Virtual Videography

RACHEL HECK, MICHAEL WALLICK, and MICHAEL GLEICHER
University of Wisconsin, Madison

Well-produced videos provide a convenient and effective way to archive lectures. In this article, we offer a new way to create lecture videos that retains many of the advantages of well-composed recordings, without the cost and intrusion of a video production crew. We present an automated system called *Virtual Videography* that employs the art of videography to mimic videographer-produced videos, while unobtrusively recording lectures. The system uses the data recorded by unattended video cameras and microphones to produce a new edited video as an offline postprocess. By producing videos offline, our system can use future information when planning shot sequences and synthesizing new shots. Using simple syntactic cues gathered from the original video and a novel shot planning algorithm, the system makes cinematic decisions without any semantic understanding of the lecture.

Categories and Subject Descriptors: H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Video (e.g., tape, disk, DVI)*

General Terms: Design

Additional Key Words and Phrases: Automated camera management, video production, attention modeling, computational cinematography

ACM Reference Format:

Heck, R., Wallick, M., and Gleicher, M. 2007. Virtual videography. ACM Trans. Multimedia Comput. Commun. Appl. 3, 1, Article 4 (Feb. 2007), 28 pages. DOI = 10.1145/1198302.1198306 http://doi.acm.org/10.1145/1198302.1198306.

1. INTRODUCTION

A "traditional" lecture, where an instructor presents in front of a chalk- or markerboard, is a common method for presenting information. Archiving these events on video can be valuable for those who did not attend or for someone who wants to review lecture contents. Unfortunately, simply recording a lecture with a fixed, unattended camera does not produce a video with the production value that viewers expect. Effective videos exhibit cinematographic qualities that in the past have required the skills of an expensive, intrusive videographer or video production crew. Our goal is to provide inexpensive, yet effective, archiving of traditional lectures, without disrupting the lecture.

In this article, we present an approach to lecture-video production that uses unattended, stationary video cameras and microphones to record the event and a novel, automatic editing system which processes this data to produce a new video exhibiting many of the properties of a videographer-produced

This work was supported in part by NSF Grants CCR-9984506, IIS-0097456, and IIS-0416284. M. Wallick is funded by a Microsoft Research Graduate Fellowship.

Authors' address: R. Heck (contact author), M. Wallick, and M. Gleicher, Computer Sciences Department, University of Wisconsin at Madison, 1210 West Dayton Street, Madison, WI 53706; email: heckr@cs.wisc.edu.

one. We call our approach *Virtual Videography*, as it simulates many aspects of a production crew. Virtual Videography uses computer vision and signal processing methods to gather information about the lecture, a planning algorithm to choose appropriate shots, and image synthesis methods to generate new images from the original source footage. With little to no human intervention, the system produces videos containing a variety of different shots and visual effects that serve to guide the viewer's attention and maintain visual interest, without violating the rules of cinematography.

A videographer relies on knowledge of the art of filmmaking and an understanding of what is happening in a scene to inform decisions. In our efforts to create a virtual videographer with these same characteristics, four key insights emerged. First, *syntactic cues*, such as marks on the board and motions of the instructor, can inform editing decisions. Since current techniques cannot reliably extract the semantics of a lecture, the ability to make editing decisions based on easily identifiable cues makes Virtual Videography viable.

Second, by processing the video *offline*, we not only gain more time to perform analysis and synthesis, but more importantly, the ability to use more information to make decisions. We can process large windows of time to provide robustness to simple image analysis algorithms, look into the future to determine how to best plan shots, and use future images to synthesize visual effects. While depending on offline processing precludes the use of our technique in live broadcasting, these advantages are helpful for lecture-video production.

Third, we pose the editing problem as a *constrained optimization over space and time*. A good video not only frames individual shots in a pleasing manner, but also combines them effectively. As such, the quality of a video cannot be determined by evaluating each shot in isolation. We therefore encode composition and editing goals in an optimization objective and solve for a whole *sequence* of shots.

Fourth, rather than relying on controlling a camera to acquire desired viewpoints, *image synthesis* can be used to create the views deemed necessary by Virtual Videography. View synthesis allows us to avoid using special-purpose movable cameras and to postpone editing decisions until after the lecture is captured.

These four insights allow Virtual Videography to produce lecture videos which exhibit the production value viewers expect. At the same time, the process is unobtrusive and does not require the instructor to alter his or her lecture style for the camera; setup requires no special equipment and works without calibration in a wide variety of classrooms; and editing requires little to no human input. Furthermore, the system can produce multiple videos of the same lecture, each tailored to a specific need or viewer preference. In particular, Virtual Videography can produce videos of varying aspect ratios from the same source footage, making better use of the available screen space on different devices.

Our emphasis on automatically creating inexpensive videos of traditional lectures places a number of constraints on our system: We capture a limited amount of source footage, the system has no true understanding of the lecture content, and we place few requirements on the structure of the lecture. Yet, despite all these constraints, Virtual Videography is able to create videos that exhibit desirable properties of good video. The system may not compete with videographers with years of experience, human understanding, and special equipment, but it can create effective lecture videos with much lower expense and intrusion.

The remainder of this article is organized as follows. Next, a review of related work shows how other attempts to address similar problems serve to inform our work. We then give an overview of our approach in Section 3 and discuss how a concept called *region objects* allows a number of components to fit together to create our Virtual Videography system. The next four sections describe each of the components of our system: media analysis, attention modeling, computational cinematography, and image synthesis. Section 8 then provides some of the implementation details necessary when making an aspect ratio change. We conclude with an overview of our results and a general discussion of our technique.

2. RELATED WORK

While fixed, unattended cameras provide an easy method for capturing the visual and aural information of a lecture, early filmmakers learned that images from a camera in a fixed location do not provide enough visual interest or guidance to the viewer [Bordwell 1998]. This observation helped lead to the development of the art of filmmaking. Our desire to automatically produce lecture videos that use this art led us to develop the offline editing scheme of Virtual Videography. Our prior papers on the subject, Gleicher and Masanz [2000] and Gleicher et al. [2002], discuss the ideas behind Virtual Videography, but provide no actual implementation details. All of the examples in these papers were created by hand as a proof-of-concept. Additionally, region objects, a central part of the Virtual Videography scheme, is also described in one of our previous papers [Wallick et al. 2005].

Many other researchers have explored techniques for automating video production. One early system, Auto-Auditorium [Bianchi 1998; 2004], poses the problem of online video editing as switching between cameras and microphones instrumented throughout a lecture hall. We call automatic video production systems that use this online camera switching scheme *online camera management systems*. There are several notable systems of this type, including those described in He et al. [1999], Chen [2001], Rui et al. [2001], Steinmetz and Kienzle [2001], Machnicki and Rowe [2002], and Wallick et al. [2004].

As online camera management systems all operate online in real time, they can be applied in domains such as live broadcasting, where our offline approach cannot. However, being online places a number of restrictions on these systems that preclude their use in the lecture domain and limit the quality of their results. Because online systems can neither look into the future nor do substantial planning, they often make cinematic mistakes, such as generating a shot that is too short, and are incapable of using cinematography to *lead* the viewer. Additionally, online systems have a very limited amount of time in which to make decisions. All of the systems listed earlier work in slide-based domains, where slide changes provide a quick cue for choosing cameras in real time. Furthermore, each image in the final video is usually taken exactly from a physical camera located in the room. For example, these systems usually display an entire slide at one time and provide little help in focusing the viewer's attention on relevant parts of the slide.

Some researchers have also taken an offline approach to automated video editing. The system of Onishi et al. [2001] focuses the view on changing pixels in the source video, while using a concept similar to our region object as a way of grouping ideas on a blackboard [Onishi et al. 2000]. Wang et al. [2004] similarly use gesture tracking and recognition to help focus a virtual camera on interesting areas of a board and/or slide. Onishi et al. [2000] and Wang et al. [2004] demonstrate the utility of combining audio and video cues to determine where the viewer's focus should be directed, an idea we apply in our system. These systems differ from our own, as they place little emphasis on cinematography, shot planning, or attention modeling.

Some equipment has been designed to help capture the contents from lectures and meetings. Lee et al. [2002] and Culter et al. [2002] built special cameras intended to capture video of meetings. Additionally, equipment used to capture writing on boards [Saund 1999; He et al. 2002; Virtual Ink 2004] is available for commercial use. While these devices, particularly the whiteboard and blackboard capture equipment, may be used to enhance the results of our system, Virtual Videography is designed to work in any classroom with only a small number of common, inexpensive, portable cameras and microphones.

Still other researchers explore ways to annotate, index, browse, and summarize recordings [Teodosio and Bender 1993; Abowd 1999; Mukhopadhyay and Smith 1999; James and Hunter 2000; Syeda-Mahmood and Srinivasan 2000; Li et al. 2001]. While these systems do not focus on producing edited video, they do serve to inform and inspire Virtual Videography. One particularly relevant system explicitly uses syntactic knowledge about the ordering of lecture components to produce a video summary

[Russell 2000]. Our system also makes use of lecture heuristics, but we do not explicitly use the ordering of lecture components.

Many authors have considered the problem of determining appropriate camera views. Gleicher and Witkin [1992] first suggested using constrained optimization to position cameras for viewing objects. Drucker and Zeltzer [1995] and Christianson et al. [1996] extended this work to plan entire camera motions. Similarly, we use a novel shot planning algorithm that casts 2D camera movements as constrained optimization. However, our algorithm is designed to work at a larger time scale with a discrete number of choices.

Shot selection in 3D synthetic environments has been used in a number of tasks. Karp and Feiner apply 3D shot planning to presentation creation [1993], and He et al. [1996] plan shots for dialogs in virtual environments. These works illustrate ways to codify cinematic rules and make editing decisions automatically. The work of Bares and Lester [1999] explores automatic cinematography and editing in complex environments by considering issues such as occlusion. Pinhanez and Bobick's [1997] robotic camera operator interprets verbal commands from a director to automatically frame objects from a scene, while Girgensohn et al. [2000, 2001] present a tool for automating the editing process by selecting suitable clips from a collection and presenting them to a user for assembly. Virtual Videography explores a different domain from this earlier shot selection work, and we require our system to automatically determine what information is important to show the viewer at every point in time.

3. A FRAMEWORK FOR VIRTUAL VIDEOGRAPHY

The main challenge to automatically producing effective lecture videos is to recreate the skills of a videographer. Rather than recreate the senses and thought processes of human videographers, we cast the decision-making process in a computational framework. This section presents an overview of how our system approaches the problem.

A goal in videography is to direct the viewer's attention towards what is important [Katz 1991]. Therefore, a central portion of our system is a representation of potentially important objects in the scene. We use a concept called a region object, which is the smallest rectangular region of the board that contains a semantically linked group of writing. For example, a region object could contain a sentence, a diagram, a bullet list, or a single thought. Like Saund et al.'s ScanScribe, the ability to conceptually divide images into individual pieces makes processing, interaction, and reasoning more natural [Saund et al. 2003]. We have found that these region objects not only provide a useful implementation strategy, but also provide a way to cast filmmaking knowledge in a computational framework.

