Improving the Presentation of Technical Material in Video Talks using Post Production

Andy Gill[1], Garrin Kimmell[2], Kevin Matlage[1]
The University of Kansas[1], The University of Iowa[2]

Abstract

In this paper, we present our experiences using our image processing toolkit ChalkBoard and other video processing tools to post-process a pre-recorded conference talk. With inexpensive video cameras, video services like `youtube.com` and `vimeo.com`, and widely available and inexpensive video editing software, we expect this new media to be increasingly used as a mechanism to both promote research agendas and deliver technical content. In order to explore the use of such media in more detail, the Functional Programming group at KU recorded a technical talk and experimented with various post-processing tricks to enhance the value of the talk. Specifically, we fixed a common lensing issue in software, added small animations and pictures which matched the gestures of the actors, improved the visual quality of the slides being talked to, and experimented with a post-hoc zoom. Overall, the post-processing stage took considerably longer than anticipated, but did add perceivable value and impact to the final video.

Introduction

Volcanos, visas and other challenging logistical problems can stop an interesting conference paper from being presented to a willing audience. Teleconferencing, the obvious solution, is simply not yet a reliable option at typical academic conferences, which often have problems supporting reliable email access, much less the bandwidth required for a quality virtual presence. Some program chairs are relying on the next best thing; prerecorded videos of speakers talking in front of slides, and playing them over a projector to the conference audience. In this paper, we present our experiences using our image processing toolkit ChalkBoard[1,2] to help perform a lightweight post-production of a recorded conference talk, the technical challenges we faced, and report on how the final talk was received by our audience.

If an interested party looks hard enough, the internet, and especially `youtube.com`, contains many lecturers presenting quality technical material to anyone that is willing to listen. We think this is the future of technical evangelism, for good or bad, where ideas are condensed almost into sound bites and presented in short, entertaining videos. Bill Nye[3], Johnny Ball[4], and other eccentric TV personalities have been doing this for many years. With good props, the actor can introduce the intuition behind complex ideas and give an interested novice an entry point into more rigorous presentation mediums, like journal articles and conference presentations. With the boundaries of science ever expanding, being able to comprehend the interesting issues in a related discipline often relies on finding a suitable apostle.

Special Feature

At the University of Kansas, the Functional Programming group had grand plans to use what we know and research (computer language technology) to help us enhance simple videos of whiteboard talks. Unfortunately, before recording and producing these smaller talks and crafting our techniques, we found ourselves producing a 30 minute special feature. Here is our story.

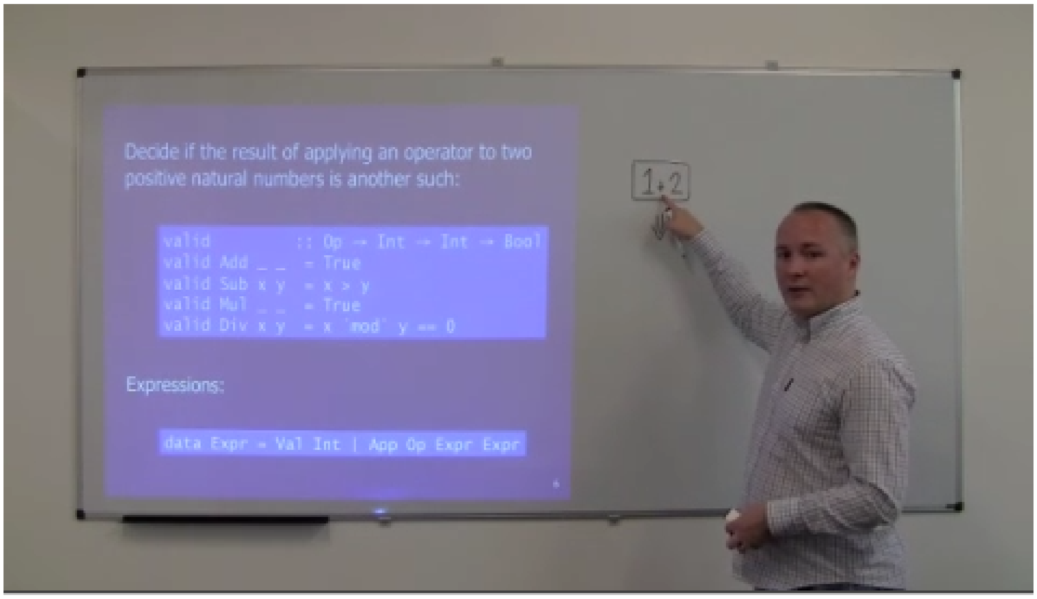
For over 20 years, the Partial Evaluation and Program Manipulation (PEPM) Symposium/Workshop series has been bringing together researchers and practitioners working in the areas of program manipulation, partial evaluation, and program generation. In Fall 2009, our research lab at KU had recently hosted an international programming contest[5] with over 800 teams taking part. We had made novel use of program manipulation and program generation to help set up a reliable infrastructure for the contest. It seemed like a perfect fit for a timely PEPM paper. Unfortunately, PEPM in early 2010 was located in Madrid, Spain, and none of our authors could make it. We did not submit a paper solely on the grounds of logistics, and not academic considerations. Though the result was not earth-shattering, we believed it would be of interest to the audience, and provide some light relief from some of the more technical papers.

