

# 一种精确定位镜头边缘的方法

马春梅, 王绍棣

(南京邮电大学 计算机学院, 江苏 南京 210003)

**摘 要:**提出了一种针对 MPEG 视频的精确定位镜头边缘的方法, 结合视频非压缩域和压缩域中相关视频帧的信息来定位镜头边缘。首先利用解压后的 I 帧信息来粗略定位镜头边缘, 然后利用视频序列压缩域中的 B 帧的运动趋势来精确定位镜头边缘, 而不用对其解压缩。实验结果表明此方法能精确地检测到视频的镜头边缘。

**关键词:**视频检索; 镜头边缘检测; 压缩域

**中图分类号:** TN911.73

**文献标识码:** A

**文章编号:** 1673-629X(2007)07-0102-03

## A Method of Accurate Shot Boundary Detection

MA Chun-mei, WANG Shao-di

(College of Computer, Nanjing University of Posts & Telecommunications, Nanjing 210003, China)

**Abstract:** Shot boundary detection is the first step of video retrieval. A method of accurate shot boundary detection is proposed in this paper. First shot boundary is generally located by I frame, then it is accurately detected by the two sequential B frames. Experimental results prove that it is an efficient algorithm in shot boundary detection.

**Key words:** video retrieval; shot boundary detection; compressed domain

## 0 引言

基于内容的视频检索(Content-Based Video Retrieval, CBVR)是近年来视频检索领域研究的热点。视频检索<sup>[1]</sup>一般分为镜头边缘检测、镜头关键帧提取、情节检测及情节代表帧选取等环节, 其中镜头边缘检测是后面镜头关键帧提取及情节检测的基础与关键, 因此镜头边缘的精确定位是做好视频检索的第一步。

镜头(shot)是指摄像机一次从开机到关机所拍摄到的一段时间上连续的视频, 它是视频数据的基本单位。对于镜头的边缘检测有在非压缩域和压缩域中进行检测两大类方法。如文献[2]提出的只在非压缩域中先解压视频中的 I, B, P 各帧, 然后利用直方图的差值来检测镜头边界, 但算法复杂, 计算量大, 而且检测到的镜头的边缘往往不精确。文献[3]提出的设定自适应阈值的方法虽然比固定阈值的方法有好的检测效果, 但计算量往往较大, 而且在文中所采用的这种方法中它的优势得不到体现。文献[4]提出的在压缩域内利用 DC 图像来检索镜头边缘, 这种方法虽然计算量

相对较小, 但准确率必然有所下降。因此笔者结合非压缩域中镜头检测的准确性和压缩域中镜头检测的快速性提出了先对完全解压的 I 帧利用直方图差值的方法来粗略定位镜头边缘, 然后根据初步确定的镜头边缘的 I 帧与其前一个 I 帧之间的所有 B 帧的运动趋势来精确定位镜头边缘。实验表明, 这种利用非压缩域与压缩域相结合的方法具有良好的精确性。

## 1 粗略定位镜头边缘的算法

视频序列中的 I 帧利用帧自身的相关性进行压缩, 采用基于 DCT 的编码技术, 编码时不需要其他的帧作参考, 因此 I 帧本身含有丰富的信息, 可以用其来切分镜头。

直方图作为图像的一个重要特性, 在进行图像比较和检索中被广泛使用。两幅不同的图像一般情况下其直方图也会有较大差异。当镜头发生切换时, 切换前后两帧的直方图会发生较大变化。直方图分为灰度直方图和彩色直方图。灰度直方图只考虑图像的灰度特性, 一般将像素点的颜色分成 256 级。彩色直方图按照像素的色彩值进行计算分类。由于直方图只考虑图像帧的亮度特性, 抛弃色彩对图像的影响, 因此算法简单方便, 对于大多数视频都能达到比较好的效果, 有很高的检测效率, 因此文中利用相邻 I 帧的灰度直方

收稿日期: 2006-09-28

作者简介: 马春梅(1978-), 女, 山东威海人, 硕士研究生, 主要研究方向为基于内容的多媒体信息检索; 王绍棣, 教授, 博士生导师, 研究方向为多媒体信息检索技术。

