The world of interactive media systems and applications

Mobile, cloud-based analytics, and user engagement are a must

Sam B. Siewert

December 10, 2013

Digital media is rapidly integrating with interactive systems and social networking. For example, television is now an experience that includes social networking for many viewers, who either use their smartphones on an ad-hoc basis or by using a device or social-networking tool provided by the cable or telecommunications network or content provider. Cable TV master facilities (headends) and digital artists need to evolve quickly to meet this growing demand to keep their content and services relevant. This article describes tools and methods for Linux® servers for digital media and interactive systems.

Digital video and data-in-motion analytics have evolved quickly as cable systems have gone digital (nationwide and globally), as digital cinema has become ubiquitous with new features such as 3D, and as mobile digital media access has become available for smartphones and tablets. Just over a decade ago, most cable providers offered digital cable as well as high-definition and on-demand programming. The trend continues with social networking integration with digital cable and with Data Over Cable Service Interface Specification (DOCSIS), which provides Internet service over a cable modem.

At the same time, Netflix, Hulu, Gaikai (a Sony subsidiary), NVIDIA Cloud, Gamasutra, and many other purely Internet-based content and interactive on-demand services or development networks have emerged (see Related topics for more). Cinema has evolved so that globally and in the United States, with the 3D Digital Cinema Initiative, most cinema is now entirely digital, and film is truly a thing of the past, along with vinyl records and tape. In 2007, the U.S. Federal Communications Commission shut down NTSC (analog television) in the United States and replaced it with Advanced Television Systems Committee digital television. Little analog media of any type survives today.

The huge advantage of digital media is that it is on demand, location- and time-shifted, can be tied to social networks and big data analytics, and it's more cost-effective and secure to produce, distribute, and consume. Finally, the all-digital media world allows for a much wider range of artist and creative developer participation, including the consumer as an interactive participant. Anyone
who has a creative mind, some computer skills, and patience can join this new creative digital culture.

**Interactive media system fundamentals**

Digital media incorporate audio, video, and rendered graphics — often integrated — to form content and applications for the ears and eyes. Interactive media systems add user controls for the presentation of the digital video, audio, and graphics, so at the very least, a user can control playback of audio, video, and graphic animations. However, options for user interaction can go much further to include advanced interactive media applications such as augmented reality (AR), where a real-time camera video source is combined with graphical rendering to annotate a view of the world. Interaction with media can include simple approaches such as web viewing while consumers watch digital cable or can be more sophisticated, such as Google Glass (see Related topics), where a user's view of the real world has interactive graphical overlays to implement AR.

The combination of analysis of user feedback — whether point and click or gestures and expressions (recognized by a camera) — with the presentation of media, including graphic overlay, provides powerful content that can better engage the user. Google, Facebook, Twitter, and early pioneers of web-based analytics have used point-and-click browser feedback to tailor HTML, but imagine applications that literally repaint an augmented view of reality. New devices like Google Glass, Oculus Rift, inertial measurement on chip (InvenSense, STMicroelectronics, for example), and numerous gesture-recognition and advanced interactive devices are opening a new world for applications that combine live camera streams with video, audio, and rendered graphics into these applications. The remainder of this article introduces you to key open source methods and tools that enable development of applications for interactive media systems.

**Interactive media system tools and methods**

Great applications for interactive media systems must include advanced interactive devices that are mobile but, to be really engaging, they must also include cloud-based analytics to map user interest to global information and social networks. This is a huge challenge because application developers must have mastery of mobile (embedded) systems and applications. The following is a list of key tools and methods that take advantage of open source technologies. First I describe them, then I dive into examples for rendering and animation, digital video encoding, and basic digital camera capture and scene analysis using computer vision methods.

