Video Chat with Multiple Cameras

John MacCormick, Dickinson College

ABSTRACT

The dominant paradigm for video chat employs a single camera at each end of the conversation, but some conversations can be greatly enhanced by using multiple cameras at one or both ends. This paper provides the first rigorous investigation of multi-camera video chat, concentrating especially on the ability of users to switch between views at either end of the conversation. A user study of 23 individuals analyzes the advantages and disadvantages of permitting a user to switch between views at a remote location. Benchmark experiments employing up to four webcams simultaneously demonstrate that multi-camera video chat is feasible on consumer hardware. The paper also presents the design of MultiCam, a software package permitting multi-camera video chat. Some important trade-offs in the design of MultiCam are discussed, and typical usage scenarios are analyzed.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Evaluation/-methodology; C.4 [**Performance of systems**]: Design studies

General Terms

Human Factors, Design, Experimentation

Keywords

Consumer video chat, multiple cameras

1. INTRODUCTION

Video chat is now commonplace for a significant proportion of computer and phone users, via popular, user-friendly software such as Skype, FaceTime, and Google Chat. Skype alone reported an average of over 120 million connected users every month in their 2010 IPO filing, stating that 40% of Skype-to-Skype chat minutes employ video [12]. This paper advocates and analyzes an aspect of video chat that has received surprisingly little attention: the use of multiple cameras. Figure 1 demonstrates some of the possibilities enabled by the MultiCam software package described later. In each case, a laptop running Skype has two or more USB webcams connected, and the chat participants at *both* ends of the conversation are able to switch at will between individual views of each camera or a tiled view of all simultaneously. The primary goals of this paper are to analyze the utility and feasibility of such multi-camera video chats, and to discuss some important trade-offs inherent in designing multi-camera software.

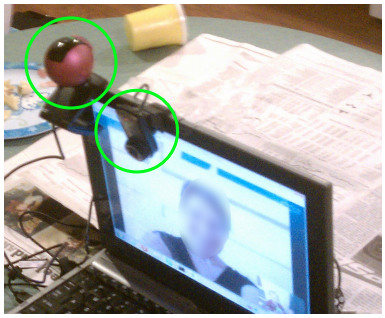
The predominant paradigm for video chat employs a single webcam at each end of the conversation. For many purposes, this is perfectly adequate. But in some cases, the single-camera paradigm is unnecessarily restrictive and burdensome. It is *restrictive* because only a single view is available from the single camera at any one time. It is *burdensome* because the onus is on the person with the camera to point it at the part of the scene that is currently of interest. We need some new terminology to discuss this further: at any particular instant in a conversation between two individuals, the person who is speaking, explaining, or demonstrating an activity or object will be referred to as the *speaker*; the person listening and watching the speaker will be referred to as the *listener*. (Of course, the identities of the speaker and listener swap frequently in a typical two-way conversation.)

It may prove impossible for technology to completely restore the freedom of face-to-face conversation under the constraint of video chats. But there are three obvious avenues to explore in seeking to partially restore this freedom:

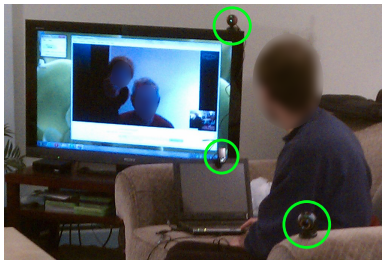
1. Employ *multiple cameras* simultaneously, each showing a different view of the scene (thus reducing—but probably not eliminating—the need for the speaker to move cameras or objects, and monitor the listener's view).
2. Permit *listener control*: allow the listener to adjust and choose between the views offered by the speaker. This includes switching between cameras, viewing all cameras simultaneously, and could also incorporate more fine-grained control such as (digital or actual) pan/-tilt/zoom.
3. Use *heterogeneous devices* to provide the listener with maximum choice. This could include standard webcams, wide-angle cameras, 3D cameras, wireless cameras, and panoramic cameras.

This paper investigates some aspects of items 1 and 2 (multiple cameras and listener control), but does not address

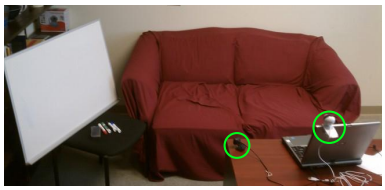
(a) two forward-facing cameras



(b) forward- and rear-facing cameras

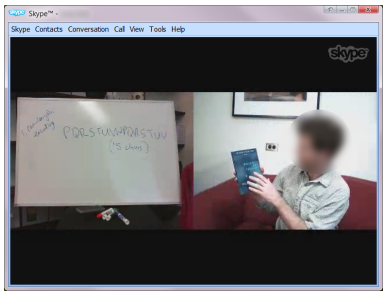


(c) wide shot, headshot, and close-up



(d) two cameras for whiteboard discussion

(e) remote tiled view of (c)



(f) remote tiled view of (d)

Figure 1: Typical MultiCam usage scenarios. Webcams are highlighted by green circles.

item 3, except for a brief discussion in Section 3. And even within items 1 and 2, the paper examines only a small subset of the possible approaches. The primary objective is to demonstrate the utility and the feasibility of multi-camera video chat in general, and especially the utility of listener control. Note that the paper specifically addresses *consumer* video chat, as opposed to commercial video conferencing or professional webcasts. Therefore, we seek solutions that: (i) involve inexpensive, standard hardware; (ii) have moderate computational costs; and (iii) require only extremely simple inputs from the user.

A final caveat is that the paper does not seek to quantify the benefits of multi-camera video chat, when compared to the single-camera approach. Any attempt to quantify these benefits suffers from a severe chicken-and-egg problem: until an ecosystem of multi-camera software and users has evolved, the overwhelming majority of video chats will be unconsciously engineered, by their participants, to be suitable for single-camera use. Hence, it is difficult to quantify the benefits of multi-camera chat. Nevertheless, there is good evidence that these benefits exist. For example, this paper describes two scenarios that are difficult or impossible without multiple cameras: the "children in the background" scenario of Section 5, and the "whiteboard lecture" scenario of Section 6. Anecdotal evidence[1] suggests that remote music lessons represent another compelling scenario for multi-camera chat. There is also some explicit evidence of demand for multi-camera chat in general: several existing systems offer it (see Section 3 for details). Furthermore, the MultiCam software package introduced in this paper, despite being a relatively immature research prototype with no publicity beyond a single posting to two Skype forums, is being downloaded dozens of times per month at the time of writing.