The region object is similar to concepts described in other lecture-based research, most notably the "written rectangle" concept of Onishi et al. [2000] and the "semantic learning objects" of Liu and Kender [2003], developed at the same time as our model. One notable difference between our region objects and these other models is the notion of a region lifespan, or progression of the writing over time. This temporal information helps to inform our attention model (described in Section 5) as to the importance of any area of the board.

Each region object encapsulates the location of related writing on the board (specifically, the position of the upper left-hand corner of the region, its width, and height), as well as three temporal events that succinctly describe the region object's lifespan:

—*Birth*. A region is born when the instructor begins to write its contents on the board.

—*Maturity*. A region is mature once the instructor has finished writing its contents on the board. At any point in time, there can be only one growing region, namely, a region that has been born, but is not yet mature.

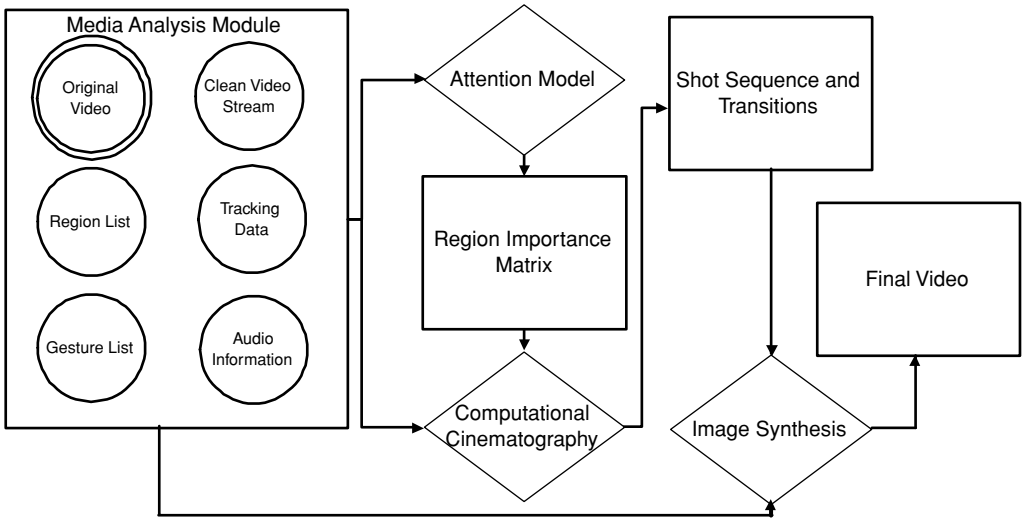


Fig. 1. A diagram describing the structure of the Virtual Videography system.

—*Death*. A region dies in one of three ways: The instructor erases all of the region's contents, the lecture ends, or the region merges with a newly created region. A merge occurs when the instructor adds new content to a region that has already matured; the old region becomes part of a new region containing the new content. A reference to the old region is kept so that the evolution of a block of writing can be traced. The decision to use merging to represent changes to a region was made for two reasons. First, using this technique, a region object can be represented compactly as its dimensions, its three lifespan numbers, and an optional reference to its parent region. Second, conceptually, adding text to a region is very similar to starting an entirely new region. In both cases, the audience is drawn to the new writing. But in the case of an expanded region, they are also drawn to the original writing. This tie is represented easily with the merging concept.

At any point in the lecture, there are some number of region objects on the board. These objects, as well as the region surrounding the instructor, describe parts of the original video that could contain important information. As such, each phase of the Virtual Videography system uses these region objects.

There are four phases in the Virtual Videography system. Media analysis identifies region objects and provides our system with simple events and properties of the original video data (Section 4). Using the set of region objects supplied by media analysis, the attention model builds an evolving probability for each region specifying the likelihood of this region being a focus of the audience's attention at any instant in time (Section 5). The computational cinematographer then determines the sequence of shots that should be used to conform to appropriate cinematic rules (Section 6). Finally, an image synthesis component constructs a final video by synthesizing views and visual effects (Section 7). See Figure 1 for a diagram of the Virtual Videography system.

4. MEDIA ANALYSIS

The media analysis phase of Virtual Videography gathers information from the original source video that can be used to make informed cinematic decisions. Currently, the system gathers very basic information from the video using simple methods. We have found this basic information sufficient for video editing, although more could be provided through enhanced media analysis. In particular, our media analysis implementation supplies: a clean plate video stream of the board (i.e., a video stream

Fig. 2. Creating a clean stream of the board. (left): original frame of video; (center): instructor removed; (right): frame in-painted using information from later frames.

Fig. 3. Regions found on a whiteboard and chalkboard.

where the instructor has been removed), a list of region objects, tracking data for the instructor, a list of pointing gestures made by the instructor, and a basic analysis of the audio. In this section, we briefly describe the information that is collected and how it is used by the system. This information can be gathered using any available techniques, but details for the methods we use are described in Appendix A.

First, the video is segmented into foreground (instructor) and background (board). The holes in the background, caused by the instructor and other obstructions, are in-painted with information from other times in the video, resulting in an approximation of a clean stream of the board. This clean stream has several uses, including identifying when writing appears, tracking the instructor, and creating special effects shots. Figure 2 shows the results of our segmentation.

Next, we identify region objects on the board. The regions are later assigned an importance and used to help frame shots in the video. Figure 3 shows the results of our region finding method on a whiteboard and chalkboard.

Tracking is then used to locate the instructor in each frame, ensuring that the instructor is properly framed in each shot. The instructor's proximity to and movement around a region object also affects its importance, as described in Section 5.

Next, gesture recognition determines if the instructor is preforming any one of a predefined number of gestures. We recognize three different gestures: pointing left, pointing right, and reaching up. We have found that these three gestures can help the attention model determine the importance of region objects.

Finally, audio analysis determines whether the instructor is speaking in any given frame. During the shot selection process described in Section 6, the system will avoid shots that focus on the instructor when he or she is not speaking.

5. ATTENTION MODEL

To make informed decisions about which shots to show throughout the video, we need to model where the instructor is directing the audience's attention. As described in Section 3, region objects and the area surrounding the instructor encapsulate parts of the video that could contain important information. Without any true semantic understanding, our attention model determines which region objects are likely to be a focus in the lecture. While the instructor's importance probability is not directly modeled in our system, it is assumed that the instructor is always important enough to be on the screen.

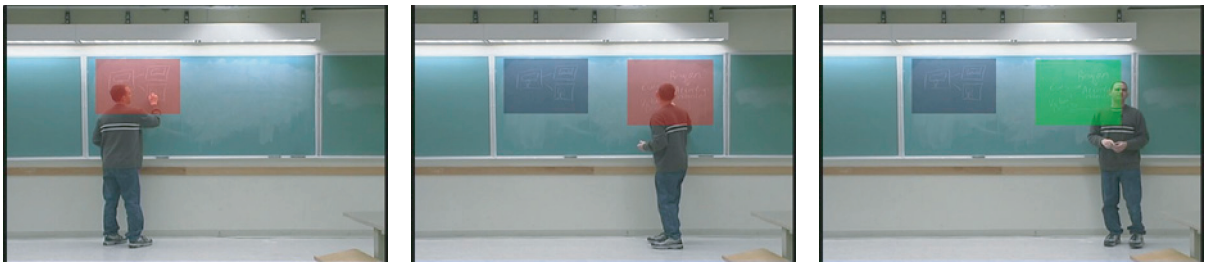


Fig. 4. The images above show the results of our attention model over time. Regions in red are considered highly important, those in green are moderately important, and those in blue are not likely to be important.

Though an instructor uses a wide variety of (often subtle) cues to direct attention towards areas of the board, there is a small set of simple syntactic events that can inform our attention model:

(1) A region is not important if it has not been born or has died;
(2) a region is likely to be important if it is growing or has just reached maturity;
(3) a region will not remain important for too long or too short a time;
(4) a region is more likely to be a focus when a new region is born nearby; and
(5) a region is more likely to be a focus if the instructor points to it.

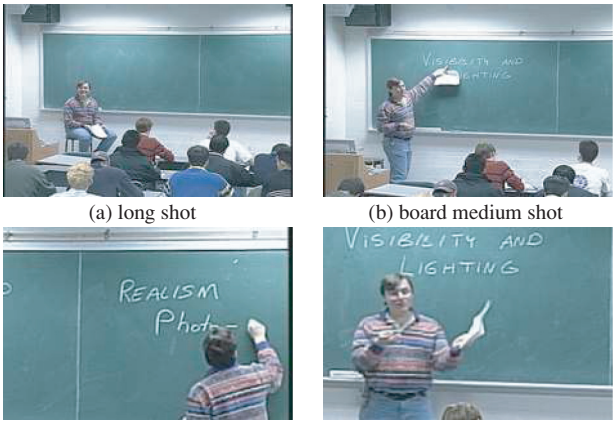
Using these heuristics, our attention model determines the likelihood that each region object is important at every point in time (see Figure 4). In particular, we compute the probability of region r being a focus at frame f recursively by evaluating the function $p(r, f)$. On each line of the definition of $p(r, f)$, the number in parentheses refers to the aforementioned heuristic encoded by this line:

$$p(r, f) = \begin{cases} 0 & \text{if } f < b_r \| f \geq d_r \quad (1) \\ 1 & \text{if } b_r \leq f \leq m_r \quad (2) \\ min \begin{pmatrix} 1, & p(r, f-1) \times (1-D) + \\ & c(r, f) \times C + \\ & g(r, f) \times G \end{pmatrix} & \text{otherwise} \quad \begin{matrix} (3) \\ (4) \\ (5) \end{matrix} \end{cases}$$

In the preceding equation, b_r, m_r, and d_r are the birth, maturity, and death times of region r, respectively. D is a real number between 0 and 1 that controls how fast a region's probability should diminish over time. The Boolean function $c(r, f)$ returns 1 if and only if there is a region whose birth is at frame f and that is less than some closeness threshold away from region r. The parameter C is a real number between 0 and 1 that specifies how much the creation of a nearby region should affect the probability of region r. The function $g(r, f)$ returns 0 if there are no pointing or reaching gestures made toward region r at frame f. Otherwise, $g(r, f)$ returns a value between 0 and 1 representing the system's confidence in the gesture. Our system computes the confidence of a gesture based on both the number of frames over which the gesture was recognized and the distance of the possible target region from the instructor. Finally, the parameter G is also a real number between 0 and 1, and this specifies to what extent a pointing gesture with 100% confidence should affect the probability of region r. In practice, we have found that setting the closeness threshold to 70 pixels, $C = .15$, $G = .45$, and $D = .0011$ works well with our data (as given in Appendix B). We have also noted that these parameters are not very sensitive. For example, changing C to .2 does not noticeably affect the overall results.

6. COMPUTATIONAL CINEMATOGRAPHY

The computational cinematography component of our system does the job of the cinematographers, directors, camera operators, and editors in a conventional video production crew. First, virtual cameras



(a) long shot (b) board medium shot

(c) board closeup shot (d) instructor closeup shot

Fig. 5. The four basic shot types.

analyze the available information to frame effective shots throughout the video. These shots are then assembled into a *shot sequence graph* that encodes all of the possible shot sequences. Next, we use an optimization strategy to find that shot sequence which minimizes an objective function designed using the art of film and knowledge of the lecture domain. Finally, transitions are added between appropriate shots to smooth the virtual camera movements. This section discusses each of these steps in more detail. For simplicity, this description assumes that the source and target aspect ratios are the same. Section 8 will address aspect ratio changes.

