# 基于 MPEG-7 的视频检索系统的设计与实现①

郑 烇 肖碧宇 (中国科学技术大学 自动化系 安徽 合肥 230027)

**摘 要：** MPEG-7 为视频信息提供了一种标准化的描述方式，使得我们能够以文本的方式标注出视频文件的标题、作者、日期以及内容等各种属性，方便了在多种应用中交换视频的信息。设计和实现了一个视频检索系统：语寻，该系统利用已有的视频处理工具对视频文件进行处理，生成其 MPEG-7 描述文件，然后对视频的 MPEG-7 文件建立全文索引；当用户通过系统的 Web 界面输入查询关键字，便能够迅速的检索到感兴趣的视频文件。同时，系统采用分布式架构，具备良好的可扩展性，能够支持海量视频信息的检索。

**关键词：** MPEG-7; 视频检索; Lucene; 分布式; 系统设计

## Design and Implementation of a Video Retrieval System Based on MPEG-7

ZHENG Quan, XIAO Bi-Yu

(Department of Automation, University of Science and Technology of China, Hefei 230027, China)

**Abstract:** MPEG-7, which provides a standardized description for a video, allows one to mark out various properties of a video such as the title, author, dates and content to help facilitate the exchange of video information among various applications. This paper discusses the design and features of a video retrieval system, YuXun. It first uses existing video processing tools to generate the MPEG-7 description file for videos and then establishes full text index for the MPEG-7 documents. Therefore, the user can enter a query string in the system's Web interface and retrieve interested video files quickly. At the same time, the system uses a distributed architecture, which has high scalability, to support mass video information retrieval.

**Keywords:** MPEG-7; video retrieval; Lucene; distributed system; system design

## 1 引言

随着信息技术的发展和个人摄像设备的普及，人们产生的视频数量迅速增长，海量的视频信息对存储技术和检索技术都提出了新的挑战，许多人开始研究相关问题。视频检索技术是信息检索领域最热门的方向之一。传统的视频检索技术分为两种，一种是基于视频的标题、描述等元信息的检索；另一种是基于视频内容和图像的检索[1]。基于元信息的视频检索技术在互联网上使用的比较多，如 Google、百度等搜索引擎提供的视频搜索，以及 youtube、优酷、土豆等视频分享网站提供的站内视频搜索等。这是由于 Web 上的视频大多不是单独出现的，一般都包含在网页中，而网页里还包含了视频内容的描述信息，这就为搜索引擎索引视频提供了途径。基于元信息的视频检索技术无法充分利用视频本身所包含内容的信息，因而检索功能有一定局限。特别地，当用户关心的不只是找到某部视频，还想找到视频中的内容的时候，基于元信息的视频检索技术就无能为力了。近年来发展比较迅速的视频检索技术是基于视频内容的，它利用计算机图像、视觉和多媒体数据库管理领域的技术，通过对视频内容进行分析，对一段视频做场景和镜头两个层次上的分割，提取视频的纹理、形状、色彩等低层

① 基金项目:国家高技术研究发展计划(863)(2008AA01Z147);安徽省高校省级自然科学研究重点项目(KJ2009A152)
收稿时间:2010-01-15;收到修改稿时间:2010-03-19

次特征，最后通过特征的匹配来检索视频。这方面的代表有 IBM 的 Multimedia Search and Retrieval System[2,3]和都柏林城市大学的 Fischlar 系统[4]等。目前，这类系统仍然存在检索速度慢、精确度不够等问题。

目前，视频检索技术逐渐向视频内容的低级特征(颜色、形状、纹理等)和高级特征(语义信息)相结合的方向发展。MPEG-7(多媒体内容描述接口)的提出，提供了一种通用的、可扩展的描述视频内容的标准方式，它使用结构化的文本的形式来描述视频内容[5]，通过对 MPEG-7 文件进行分析，可以将文本处理技术引入视频处理中，这种方式比传统的图像处理技术更高效，并且可以处理高级语义信息。

本文第 2 节介绍相关技术，第 3 节描述语寻的总体架构以及各个模块的设计，第 4 节给出结论以及未来的工作。

## 2 相关技术

### 2.1 MPEG-7

MPEG-7 全称为“多媒体内容描述接口”，由动态图像专家组于 1998 年 10 月提出，为各类多媒体信息提供一种标准化的描述，这种描述将与内容本身有关，并且可以同时描述低级特征和高级特征。MPEG7 文件本身是一种 XML 格式的文件，可以由一些视频处理软件生成。

