Video Lens: Rapid Playback and Exploration of Large Video Collections and Associated Metadata

Justin Matejka, Tovi Grossman, and George Fitzmaurice
Autodesk Research, Toronto, Canada
{*firstname.lastname*}@Autodesk.com

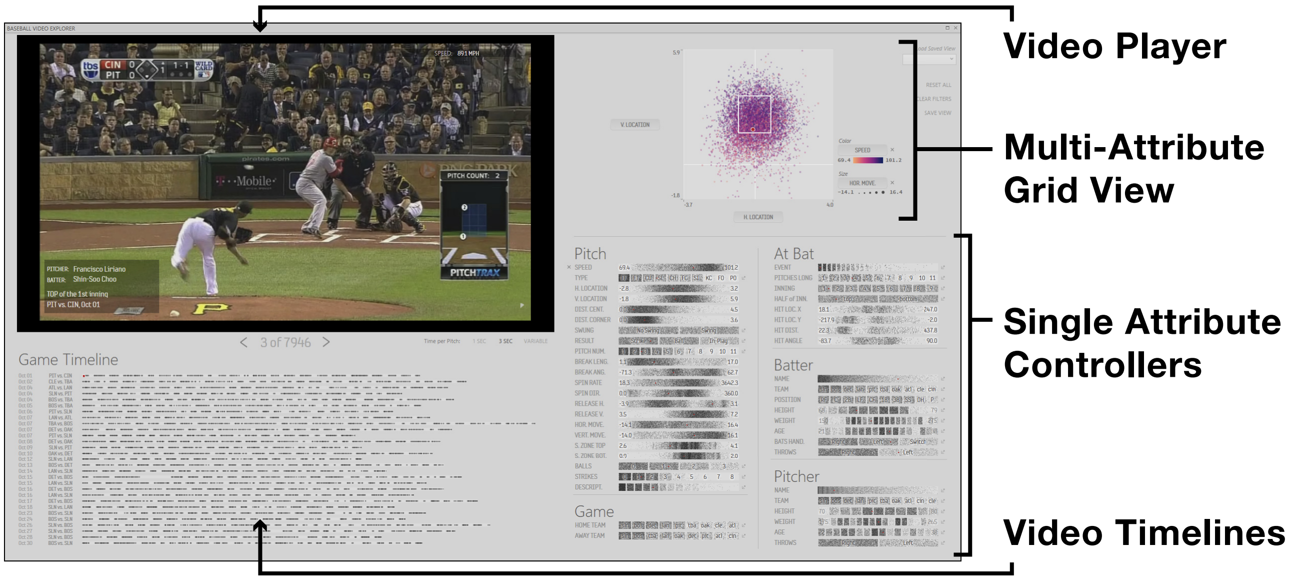

Figure 1. Overview of the Video Lens interface consisting of a video player in the top left surrounded by the Multi-Attribute Grid, Single Attribute Controllers, and Video Timelines which support the faceted search and selection of video events based on a multi-dimensional set of timeline-based metadata.

ABSTRACT
We present *Video Lens*, a framework which allows users to visualize and interactively explore large collections of videos and associated metadata. The primary goal of the framework is to let users quickly find relevant sections within the videos and play them back in rapid succession. The individual UI elements are linked and highly interactive, supporting a faceted search paradigm and encouraging exploration of the data set. We demonstrate the capabilities and specific scenarios of *Video Lens* within the domain of professional baseball videos. A user study with 12 participants indicates that *Video Lens* efficiently supports a diverse range of powerful yet desirable video query tasks, while a series of interviews with professionals in the field demonstrates the framework's benefits and future potential.

INTRODUCTION
In the last 10 years, the trend of storing large collections of online videos has exploded. YouTube and Netflix alone account for a significant percentage of all internet traffic[1], while domain specific systems such as sports video streaming [27, 37] and software video capture [7] are active commercial and research areas.

