Appscio: A Software Environment for Semantic Multimedia Analysis

Gerald Friedland[1,2] Eden Hensley[1] Ramesh Jain[1,3] Jerry Schumacher[1]
[1]*Appscio Inc, 80 Airport Boulevard Street, Ste. 206a, Freedom, CA 95019*
[2]*International Computer Science Institute, 1947 Center Street, Suite 600, Berkeley, CA 94704*
[3]*Bren School of Information and Computer Science, University of California, Irvine, CA 92697*
{gfriedland,ehensley,rjain,jschumacher}@appscio.com

Abstract

The goal of the Appscio(tm) software platform is to ease the creation of multimedia content analysis applications that consist of components provided from multiple sources, in different programming languages, and for various operating systems. Appscio provides a unified approach that standardizes the entire process of development, deployment, and integration of components into productive applications. In addition, the aim is to facilitate the integration of analytic approaches with traditional sensor output. Therefore the framework allows the combination of multimedia analytics with any other event generating sources, as used for observational systems. A basic concept of the platform is to allow mainstream application developers to create semantic-rich web applications that integrate components previously only accessible to scientists.

1. Introduction

In recent years, artificial intelligence and signal processing research has developed a large amount of algorithms and methods to analyze the semantics of signals originally encoded for direct human perception, e.g. images or sound. Fields such as computer vision and audio processing, although still active research fields, start to produce more and more mature algorithms that are of potential use in products. Unfortunately, the integration of research results into larger real-world applications is often hindered because of the lack of a common platform that is able to integrate research components in an efficient way. As a result, the software engineering quality of research components is often very low. Therefore prototype systems are many times ad-hoc solutions that are difficult to reuse for anybody else but its original creator. In the end, many components have to be re-invented over and over again and still never end up in a real production environment.

Although the extraction of meaning from multimedia content of various types is an active field of research in a large number of institutions, multimedia is often regarded a synonym for audio and video. In the philosophy of the Appscio framework, multimedia includes not only video, audio, and related metadata, but any other source of sensory information that can help accomplish a certain task. This particularly includes the use of traditional sensors, such as RFID tags or light sensors. The combination of traditional sensors can sometimes provide the right amount of context information that is needed achieve production-level robustness of audio and/or video analysis approaches.

The following article presents the architecture concept of the Appscio platform and provides a usage example. Section 2 starts with the discussion of related work. Section 3 summarizes the requirements for a multimedia content analysis framework in contrast to the framework presented in Section 2. Section 4 provides an overview of the architecture before Sections 5 and 6 continue with a more detailed description and Section 7 discusses a usage example. Section 8 concludes the article with references to further documentation.

2. Related Work

The need to create standardized interfaces for multimedia streaming is widely accepted and several solutions exist. Current multimedia software frameworks such as Sun's Java Media Framework, Microsoft Direct X, Apple Quicktime, Real Networks' Helix DNA, GStreamer, Phonon, Gegl, xine, and ffmpeg are perfectly suited for the integration of stream processing components from different development sources. However, their primary purpose is the composition of plugins for audio and video recording, playback, and network transmission. The presented frameworks mainly focus on the integration of components on a software engineering level, i.e. the composition is performed by software developers, often people who develop individual components. Additionally, the amount of concepts that have to be understood in order to implement a certain component usually forms a large entry-barrier and prevents many

978-0-7695-3279-0/08 $25.00 © 2008 IEEE
DOI 10.1109/ICSC.2008.56

researchers from adapting their prototypes into a certain framework. Rather than targeting the integration of different signal processing algorithms, these framework were originally designed to facilitate cooperation with different hardware devices. For this reason, some of the frameworks are closely tied to the underlying operating system, which results in them being platform-dependent. Most importantly, semantic video analysis, has additional requirements that are not fulfilled by traditional multimedia frameworks. As a consequence, implementing machine learning techniques in these frameworks becomes a cumbersome task.

3. Requirements for a Content Analysis Framework

Many of the tasks performed by traditional multimedia frameworks, such as transcoding of formats, or random positioning in streams are also needed for multimedia analytics. For this reason, the analytic part of the Appscio framework is built on top of a popular open-source streaming framework, namely GStreamer (see Section 5). Semantic analysis of multimedia content, however, has additional requirements that are imposed by the algorithms that are currently in use.

Multimedia streaming frameworks do not contain standard feature extraction components or machine learning modules, such as Neural Networks or Support Vector Machines. A content analysis framework has to be able to process data offline as many semantic computing algorithms are not able to handle data in chunks. The integration of these algorithms requires more than a linear pipeline as sometimes data has to be passed back and forth between processing steps. The communication between modules consists of more than single stream of audio and/or video data in the form of byte arrays. Rather, events must be passed between the different steps of analytics. The metadata created by the algorithms must be interpreted by all relevant components, especially during the playback and persistent storage of a stream.

In order to be suitable as an experimental environment, right from the beginning of a first experiment, technical requirements include the cross-language portability and the ability of the framework to run in user space.

4. Architecture Overview

The following sections provide an architecture-level overview of the Appscio platform which incorporates the concepts discussed in the previous sections.