MPEG-7 规定了一套标准描述子(Descriptors)来描述各种多媒体信息，并且预先定义了描述子的结构以及它们之间的关系[6]。与视频相关的常用描述子分为颜色(Color)，质地(Texture)、形状(Shape)和动作(Motion)等类型，每一类包含若干种描述子，以不同的方式和结构来描述视觉信息。同时，MPEG-7 还提供了描述定义语言，用户可以自定义新的描述模式。在实际的应用中，我们可以根据需要选取一部分或全部描述子进行处理。

### 2.2 Lucene

Lucene 是一个开放源代码的全文检索引擎工具包，最初由 Doug Cutting 开发，现在是 Apache 软件基金会下的一个子项目。Lucene 原本使用 Java 开发，是一个完全面向对象的全文检索引擎框架，可以方便的嵌入目标程序中，以实现全文检索功能；同时，开发人员也可以很容易对 Lucene 进行扩展，以定制自己的检索功能。目前 Lucene 已经被移植到.NET、Python 等多个平台，获得了广泛的应用。

Lucene 包含文本分析引擎、索引引擎和查询引擎。Lucene 的发行版包含了英文和德文的分析引擎，其它语言如中文，可以按照 Lucene 提供的文本分析接口，方便地打造各种语言的分析引擎。Lucene 使用的是倒排文件索引方式，其索引文件的格式独立于应用平台。一份 Lucene 的索引可以由一段或多段索引组成，Lucene 对新加入的文件建立小段的索引，当文件数目超过一定的阈值(例如 10)，再将其合并到大的索引段中，通过这种方式，在配置较低的机器上也能高效地索引大量的文件。Lucene 的查询功能非常的丰富，不但支持常用的多关键字查询，还支持模糊查询、范围查询、通配符查询等多种查询方式，其自身提供查询解析器，可以处理复杂的查询语法。

## 3 视频检索系统的设计与实现

语寻是一个结合了视频的低级特征和高级特征的检索系统，它既支持通过关键字的方式检索视频，也支持通过样例图片来检索相似的视频(Query-by-example)。语寻通过分析、索引视频的 MPEG-7 信息，能够在大量视频中快速定位到视频内部的相关片段，具有检索速度快、精度高的特点。

### 3.1 系统整体框架

语寻视频检索系统将视频的 MPEG-7 文件、视频快照、引用位置等信息存储在数据库中，在中心管理节点的控制下，对 MPEG-7 等信息进行解析，将解析出来的数据传输到索引节点建立索引，再将索引部署到查询节点。用户通过 Web 服务器提交查询请求，系统首先在各个查询节点分别查询，通过收集节点合并查询结果，然后返回给 Web 服务器，最终显示给终端用户。

系统整体框架如图 1 所示，各模块的具体介绍如下：

(1) InfoDB：视频信息存储节点，负责存储视频的元信息、ID，解析视频的 MPEG-7 文件，给系统管理员提供视频索引状态的查看和修改功能。

(2) Manager：核心管理节点，负责管理 Indexer 节点，searcher 节点，以及 Collector 节点；负责接收 InfoDB 发来的索引操作命令，启动索引的添加、删除以及更新等操作；负责维护全局 df（Document Frequency，文档频率），管理索引的部署策略等。

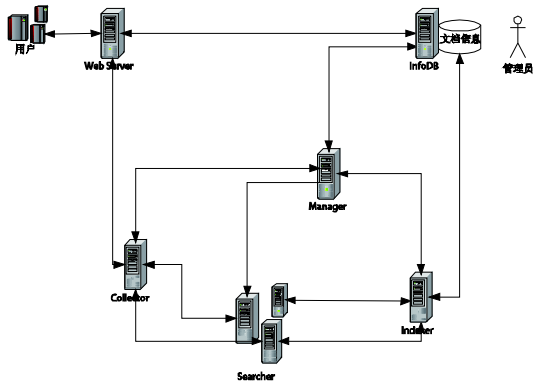


图 1　语寻视频检索系统整体架构

(3) Indexer：索引节点，负责接受 Manager 节点发来的索引建立和部署指令；负责从 InfoDB 提取对 MPEG-7 文档解析得到的文档对象，生成索引并计算 df，将 df 发送给 Manager 节点；负责将索引文件发送给指定的 Searcher 节点。

