结合图像融合算法与人眼跟踪的立体视频①

金诗玮, 杨 青, 李建军

(中航华东光电(上海)有限公司, 上海 201114)

摘 要: 由于多视点立体视频合成具有数据量大, 图像处理速度要求较高, 支持的立体视角有限等特点, 这些问题一直没有很好的解决并已成为多视点立体视频产业化的瓶颈. 针对这种情况, 提出了一种基于立体图像融合算法与人眼跟踪算法的立体视频处理系统. 首先, 按顺序循环读取立体视频中的每一帧, 然后用立体图像融合算法对每一帧进行合成运算, 接下来将融合后的图像依原有顺序进行显示与播放. 同时加入人眼跟踪算法, 根据观看者眼部所处的位置实时投放对应视区的图像. 图像融合算法与人眼跟踪的结合有效地扩大了立体视角. 实验结果表明, 该方法实现了将多视点视频在立体显示器中以自由立体显示的方式展现出来, 使观看者在屏幕前可以自由移动而不影响立体观看效果, 同时播放速度流畅, 能给观众带来比较真实的立体感受.

关键词: 自由立体显示; 多视点; 立体图像融合; 立体视频; 人眼追踪

Multi-View Stereo Video Based on Fusion and Eye-Tracking

JIN Shi-Wei, YANG Qing, LI Jian-Jun

(Avic Huadong Photoelectric Co. Ltd, Shanghai 201114, China)

Abstract: As multi-view stereo video has some new characteristics, such as massive volumes of data, high-demand in the velocity of transition, limited 3D view fields supported. These problems have not been well-solved and blocked the industrialization of multi-view stereo video technology. In view of this, this paper proposes a method of multi-view stereo video transition and display based on an image fusion algorithm and eye-tracking. First, we capture each frame from the original video, then, we use an image fusion algorithm, process each frame of the multi-view stereo video and output the frames at the original rate. At the same time, we use eye-tracking, which effectively enlarged the stereo view field. Experimental results show that the Two-dimension views image can be shown on the stereoscopic display in a 3D way. It could transmit the video smoothly and effectively while computing and eye-tracking. It could bring the viewers three-dimensional feelings when watching.

Key words: auto stereoscopic display; multi-view; 3D image synthesis; stereo video; eye-tracking

近年来, 立体图像显示引起了社会各界的广泛关注, 立体显示技术[1,2]一跃成为显示行业发展的重要方向之一, 是新一代显示器件的研究热点. 它能够为观看者提供一定的深度信息, 使观看者能够获得更加全面和直观的信息, 给人以身临其境之感, 深受广大消费者的喜欢.

立体显示技术主要分为头戴式立体显示技术和自由立体显示两大类. 前者需要佩戴诸如偏光眼镜, 快门眼镜之类的辅助装置, 尽管观看的立体显示效果较好, 但降低了观看者的舒适度. 而自由立体显示技术则无需佩戴眼镜或其他辅助视具就可以为观看者呈现立体图像[2], 大大提升了观看的舒适度, 而且方便多人同时观赏, 可增加情感交流, 提升娱乐效果. 本文所使用的立体显示器使用的是基于柱镜光栅的多视点自由立体显示技术[3]. 其原理是利用柱透镜单元的折射作用, 引导光线进入特定的观察区, 产生对应左右眼的立体图像对, 并最终在大脑的融合下产生立体视觉. 由于柱镜光栅是透射式的, 因此利用这种技术生产的 LCD 自由

①收稿时间:2014-01-09; 收到修改稿时间:2014-03-24

立体显示器的最大优点是不遮挡画面, 不影响显示亮度, 立体显示效果比较好[3]. 作为一种新型的显示设备, 立体显示器使人机交互更加直观, 给观众带来了传统显示器所不能比拟的高度临场感, 身临其境的逼真感觉和无与伦比的立体视觉享受.

作为公认的重要的 3D 显示技术, 多视点立体视频技术越来越受到学术界和工业界的重视, 并且成为当前视频研究领域的热点之一, 通过该技术实现的商业应用也将日渐成熟并普及.

1 相关工作与问题分析

本文基于立体图像融合算法, 提出了一套基于 OpenCV 的多视点立体视频处理系统的系统架构. 进行了多视点立体视频编解码的研究. 将立体图像融合算法嵌入到视频的解析与播放过程中. 同时在此过程中加入人眼跟踪算法, 扩大了用户观看的视角, 使用户在屏幕前无需主动寻找合适的视区就能方便的观看到自由立体视频. 其生成的立体视频在立体显示器上播放, 可以给观众带来真实的立体感觉. 本软件的开发建立在对自由立体显示器光学原理的研究基础上, 结合了对用户体验的优化, 界面易于操作, 合成过程速度快, 画面精细准确, 在自由立体显示器研发中得到了良好的应用.

