Video Traces

Michael N. Nunes
University of Calgary
nunes@cpsc.ucalgary.ca

ABSTRACT

In this paper we present video traces, a project that looks to explore the design space for visualizations showing the history of a video stream. When using video channels to create a media space for communication, it becomes possible to broadcast countless hours of streaming video. If we are to understand the activities and events taking place in the space, we need methods that allow us to navigate the video history to find these events. In this paper we present and discuss several example visualizations of video history. We consider how certain visualizations may be useful, what their limitations may be, and what issues may arise from their use.

General Terms

Design, Human Factors.

Keywords

Video communication, groupware systems, awareness, traces.

1. INTRODUCTION

The use of a video channel to augment communication amongst remote collaborators is becoming increasingly popular with the wide availability of inexpensive web cams. The capability to include an office or a public space within a larger media space becomes easy as the supporting technology is made available, but this leads to the potential for countless hours of video footage to be broadcast within the media space. With this wealth of video data available, the opportunity to explore new techniques that allow users to easily discover the activities and events occurring within the media space arises.

The video traces project presented here is an exploration of several visualizations that allow users to quickly explore the history of a video stream from a media space. These visualizations allow us to display video over time, which we can use to gather information from more than just a single instant of video at once. The use of these visualizations is intended to provide an observer with an easy means to discover and understand the history of activities and events that have occurred within the media space.

Traditional methods of navigating video include fast-forward and rewind buttons, or seek bars. These methods may work well for quickly perusing the content of a video, or skipping to an approximate location, but they can be overly cumbersome when attempting to look for specific events within a large video stream. This is particularly so when the observer does not know when an event has occurred and must manually search through hours of video to find what might constitute only a few seconds of activity. In the situation where the video stream is used to provide causal awareness, the user may only periodically look at the stream and will not know about potentially important events that may have occurred and been missed. Because they are unaware of the missed events, they have no motivation to expend the effort to look through the incoming video stream.

The objective of this project is to explore the design space for visualizations that allow observers to easily explore the history of a video stream. In order to accomplish this we have developed several example visualizations that will be presented. From these examples we hope to show how different visualizations of history in a video stream might be used. We examine the potential strengths and weaknesses of the visualizations, and consider the issues that may arise from their use in systems for group awareness and communication.

2. RELATED WORK

Hill and Hollan presented a shared document editor called Edit and Read Wear in which the interface elements showed artifacts of past interaction [4]. This interface design shows how visualization of the past can be used to promote change awareness in a groupware system. The visualization of interaction history provides users with the capability to "see through time"; this capability is not possible in the real world, but can be implemented in the virtual world to help users comprehend past activities and events.

The traces work by Gutwin [3] looked at applying visualizations of the recent past to virtual embodiments of users in a groupware system rather than to interface elements. This work augmented the telepointers representing users with a visualization of recent movement history, such as motion blur, to increase understanding of their actions over time. It was shown that the telepointer traces were useful in combating service degradation during conditions of network jitter; when delay occurred telepointer updates would pile up resulting in jerky motion, but the original movement could be reconstructed and shown as a trace. Gutwin also suggested that traces could similarly be used with video to allow for casual awareness, noting that people might only glance at a video stream periodically and may miss important events.

The "When Did Keith Leave?" system presented by Hudson and Smith [5] is an example of a system for casual awareness where a recent history of images is pulled from a video stream showing the last few moments of heavy activity detected within the space. The main focus of this system was to demonstrate how privacy issues might arise when providing awareness information; the activities of a person within the space will not be hidden within a large video stream, they are extracted and displayed in a way that makes them persistent and easily visible. It is also noted that while this system may cause issues of privacy, it is powerful in that it easily allows users to see the recent activities and events that have occurred within the space.

Other projects have similarly looked into the display of video over time. The Video Streamer project by Elliott and Davenport [2] looks at how more than an instant of video can be viewed at once. In the Video Streamer, video is shown streaming into view and forms a three-dimensional solid block. The intent of this project is to look at how video can be turned into a malleable object with the focus being on its use for video editing, rather than how it might be used to provide awareness of activities and events within the video stream.

In Stylized Video Cubes [6], Klein et al. used the metaphor of video as a cube to construct a tool for non-photorealistic rendering and processing of video data. The two-dimensional video frames were used to construct a three-dimensional cube that could be sliced and used as a rendering solid. In this project, the metaphor of video as a cube was presented, but was being used for rendering rather than for navigating or extracting information from the video being processed.