(4) Searcher：查询节点，负责接受来自 Collector 的查询请求，对关键字进行查询，然后将查询结果返回给 Collector；协助维护全局 df，以提供对查询结果的评分；负责索引文件的存储，迁移，和删除等操作。

(5) Collector：查询结果合并节点，负责接受 Web 服务器的查询请求，然后向各个 Searcher 节点发送查询指令；负责合并 Searchers 返回的查询结果，排序后，将结果返回给 Web 服务器。

(6) Web Server：视频检索系统的 Web 服务器，提供用户查询界面，接收用户的查询请求并转发给 Collector 节点，从 Collector 节点获取查询结果之后，从 InfoDB 中获取视频的信息并将其显示给用户。

### 3.2 视频检索流程

#### 3.2.1 视频索引流程

语寻在提供用户查询功能之前需要对视频进行索引，管理员可以通过 InfoDB 的管理界面选择要索引的视频列表，点击“索引”按钮后，系统便开始执行添加索引流程，在 Manager 的控制下，Indexer 从 InfoDB 获取 MPEG-7 信息，建立倒排索引，最后部署到 Searcher 上。索引流程如图 2 所示，具体为：

① InfoDB 通知 Manager 一个添加视频索引的任务，其中包含需要添加的 VideoIDs 数组。

② Manager 接收到任务后，把 VideoIDs 数组添加到索引添加队列 AddVideoQueue 中。

③ Manager 上的一个监视线程从 AddVideoQueue 队列中取出一定个数的视频 VideoIDs 数组，给 Indexer 发送添加视频索引的任务。

④ Indexer 接到通知后，根据 VideoIDs 从 InfoDB 中取回对应的视频及其片段的 MPEG-7 信息，以视频片段为单位开始建立索引。

⑤ Indexer 生成好索引后，通知 Manager 索引完成，并发送 df 信息给 Manager。

⑥ Manager 更新全局 df+=df，计算部署策略，然后将索引部署的目标 Searcher 地址发送给 Indexer。

⑦ Indexer 根据部署策略，通知指定的 Searcher 将要传送视频索引，通知成功后，开始建立一个 socket 连接，传送视频索引。

⑧ Searcher 接收索引文件，并将其与本地索引合并完成后，通知 Indexer 索引部署成功。

⑨ Indexer 接收到 searcher 的成功接收索引的通知后删除本地临时的视频索引，之后通知 Manager 部署成功。

⑩ Manger 删除 AddVideoQueue 队列中的已部署的视频 VideoIDs，通知各个 Searcher 更新全局的 df， 最后通知 InfoDB 对应的视频 VideoIDs 已经成功完成部署。

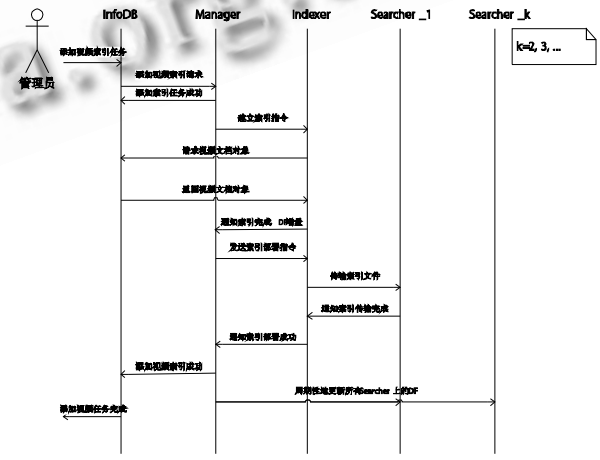


图 2　视频索引流程图

#### 3.2.2 视频查询流程

语寻的查询页面非常简洁，中间有一个文本输入框用于键入查询关键字，文本输入框下面还有一个文

件上载控件可以上传样例图片，点击“查询”按钮，Web 服务器将查询请求转发给 Collector，Collector 再向各个 Searcher 节点查询，并将结果合并后返回给 Web 服务器，由 Web 服务器以合适的方式显示给用户。查询流程如图 3 所示，具体为：

① 用户通过搜索框输入查询关键字，并上传一张查询图片(可选)，点击“搜索”按钮后产生查询动作发送到 Web Server 上，触发该动作的也可能是用户在查询结果中翻页。