[1]This evidence comprises multiple unsolicited messages sent to the author from music teachers wanting multi-camera functionality.

2. OVERVIEW OF MULTICAM USAGE

The experiments described later employ a software package, called MultiCam, written by the author specifically for this research. It is not the primary contribution of the paper, but the MultiCam software does have novel aspects (see Section 3), and it is an important secondary contribution of the paper. MultiCam is free and open source, and is available for Microsoft Windows 7 and later. The local camera-switching functionality of MultiCam works, in principle, with any video chat software, since it relies only on installing a virtual camera. Remote camera-switching, on the other hand, works only with Skype, since it relies on Skype's so-called *desktop API* [13]. For concreteness, the remainder of the paper focuses on running MultiCam with Skype only.

MultiCam consists of two components: the *MultiCam application*, and the *MultiCam virtual camera*. The MultiCam application, shown on the left of Figure 2, is a stand-alone GUI application that allows the user to adjust settings and to perform camera-switching functions during a video chat. The MultiCam virtual camera appears, to the operating system, to be a standard video camera device. Video chat software such as Skype therefore offers "MultiCam" as one of the options when a user selects a video input device.

In reality, of course, the MultiCam virtual camera is not a physical camera. Instead, it multiplexes the machine's physical cameras: it passes video data from one or more of the physical cameras to the video chat software, possibly after transforming the data in some way. To be more specific, MultiCam has two high-level modes: *tiled*, and *non-tiled*—

Figure 2: MultiCam screenshots. Left: the MultiCam application. Middle: tiled mode display. Right: non-tiled mode display (note small thumbnails of the other cameras).

these are shown in Figure 2. The tiled mode places subsampled versions of the input from each physical camera into a single output image. When in non-tiled mode, one of the physical cameras is designated by the user as the *primary camera*. The input from the primary camera is transferred unaltered to the output image, but some small subsampled versions of the other (non-primary) physical cameras are overlaid at the bottom left of this output. The MultiCam application permits users to switch the identity of the primary camera, and to switch between tiled and non-tiled modes, with a single keystroke or mouse click (see Figure 2, left panel).

This brings us to the most important design decision for the MultiCam UI: how should the user switch cameras? Some early experimentation demonstrated that control of multi-camera chats can be bewildering, so MultiCam permits only two actions: *advance* the local camera, or *advance* the remote camera. The word "advance" here has a specific technical meaning, defined as follows. The N cameras connected to a machine have numerical IDs $1, 2, \ldots N$. If the system is in non-tiled mode when the user advances the camera, the ID of the primary camera is incremented by one—except that, if the primary camera ID is already equal to N, the system switches into tiled mode. If the system is currently in tiled mode, it switches to non-tiled with primary camera ID equal to 1. Thus, the user cycles through the $N + 1$ possible views in a fixed order. There is some evidence from the user study (Section 6) that users would prefer to have a method of jumping directly to the desired view, but investigation of this is left for future work.

MultiCam permits one additional method of switching cameras. When the MultiCam application is running, any Skype instant message (IM) received from the remote participant advances the local camera setting. In other words, if A is chatting with B, and B is running MultiCam, then any IM from A to B advances A's view of B's cameras. Importantly, this works when A is not running MultiCam. Because every Skype client includes the IM feature, A could even be on a device or operating system for which MultiCam is not available. This *switch-by-IM* feature can legitimately be described as a hack, since it is an abuse of the intended functionality for Skype instant messages. But it is a useful hack: on most Skype clients, switch-by-IM lets the user advance the remote camera with two keystrokes, provided the focus is already in the IM box. Specifically, the user hits one printable key, followed by the Enter key to send the message.

3. RELATED WORK AND CONTRIBUTION

In this section, we survey two strands of related work: (i) multi-camera video chat, and (ii) more immersive telepresence projects. It is claimed that this paper occupies a vacant niche in the literature, because academic projects and publications have focused on (ii), whereas this paper focuses on (i). More specifically, software for (i) has been available for at least a decade, but the utility and feasibility of such software—especially the possibility of listener-controlled camera-switching—has not been rigorously analyzed. This paper provides that analysis.

Multi-camera video chat software and hardware: Several existing software products offer convenient ways for the speaker to switch between cameras during video chat. These include ManyCam, WebcamMax, and VH MultiCam Studio (VHMS). The first two are limited to two simultaneous physical camera inputs, and don't permit listener-controlled switching; VHMS permits listener-switching via an interface suitable for advanced users only. All three products are closed-source.

A relatively recent development is the emergence of mobile devices and tablets with two cameras (e.g. Apple's iPad 2, HTC's Droid Incredible 2). These devices have one camera on the front, intended primarily for video chat; and one on the back, intended primarily for capturing photos and video. But of course it is possible to use both cameras during video chat, and some chat clients already support this at the time of writing (e.g. Google Talk, Skype Mobile). These clients support convenient, intuitive speaker-controlled switching between the two cameras. However, they do not support simultaneous views of both cameras, nor do they support listener-controlled switching.

Although outside the scope of this paper, it's important to realize that multi-camera video chat could be enhanced by non-standard cameras. One simple but liberating possibility is the use of wireless cameras. Surprisingly, at the time of writing (March 2012), there is no Bluetooth camera suitable for consumer video chat available for Windows systems, and only one such camera for Apple systems (Ecamm's BT-1). Wireless IP cameras are another option, and smartphone cameras can be converted into wireless webcams via apps such as DroidCam and SmartCam. This is a very promising approach. Presumably, the ecosystem of consumer-friendly wireless webcams will expand significantly in the near future. Panoramic cameras represent another alternative for enhancing video chat. These have been previously explored

in academic research projects such as FlyCam [6], and are now available as relatively inexpensive consumer products such as the GoPano micro. Remote-controlled pan-tilt-zoom cameras are yet another interesting alternative to complement multi-camera chat.