6.1 Virtual Cameras

In a professional camera crew, camera operators move within the space of an event, framing good shots of the subject. Similarly, our system uses virtual cameras to frame good shots within the original source footage. When assigned a segment of time in the video, a virtual camera provides a *shot description* for a shot found within the source footage. A shot description consists of a subregion of the source video, specified by the position of the upper-left corner and width of the framed region. For moving shots, these three parameters are a function of time. The system uses an extensible set of virtual cameras, each of which creates a different type of shot. For example, some cameras frame good shots of the instructor, while others attempt to produce tight shots of important regions of the board. Our system uses four basic virtual cameras (see Figure 5), each designed to mimic a "standard" shot as used in cinematography [Katz 1991]. Since it is assumed that the instructor is always important enough to remain on screen, these four shot types always keep the instructor on screen:

—*Long shot: Figure 5(a)*. A long shot is an unmoving shot showing as much of the lecture room as possible. The best long shot for any segment of video is the original camera data.

—*Board medium shot: Figure 5(b)*. A medium shot of the board is an unmoving shot framing the instructor and the set of moderately important region objects, that is, regions with an importance of .75 or above, throughout the segment of video, surrounded by a small border.

—*Board closeup shot: Figure 5(c)*. A closeup shot of the board is an unmoving shot that closely frames the instructor and the highly important regions, namely, regions with an importance of .9 or above, throughout the segment of video.

—*Instructor closeup shot: Figure 5(d)*. A closeup shot of the instructor is similar to a closeup shot of the board except that no region objects are intentionally framed by the shot.

Shots created by other virtual cameras derive from one of these four types.

Often, it is not possible for a virtual camera to find a good shot of its specified type. For instance, if there is no important writing on the board during a segment of video, a camera cannot create a good shot that frames important regions. As such, not all virtual cameras can generate shot descriptions for all segments of the video. Even when a virtual camera can supply a shot description, the characteristics of a shot described by a virtual camera may vary. For example, if there is a lot of activity on the board in a short period of time, there may be more than one region of high interest, making the best closeup shot for this segment include a large area of the board. The shots will be evaluated later to find the best shot sequence from the given options.

6.1.1 *Virtual Tracking Camera.* Tracking shots follow a moving object. Good tracking shots require skillful camera operation, as they are more complicated than simply keeping the subject in the center of the frame. A good tracking shot should consist only of smooth motions in a single direction that accelerate and decelerate gradually to avoid jarring the viewer, while maintaining the correct apparent motion of the subject (e.g., if moving from right-to-left, the subject should appear to be on the righthand side of the frame at the beginning of the tracking shot and on the lefthand side of the frame at the end) [Cantine et al. 1995]. Many systems use an expensive, complex motorized tracking camera that follows the speaker [Burns et al. 1998]. Yet, even the best motorized camera cannot look into the future to plan the best possible smooth tracking shot. We use a novel method for producing tracking shots exhibiting all of the desirable characteristics.

Since the subject of a tracking shot in our domain (i.e., the instructor) moves predominantly in the horizontal direction, we consider only horizontal tracking shots. By forcing the camera to have a smooth path with the correct acceleration parameters, we can generate good tracking shots of the instructor. To specify a tracking shot that begins with the camera at x_1 and ends with the camera at x_2, the system computes the center of the image $x(t)$ for every time t in the video segment and the width w of the image throughout the shot. Assuming ranges of time from 0 to 1, $x(0) = x_1$, and $x(1) = x_2$, we choose $x(t)$ to be a $c(1)$-continuous function that accelerates from zero velocity for some duration a, holds a constant velocity, and then decelerates for the same duration a, reaching zero velocity at $t = 1$:

$$d = \begin{cases} \frac{t^2}{2a-2a^2} & \text{if } t < a \\ \frac{a}{2-2a} + \frac{1-4a}{2-2a} * \frac{t-a}{1-2a} & \text{if } a \le t \le 1-a \\ 1 - \frac{(1-t)^2}{2a-2a^2} & \text{if } t > 1-a \end{cases}.$$

$$x(t) = (1-d)x_1 + d\,x_2$$

This function can be graphically shown as a pair of splined parabolic arcs (see Figure 6).

Given the left and right edges of the bounding box of the subject, $s_l(t)$ and $s_r(t)$, and the width of the original video, $width$, we write the following linear constraints which ensure that the shot contains the subject and is within the source footage:

$$\begin{aligned} 0 \le\ & x(t) - \tfrac{1}{2}w \ \le s_l(t) \\ s_r(t) \le\ & x(t) + \tfrac{1}{2}w \ \le width \\ & w \ge 0. \end{aligned}$$

The best tracking shot for a segment of video is the tightest shot, namely, the smallest w, that meets these constraints. We can compute this by solving a linear program over the variables x_1, x_2, and w.

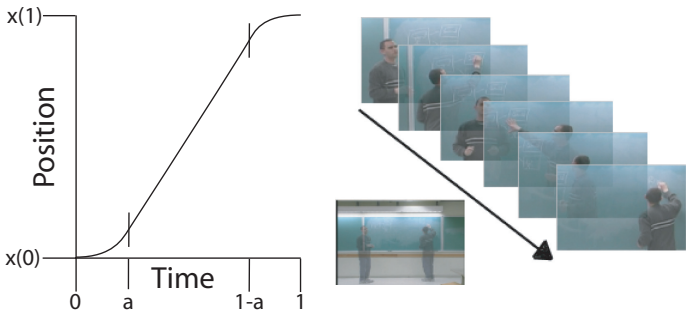


Fig. 6. On the left is a plot of the function which controls the motion of the tracking camera. Notice how the function starts with a velocity of zero, accelerates over a set duration a, holds a constant velocity, and then decelerates for the same duration a. On the right is a depiction of a tracking shot generated by our system. The image below the arrow is a composite of the first and last frames of the original data over which the tracking shot spans. Above the arrow, we can see the progression of the shot as the camera moves from the left side of the original video data to the right.

Because the subject is always in the source image, there is always a solution to this constrained optimization (at worst, $w = width/2$).

This linear system does not, in general, generate tracking shots that maintain apparent motion. To include apparent motion in the optimization, we add two new parameters (e_1 and e_2), two new constraints, and a new objective function to the linear system. The constraints are dependent on the dominant direction of the subject's motion:

$$e_1 = \begin{cases} s_r(0) & \text{if moving left} \\ s_l(0) & \text{if moving right} \end{cases}$$

$$e_2 = \begin{cases} s_l(1) & \text{if moving left} \\ s_r(1) & \text{if moving right} \end{cases}.$$

The optimization function changes similarly. If moving left, we should minimize a weighted sum of w, $x(0)+\frac{1}{2}w-e_1$, and $e_2-x(1)+\frac{1}{2}w$; if moving right, we should minimize a weighted sum of w, $e_1-x(0)+\frac{1}{2}w$, and $x(1)+\frac{1}{2}w-e_2$.

Generally, it is only worthwhile to use a tracking shot if the motion permits a tighter shot than would otherwise be possible with a fixed closeup shot. As described in Section 6.2.2, this fact is taken into account when evaluating possible shot sequences for the final video.

6.1.2 *Visual Effects Virtual Cameras.* Visual effects virtual cameras produce shots that cannot be synthesized by simply cropping frames from the original camera data. Visual effects are particularly useful for lecture video, as the instructor often obscures areas of importance. Our system produces two different types of special effects shots to offset this problem: *ghosting* shots and *picture-in-picture* shots. A ghosting shot is one where the instructor appears partially transparent, allowing the board and its contents to be seen through his or her body (see Figure 7); the important information is made visible, without losing the context of the rest of the lecture. This type of shot can be accomplished easily by setting α of the original video data to be less than 1 and compositing it over the clean video stream. If the instructor is ghosted while writing on the board, the writing that can be seen through the instructor's body depends on whether the clean video stream was filled with data from the future or the past (see Section 4). If the clean video stream is filled backwards, writing will appear on the board as soon as this area is blocked, often before the instructor actually writes it. While some may find this distracting at first, filling the clean stream backwards can be beneficial, as the viewer does

Fig. 7. Example of a ghosting shot; (left): original frame; (middle): clean stream frame; (right): ghosted frame.

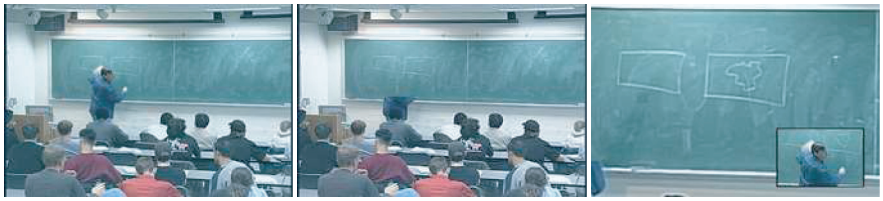


Fig. 8. Example of a picture-in-picture shot; (left): original frame; (middle): clean stream frame; (right): final image.

not need to wait for the instructor to move from in front of the text to actually read the writing. Virtual Videography uses three different ghosting virtual cameras corresponding to three of the four basic shot types. There is not a virtual camera for producing ghosted instructor closeup shots, as the focus of these shots is the *instructor*.

A picture-in-picture shot fills the screen with a completed, still image of the important writing taken from the clean video stream. An inset in the lower right corner tracks the professor's upper body so that the viewer can see his or her interactions with the important region without getting disoriented (see Figure 8).

Because ghosting and picture-in-picture shots address the same problem, each of the videos produced for this article use only one of the two effects types.

6.2 Choosing a Shot Sequence

While virtual cameras generate the best shots possible over a window of time, any individual shot may violate a number of cinematic principles when placed in the context of the rest of the sequence. For example, a shot sequence made up entirely of long shots and closeup shots does not provide much variety in the final video. As such, the quality measure for any shot sequence is dependent on the entire length of the sequence and cannot be determined by analyzing isolated shots.

Given a shot sequence, we assess its quality using an objective function. The problem of determining the best shot sequence for an entire video is thus a problem of constrained, combinatorial optimization. As there is a compact way to represent the possible shot sequences for a video in a graph, we pose this problem in a graph searching framework, allowing us to use a branch-and-bound technique to find the optimal shot sequence.

This section will start by describing how to build and understand the shot sequence graph. Then, we will give details about our objective function, explaining how we encode rules of cinematography and classroom heuristics. The end of this section will give the mechanics of searching the graph.