② Web Server 提取用户上传图片的各个特征描述子信息，结合查询关键字形成查询请求发送给 Collector，并根据分页的要求指明返回第 start 条至第 start+count-1 条搜索结果。

③ Collector 从 Manager 获取所有 Searchers 的地址，并将查询请求发送给 Searchers。

④ Searchers 接收到查询请求，从本地查询，将匹配结果集中的第 1 条至第 start+count-1 条查询结果的 ids 及其对应的 scores 返回给 Collector。

⑤ Collector 收集 Searchers 返回的结果，对其进行合并、排序，然后将第 start 条至第 start+count-1 条查询结果 ids 返回给 Web Server。

⑥ Web Server 接收到 Collector 发送来的查询结果 ids，然后向 InfoDB 请求相应的视频信息。

⑦ InfoDB 接到 Web Server 的请求，将视频信息发送给 Web Server。

⑧ Web Server 接收到视频信息后，将其显示给用户。

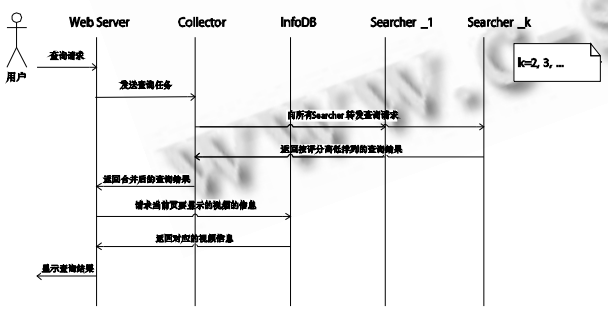


图 3　视频查询流程图

### 3.3 系统各模块实现

#### 3.3.1 InfoDB

视频在加入检索系统之前，需要先经过视频处理工具的处理，这些工具会对视频按场景进行分割、提取关键帧及其特征描述子信息，还可以添加人工标注，这些信息最终包含在一个 MPEG-7 文件中（以及额外的关键帧截图文件）。InfoDB 负责解析并存储这些 MPEG-7 文件内容以及关键帧截图，这些信息主要是一些结构化的数据，可以使用关系型数据库来存储（例如 MySQL），目前，主要信息存储在了两张表中：视频信息表和片段信息表。视频信息表以视频为单位，保存了每部视频的标题、长度、发布时间、内容描述、索引状态等数据，该表结构如下：

表 1　视频信息表

| 列名称 | 数据类型 | 说明 |
|---|---|---|
| VideoID | String | 影片 ID |
| Title | String | 名称 |
| Length | Time | 影片长度 |
| PublishTime | Date | 发行日期 |
| Content | String | 内容描述 |
| IsIndexed | Boolean | 是否被索引 |
| …… | …… | …… |

片段信息表则以片段为单位，保存了视频经过分割后产生的每个片段的信息，如片段所属视频、偏移量、内容描述、关键帧截图以及各个特征描述子的数据。该表结构如下：

表 2　片段信息表

| 列名称 | 数据类型 | 说明 |
|---|---|---|
| SegmentID | String | 视频片段 ID |
| VideoID | String | 所属影片 ID |
| Offset | Int | 片段偏移量（单位为 ms） |
| Length | Time | 片段长度 |
| Content | String | 内容描述 |
| KeyFrame | BLOB | 关键帧截图（jpg 格式） |
| EdgeHistogram | String | 边缘直方图描述子 |
| ScalableColor | String | 扩展颜色描述子 |
| …… | …… | …… |

InfoDB 有自己的管理界面，系统管理员可以查看和修改视频索引状态。当管理员选择添加一部视频索引时，InfoDB 将对应视频的 VideoID 包装为一个任务对象发送给 Manager 节点，通知其进行索引。

#### 3.3.2 Manager

Manager 节点是视频检索系统的核心管理节点，它接收 InfoDB 发来的索引操作命令，然后启动索引

的添加、删除以及更新等操作；同时需要维护全局 df (Document Frequency，文档频率)，管理索引的部署策略等。

因为索引分布在多个 Searcher 节点上，而我们又希望各个节点上的查询结果可以比较、整合，所以需要维护一份全局 df。df 是一个<word-count>对的集合，其中 word 是单词，count 是包含该单词的文档的数目，所以可以使用 HashMap 来存储 df 信息，其中 word 是 key，类型为 String，count 是 value，类型为 int。同时，Manager 还需要在硬盘上保存 df 的信息，以方便在意外宕机后能够恢复全局 df 信息。