In Artifacts of the Presence Era, Viegas et al. [7] created an art installation system that provided a temporal visualization of a video stream from a museum. The system used a rock formation metaphor to construct the visualization. Layers were added to the visualization by periodically taking slices from video frames. Also, as new layers were added to the top of the visualization, older layers were compressed by combining multiple frames and replacing them with the resulting image. Visitors were able to navigate the visual history accumulated by the system via a knob that allowed them to uncover the past images captured within the visualization. The intent of this project was to create an interesting visualization for the passage of time, and to preserve memories of the museum space. The visualization did allow for navigation of the video history, however, it was not specifically intended to be used as a way to retrieve information from the video stream.

The Last Clock project demonstrated by Angesleva and Cooper [1] presented a visualization that shows the dynamics of space over time. A slice from a live video feed was used to populate a visualization that resembled the face of a clock. As the hands move around the clock, they replace the underlying portion of the clock face with a single row slice from a video stream. The Last Clock was intended as a pleasing visualization to represent the passage of time rather than as a means for extracting information from the video stream. No means for navigating the accumulated video history was provided.

3. VISUALIZATIONS

The related works described in the previous section show examples of how visualization of past events can be useful in providing awareness, and described a few ways in which the history of video data can be visualized. However, the focus of the example visualization projects were geared more toward visualizing the passage of time rather than seeing how these visualizations may help us find and understand the activities and events occurring within the video stream.

In this section we present several example visualizations we built, and discuss how they might be useful in navigating a video stream, what their limitations are, and what issues arise from their use.

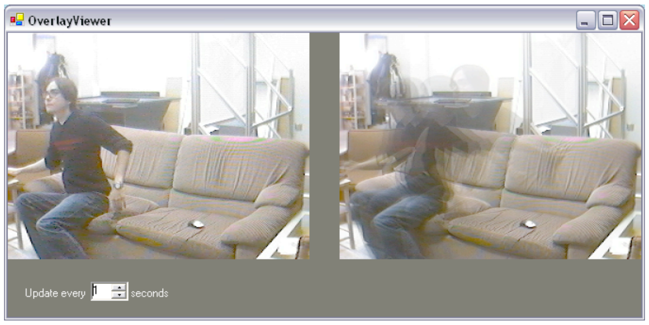

Figure 1: Motion blur visualization applied with an update rate of once per second. Notice that the visualization on the right shows a blurred image of the subjects movement.

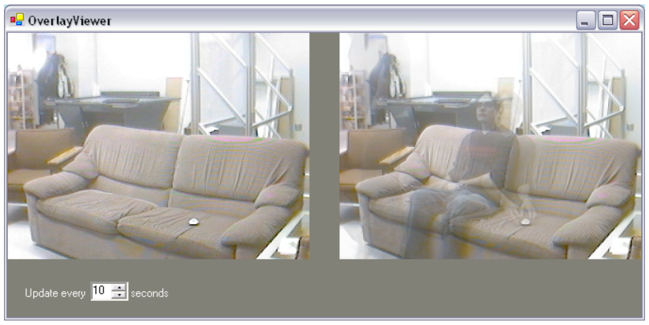

Figure 2: Motion blur visualization applied with an update rate of once every ten seconds. The visualization on the right shows ghost images of the subject in different areas of the space.

3.1 Motion Blur

The first of the visualizations we have developed and will look at is motion blur. The motion blur visualization allows us to see more than an instant of video within a single frame. This is done by compositing an update frame from several past frames.

Our demo application using the motion blur visualization can be seen in figures 1 and 2. In the examples shown, the real time video stream is displayed on the left, and the image updates using the visualization are displayed on the right. In our demo application image updates occur on a regular interval that is adjustable. The motion blur image can be updated with a frequency of once per second or lower. The updated images displayed are a composite of five images that have been sampled over the period of time since the last update. The result is an image that shows traces of the activities and events that have occurred over the period of time since the last update.

It is notable that when using motion blur with a fairly rapid update rate (around once every second) moving objects in the video stream simply appear blurred along their path of motion. An example of this is shown in figure 1 where the visualization is being used with an update rate of once per second; we can see the blurred movement of the subject getting up. However, if a slower update rate is used the update visualization shows ghost-like images as people move about the scene. An example of this is shown in figure 2 where the visualization is being used with an update rate of once every ten seconds. We can see that the subject has been moving about the room recently as evidenced by their ghost appearance in multiple positions, but we do not see the complete set of actions that may have taken place over the period of time.

