文章编号:1671-4598(2019)06-0231-05 DOI:10.16526/j.cnki.11—4762/tp.2019.06.050 中图分类号:TP391.3 文献标识码:A

基于深度学习的视频检索系统设计与实现

姚锦江[1], 程允权[2]
(1. 华南理工大学 广州学院, 广州 510800; 2. 广东经传多赢投资咨询有限公司, 广州 510800)

摘要: 针对摄像头产生的海量视频信息, 检索工作需要花费大量的人力、物力以及时间成本问题, 分析发现传统检索的功能大多都基于文本关键词, 对视频内容的覆盖率低且容易依赖于相关工作人员的主观性; 提出如何应用传统的机器视觉技术以及深度学习技术去构建一个高效的视频检索系统; 创新点在于从视频帧图像内容的角度去充分发掘其中的信息, 其信息挖掘的过程无需人工干预, 从而提高了信息利用率。

关键词: 视频信息; 视频检索; 机器视觉; 深度学习; 信息挖掘

Design and Implementation of Video Retrieval System Based on Deep Learning

Yao Jinjiang[1], Cheng Yunquan[2]
(1. Guangzhou College, South China University of Technology, Guangzhou 510800, China;
2. Guangzhou Jingzhuan Information Technology Co., Ltd., Guangzhou 511400, China)

Abstract: In view of the huge amount of video information generated by the camera, the retrieval work needs a lot of manpower, material resources and time cost. The analysis shows that most of the traditional retrieval functions are based on text keywords, and the coverage of video content is low and easy to rely on the subjectivity of the relevant staff. This paper proposes how to use traditional machine vision technology and deep learning technology to build an efficient video retrieval system. The innovation lies in fully exploring the information from the perspective of video frame image content. The process of information mining does not need manual intervention, thus improving the utilization rate of information.

Keywords: video Information; video retrieval; machine vision; deep learning; information mining

0 引言

随着通信和多媒体技术的迅速发展, 数字视频数据在互联网上的传播越来越广, 检索和浏览海量的数字视频数据成为一个十分困扰的问题, 采用传统的人工描述方法存在很大的弊端[1-2]。目前, 国外的 Google 视频分析云可以通过 HTTP 协议去提供 restful 风格视频分析 API[3]。国内的腾讯优图天眼利用计算机视觉以及深度学习技术去实现高精度且实时的人物重识别任务[4], 北京大学彭宇新教授及其研究团队提出了视觉注意力驱动的图像视频分类与检索研究、监控视频语义感知和服务系统研发及应用示范[5-7]。

在现有的视频检索技术上, 缺少结构化的描述, 从而使得检索时难以用精确的语言来表示它的特征[8]。为此, 本文通过对视频关键帧的识别、特征向量的提取, 从而把无结构化的视频帧图像中得出有用的信息以便于检索。用户在使用本系统的时候, 只需要投入较少的人力即可大大地提高视频的检索效率[9-11]。

1 系统架构及体系

本系统主要由 4 大模块组成, 分为镜头分割引擎、相似帧定位检索引擎、目标检索引擎以及人脸检索引擎, 它们共同构成了后端系统。整个后端系统, 模块间只需纵向地对上层的网关接口层负责, 而无需横向地和同级的模块进行交互。模块与上层的网关接口之间使用基于 Protobuf3 的 GRPC 框架进行通信。底层的数据库引擎使用 PostgreSQL 数据库、文件存储文件系统为 ext4。在宏观角度上, 本系统采用了 B/S 架构, 与用户的交互主要发生在 Web 浏览器上。前端 UI 的交互界面在 MVVM 架构上使用 ES6 和 VueJS 进行开发。系统架构如图 1 所示。