由于索引视频信息需要消耗较长的时间，InfoDB 向 Manager 发送添加视频索引的请求后，不等待索引任务完成就立即返回，而 Manager 将来自 InfoDB 的索引请求保存在一个任务队列 AddVideoQueue 中。Manager 上有一个任务处理线程，它周期性地检查 AddVideoQueue，当发现 AddVideoQueue 不为空，就从中取出一条记录，将 VideoIDs 发送给 Indexer，通知其开始建立索引。Indexer 建立好索引后会将新建索引的Δdf 通知 Manager，Manager 指示 Indexer 将新增索引部署到目标 Searcher 节点，然后再向所有 Searcher 节点发送 df 更新消息。

### 3.3.3 Indexer

Indexer 节点负责对视频的 MPEG-7 信息建立索引，它接收来自 Manager 的索引命令，其中包含需要索引的 VideoIDs，然后从 InfoDB 中获取对应视频及其所有片段的 MPEG-7 信息。由于这些数据都是存储在关系型数据库中，可以通过 JDBC 连接到远程的数据库服务器，获取视频信息数据，再在本地转换为对应的格式。

Indexer 的索引引擎基于开源工具 Lucene，它具有索引效率高、方便扩展的特点，并且支持按域索引，能对待索引数据的各个部分进行不同的处理。Indexer 索引视频的基本单位是片段（即查询返回的结果是视频片段），我们对视频片段 MPEG-7 信息的长度、内容、各个描述子以及所属视频的名称等不同部分分别创建各自的域，这样在查询的时候可以按域进行检索，有利于提高查询精度。不同的域处理方式也有所不同，例如内容描述这些文本类型的数据，需要首先经过分词器，将其切分为一个个的词，同时记录这些词出现的位置；我们将相同的<词—位置>对归结到一起，以词为索引，后面是记录其出现的位置列表，这便构成了视频信息的倒排索引，倒排索引的数据结构使得在一个文档集合中检索包含特定单词的文档的速度变得非常快。对于各个特征描述子，它们都是数值类型的数据，将在后面的查询过程中用于计算视频片段的相似度，只需要存储即可。

Indexer 完成索引任务后，需要通知 Manager，Manager 根据当前的索引部署状况计算出最佳部署节点（可能是包含索引数量最少的 Searcher），并将索引部署的目标节点地址发送给 Indexer。Indexer 创建一个 socket 连接到目标 Searcher，然后将索引文件传输到该 Searcher 节点。

### 3.3.4 Searcher

Searcher 节点接收来自 Collector 的查询请求，并将查询结果返回给 Collector。来自 Collector 的查询一般包含了若干个关键字以及特征描述子(如果上传了样例图片的话)，Searcher 需要读取倒排索引文件，在词汇表中找到每一个关键字，并且获取其对应的倒排列表，倒排列表中记录的是包含对应关键字的文档编号，获取了这些数据，再结合 Searcher 上保存的全局 df 信息，便可以计算出每个文档与查询关键字的相似度，如果查询中包含特征描述子，还需要计算其与索引中的对应特征描述子的相似度，文档的相似度综合了关键字和各个特征描述子的相似度，最后对相关文档按相似度从高到低排序，返回前面若干份文档给 Collector 即可(具体数量由 Collector 在查询指令中指明)。

Searcher 的查询引擎也是基于 Lucene，但修改了其 df 模块。Lucene 中实现索引查询功能的是 IndexSearcher 类，该类中的 DocFreq 方法获取指定单词的本地 df 值，由于实际需要的是全局 df 值，因而对 IndexSearcher 进行了改造，重写了 DocFreq 方法。Searcher 使用 HashMap 在本地记录了一份全局 df，它是 Manager 上全局 df 的副本，并且保持同步，一旦 Manager 上的全局 df 发生变化，Manager 便会将 df 增量向所有 Searcher 发送，Searcher 根据 df 增量更新副本。在重写的 DocFreq 方法中，直接在本地的全局 df 副本中查找指定单词的 df 值并返回，由于 HashMap 的查找效率很高，修改后的 DocFreq 方法也更快。