6.2.1 *Constructing the Shot Sequence Graph.* To facilitate the search for the best shot sequence found within the original data, we build a discrete structure that represents all possible shot sequences within the video. First, the original video sequence is divided into pieces of size l (information on choosing the value of l will be provided later in this section). For every segment, each virtual camera

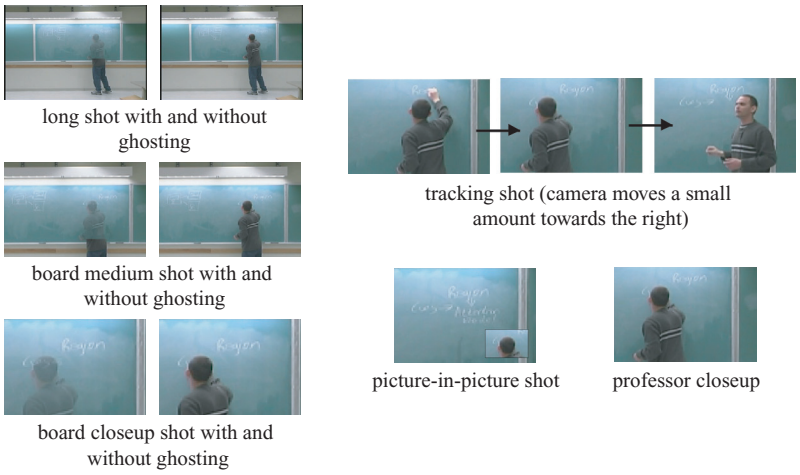


Fig. 9. For any segment of video, several different shots can be created by the virtual cameras. This figure shows the nine different shots that can be generated for one specific moment in a video. Our system encodes these shots into a shot graph that can be searched for the best possible shot sequence.

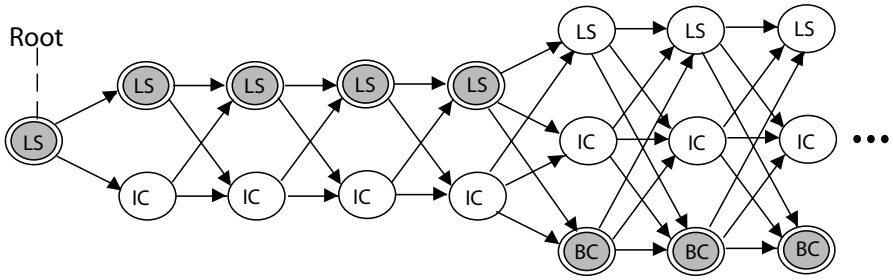


Fig. 10. A shot graph containing three different types of shots: LS = long shot, IC = instructor closeup, and BC = board closeup. The chosen shot sequence is shown by shaded nodes. In this example, a five-node long shot was chosen to begin the video and is followed by a closeup of the board.

can supply a shot description for a maximum of c possible choices, where c is the number of virtual cameras (see Figure 9). We represent these shot descriptions as graph nodes. Second, directed edges are used to attach nodes which represent shots that can be strung together to produce valid shot sequences. For instance, a node n that represents a shot beginning at frame h can only be followed by one of the j nodes which represent possible shots beginning at frame $h + l$. There will be exactly j outgoing edges for node n and each will lead to one of the j nodes that can come next in the shot sequence. Figure 10 shows the structure of a shot sequence graph.

To ensure that there is an available shot for all moments in the video, at least one of the virtual cameras should be designated as a default camera. A default camera must be able to produce a shot description, no matter what segment of the video it is given. We use our long shot virtual camera as a default camera, since it specifies the original camera data as a shot description. Furthermore, as one filmmaking guide says, "most scenes include, within the first few shots, an *establishing shot* . . . and its purpose is to define the physical space" [Cantine et al. 1995]. Based on this guideline, we chose to begin all videos with an establishing shot; only the node representing a long shot is included in the graph at frame 0. This node is the root of the graph.

For any path through the graph (i.e., shot sequence), two or more sequential nodes that represent identical shot descriptions are a single *shot*. For example, if three board closeup nodes that frame the same regions are next to one another in a shot sequence, they represent a single board closeup shot that spans the frames corresponding to these nodes. When multiple tracking nodes or multiple instructor closeup nodes follow one another in a shot sequence, they also represent the same shot. However, since shots are composed over their duration, the exact shot framing described by one of these nodes changes, depending on its context. For example, consider a node generated by a tracking shot virtual camera. In a particular shot sequence, this node may be at the beginning of a tracking shot. In this instance, the node represents that part of the tracking shot where the camera begins to speed up so that it can track the instructor across the room, but if we were to choose a different path through the graph that makes this node the end of a tracking shot, the node would then represent that part of the tracking shot where the camera slows down in preparation for a stop. In this way, shots that last longer than l frames are represented within the graph. The selected path in Figure 10 depicts one shot sequence where a long shot of length $5 \times l$ frames is followed by a board closeup.

Ideally, l should be as small as possible, since shot breaks can only occur between nodes in the graph. But making l too small increases computation time when searching for the ideal shot sequence. Furthermore, a slightly larger l helps filter out the errors that may have been introduced by noisy original camera data. We have found that a segment length of 30 frames is small enough to facilitate precise shot changes, while still allowing fast and accurate calculation on the resulting shot sequence graph.

While our system does not require any input from the user to help in the shot selection process, if desired, user supplied constraints can be built into the graph. A user can specify a shot to use at a particular point, or disallow certain virtual cameras to produce shots at specific moments in the video. The system simply removes those nodes from the graph which represent the shots that the user does not wish to see. User supplied constraints were not used when producing the examples found in this article.

6.2.2 *The Objective Function.* To evaluate a shot sequence q, we define an objective function $e(q)$, where worse shot sequences produce higher values. This function consists of a number of terms, each penalizing a specific trait. The objective terms are based on the art of film and knowledge of the lecture domain.

Our objective function can be divided into two parts. The first part is used to help evaluate very general characteristics of a shot sequence. In the following definitions, S is the number of shots in the shot sequence q, and s_i is the ith shot. Similarly, F is the number of frames, while f_i is the ith frame. Moreover, R and r_i correspond in the same way to region objects. Finally, each E_i is a tunable weighting variable that is always ≥ 0:

—*Screen coverage*. The screen coverage term $e_c(q)$ focuses the resulting video on important areas of the source video by penalizing shots where important region objects occupy little screen space. The following equation uses the importance probability function $p(r, f)$ defined in Section 5. Here, $a(r, f)$ is the percentage of the screen that region r fills at frame f.

$$e_c(q) = \sum_{i=1}^{F} \sum_{j=1}^{R} max(0, p(r_j, f_i) - a(r_j, f_i)) \times E_1$$

—*Length*. Shots that are too short are often jarring, while those that are too long do not promote visual interest. The length term $e_l(q)$ encourages shots chosen for the final video to be close to some ideal length I by adding a linear penalty for shots that deviate from this length. We assign a second penalty

to shots whose length is either shorter than a minimum M_1 or longer than a maximum M_2. Here, $k(s)$ is used to denote the length of shot s.

$$e_l(q) = \sum_{i=1}^{S} \begin{pmatrix} |k(s_i) - I| \times E_2 + \\ max(0, M_1 - k(s_i)) \times E_3 + \\ max(0, k(s_i) - M_2) \times E_4 \end{pmatrix}$$

For our examples, we set $I = 14$ seconds, $M_1 = 5$ seconds, and $M_2 = 20$ seconds.

—*Similarity*. To avoid jump cuts, namely, cuts "between two shots in which there is too small a change in the image size... resulting in a slight, but very noticeable, displacement in the image on the screen" [Cantine et al. 1995], or introducing pan and zoom transitions that barely move the camera, the system avoids placing two very similar shots one after the other. In $e_s(q)$, $x(a, b)$ is a Boolean function that returns 1 if and only if the last frame of shot a and the first frame of shot b have similar centers and zoom levels, or widths in pixels.

$$e_s(q) = \sum_{i=1}^{S} x(i, i+1) \times E_5$$

—*Variety*. A variety of different virtual cameras should be used to help add visual interest to the video. More specifically, the shot sequence should avoid alternating between two camera operators or only choosing shots from a single camera operator. In the definition of $e_v(q)$, the Boolean function $t(a, b)$ returns 1 if and only if shots a and b were created by the same virtual camera.

$$e_v(q) = \sum_{i=1}^{S} t(i, i+1) \times E_6 + t(i, i+2) \times E_7$$

Each of the terms in the second half of the objective function are designed to coerce the system to use specific shot types at appropriate moments.

—*Ending*. To control how a shot sequence ends, we use the term $E_8 \times e_e(q)$. Here, $e_e(q)$ penalizes a sequence for not ending with a long shot. Unlike the establishing shot at the beginning of the lecture, this final long shot is not built into the shot sequence graph, allowing the system to use a different shot at the end of the video if this would allow a better sequence of shots overall.

—*Special effects shots*. Since the two special effects shots that our system produces are intended to make obstructed, important text visible, we use the term $E_9 \times e_x(q)$ to control the use of these shot types. $e_x(q)$ encourages the system to use a special effects shot when and only when the instructor blocks important text by introducing a penalty when appropriate. As with all of our heuristics, determining when effects should be used is an artistic decision. Our method for determining the appropriateness of a special effects shot is not the only good one.

—*Instructor shots*. The final term in our objective function, $E_{10} \times e_i(q)$, controls the use of instructor closeups. $e_i(q)$ introduces penalties in two different situations. First, it penalizes shot sequences that contain a closeup shot of the instructor when the instructor is not speaking. Second, if a tracking shot or basic instructor closeup shot is used when the other would produce a tighter shot, the shot sequence is penalized.

The objective function $e(q)$ is simply the sum of error terms. The weighting variables E_i can be hand tuned to provide a desired editing style. In practice, only a small number of iterations is needed to find appropriate values for these weights. These variables were set only once for examples found in this article.

6.2.3 *Searching the Shot Sequence Graph.* The shot sequence graph for a video describes a maximum of $c^{\lceil v/l \rceil - 1}$ shot sequences, where v is the length of the final video in frames. We use a graph searching optimization algorithm to quickly find the shot sequence that minimizes the objective function from among this large number of possibilities.

As the objective function defined in Section 6.2.2 includes nonlocal terms, such as the *variety* term, dynamic programming cannot be used to find the optimal shot sequence. However, we can use branch-and-bound optimization to keep from evaluating the objective function for every shot sequence in the graph. We perform a depth-first iterative calculation of the objective function for each possible path through the graph. If at any point during the computation, the error for a branch of the graph exceeds the error for the best shot sequence found so far, the rest of this branch is not searched. To begin our search, we use the value of the objective function for the shot sequence comprised entirely of long shots as our lower bound, as this sequence represents original video data.

The objective function $e(q)$ can be calculated incrementally in the following way:

$$e(q) = \sum_{i=1}^{N} o(n_i, q_{i-1}).$$

where N is the length of sequence q in number of nodes, n_i is the ith node in the sequence, q_i is the sequence comprised of the first i nodes of sequence q, and $o(n, q)$ is a monotonic function that provides the additional error accrued by adding node n to the partial sequence q. Building $o(n, q)$ such that it is monotonic takes some care, since $o(n_i, q_{i-1}) \neq e(q_i) - e(q_{i-1})$. The function $e(q)$ deals only with completed sequences. A completed sequence that ends with a shot which is shorter than the minimum desired length should be penalized for the short shot. A partial sequence that ends with a short shot should not be penalized. The short shot penalty should only be added when a new shot begins or the end of the shot sequence is reached, creating a completed shot sequence. In practice, $o(n, q)$ can be coded efficiently by retaining the state of the partial sequence, such as the length of the last shot, and using this information to calculate the additional error accrued by attaching the node q to the end.