As a result, there has been a recent surge in interest looking at ways to allow users to efficiently locate and navigate to scenes of interest within videos [9, 15, 16]. However, finding a specific scene or event within a *large collection* of videos remains an open challenge. Furthermore, if multiple results match a query of interest, there are few known techniques that would allow the user to efficiently play back all of the results. Status quo solutions, such as playlists, work when interested in a collection of entire videos, but do not suffice if only a small part of each video is of interest.

For example, consider a baseball fan who wishes to watch all home runs hit by their favorite player during the 2013 baseball season. Even if the user manages to create a playlist of all videos (games) where the events of interest (home runs) occurred, it would still be laborious to find the event of interest within each of these videos.

Learning from data exploration techniques, a faceted search paradigm [33] could be of use. Faceted search is commonly employed by users working with collections with many dimensions of metadata, allowing them to filter the data set into small subsets that are of direct interest [8]. Indeed, many

[1] http://www.cnet.com/news/netflix-youtube-gobble-up-half-of-internet-traffic/

forms of video collections do have associated timeline metadata, such as the plays that occur during a sporting event [6], the commands that occur during a software tutorial [7], or the faces that appear in a TV show [38]. However, the possibility of using such meta-data to support the faceted search of large video libraries has been under-explored.

In this paper, we present *Video Lens*, a framework which allows users to visualize and interactively explore large collections of videos and associated meta-data. The interface of *Video Lens* is comprised of four main components: Video timelines, which show each individual video in the library with its associated event locations; single-attribute controllers, which provide support for faceted search; a multi-attribute grid, which visualizes the current data set across a user-specified set of attributes; and a video player, which supports rapid playback of query results (Figure 1).

We describe the design goals of the *Video Lens* system and its framework in detail, and discuss some of the specific challenges that arise when aligning large collections of video with metadata. We then demonstrate the capabilities and specific usage scenarios of *Video Lens* within the domain of professional baseball videos, known to be a rich source of data and analysis [22]. A user study with 12 participants indicates that *Video Lens* efficiently supports a diverse range of powerful yet desirable video query tasks, while a series of interviews with professionals in the field demonstrates the framework's benefits and future potential.

RELATED WORK

Finding/Watching Sections of Video

With the increase in the quantity and length of digital videos available a number of projects [9, 15, 16, 23] have explored ways to improve the process of manually finding a particular section of a video file. However, these techniques do not consider searching through a library of videos, and they do not use any metadata about the video content to assist in the search task. Chronicle [7] uses metadata about the events in a software tutorial video to facilitate searching and jumping to relevant sections. Pause-and-Play [21] similarly uses metadata to link the playback of a tutorial video to the progress of a workflow. However, both of these operate on a single video at a time.

The Ambient Help [14] system looks at a collection of software tutorial videos along with metadata describing which commands occur at which times in each video, and automatically plays back video segments which might be relevant to the user. This is similar to *Video Lens* in that it uses a collection of videos with timeline metadata, but the metadata in Ambient Help only has only one dimension (command name), and the user does not have any control over which video clips are presented.

Some work has looked at generating metadata for video files by automatically analyzing audio and video streams [1, 5, 14, 17, 25, 26], as well as through crowdsourcing [10]. The *Video Lens* system assumes that videos already have a set of rich timeline metadata, but a system to automatically extract time stamped metadata would be a useful companion.

Faceted Search

The idea of using faceted search techniques to filter though a multi-dimensional set of data has been well explored [8, 11, 28, 30, 33] and applied to finding items in many domains including research publications [13], webpages [4], e-commerce [29], image search [32, 34], and to a lesser extent, video collections [5, 31].

The interfaces for faceted search systems typically consist of a list of attributes, or facets, and a set of valid options or ranges for each. Similar to *Video Lens*, the Selfiecity Project [36] displays the distribution of values for each attribute within the individual facet control. The main differences being that Selfiecity uses area charts to represent the distribution while *Video Lens* uses an array of individual dots, and the Selficity controls are manually crafted to represent the underlying data, while the *Video Lens* attribute controls are meant to be generic and automatically adapt to the structure of the underlying data.