This type of visualization may be useful for casual awareness as suggested by Gutwin [3]. Observers may only glance at the video stream periodically, and as such may miss important events such as a subject arriving at their desk in the morning and then leaving to meet with another colleague. Another use for this type of visualization may be to help reduce network load. When several full frame rate video streams are being broadcast amongst users in a system for awareness, the strain on the network can be high. If the video frame rate is lowered to accommodate for the available network bandwidth, users may similarly miss important events. By sending an image that is composed of several frames sampled over the update interval, we can reduce the network load and provide some information detailing the activities that have occurred between updates.

A limitation of this visualization is that it can only be used to display a fairly recent history adequately. As the update rate is lowered to show a longer period of time, we need to increase the number of samples used to create the composite image if we hope to create a reasonable display of the increased amount of activity that may occur over the longer period of time between updates. If we try to display too much activity, the frames will become unintelligible, showing only a large amount of blur. Because we are displaying the updated image as a single frame, we are limited by the amount of information that can be displayed within that frame. Thus, this visualization is best if used only to show only a relatively short period of time.

3.2 Storyboard

The next visualization we have developed is the storyboard. The storyboard visualization uses a technique similar to a movie storyboard to display images showing the history of the video stream.

Our demo application using the storyboard visualization can be seen in figure 3. The real time video stream is shown on the left, and the storyboard visualization is shown on the right. The storyboard displays the video history as a series of miniature images arranged in a grid. The storyboard is updated periodically with a new image being added each time. Initially the storyboard grid is blank, and images are filled in on each update starting with the upper left corner, and working across and down. When the storyboard becomes full the visualization starts updating again from the upper left corner, overwriting the previous image with the recent one. The result is a series of images showing the history of activities as they unraveled.

An advantage of using the storyboard visualization is that we can show more activity than with the motion blur visualization – the updated miniature frames show only a single snapshot clearly, and will not suffer from the same problem of becoming unintelligible. If a longer history is required, a lower update rate could be used (at the risk of missing quick events), or a larger storyboard could be used to display more images (at the expense of more screen real estate). Also, because the miniature frames only show snapshots from the video stream in order, detail regarding when activities happen is clearer. With the motion blur visualization it can be difficult to tell the order of the activities taking place in the composite frame, particularly with a long time between updates. However, with the miniature frames in the storyboard we can tell the order of events by scanning along the grid.

One of the difficulties that stems from the use of the storyboard visualization is that it may be difficult to pick out detailed information from the miniature images. If we are looking for a small change in the image, such as someone walking past a doorway for example, it may be too difficult to discern the change in the miniature images. Another problem with the storyboard visualization is that if we increase the size of the storyboard, we increase the number of frames that an observer must scan through to find some activity. Thus we may be again limited in the length of history that can reasonably be shown due to the difficulty in comprehending a large storyboard.

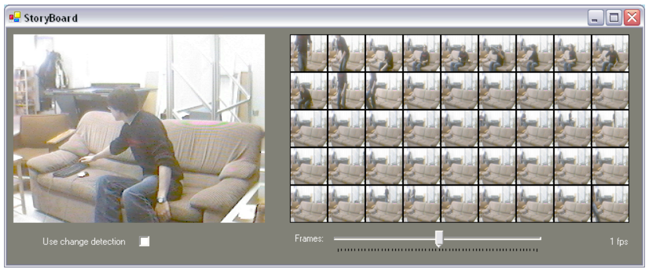

Figure 3: The storyboard visualization, miniature images in the storyboard show a sampling of the video over time.

3.3 Video Cubism

The next visualization we have developed is called video cubism. The reason for the name is that it is developed using the metaphor that the two-dimensional video frames could be stacked together to make a three-dimensional cube. This is similar to the metaphors used in the video streamer [2], and in stylized video cubes [6].

The idea behind the video cubism visualization is that if we were to slice the video cube and observe along the cut side, we would be able to see the change occurring along a boundary in the video stream. The result would give us a sense of the dynamics of the space along that boundary. Thus, the video cubism visualization creates a composite image by taking a specified column from each frame from the video stream, building the visualization in a manner which is similar to those used in Artifacts of the Presence Era [7], and the Last Clock [1].