图的差来初步确定镜头的边缘是在哪个 I 帧上。计算公式如下:

$$z(k, k+1) + \frac{1}{M} \sum_{i=1}^N |I_k(i) - I_{k+1}(i)| \quad (1)$$

其中  $M$  是 I 帧的总的像素数,  $N$  为灰度等级, 在这里取  $N = 256$ ,  $I_k(i)$  和  $I_{k+1}(i)$  分别是第  $k$  个 I 帧与第  $k+1$  个 I 帧的第  $i$  级灰度值。当直方图差值大于给定的阈值  $T$  时, 则认为在第  $k+1$  个 I 帧处切分镜头。文中  $T$  取 0.44。

通过这种对相邻 I 帧直方图的比较初步确定了镜头的边缘, 因为在视频序列中若不是出现速度急骤变化的镜头, I 帧的分布是均匀的, 大约每隔 12 至 15 帧插入一个 I 帧, 这样若镜头切分点发生在两个相邻 I 帧之间的 P 帧或 B 帧, 则这种定位在 I 帧的方法显然是不精确的, 因此要再利用切分点处的 I 帧与其前面的 I 帧之间的 B 帧的运动信息来精确定位镜头边缘。

## 2 精确定位镜头边缘的算法

对于 MPEG 压缩视频来说, 视频序列由 I, B, P 三种类型的帧图像组成, 它们之间的关系如图 1 所示。

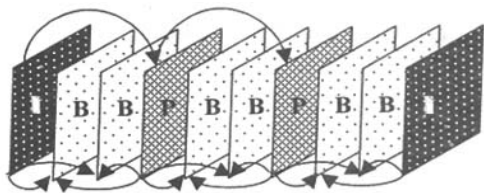


图 1 I, B, P 帧之间的关系

其中 I 帧利用自身的相关性进行压缩, 采用基于 DCT 的编码技术, 编码时不需要其他的帧图像作参考。I 帧内的宏块类型只有帧内宏块 (Intra, 此为宏块表示方法, 下同) 和帧内量化宏块 (Intra-A)。

P 帧是参考过去的 I 帧或 P 帧用运动补偿预测技术进行编码的图像, 如图 1 中的 4 号 P 帧是由 1 号 I 帧预测编码而得到的。P 帧内的宏块类型有以下几种: 前向运动编码宏块 (Inter-FD)、预测编码宏块 (Inter-FD)、前向运动宏块 (Inter-F)、帧内编码宏块 (Intra)、前向运动编码量化宏块 (Inter-FDA)、编码量化宏块 (Inter-DA)、帧内量化宏块 (Intra-A)、跳跃宏块 (skipped)。

B 帧可参考其前后的 I 帧或 B 帧图像进行预测, 也可以同时参考前后的 I 帧或 P 帧图像 (双向预测) 进行预测。B 帧内的宏块类型有以下几种: 双向运动宏块 (Inter-I)、双向编码宏块 (Inter-ID)、后向运动宏块 (Inter-B)、后向运动编码宏块 (Inter-BD)、前向运动宏块 (Inter-F)、前向运动编码宏块 (Inter-FD)、帧内编码宏块 (Intra)、后向运动编码量化宏块 (Inter-

BDA)、帧内量化宏块 (Intra-A)、前向运动编码量化宏块 (Inter-FDA)、双向编码量化宏块 (Inter-IDA)、跳跃宏块 (skipped)。

从 I, P, B 帧的宏块类型可以看出: P 帧中只有前向运动宏块, 而 B 帧中既有前向运动宏块又有后向运动宏块及双向运动宏块, 它含有丰富的帧运动信息, 因此可利用 B 帧的运动趋势来精确定位镜头边缘。对于发生镜头切换的两个相邻 I 帧之间的所有帧, 可能发生镜头切换的位置有以下三种, 如图 2 所示。