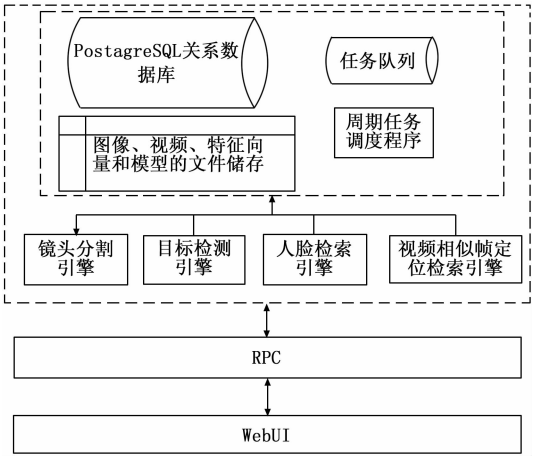

图 1 系统架构图

1.1 模块微服务

本系统的开发语言主要基于 C++语言, Web 处理层

收稿日期:2018-12-06; **修回日期**:2018-12-29。
基金项目:华南理工大学广州学院教育教学改革项目。
作者简介:姚锦江(1984-),男,广东广州人,硕士研究生,实验师,主要从事信息处理技术、实验室信息化建设方向的研究。

基于 Go 语言，前端使用 ES6 语言，还有部分涉及传统机器学习的功能则使用 Python 开发。多语言的使用可以方便去发挥他们不同的长处，更好地去利用已有的资源进行系统开发，但多语言的使用会使得模块间的通信调用变得复杂化。因此本系统利用 Kong 作为 API 网关、GRPC 作为 RPC 框架、以 HTTP2 为底层的 Protobuf 数据传输格式去搭建本系统的微服务架构。这样既可以帮助开发者屏蔽跨语言模块间远程调用的底层的复杂度，同时大大地降低系统模块间的耦合度，提高系统的可扩展性[12-14]。

1.2 任务队列

在系统的运行中，涉及到很多需要长时间计算的任务，例如视频帧切割、关键帧探测、人脸分类器训练等。采用传统的同步方式去处理任务，必然会导致系统的整体卡顿，从而影响用户体验。为了高效的利用硬件资源而又兼顾系统的稳定，在 ZeroMQ 的基础上设计了系统内统一的任务队列，如图 2 所示。在队列的角度来看，用户的远程调用请求即是生产者生产的内容，调度器会将其放入到等待任务队列中去等待。当系统模块处理完任务后，会继续去执行等待队列中的任务。通过任务队列，系统就可以实现异步任务处理的功能。

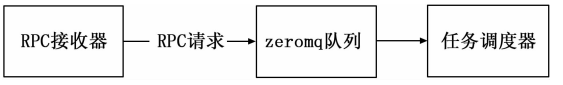

图 2 队列处理流程图

1.3 前端 UI 架构

本系统的前端 UI 界面使用 HTML5 与 CSS3，其中的前端逻辑层使用 ES6 语言编写，设计模式是基于 Vue.js 的 MVVM 设计模式，利用 Vue.js 和 vue－router 去创建单页应用，其中 vue－router 组件负责提供前端页面的路由功能。在这基础上，实现了前端页面路由无刷新跳转，提示用户体验。由于前端 UI 项目经常需要处理复杂的状态管理，当应用遇到多个组件共享状态时，传统的传参数据流动的方式会破坏视图对状态的单一依赖性，导致多层嵌套的组件的开发将会非常繁琐，并且对于兄弟组件间的状态传递无能为力，因此引入 Vuex 作为前端 UI 状态管理的组件，如图 3 所示。

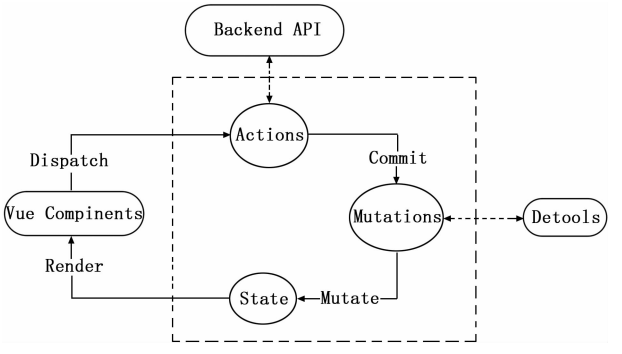

图 3 基于 Vuex 的数据流动图

在以上基础上，就可以用数据去驱动前端的运作。在底层的数据传输上，对于实时性要求低的数据请求，一般使用 Restful 风格的 HTTP 接口。对于实时性要求比较高或需要服务端主动推送数据的接口，本系统独立封装一套 Websocket 协议。在前端组件的页面路由分布上，安排如图 4 所示：