While using this branch-and-bound technique is likely to accelerate optimization, the search problem is still exponential in time. Thus, we process the graph iteratively in overlapping windows of size u nodes. The first w nodes from the result of each iteration are kept as part of the final shot sequence, and the wth node is used as the starting point for the next iteration. The value and state of $o(n, q)$ up through the wth node is retained, allowing each window to use stored global and local information about the chosen partially complete sequence in the rest of the evaluation. Using this method, we cannot guarantee that the optimal shot sequence will be found, but it will approximate the optimal. In our experiments, we process our video in 30-second chunks, retaining only the first 10 seconds of the result (in other words, we use $u = 30$ nodes and $w = 10$ nodes). With these values, we are able to quickly find a shot sequence that is close to optimal in terms of the objective function. For example, even on our shortest example video, which runs for one minute and 14 seconds, the optimal branch-and-bound technique takes about 31 minutes to complete. The windowed branch-and-bound method takes only seven seconds and finds a path for which $e(q)$ is within 3% of that of the optimal path. Since the windowed branch-and-bound method is linear in time, it should be even more advantageous as the length of the video grows.

6.3 Transitions

Once the shot sequence is selected, transitions are added. Transitions, unlike the shots discussed thus far, are dependent on those shots surrounding them; they exist for the purpose of making a shot change less abrupt. A transition is applied between any two shots that meet particular criteria and not applied

between other shots. As such, transitions do not need to be included in the search process and can be added after the final shot sequence has been selected.

In our implementation of Virtual Videography, two types of transitions are inserted into the shot sequence:

—*Pan and Zoom Transitions*. When the location of the center of a shot is similar to the location of the center of the shot that follows, it is desirable to zoom between these shots to help avoid jump cuts (see Section 6.2.2 for a definition of a jump cut). Similarly, if framed portions of the original video in two consecutive shots are approximately the same width in pixels (in other words, at about the same zoom level) and located mostly horizontally or mostly vertically from one another in the original video data, a pan is added between the two shots.

—*Special Effects Transitions*. When a ghosting follows a nonghosting shot, the instructor fades from the scene instead of abruptly disappearing. The opposite is true when a ghosting shot is followed by a nonghosting shot. Similarly, picture-in-picture effects slowly appear or disappear from the screen.

7. IMAGE SYNTHESIS

While view synthesis is an important and open problem in computer graphics, our nearly planar domain and the limits on the types of shots we use make our image synthesis needs simple. The problem that we face is the same as that faced by documentary filmmakers who must create videography from legacy footage and photographs. Historical documentaries by Ken Burns, such as Jazz [2001], provide well-known examples where synthetic camera motions are used effectively. Recently, special-purpose tools for creating these shots have appeared on the market [Canopus 2002; Stagetools 2002; Machrone 2003]. Our image synthesis component provides similar functionality to these tools for our synthetic videographer.

Once a shot sequence has been chosen by the system, all of the synthetic views are synthesized using some combination of digital zoom, subpixel accuracy cropping, and compositing. In all cases, bicubic interpolation is used to resample the original video data so as to create high-quality final images.

The resolution of the final video, w_{target} x h_{target}, is determined using the formula:

$$h_{target} = \begin{cases} h_{source} & \text{if } h_{source} * a_{target} \leq w_{source} \\ \frac{w_{source}}{a_{target}} & \text{otherwise} \end{cases},$$

$$w_{target} = h_{target} * a_{target}$$

where w_{source} and h_{source} are the vertical and horizontal resolutions of the source video, respectively, and a_{target} is the target aspect ratio. This formula finds the maximal dimensions for a rectangle with aspect ratio a_{target} that fits within the source data.

Since the final video is usually rendered to the same or close to the same resolution as the original video, synthesizing a shot may require the original pixel data to be blown up significantly. Even using bicubic interpolation, the quality of the final images can be very poor if some limits are not placed on how much the original pixels can be scaled. For this reason, we do not allow cameras to specify shots that frame areas of the original video smaller than one-quarter of the rendering size in either dimension. While this limit still allows significant digital zoom, most of the degradation is isolated to the instructor, not the writing on the board. If a region smaller than this limit is blown-up, we have found that the writing on the board begins to distort badly, making it hard to read.

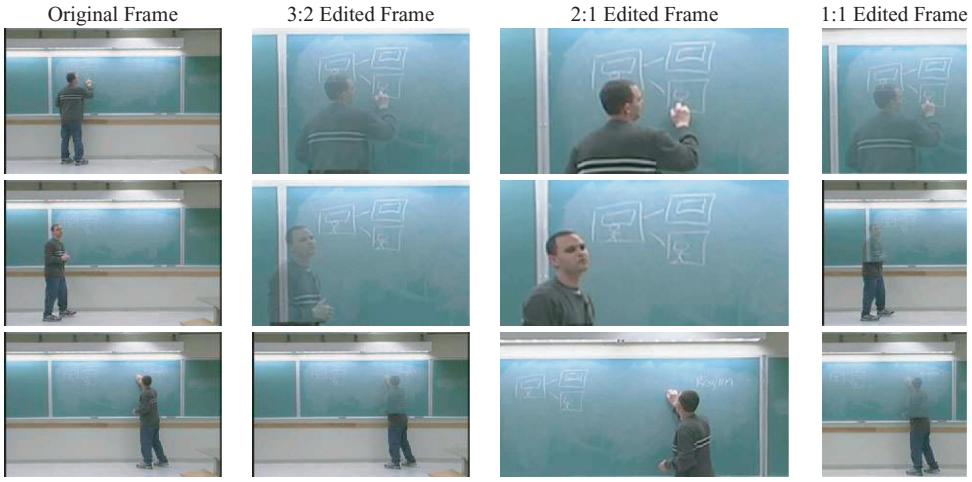


Fig. 11. Three samples taken from a 30-second video segment. The first shot in each sample is the original camera data, while the remaining samples were generated by the Virtual Videography system with varying target aspect ratios (3:2, 2:1, and 1:1, respectively).

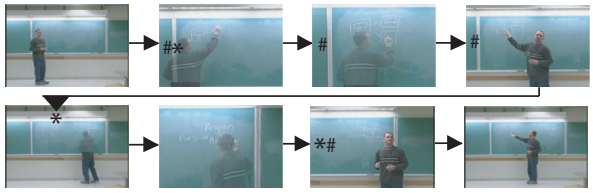


Fig. 12. This storyboard shows a frame from each shot in a 51-second segment of video created by our system. An ∗ after an arrow denotes the use of a special effects transition, and a # after an arrow denotes the use of a pan and zoom transition. When no symbol is shown, a cut was used to transition between the two shots.

8. ASPECT RATIO CHANGES

Our system does not require the source and target videos to have the same aspect ratio. We have been able to produce videos from DV sources with aspect ratios of 2:1, 3:2, 1:1, and even 1:2. We have also produced fullscreen (4:3) videos from widescreen (16:9) HD data. Figure 11 shows the results of creating video of varying aspect ratios from the same source footage.

Because Virtual Videography is designed specifically to frame shots of varying sizes from video data, only two details from the article are different when performing an aspect ratio change: First, a long shot cannot be defined as the original camera data, since the original data is of the wrong aspect ratio. Our system creates long shots by framing the instructor and as many of the regions as can fit within the screen, preferring regions of high importance when all will not fit. Second, Section 6.1.1 states that the linear system for finding good tracking shots always has a solution. For aspect ratio changes, this is no longer true. As such, the system does not perform tracking shots when it is impossible to solve the linear system.

9. RESULTS

Virtual Videography employs the art of filmmaking to inexpensively produce effective lecture videos, without the intrusion of a professional video crew. Figures 12 and 13 show examples of our results in

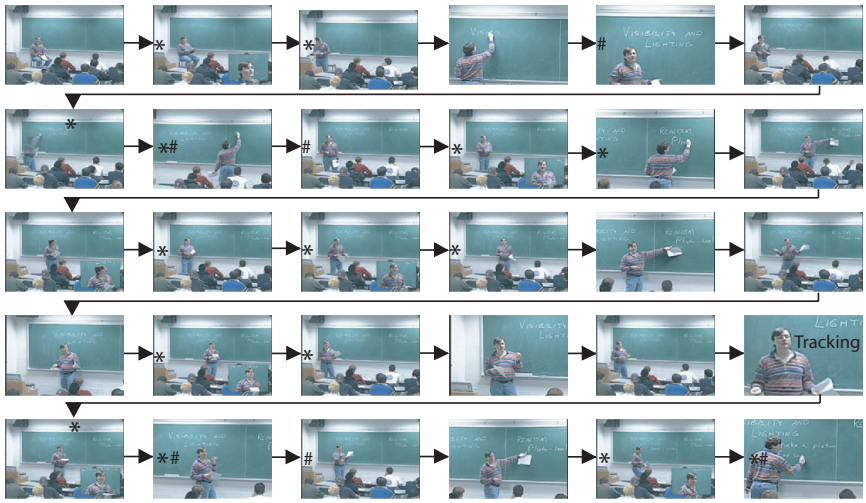


Fig. 13. This storyboard shows a frame from each shot in a 6-minute, 17-second segment of video created by our system. An ∗ after an arrow denotes the use of a special effects transition, and a # after an arrow denotes the use of a pan and zoom transition. When no symbol is shown, a cut was used to transition between the two shots.

storyboard form. Notice how these edited videos use a number of shot types to help guide the viewer's attention towards important aspects of the lecture and promote visual interest. Transitions and special effects are used to smooth shot changes and allow the viewer to see important writing on the board.

With the exception of the examples presented in Figures 12 and 18, all of the results presented in this section come from recordings of an actual undergraduate computer graphics course. The entire semester's worth of lectures were recorded on miniDV tapes using two consumer-grade cameras mounted in the back of the classroom. Each of the cameras captured a little more than half of the board. The audio was recorded using either a ceiling-mounted microphone or a lavaliere microphone worn by the instructor. The digital videos were captured from the miniDV tapes using a Dell Precision 530 workstation installed with a Canopus DV Storm video editing card, a 2.0 GHz Intel Xeon processor, and one gigabyte of RAM. For the examples in this article, we only use the data from one of the two video cameras that captured the lectures. Multiple camera processing will be discussed in Section 10.

Once the video is digitized, processing is highly automatic. Our current implementation of the media analysis step requires the user to specify the location of the board and to select a few example frames of each gesture. The rest of the process only requires the user to specify which virtual cameras to use for creating shots in the final video. The system parameters discussed throughout this article were set once for all of the data that was captured. The values we use for each tunable parameter are listed in Appendix B.