### 3.3.5 Collector

Collector 节点负责接受 Web 服务器的查询请求，向各个 Searcher 节点发送查询指令；然后合并 Searchers 返回的查询结果，排序后，将结果返回给

Web 服务器。

Collector 接收来自 Web Server 的查询请求，请求中包含查询关键字、特征描述子以及所需的结果的区间(由于 Web Server 需要对结果进行分页显示，可能取排序结果中间的某一些视频片段)。Collector 再将查询请求发送到所有 Searcher 上，并且指明返回结果的总数，Searcher 返回的是相关视频片段的 SegmentID 及其 score，并且结果是按照 score 从高到低排序的。Collector 收集了所有 Searcher 返回的结果后，需要对这些结果按照得分高低进行归并，归并的算法如下(假设 Web Server 需要第 begin 条至第 end 条结果，Collector 需要合并 k 个 Searcher 的结果)：

```
result = a queue of segment_ids ordered by ascending score, initialized empty
sq = a heap of <score, segment_id, searcher_no>, initialized k segment_ids with max scores of k Searchers's result
while (length(result) < end && !empty(sq))
    <score, id, no> = pop(sq)
    push_back(result, id)
    if searcher no's result is not empty, remove the first one and put it into sq
return the result[begin, ..., end] to Web Server
```

图 4　查询结果合并算法

### 3.3.6 Web Server

Web Server 提供了用户接口，接收用户的查询输入，显示对应的查询结果，并且提供了视频播放界面。Web Server 主要基于动态网页技术 JSP 构建，同时使用了 JavaScript 技术来提供丰富的显示效果。每条查询结果都包含了视频快照、标题、日期等信息，如果用户将鼠标移动到视频快照上，还能在浮动窗口中显示该视频片段的内容描述。如果用户检索时上传了样例图片，Web Server 还会从图片中抽取各个特征描述子信息，并将其包含在查询请求中。

由于与查询相关的视频片段可能很多，在一个页面中显示所有的相关视频片段会使加载页面消耗的时间很长，同时也不方便用户查看，所以需要对查询结果进行分页显示。用户的第一次查询会显示第一页的结果(默认一页包含十条视频信息)，如果结果总数超过十条，会在页面下方显示翻页按钮。分页显示的实现方式有两种：缓存或者重查，考虑到结果集一般很大，当并发用户数很大的时候会占用 Web Server 较多的内存，实际实现时使用了重查的方式。每次用户点击翻页按钮时，Web Server 会计算出下一页将包含的显示结果范围，然后将查询请求和需要的结果范围发送给 Collector，Collector 合并各个 Searcher 上查询结果之后，再将指定范围的结果返回给 Web Server。

每次用户查询或者翻页，Web Server 都会从 Collector 获取将在当前页显示的视频结果的 SegmentIDs，到 InfoDB 中读取对应的视频描述信息，再按照指定的格式显示在用户的浏览器上。

## 4　结束语

本文基于 MPEG-7 标准，设计和实现了一个视频检索系统：语寻。该系统通过对视频的 MPEG-7 信息建立全文索引的方式，实现对视频内容的高效检索。由于利用了 MPEG-7 标准对视频内容的低级特征和高级特征的强大描述能力，语寻能够综合多种方式对视频进行检索，提高了检索的准确度。考虑到视频数据快速增长的现状，语寻采用分布式架构，具备良好的可扩展性，以处理海量视频信息的检索任务。

今后的工作是研究针对不同用户的个性化视频检索方案，考虑利用用户的评价反馈来改善检索结果的准确度，同时提高系统的容错能力和稳定性。

## 参考文献

1 Baeza-Yates R, Ribeiro-Neto B. Modern information retrieval. Addison-Wesley Harlow. England. 1999.

2 Amir A, Argillander J, et al. IBM Research TRECVID- 2005 video retrieval system. TRECVID Workshop. Washington DC. 2005.

3 Campbell M, Haubold A. IBM research TRECVID-2006 video retrieval system. TREC Video Retrieval Evaluation Proceedings, 2006.

4 Lee H, Smeaton A. The Fischlar digital video recording, analysis, and browsing system. Proc. of the RIAO 2000-Content-based Multimedia Information Access. 2000.

5 Martinez JM. MPEG-7 Overview (version 10). ISO/IEC JTC1/SC29/WG11. 2002.

6 MediaLab S. MPEG-7 White Paper. Outubro. 2003.