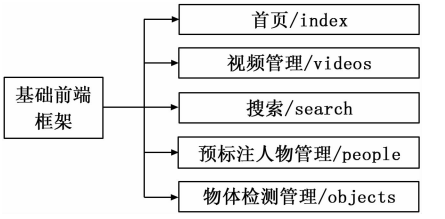

图 4 前端路由分布

2 模块的具体实现

为了提高系统的运行性能，在系统的核心功能实现上，使用 C＋＋ 11 语言进行开发。而在一些较为灵活的网络中间层处理上，则使用 Go 语言开发，前端界面则使用 ES6 开发。此外为了能够充分地利用一些已有的开源机器学习库，部分功能使用 Python 开发。通过多语言的混合式开发，能够充分地发挥不同语言的优点并提高开发效率。

2.1 视频帧分割模块

视频都是由一系列连续的图像组成，所以本系统视频处理最终还是要归结于图像处理。因此，要实现后续的功能拓展类工作，必须先要对视频进行帧分割。本系统的分割工作主要依赖 OpenCV 库。

其中视频帧分割模块的头文件定义如下：

```
/* @brief 视频帧分割
 * @param video_path 输入视频路径
 * @param out_folder 输出视频帧图像文件的文件夹路径
 * @return 返回帧图像文件列表
 */
std::list< std::string > extract(const std::string &video_path,const std::string &out_dir_path);
```

视频帧分割的流程如下：

1）根据输入路径构建 VideoCapture 实例

2）从二进制流中读取帧数据

3）把帧数据转码为 JPG 格式并按照系统的约定规则存放到相应的路径

4）把分割的记录数据存入 PostgreSQL 数据库

2.2 视频关键帧提取模块的实现

视频关键帧提取模块是在 OpenCV 的直方图计算相关静态方法上使用 C＋＋去实现的。对于一个已分割的视频，可以从数据库中读取出相关的分割结果，然后把视频序列以动态数组的形式送入提取器，提取器再调用系统中的 shot_detector::shot_bound 方法去调用。本方法在头文件中的定义如下：

```
/* @brief 通过对比当前帧和前一帧去判断是否检测到了一个镜头,检测镜头返回 true,否则返回 false
 * @param curr_frame 当前帧矩阵
```

```
* @param prev_frame 前一帧矩阵
* @param threshold 阈值,默认为 0.7
* @return 如果两个图像矩阵不属于同一个镜头返回 true,否则返回 false */
bool shot_bound(const cv::Mat& curr_frame, const cv::Mat& prev_frame, double threshold = 0.7);
```

在具体的实现上，相邻两帧的图像矩阵以引用指针的形式传入方法中，然后使用 cv::resize 方法进行一定的缩放，以防止单点计算量过大，接着使用 cv::cvtColor 进行灰度图的转换，再使用 cv::calcHist 进行直方图计算，最后规一化后会利用 cv::compareHist 进行相似度比较。当相似度超过 threshold 指定的阈值后，则认为此相邻的两帧是处于同一个镜头的。经测量，本系统选择了 0.7 作为默认的阈值。若发现 shot_detector::shot_bound 方法返回的对比结果为 false，则说明这两个相邻的帧是处于不同的镜头当中，系统会把后一帧记为一个新镜头的起点，即把视频帧实体的 is_shot_frame 字段设为 true。

2.3 视频关键帧非对称相似检索模块的实现

视频关键帧非对称相似检索模块的实现由 C++语言开发，其中依赖的库有 OpenCV、VLFeat 和 Yael 库。利用 OpenCV 的图像矩阵和颜色空间的相关方法进行关键帧的基础图像解析。由于 OpenCV 自带的 SIFT 功能没有充分利用到 CPU 的浮点运算指令，本系统选用了 VLFeat 库去提取 SIFT 特征向量，它可以利用诸如 AVX 等向量运算指令集。Yael 主要用于建立 Fisher Vector。

