CRAN 基础包安装

根据作者经验，RStudio 对中文目录索引的支持性很差，为避免不必要的麻烦，安装包均选为英文操作语言。由于主要操作环境为 RStudio，因此 R 语言 CRAN 基础包的安装均选择默认选项（作者已安装故后续步骤省略，请记住安装目录，后面需要使用）。



## 5.2.2 RStudio 安装

与安装 CRAN 包类似, RStudio 也均可以按默认选项(推荐使用默认路径)安装, 作者已安装故后续步骤省略。



## 5.2.3 Git Bash 安装

类似 CRAN 包和 RStudio 的安装, 均选择默认选项, 此处不做赘述。

## 5.3 创建与配置

### 5.3.1 克隆 R 语言脚本

在需要下载 R 语言脚本的文件夹中右键空白处, 选择 Git Bash 即可进入 Git Bash 并将当前目录设置为点选位置的目录。

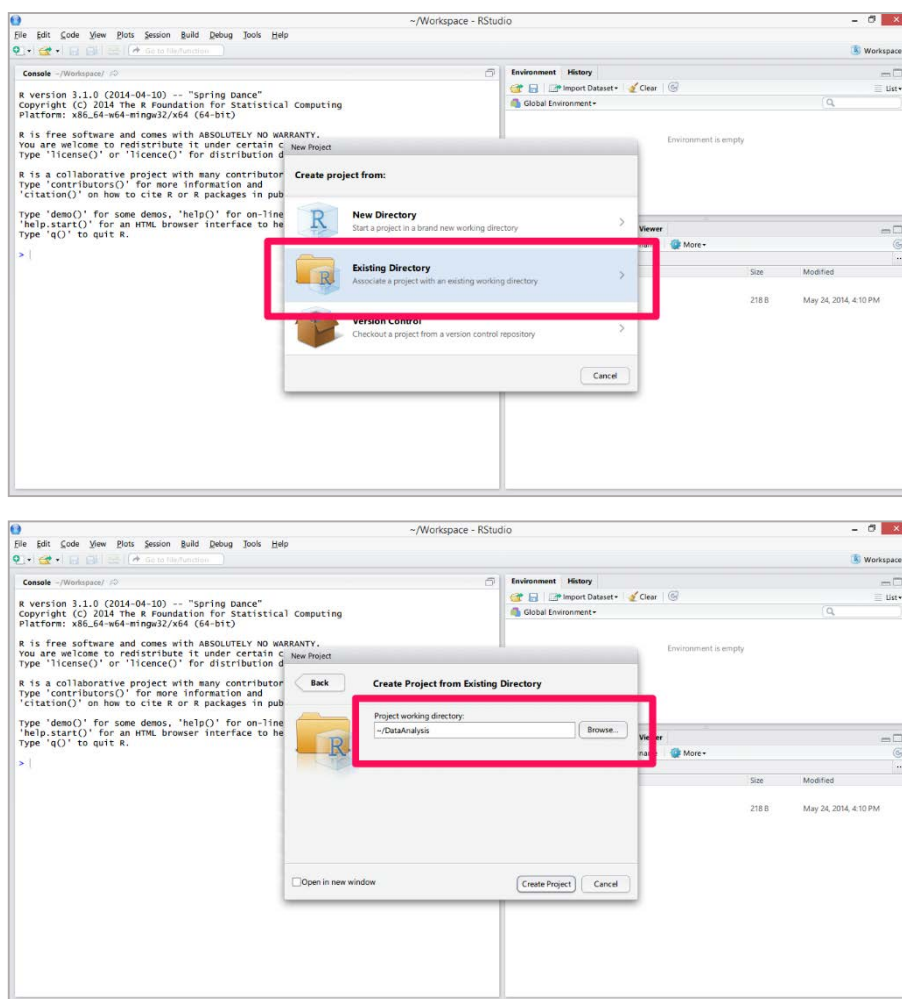
手动键入命令 `git clone https://github.com/willowchan/DataAnalysis.git`

即可看见文件夹中，创建了名为 `DataAnalysis` 的子文件夹，并且其中包含了所需的 R 语言脚本文件以及本次实验的原始数据，文件夹里同时包含了文本文档和电子表格的数据表格式。（注:文件均为作者制作，如需转载请注明出处）