In contrast to all the above alternatives, the MultiCam software presented in this paper offers single-keystroke (or mouse-click) switching by both speaker and listener, between an arbitrary number of cameras, and includes a tiled mode. Hence, there is a certain amount of novelty in the software itself, especially given that MultiCam is open source.

Immersive telepresence: The goal of this paper is related to, but separate from, the goal of immersive telepresence. In this paper, we seek to enhance the listener's experience by providing multiple views of the speaker's location, and by giving the listener control over switching between those views. In contrast, immersive telepresence seeks to enhance the listener's experience by creating the impression that the listener is immersed in the speaker's location (or perhaps a virtual location instead). Examples include BiReality [7], 3DPresence [5], ViewCast [15], Coliseum [1], and commercial telepresence systems such as Cisco Telepresence and HP Halo. Implicit in all these projects is the assumption that the quality of the listener's experience will increase with the extent and fidelity of the immersiveness. This assumption may be true in general—and is particularly apt for certain facets of communication such as gaze and gesture awareness [8]—but it does not preclude improving the listener's experience through other, simpler means. The goal of this paper is to do just that: without seeking immersiveness, we can give the listener more options and more control by employing multiple views.

Contribution of this paper: To summarize, the two primary contributions of the paper are: (i) it demonstrates the *utility and feasibility of multi-camera video chat* for certain applications, and (ii) it analyzes the desirability of *remote control over the camera view*. To the best of the author's knowledge, no previous publication has addressed these points in detail. Secondary contributions of the paper include: (i) it describes the design trade-offs inherent in building multi-camera video chat software; (ii) it offers an open-source solution to this problem, which is valuable both as a research platform and as a consumer software product that will hopefully assist the growth of the multi-camera chat ecosystem; (iii) it identifies several areas in which webcam manufacturers and video chat software developers could enhance their support of multi-camera use.

4. DESIGN OF MULTICAM

This section highlights key aspects of the MultiCam design; many details are relegated to the accompanying technical report [9]. As already discussed, the MultiCam software consists of two largely independent modules: the MultiCam *application* (`MultiCam.exe`) and the MultiCam *virtual camera* (`MultiCamFilter.dll`). The virtual camera is implemented in Microsoft's DirectShow framework [10]. In this framework, modules that create, consume, or transform multimedia data are known as *filters*. Hence we refer to the virtual camera as the *MultiCam filter* from now on.

Figure 3 gives an overview of how the MultiCam application and filter communicate with each other and with Skype. Four types of communication are used: standard Windows

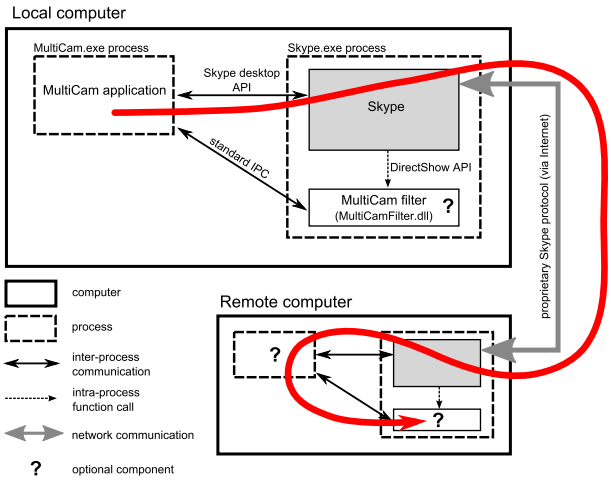


Figure 3: Communication between MultiCam components. The red arrow shows the route taken by an "Advance remote camera" request issued by a local MultiCam instance. The question marks indicate components that may be absent.

interprocess communication (IPC), API calls from the DirectShow framework [10], the publicly-available *Skype desktop API* [13], and the proprietary Skype protocol. As a concrete example, the red arrow in Figure 3 shows the chain of communication that occurs when a user clicks on "Advance remote camera" in the MultiCam application. A subset of the Skype desktop API known as "application to application" (AP2AP) messages is crucial here. The MultiCam application registers with the local Skype instance for AP2AP communication, permitting exchange of arbitrary data with a remote instance of the MultiCam application (assuming the remote MultiCam application is registered with a remote Skype instance). This exchange is tunneled through the proprietary Skype connection.

The chief abstraction in DirectShow is a directed graph known as a *filter graph*. Its edges represent paths along which video and audio data can flow, and vertices represent filters. The MultiCam filter is implemented so that it appears, from the point of view of any video chat software, to be a *source filter*—just like a physical camera. When the MultiCam filter detects that it has been added to a DirectShow filter graph, it immediately creates some new vertices in the graph—one for each physical camera in the system—and creates connections from the new vertices to itself. It is a simple matter to reassemble the frames from the physical cameras into the desired output, by subsampling and shifting the inputs if necessary, then placing them in the filter's output buffer.

Implementation of camera-switching: The choice of mechanism for switching cameras is perhaps the chief design decision for a multi-camera virtual camera. Let us initially ignore the possibility of tiled mode and concentrate on switching the current primary camera between two or more physical cameras. There are at least two obvious alternatives, which we will call *one-at-a-time* and *all-at-once*. The one-at-a-time approach uses exactly one camera at any instant: the DirectShow graph consists of the current primary camera as a source filter, connecting to the virtual cam-

era filter, which probably does nothing but pass the physical camera's frames untouched to the downstream filter. In this approach, the software needs to perform surgery on the DirectShow graph whenever the input camera is switched, which imposes additional latency.

The all-at-once approach connects source filters from all desired physical cameras to the virtual camera filter when the DirectShow graph is first created. Video data is continuously streamed from all cameras simultaneously, and the job of the virtual camera filter is to pass on the frames of the current primary camera while dropping data from the other cameras. Clearly, this consumes more resources than the one-at-a-time approach. However, it has the benefit of rapid camera switching, as the costly graph surgery operation is eliminated. Note that the all-at-once approach also permits arbitrary combinations of the input images, such as a tiled view of all cameras, or small overlay views of the other cameras placed on top of the primary camera view. Hence, MultiCam uses the all-at-once approach. Note also that MultiCam always copies the video data to an output buffer, rather than transforming it in place, as this simplifies the implementation of tiled mode. The benchmark experiments (Section 7) vindicate this decision, as the total resources consumed by MultiCam are perfectly acceptable, and in some cases less than competing approaches.