Our demo application showing the video cubism visualization is shown in figure 4. The real-time video stream is being shown on the left, and the video cubism visualization is shown on the right. The visualization is built by taking a single pixel column from each frame received from the video stream and adding it to the composite image. The image is built with the most recent column added on the right. When the composite image is updated with a new column, the current image is shifted to the left to accommodate. Initially, the visualization is blank, and as frames are added the composite image streams out to the left. When the image has reached its maximum size, the oldest column (on the right) is removed to make way for the incoming column.

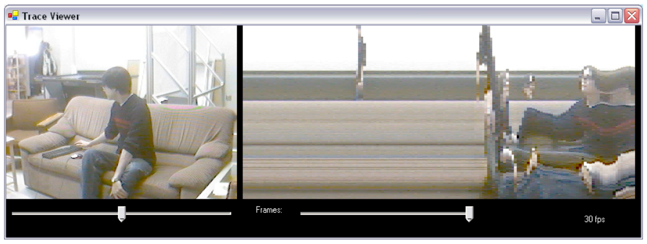

Figure 4: The video cubism visualization, the image on the right is a composite of a single column of pixels taken from each frame. Notice that with a frame rate of 30fps it is possible to see images of the subject as they move past the slicing column.

Additionally, the video cubism visualization allows the observer to select the column that the video stream is being sliced on. In our demo application, a slider is shown just under the real-time video stream. Adjusting this slider allows the user to choose which column is being used to make the composite image. When the slider is adjusted, the current visualization is re-sliced from the original stored video frames and updated. Any subsequent frames added are also sliced along the new column. In this way the visualization allows the observer to focus on a particular region of the video stream. For instance, the observer could place the slicing region along a doorway in order to easily see when someone comes in and out of a room. It is also notable that when using a high enough frame rate to populate the visualization, it is possible to see images of objects as they cross the boundary set by the slicing column. When an object is moved across the slicing column, the result is similar to scanning the object one column at a time. This can be seen in figure 4, as the subject moves across the boundary, a recognizable image of their face is visible within the visualization. Thus it may be possible not only to discern when some activity takes place in the video stream, but also who may be moving around.

The video cubism visualization also allows observers to navigate the video stream using the composite image. By simply clicking and scrubbing over the visualization, the image corresponding to the frame under the cursor is retrieved and can be displayed. In our demo application if the user clicks and moves within the visualization on the right, the retrieved image is displayed on the left rather than the current real-time stream. In this way the observer can easily determine when periods of interesting activity may have occurred from the visualization, and see them in full detail by scrubbing over the visualization in that area.

The video cubism visualization is very powerful in that it provides observers with an easy way to find and replay the events and activities occurring within the video stream. However, the downside to providing this power is that it will raise privacy issues. By providing a persistent history of the video stream, and a way to easily navigate it, we risk violating the privacy of those being captured by the video stream. A user providing a web cam stream from an office might expect the images sent to others to be ephemeral. If some event accidentally happened on camera that they did not want others to see they might move the camera and hope no one was watching. However, the video cubism visualization makes the video history persistent, and any observers connected to the stream can easily go back and replay

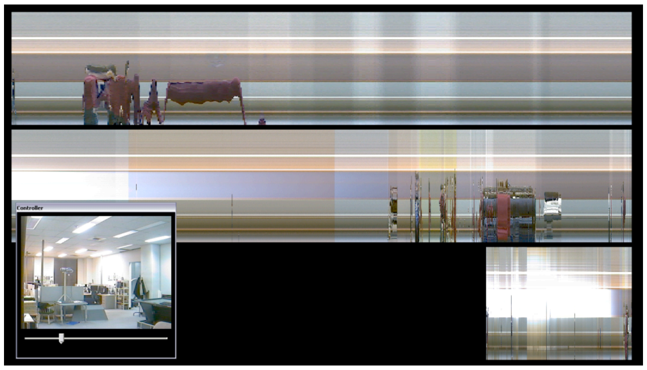

Figure 5: Example of the time line visualization after a demonstration. The minute line (top) shows the presenter entering the space to shut down the system. The hour line (mid) shows when the audience arrived for the demonstration.