Sports Video Systems

The topics of analyzing, searching, and viewing sports videos have received significant attention recently, both in the literature and as deployed systems.

Olsen *et al.*'s Time Warp Sports [18] presented a customized interface for watching videos of sporting events, making use of event markers to support operations such as jumping to the next or previous play. The *Video Lens* system can support a similar workflow by selecting all of the events in a single video, and then moving to the next and previous events in the resulting set.

Major League Baseball (MLB) has an online video presence in the form of MLB.tv [27] which gives viewers access to an entire season's worth of baseball games and highlights. The video player has some advanced browsing features such as jumping to a specific inning by clicking on the corresponding cell in the box score. The Baseball Savant website [39] has a comprehensive tool to query the same advanced PITCHf/x event metadata [6] we use in the *Baseball Video Lens* example, however, the search interface is more traditional in the sense that it does not allow for iterative refinement of a query, and video clips are available for only a small set of "highlight" plays, such as home runs. Professional (and some amateur) baseball teams often make use of proprietary software for video analysis. Our interviews with professional teams indicated that the BATS package from Sydex Software [40] is the most frequently used package, however, it does use a faceted search paradigm.

The National Basketball Association (NBA) recently released a compelling Stats Video feature on their website [37]. By clicking on a stat in a game timeline or on a player's individual stat page, a custom playlist is created to show all of the plays where those events happened. The system however does not allow for the combining of multiple search criteria into a single query, limiting the expressiveness of the resulting clip collections.

THE VIDEO LENS FRAMEWORK

There are many usage domains where a user may want to view multiple clips from a collection of video files, such as sports [41], entertainment [42, 43], and learning [12, 14]. Typically, the process of finding and viewing the relevant clips is a manual and time-consuming process.

To support the process of finding and viewing relevant clips within a collection of videos, we created the *Video Lens* framework (Figure 1). The primary goal of the framework is to allow interactive exploration of a collection of videos with associated metadata information. Besides being useful for purposeful, directed tasks, the interface also encourages free-form exploration through the use of highly-interactive controls and immediate viewing of results.

Design Goals

The design of the *Video Lens* framework was directed by a set of four design goals.

D1: Maximize Information Density

One design goal for the *Video Lens* framework is to maximize information density to reveal as much about the underlying data as possible. When considering a large collection of videos, each with a rich set of metadata, there are lots of data points to be displayed. We want *Video Lens* to display as much of that information as possible.

D2: Treat Events as the Primary Entity

Even though *Video Lens* works with a collection of videos, the primary entity of interest are individual "events" within those videos. To preserve as much information as possible, the aggregation of the event data should be avoided.

D3: Support Rapid Playback of Video Results

Playing back the selected video clips should be both immediate to start, and then advance quickly through the clips. Only the relevant parts of a clip should be played, and the next clip should begin without any noticeable delay. This design goal allows for a user to watch as many relevant events as possible in the shortest amount of time.

D4: Encourage Exploration of the Data

Video Lens should work for users who have predefined, predetermined queries they need to perform, but perhaps more importantly, it should also encourage users to freely explore the data. The system can accomplish this goal by exposing as many unique attributes/data points as possible, employing highly responsive and interactive data controls, and minimizing the "cost" of trying something out.

Metadata Requirements and Processing

We assume that each of the videos in a corpus already has an associated set of metadata pertaining to individual "events" occurring within the video. For example, in a software tutorial video, the relevant "events" might be command invocations [7], and for TV shows or movies the events could be individual scenes [38]. Our implementation requires each event to have a timestamp indicating where in the video it occurs, plus some number of descriptive attributes about the event. The event data is stored in a database.

Individual attributes are classified as *discreet* or *continuous*. *Discreet* attributes have a finite number of possible values, while *continuous* attributes can have any value between a maximum and minimum value. If the discreet values are numerical, the attribute is considered to be *ordered discreet*, otherwise it is *unordered discreet*. These attributes form the basis of how specific events within a video are selected for playback, and how the events are visually represented throughout the interface.