Managing heterogeneous resolutions, formats and frame rates: The all-at-once approach, by definition, outputs frames with a fixed resolution and format. Thus, it needs to address the fact that, when working with a heterogeneous set of cameras, the cameras may offer different resolutions and formats. Details of this can be found in the technical report [9]. Here, it suffices to say that MultiCam has a configurable target resolution. At startup, it requests that each camera output its largest available resolution no greater than the target, and performs explicit conversion into a common format (24-bit RGB) if the camera doesn't offer it. Camera outputs are subsampled and re-centered as necessary. MultiCam takes no explicit steps to address issues of timing and frame rate. It relies on default DirectShow behavior to manage the flow of data within the graph, which may include dropping frames if a given filter is operating faster than a downstream filter.

5. EXPERIENCE WITH MULTICAM

At the time of writing, MultiCam has been employed for a genuine Skype chat approximately once per week by the author, over a period of five months. Here, "genuine" means that the chat was not part of a deliberate experiment, and its primary purpose was communication with friends or family. In every case, the reason for using multiple cameras was that one or more additional family members were present and I wanted to include them in the video stream. Obviously, the impressions gained from this experience have limited scientific rigor, but it nevertheless seems useful to report briefly on the experience.

With rare exceptions, the remote participants showed little interest in controlling the cameras. In general, therefore, I was not relieved of the burden of camera-switching. On the other hand, I felt the total effort of camera management was significantly reduced in most cases. Rather than constantly having to adjust a single camera to show the current region of interest, I was frequently able to leave the cameras in a fixed position for long periods and simply switch between them.

Figure 1(a)–(c) shows the three camera setups that proved most useful in these conversations. In Figure 1(a) we see a two-camera scenario in which one camera is perched on a laptop for a headshot of the main Skyper, and another camera is on the table, trained on a child in the background. Figure 1(b) shows another two-camera scenario, again with one camera capturing the standard Skyper headshot. The other camera is also perched on the laptop, but faces the opposite direction. This mimics the setup of dual-camera smartphones and tablets, but with more flexibility, since the exact direction of the cameras can be adjusted individually. In this scenario, I often pick up the outward-facing camera and direct it manually for a period of time before placing it back on the laptop.

Figure 1(c) shows a three-camera scenario. Skype is still being run from a laptop, but using a living room TV as a display. The remote participant's tiled mode view of this scenario is shown in Figure 1(e). One camera is mounted on top of the TV, showing a wide view of the entire scene. Another camera is perched as usual on the laptop for a headshot of the laptop controller. A third camera, on the arm of a sofa at this particular instant, is available to be moved around as needed, capturing the activity of a small child on the floor. This setup has been particularly successful for group events, such as opening presents, in which attention naturally focuses on different people at different times.

6. USER STUDY

A user study was conducted to examine some of the benefits and drawbacks of using multiple cameras with video chat, focusing especially on a comparison between speaker-controlled and listener-controlled camera-switching.

6.1 Participants

A group of 23 individuals was recruited to participate in the study. Participants were all acquaintances of the author who voluntarily responded to email requests or similar; the resulting participant pool comprised friends, family, colleagues, and one student. Participants' ages ranged from 20 to 70 (median 40). Two participants were new to Skype; the remainder had frequently used Skype for single-camera video chat. Two participants had used the MultiCam camera-switching functionality previously; of the remainder, four had some knowledge of the MultiCam project, and the remaining 17 participants had no knowledge of it. Nine of the participants could reasonably be described as technically savvy (i.e. work in a computer-related profession, or maintain an active amateur interest in technology); the remainder had no particular skills or affinity with computer technology. Geographically, there was a three-way split between participants: five in the same North American town as the author, eight in other North American locations, and ten outside North America (all either Europe or Oceania). Approximately 70% of participants employed laptop monitors, with the remainder using larger desktop monitors. Fourteen users employed a single webcam at their own end of the conversation; nine used no camera at all; none used multiple cameras. Hence, although the sample is relatively small and was not selected via random sampling, it contains a good cross-section of video chat users.

6.2 Method

The user study employed the two-camera setup shown in Figure 1(d), in which a person (the speaker) can sit on a sofa and communicate with the study participant (the listener), using a whiteboard adjacent to the sofa when desired. We will refer to this video chat scenario as the *whiteboard lecture scenario*. One camera, positioned on top of the laptop, presents a head-and-shoulders view of the speaker sitting on the sofa. The other camera, positioned on the desk, displays the whiteboard. Thus, exactly 3 views were available to study participants: the speaker, or the whiteboard, or a tiled view of both. The tiled view is shown in Figure 1(f). The whiteboard is positioned such that, on a typical monitor and under typical Skyping video quality, writing on the whiteboard can be read reasonably easily when the whiteboard camera is the primary camera, but is not very legible in the tiled view. This is important because it provides an incentive to switch between views; otherwise, it would probably be optimal to remain in tiled view at all times, and this would reveal no useful information comparing local and remote camera control.

As will be described in more detail shortly, participants needed the ability to switch between the three camera views in this study. As explained in Section 2, the only camera-switching method guaranteed to be available to all users is the switch-by-IM method. For consistency, therefore, all participants used the IM method for switching cameras in this study.

Each user in the study participated in a Skype session with the author, lasting about 10 minutes. The core of the session involved two three-minute lectures, delivered by the author using the whiteboard and a handheld prop. The most important feature of the session was that in one of the three-minute lectures, the speaker had exclusive control of the camera-switching, and in the other lecture, the listener had exclusive control. The ordering of these two camera-control options was alternated for each participant, so the first lecture was speaker-controlled in half of the sessions. Both mini-lectures involved the same routine of alternately talking directly at the camera while sitting on the sofa, and writing on the whiteboard. The specific set of states for each lecture was: sofa, whiteboard, sofa, whiteboard, sofa. The middle "sofa" segment involved, for both mini-lectures, the use of a handheld prop (actually a paperback book that was opened to show some example data). Hence, even listeners who might have been happy to stare at a whiteboard while listening to a disembodied voice had an incentive to switch back to the sofa view during the middle segment.

