MiniDiver: A Novel Mobile Media Playback Interface for Rich Video Content on an iPhone™

Gregor Miller[1], Sidney Fels[1,2], Matthias Finke[2], Will Motz[1], Walker Eagleston[1], and Chris Eagleston[1]

[1] Dept. of Electrical & Computer Engineering, University of British Columbia, Vancouver, BC, Canada
[2] Media and Graphics Interdisciplinary Centre, University of British Columbia, Vancouver, BC, Canada

Abstract. We describe our new mobile media content browser called a *MiniDiver*. MiniDiving considers media browsing as a personal experience that is viewed, personalized, saved, shared and annotated. When placed on a mobile platform, such as the iPhone™, consideration of the particular interface elements lead to new ways to experience media content. The MiniDiver interface elements currently supports multi-camera selection, video hyperlinks, history mechanisms and semantic and episodic video search. We compare performance of the MiniDiver on different media streams to illustrate its feasibility.

1 Introduction

Currently, on a mobile platform, video is normally watched with a video browser that still uses a tape playing metaphor. That is, it has transport controls for *play*, *rewind*, *forward*, *skip forward* and *backward*. However, when video contains metadata, such as hyperlinks, semantic content (i.e., keywords) and multiple camera angles, the interface does not allow a simple way to navigate through this complex video space. Thus, for example, when browsing YouTube™ it is easy to get lost in a multitude of video sources. As well, there is no obvious way to record history or share partial content or provide annotation. Likewise, mechanisms to hyperlink content or do episodic or semantic based searches are limited. We have been exploring new interaction paradigms for video browsing, navigation and annotation as part of our *MyView* research to provide personalized video experiences using a novel video browsing MyView client called a *Diver*.

The concept behind the MyView research program is that in the future, tracking technology and pervasive video/audio capture technology will be used to automatically tag video content making for rich, complex video data space that can be viewed differently by each person. In this paradigm, the notion of the video clip becomes less clear since it can be at the granularity of a single person in a single instant. For example, at an indoor Olympic event, such as ice hockey, we can create personal views of the game while it is being played as well as

S. Natkin and J. Dupire (Eds.): ICEC 5709, pp. 98–109, 2009.

Fig. 1. Interface of Video Player with Heads Up Display (HUD) and Toolbar

after the game that can be delivered over wireless protocols to cell phones or other devices. Audio and video data may be captured by multiple sensors and streamed to a multimedia server continuously. Hockey players may be tracked using video and/or other mechanisms to detect when they are in view of each video camera providing tracking and view orientation meta-data. Likewise, additional semantic data, such as keywords, may be provided by broadcasters and other viewers. With this meta-data combined with the multiple video sources, a valuable, personalized, fun memory of an event can be viewed and shared. The Diver provides mechanisms to view this rich video space, though, is currently intended to be embedded in web-browsers and WIMP interfaces.

As part of MyView, we also address the emerging trend that video entertainment on a mobile platform is quickly becoming typical. Thus we also have constructed a version of the Diver for a mobile environment; specifically, an iPhone^{TM}. We call this a *MiniDiver* and it takes advantage of the specifics of limited screen real-estate and particular input mechanisms found on mobile phones. Figure 1 shows an example of the iPhone^{TM} interface for the MiniDiver. In this paper, we focus on four main issues that arise when rich video content is viewed on the cell phone. Specifically, we have investigated interface mechanisms in the MiniDiver to allows users to: 1) select multiple camera views, 2) navigate hyperlinked video content, 3) save and retrieve complex MiniDiving history, and 4) use both episodic and semantic mechanisms to access video meta-data.

Each of these four items are emerging as key components of the video experience. For example, at a sporting event, such as ice hockey, there may be multiple cameras focused on the action synchronized in time. Likewise, each player's location may be tracked and appropriate hyperlinks to new video or other web content be integrated with the video experience. As users Dive through the hockey game, their history becomes complex as they follow hyperlinks and change camera angles. In this situation, going "back" isn't obvious for what should happen. Finally, the meta-data associated with the data allows for automatic assembly of personalized video based on combinations of episodic (i.e, image/video based)

or semantic (i.e. keywords) mechanisms. These are integrated into the MiniDiver as we discuss below.