2 立体显示原理介绍

立体显示器的原理是基于 “视差创造立体”的理论. 所谓“视差”, 是指人的两只眼睛是从不同的角度观看世界的, 就是说, 人的左眼看到的物体与右眼看到的同一物体相互之间有小的差别, 平均相差约 6.5 厘米左右. 由于两眼看到的物体有细微差别, 因而描述场景轮廓的方式也不尽相同. 大脑根据这两个有细微差别的场景进行综合处理, 产生精确的 3D 物体, 以及该物体在场景中的定位, 这就是具有深度感的立体感觉[4].

本文所基于的立体显示器使用的是基于柱镜光栅的自由立体显示技术. 显示屏上每个 RGB 子像素的高度为其宽度的 3 倍. 每个柱镜光栅单元在水平方向上覆盖 N 个 RGB 子像素, 其透镜轴是与显示屏相平行的. 在柱镜光栅的作用下, N 个视点的子像素会被全部放大投射到观察者眼前, 只要观察者的眼睛落在任意两个相邻视点, 就可以看到相应的视点图像进而形成立体视觉. 以 N=8 为例, 8 视点的立体显示器显示原理如图 1 所示.

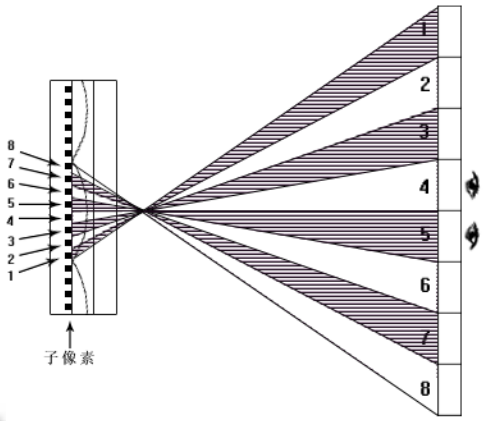


图 1 8 视点立体显示器显示原理示意图

3 图像合成算法介绍

本节将介绍基于柱镜光栅的自由立体显示技术中的多视点子像素映射关系, 然后在此基础上实现一种通用的立体图像合成方法以及一种通用的立体图像分解方法.

3.1 多视点子像素映射关系

多视点子像素映射的目的是为了确定显示屏上给定的 RGB 子像素(x, y)应该取自哪个视点的 RGB 分量. 设N_{tot}表示视点数目, 本文以 8 视点为例, 假设(k, l)表示像素点在坐标系中的位置, X 表示片源个数, N 表示像素取自哪幅视点图像的序号, α 为光栅倾斜角度, 由多视点子像素映射的通用计算公式[5,6], 如式(1)所示, 可知:

$$N=\frac{(k\times 3\cdot l\cdot\tan\alpha)\times \mathrm{mod} X}{X}N_{tot} \tag{1}$$

选定一个子像素(k, l),其 x,y 轴坐标的值是已知的, 而 X,α,和 N_{tot} 的值是固定的, 因此我们可以计算得到该子像素的 RGB 分量应该取自视点 N 所对应的 RGB 分量, 以此类推, 我们还可以进一步计算得到立体图像与各视点图像之间的子像素映射表[7]. 例如, 当 N_{tot} = 8, X = 8, α= $arctan$(2/9),分辨率为 1920*1080 时, 我们可以得到如图 2 所示的 8 个视点的 RGB 分量与立体图像的 RGB 分量相对应的映射表.

3.2 立体图像合成算法

基于实现复杂度和通用性的考虑, 与现有的基于多视点的自由立体显示设备配套的立体图像合成方法一般只适用于具有特定视点值和倾斜角 α 值的柱镜光栅, 而且一般也只能合成具有固定视点个数和固定分辨率的

立体图像, 这样的立体图像合成方法显然缺乏普适性, 大大限制了基于柱镜光栅的自由立体显示设备的应用领域. 对此, 本文使用了一种基于多视点自由立体显示设备的立体图像合成方法, 该方法适用于任意倾斜角度的柱镜光栅, 而且能够快速而高效地合成任意分辨率的立体图像, 从而扩展了基于多视点柱镜光栅的自由立体显示设备的应用领域.