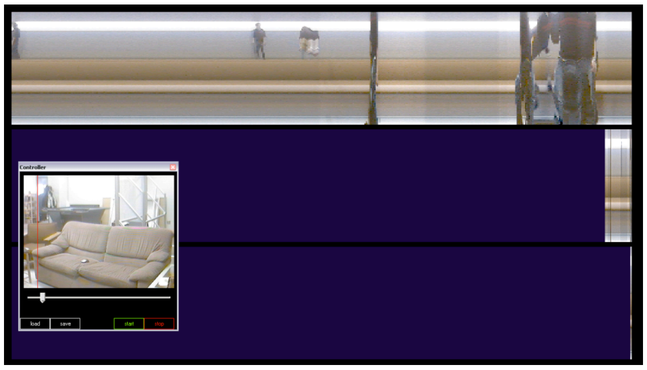

Figure 6: Recent version of the time line visualization adds the capability to load and save the captured contents to video files.

the events. In fact the change in dynamics that would occur if the camera were moved would in fact draw attention to the moment just before the user was able to move it in an attempt to protect their own privacy.

3.4 Time Line

The final visualization we have developed and will discuss is called the time line. The time line visualization is an extension of the video cubism visualization that is designed to display and allow the navigation of a large amount of video data, representing days or weeks of footage. The approach taken in the time line visualization is similar to that of the Last Clock [1] in that multiple lines are used to show the last minute, hour, day, or week of video. In this way the time line can behave like a timepiece, but rather than being simply a visualization for the passage of time the time line can be used as a means to navigate the video stream.

Our demo application showing the time line visualization is shown in figures 5 and 6. Because the time line is used to represent a large amount of video data, the demo application has been designed to run in a full screen window, suitable for a public display. In the demo, the real-time video stream is being displayed in the floating window. Underneath, three visualization lines are used to display specified time periods; the top line represents the last minute of the video stream, the middle line represents the last hour, and the bottom line represents the last day of video.

Depending on the available screen real estate, a fourth line (not shown) will become visible representing the last week of video below the day line. Individually, the lines work in a manner that is similar to the video cubism visualization; the lines are periodically updated by adding a single pixel column from the most recent frame. Also, a slider in the video window is available, and allows the observer to adjust the slicing column to focus on a specific area of the image. Much like the video cubism visualization, when the slicing column is changed the entire visualization is re-sliced and updated to show the new slicing position.

Unlike the video cubism visualization, the individual time lines are not simply updated at some arbitrary rate (for example, the frame rate of the camera being used), but rather they are updated with new data only frequently enough to fill the line with the appropriate length of data. These lines allow the user to navigate the stream captured by the visualization; in the demo application clicking and scrubbing over a time line will retrieve the image under the cursor and display it in the video window. From this we see that the most frequently updated minute line will provide the finest grain of video replay; events captured in the minute line will be accurate the most accurate. Conversely, the infrequently updated day or week lines will provide only course grain video replay; the video stream represented in these lines is likely to miss quick events occurring in the long period of time between updates. The example in figure 5 shows the time line after running for several hours as a public demonstration. In the minute line (top) we see the movement of the presenter as they prepare to shut down the display. In the hour line (middle) we see when the audience arrived and the demonstration was given. In the day line we see that the stream had been running for approximately five hours, and as the slicing column was set near a window we see a decrease in light as the sun sets in the evening.

Additionally, our time line demo application has the capability to save its contents to a file, and reload them for future observation. The time line visualization is powerful in that it can represent a large amount of video data, and provides a means to navigate the stream easily. A down side to its use is the screen real estate involved in showing multiple time lines rather than just the single line used in the video cubism visualization. Because it is making the history of the video stream persistent, we again raise privacy issues similar to those seen with video cubism. Thus this visualization is best suited as a display piece in an area that is largely accepted as public, rather than in a personal office or workspace. Another potential application of the visualization would be for security, where large amounts of video surveillance data might need to be searched to find activities and events.

3.5 Change Detection

The next idea we discuss is not a visualization in itself, but can be used in the design of other visualizations to improve the quality of what they display. Motivation for incorporating change detection within a video trace visualization can be found in some of the examples seen thus far wherein the visualization is being infrequently updated with sampled frames. When a long period of time elapses between when samples are taken, we run the risk of missing important events. Events such as a person entering a room can be quick and can easily go unrepresented in the visualization.