Precise details of the session script and the questionnaire administered at the end of each session are given in the technical report [9]. The most important questions gauged whether the users preferred speaker-controlled cameras, listener-controlled cameras, or neither. Other questions asked users to list any aspects of the experience they liked or disliked during the speaker-controlled and listener-controlled segments. Users were also asked how much they used the tiled view, and a final open-ended question asked for any further comments or feelings about the experience.

It is important to note that it is definitely not the goal of the study to evaluate the raw efficacy of the whiteboard lecture scenario for distance learning or collaborative web conferencing. The scenario is contrived solely to provide an easily-controlled, replicable situation in which remote and local control of camera-switching can be compared while keeping other factors constant. Indeed, numerous software products targeted at distance learning and web conferencing are available,[2] and MultiCam is not envisaged as a direct competitor to these products. In fact, they are complementary: any such product receives input from a webcam, and can therefore be enhanced by using MultiCam-style virtual camera software to provide simultaneous multiple-camera functionality if desired.

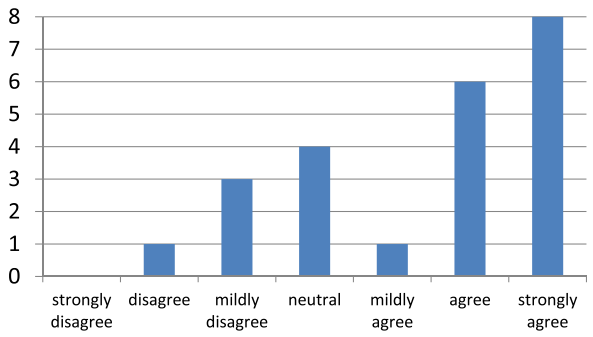


Figure 4: User preferences for speaker-controlled camera-switching vs listener-controlled camera-switching. Frequencies of agreement levels with the following statement are shown: "When the speaker controlled the camera, the overall experience was more satisfactory."

6.3 Results and discussion of user study

Camera control preference: Figure 4 shows the strength of participants' preferences between speaker-controlled and listener-controlled camera-switching. For simplicity, the graph shows results coded with Likert-type categories (i.e. the level of agreement or disagreement) applied to the statement "When the speaker controlled the camera, the overall experience was more satisfactory." However, to eliminate acquiescence bias,[3] the data was obtained in a different way, resulting in perfect symmetry between preferences for listener control and speaker control. Participants were first asked whether they preferred speaker control, listener control, or neither. Those expressing a preference were then asked to follow up by selecting from "strongly agree," "agree," or "mildly agree" in reaction to the statement "When the [speaker/listener] controlled the camera, the overall experience was more satisfactory." Of course, the word "speaker" or "listener" in this statement was selected according to the participant's previously-stated preference.

A glance at Figure 4 gives the strong impression that users preferred speaker-controlled camera-switching, and this impression is confirmed by statistical analysis. The median response is "agree"—the second-highest response on the 7-point scale. To check our intuition that this median differs

[2]For example: Elluminate and Wimba Classroom for distance learning; GoToMeeting and Microsoft's Live Meeting for web conferencing—to mention just two of the many products available in each category.

[3]*Acquiescence bias* is the tendency of respondents to agree with statements. See texts on psychology or market research for details (e.g. [11]).

Advantages of speaker control:

could concentrate more easily (not distracted by thinking about switching cameras)	9
lecturer can anticipate the need for a switch and thus switches at the right time	8

Advantages of listener control:

had control over the experience	10
had the ability to go back to the whiteboard when desired	5
concentrated better because had to pay attention	2

Disadvantages of listener control:

poor interface for switching cameras	8
harder to concentrate/distracting to switch cameras	5
switching delay was annoying	4
lose a few seconds of attention at every switch	3

Figure 5: Theme analysis of user study comments. All themes that occurred twice or more are listed, with the frequency of occurrence in the right column.

by a statistically significant amount from the "neutral" response, we can perform a chi-squared test of the null hypothesis that the population median is "neutral." To do this, restrict attention to the 19 participants who expressed a preference: 4 for listener control and 15 for speaker control. If the null hypothesis held, we would expect 9.5 in each category. Computing a chi-squared statistic in the usual way, we find $\chi^2 = 6.37$ on 1 degree of freedom, which yields a p-value of 0.012. Hence, we can reject the null hypothesis at, say, the 2% level of significance, and conclude that there was a statistically significant preference for speaker control.

On the other hand, we also see that the results were not a complete landslide for speaker-controlled camera-switching: 15 participants expressed a preference for speaker control, and 8 did not. A simple statistical analysis (see [9]) concludes from this that a significant minority (perhaps a quarter to a half) of the population does *not* prefer speaker control.

Combining the conclusions of the previous two paragraphs, we see that for the particular whiteboard lecture scenario tested, an ideal multi-camera system would function primarily by speaker-controlled switching, to satisfy the statistically-significant preference of the population for speaker control. However, the ideal system would also permit control by the listener (to whatever extent desired), which is especially important for the significant minority of listeners who prefer to be in control.

Pros and cons of camera-switching options: Figure 5 lists all the important themes to emerge from the questions asking participants to list any likes or dislikes of the two camera-switching options (speaker control and listener control). The figure shows any theme that was mentioned by at least two participants. Classification of responses was done by the author, and is of course subjective. Nevertheless, several clear points emerge.

The strongest reason for liking speaker control was that it was easier to concentrate on the content of the lecture—these participants considered camera control a burden, and devoting thought to camera control detracted from the attention that could be paid to the lecture itself. For example, one participant stated: "I can concentrate on the speaker, not on the technology." A related but separate point is that the speaker knows in advance when a switch will be required, and thus is able to time the switches appropriately. In contrast, the listener realizes a switch is required only *after* the speaker performs whatever action triggers the need for a switch. Thus, even for a user who does not find camera-switching burdensome, listener control has the disadvantage that most camera switches occur late. One participant spoke of losing "a few seconds" of relevant viewing at every such switch.