R 7	G 6	B 5	R 4	G 3	B 2	R 1	G 8	B 7	R 6	G 5	B 4	R 3	G 2	B 1	
R 6	G 5	B 4	R 3	G 2	B 1	R 8	G 7	B 6	R 5	G 4	B 3	R 2	G 1	B 8	
R 5	G 4	B 3	R 2	G 1	B 8	R 7	G 6	B 5	R 4	G 3	B 2	R 1	G 8	B 7	
R 5	G 4	B 3	R 2	G 1	B 8	R 7	G 6	B 5	R 4	G 3	B 2	R 1	G 8	B 7	
R 4	G 3	B 2	R 1	G 8	B 7	R 6	G 5	B 4	R 3	G 2	B 1	R 8	G 7	B 6	
R 3	G 2	B 1	R 8	G 7	B 6	R 5	G 4	B 3	R 2	G 1	B 8	R 7	G 6	B 5	
R 3	G 2	B 1	R 8	G 7	B 6	R 5	G 4	B 3	R 2	G 1	B 8	R 7	G 6	B 5	
R 2	G 1	B 8	R 7	G 6	B 5	R 4	G 3	B 2	R 1	G 8	B 7	R 6	G 5	B 4	
R 1	G 8	B 7	R 6	G 5	B 4	R 3	G 2	B 1	R 8	G 7	B 6	R 5	G 4	B 3	

图 2　8 个视点的子像素映射表

以 8 视点为例, 基于 8 视点的自由立体显示设备的立体图像合成方法包括以下步骤:

1) 确定柱镜光栅的实际视点值和实际倾斜角值;

2) 读取片源中的多个视点的图片, 生成一个图像序列. 如果输入格式为 3x3 形式的 9 格图片, 需要先将 9 格图片切割为 9 个部分, 依次读取各部分的图片像素并各自生成一幅单视点图片, 然后将每张图片依次插入一个源图像列表中. 将该列表中的图片放大至目标立体图像的像素尺寸.

3) 根据映射表, 将列表中前 N_{tot} 个视点图像的 RGB 分量填充到立体图像的 RGB 分量中:

对于立体图像的(k,l)位置处 RGB 分量, 用映射表中对应数字的视点图像的相应 RGB 分量来填充. 例如, 对于立体图像第 1 行中的第 1 个 RGB 分量, 根据映射表, 其 R 分量用第 7 个视点图像的第 1 行第 1 个 R 分量来填充, 其 G 分量用第 6 个视点图像的第 1 行第 1 个 G 分量来填充, 其 B 分量用第 5 个视点图像的第 1 行第 1 个 B 分量来填充. 依此类推, 直到立体图像的所有 RGB 分量都被 N_{tot} 个视点图像的 RGB 分量所填充. 如图 3 所示:

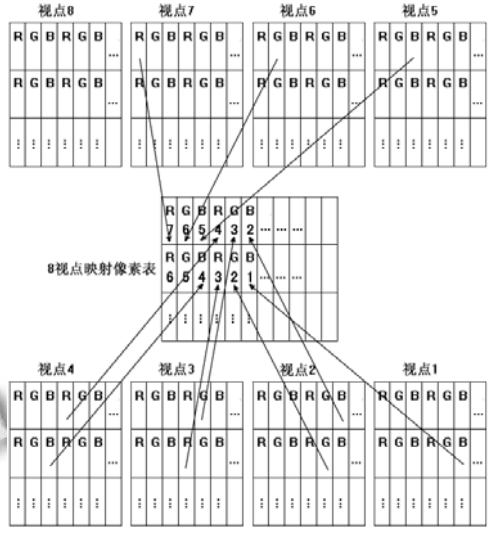


图 3　8 个视点的子像素映射示意图

4　视频播放方法

本文设计的视频播放器使用 OpenCV 与 MFC 进行开发. 首先, 使用 MFC 构建软件用户界面. 然后调用 OpenCV 的图像与视频处理函数对视频进行循环读取与图像融合处理, 在得到新的立体图像之后使用 OpenCV 将其按原始视频的帧率和分辨率大小进行缩放, 图像融合与播放. 在此过程中, 使用人眼跟踪算法对融合后的显示的图像进行重新融合与调整, 使其根据人眼位置投放对应视区的图像.