Around the same time, we had been working on some basic video processing technology. We had developed a customized language for expressing and dynamically processing images, called ChalkBoard. We also developed a compiler for ChalkBoard that compiled this language in a way that allowed standard off-the-shelf graphics cards to quickly process complex images. With the ability to quickly process a single image comes the possibility of chaining together a workflow that can process small movies and perform near real-time effects like alpha blending, zooming, animation overlays, and picture in picture mixing. All of this technology is commercially available, but we had a prototype open source *scriptable* variant, where we *program* edits using a programming language, not a GUI.

So we contacted the chairs of PEPM, John Gallagher and Janis Voigtländer, and explained how we had some interesting research experience to present in the context of PEPM, had genuine logistical challenges, and furthermore we were developing some language-based video processing technology, and the language aspect of this video processing would also be of interest to the PEPM audience. Together we crafted a compromise. There would be a special feature in the PEPM program; a video talk premiering new movie processing technology that our team was developing.[6] This paper is the technical story about the design, implementation and presentation of this PEPM special feature.

Observations on Recording a Technical Talk

Our chosen model of interaction with the audience is a speaker (actor) with a writable board (canvas) and projected slides (slides). Figure 1 gives a single frame from a classic example of these three components being used well in a video talk given by Graham Hutton[7]. This talk was recorded in a single sitting, using a fixed HD video camera with a built-in microphone, a video projector, and an experienced and well prepared speaker. The potential for dissemination of technical information is huge; this video was actually viewed over 60,000 times. We want to start with this video format, and make some observations and suggestions for better use of post-processing.



Decide if the result of applying an operator to two positive natural numbers is another such:

```
valid          :: Op → Int → Int → Bool
valid Add _ _  = True
valid Sub x y  = x > y
valid Mul _ _  = True
valid Div x y  = x `mod` y == 0
```

Expressions:

```
data Expr = Val Int | App Op Expr Expr
```

Figure 1: Example online talk

The first observation is that **the scene is static**. Even though the actor, the slides and the canvas change, there is no movement or enhanced focus. This means that all three components share the same stage. The slides only form a small part (approximately a quarter) of the real estate of the video image, and any writing on the whiteboard needs to be big and clear enough to be read at a distance. Presenting the video full screen on a higher resolution monitor mitigates this, but we want to be able to present using a projector, which typically offers more mundane resolutions. Nevertheless, the actor's gestures, which give the video life, come across well in this chosen box. The proposed solution is that such video presentations need a basic zoom capability to focus in on the slides, or the writing, when necessary. This has to be done in a way that the actor's gestures are not compromised, because this often is part of the message. Picture-in-picture, and slide/speaker flipping are alternatives, but a good post production process with zooming can really enhance the experience.

The second issue is **the poor quality of the projected slides**. A projector is used to display the image, which is then captured by the camera. There has to be a better way to import slides into the movie without first projecting them. The proposed solution is to provide the ability to import slide images directly into our image processing technology. This is not as easy as it sounds; what happens if the actor is pointing to something on the slides? The actor's hand should not be obscured, making the presentation appear completely artificial.