After media analysis, an average ten-minute video segment takes approximately one hour and fifteen minutes to process on our unoptimized research prototype. Since the attention model and computational cinematography components together take an average of two minutes, most of the processing time is spent in disk I/O, rendering video, and video compression/decompression. While we need to use bicubic interpolation to render each frame of the final video, our system is capable of producing real-time previews at the expense of quality. A high-quality rendering of an entire lecture can be made overnight on a typical PC.

(a) (b) (c)

Fig. 14. In (a) and (b), we show two frames of a closeup shot chosen by the VV system. The shot is correctly framed in (a), even though the instructor has not finished writing the block of text, as shown in (b). Since an online system does not have future knowledge, it cannot exhibit this characteristic. Moreover, (c) shows an incorrectly framed closeup of the same region that could mistakenly be selected by an automated online system. A shot like the one shown in (c) will never be framed by our system because of the availability of later video data.

Fig. 15. In this VV-edited example, the video makes a cut from a closeup of the board to a long shot of the classroom in response to a pointing gesture directed towards an unframed region of writing.

9.1 Evaluation

Filmmaking is an art, making video evaluation highly subjective. We have chosen to have our system edit in a conservative style that we feel is appropriate for lecture videos. In particular, our system strictly adheres to the rules taught to beginning filmmakers, resulting in videos with a moderate pacing. Experienced videographers can often break these rules, as they understand the tradeoffs. For example, a professional editor who edited a clip of our source footage[1] used many cuts to create a fast-paced style such as we one would see in a action-oriented television short. Less experienced editors are taught to avoid too many cuts, as they lack the judgement as to when the effect is balanced by other elements.

Because of the subjective nature of video style, when evaluating our videos, we do not focus on the style of the result. Instead, we evaluate our system's ability to mimic the editing style that we prefer. To do this, we inspect our results for important properties and behaviors that illustrate our chosen editing style, compare our results with manually edited videos, and gather informal feedback from viewers.

The videos our system produces exhibit the properties of our chosen editing style. For instance, Figure 14 shows an example where our system chose to do a closeup shot. Despite the fact that the instructor has not finished writing the text contained in the shot, our system is able to use future information to correctly frame the region, avoiding unnecessary camera motion and leading the viewer's attention towards areas of importance. There are many examples in our videos where camera motions help direct the viewer's attention. In Figure 15, the camera makes a cut from a closeup of the board to a long shot before the instructor points to the right, allowing the viewer to see the writing that the instructor is about to point to. Finally, the system produces a wide variety of shots which do not follow a simple pattern; the objective function ensures that these shots are combined in the most effective way possible. For example, throughout the duration of the video depicted in Figure 12, the system chose transitions from every possible shot type to every other possible shot type with no recognizable pattern.

[1]The services of the editor were arranged anonymously through the journal editor.

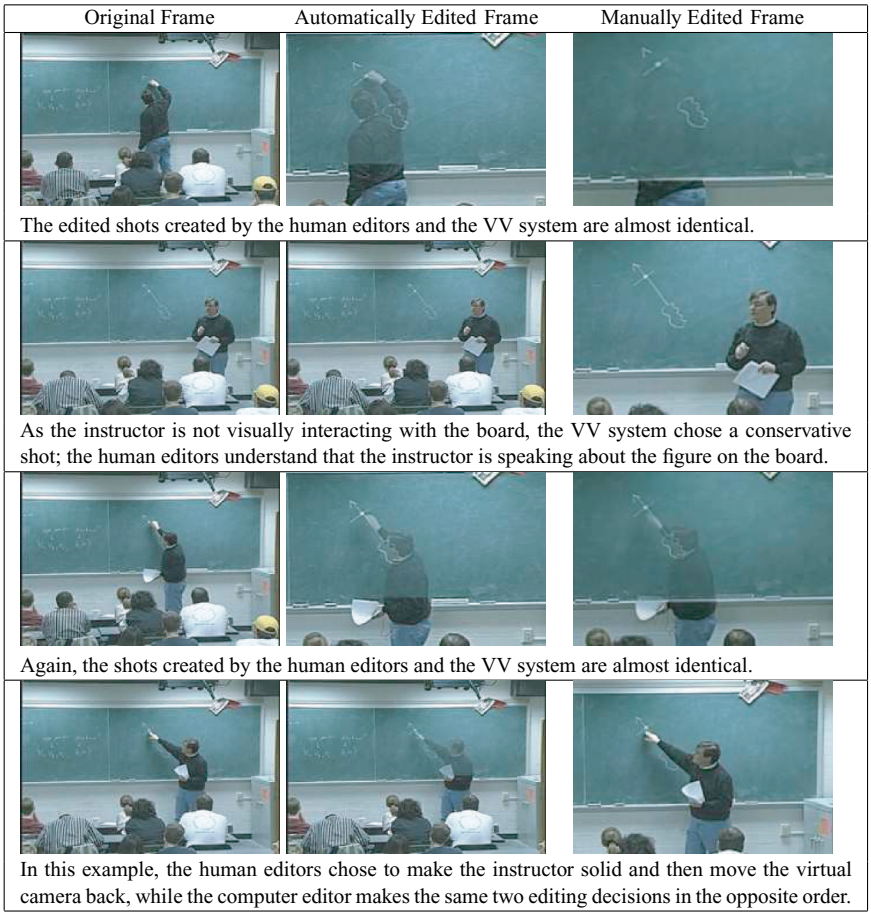


Fig. 16. Four samples taken from a two-minute video. The first shot in each sample is the original camera data, the second was chosen by the Virtual Videography system, and the third was edited by undergraduate film students. Notice how many of the shots look similar, but noticeable differences occur when the human editors have more information about the scene.

We compared the results of our system to videos produced by undergraduate film students and an inexperienced video editor. Figure 16 shows comparison shots from a Virtual Videography-produced video and one edited by two undergraduate film students using the same data. A graphical comparison of these two videos is shown in Figure 17. Figure 18 shows a video created by Virtual Videography compared with one produced by an inexperienced video editor. In this example, the original video was captured using a prosumer, JVC JY-HD10 HD camcorder, and the source and target aspect ratios were not the same. Neither the film students nor the inexperienced video editor were given instructions that might influence their results, and they did not see the videos produced by our system until after they edited their own. While not easy to see on paper, the videos are strikingly similar. The automatically produced videos use shots with almost identical framing to many of the shots in the manually produced videos. The Virtual Videography-produced videos also follow the instructor in much the same way as the manually produced ones, though shot changes occur at slightly different times.

We conducted a fair amount of informal viewer evaluation. We showed our video results to approximately 150 people in various disciplines, including, but not limited to, computer science, film, biology,

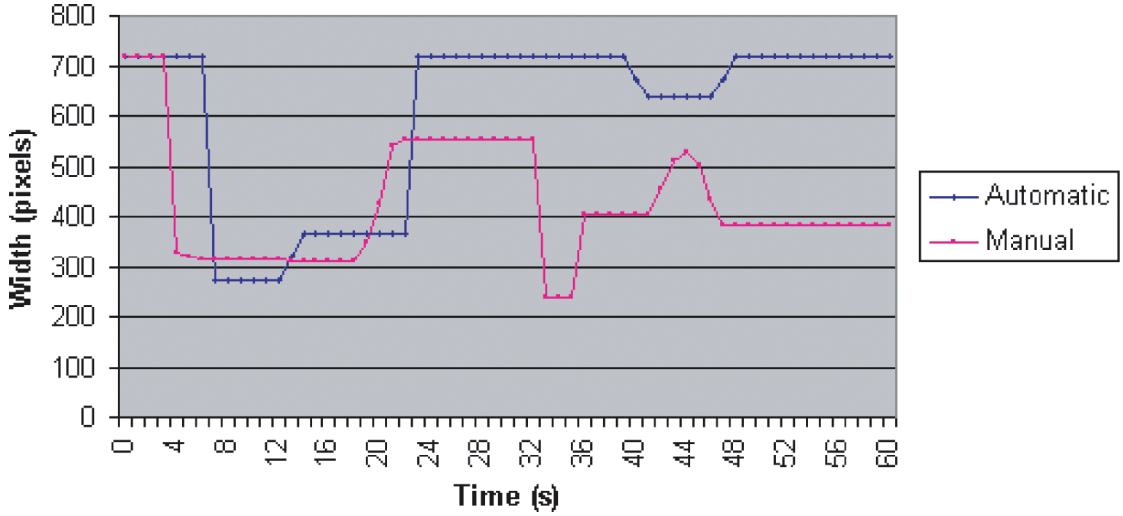


Fig. 17. This chart compares an automatically edited video with a manually edited one. It plots the width in pixels of images cropped from the original video by each of the two editors throughout the videos' durations. In other words, it shows the zoom level of each virtual camera. Notice how the structure of the camera motion, specifically around the 20s and 40s marks, is similar for both videos. Overall, the automatically edited video is more conservative in its shot choices, choosing shots further away when uncertain of the content of the video.

and learning sciences. The feedback we received was overwhelmingly positive. Some who were not warned ahead of time were amazed to find that the videos were produced by a computer with almost no human intervention. Since our system is easily extensible, the comments we received helped us improve our results. For example, while almost everyone who saw our ghosting effect remarked on how useful it would be for lecture video and even went so far as to wish instructors could be ghosted in real life, some told us that they found the effect disconcerting. This was usually only the first few times a viewer saw the instructor ghosted. A few others did not like that writing appeared before the instructor had written it, since we always filled the board in the clean plate video with future information. To compensate for the concerns of these few viewers, we added the picture-in-picture effect as an alternative to the ghosting shot and added the option to have the clean plate video stream filled with past information.

The impact of media quality on educational effectiveness has been a subject of debate in the learning science community. The argument of Richard Clark [1983] is that media is merely a vehicle to deliver material; the material itself is responsible for education. This debate as to whether "media matters" still rages in the education community (see Clark [1991] for an early discussion, or the web page [2005] for a more up-to-date summary). While we believe in the value of media quality (if only to keep viewers engaged long enough to receive the message), the existence of this debate shows the challenge in assessing the impact of media quality on learning. Studies such as Rowe et al. [2003] show the value of lecture video, especially in terms of cost-effectiveness.

9.2 Limitations

The limitations of our system fall into two main categories: those that stem from our implementation and fundamental limitations of our approach.