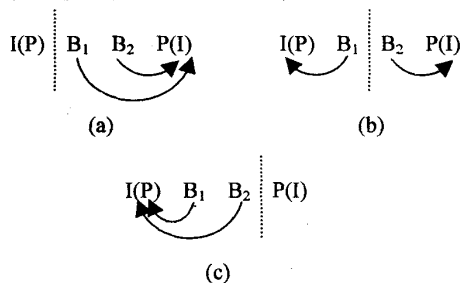


图 2 两相邻 I 帧中镜头切换的可能位置

图中虚线表示镜头转换的位置。图 2(a) 中镜头转换发生在两个连续 B 帧的第一个 B 帧  $B_1$  之前, 那么  $B_1$  帧和前面的 I(P) 帧相差很大,  $B_1$  和  $B_2$  帧中的宏块将大量采用后向补偿编码方式, 即两 B 帧中的宏块类型大部分为后向类型; 图 2(b) 中镜头切换发生在  $B_1$  和  $B_2$  之间, 那么  $B_1$  中的宏块将大量采用前向补偿编码方式,  $B_2$  中的宏块将大量采用后向补偿编码方式; 图 2(c) 中镜头切换发生在  $B_2$  之后, 则  $B_1$  和  $B_2$  中的宏块将大量采用前向补偿编码方式。

设  $f_{mn}$ ,  $f_{bmn}$ ,  $b_{mn}$  分别为前向预测宏块数 (包括前向运动块和前向编码块)、双向预测宏块数 (包括双向运动块和双向预测编码块) 及后向预测宏块数 (包括后向运动块和后向编码块),  $MB_{Amax}$  为帧内宏块总数。

定义 1: B 帧的前向和后向运动趋势 forward 和 backward 为:

$$\text{forward} = \frac{f_{mn} + f_{bmn}}{MB_{Amax}} \quad \text{backward} = \frac{b_{mn} + f_{bmn}}{MB_{Amax}}$$

定义 2: 第  $i$  和第  $i+1$  处的相邻两个 B 帧的前后运动趋势之差 diff 为

$$\text{diff}(i, i+1) = \text{abs}[\text{forward}(i) - \text{backward}(i)] + \text{abs}[\text{forward}(i+1) - \text{backward}(i+1)] \quad (2)$$

结论: 在粗略分镜头的第  $k$  个 I 帧及第  $k-1$  个 I 帧之间, 对于每两个连续的 B 帧求  $\text{diff}(i, i+1)$ , 设  $\text{diff}(m, m+1) = \max\{\text{diff}(i, i+1)\}$ , 则分以下三种情况来精确定位镜头边界:

(1) 若  $\text{forward}(m) < \text{backward}(m)$  且  $\text{forward}(m+1) < \text{backward}(m+1)$ , 说明  $B_m$  和  $B_{m+1}$  中大量宏块

为后向运动类型,符合图 2(a) 的情况,则  $B_m$  前的  $I(P)$  为前一镜头的最后一帧,  $B_m$  是下一镜头的第一帧。

(2) 若  $\text{forward}(m) > \text{backward}(m)$  且  $\text{forward}(m+1) < \text{backward}(m+1)$ , 说明  $B_m$  中大量宏块为前向运动类型,  $B_{m+1}$  中大量宏块为后向运动类型, 符合图 2(b) 的情况, 则  $B_m$  为前一镜头的最后一帧,  $B_{m+1}$  是下一镜头的第一帧。

(3) 若  $\text{forward}(m) > \text{backward}(m)$  且  $\text{forward}(m+1) > \text{backward}(m+1)$ , 说明  $B_m$  和  $B_{m+1}$  中大量宏块为前向运动类型, 符合图 2(c) 的情况, 则  $B_{m+1}$  为前一镜头的最后一帧,  $B_{m+1}$  后的  $P(I)$  是下一镜头的第一帧。

3 实验结果分析

图 3 中的(a)和(b)是截取的同一视频段(风光 1. mpg)中镜头分割效果的曲线图。曲线图(a)和(b)中值为 2 的点均是实际镜头分割点, 从图中可以看出截取的本段视频有 7 个镜头分割点, 共 8 个镜头。图(a)中值为 1.8 的点是仅利用式(1)得到的, 即只利用 I 帧分镜头的结果。可以看出与实际镜头相差帧数较多, 不精确。图(b)中值为 1.8 的点是利用式(1)与式(2)共同得到的, 即在利用 I 帧粗略分镜头后, 再加上根据相应分镜头的 I 帧前几个 B 帧的运动趋势来精确定位镜头边缘的。从图中可以看出与实际镜头边缘处相差无几, 很精确地实现了镜头定位。