4.1 OPENCV 库与 MFC 结合使用

OpenCV 是 Intel 开发的一个开源的计算机视觉库, 功能涵盖图像处理、计算机视觉和模式识别等领域. OpenCV 实现了大量通用算法, 涉及到图像处理、结构分析、运动检测、摄像机标定、三维重建以及机器人等方面[8], 并有较高的运行效率, 可以在商业和研究领域免费使用. MFC 是一套应用窗体程序, 可以为程序提供控件、容器、菜单、工具栏、对话框等可用组件. 此外, 还可以利用其组件接口自定义开发所需的用户界面.

本视频处理系统使用 MFC 构建视频播放界面与用户菜单, 使用 OpenCV 的图像处理函数处理视频与摄像头影像, 同时, 使用 OpenCV 的 Haar 分类器进行人脸的检测与跟踪.

4.2 视频读取

使用 OpenCV 处理视频相关问题, 首先, 要用到的是读写视频文件的函数. CvCapture 结构包含从摄像机

或视频文件中读取影片的帧所需要的信息. 将视频文件名传递给 CvCreateFileCapture(), OpenCV 会导入并准备视频. 如果导入成功, 将返回指向已经初始化了的 CvCapture 结构的指针, 随后可以读入视频的帧.

创建一个有效的 CvCapture 结构之后, 使用 CvQueryFrame()函数处理读入的视频数据, 并将视频逐帧读入内存. 同时使用一个变量, 存储从 CvCapture 结构中读取的实际帧率.

4.3 合成图像

得到视频的帧信息后, 可使用上一节所述的图像融合算法对当前图像做融合处理. 然后将其存入新分配的图像结构空间中, 并将所有新图像按输入顺序组成帧序列.

4.4 视频播放与控制

使用 OpenCV 播放视频, 与使用它显示图像原理相同. 本文使用 CvShowImage()来逐帧显示融合后的新图像. 在上一步视频读取的操作中, 我们使用一个变量存储帧率. 在播放视频的过程中, 使用该帧率可以控制载入帧图像与显示帧图像的间隔时间, 将其调整为原始视频文件的帧率.

OpenCV 提供的 HighGUI 工具包不仅提供了一些简单的显示函数, 还包括一些图像和视频控制方法. 其中一个经常使用的就是滚动条, 可以方便的拖动视频从一帧跳到另外一帧. 但此功能目前限于 AVI 格式视频文件.

4.5 人眼跟踪

1) 立体视角的形成

在多视点立体显示图像的排列组合中, 以 4 视点为例, 合成图像的排列一般为 1,2,3,4,1,2, …, 1 到 4 分别代表视点数. 观看者在屏幕前的视区范围内任意移动, 双眼始终会观察到左右影像. 可看到 1-2,2-3,3-4…等左-右图像对, 这些左右图像对都属于左右视差. 观看者的双眼在这个范围内变化, 都会看到正的立体显示效果[9]. 然而, 当屏幕图像静止不动时, 立体视角存在一定的观看范围, 视点数越小, 立体视角越小. 因此, 要扩大观看移动范围, 本文采出一种人眼跟踪技术, 实时的根据人眼的移动, 调整屏幕图像像素排布方式. 使人眼在水平位置上移动时, 看到的始终是同一个视点图像投放的内容, 以此来扩大立体视角.

2) 人眼跟踪介绍

人眼跟踪功能是指, 当观众位置发生移动时, 使用摄像头实时捕捉用户的影像, 同时通过图像处理技术实时识别人眼区域并进行三维坐标计算. 以计算出的三维坐标为基础, 推算出对应的图像融合方式, 并将该方式按照自定义的通信协议发送给图像处理模块. 借以实现图像相对于人眼位置的实时调整. 该功能可以实现自动侦测人眼位置并投放 3D 影像, 以消除用户位置变化导致的串扰影像, 增大 3D 视角. 原理如图 4 所示:

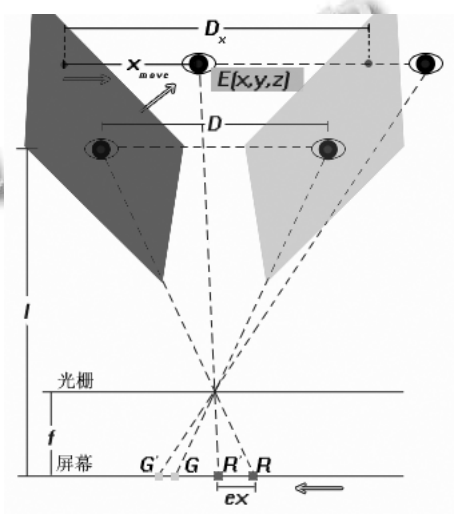


图 4 使用人眼跟踪调整立体显示原理图

设人的左(右)眼 3D 坐标为 $E(x,y,z)$, 设最佳观看区域:

$$z \in (600,750)$$

当人眼水平方向移动时, x 值发生变化. 当人眼前后移动时, z 方向值发生变化, 在最佳观测距离时, z 值为 l.