The third issue is more of an opportunity. We have complete editorial control over the final presentation, and given the right tools, **we can add any effect we want to help tell our story**. Given the basic framework, can we add instructive animations that support our technical narrative? Can the actor interact with these virtual renditions? **There is considerable scope here for innovation in education**. There are also some minor issues that could be addressed with post production; for example, in this video, there is slight a bowing along the top and bottom of the whiteboard in the middle of the frame. We would want our processing technology to correct these issues algorithmically.

Filming

The allocated time for our presentation was around 30 minutes, which was longer than we had hoped to prepare for our premier, but we wanted to see what was possible. We executed the following game plan for recording our talk:

- We prepared a set of around 30 slides ahead of time; this was a conference style presentation. In a real sense, these slides were our storyboard as used in traditional filmmaking.
- A small number of these slides were left blank, with the intention that they would be replaced with animations in post-production. We scribbled a rough draft of the animation on an easel pad, out of view of the camera, to aid gesturing.
- We had successfully experimented with with automatic matte detection technology, and were confident we could extract our actors, if needed, from the image.
- The three actors (Andy Gill, Garrin Kimmell, Kevin Matlage) would each give different parts of the talk, in somewhat of an ad-hoc order.
- Taking advice from a sound professional[8], we used a clip-on microphone.

The filming of our special presentation was done in a single afternoon over about four hours. We used what we had available for filming: a 2002 Canon DV video camera. There were many stops and starts, coordination pauses, and even possible candidates for blooper reels. Furthermore, there is something fundamentally unnatural about interacting with an animation that does not exist yet, and we all found it challenging to pull off. Finally, giving a presentation to a camera is harder than a live audience; there is no feedback like eye contact to gauge the level of interest or comprehension.

Post Production

After recording the talk, we ended up with approximately 90 minutes of raw footage. Figure 2 gives our basic workflow for post production to the finished product. We used the camera to get DV footage, which **iMovie** accepted as input. Inside **iMovie** we edited our footage into a watchable presentation with the correct length, slide order, and overall narrative. This footage was without animations. Unlike most of our workflow, **iMovie** is not open source, but it is available on any recently purchased Apple laptop.

At this point, we split the recording into video and sound using the popular open source `ffmpeg` tool. `ffmpeg` give our ChalkBoard tool a stream of unconnected images (frames from the edited footage), allowing us to add a number effects using ChalkBoard. First, we warped the image to make the board straight. Second, we superimposed the slides directly over those projected in the image, scaled to the correct size. Third, we added a number of animations, with the goal of improving comprehension and impact. At post-processing time, we can time the changes in (for example) satellite thrust and velocity to match the actor's gestures. Fourth, we added a zoom effect to show *only* the slides, the whole scene, or anything in between. Unfortunately, we did not have sufficient time to use this ability as much as we would have liked – this could have been used to edit over various pauses and discontinuations. For future productions, we will certainly complete our zoom facility. A final feature we implemented but did not use was alpha matting, which would allow the actor to point to the superimposed slides. Again, we will certainly use this feature in the future.