The Appscio platform provides mainly two different views: A developer interface is provided for different programming languages (such as C or Java) and an application-integrator view is provided through both a typical web-development interface and an integrated development environment.

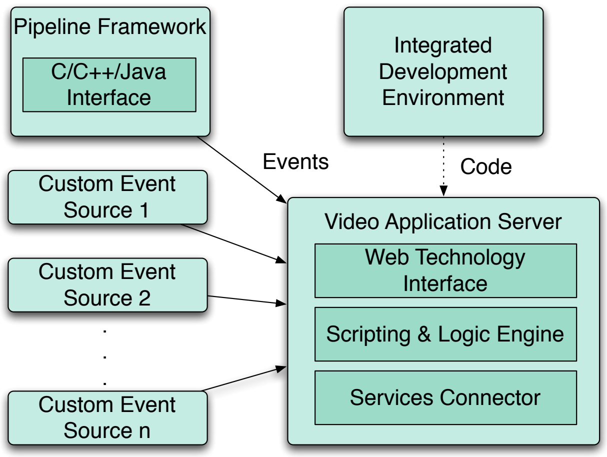

Figure 1. Conceptual Overview of the Appscio Platform

Figure 1 is a diagram of the overall architecture of the Appscio Platform. The Pipeline Framework is mainly accessed by component developers. Video and audio signals may be processed and analyzed by components inside the pipeline to extract information and generate appropriate events. Events can be generated by all kinds of different sources, e.g. sensor output or output of semantic computing algorithms. They are passed to the server via a web-services call.

The Application Server processes incoming events from different sources. The server can be programmed using different web-technology interfaces and using a web-integrated development environment. It can interface with various web services, such as e-mail.

The following sections describe the two main components in a higher level of detail.

5. Pipeline Framework

The Appscio Pipeline Framework is a core component of the platform. It provides a standard interface for audio and video analysis components. While the Pipeline Framework builds a layer on top of GStreamer, it abstracts many details of the original streaming framework. In fact, the layer is built such

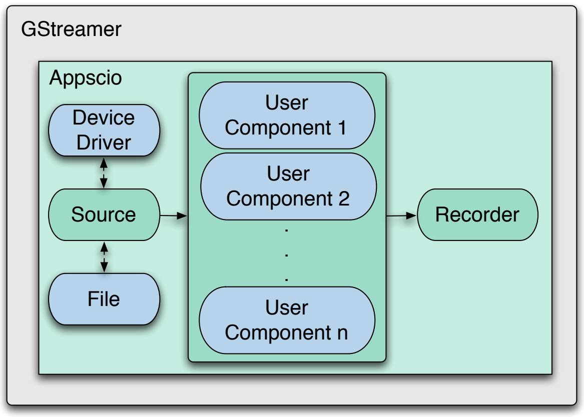

Figure 2. Conceptual overview of the Pipeline. The Pipeline is a layer on top of GStreamer abstracting away its details while extending is functionality for multimedia analysis

that it could be on top of any similar platform. This way, the Appscio Pipeline Framework not only facilitates the development of components, but it also extends the framework's scope from a recording-transmission-playback interface to a mixed-sensor analytics standard. Figure 2 provides a diagrammatic overview of the Pipeline Framework.

GStreamer provides a generic framework to build processing chains of video and audio processing components. It provides a standard interface to support and synchronize different formats and devices. Data between graph nodes is exchanged via memory buffers. This provides a lightweight and efficient way of communication. Therefore it builds a widely accepted base for many different media recording, playback and/or transmission applications.

The Appscio Pipeline Framework Host inserts itself as a graph node into the GStreamer framework. A multimedia application can either read from a file or from a device. Application developers can either rely on existing device drivers and file readers or may create their own. Processing and analysis is mainly performed in filter components. Stream formats between different components are defined using variable types. The Appscio Pipeline Framework takes care of any format synchronization issues by relying on GStreamer. Individual components are configured using properties.

The Appscio Pipeline Framework can be instructed to record the stream to a file. A central feature of the Appscio Pipeline Framework is the support of asynchronous recording and processing requests. Asynchronous requests allow to record and/or process portions of the stream independently of the current stream position. That means the processing time-interval may start in the past and/or end in the future. This allows a processed logging of past and/or future content as an immediate reaction to an event. For example, if an analysis algorithm detects a "stolen purse" event, the recording of the video stream should start a couple of seconds back in time. In order for this to work, each asynchronous request is internally assigned an individual pipeline with filter components and a recording sink. A buffer with configurable length is used to cache the stream in order to access past stream positions.

6. Application Server

The Appscio Application Server is the central instance to control pipeline execution and maintain system configuration and state. It is controlled through different interfaces. A web development interface provides control through Javascript and PHP. Events may be subscribed through JEXL. Multimedia content can be stored persistently through the Application Server Database, which is basically an RDF store. This content and all metadata may be accessed through SPARQL, which returns URLs pointing to the resources. The Application Server can be accessed by an Integrated Development environment which enables most of the programmatic functionality to be performed through a mouse-driven interface. The IDE provides means to create and edit pipelines through drag and drop as well as subscribing to events and to create scripts to handle them.