- **Mobile-to-cloud operating systems**, such as Android, that build on a native development kit (NDK) using Linux have a huge advantage for couple of reasons: They enable development of new device interfaces and drivers (for example, for a 3D digital camera with an on-chip inertial measurement unit), and they come with an SDK, which includes development tools you can use to create and debug Java™ applications. The SDK is a toolkit available for download (see Related topics), that runs on any system such as Windows® or Linux, but targets Android. The huge value of Android for Google is that Linux servers and mobile Linux devices can now interact to gather user input, analyze it in near-real time, and feed this data back to mobile devices (see Related topics).
  Apple similarly has iOS for mobile devices (with the Objective-C SDK and Mac OS X native layer) that Mac OS X server machines in the cloud can support, but has taken a more
proprietary approach than Google. Either way, both companies have created an application SDK that makes the development and deployment of interactive media applications simple. In this article, I focus on Linux because it is open all the way down to devices at the NDK layer. Furthermore, Linux can be found on everything from digital cable set-top boxes to smartphones and tablets to high-end servers.

- **Digital video and audio encoding** built on standards such as H.264/H.265 for MPEG (see Related topics) is a must, and the deeper the understanding a systems developer has for digital media encode, transport, decode, and presentation the better. This is a nontrivial, highly technical area of knowledge that the creative content developer can mostly ignore, but the interactive system developer must master this field. This article goes deep with an example of how to use OpenCV to start the development of a simple first cut at encoding images into a compressed format similar to JPEG and MPEG. Furthermore, this example provides insight into what is known as an elementary stream (for audio or video), but further knowledge is required to understand program and transport streams used for standards like H.264. Pointers to resources are provided.

- **Graphical rendering and digital video frame annotation** ranging from offline, complex, photo-realistic ray-trace rendering to polygon-based graphics for game engines to simple annotation of video (think the ESPN chalkboard on Monday Night Football) are all needed to provide overlays for interaction on camera data and playback video. This ability to modify reality and enhance digital cinema, digital cable, and Internet content is the heart of interactive media systems. In this article, I look at the Pixar RenderMan tool, which is a high-end ray-trace rendering tool (very computationally expensive), but also provide pointers to polygon-rendering tools and methods, as well as simple frame-based transformations and annotations.

- **Advanced user interaction and sensing** is the last and perhaps most critical technology that must be mastered because this defines the interaction. This component includes web-based point and click, but much more: These systems must incorporate gesture recognition and facial expression analysis to become less intrusive and more engaging. The requirement to stop and click Like, has interrupted my experience with the media and in fact is impossible for AR applications where I might be walking, playing a game, driving, or flying an aircraft (for obvious reasons).

This is by no means an exhaustive list of interactive media system and application required technology, but you can find more to explore in Related topics. The goal here is to reset your thinking as an application developer or systems designer. The world of interactive media systems and applications must be mobile, must include analytics from the cloud, and must engage but not distract the user.

### Build and learn examples for interactive media

**Interactive mobile media systems**

*Augmented reality*, where the world is simultaneously observed in person and through a display fed by a camera and graphics, is only getting started. The ability to overlay information, in a non-intrusive fashion, from the cloud to users immersed in an activity in their environment so the overlay becomes a sixth sense (note that humans actually have several dozen senses, so perhaps a 25th sense, to be more accurate) is the goal. For example, if I'm learning golf, my AR goggles can indicate distance from the tee box to the flag down the fairway and suggest a club, perhaps based on analysis of my last drive from a similar hole.
The ability of AR devices, systems, and applications to sense more than just where I am, to include what I'm looking at and to take advantage of advanced sensing such proprioception, is the future.

Proprioception is the sense athletes use for muscle memory, to train themselves to drive a golf ball well, for example. An inertial measurement unit on a chip (see Related topics) combined with 3D cameras enables similar sensing for the AR device, which can be uplinked to the cloud to correlate to course information, past performance, and a wide range of information to the delight of an advertiser, perhaps to the highly targeted suggestion to purchase a new driver.

Building interactive media systems and applications that incorporate cloud-based analytics requires skill with digital media encode, transport, decode, computer vision, and correlation to databases. For more advanced applications like AR, it would also require experience with graphical rendering. In this section, I provide simple examples to get you started.