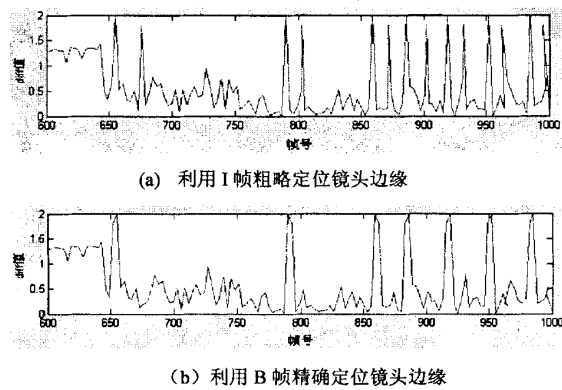


图 3 同一段视频镜头分割效果曲线图

如图 3, 视频段风光 1. mpg 中帧号为 803 的 I 帧和其前面相邻的帧号为 788 的 I 帧的直方图差值超过指定的阈值  $T$ , 利用公式(1)将镜头切分点粗略定位在序号为 803 的 I 帧上。再利用公式(2)分析这两个 I 帧之间的所有 B 帧的运动趋势之差, 得出第 790 帧为前一镜头的结束帧, 791 帧为后一镜头的起始帧。这两个相邻 I 帧中粗略定位处和精确定位处的帧图像如图 4 所示。

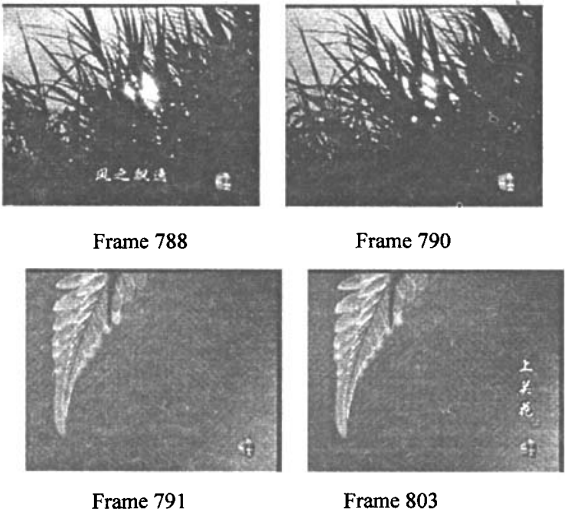


图 4 粗略定位处和精确定位处的帧图像

衡量镜头检测有查准率(precision)和查全率(recall)两个指标<sup>[5]</sup>。其中:

查准率 = 正确检测的镜头数 / (正确检测的镜头数 + 错误检测的镜头数)

查全率 = 正确检测的镜头数 / (正确检测的镜头数 + 漏掉检测的镜头数)

为检验文中的算法, 选取了几段旅游视频、动物视频、教学视频和风光视频来进行实验。实验环境为 VC6.0 系统, 视频段中帧尺寸大小为  $352 \times 240$  (pixel  $\times$  pixel), 帧率为 30 frames/s, 各视频段的长度及实验结果参见表 1。

表 1 实验结果

视频类型	长度(帧)	正确镜头数	错检镜头数	漏检镜头数	查准率(%)	查全率(%)
旅游 1	1913	16	0	1	100	94.12
旅游 2	1814	18	2	2	90.00	90.00
动物 1	2737	20	1	1	95.24	95.24
动物 2	2492	25	1	0	96.15	100
教学 1	3210	21	0	1	100	95.45
教学 2	2985	23	1	1	95.83	95.83
风光 1	2831	26	1	3	96.29	89.65
风光 2	1963	22	2	1	91.66	95.65

4 结束语

提出了一种利用在非压缩域和压缩域相结合的方法来精确定位镜头的边缘, 从实验结果看出, 这种方法比单独在非压缩域或压缩域中做都要好。实验中漏检的镜头主要是因为渐变引起的, 而且这种渐变在镜头边缘前后帧图像是相差无几的。错误的检测主要是由于一个镜头中光线亮度的突然变化引起的。对于这种前后相差不大的渐变的检测还不够理想, 因此这将是