7. Example Scenario

The following scenario provides a typical use case for the Appscio platform.

7.1. Scenario

A retail store wants to make sure, that people only exit after going through the cashier line and use the entrance only for going into the building. Due to a difficult architectural situation, the entry of the retail store is not clearly visible for the cashiers and other staff working at the market. Therefore they use a dual photo barrier to make an alarm sound whenever people try to exit the market though the entrance. For disabled people, however, exiting through the entry would be much easier. In addition, during the last moths the store manager recorded an increase of stolen goods and the detective reports more cases on persons trying to exit through the entrance by circumventing the photo barrier.

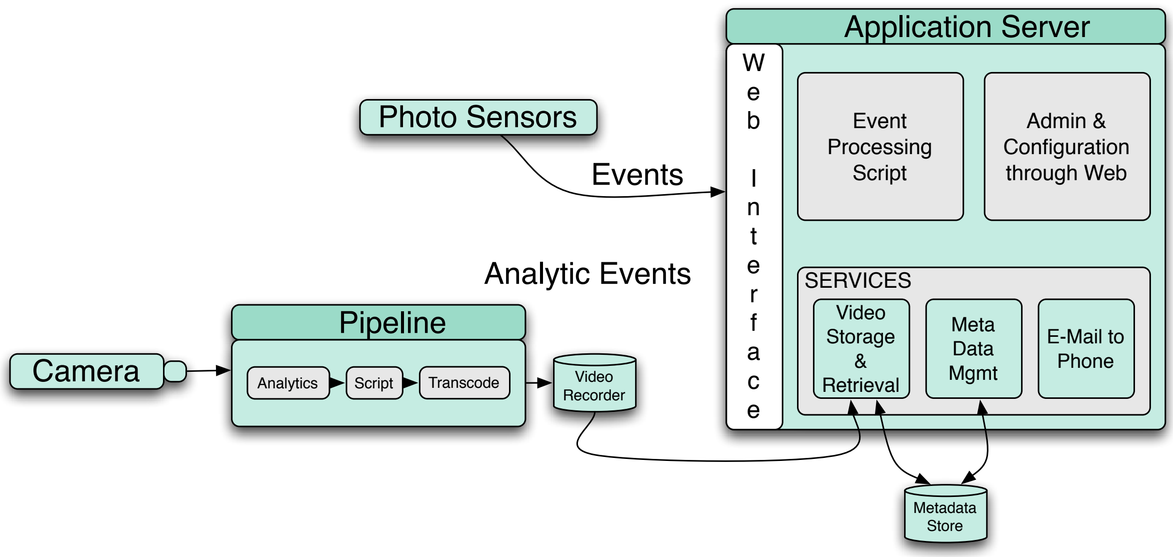

Figure 3. The example described in Section 7 as concrete implementation in the Appscio platform.

7.2. System Design

This problem can be solved by the following system: A camera is installed in addition to photo barrier. A vision system analyzes the camera picture to detect activity in the area and to figure out the direction of the movement of the activity. A support vector machine is used to classify the moving video blobs into either walking persons or wheel-chair persons. The goal is to fire-off an alarm if, and only if, a non-wheel chair persons tries to exit the store. This means, even if the photo barrier is detecting an exiting person, the alarm should not sound. If the photo barrier is not detecting anything, the vision system might catch an illegal exit attempt. Additionally, any illegal attempts should be recorded after the system has detected a possible fraud.

Currently, vision systems are not accurate enough to provide an absolutely exact means to detect possible frauds in this scenario. As observed in many cases, simple photo barriers are accurate but can be easily circumvented. Therefore it makes perfect sense to combine two systems in order increase the fraud detection rate.

7.3. Implementation in the Framework

Figure 3 illustrates the implementation of the example application into the Appscio platform. The above mentioned photo sensors are connected to a computer that converts the binary information collected by them into events that are sent to the Appscio Application Servers. The vision system consists of a set of Appscio components that are controlled and executed inside the analytical pipeline. The output of the pipeline also consists of events that are sent to the server. The Application Server is programmed to handle the events from the sensors and the vision. The configuration and maintenance of the system is performed through web pages in the Intranet of the store, internally controlled by the Appscio Application Server web interface. If a possible fraud is detected, an asynchronous recording request is triggered to store the last couple of minutes of the captured video in the Application Server's RDF store, for possible later review by law enforcement. Instead of triggering an awfully loud siren, the web services interface of the Application Server is used to send an e-mail to the mobile phone of the current shop detective, who may silently choose to help a disabled person that caused a false-alarm in the vision system.

8. Conclusion

Semantic multimedia analysis involves the use of domain specific algorithms and combining them in an engine that facilitates the interpretation of the derived information in relation to the problem's domain knowledge. This article presented the concepts of the Appscio multimedia analytics platform which is guided by the idea of semantic multimedia analysis. The framework, currently under development, will be released open source, enabling organic development of the community of researchers and users. The hope is that contributors will help to make the platform a valuable and productive part of the semantic computing community. For further information visit the website: http://www.appscio.com.