In Section 2 we cover some of the related work that has investigated rich video content mechanism and means to access it by users. Section 3 provides a description of the main types of interface paradigms we have created for the video experience. All our examples use ice hockey as our event that we are MiniDiving. We have used multiple cameras to capture footage of an ice hockey match and have created meta-data tags associated with the players from all camera angles for our test data. We cover some of the performance issues in Section 4 and conclude with a discussion of the emerging complexity of navigating rich video data spaces and the interaction design approaches needed to deal with them.

2 Related Work

For many years traditional TV stations utilize multi-camera views to enhance the user's experience especially when broadcasting sport events. Relaying on a broadcast channel viewers are dependent on directors who select camera angles to provide a "best view" for an entire audience. Taking the "best view" experience to the next level implies viewers can choose for themselves the "best view" based on their personal preference. This requires new interface concepts for video players that enable viewers to switch between available camera views. Lou et al. [1] developed a video player for their multi-view video system which includes a slider within the interface to allow viewers to switch seamlessly among views of the current video presentation. Navigational aids within the video space are not supported, which makes it especially difficult for longer video sequences including multiple views.

In computer vision research multi-camera views are widely used, e.g. for object tracking [2] or view synthesis [3]. Sport video analysis especially takes advantage of object tracking algorithms that employ multiple camera sources in order to extract context sensitive metadata e.g. player location [4]. Such location data can be used to create object-based video annotations [5], also called hypervideo links [6]. Similar to a hyperlink in a web page viewers can interact with a hypervideo link to access additional information. Hypervideo links follow the associated video objects and hence contain spatial and temporal information. Usually, hypervideo links have a visual representation on top of the video object to announce their existence to the viewer [7]. Until now research has primarily focussed purely on single video sources that include hypervideo links.

According to Cockburn et al. [8] revisiting information from the past is a common activity of users. In the web domain, standard browsers allow users to set bookmarks, use back and forward buttons or history lists in order to access previously visited web pages. Tools such as WebView [9] or Global Tree Browser [10] are examples that provide special functionality for revisiting Web pages. For local file access GoogleDesktop [11] and TimeScape [12] keep a history record allowing users to easily revisit their content. Based on the richness of context-sensitive video including multi-camera views the demand for a history

tool enabling easy and fast access to previously selected video sequences has to be considered. Though, with time-based media, it requires more sophistication than for static webpages.

3 MiniDiver: Exploring Video Content on the Mobile Platform

The MiniDiver is designed to provide a user interface that is intuitive and responsive, yet powerful enough to take advantage of a context-aware video space on a mobile platform. The interface is based upon the concepts developed for the desktop MyView client, but with a focus on the iPhone™ which is a touch-enabled mobile device. There is also a personalized viewing mode where the user can browse video according to their unique preferences and interests. The MiniDiver requires options to display relevant metadata such as player positions, and names by overlaying them on top of the video content as in our ice hockey example. The MiniDiver supports the ability of users to control video with the usual transport control mechanisms (play, stop, backward and forward) but also provide mechanisms to change viewpoint, save and retrieve MiniDiving history and search/navigate using both episodic and semantic queries.

In this paper, we have prototyped our MiniDiver so that it uses local content residing on the iPhone™ or data over a network from a MyView server. The MyView server has a Director component that contains rules to feed video content to the MiniDiver appropriately, however, this is outside the scope of this paper. For communication with a simple networked client, the MiniDiver uses the Hive [13] communications protocol in order to stream content to the client device. Section 3.1 discusses the design of the MiniDiver that address the main four interaction mechanisms listed in Section 1.

3.1 MiniDiver: MyView Client Design

Our design concept of the mobile client is based on four user interfaces serving mobile context sensitive video. These interfaces are Content Browser, Video Player, Multi-Camera Browser and History Browser. In the following we will discuss and present each interface in more detail.

Content Browsing. Once the application loads the user will want to select some content to watch in the video player. We present the content selection interface using the usual iPhone™ table view hierarchy. In this commonly used interface style the user drills down into a series of lists that end in a detail view. In our navigation hierarchy the home screen allows the user to select a content source, and the next level allows them to select which events they would like to watch. A single event can be selected by tapping on it, and multiple events can be selected by tapping the navigation bars select button. This pops up an action toolbar at the bottom of the screen and transitions the list view into a selection mode. Events can then be selected by tapping on them and the action toolbar

Fig. 2. Modal bookmarks view activation animation

can be used to filter the result, or jump right into video playback. If the user has previously saved a bookmark they can access it from the startup screen by selecting the bookmarks button in the navigation bar. The bookmarks browser is presented as a modal view which slides up over the home screen. This was chosen to mimic the bookmarks functionality in MobileSafari (the iPhone^{TM}'s web browser). Tapping on a bookmark will take the user directly into the video player.