### 5.3.2 创建工程

由于 CRAN 包仅仅作为依赖包，因而后续步骤均在 RStudio 中操作。

打开 RStudio，依次点选菜单栏 `File – New Project`，点选 `Existing Directory` 选择刚才创建的 `DataAnalysis` 目录路径。



需要注意的是默认工程的目录路径位于系统文件夹的 `Documents`(文档)目录下，在操作界面用“~”符号表示。

将需要分析的数据以 `xlsx` 形式的表格命名为“`RawData.xlsx`”保存在工程目录下（`./DataAnalysis/RawData.xlsx`）。

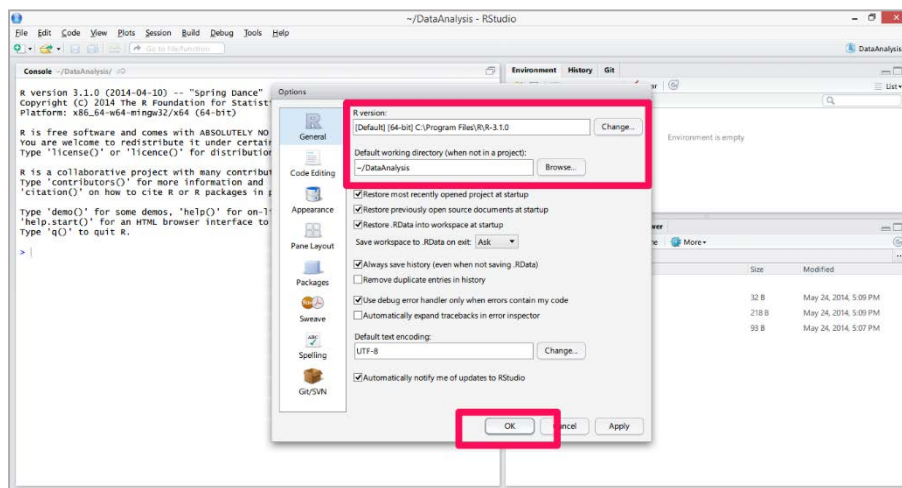


### 5.3.3 修改配置文件

为了正常使用之前安装的 R 语言 CRAN 基础包，点选菜单栏 Tools – Global Options，弹出界面左边栏选择 General，确保 R version 一栏中的目录与之前安装 CRAN 包目录一致（注意区别 32 位和 64 位）。