2.3.1 SIFT 特征向量提取

对于一个输入的关键帧图片路径，系统提供 sift_feat::get_keypoints_and_descriptors 方法去从图像文件中计算得出关键点和描述算子，如果获取成功则返回 true，否则返回 false。

```
/* 根据图像去获取其关键点和描述算子。如果获取成功,则返回 true。
* @param image_path 图像路径
* @param divide_512 如果为真,则描述算子元素会处于 0～1 之间
* @param keypoints 关键点包含 x,y,s,o 信息。向量中每一个元素都是作为一个关键点
* @param descriptors 向量中每一个元素都是作为相对于关键点的描述算子。
* @param num_descriptor 输出的描述子数量
*/
static bool get_keypoints_and_descriptors(
const char *image_path, bool divide_512,
std::vector<float*>& keypoints,
std::vector<float*>& descriptors, uint& num_descriptor);
```

其中的提取流程如下：

1）利用 cv::imread 读取图形文件为图形矩阵数据；

2）把 cv::Mat 类型的数据转换为 std::uint32 类型的一维的像素向量；

3）用函数 vl_sift_new（）初始化 SIFT 过滤器对象；

4）用函数 vl_sift_first_octave（）及 vl_sift_process_next（）遍历缩放空间的每一阶，直到返回 VL_ERR_EOF 为止；

5）对于缩放空间的每一阶，用函数 vl_sift_detect（）来获取关键点；

6）对每个关键点，用函数 vl_sift_calc_keypoint_orientations（）来获取该点的方向；

7）对关键点的每个方向，用函数 vl_sift_calc_keypoint_descriptor（）来获取该方向的描述；

8）最后，用函数 vl_sift_delete（）来释放资源。

2.3.2 GMM 聚类参数训练

在构建全局索引前，需要利用 INRIA 提供的假日数据集去进行 GMM 聚类参数的训练。假日数据集是一组图像，主要包含一些国外景点的假日照片。其中图片包含着多种的变化，例如旋转、视点和光照变化、模糊等。数据集包括非常多种高分辨率的场景类型（自然、人造、水和火效果等）。数据集包含 500 个图像组，每个图像组代表不同的场景或对象。每个组的第一个图像是查询图像，其余为该图像的变化。

2.3.3 全局索引建立

本系统的索引建立在 gmm 参数训练后，利用 yael 的 fisher 向量转换功能实现。其中的流程如下：

1）遍历数据库中的视频帧表，根据 is_shot 和 img_path 字段去获取关键帧相应的图像，然后利用 vfleat 进行逐个关键帧建立 siftb 特征文件

2）加载上一步建立的 gmm 参数

3）利用 yael 的 fisher 方法去进行 fisher 向量建立

4）合并多个 fisher 向量为一个矩阵并进行序列化保存

当系统接收到一个非对称相似视频帧检索请求时，会把请求图片转换为 fisher 向量化的 sift 特征，然后和全局索引进行遍历对比。对比得分进行排序并返回给用户。

2.4 人脸检索模块的实现

在使用人脸检索功能前，需要进行人脸预标注库的建立。

其中人物的实体结构体定义如下：

```
struct person {
std::uint32 person_id;
std::string name; //人物姓名
std::string description; //描述
std::uin32 created_at; //创建时间
std::uint32 modified_at; //修改时间
}
```

人脸实体结构体定义如下：

```
struct person_face {
std::uint32 person_face_id;
std::uint32 person_id; //人物 ID
std::string img_path; //人物的单人照片路径
std::uin32 created_at; //创建时间
```

std::uint32 modified_at; //修改时间

}

2.4.1 基于的 HOG 和 SVM 的人脸探测的实现

为了提高人脸探测的速度，本系统在 OpenCV 上，使用基于 HOG（Histogram of Oriented Gradient）特征的人脸探测方法。构建一个基于 HOG 的人脸探测器，实际上就是利用人脸数据的 HOG 特征去训练一个 SVM 分类器。其中的构建流程如下：

1）利用加州理工学院互联网人脸数据集（Caltech Web Faces）的 13436 张各种不同角度的 36×36 的人脸裁剪照片作为正样本。对于负样本，我们采用多尺度非人脸场景中随机裁剪 36×36 的图片，其中负样本的数量为 85000 个。然后以 cell 等于 4 为参数，利用 OpenCV 的 HOGDescriptor 去提取样本的 HOG 特征。

2）利用 OpenCV 的 CvSVM::CvTermCriteria 定义迭代，终止条件为当迭代满 1000 次或误差小于 FLT_EPSILON。利用 CvSVM::CvSVMParams 去指定 SVM 分类器的核函数为线性函数、松弛因子为 0.01。最后使用 CvSVM::train 方法进行训练迭代并使用 save 方法进行 svm 模型保存。