Once we enter the video player the system status bar is hidden to provide the user with a fully immersive experience. The interface is also re-oriented to landscape mode with the home button to the right of the screen. At this point we support only the landscape orientation for the Video Player and its siblings, the history browser and multi-camera browser.

Video Player. The video player screen, shown in Figure 1, contains much of the functionality of MiniDiver. From here users can navigate through time by grabbing the playhead or by using a two finger scrub gesture.

MyView content directly supports multiple video streams, so the user can also choose which camera angle they wish to view the action from. This is accomplished by a swipe gesture on the VideoPlayer screen. For example swiping left pushes the current video stream offscreen and brings in the stream of the camera physically located to right as shown in Figure 3. This gesture only allows for movement between streams in a one dimensional space, however we found that this was enough to spatially locate the streams in our test content. The one dimensional interface that we have developed is modelled after the photos application where users can swipe left and right to browse their photos. We have had to adjust this behavior slightly so that users can not slide a video stream

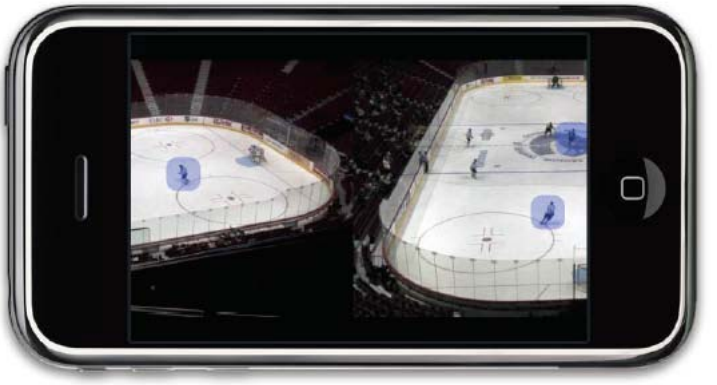

Fig. 3. Video stream switch animation. Interactive objects are highlighted in blue, and maintain consistency across views.

partially offscreen because we are limited by the mobile device processing power and the wireless network bandwidth to display two live video streams at once.

Notice that this approach only provides for relative camera view selection. That is, using the swipe interface you can move to cameras that are left or right (up or down if vertical swiping is added). Further, left and right camera views can be selected by the MyView Director service on the MyView server to provide meaningful interpretation as to viewing content more to the right or left. This can be done based on a quality-of-view analysis [14] or virtual viewpoints [15,16]. To have an absolute camera position selection requires meta-data tags that include a layout of the cameras in an absolute coordinate frame. We have not included this at this point since it is a variation on the Multi Camera Browser covered in Section 3.1, except we require mechanisms to show the video feeds from each of the absolute camera position. As well, like the relative camera position selection, the MyView Director will select a reasonably small set of either real or virtual cameras based on lowering cognitive load of the user, quality-of-view constraints and user preferences.

MyView content can be tagged with various information about objects that are onscreen and events that occurring at a given time. In our hockey example, content is tagged with player names and locations using hypervideo links and it also includes background masks. We use this information to allow users to select players in the video player view. This can be accomplished via a tap (single click), which triggers a player/object selection mechanism. Tapping the screen enters the selection mode which pauses the video and highlights all objects which are selectable. The user may then select an object by clicking on them; multiple objects may be selected in this manner. Video is started again by clicking on a non-selectable region of the screen.

The video player does have a simple translucent heads up display (HUD) and toolbar that can be activated by a single tap as shown in Figure 1. The HUD contains a play/pause button and a button to enter the multi-camera browser.

Fig. 4. Interface of Multi-Camera Browser

The toolbar contains a playhead and scrub bar, as well as a button to enter the history browser.

Multi-Camera Browser. Non-linear context sensitive video presentations based on a multi-camera scenarios require a visual disclosure of all available useful video streams to assist users to find their "best view". Hence, to allow users to easily navigate through available video streams we have created a multi-camera browser interface as shown in Figure 4.