可选的是，将 Default Working Directory 设置为“~/DataAnalysis”，与创建的工程目录一致。

点选 OK，完成配置。



## 附录 2 脚本代码

### 1. VSP\_analysis.R

```
#####  
#This script is used for analyzing effect of VSP algorithm  
#####  
  
## Step 1: read raw data from work directory  
# Uncomment next line if.xlsx package haven't been installed  
#install.packages("xlsx")  
# Load dependent library so that .xlsx file can be read  
library(rJava)  
library(xlsxjars)  
library(xlsx)  
RawData <- read.xlsx(file="/RawData.xlsx", sheetIndex=1)  
  
## Step 2: Subset by observations  
# With DMVP enable and ALC enable  
# Part of quantization parameter (QP) values are chosen  
SubData <- RawData[RawData$DepthBaseMVP=="Enable" &  
                    RawData$AdaptiveLuminanceCompensation=="Enable" &  
                    RawData$FrameToBeEncoded==100, ]  
VSP_SubData <- SubData[SubData$Texture_QPISlice==SubData$ Depth_QPISlice, ]  
  
## Step 3: Subset by variables  
VSP_sData <- data.frame(OnOff=VSP_SubData$VSP_Enable,  
                        Rate=VSP_SubData$SUM_Rate,  
                        PSNR=VSP_SubData$AVE_PSNR)  
# Order by OnOff variable for precise observation  
VSP_Data <- VSP_sData[order(VSP_sData$OnOff),]  
  
## Step 4: Use ggplot package to plot graph  
# If ggplot2 package haven't been installed, please uncomment next line  
#install.packages("ggplot2")  
# Load ggplot2 library, which implements the grammar of graphics  
library(ggplot2)  
VSP_OnOff <- aggregate(PSNR ~ Rate + OnOff, data = VSP_Data, FUN = sum)  
with(VSP_OnOff, qplot(Rate, PSNR, col = OnOff,
```

```
geom = c("point", "line"),
xlab = "Bit Rate (Kbps)",
ylab = "PSNR (dB)",
main = "VIEW SYNTHESIS PREDICTION"))

## Step 5: Copy graph from device into hard disk in work directory
dev.copy(png, file="VSP_analysis.png")
dev.off()

## Step 6: Analyze the average value of PSNR based on difference QP_texture
AVE_VSP_Difference_PSNR <- mean(VSP_Data[VSP_Data$OnOff=="Enable",]$PSNR) -
  mean(VSP_Data[VSP_Data$OnOff=="Disable",]$PSNR)

## Step 7: Compare the difference when VSP is enable or disable
VSP_Difference_Rate <- vector()
for (i in 1:(length(VSP_Data$OnOff)/2)) {
  VSP_Difference_Rate <- cbind(VSP_Difference_Rate,
(VSP_Data[VSP_Data$OnOff=="Disable",]$Rate[i] -

VSP_Data[VSP_Data$OnOff=="Enable",]$Rate[i]) /

VSP_Data[VSP_Data$OnOff=="Disable",]$Rate[i])
}
AVE_VSP_Difference_Rate <- mean(VSP_Difference_Rate)

# For optional format for analysis
# Txt format of raw data as well as organized data is supplied
write.table(RawData, file="./RawData.txt")
write.table(VSP_Data, file="./VSP_Data.txt")
```

## 2. DMVP\_analysis.R

```
#####
#This script is used for analyzing effect of DMVP algorithm
#####

## Step 1: read raw data from work directory
```

```

# Uncomment next line if xlsx package haven't been installed
#install.packages("xlsx")

# Load dependent library so that .xlsx file can be read
library(rJava)
library(xlsxjars)
library(xlsx)
RawData <- read.xlsx(file="./RawData.xlsx", sheetIndex=1)

## Step 2: Subset by observations
# With VSP disable
DMVP_SubData <- RawData[RawData$VSP_Enable=="Disable" &
                        RawData$FrameToBeEncoded==100, ]

## Step 3: Subset by variables
DMVP_Data <- data.frame(OnOff=DMVP_SubData$DepthBaseMVP,
                        Rate=DMVP_SubData$SUM_Rate,
                        PSNR=DMVP_SubData$AVE_PSNR)

## Step 4: Use ggplot package to plot graph
# If ggplot2 package haven't been installed, please uncomment next line
#install.packages("ggplot2")
# Load ggplot2 library, which implements the grammar of graphics
library(ggplot2)
DMVP_OnOff <- aggregate(PSNR ~ Rate + OnOff, data = DMVP_Data, FUN = sum)
with(DMVP_OnOff, qplot(Rate, PSNR, col = OnOff,
                      geom = c("point", "line"),
                      xlab = "Bit Rate (Kbps)",
                      ylab = "PSNR (dB)",
                      main = "DEPTH BASED MOTION VECTOR PREDICTION"))

## Step 5: Copy graph from device into hard disk in work directory
dev.copy(png, file="DMVP_analysis.png")
dev.off()

## Step 6: Analyze the average value of PSNR based on difference QP_texture
AVE_DMVP_Difference_PSNR <- mean(DMVP_Data[DMVP_Data$OnOff=="Enable",]$PSNR) -
mean(DMVP_Data[DMVP_Data$OnOff=="Disable",]$PSNR)

```

```

## Step 7: Compare the difference when DMVP is enable or disable
DMVP_Difference_Rate <- vector()
for (i in 1:(length(DMVP_Data$OnOff)/2)) {
    DMVP_Difference_Rate <- cbind(DMVP_Difference_Rate,
    (DMVP_Data[DMVP_Data$OnOff=="Disable",]$Rate[i] -
                                DMVP_Data[DMVP_Data$OnOff=="Enable",]$Rate[i]) /
                                DMVP_Data[DMVP_Data$OnOff=="Disable",]$Rate[i])
}
AVE_DMVP_Difference_Rate <- mean(DMVP_Difference_Rate)

# For optional format for analysis
# Txt format of raw data as well as organized data is supplied
write.table(RawData, file="/RawData.txt")
write.table(DMVP_Data, file="/DMVP_Data.txt")