Our solution to this problem is to use a form of change detection in order to make smarter choices when picking frames to update the visualization. Our method for change detection works by selecting the last frame included in an update as the base frame for comparison. We compare every frame received during the period between updates with this base frame. Comparison between the base frame and the received frame is done by subtracting the received frame from the base frame, and taking the absolute value of the average pixel intensity in the resulting frame. The resulting value gives us a way to quantify the amount of change between the two frames. We can select a frame that is "most different" in the update period by keeping track of the frame with the highest change value. We can reduce the risk of missing events by updating the visualization with the frame selected by the change detection algorithm, rather than just using the frame sampled on the update period. When the update has been made we reset the

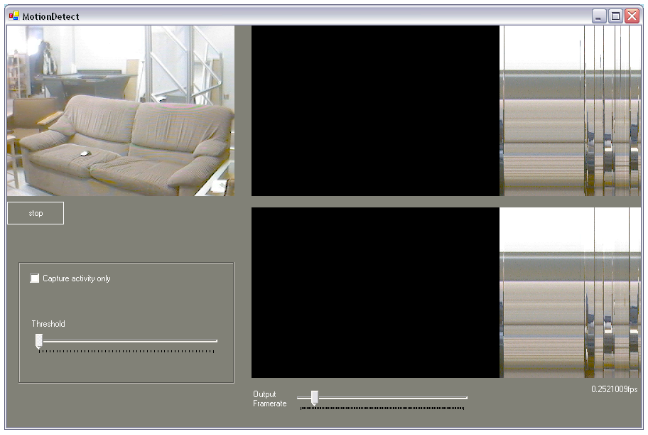

Figure 7: Change detection algorithm is used with the video cubism visualization on a regular sampling interval. The visualization using change detection shown on top picks up events that are missed by regular sampling shown on the bottom, as evidenced by the appearance of more lines within the visualization.

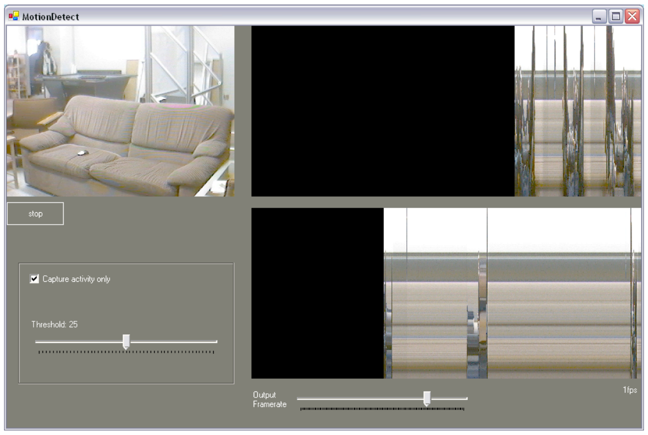

Figure 8: Change detection algorithm used with the video cubism visualization with a threshold for update. Notice that there are three periods of activity separated by lengthy spaces shown in the regular sampled visualization (bottom). The top visualization using the change detection threshold displays these moments in greater detail, and minimizes the periods of inactivity.

base frame to be the newest frame we have selected to include in the update.

With this change detection algorithm, we have developed two schemes for updating the display. These are demonstrated when used with the video cubism visualization in figures 7 and 8. The first method is to update the visualization with the most different frame within a regular sampling interval as shown in figure 7. In this figure, two versions of the visualization are shown; the lower visualization is only updated by sampling on a regular interval, the top visualization is updated using change detection to choose the most different frame within the update interval. In the figure, an update rate of approximately one frame every four seconds is used. We can see from the figure that the visualization using change detection is capturing quick events that are not added to the visualization that is not using change detection. We can see this from the presence of more lines within the visualization, indicating that some activity has been captured within that frame. The advantage of using this method is that we more capable of capturing the events occurring in the video stream, and we are also able to keep the temporal relationship between events.

The second method using change detection to update the visualization is to set a threshold value, and to update only when a frame has been received that exceeds this threshold change value. An example of this using the video cubism visualization is shown in figure 8. In this figure, the lower visualization shows the result of populating the display with frames sampled on a regular interval. The top visualization shows the result of populating the display with frames that exceed the threshold change value. The lower visualization shows three periods of activity with a large amount of time elapsing between them. The top visualization using change detection also captures three periods of activity, but captures each of them in greater detail, and condenses the periods in between in which no activity occurs. The advantage afforded by this approach is that we do not have to search through long periods of inactivity to find frames of interest where activity is occurring. The drawback of this is that we lose the temporal relationship between events; we can no longer tell when an event occurred using the visualization, we only know the relative order of events captured.