We wanted the multi-camera browser to be a natural extension of the video player and we also anticipate that it will be used frequently. To best serve these requirements we made the multi-camera browser activate immediately when requested and only have a short activation animation. To achieve this we start loading visible video streams asynchronously right after kicking off the activation animation. Besides keeping initial load times down this also allows the browser to be fully responsive to touch input as soon as it is activated regardless of whether images are still being loaded from the network or disk.

Users access the multi-camera browser through a double tap on the HUD of the video player. When activated the application scales down the currently playing video to a grid view which also shows all of the other camera angles that are available. Video frames load asynchronously and fade in when ready, allowing the view to be interactive as soon as it is activated. The grid of camera angles can easily be scrolled with a vertical swipe gesture, and tapping on video frame will switch to that camera and transition back into the video player. If the user does not wish to select a camera there is a cancel button on the toolbar which will take them back to the video player.

History Browser. Context sensitive video content that includes multi-camera video stream requires new forms of navigation aid for its users. The History View allows users to navigate back to video sequences they had already selected in the past. To achieve this functionality, our mobile client keeps a record of every video scene users access combined with its start and end time of the video

playback. With this data we are able to create an interactive History View Grid that can be used for navigation purposes. The History View Grid is shown in Figure 5.

The History View has two view modes: time-based and node-based. In both modes, the History Tree Structure is displayed with each clip being a rounded rectangle. Each rectangle is colored to indicate which camera the clip has video from, and the first frame of the clips video is also displayed to assist with differentiating cameras. The default state of the History View interface is the time mode.

In the time mode, the width of the rectangles are determined by the length of each clip while in the node mode the clips all have equal width. In both the heights are all the same, just big enough for a finger to tap. To switch to a previous clip or sequence, the user taps on the location in the history tree where they want to go to. The tree will animate to its new structure and keep the current sequence up to date in the player. So when a user taps the middle of a clip, that clip is made active and its sequence is loaded into the player. The playhead is also moved to the point in time corresponding to the location of the tap in the clip.

There are two axes in the default time view, a sequence axis and a time axis. The y-axis is the sequence axis and indicates which sequence falls at that level of the history tree. The time axis displays seconds and is inline with the start times and lengths of the displayed clips. The time axis can be switched to a "depth" axis by tapping the toggle button (tapping the toggle button again switches back to the time mode). In the node mode, all clips are displayed as having equal lengths, and the x-axis indicates their depth in the tree. This mode is useful if there are a large number of cuts over a short period of time, making them difficult to see on a scale proportional to time. As a history tree becomes larger, some portions will move off screen but can be reached just by dragging the tree until the desired sections are visible. The axes follow this motion and adjust their values accordingly, so the user always knows where they are. The

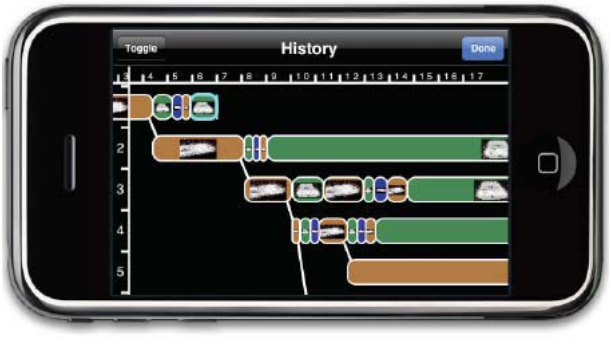

Fig. 5. History Browser Interface: horizontal axis represents the absolute time-scale of the synchronized videos; vertical axis represents branches showing user interaction

active clip is indicated by a blue border, and the playhead position by a blue vertical line on the active clip.

3.2 MiniDiver and MyView Communication Architecture

The MiniDiver uses the same underlying communication architecture as the MyView system, a flexible and modular framework called Hive[13,17]. All the components of MyView are based on Hive to facilitate simple communication and re-use of components on various platforms. Hive provides a communication protocol and an abstracted transport service for the transfer of data to different devices. The transport service can be implemented using various technologies (such as shared memory, ethernet or Bluetooth) to increase the number of devices that can communicate in the MyView framework. The connection paradigm is peer-to-peer to increase bandwidth but still allow centralized control.