(下转第 107 页)

```
key="hibernate.connection.provider"
value="NHibernate.Connection.DriverCon-
tionProvider"
/>
<add
key="hibernate.dialect"
value="NHibernate.Dialect.MsSql2000Dialect"
/>
<add
key="hibernate.connection.driver_class"
value="NHibernate.Driver.SqlClientDriver"
/>
<add
key="hibernate.connection.connection_string"
value="Server=localhost;initial catalog=nhibernate;In-
tegrated Security=SSPI"
/>
</hibernate>
```

配置文件中的参数具有的含义如表 1 所示。

表 1 NHibernate 配置文件参数表

参数	值
hibernate.show-sql	True 表示向控制台输出运行中产生的 SQL,用于调试目的
hibernate.connection.provider	表示使用指定的类来提供数据库连接缓冲池
hibernate.dialect	表示 NHibernate 方言的类名,可以让 NHibernate 使用某些特定数据库平台的特性,目前 NHibernate 支持 SQL Server, Oracle, MySQL, Firebird, Sybase, PostgreSQL 等数据库方言
hibernate.connection.driver-class	表示 ADO.NET 的驱动类,支持 SQL Server, Oracle, MySQL, OleDb, ODBC, Firebird 等驱动
hibernate.connection.connection-string	对应于 ADO.NET 的连接串

2.5 持久化对象操作

现在可以在代码中恰当简洁地使用 NHibernate。简化的版本如下:

```
//首先连接数据库
Configuration cfg = new Configuration();//创建配置类
cfg.AddAssembly("NHibernate");
```

(上接第 104 页)  
下一步研究的重点。

参考文献:

[1] 陆海斌,章毓晋,杨卫平.基于镜头和情节的视频非线性组织[J].计算机学报,2000,23(5):548-552.  
[2] Koprinska I, Carrato S. Temporal video segmentation: A survey[R]. Trieste: Institute for Information Technologies, University of Trieste, 2001.

```
//Configuration 对象会搜索装配件里的任何以 hbm.xml 结
尾的文件。还有其他方法显式加载映射文件,但这种方式是最
简单的
//创建会话工厂
ISessionFactory factory = cfg.BuildSessionFactory();
//打开一个会话
ISession session = factory.OpenSession();
//创建事物处理
ITransaction transaction = session.BeginTransaction();
//建立一个 Message 对象
Message msg = new Message ();
Message.Text = "My Message";
//保存 Message 对象到数据库
session.Save(msg);
//执行事务
transaction.Commit();
session.Close();
其他操作读者可以参照以上方法编写。
```

3 总 结

文中介绍了 NHibernate 和使用其进行开发的方法,实际开发中存在生成的 SQL 语句查询效率需要进一步优化等问题。但总的来说,使用 NHibernate 进行开发可以使程序结构清晰,易于维护,具有更好的扩展性,更容易开发可复用的组件。同时,减少了出错机会,提高开发效率。

参考文献:

[1] 贾新茹,须  德.对象/关系映射机制研究与实现[J].铁路计算机应用,2005(14):11-13.  
[2] 胡建华.基于 Hibernate 开发与数据库无关的系统[J].计算机与现代化,2005(9):50-53.  
[3] 宋汉增,沈  琳.利用 Hibernate 对象持久化服务简化 Java 数据库访问[J].计算机应用,2003(12):135-137.  
[4] 王立新,刘厚胜.一种对象/关系模型映射新方法的研究与设计[J].微机发展,2005,15(12):29-32.  
[5] Project: NHibernate[DB/OL]. 2005-10-19. <http://sourceforge.net/projects/nhibernate>.

[3] Vaconcelos N, Lippman A. Statistical Models of Video Structure for Content Analysis and Characterization [J]. IEEE Trans on Image Processing, 2000, 2(4):240-254.  
[4] YEO B-L, LIU B. Rapid scene analysis on compressed video [J]. IEEE Transactions on Circuits and Systems for Video Technology, 1995, 5(6):533-544.  
[5] 耿玉亮,须  德.视频镜头边界检测的统一策略[J].中国图象图形学报,2005,10(5):650-655.