当人眼位于 $E(x,y,z)$位置时, 根据该点 z 值, 计算 E 点水平方向上菱形区域中心点间距:

$$D_x=(z-f)\times D/(l-f)$$

其中 D 为菱形区域中心点间距. 做取余运算, 设 n 为 临时变量, 表示间距的倍数:

$$n = (x + D_x/2)/(2 \times D_x)$$

要跟踪 E 点并使 E 点上的人眼可看到红色区域, 需要将菱形区中心点随着跟踪位置的变化而偏移:

$$X_{move} = x + D_x/2 - n \times D_X \times 2 + D_X/2 \times R$$

其中 R 为光栅透光比例. 如果($X_{move}<0$ && $X_{move}<-D_x$), 则反方向移动:

$$x_{move} = 2 \times D_x + x_{move}$$

然后, 根据菱形区偏移量与其对应的子像素的位

置关系, 换算出子像素移动的距离:

$$ex = -x_{move} \times f/(z-f)$$

f 为光栅相对于屏幕的放置距离. 根据 ex 值移动子像素, 使 E 点被红色菱形区包围, 跟踪完成, 如图 5:

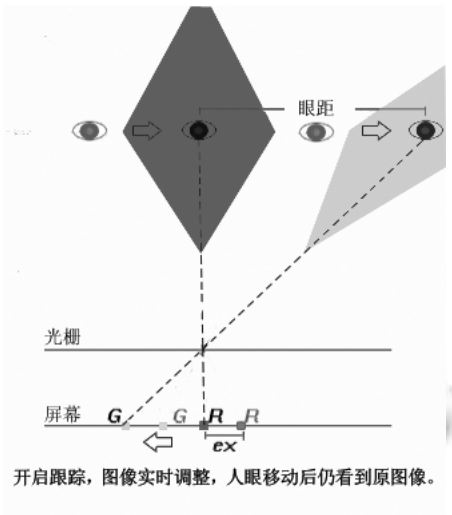


图 5　实时跟踪人眼调整立体显示示意图

在人眼的检测中, 本文采用了 Haar 型特征[10]构建特征空间, 这种类型的特征在人脸和人眼检测中有普遍的应用[11].

4.6 设计流程

本文提出的立体视频播放流程如图 6 所示.

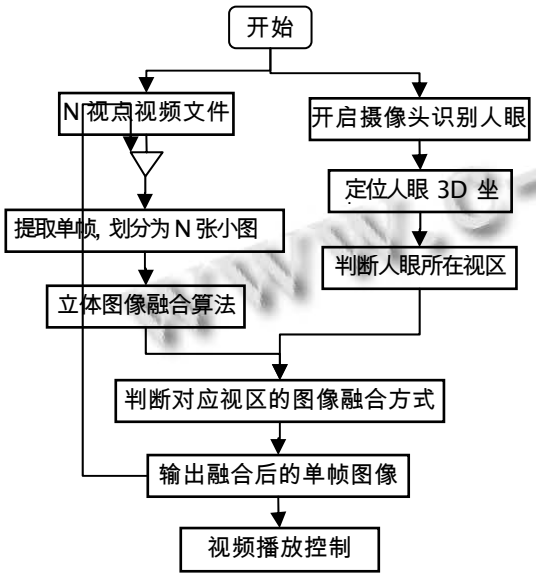


图 6　立体视频播放流程

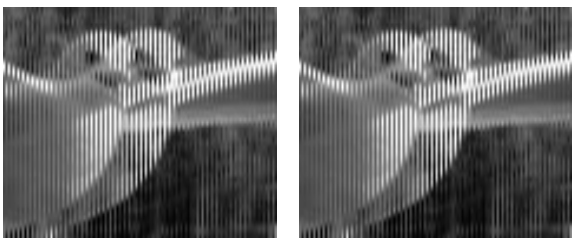
首先, 绘制视频处理系统的界面, 使用 MFC 构建视频播放界面与 UI 菜单. 当用户选择打开视频文件之后, 程序将使用 OpenCV 打开影片并准备读取, 同时开启摄像头准备识别人脸.