The complicated processing of data (compression, background subtraction, player tracking) is performed centrally on a group of machines and the results are sent to the MiniDiver, using Hive as the transport medium. Each component is defined as a Hive *drone*, which is a module capable of performing a task and is controlled by an *application*. Applications have ultimate control over the drones in the system, and can construct pipeines by connecting drones together; the MiniDiver acts as a Hive application, controlling the drones and their connections. In future the MiniDiver will connect to a single point to receive data, to accommodate multiple MiniDiver clients.

3.3 Video Streaming

Getting video onto the MiniDiver is accomplished by streaming footage to the device using Hive for transportation. Multiple camera footage is stored centrally on a video server accessible via a drone operating as the interface between the video database and other Hive modules. Data transfers can operate in various modes, one of which is called the *request* mode. In this mode, data is only sent when requested by the destination drone; using this mode the images streamed to the device can change the camera footage currently being streamed by changing the request. This results in a lower latency than using a continuous streaming mode, and is scaleable to a larger number of camera sources.

Due to memory constraints on the mobile device, previous frames are not cached, so scrubbing along the time bar or navigating the history results in new requests for the previous data.

4 Results

This section presents the formats used for video streaming and the results of tests on latency and achievable frame rates on the mobile platforms of the iPhone and the iPod Touch.

The table in Figure 6 presents the quality of video used in the tests and the bandwidth required for each format. Due to the lack of support for developers

Content	Resolution	Format	Frame Rate	Data Rate
High Quality	480 × 270	JPEG Sequence	30Hz	630 KB/s
Low Quality	240 × 135	JPEG Sequence	30Hz	248 KB/s
Low Quality	240 × 135	JPEG Sequence	15Hz	124 KB/s
High Quality	480 × 270	H.264	30Hz	55 KB/s
Low Quality	480 × 270	H.264	15Hz	25 KB/s

Fig. 6. Comparison of data formats and rates - the iPhone™ supports H.264 which is much more efficient, but not directly accessible to the developer. Our application employs JPEG sequences for compressed video streaming.

Content	Location	Frame Rate	
		iPhone™ 3G	iPod Touch 2G
High Quality	Local	5 FPS	12 FPS
Low Quality	Local	17 FPS	30 FPS
High Quality	Network	×	8 FPS
Low Quality	Network	×	23 FPS

Fig. 7. Video playback performance using the JPEG sequence format (network testing performed only on the iPod Touch 2G)

to use the native video formats (H.264) the MiniDiver uses compressed JPEG image sequences for video. This also allows for frame-level requests from the video servers, which would be more complicated with H.264 because keyframes are widely spaced. H.264 formats are presented in the table to provide a comparison of data rates used.

Based on the tests we performed on the iPhone™ 3G and the iPod Touch 2G there are obvious performance differences, shown in Figure 7. The iPod Touch 2G has an upgraded ARM processor from previous revisions, which explains its increased performance. Generally, using high resolution JPEG sequences results in low frame rates, with the iPod having the only acceptable rate for local access to video.

Network-based tests were performed only on the iPod using the request mode outlined in Section 3.2. There was a substantial drop in performance using the network, although we believe this could be significantly improved with optimization of the Hive implementation on the device. The MiniDiver performs well with the lower quality content either from local content or over the network. The latency in requesting a different camera angle is also reduced with lower bandwidth footage.

5 Conclusion and Future Work

We have created a new context-sensitive video browsing environment for the iPhone™ called a MiniDiver. It provides mechanisms to select multiple camera views, navigate hyperlinked video, save and retrieve complex MiniDiving history, and some episodic and semantic search mechanisms. These mechanisms are necessary to allow people to experience the enormous amount of video content that is being generated. We have shown how multiple synchronous camera views can be accessed using a touch interface. We have included a mechanism to allow users to select moving objects in the video scene and get hyperlinked data associated with them. We have included a mechanism to support users traversing their history in complex, non-linear ways that re-interprets what it means to rewind and fast-forward in video. Our implementation of the episodic and semantic search mechanisms remains to be refined as we only provided semantic mechanisms and basic episodic mechanisms as found in the camera views in the MiniDiver. Our Hive implementation has provided performance measures to illustrate that delivering rich, interactive media data on a mobile device is feasible.

All of these mechanisms have yet to be thoroughly user tested as we have just defined the requirements needed in a video browsing environment to deal with the complex demands of large-scale, meta-tagged video. We continue to explore improved searching, enhancements for selecting one-to-many hyperlinked video data and developing means for the MyView server to supply only relevant data to the user based on their queries, context and profile.