Interface Elements

The *Video Lens* interface consists of four main components: three data viewing/filtering components, and the video playback window.

The UI elements in each data viewing/filtering component are tightly linked to each other so that hovering over events in one view highlights them in the other views, and selections in one view are immediately represented in the others. In response to our desire to treat events as the primary entity (*D2*), a constant mapping of "1 dot = 1 event" is followed. That is, a single dot in any of the viewing/filtering UI elements always represents a single event. Additionally, a red dot is universally used to highlight the event currently being played.

Single-Attribute Controller (SAC)

Each attribute from the metadata is represented by a single-attribute controller (*SAC*). Each *SAC* has a relatively short height (20 pixels) to allow for many to be stacked vertically on the display and help maximize information density (*D1*). Each *SAC* has a label with the attribute's name on the left, and a value strip showing the attribute values on the right (Figure 2).

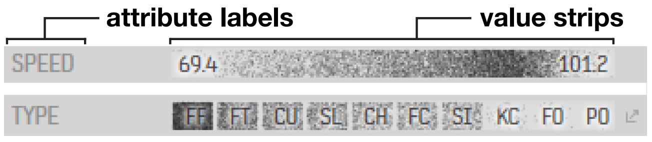

Figure 2. Two example Single-Attribute Controllers. On top, "Speed" is a *continuous* attribute, and on the bottom, "Type" is a *discreet* attribute.

In the value strip, each event is represented by one point. To improve legibility, the opacity of each dot is dynamically lowered as the number of data points increases. The horizontal position of each point is based on the value of the variable for each event. For *continuous* variables the horizontal position is based on the value's relative position between the minimum and maximum values of the attribute, and for *discreet* variables, the value strip is divided into discreet buckets, with the point placed randomly within the width of the proper bucket. With *ordered discreet* variables the buckets are sorted numerically, while *unordered discreet* variables are sorted from highest to lowest by the number of events in each category. For all variable types, the vertical position of each dot is randomly chosen.

Real-Time Brushing

As the cursor moves over a value strip, the events at the cursor position are placed into a "hovered" state. For *discreet* variables, all events within the same bucket as the cursor are considered hovered. While the cursor is moving over one *SAC*, the value strips on all other *SACs* are faded, and the hovered events are highlighted (Figure 3).

This "linking" behavior between multiple plots [2, 3] is typically done in combination with "brushing" or selecting data in one data view, and then applying the highlighting effect to the selected points in the other views. Our real-time hover based approach can expose the user to interactions between the attributes that they may not have even been looking for, thereby encouraging exploration of the dataset (*D4*).

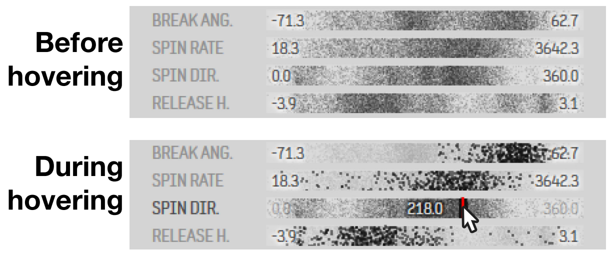

Figure 3. A collection of SACs before and during a hover operation.

Filtering

Filtering the events based on a single attribute is primarily accomplished through clicking or dragging within the value strip. In a *continuous SAC*, clicking selects a very narrow slice of the data range (Figure 4, top), and click-dragging selects a broader region (Figure 4, middle). In a *discreet SAC*, clicking selects all the points in a discreet bucket, and dragging in *discreet SACs* is only supported when the data is *ordered. Discreet* attributes have an icon to the right of the value track which launches a dialog allowing for multiple selections (Figure 4, bottom). Once a selection has been made in a SAC, the areas which have been filtered out are indicated with a hatched background pattern. Additionally, a "clear" icon appears to the left of the attribute label to clear the selection from this attribute (Figure 4).