**Camera capture and frame transformation in Linux**

OpenCV, the Open Computer Vision application programming interface (API) developed by Intel and returned to open source, came about because of the observation that universities researching computer vision and interactive systems benefited greatly from reusable algorithms for image processing, the mathematics of scalar/tensor transformations (a technical term for sharpening or color enhancing a frame of red, green, blue pixel data in a video frame), and for advanced concepts such as detection of faces and recognition.

However, before anything else, you must be able to capture real-time camera data. The sample code in simple-capture.zip (found in opencv-examples.zip in the Downloadable resources section) shows how this used to be complicated, even with APIs such as Video for Linux 2. But the process has been vastly simplified and abstracted with OpenCV, as shown in the sample code in simpler-capture.zip, also found in the opencv-examples.zip file in the Downloadable resources section.

Computer vision must have access to uncompressed video frames prior to encoding with MPEG for transport. Uncompressed tri-color RGB video frames constitute very high bandwidth: 1080x1920 3-byte pixels at 30Hz for high-definition video is 180MB/second. Uncompressed video at 180MB/sec, which would likely be 20Mbits/sec for compressed H.264 MPEG4 Part 10 standard (about 72:1 compression), is considered good quality. Higher bit rates, for example, 36Mbits/sec, would provide excellent quality (about 40:1 compression).

For interactive media systems and applications, you will likely want to annotate the raw frames, perhaps transform them to find edges of objects, or segment scenes or recognize faces, but you may also want to uplink highly compressed H.264 video to the cloud. Streaming can be handled by GStreamer (see Related topics) and MPEG encoding software such as avconv. To better understand this, let's look at one step in MPEG encoding using OpenCV: the transformation of a single digital video frame into macroblocks with the Discrete Cosine Transformation (DCT). The overall MPEG encoding flow from raw digital video to 188-byte MPEG program/transport stream packets is shown in Figure 1.
Figure 1. Agilent Technologies' MPEG encoding diagram

Frame transformation for compression to encode and transport to devices

Figure 1 comes from Agilent Technologies' documentation for MPEG2, which digital cable systems still use, but the technology is rapidly being replaced by MPEG4, which has higher compression ratios and better digital video quality. MPEG2 is still useful for anyone new to encoding, however (see Related topics for the original Agilent documentation). MPEG2 is a standard documented in International Organization for Standardization (ISO) 13818-1 and 13818-2. The newer MPEG4 and transport encapsulation with International Telecommunications Union H.264 (and H.265, available in 2013 as standard the High Efficiency Video Coding ISO/IEC 23008-2 MPEG-H Part2, ITU-T H.265) is found in the ISO/International Electrotechnical Commission (IEC) 1449610 standard (see Related topics to download the H.265 standard).

To truly appreciate MPEG compression, it's best to implement your own encoder, which you can do by leveraging OpenCV to avoid the mathematical details of transforms, such as the DCT. (For those who can't resist, I have included an unoptimized 2D DCT and instructions for verification using Octave, found in Downloadable resources.) Overall, MPEG I-frame compression involves:

- Subsampling color (red and blue) compared with green
- Division of each frame into 8x8 macroblocks
- DC transform of each macroblock
- Scaling
- Weighting and truncation of the DCT macroblock
- Zig-zag lossless compression of each macroblock

These events constitute the MPEG I-frame (intra-frame compression), which in turn is used as an anchor in a group of pictures to compress frames based on pixels that don't change significantly frame to frame (inter-frame compression).

Figure 2 shows an 8x8 macroblock DCT of one color channel for an image (green). It looks gray, but this is what happens when just one color channel is displayed. Play with the DCT and images, and convince yourself that the DCT can be either lossless or lossy — lossless if real values are maintained, but lossy if you use quantization to truncate the DCT to an integer range. If you don't truncate the DCT, you get the same data back with an inverse DCT (iDCT), as shown in Figure 2. For color images, you simply need to do this for each color plane.
Figure 2. OpenCV example of macroblock DCT of a single color channel

Frame transformation for computer vision

Figure 2 also provides an example of frame transformation — in this case for the purpose of image compression, but transformation in general can be used to find edges (for parsing and recognizing text) or to detect and recognize faces. This process is covered in an earlier developerWorks article titled "Cloud scaling, Part 3: Explore video analytics in the cloud."