当影片被成功读入, 程序将对读入的每一帧做必要的处理, 即将一帧中的 N 视点画面, 分解成 N 幅单视点图像, 然后分别对每一幅单视点画面根据屏幕尺寸进行缩放. 完成后, 将这些单视点图像存储在一个数组中. 接下来, 使用上一节提到的图像融合算法, 合成一幅 N 视点裸眼立体图像, 并通过 OpenCV 的绘图函数显示在屏幕上. 至此, 对一帧的图像处理暂时结束. 依次循环读取视频中的每一帧, 并按上述流程处理, 即可在屏幕上显示一部完整的裸眼 3D 影片.

与此同时, 在摄像头开启后, 读取摄像头画面, 并识别人脸, 使用 OpenCV 的 Haar 分类器进行人脸的检测与跟踪. 在计算出人眼的三维坐标后, 在空间中判断出人眼所在的立体视区, 求解出该视区理论上应该对应的某个视点图像(假设为视点 x, $x< N$), 然后重新对图像进行融合计算, 使得合成后的图像显示 x 视点的视区覆盖人眼所在的位置.

5　实验结果

为了验证本文算法的有效性, 本文对一部 2 视点视频在立体播放器中进行了播放. 在播放过程中, 随机截取视频帧, 与原始 2 视点视频进行静态图像处理的结果进行对照. 对照结果表明, 两种合成方式结果一致, 如图 7 所示, (a)图是用静态图像利用图融合算法合成的结果, (b)图为立体视频播放器实时合成结果. 此外, 为了验证人眼跟踪的准确性, 本文也使用红绿图对跟踪效果进行了验证, 结果证明当观看者左右移动头部时, 始终能看到不变的红(绿)图像. 立体视频播放能看到明显立体效果, 合成速度快, 播放流畅.



(a)静态合成图　　(b)视频中的合成效果

图 7 两种实验合成结果

本播放器同时还实现了 3D 区域的实时缩放, 分区

3D 显示, 以及 2D/3D 兼容, 在实际立体视频演示过程中, 收到了良好的反馈.

6 结语

多视点自由立体显示技术突破了以往立体显示技术需要辅助视具的局限, 使观众能够在较大角度内的多个位置用裸眼自由清晰地感受到立体画面所带来的视觉冲击, 从而极大地推动了立体显示的发展和应用.

本文基于立体图像融合算法, 提出了一套基于 OpenCV 的多视点立体视频处理系统. 并在原有算法基础上, 新增了人眼跟踪功能, 有效地扩大了用户观看的视角, 用户可以在屏幕前自由的观看立体视频. 本系统运算速度快, 播放流畅无延时, 同时在实际观看过程中能观看到较好的立体效果.

本文提出的多视点立体视频合成方法, 不仅适用于多视点立体视频, 而且可以进一步演化应用于多媒体, 游戏, 电子地图等内容的显示. 从而扩展了立体显示器的应用领域.

参考文献

1 王元庆.基于 LCD 的自由立体显示技术.液晶与显示,2003, 18(2):116–120.

2 汪洋,王元庆.多用户自由立体显示技术.液晶与显示, 2009,24(3):434.

3 何赛军.基于柱镜光栅的多视点自由立体显示技术研究[学位论文].杭州:浙江大学,2009.

4 颜轲.数字图像三维立体显示算法研究与实现[学位论文].长沙:国防科技大学,2010.

5 Van Berkel C. Image Preparation for 3D-LCD. Proc. of SPIE, 1999, 3639: 84–91.

6 况海斌,徐成.基于柱镜光栅的自由立体显示图像融合算法. 计算机工程,2011,37(3):207–209.

7 潘春艳,杨青,李建军.New Image Fusion Algorithm for Auto-stereoscopic Display. Proc. SPIE. Beijing, China, 2013.

8 Bradski G, Kaehler A.于仕琪,刘瑞祯,译.学习 OPENCV 中文版.北京:清华大学出版社,2009.

9 王宝亮.基于 H.264 的多视点立体视频关键技术研究[学位论文].天津:天津大学,2010.

10 Viola P, Jones M. Rapid object detection using a boosted cascade of simple feature. IEEE Conf. Computer Vision and Pattern Recognition. Kauai, Hawaii,USA. 2001.

11 周瑾,王元庆,张兆扬,范科峰.一种面向立体显示的实时人眼检测方法.计算机应用与软件,2013,30(4):5–7.