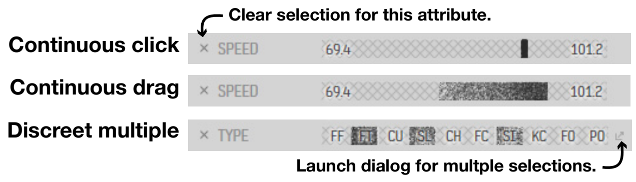
Clear selection for this attribute.
Continuous click
× SPEED 69.4 101.2
Continuous drag
× SPEED 69.4 101.2
Discreet multiple
× TYPE FF FT CU SL CH FC SI KC FO PO
Launch dialog for multple selections.

Figure 4. Example selections made in *continuous* and *discreet* Single-attribute Controllers.

Search and Highlight

Since the *Video Lens* framework supports and encourages the inclusion of many attributes (*D1*), finding a particular event type may be difficult. A real-time search field allows for searching within the attribute names, tooltips, and discreet value labels. As text is entered in the search box, the label or "launch selection dialog" button for any matching *SAC* is highlighted as are any matching entries in the variable selection dialog (Figure 5).

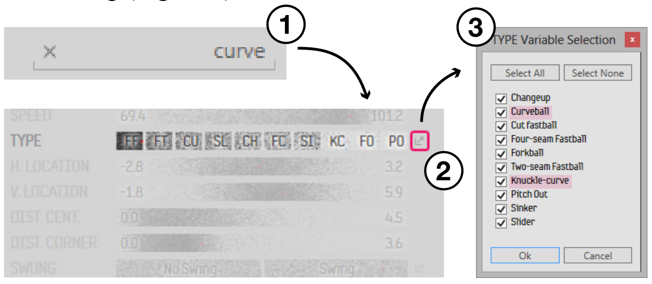

Figure 5. Workflow of the search/highlight feature.

Multi-Attribute Grid (MAG)

The top-right corner of the interface contains a Multi-Attribute Grid (*MAG*) component (Figure 6). The main area of the *MAG* is a two-dimensional grid with an attribute mapped to each of the horizontal and vertical axes. Within the grid are points mapped to each of the events in the active collection (Figure 6B,C). Besides the horizontal and vertical axes, attributes can also be mapped to the color and size of the individual points (Figure 6D,E). These mappings are set by dragging an attribute label from a *Single Attribute Controller* onto one of the four variable dimensions on the *Multi Attribute Grid* (horizontal axis, vertical axis, color, size). The simple interaction of dragging *SAC* labels onto dimensions of the *MAG* encourages exploration of the data (*D4*). The *MAG* also contains controls to clear the current selection, reset the application state, and to save/load specific views (Figure 6H).

For continuous attributes the grid is scaled so the minimum and maximum values span the entire length of the axis. When one axis is mapped to a discreet variable, the axis is split into a set of groups with a small gap between categories (Figure 7, bottom left, bottom right). With both axes mapped to discreet variables, the plot is divided into a number of equal sized cells (Figure 7, top right). Within the grid, events are selected with a lasso (Figure 6F). As with the *SACs*, while hovering the cursor over the *MAG*, the events under the area around the cursor are highlighted in the other event visualizing UI components.

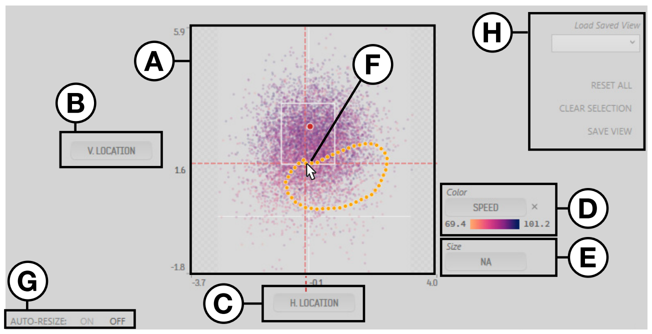

Figure 6. Elements within the Multi-Attribute Grid (MAG) area of the interface.