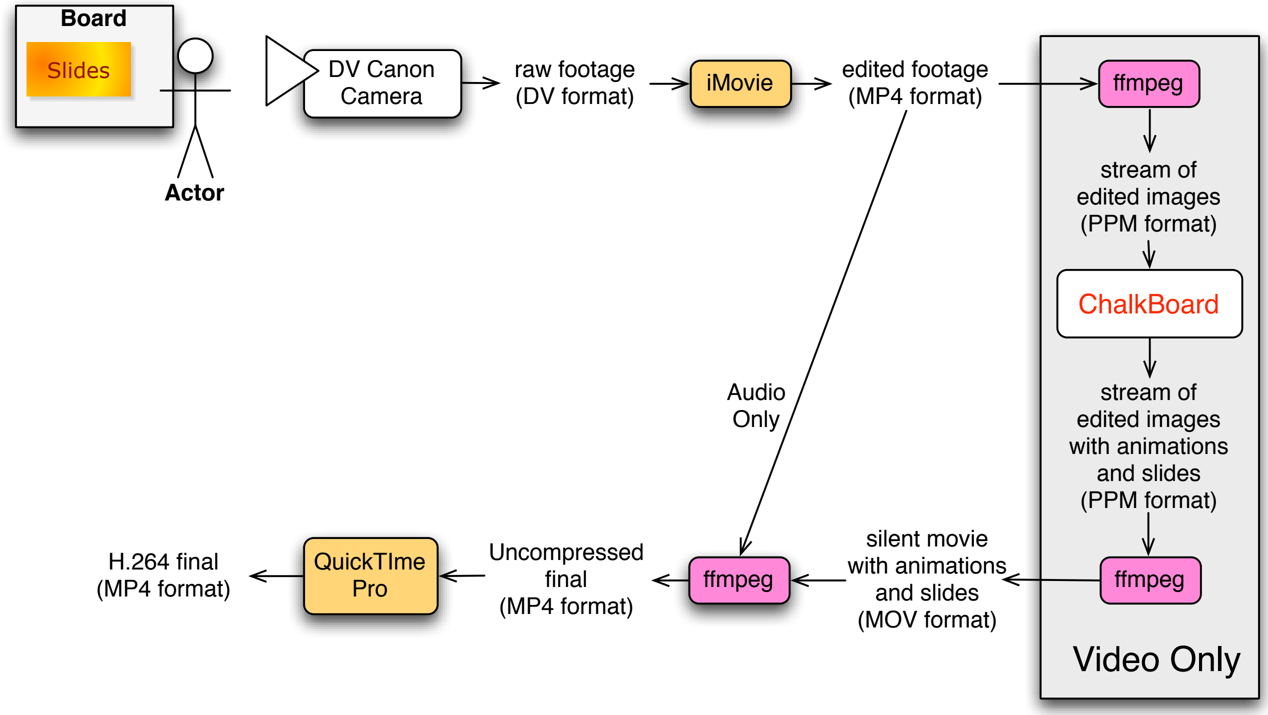

Board
Slides
Actor
DV Canon Camera
raw footage (DV format)
iMovie
edited footage (MP4 format)
ffmpeg
stream of edited images (PPM format)
ChalkBoard
stream of edited images with animations and slides (PPM format)
ffmpeg
Video Only
Audio Only
silent movie with animations and slides (MOV format)
ffmpeg
Uncompressed final (MP4 format)
QuickTIme Pro
H.264 final (MP4 format)

Figure 2: Post-production workflow

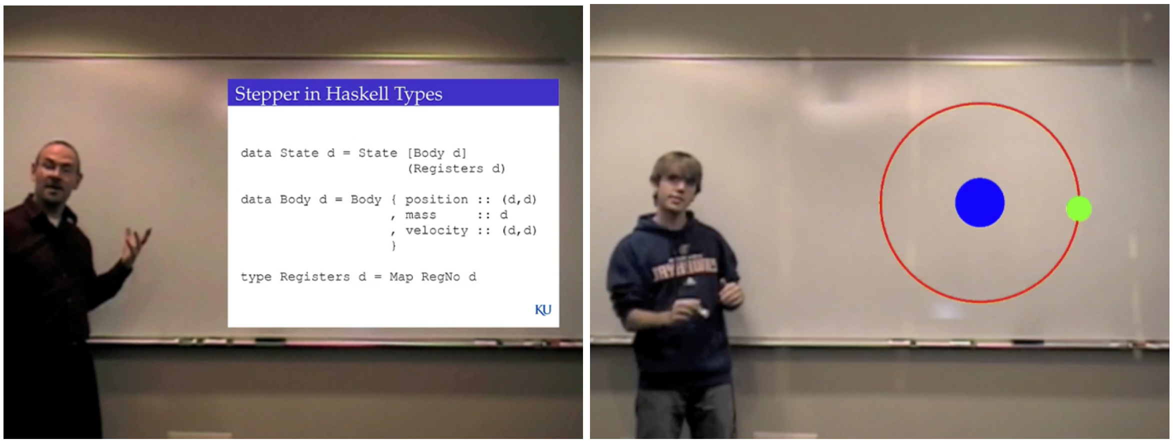
We used a small domain specific language to express each of these additions. As an example, for the slide superimposition, we had:

```
slideTable :: [(String, Int, Int)]
slideTable = [ ("slides/final-0.png", 0, 312),
               ("slides/final-1.png", 312, 2088),
               ("slides/final-2.png", 2088, 4877),
                ...
```

Slide `final-0.png` was used for frames 0 to (just before) 312, `final-1.png` was used for frames 312 to (just before) 2088, and so on. We used similar tables for our custom animation language [9] to build the animations. As an example of our animation language, a change in orbit (called a Hohmann transfer) was coded using:

```
hohmannAnim :: [Active (Board (RGBA -> RGBA))]
hohmannAnim = [ -- show earth
                earth,
                -- add orbit path
                for 3 (color red (intro 2 (satPath o1))) `over` (snap 1 earth),
                -- add orbiting satellite
                color green (intro 2 (satMove 5 o1)) `over` ...
```

In this case, a set of animations are stacked and set up to play one element at a time. So first we have the earth, then a red orbit around the earth, etc. This animation language component worked well during post-production, and developing the animations was a nominal part of the overall cost.