9.2.1 *Implementation Limitations.* Our decision to use simple media analysis algorithms can limit the quality of our results. In particular, when creating the clean stream of the board, we chose to

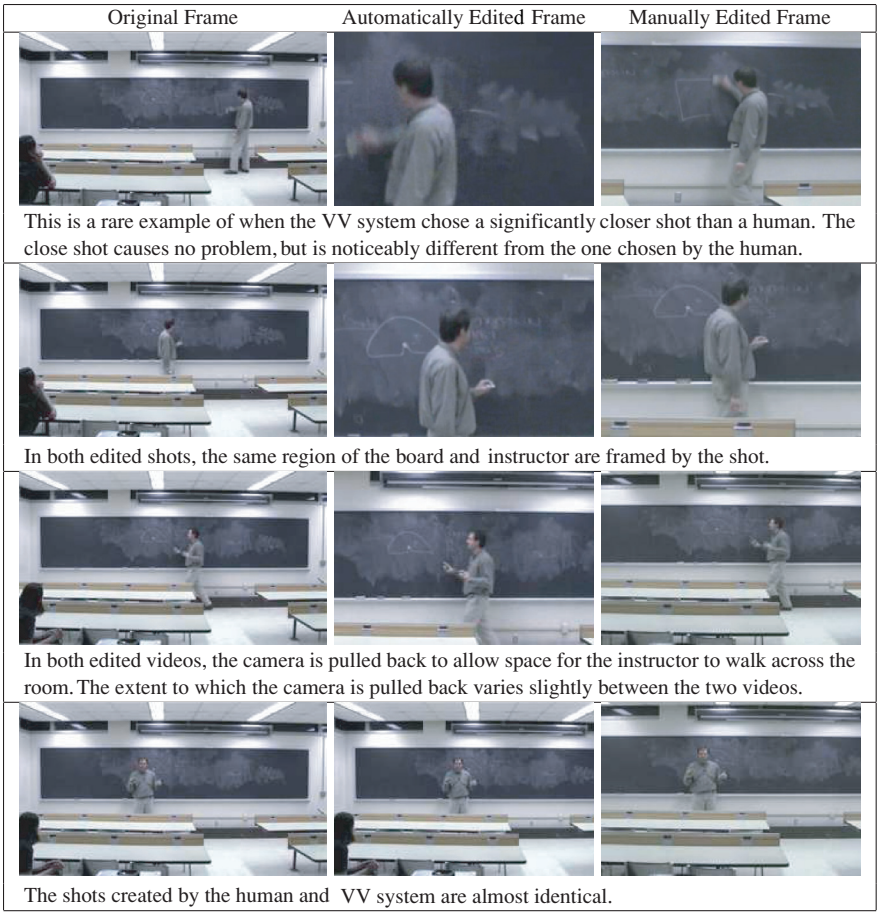


Fig. 18. Four samples taken from a two-minute video. The first shot in each sample is the original HD camera data, the second is from a DV video produced by the Virtual Videography system, and the third is from a DV video edited by an inexperienced video editor using the same data.

use a color classification segmentation method for reasons discussed in Appendix A. But our method will break down if the instructor is wearing clothing that is the same color as the board. There are also circumstances when the region-finding method described in Appendix A will produce erroneous or undesirable results. Our technique for identifying writing from erasing assumes that writing produces high frequencies, while erasing removes them. This may not be the case if a wet eraser is used to erase a chalkboard, as lines may appear on the edges of the wet streak on the board. Additionally, if the bounding boxes for each region end up overlapping, all regions across the board may merge, forming one large region. These degenerative regions are usually caused by representing a region object as an axis-aligned bounding box, and the problem can be curbed by using some other way of describing the location of the writing.

The quality of the shot sequence chosen by the shot selector is also limited by the restricted information provided by media analysis. If important pieces of the lecture are not accurately identified, the shot selector will choose to make a shot that shows more of the room so as not to inadvertently cover up an important aspect of the lecture. This limitation may lead to videos with more long shots than we

Fig. 19. The VV system chooses to move the camera back to catch a late student entering the class.

would see in a manually edited video. For example, in one of our source videos, a late student enters the classroom, blocking a part of the board for a significant amount of time. Since the system has no way of knowing better, this late student is considered an important aspect of the lecture, and the shot selector responds by cutting to a long shot to catch the student walk to his seat (see Figure 19). Gathering more data through media analysis or user-supplied hints could allow the shot selector to make fewer of these conservative choices.

9.2.2 *Fundamental Limitations.* Some limitations arise from the way we choose to approach the problem of lecture-video editing. Our method of using a small number of inexpensive, portable sensors often does not provide the source footage necessary for some interesting shots. For example, our system does not produce extreme closeups of the instructor, as there is generally not enough resolution to synthesize such a shot. However, these types of shots tend to be less important in a lecture environment. If a detailed facial expression made by the instructor is of particular importance, the expression would have to be exaggerated so that audience members in the back could see it. For Virtual Videography to work, all important aspects of the scene must be clear in the captured video data. The system cannot create information that is not captured by the cameras. In fact, we captured a second semester's worth of lecture data using our HD camcorder. Unfortunately, all of this data was unusable due to poor light quality in the lecture room.

A different fundamental limitation of our approach is that we do not change the timing of the lecture. While removing unimportant moments from a video of an event can be beneficial, cutting a segment of a lecture requires a system that can reliably determine when nothing important is being covered.

Finally, while the extensibility of our system allows for encoding still more of the art of filmmaking, the subjective nature of the art still exists. For example, it is easy to add visual effects into our system, such as shots of mature regions of the board or shots where regions have been rearranged on the board; it is much harder to decide when to use these shots. Similarly, our system assumes that the instructor should be visible in the edited video at all times. Experimentation with cameras that do not always frame the instructor may lead to better rules for determining when it is necessary to include the instructor in a shot.

10. DISCUSSION

In this article, we presented *Virtual Videography*, a system for automatically producing edited videos of classroom lectures from fixed camera footage. Using only a single audio/video stream, our system assembles videos that exhibit multiple shot types, visual effects, and transitions, while following many of the rules of cinematography.

Though our Virtual Videography system only works for traditional lectures, we believe that our four key insights apply to a wide range of domains: offline processing allows for better analysis, decision making, and image synthesis; simple syntactic cues are sufficient for cinematic decision making; constrained optimization over time is a natural way to phrase the shot planning problem; and image synthesis is a useful alternative to special purpose cameras.

All of the examples discussed in this article use data from only a single video camera focused on the board. While we have not implemented multiple input streams, our system should be able to use different virtual cameras to deal with different streams of video. In the simple case where several cameras in the back of the lecture room are pointed at different parts of the board, the multiple camera streams can be treated as a single camera until rendering. Being able to use several different camera streams can also allow us to use different camera types. For example, while we chose not to use automated tracking cameras in our current implementation, an automated tracking camera may be able to provide another shot option for the final video. Another possibility could be a camera that records the audience, allowing videos which convey a sense of presence or relay both sides of a classroom dialog.

While Virtual Videography is designed to produce traditional lecture videos, simple extensions could allow the creation of other less traditional media. For example, the data collected by the system can be directly used for lecture indexing and annotation, allowing automated authoring of a nonlinear multimedia presentation, rather than only a traditional video.

Since our system postpones the video planning process until after the lecture, it is able to produce several different videos of the same event tailored to specific needs. In fact, unlike any other automated editing system, Virtual Videography can even create novel videos from archived lectures. We have shown that our system is capable of producing videos of the same lecture at different aspect ratios. In particular, we have been able to produce 4:3 video from a 16:9 HD video. Our ability to deal with this type of data will become more useful in the future, since HD video is becoming increasingly popular because of its high resolution. While we are pleased with our results, to make the most effective use of a screen with a specific aspect ratio, the objective function should be tailored to this aspect ratio. This is necessary because the rules of cinematography change as the target media changes. Furthermore, other issues, such as lag, frame rate, and actual resolution, should be taken into account when authoring for a particular device.

Our system meets our goal of producing videos in our chosen editing style from limited inputs. One particulary interesting area of future research is determining what makes lecture videos educationally effective. Our system is designed in an extensible way, making it possible to incorporate better rules as they are discovered.

APPENDICES

A. Media Analysis Details

This appendix details the simple techniques we use for media analysis. Other techniques could be used to gather similar or more complex information.

(1) *Segmentation*. To perform segmentation on the original video, we use a color classifier to identify the instructor as described by Jones and Rehg [1998]. Our implementation makes two extensions to this work. First, we use separate color arrays to represent different objects: one array for the instructor and one for the board. Second, we automatically train the arrays using every nth frame of the original video data by observing that instructor pixels change often. Thus, pixels which change drastically from one training frame to the next are used to train the instructor array. The other pixels are used to train the board array.

Once the array has been trained, we classify a pixel as a member of the object whose array returns the highest confidence. Instructor pixels are removed and the holes filled using known board pixels at the same spatial location at different, but temporally close, times in the video. Since this processing is done offline, the pixels may be taken from the past, future, or a combination of both. If we use future information, writing that is obscured by the instructor will become visible shortly before it is written; if we use past information, the writing will appear once it is no longer obstructed. The decision to use

past or future information is most important for producing ghosting special effect shots, discussed in Section 6.1.2.

We choose to use color classification to segment the board for a number of reasons. First, every type of board (blackboard, chalkboard, markerboard, etc.) will be one constant color. Second, color classification works when there are multiple occlusions, such as more than one instructor or a student standing between the camera and board. Finally, color classification works when there is little or no motion between frames, such as when the instructor is writing on the board or standing still and lecturing.

(2) *Finding Region Objects*. The first step in our region-finding process is to identify *strokes* of writing or erasing. A stroke is writing or erasing that occurs over a small amount of time, such as a few seconds. Strokes are found by subtracting two frames that are close in time from the segmented video. Difference between the two frames indicates that there has been writing or erasing. To provide a better estimate of when a stroke occurs, we repeat the subtracting process, decreasing the time between the two frames until there is no longer a difference.

To distinguish writing from erasing, we note that writing creates high frequencies and erasing removes them. Thus, a stroke can be identified as either writing or erasing by applying a high-pass filter to the area of difference in the later frame. The presence of high frequencies indicates that there has been writing, while a lack means that there has been erasing.

To form region objects, strokes are combined in chronological order using heuristics based on the structure of writing. We use the type of stroke and its proximity to the *active region*, namely, the region that is currently being formed, to advance the system:

—If there is no active region and a write stroke appears, a new region is created and is considered the active region. The dimensions of the stroke are used for those of the region. The birth and maturity times are set to the time that the stroke occurs, and death is set to the end of the video.

—Whenever a write stroke occurs far from the active region in either time or space, the active region is considered mature and a new region created, becoming the active region. The original dimensions of the stroke are used for the new region. The birth and maturity times are set to the time that the stroke occurs, and the death is set to the end of the video.

—A stroke that occurs close to the active region in time and space extends the active region. The dimensions of the active region are increased to include the current stroke, and the maturity is changed to when the stroke occurs.

—When an erase stroke is detected, all existing regions are inspected. If the erase stroke greatly overlaps a region, then the region is erased, and the death time of this region is set to the time of the erase stroke.

We determine the final dimensions of a region object by subtracting a frame of the segmented video from before the birth of the region from a frame after the maturity of the region. The difference between the two frames represents all writing in the region; this technique defends against errors in the stroke-finding process.

After all regions have been formed, we merge those that overlap in time and are significantly close in space. This is achieved by setting the death time of the older region to the birth time of the younger. The younger region's size is then adjusted to include the older image.

(3) *Tracking*. We use the following test to find those pixels in the original video that represent the instructor:

$$|O_{i,j} - C_{i,j}| > T$$

In this equation, O is the original video, and C is the clean stream, that is, the segmented video found in step one. Any given pixel location (i, j) that has a difference greater than some threshold T is marked.

The set of pixels marked for a particular frame is called the *tracking mask*. A bounding box is placed around the tracking mask. The final tracking data is a filtered list of these per-frame bounding boxes. We filter the list by applying a median filter to coordinates of the bounding boxes.