Transformations in general modify the look of individual frames or provide simplified encoding of the key features of a frame. In the ideal sense, transformations can be used to reduce an image acquired by a camera to key features to understand the scene that the camera captured — the basic goal of computer vision. Today's computer vision is nowhere near a human's capability to understand a scene, but in some cases, it can offer advantages by assisting users with focus and scene meta-information (for example, not only recognition of a plane but the most likely make and model of that aircraft as seen from the ground).

What if you want the opposite? Rather than reducing an image to a scene description, you can turn a simple scene description into a photorealistic image (i.e., you can render it). A human-like ability to truly recognize objects in complex, uncontrolled environments is more advanced than today's computer vision enables. Details of make and model, and reliable scene descriptions in outdoor or other uncontrolled environments is more advanced than current computer vision can support. The ability to parse complex scenes in real time and augment them with graphical annotation is an open R&D effort. Progress is being made, but most computer vision solutions operate at a level comparable to a 2-year old human at best.

As you know, Hollywood produces rendered digital cinema that is becoming almost indistinguishable from real images. Let's see how this is done (using the pixie-examples.zip Downloadable resources). See Figure 3.
Integration of 3D rendering animation with digital video

RenderMan is a language for describing scenes with geometric objects, lighting, perspective for the viewer, colors, and textures and for coordinating system handling for the 3D scene that is rendered on a 2D screen. Furthermore, although Figure 3 looks crude, a more patient artist than I am, can create photorealistic scenes given sufficient time and mastery of the language.

Each scene rendered into a frame can in turn be animated into a digital video with simple frame-by-frame modifications. For example, I have included an encoded MPEG4 of Figure 3 where the observer rotates about the camera coordinate y-axis, which gives the sense of flying over the top of the cones, sphere, and cylinder I described with RenderMan. I have included C code and RenderMan scripts for this so you can play with and learn about rendering. Pixie is open source and can be downloaded, built, and installed on Linux (see Related topics). The resulting TIFF frames that Pixie renders can be encoded into MPEG4 by using avconv (FFmpeg) with the following command:

```
ffmpeg -f image2 -i ./test_animation%d.tif -vcodec mpeg4 -qscale 1 -an test_animation1.mp4
```

Graphical frame annotation

Rather than generating an alternate reality with rendering, as is done with immersible virtual reality and photorealistic rendering of scenes, you can annotate reality as observed by the user and a camera or cameras that see the same scene at the same time. This is the concept behind AR.

Simple annotations like that of a football field with a graphical first-down line are powerful, allowing viewers to correlate information that is vital to scene understanding with the scene, through annotation. Uses include applications in which technicians use AR goggles to see part information while fixing a vehicle. Manual correlation to match a part observed to an entry in a parts catalog is painstaking.

In many cases, machine vision can be superior to human vision, which is limited to a small part of the spectrum (the visible, with color wavelengths known as the tri-stimulus). Instrumentation
can transform infrared into visible colors so humans can see at night and visualize thermal properties of objects. AR applications are, without doubt, the ultimate interactive media systems and applications. They will be greatly enhanced not only by mobile devices but by the cloud-based digital video analytics systems they connect to for correlated information display.

The back-haul of video as seen through AR goggles will require H.264 and H.265 with 3D encoding and transport to enable highly compressed video uplink to the cloud or perhaps even more advanced scene descriptions (or key features). Today, this can be done using GStreamer on Linux devices to uplink or distribute media to and from mobiles that run Linux (see Related topics).

The future of interactive streaming media systems and applications

This article makes an argument for the value of interactive media systems — not just for one-way distribution to consumers from a server (headend) but for two-way interaction, with streaming uplink from observers to cloud-based video analytics. Why? Because observations can be correlated, coalesced into bigger pictures (crowd-sourced video), and observers can be better informed about what they are viewing.