Stepper in Haskell Types

```
data State d = State [Body d]
                     (Registers d)

data Body d = Body { position :: (d,d)
                   , mass     :: d
                   , velocity :: (d,d)
                   }

type Registers d = Map RegNo d
```

Figure 3: Frames from movie

The resulting images were spliced back together into a (silent) movie, and the sound component was added back. The ChalkBoard process only modified frames, and did not add any new frames, allowing the actors gestures and voices to match. The result was a QuickTime movie, about 300M in size. In all, post production took about three days, and our ChalkBoard tool required roughly eight hours to process 30 minutes of video. Figure 3 gives two frames from the final movie. One is with a slide, the other with some orbital animation in progress.

Screening

The screening was relatively well received by the PEPM audience, though some people said that it "felt" like a videotaped talk. The first time an actor drew on the board and an animation appeared, the audience laughed. This was not the intended reaction, but didn't detract from the message. There was also a feeling of discontinuity between the three actors, and in the future a single actor should be used. We actually had intended to have actors interact during the video, which may have addressed this criticism, but we only had one clip-on mic. Even with more equipment were unsure how to incorporate multiple channels of audio into our workflow. Finally, in practice, the technique of "writing on the wall" did not work as well as intended, and was actually found to be distracting. One piece of feedback said we should consider using a magic "wand" as an alternative to the marker, as gesturing looks more natural with the animations than writing does.

There were also some technical issues with presenting the recorded video over a projector, perhaps because we had used a high bitrate in our talk. Sound could be an issue, but this was a smaller (<50 people) audience and the standard sound options were adequate. After the talk was presented we took questions (over skype, of course), and there were a number of questions about both the technology we discussed and the technology using to produce the talk. In summary, we think the presentation worked well, and we hope to continue using these post-processing techniques and experimenting with novel presentations of technical content.

Acknowledgments

Thank you to John Gallagher and Janis Voigtländer for allowing us to try out our ideas on the PEPM audience and providing feedback about the talk. Thank you to John Seguin of Seguin Sound for suggestions to improve sound quality and to Graham Hutton for letting us pick on his video.

Bibliography and Endnotes

1. http://www.ittc.ku.edu/csdl/fpg/Tools/ChalkBoard
2. Kevin Matlage and Andy Gill, "ChalkBoard: Mapping Functions to Polygons", In Proceedings of the 21^{st} Symposium on Implementation and Application of Functional Languages, Sep 2009.
3. Bill Nye, also know as the "The Science Guy". http://www.billnye.com/
4. Johnny Ball, BBC TV host of the 80's math show "Think of Number" and "Think Again".
5. http://www.icfpcontest.org/
6. http://strategoxt.org/PEPM10/SpecialFeature
7. Online Talk, "The countdown problem", Graham Hutton, The University of Nottingham, guest lecture in Microsoft's series on Programming in Haskell, used with permission of the author. http://ecn.channel9.msdn.com/o9/ch9/4/1/2/4/0/5/C9LecturesFPGrahamHuttonC11_2MB_ch9.wmv
8. John Seguin, Seguin Sound. http://seguinsound.com/
9. Kevin Matlage and Andy Gill, "Every Animation Should Have a Beginning, a Middle, and an End", Submitted to Trends in Functional Programming, June 2010.

Biographical Information

ANDREW (ANDY) GILL
Assistant Professor
Information Technology and Telecommunication Center
Department of Electrical Engineering and Computer Science
The University of Kansas
andygill@ku.edu

GARRIN KIMMELL
Postdoctoral Researcher
Department of Computer Science
University of Iowa
gkimmell@cs.uiowa.edu

KEVIN MATLAGE
Graduate Student
Information Technology and Telecommunication Center
Department of Electrical Engineering and Computer Science
The University of Kansas
kmatlage@ku.edu