2.4.2 基于 Dlib 的脸部编码的实现

本系统的脸部编码是基于 Dlib 库并使用 Python 语言实现，其编码流程如下：

1）利用 dlib.deserialize 方法加载预训练的 FaceNet 模型；

2）使用 dlib.compute_face_descriptor 方法计算脸部的特征向量。

2.4.3 基于 KNN 人脸分类器的实现

当要对某一个视频进行人脸检索前，需要提前对其中包含的人脸特征值提取然后加载到 knn 中并以 person_id 作为标签去训练一个 KNN 分类器。训练过程依赖于 scikit－learn，过程如下：

1）从数据库中加载预标注人脸库，以 X 作为人脸特征矩阵，Y 作为 person_id 的标签矩阵

2）以参数 algorithm＝knn_algo、weights＝'distance'，然后利用 sklearn.KNeighborsClassifier 方法进行 KNN 分类器的训练

2.5 基于 darknet 的物体检测实现

本系统的目标检测引擎主要在视频关键帧上匹配出已出现过的常见物体，物体检测是基于 darknet 引擎。在深度神经网络模型上，选用已预先训练好的 yolov3.weights 模型。由于 darknet 本身是基于 C 语言编写的，其在面向对象方面会比较弱，所以本系统使用 C＋＋将其进行封装。

其中的封装中，提供了如下方法：

· static Darknet * get_current（）

说明：返回利用单例模式返回 darknet 的实例

· Darknet（）

说明：darknet 的构造方法，构造返回 darknet 实例

· ～Darknet（）

说明：darknet 的析构方法，清理 darknet 所占用的资源

· void initialize（int gpu_id ＝ 0）

说明：初始化方法，其中可以指定是否启用 GPU 去进行加速运算，同时它也会负责加载模型文件

· void run（）

说明：运行 darknet 监听线程。使得整个的检测工作可以使用异步方式去工作，从而防止系统因长时间运算而导致的停机状态

· void process（cv::Mat& image，process_func_ptr process_func ＝ nullptr）

说明：图像物体检测方法。其中它会把 OpenCV 类型的图像矩阵送进去队列中去等待处理。消费者对象会根据指定的算法去进行物体的检测

3 运行与测试

3.1 系统运行环境

表 1 系统运行软件版本需求

项目	版本要求
操作系统	Ubuntu 17.10
c 语言编译器	GCC 7
C＋＋编译器	G＋＋ 7
Go 语言编译器	Go 1.10
Node.js	Node.js 8
OpenCV	OpenCV 3.2
Boost	Boost 1.6
PostgreSQL	PostgreSQL 10

3.2 系统运行流程

系统首次启动时，需要完成以下流程的开启：

1）导入系统数据库表；

2）开启 PostgreSQL 数据库；

3）需要在 HOME 目录下新建 vrs_storage 目录；

4）修改系统根目录下的 config.yaml 文件进行配置。

3.3 实验结果与分析

为了验证本系统的性能，对 CNN 于 2018 年 12 月 20－26 日发布在 youtube 上的新闻短片分别进行了非对称相似帧检索、视频人物检索及物体检索等检索实验。各检索结果及数据分析如下：

3.3.1 视频非对称相似帧检索：

视频来源：CNN

链接：youtube

视频：Who's been naughty and nice in 2018 politics ｜ With Chris Cillizza

检索图片的来源：视频《Who's been naughty and nice in 2018 politics ｜ With Chris Cillizza》的第 164 秒

检索结果数：100

检索结果如图 5 所示，检索信息如表 2 所示。从表 2 可

知，近距离全屏拍摄检索的耗时最长，主要是因为近距离全屏拍摄时，待检索照片像素较高，数据量大，图像预处理耗时较长，但在检索的 100 个结果中，对应的图片排在第 1 位。相反，虽然远距离全屏拍摄的耗时最短，但检索结果却排在第 8 位，说明待检索照片图像预处理耗时较短，但同时会在检索过程中跟其他类似的图片有较高的匹配度。整体上，视频非对称相似帧检索都能在预期的时间上检索出所需要的图片。

图 5　视频非对称相似帧检索结果图

表 2　检索结果数据表

检索类型 \ 检索结果	检索结果位置	耗时/s
近距离整屏拍摄	1	4.2
远距离半屏拍摄	1	3.54
远距离全屏拍摄	8	3.28

3.3.2　人脸检索和物体检索实验数据

数据：2018 年 12 月 20 日－2018 年 12 月 26 日 CNN 发布于 youtube 上的新闻短片；

总时长：29 815 秒；

总帧数：29 815；

关键帧数：2 535.