No themes for disliking speaker control emerged; the only comment in this category was from a single participant, who noted that he or she "couldn't check something on the whiteboard."

The strongest reason for liking listener control was the somewhat tautological notion of being "in control." Some participants perceived explicit educational value in being able to time their own switches, especially for lingering on, or extra glances at, the whiteboard. In fact, four of the five users who mentioned the ability to go back to the whiteboard as an advantage of listener control actually preferred speaker control in general. This is important, as it demonstrates that even users who prefer speaker control can benefit from the ability to seize control occasionally. A more subtle and surprising effect was also apparent: some users derive intrinsic satisfaction from being in control, without necessarily perceiving a causal link to an educational outcome. Comments along these lines include: "it was kind of fun to be the one in charge," and "the part of me that likes to flip through the channels liked it." Two participants preferred listener control for another surprising reason: they found the requirement to be alert and ready to switch cameras when necessary forced them to pay more attention to the lecture, resulting in a more satisfactory outcome. This reasoning directly contradicts the 10 users who found camera-control detrimental to concentration—more evidence that the user base has diverse preferences and multi-camera video chat should try to account for them.

The main stated disadvantage of listener control was the poor interface for switching cameras. There were two aspects to this. As remarked above, remote camera-switching was performed via switch-by-IM, which requires a minimum of two keystrokes and, more importantly, is not at all intuitive. It is not surprising that users disliked this. However, six users were also frustrated by having to cycle through the three view settings in a fixed order. This calls into question one of the hypotheses on which the MultiCam interface was based: namely, that switching between multiple views, including a tiled view, is excessively complex and that the simplest possible interface (a single advance-to-next-view operation) is therefore preferable. It seems this hypothesis is not correct for a significant fraction of users. Thus, alternative interfaces should be explored in future multi-camera chat systems.

Another important dislike of listener control was the delay between requesting a switch and receiving it. Average round-trip times were not recorded during the user study chat sessions, so it is not known if these complaints correlate with large network latencies. (Two of the four who

mentioned this problem were in Oceania, but the other two were in North America—the same continent as the lecturer.) In any case, it is interesting that delay was perceived as a disadvantage specific to *listener* control. Speaker-initiated switches would have suffered delays of similar magnitude (although perhaps up to 50% less, depending on the root cause), but were not perceived as problematic.

Use of tiled mode: It is natural to wonder whether multi-camera video chat systems should provide a tiled mode: is it a beneficial feature, or does it just clutter the interface and confuse the users? The user study was not specifically designed to answer this question, and the utility of tiled mode clearly depends on the application. Nevertheless, we can glean a little insight from the participants' responses. Two participants chose to use tiled mode most of the time during the listener-controlled mini-lecture. A further nine participants used tiled mode at least once. The remaining 12 participants did not use tiled mode. Hence, it seems that for this application at least, tiled mode is attractive to a significant fraction of users.

6.4 Conclusions from the user study

The main conclusion of the user study is: for the whiteboard lecture scenario, a majority of users prefer speaker-controlled camera-switching to listener-control, but a significant minority do not. Note, however, that care is needed when extrapolating this conclusion beyond the particular version of the whiteboard lecture scenario tested. Indeed, even if we restrict consideration to the whiteboard lecture scenario, it seems clear that generalization is problematic. This is because certain aspects of the scenario could be varied in such a way as to produce preferences tilted strongly towards speaker or listener control. For example, the speaker could have deliberately "forgotten" to switch cameras several times during the speaker-controlled test.[4] This would be immensely frustrating to the listeners, and could be made as extreme as desired, resulting in virtually 100% of participants expressing a preference for listener control. On the other hand, the speaker could have made listener control difficult and frustrating by frequently moving on and off the whiteboard, picking up props for only one or two seconds, and making very brief references back to the whiteboard, all without verbally telegraphing any intentions.

These thought experiments demonstrate that preference for listener- or speaker-control is highly application-dependent. And there are two other factors that may have influenced the results: (i) the use of the non-intuitive switch-by-IM method for switching cameras; and (ii) the fact that the vast majority of participants had never used MultiCam before, and had only a brief 30–60-second practice session to gain familiarity with switching cameras. Both of these factors would tilt the results towards a preference for speaker control.

But the application-dependence and other sources of variability do not render our conclusions from the user study irrelevant—they simply mean we must be careful in making generalizations. For example, it would be wrong to conclude that a majority of users prefer speaker-control to listener-control for multi-camera video chat in general. On the other hand, it does seem reasonable to infer the following conclusions:

- For any given multi-camera video chat scenario, there can be both a significant proportion of users who prefer local control of camera-switching, and a significant proportion of users who prefer remote control.
- Even users who have a preference for not controlling the camera-switching in a given scenario can derive benefits from seizing control occasionally.
- A significant fraction of unpracticed users find that controlling the cameras detracts from their ability to concentrate on the video chat (but this may not be true of users with substantial practice, especially if a more convenient interface than the switch-by-IM were provided).
- Significant delays between a switch request and its execution can be a source of frustration.
- Tiled mode is useful for a significant fraction of users.

7. BENCHMARK EXPERIMENTS

This section describes experiments to investigate the resource usage and performance of MultiCam. The experiments employ four different USB 2.0 webcams: a Logitech QuickCam Chat, a Logitech QuickCam Easy/Cool, a Microsoft LifeCam VX-3000, and a Microsoft LifeCam HD-3000. These are all low-cost cameras ($20–30 at the time of writing), in keeping with the goal of targeting consumer video chat. The selection of cameras is heterogeneous for two reasons: (i) it allows us to investigate the amount of variability in resource usage and performance between these cameras, and (ii) it is perhaps more representative of a consumer whose collection of webcams has grown piecemeal over time.