It may seem like an odd world, but for those who master this technology, they can become part of the new, creative culture that uplinks its own content and applications and even innovates new system devices for this interactive media-engulfed lifestyle.
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCV media and analytics examples</td>
<td>opencv-examples.zip</td>
<td>302KB</td>
</tr>
<tr>
<td>Pixie Renderman Examples</td>
<td>pixie-examples.zip</td>
<td>173KB</td>
</tr>
<tr>
<td>DCT Example</td>
<td>dct-2d-example.zip</td>
<td>4KB</td>
</tr>
</tbody>
</table>
Related topics

- The OpenCV API for Computer Vision is well documented at [OpenCV.org](http://OpenCV.org) and in numerous books, such as *Learning OpenCV* (Adrian Kaehler and Gary Bradski, O'Reilly Media, 2013) and *Mastering OpenCV with Practical Computer Vision Projects* (Shervin Emami et al., O'Reilly Media, 2012). Note that OpenCV was implemented in C but has been updated with a C++ implementation that is encouraged for future applications. You can learn computer and machine vision theory from a variety of excellent academic texts, including *Computer Vision: Algorithms and Applications* (Richard Szeliski), *Computer Vision: Models, Learning, and Inference* (Simon J.D. Prince), and *Computer and Machine Vision* (E. Davies).

- The RenderMan language for scene descriptions and shading is well documented in many excellent texts, including *The RenderMan Companion* (Steve Upstill), *The RenderMan Shading Language Guide*, and *Advanced RenderMan* (Anthony Apodaca and Larry Gritz, Elsevier, 1999).

- Digital audio and video encoding are best implemented by referring to the standards (13818-1, 13818-2 for MPGE2 and ISO/IEC 14496-10 for MPEG4 and H.264/265), but many excellent summary texts are available, including *Digital Media Primer*, a great starter book that also covers Adobe Flash animation techniques, *Video Engineering* (Arch Luther and Andrew Inglis, McGraw-Hill, 1999), *A Practical Guide to Video and Audio Compression* (Cliff Wootten).

- Download the new H.265 standard, which offers several extensions to H.264, including 8K Ultra-High-Definition and 3D extensions, along with even better compression than H.264 (about twice as good).

- Many new wearable cameras are rapidly becoming ubiquitous, including GoPro, and adding AR-type features, such as Google Glass. Some cameras are designed for animals, such as National Geographic's Crittercam, now commoditized for dogs and cats with products like Eyenimal.

- Download the Android operating system (AOS) SDK to create and debug Java applications.

- Many computer vision researchers use MATLAB, but as you can see from my method for verifying my DCT and iDCT code for 2D spatial transformations, I prefer GNU Octave for teaching because it works well and is open source. I often also use GIMP, avconv (FFmpeg), VLC, and GStreamer when working on open source digital media applications and systems.

- Many interactive applications use OpenGL for rendering with polygons, but you'll also want to learn the photorealistic rendering that can be done with ray-tracing using Pixar RenderMan, with frames rendered by Pixie for open source RenderMan or rendering interactively using Blender. Although ray-tracing and polygon rendering have different histories and implementations, highly detailed polygon rendering (with fine polygons) is essentially indistinguishable from ray-tracing when polygons become a single pixel in size. However, to date, ray-tracing still creates more realistic-looking frames but at high computational cost. Try both: You'll see. This may change as graphics processing units start to support both ray-tracing and polygon rendering.

- Mobile perception systems for interactive augmented reality can use IMU on a chip products that can sense real-time acceleration and orientation, which, combined with VR such as Oculus Rift, becomes immersive and can be used for proprioceptive applications. Humans, of course, have way more than the five senses commonly known, especially when sensor fusion is considered.
• Gesture recognition has been made popular by Kinect but is also possible using the Intel Creative Camera and Perceptual Computing SDK. Interactive digital media systems and applications will no doubt integrate computer vision, gesture recognition, voice recognition, and many far less intrusive devices compared with common desktop input/output devices used in the past. Likewise, exploration of 3D data and models is becoming more common, along with the use of point cloud data, so PCL may be of interest for use with OpenCV.