The power of the change detection algorithm is that we can use it to drive the various visualizations. We have already seen examples of its use to drive the video cubism visualization. Using change detection to update the display with the most different frame in a given interval would be useful in the time line visualizations where we have the infrequently updated lines displaying the last hour or day of video. This would allow a greater likelihood of capturing notable events within these lines. However, the use of change detection would not make a noticeable impact on a frequently updated visualization such as the minute line, as samples are being taken often enough that activity is unlikely to be missed.

Similarly, we have used the change detection algorithm to drive the motion blur visualization shown in figures 1 and 2. By dividing the update interval into five periods of time, we can use change detection to choose the most different frames in each of these periods to form the composite image. Also, the storyboard visualization shown in figure 9 has the option to make use of change detection. The implementation shown uses change detection to choose the most different frame between updates as the next image to add to the storyboard of miniatures. With the storyboard visualization it would also be sensible to use change detection based on a threshold rather than a sampling interval. This would allow us to populate the storyboard with images showing change, similar to the way a storyboard might be used in movie making.

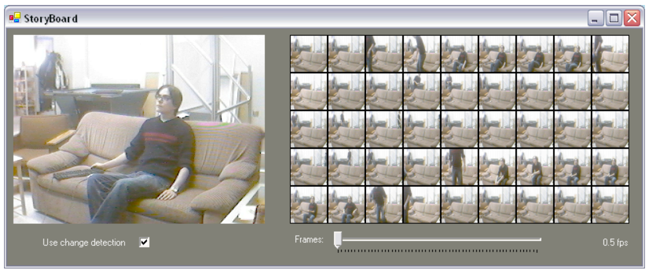

Figure 9: Storyboard visualization using change detection on a regular sampling interval. The miniature images added to the board are those that show the most change between update periods.

3.6 Inclusion In Other Applications

One of the goals in developing the example visualizations was to allow their inclusion within other applications relatively easily. This gives us the ability to explore the design space for video traces with the opportunity to rapidly test how the visualizations may be used within real applications, and what issues might arise from their use.

In order to achieve this, the visualizations have been developed in C# as .NET user controls. Including a visualization is simply a matter of dragging the desired control onto a form, and making the appropriate connections so that updates will happen. In the simplest case, the video cubism visualization can be updated by simply calling the addImage method every time a new frame is available to be added to the visualization. The addImage method accepts frames as a Collabrary Photo object, making it easy to use in conjunction with a Collabrary Camera object that may be capturing the video stream. In more complex cases, a few parameters may need to be set in order to start the visualization updating correctly. These might include the incoming frame rate, the desired update rate, which column to slice on when using the video cubism or time line, and choice of using change detection or not.

In order to allow navigation of the video stream, the video cubism and time line visualization controls fire image retrieved events as the user scrubs over the visualization. To allow navigation, the application must attach an event handler that displays the retrieved image when the event is called.

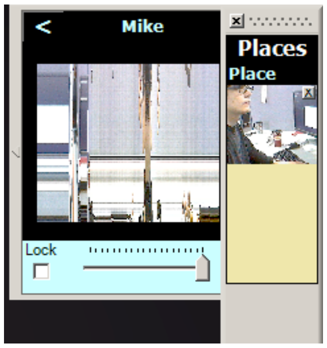

Figure 10: Community Bar media item developed using the video cubism visualization.

As an example, a Community Bar media item was developed using the same video cubism visualization as used in the demo application. This example is shown in figure 10. This item allows a user to share a video stream over the Community Bar, which is augmented with the video cubism visualization to allow observers to navigate the history of the video stream. In the figure, the current video stream is displayed with in the tile. Opening the transient reveals the visualization, and scrubbing over this visualization will display the corresponding frames in the tile.

One of the issues encountered with the decision to implement each visualization as a user control is that the underlying data is stored within the control itself to reduce the complexity in incorporating the visualization. This is not an issue in the demo applications as they are meant to only show an example of the visualization use, however in real applications the visualization may disappear and reappear when needed, or we may wish to switch to an expanded view of the visualization. Since the stored video history is contained within the visualization itself, it is difficult to keep the data persistent while the visualization is not. Similarly, it is difficult to populate an expanded visualization with data that has been captured in a smaller version.