This paper described our current prototype of a MiniDiver on the iPhone™. We are pursing design guidelines and prototypes for mobile devices for dealing with the anticipated changes in video content that are emerging due to the proliferation of video capture devices, video processing techniques and personalized video content. We believe a shift is needed for users to be able to effectively manage and share the wealth of video experiences the will have in the future. The MiniDiver provides some of the functionality that addresses this shift.

Acknowledgments. This research is funded by Bell Canada Inc. and the Natural Sciences and Engineering Research Council (NSERC), Canada. We thank all the members of the MyView team for their contributions.

References

1. Lou, J.G., Cai, H., Li, J.: A real-time interactive multi-view video system. In: MULTIMEDIA 2005: Proceedings of the 13th annual ACM international conference on Multimedia, pp. 161–170 (2005)
2. LeoAnnotation2008: Real-time multiview analysis of soccer matches for understanding interactions between ball and players. In: CIVR 2008: Proceedings of the 2008 international conference on Content-based image and video retrieval, pp. 525–534. ACM Press, New York (2008)
3. Grau, O., Hilton, A., Kilner, J., Miller, G., Sargeant, T., Starck, J.: A free-viewpoint video system for visualisation of sport scenes. SMPTE Motion Imaging, 213–219 (2007)

4. Lu, W.L., Okuma, K., Little, J.J.: Tracking and recognizing actions of multiple hockey players using the boosted particle filter. Image Vision Comput. 27(1-2), 189–205 (2009)
5. Goldman, D.B., Gonterman, C., Curless, B., Salesin, D., Seitz, S.M.: Video object annotation, navigation, and composition. In: UIST 2008: Proceedings of the 21st annual ACM symposium on User interface software and technology, pp. 3–12. ACM Press, New York (2008)
6. Shipman, F., Girgensohn, A., Wilcox, L.: Authoring, viewing, and generating hypervideo: An overview of hyper-hitchcock. ACM Trans. Multimedia Comput. Commun. Appl. 5(2), 1–19 (2008)
7. Stahl, E., Zahn, C., Finke, M.: How can we use hypervideo design projects to construct knowledge in university courses? In: CSCL 2005: Proceedings of th 2005 conference on Computer support for collaborative learning, International Society of the Learning Sciences, pp. 641–646 (2005)
8. Cockburn, A., Mckenzie, B.: What do web users do? an empirical analysis of web use. International Journal of Human-Computer Studies 54, 903–922 (2001)
9. Cockburn, A., Greenberg, S., McKenzie, B., Jasonsmith, M., Kaasten, S.: Webview, a graphical aid for revisiting web pages. In: OZCHI 1999: Proceedings of the 1999 Australian Conference on Human Computer Interaction (1999)
10. Killam, B.: A study of three browser history mechanisms for web navigation. In: IV 2001: Proceedings of the Fifth International Conference on Information Visualisation, Washington, DC, USA, p. 13. IEEE Computer Society, Los Alamitos (2001)
11. GoogleDesktop, http://desktop.google.com/
12. Rekimoto, J.: Time-machine computing: a time-centric approach for the information environment. In: UIST 1999: Proceedings of the 12th annual ACM symposium on User interface software and technology, pp. 45–54. ACM, New York (1999)
13. Afrah, A., Miller, G., Parks, D., Finke, M., Fels, S.: Hive: A distributed system for vision processing. In: Proc. of the Int. Conf. on Distributed Smart Cameras, September 2008, pp. 1–9 (2008)
14. Shen, C., Zhang, C., Fels, S.S.: A multi-camera surveillance system that estimates quality-of-view measurement. In: Proceedings of The IEEE International Conference on Image Processing (ICIP 2007), pp. III–193–III–196 (2007)
15. Miller, G., Starck, J., Hilton, A.: Projective surface refinement for free-viewpoint video. In: Proc. 3rd European Conference on Visual Media Production, IET, November 2006, pp. 153–162 (2006)
16. Zitnick, C., Kang, S., Uyttendaele, M., Winder, S., Szeliski, R.: High-quality video view interpolation using a layered representation. In: SIGGRAPH, pp. 600–608 (2004)
17. Miller, G., Afrah, A., Fels, S.: Rapid vision application development using hive. In: Proc. International Conference on Computer Vision Theory and Applications (February 2009)