(4) *Gesture Recognition*. The tracking mask described earlier is also used for gesture recognition. First, we generate gesture templates by manually identifying a few example frames where the instructor makes a particular gesture. The normalized tracking mask of each of these frames forms a template for this gesture. Then, for each frame in the original video, the tracking mask is compared to the set of gesture templates. A frame is labeled with the closest-matching gesture. If the same gesture persists for one second, then the instructor is said to be making this gesture. Currently, we recognize three separate gestures: pointing left, pointing right, and reaching up. Additional gestures can be included by adding to the template database, however, we have found that these three are sufficient.

(5) *Audio*. To determine whether the instructor is speaking, we separate the audio into segments based on frame correspondences dividing each second of audio into 30 pieces. If the root-mean-square average of the sample segment is much smaller than that of the entire video stream, then the instructor is considered silent for the frame.

B. Tunable Parameters

The following table lists the tunable parameters found in this article. For each parameter, the section where it first appears is listed and a starting value provided.

Section	Name	Value
5	C	.15
5	D	.0011
5	G	.45
6.1	Moderately Important	$\geq .3$
6.1	Highly Important	$\geq .8$
6.1.2	α	.4
6.2.1	l	30
6.2.2	I	420 frames
6.2.2	M_1	150 frames
6.2.2	M_2	600 frames
6.2.2	E_1	.0167
6.2.2	E_2	.034

Section	Name	Value
6.2.2	E_3	2.5
6.2.2	E_4	.5
6.2.2	E_5	35
6.2.2	E_6	20
6.2.2	E_7	19
6.2.2	E_8	5
6.2.2	E_9	50
6.2.2	E_{10}	2.34
6.2.3	u	30
6.2.3	w	10
Appendix A.3	T	100
Appendix A.5	Much Smaller	$\leq 50\%$

ACKNOWLEDGMENTS

This work was supported in part by NSF Grants CCR-9984506, IIS-0097456, and IIS-0416284. Michael Wallick is funded by a Microsoft Research Graduate Fellowship. We would like to thank James Masanz, Rob Iverson, Min Zhong, Jeff Lulewicz, and Amber Samdahl for their contributions to early phases of this project. Finally, we thank the referees and other readers who have helped us with the exposition in this article, especially Associate Editor Larry Rowe for his patience and feedback and the anonymous referee who graciously performed an edit by hand.

REFERENCES

ABOWD, G. 1999. Classroom 2000: An experiment with the instrumentation of a living educational environment. *IBM Syst. J. 38,* 4, 508–530.

BARES, W. AND LESTER, J. 1999. Intelligent multi-shot visualization interfaces for dynamic 3D worlds. In *Proceedings of the 1999 International Conference on Intelligent User Interfaces*. 119–126.

BIANCHI, M. 1998. Autoauditorium: A fully automatic, multi-camera system to televise auditorium presentations. Presented at the *Joint DARPA/NIST Smart Spaces Technology Workshop*.

BIANCHI, M. 2004. Automatic video production of lectures using an intelligent and aware environment. In *MUM: Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia*. ACM Press, New York. 117–123.

BORDWELL, D. 1998. *On the History of Film Style*. Harvard University Press, Cambridge, MA.

BURNS, B., MARKS, R., BEYMER, D., WEBER, J., GLEICHER, M., AND ROSENCHEIN, S. 1998. Person tracking research at autodesk. In *3rd Bay Area Vision Meeting: Visual Analysis of People*—Invited Talk.

BURNS, K. 2001. Jazz: A Film by Ken Burns. PBS.

CANOPUS 2002. Canopus Imaginate. http://www.canopus-corp.com

CANTINE, J., HOWARD, S., AND LEWIS, B. 1995. *Shot By Shot: A Practical Guide to Filmmaking*. Pittsburgh Filmmakers.

CHEN, M. 2001. Design of a virtual auditorium. In *Proceedings of the ACM Multimedia Conference*. ACM Press, New York.

CHRISTIANSON, D. B., ANDERSON, S. E., HE, L.-W., WELD, D. S., COHEN, M. F., AND SALESIN, D. H. 1996. Declarative camera control for automatic cinematography. In *Proceedings of the AAAI Conference*. 148–155.

CLARK, R. 1983. Reconsidering research on learning from media. *Rev. Educ. Res. 53,* 4, 445–459.

CLARK, R. 1991. When researchers swin upstream: Reflections on an unpopular argument about learning from media. *Educ. Technol.*, 34–40.

CUTLER, R. ET AL. 2002. Distributed meetings: A meeting capture and broadcasting system. In *Proceedings of the ACM Multimedia Conference*. ACM Press, New York.

DRUCKER, S. AND ZELTZER, D. 1995. Camdroid: A system for implementing intelligent camera control. In *Proceedings of the Symposium on Interactive 3D Graphics*, 139–144.

GIRGENSOHN, A., BLY, S., BORECZKY, J., AND WILCOX, L. 2001. Home video editing made easy–Balancing automation and user control. In *Proceedings of the Human-Computer Interaction INTERACT Conference*. 464–471.

GIRGENSOHN, A., BORECZKY, J., CHIU, P., DOHERTY, J., FOOTE, J., GOLOVCHINSKY, G., UCHIHASHI, S., AND WILCOX, L. 2000. A semi-automatic approach to home video editing. In *Proceedings of the UIST Conference*. 81–89.

GLEICHER, M. AND MASANZ, J. 2000. Towards virtual videography (poster session). In *Proceedings of the ACM Multimedia Conference*. ACM Press, New York.

GLEICHER, M. AND WITKIN, A. 1992. Through-the-Lens camera control. *Comput. Graph. 26,* 2 (Jul.), 331–340.

GLEICHER, M. L., HECK, R. M., AND WALLICK, M. N. 2002. A framework for virtual videography. In *Proceedings of the 2nd International Symposium on Smart Graphics*. ACM Press, New York. 9–16.

HE, L., COHEN, M., AND SALESIN, D. August 1996. The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In *Proceedings of the SIGGRAPH Conference*. 217–224.

HE, L., LIU, Z., AND ZHANG, Z. 2002. Why take notes? Use the whiteboard capture system. Tech. Rep. MSR-TR-2002-89, Microsoft Research. September.

HE, L., SANOCKI, E., GUPTA, A., AND GRUDIN, J. 1999. Auto-Summarization of audio-video presentations. In *Proceedings of the ACM Multimedia Conference*. ACM Press, New York.

JAMES, D. AND HUNTER, J. 2000. A streamlined system for building online presentation archives using smil. In *Proceedings of the Australasian Conference on Computing Education*. ACM Press, New York. 145–152.

JONES, M. J. AND REHG, J. M. 1998. Statistical color models with application to skin detection. Tech. Rep. CRL 98/11, Cambridge Research Laboratory.

KARP, P. AND FEINER, S. 1993. Automated presentation planning of animation using task decomposition with heuristic reasoning. In *Proceedings of the Graphics Interface Conference*. 118–127.

KATZ, S. 1991. *Film Directing Shot by Shot: Visualizing from Concept to Screen*. Michael Wiese Productions.

LEE, D.-S., EROL, B., GRAHAM, J., HULL, J. J., AND MURATA, N. 2002. Portable meeting recorder. In *Proceedings of the ACM Multimedia Conference*. ACM Press, New York.

LI, Y., ZHANG, T., AND TRETTER, D. 2001. An overview of video abstraction techniques. Tech. Rep. HPL-2001-191, Imaging Systems Laboratory, HP Laboratories, Palo Alto, California. July.

LIU, T. AND KENDER, J. R. 2003. Spatial-Temporal semantic grouping of instructional video content. In *Proceedings of the 2nd International Conference on Image and Video Retrieval*.

MACHNICKI, E. AND ROWE, L. 2002. Virtual director: Automating a webcase. In *Proceedings of the SPIE Conference on Multimedia Computing and Networking*.

MACHRONE, B. 2003. Every photo tells a story. http://www.pcmag.com/article2/0,4149,1404041,00.asp.

MUKHOPADHYAY, S. AND SMITH, B. 1999. Passive capture and structuring of lectures. In *Proceedings of the ACM Multimedia Conference*.

ONISHI, M., IZUMI, M., AND FUKUNAGA, K. 2000. Blackboard segmentation using video image of lecture and its applications. In *Proceedings of the International Conference on Pattern Recognition.* 615–618.

ONISHI, M., KAGEBAYASHI, T., AND FUKUNAGA, K. 2001. Production of video images by computer controlled camera and its application to TV conference system. In *Proceedings of the CVPR01 Conference*. 131–137.

PINHANEZ, C. S. AND BOBICK, A. F. 1997. Intelligent studios: Using computer vision to control TV cameras. *Appli. Artifi. Intell. 11,* 4, 285–306.

ROWE, L., HARLEY, D., PLETCHER, P., AND LAWRENCE, S. 2003. Bibs: A lecture webcasting system. Tech. Rep. 1005, Center for Studies in Higher Education, UC Berkeley. http://ideas.repec.org/p/cdl/cshedu/1005.html.

RUI, Y., HE, L., GUPTA, A., AND LIU, Q. 2001. Building an intelligent camera management system. In *Proceedings of the ACM Multimedia Conference*.

RUSSELL, D. M. 2000. A design pattern-based video summarization technique: Moving from low-level signals to high-level structure. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*.

SAUND, E. 1999. Image mosaicing and a diagrammatic user interface for an office whiteboard scanner. http://www.parc.xerox.com/spl/members/saund/zombieboard-public.html.

SAUND, E., FLEET, D., LARNER, D., AND MAHONEY, J. 2003. Perceptually-Supported image editing of text and graphics. In *Proceedings of the UIST Conference*. 183–192.

STAGETOOLS. 2002. Moving Pictures. http://www.stagetools.com/

STEINMETZ, A. AND KIENZLE, M. 2001. The e-seminar lecture recording and districution system. In *Proceedings of the SPIE Conference on Multimedia Computing and Networking*. vol. 4312.

SYEDA-MAHMOOD, T. AND SRINIVASAN, S. 2000. Detecting topical events in digital video. In *Proceedings of the ACM Multimedia Conference*. ACM Press, New York.

TEODOSIO, L. AND BENDER, W. 1993. Salient video stills: Content and context preserved. In *Proceedings of the ACM Multimedia Conference*. ACM Press, New York.

VIRTUAL INK. 2004. Mimio. http://www.mimio.com/index.shtml.

WALLICK, M., RUI, Y., AND HE, L. 2004. A portable solution for automatic lecture room camera management. In *Proceedings of the IEEE International Conference on Multimedia and Expo*.

WALLICK, M. N., HECK, R. M., AND GLEICHER, M. L. 2005. Chalkboard and marker regions. In *Proceedings of the Mirage—Computer Vision/Computer Graphics Collaboration Techniques and Applications Conference*.

WANG, F., NGO, C., AND PONG, T. 2004. Gesture tracking and recognition for lecture video editing. In *Proceedings of the ICPR Conference*. 934–937.

WEB PAGE. 2005. The great media debate. http://hagar.up.ac.za/rbo/construct/media.html.

Received July 2004; revised December 2005; accepted February 2006