```

### 3. ALC\_analysis.R

```

#####
#This script is used for analyzing effect of ALC algorithm
#####

## Step 1: read raw data from work directory
# Uncomment next line if xlsx package haven't been installed
#install.packages("xlsx")
# Load dependent library so that .xlsx file can be read
library(rJava)
library(xlsxjars)
library(xlsx)
RawData <- read.xlsx(file="/RawData.xlsx", sheetIndex=1)

## Step 2: Subset by observations
# With VSP enable and DMVP enable
# Part of quantization parameters (QP) are chosen
SubData <- RawData[RawData$VSP_Enable=="Enable" &
                                RawData$DepthBaseMVP=="Enable" &
                                RawData$FrameToBeEncoded==100, ]
ALC_SubData <- SubData[

```

```

SubData$Texture_QPISlice==SubData$Depth_QPISlice, ]

## Step 3: Subset by variables
ALC_sData <- data.frame(OnOff=ALC_SubData$AdaptiveLuminanceCompensation,
                        Rate=ALC_SubData$SUM_Rate,
                        PSNR=ALC_SubData$AVE_PSNR)

# Order by OnOff variable for precise observation
ALC_Data <- ALC_sData[order(ALC_sData$OnOff),]

## Step 4: Use ggplot package to plot graph
# If ggplot2 package haven't been installed, please uncomment next line
#install.packages("ggplot2")
# Load ggplot2 library, which implements the grammar of graphics
library(ggplot2)
ALC_OnOff <- aggregate(PSNR ~ Rate + OnOff, data = ALC_Data, FUN = sum)
with(ALC_OnOff, qplot(Rate, PSNR, col = OnOff,
                      geom = c("point", "line"),
                      xlab = "Bit Rate (Kbps)",
                      ylab = "PSNR (dB)",
                      main = "ADAPTIVE LUMINANCE COMPENSATION"))

## Step 5: Copy graph from device into hard disk in work directory
dev.copy(png, file="ALC_analysis.png")
dev.off()

## Step 6: Analyze the average value of PSNR based on difference QP_texture
AVE_ALC_Difference_PSNR <- mean(ALC_Data[ALC_Data$OnOff=="Enable",]$PSNR) -
  mean(ALC_Data[ALC_Data$OnOff=="Disable",]$PSNR)

## Step 7: Compare the difference when ALC is enable or disable
ALC_Difference_Rate <- vector()
for (i in 1:(length(ALC_Data$OnOff)/2)) {
  ALC_Difference_Rate <- cbind(ALC_Difference_Rate,
  (ALC_Data[ALC_Data$OnOff=="Disable",]$Rate[i] -

  ALC_Data[ALC_Data$OnOff=="Enable",]$Rate[i]) /

  ALC_Data[ALC_Data$OnOff=="Disable",]$Rate[i])
}

```

```
AVE_ALC_Difference_Rate <- mean(ALC_Difference_Rate)
```

```
# For optional format for analysis
```

```
# Txt format of raw data as well as organized data is supplied
```

```
write.table(RawData, file="/RawData.txt")
```

```
write.table(ALC_Data, file="/ALC_Data.txt")
```