Experiments were conducted on two different machines: a relatively recent (2011) standard office desktop with four 2.66 GHz cores, and an older (2007) laptop with two 1.83 GHz cores. The technical report [9] contains detailed results of all experiments, but due to space constraints we focus on the desktop results here. The desktop machine was a Dell Optiplex 780, with a 2.66 GHz Intel Core2 Quad (Q9400) processor, 16 GB of main memory at 532 MHz, an ATI Radeon X1550 GPU, and eight USB 2.0 ports.

Experiment 1: MultiCam resource usage: The objective of the first experiment is to measure the resource usage of MultiCam with up to four cameras, both in isolation and as part of a video chat. The two primary resources consumed by the cameras are (i) CPU, and (ii) bandwidth of various internal buses, especially USB buses. In this experiment we report CPU utilization directly, whereas the effects of bus saturation are demonstrated indirectly, by measuring the video frame rate of the MultiCam filter. There was a separate run of the experiment for each nonempty subset of the four cameras, resulting in a total of 15 camera combinations. Each camera set was tested in two ways: a *raw run* and a *Skype run*, described next.

A *raw run* consisted of executing a simple benchmark program that displays a MultiCam video stream on the monitor in tiled mode. Specifically, this was a lightly-altered version of the PlayCap example code in DirectShow. Note that the

[4]In fact, this did happen twice, by accident. Participants were instructed to disregard the mistakes, but they may have been influenced anyway, of course.

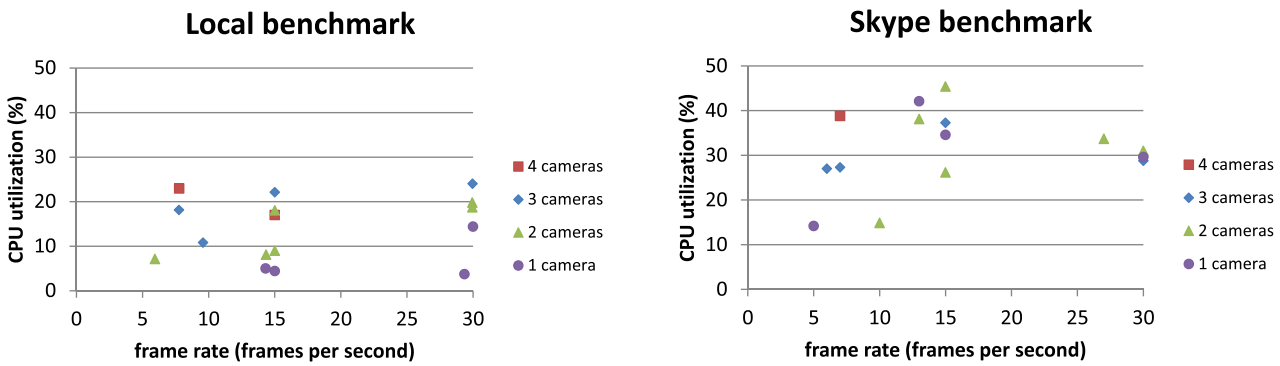


Figure 6: MultiCam CPU utilization and frame rate. Left: the PlayCap local display benchmark. Right: Skype benchmark.

raw runs therefore did not involve any video chat software—the objective was to measure the bare-bones resource consumption of the cameras connected to the MultiCam filter, without any additional overhead for video chat. A *Skype run* consisted of a Skype video chat between the two experiment machines described above. Only one end of the chat transmitted video in each run. The laptop was connected to a residential broadband service via 802.11g wireless, and the desktop employed a wired connection to a university campus network. The physical distance between the two machines was about 0.5 miles, and Skype reported the vast majority of round-trip times in the range 50–60 ms.

As discussed in Section 4, cameras are always requested to deliver data at a target resolution of 640×480 with 30 fps,[5] in 24-bit RGB format, and are subsequently subsampled by the MultiCam filter if necessary for display in tiled mode.

Figure 6 shows the results of the raw and Skype runs. There are several interesting features of these results. First, CPU consumption by the cameras is approximately additive, but with considerable variation. (In other words, using two cameras costs, on average, twice as much as one camera, and similarly for three and four cameras.) Second, there is a weak but clear relationship between frame rate and CPU utilization ($R^2 = 0.27, 0.67, 0.51$ for $1, 2, 3$ raw cameras respectively). Third, by comparing raw and Skype runs, we see that Skype adds significant CPU overhead to the local display benchmark—the average overall jumps from 9% to 31%—and in some cases this resulted in a lower frame rate. Presumably, this overhead is primarily due to Skype's proprietary compression and encryption, which have been analyzed in several prior works (e.g. [2, 16]).

Fourth, there can be great variation in the CPU cost and performance of individual cameras. Examples include: (i) three of the cameras consume about 5% CPU, but the remaining camera consumes about 15%—three times as much; (ii) the offending camera consumes only half as much CPU (about 8%) when switched from a USB port on the front of the machine to one on the rear;[6] (iii) one raw two-camera combination languishes at 6 fps—worse than any three-camera combination. Such mysterious results were not investigated further. Presumably they derive from subtle interactions between several hardware and software modules, including the camera drivers, the USB controllers, and the DirectShow framework. The main point is "video chatter beware!": individual camera performance can vary for obscure reasons, and these variations can be exacerbated when using multiple cameras simultaneously.

The good news is that this experiment did uncover some sweet spots: for example, we see from the Skype runs that even during a video chat there exist sets of one, two or three cameras that can operate at 30 fps for less than 30% CPU. On the more impoverished laptop setup omitted here, the most CPU-intensive runs still leave some room for other tasks to use the CPU. Hence we can conclude that multi-camera video chat is comfortably feasible on consumer PC hardware.

Experiment 2: Display latency of multiple cameras: It has been shown that for audio calls, Skype users' satisfaction is much more strongly influenced by the transmitted signal's bitrate and jitter than by its latency [4]. But there do not appear to be any similar results for video chat, so it seems desirable to understand whether or not the simultaneous use of multiple cameras affects video latency. Experiment 2 investigates this, again using the PlayCap local display benchmark. (Network latency is thus excluded—not because it is unimportant, but because it is a constant added to any delay due to multiple-camera use.) Please see the technical report [9] for details of the experimental setup.