检索结果如表 3 和表 4 的所示。

表 3　物体检索信息表

	人类	自行车/摩托车	背包	杯子	汽车	狗	酒杯	沙发	交通灯
结果数	2248	2	3	17	36	3	1	28	2
非预期结果数	0	0	1	1	1	2	0	0	1
检索准确率/%	100	100	66.7	94.1	97.2	33.3	100	100	50

表 4　人脸检索信息表

	特朗普	Chris Cuomo	Bianca Nobilo	David J. Shulkin
简介	美国总统	CNN 记者兼主持人	CNN 记者兼主持人	CNN 记者兼主持人
结果数	119	8	8	24
非预期结果数	16	1	0	9
检索准确率/%	86.6	87.5	100	62.5

由表 3 可知，人类检索结果的数据及准确率最高，狗的准确率最低。由于人类的检测算法相对较成熟，而对于一些特征及背景较复杂物体，则会出现相对较大的误检索。由表 4 可知，检索准确率取决于人脸的特征值及人脸训练度，特征值或训练度高的，则检索准确率也会相应的增高。

4　结束语

本文探索了如何结合传统的图像处理算法和深度学习去构建一个视频检索系统。通过 B/S 架构可以让用户直接使用 Web 浏览器进行检索，也可以方便的部署到云端、充分利用云计算服务商提供的相对低廉的机器成本、快速伸缩和多容器备灾等特性。另一方面通过模块间的耦合度底且模块与上层的通信使用基于 HTTP2 的 Protobuf 协议的 GRPC 实现，实现了模块间分布式部署的可能。在后续的改进上，可以尝试利用深度哈希算法进行检索工作，这样就可以利用深度神经网络里面的隐藏层自动地提取特征信息，另外，还可以利用语音识别等技术，生成语义性更好的文本关键词或标签等信息。

参考文献：

[1] 徐　俊. 基于视觉的文本生成方法研究 [D]. 合肥：中国科学技术大学，2018.

[2] 唐　敏. 基于深度学习的中文实体关系抽取方法研究 [D]. 成都：西南交通大学，2018.

[3] 赵　静. 基于多特征融合的视频文本检测 [D]. 西安：陕西师范大学，2018.

[4] 赖梦芳. 视频人脸快速检索关键技术研究与实现 [D]. 成都：电子科技大学，2018.

[5] 路　程. 视频内容检索技术概述 [J]. 山西科技，2018，33 (2)：56－58.

[6] 冯兆华，朱允斌，李卫强. 基于深度学习的视频近似拷贝检索 [J]. 计算机应用与软件，2018，35 (1)：160－163，182.

[7] 李　想. 基于深度学习的视频敏感信息检索的研究 [J]. 电子设计工程，2017，25 (21)：137－140.

[8] 孙　彬. 基于内容的视频分析关键技术研究 [D]. 南京：南京邮电大学，2017.

[9] 杨　琳. 基于关键帧复杂度的视频场景边界检测算法 [D]. 武汉：武汉轻工大学，2017.

[10] 夏洋洋. 基于深度学习的非限定条件下人脸识别研究 [D]. 成都：西南交通大学，2017.

[11] 皮　洋. 视频图像内容匹配与检索研究 [D]. 长沙：湖南大学，2017.

[12] 操顺德，华　宇，冯　丹，等. 面向海量高清视频数据的高性能分布式存储系统 [J]. 软件学报，2017，28 (8)：1999－2009.

[13] 尚佳敏. 基于特征聚类的视频摘要生成技术研究 [D]. 西安：西安理工大学，2016.

[14] 徐向茹. 基于人脸识别身份验证系统的研究与实现 [D]. 成都：电子科技大学，2016.