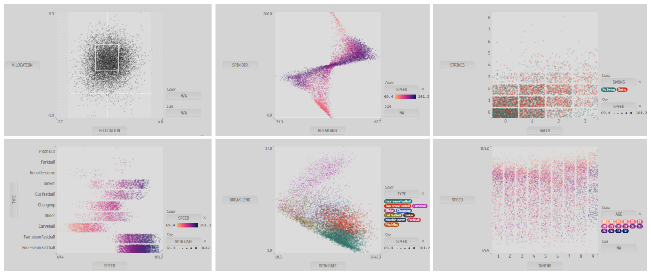

Figure 7. Examples of Multi-Attribute Grid with different combinations of attributes mapped to the dimensions.

The color dimension will accept any facet type, and colors each dot using a coloring scheme based on the variable type (Figure 8). When a discreet variable is used for the color dimensions, small 'pill' buttons show only the values of the attribute which are present in the currently set of events. Hovering over an individual value highlights those events in the grid (and other UI components), and clicking on one filters the selection down to only those matching events. The size dimension is only available for continuous variables.

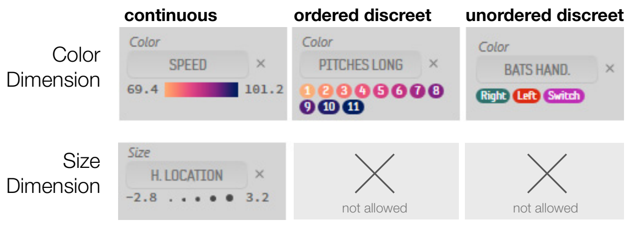

Figure 8. Color and Size dimension appearances for each of the attribute variable types.

Video Timelines

The final set of UI components displaying the events are a set of *video timelines*. There is one timeline for each video in the collection. The events are placed along the timeline based on their timestamp (Figure 9). A scroll bar is used if there are more videos in the collection than will fit on the screen. Hovering, clicking, and dragging operations on the timeline work much the same way as they do for the *SACs*; they highlight and select a set of events. Clicking on the video description label selects all events in the video, and an 'x' icon clears the selection made in the video timeline.

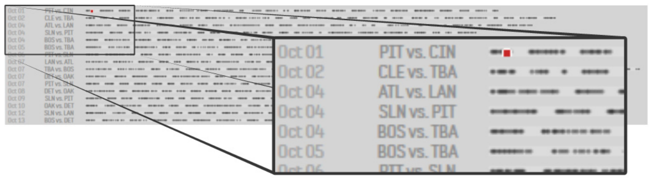

Figure 9. A section of the video timelines. The red dot highlights the event which is currently being played.

Video Playback

The relevant video segments are played in the top-left corner of the interface (Figure 10). Clicking on the video will pause and resume playback. Controls below the video allow the user to manually move forward and back through the collection of clips (Figure 10A). Semi-transparent overlays shows general information about the currently playing clip (Figure 10B), and specific values for the attributes which have filters applied to them (Figure 10C).

By default, the selected events are played back automatically, advancing from one segment to the next once the first event has finished. This is referred to as the *variable* auto-advance setting, since each event is played for a variable amount of time depending on the length of the event. Options are also available to display 1 second or 3 seconds from the start of the event for cases where only the beginning of an event is of interest. Auto-advance can also be disabled completely giving the viewer control over when to view the next clip.

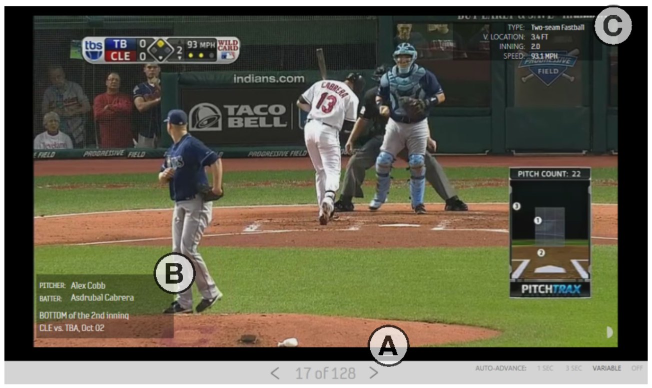

Figure 10. Video Playback section of the interface.