Measurements were made for each of the four webcams used in the previous experiment. More specifically, each camera's latency was measured in two scenarios: (i) the given camera is the only one connected to the MultiCam filter (the others might as well be disconnected; they have no effect on the system), and (ii) all four cameras are connected to the MultiCam filter and are simultaneously displayed in tiled mode, but we measure latency in the tile whose content comes from the relevant camera.

Figure 7 shows mean and standard deviation for each camera and scenario. For any given camera, we see a relatively small difference between the single-camera and four-camera scenario; two of these differences are increases and two are decreases. Hence, it seems safe to conclude that simultane-

[5]It turns out that one of the cameras (the VX-3000) can only support up to 15 fps at this resolution.

[6]This also explains the two data points in the 4-camera raw category: one camera was swapped from the front to the back to produce a second 4-camera data point.

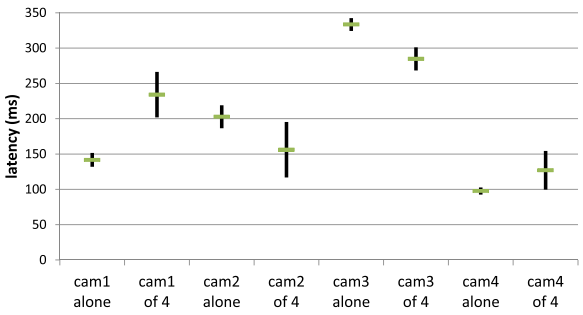


Figure 7: MultiCam display latency for single and multiple cameras. Horizontal lines show the mean and vertical lines show the standard deviation of the 10 latency measurements taken for each scenario.

ous use of up to four webcams does not significantly alter the latency of video observed by video chat users. We can also see the dramatic differences in latencies between different cameras—as much as a factor of 3, ranging from 100 to 300 ms. This is yet another example of the "chat user beware" maxim emerging from these experiments.

Experiment 3: Comparison with other multi-camera software: Some aspects of MultiCam's design sacrifice resources for performance. These design decisions were vindicated by performance experiments described in the technical report [9]. The results show that MultiCam is comparable to competitors in terms of CPU utilization (±5% of CPU), while achieving 2–3 times better camera-switching latency (330 ms, compared to 749 ms and 1122 ms for the two competitors tested).

8. CONCLUSION AND FUTURE WORK

The most obvious opportunity for future work is to incorporate non-standard imaging devices, such as panoramic cameras, into the mix of cameras. Improving the UI for camera-switching should be a priority, perhaps using image stitching [3] to combine views, or navigation between views inspired by the Photo Tourism of Snavely *et al.* [14].

This paper has also highlighted some areas in which webcam manufacturers and video chat software developers could improve the multi-camera chat experience. These include: standardizing a protocol for remote camera switching (the technical report [9] provides specific suggestions); providing cameras with a "ready" mode, whereby they can begin transmitting video data upon request, essentially instantaneously; addressing the unpredictable CPU usage, frame rate, and latency of cameras identified by Experiments 1 and 2 (Section 7).

Multi-camera video chat seems to be a promising and underutilized tool in the multimedia milieu. This paper has demonstrated the feasibility of multi-camera chat on standard consumer hardware, and suggested scenarios in which multiple cameras improve the chat experience. A user study provided strong empirical findings on the advantages and disadvantages of listener-controlled switching between camera views. Some design trade-offs inherent in multi-camera chat software were discussed, and the paper also presented MultiCam, an open-source package providing multi-camera chat. Perhaps researchers, software developers, and hardware designers can build on these ideas to provide rich, easily-controlled, multi-view video chat experiences in the future.

9. REFERENCES

[1] H. H. Baker et al. Computation and performance issues in Coliseum: an immersive videoconferencing system. In *Proc. ACM Multimedia*, pages 470–479, 2003.

[2] S. A. Baset and H. G. Schulzrinne. An analysis of the Skype peer-to-peer internet telephony protocol. In *Proc. INFOCOM*, pages 1–11, 2006.

[3] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *Int. J. Comp. Vision*, 74(1):59–73, Aug. 2007.

[4] K.-T. Chen, C.-Y. Huang, P. Huang, and C.-L. Lei. Quantifying Skype user satisfaction. In *Proc. ACM SIGCOMM*, 2006.

[5] O. Divorra, J. Civit, F. Zuo, et al. Towards 3D-aware telepresence: Working on technologies behind the scene. In *Proc. ACM CSCW: New Frontiers in Telepresence*, 2010.

[6] J. Foote and D. Kimber. FlyCam: practical panoramic video and automatic camera control. In *Proc. ICME*, volume 3, pages 1419–1422, 2000.

[7] N. P. Jouppi, S. Iyer, S. Thomas, and A. Slayden. BiReality: mutually-immersive telepresence. In *Proc. ACM Multimedia*, pages 860–867, 2004.

[8] C. L. Kleinke. Gaze and eye contact: A research review. *Psychological Bulletin*, 100:78–100, 1986.

[9] J. MacCormick. Video chat with multiple cameras. Technical report, Dickinson College, 2012. Available as `http://arxiv.org/abs/1209.1399`.

[10] M. D. Pesce. *Programming Microsoft DirectShow for Digital Video and Television*. Microsoft Press, 2003.

[11] G. Richards. *Psychology: The Key Concepts*. Routledge, 2008.

[12] Skype S.A. United States Securities and Exchange Commission Form S-1 registration statement (preliminary prospectus), 2010. Registration number 333, Filed August 9, 2010.

[13] Skype SA. *Skype Desktop API Reference Manual*, 2011. Available from `http://developer.skype.com/public-api-reference`.

[14] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *Proc. ACM SIGGRAPH*, pages 835–846, New York, NY, USA, 2006. ACM Press.

[15] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy. ViewCast: view dissemination and management for multi-party 3D tele-immersive environments. In *Proc. ACM Multimedia*, pages 882–891, 2007.

[16] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang. Profiling Skype video calls: Rate control and video quality. In *Proc. INFOCOM*, 2012.