4. CONCLUSION

4.1 Summary

In this paper we have taken a look at some of the design space for visualizations of history in a video stream by presenting and examining several examples of such visualizations. We have seen visualizations that are good at displaying a recent history of the video stream in the motion blur and storyboard examples. These may be useful in a system to support casual awareness, where users may only view the video stream periodically and may miss events that occur in between these periodic glances. These visualizations, however, are limited in the amount of data that can comprehensibly be shown at once. We have also looked at other visualizations that are designed to allow navigation and exploration of a large amount of video data – the video cubism and time line examples. These visualizations are powerful in that they allow the observer to quickly pinpoint and replay sections of activity, however, with this power comes the potential for privacy invasion.

It is important to keep in mind privacy concerns when dealing with visualizations that allow observers to easily browse video history. By keeping a history of the video stream, we break the expectation that our actions are short-lived, and will not be reviewed. Because of this, we must ensure that we do not reveal more information than the subjects within the media space are willing to allow.

Additionally, we took a look at how we can augment the design of these visualizations by using a change detection algorithm to drive visualization updates. The change detection algorithm allows us to reduce the risk that important events will be lost when infrequent updating occurs, and can provide us with a way to show only the activities that have occurred within the history of the stream, and eliminate the need to sift through long periods where nothing happens.

4.2 Future Work

Avenues for future work include increasing the power of the time line visualization for searching through large video streams. This could be done by allowing the observer to delve down into the video stream beginning with the course-grained week or day time lines. Selecting a frame in these time lines could trigger the finer grained hour and minute timelines to be populated with data surrounding the time represented by the selected frame. This would provide the user with a powerful time-based method for searching a long video stream to find detailed records of the events that have occurred.

Another possibility for future work may involve finding other methods for driving visualizations using change detection. The two methods described allow us to either preserve temporal relationships, or to maximize the visibility of events. It would be of interest to see if another scheme could be devised that loosely preserves time, but favors the inclusion of frames during periods of high activity. This may be difficult to achieve when populating a visualization from a real-time stream as optimal frame choice would require some knowledge of future events.

Lastly, it would also be of interest to examine ways to store and share these video histories. The trouble with the current visualizations is that the underlying history data is contained within the visualization itself. Users connecting to a media space would be able to receive new frames, but would have no way of acquiring the history in progress without some access to a stored video history. The beginnings of this have been shown in the time line application, which has the capability to save its accumulated history to files, and to re-load from those files, but the challenge is in finding a way to distribute and access long histories of video data.

5. REFERENCES

[1] Angesleva, J. and Cooper, R. 2005. Last Clock. IEEE Comput. Graph. Appl. 25, 1 (Jan. 2005), 20-23.

[2] Elliott, E. and Davenport, G. 1994. Video streamer. In Conference Companion on Human Factors in Computing Systems (Boston, Massachusetts, United States, April 24 - 28, 1994). C. Plaisant, Ed. CHI '94. ACM Press, New York, NY, 65-68.

[3] Gutwin, Carl. 2002. Traces: Visualizing the Immediate Past to Support Group Interaction. In Proceedings of Graphics Interface 2002.

[4] Hill, W.C. & Hollan, J.D. Edit wear and read wear. Proc. ACM CHI'92, 1992, ACM Press, 3-9.

[5] Hudson, S. E. and Smith, I. 1996. Techniques for addressing fundamental privacy and disruption tradeoffs in awareness support systems. In Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work (Boston, Massachusetts, United States, November 16 - 20, 1996). M. S. Ackerman, Ed. CSCW '96. ACM Press, New York, NY, 248-257.

[6] Klein, A. W., Sloan, P. J., Finkelstein, A., and Cohen, M. F. 2002. Stylized video cubes. In Proceedings of the 2002 ACM Siggraph/Eurographics Symposium on Computer Animation (San Antonio, Texas, July 21 - 22, 2002). SCA '02. ACM Press, New York, NY, 15-22.

[7] Viégas, F. B., Perry, E., Howe, E., and Donath, J. 2004. Artifacts of the Presence Era: Using Information Visualization to Create an Evocative Souvenir. In Proceedings of the IEEE Symposium on information Visualization (infovis'04) - Volume 00 (October 10 - 12, 2004). INFOVIS. IEEE Computer Society, Washington, DC, 105-111.