#### 4. QP\_analysis.R

```
#####
```

```
#This script is used for analyzing effect of different QP values
```

```
#####
```

```
## Step 1: read raw data from work directory
```

```
# Uncomment next line if xlsx package haven't been installed
```

```
#install.packages("xlsx")
```

```
# Load dependent library so that .xlsx file can be read
```

```
library(rJava)
```

```
library(xlsxjars)
```

```
library(xlsx)
```

```
RawData <- read.xlsx(file="/RawData.xlsx", sheetIndex=1)
```

```
## Step 2: Subset by observations
```

```
# With DMVP enable and ALC enable
```

```
# Part of quantization parameter (QP) values are chosen
```

```
SubData <- RawData[RawData$DepthBaseMVP=="Enable" &  
                  RawData$AdaptiveLuminanceCompensation=="Enable" &  
                  RawData$VSP_Enable=="Enable" &  
                  RawData$FrameToBeEncoded==100, ]
```

```
QP_SubData <- SubData[SubData$Depth_QPISlice!=40 &  
                    SubData$Depth_QPISlice!=26, ]
```

```
## Step 3: Subset by variables
```

```
QP_Data <- data.frame(PSNR=QP_SubData$AVE_PSNR,  
                    QP_texture=QP_SubData$Texture_QPISlice,  
                    Rate=QP_SubData$SUM_Rate)
```

```
## Step 4: Use ggplot package to plot graph
```

```

# If ggplot2 package haven't been installed, please uncomment next line
#install.packages("ggplot2")
# Load ggplot2 library, which implements the grammar of graphics
library(ggplot2)
QP_value <- aggregate(PSNR ~ Rate + QP_texture, data = QP_Data, FUN = sum)
with(QP_value, qplot(Rate, PSNR, col = QP_texture,
                     geom = "point",
                     xlab = "Bit Rate (Kbps)",
                     ylab = "PSNR (dB)",
                     main = "QUANTIZATION PARAMETER"))

## Step 5: Copy graph from device into hard disk in work directory
dev.copy(png, file="QP_analysis.png")
dev.off()

## Step 6: Analyze the average value of PSNR depending on difference QP_texture
PSNR_Statistic <- c(mean(QP_SubData[QP_SubData$Texture_QPISlice==26,$AVE_PSNR),
                     mean(QP_SubData[QP_SubData$Texture_QPISlice==28,$AVE_PSNR),
                     mean(QP_SubData[QP_SubData$Texture_QPISlice==30,$AVE_PSNR),
                     mean(QP_SubData[QP_SubData$Texture_QPISlice==32,$AVE_PSNR),
                     mean(QP_SubData[QP_SubData$Texture_QPISlice==34,$AVE_PSNR),
                     mean(QP_SubData[QP_SubData$Texture_QPISlice==36,$AVE_PSNR),
                     mean(QP_SubData[QP_SubData$Texture_QPISlice==38,$AVE_PSNR),
                     mean(QP_SubData[QP_SubData$Texture_QPISlice==40,$AVE_PSNR))

PSNR_Difference <- vector()

## Step 7: Get the mean value of PSNR_Difference
for (i in 1:(length(PSNR_Statistic)-1)) {
    PSNR_Difference <- cbind(PSNR_Difference, PSNR_Statistic[i] - PSNR_Statistic[i+1])
}
AVE_QP_Difference_PSNR <- mean(PSNR_Difference)

## Step 8: Analyzing rate of average based on different QP
Rate_Statistic <- c(mean(QP_SubData[QP_SubData$Texture_QPISlice==26,$SUM_Rate),
                     mean(QP_SubData[QP_SubData$Texture_QPISlice==28,$SUM_Rate),
                     mean(QP_SubData[QP_SubData$Texture_QPISlice==30,$SUM_Rate),
                     mean(QP_SubData[QP_SubData$Texture_QPISlice==32,$SUM_Rate),
                     mean(QP_SubData[QP_SubData$Texture_QPISlice==34,$SUM_Rate),

```



```
mean(QP_SubData[QP_SubData$Texture_QPISlice==36,$SUM_Rate),
mean(QP_SubData[QP_SubData$Texture_QPISlice==38,$SUM_Rate),
mean(QP_SubData[QP_SubData$Texture_QPISlice==40,$SUM_Rate))

## Step 10: Get the mean value of Rate_Difference
QP_Rate_Difference <- vector()
for (i in 1:(length(Rate_Statistic)-1)) {
  QP_Rate_Difference <- cbind(QP_Rate_Difference,
                             (Rate_Statistic[i] - Rate_Statistic[i+1])/Rate_Statistic[i])
}
AVE_QP_Rate_Difference <- mean(QP_Rate_Difference)

# For optional format for analysis
# Txt format of raw data as well as organized data is supplied
write.table(RawData, file="./RawData.txt")
write.table(QP_Data, file="./QP_Data.txt")
```