When initializing playback of a video at a specific location, the video decoder requires some amount of time to load the video, and queue playback to the desired start location. To support rapid playback of the results (*D3*), it is important to eliminate the delay between clips. Video Lens uses two video players (*Player*$_1$, *Player*$_2$) overlaid at the same location on the screen. To the viewer there appears to be only one video player. While *Player*$_1$ is in the foreground playing video clip C_n which has a duration of D_n, *Player*$_2$ is in the background loading the next video clip C_{n+1} and beginning playback D_n seconds before the actual start time of C_{n+1}. This way, once C_n is complete, C_{n+1} is already loaded and playing at the start time of the next event. The video players seamlessly swap, with *Player*$_1$ going to the background and preloading the next clip, C_{n+2} (Figure 11).

To accommodate delays in the loading of the background video, either due to network latency or vide decoding, a timer fires every .25s to check the playback position of the background video player against where playback should be, and the playhead position is adjusted if needed.

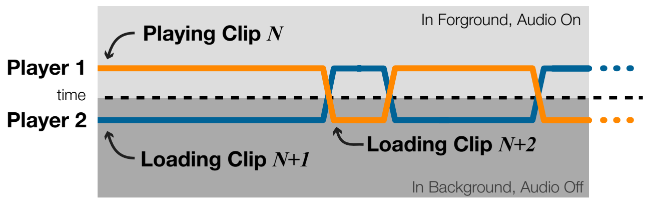

Figure 11. The next video clip in a sequence is always being preloaded by a second background video player.

BVL: BASEBALL VIDEO LENS

Conceptually, the *Video Lens* framework is general enough to accommodate any large set of videos with metadata. In practice however, an implementation for any specific usage domain may impose unique design challenges.

As a test case for the *Video Lens* framework we created *BVL*, or *Baseball Video Lens*. In general, sports are a popular test bed for visualization systems [19, 20, 22], and we chose baseball as the target domain due to the large amounts of professional baseball video produced each year and the depth of event-based metadata which has become available for these games in recent years.

The *BVL* system is designed with two main usage types in mind: *casual* usage by serious baseball fans, and *professional* use by baseball teams, broadcaster organizations, or journalists. While each of these user groups have different goals (Table 1), they all have a need/desire to find collections of clips within a large set of baseball videos.

CASUAL	SERIOUS BASEBALL FANS Find clips they like; *Watch all the times their favorite player hit a homerun.* Quickly watch a game; *Watch the events from game, but skip downtime between pitches.*
PROFESSIONAL USERS	BASEBALL TEAMS/FRONT OFFICE Advanced scouting on next opponent; *Where does tomorrow's pitcher usually throw his curveball?* Self assessment; *Why is our player having trouble hitting inside fastballs?*
	BROADCAST ORGANIZATIONS Find the clip of the last time a particular circumstance occurred. *Show the clip of the last time somebody hit a ball over 400 feet but only got a single.*
	JOURNALISTS/BLOGGERS Collect all the occurrences of an interesting event for a story; *Look at all the pitches that bounced before the plate yet resulted in a hit. Watch the lowest pitches hit for a homerun last year.*

Table 1. Sample use cases of the Baseball Video Lens system for each of the target user groups.

Implementation

The Video Lens framework is a C# Microsoft WPF application. The metadata is stored on the Amazon AWS cloud, and to simplify implementation, all videos are stored locally.

Data Collection

For this initial prototype version of *BVL*, we use a collection of 29 baseball games from the 2013 MLB playoffs. The "pitch" is considered the primary even type in the data, so throughout the interface the visual convention is "1 dot = 1 pitch". The total length of video in the corpus is 105 hours, and a total of 7983 pitches were thrown. The metadata for these events were obtained from the "PITCHf/x" system [6] and the MLB Advanced Media department XML files [24], both freely available for download. Together these data sources provide a wealth of pitch and game-based metadata. We downloaded and imported these files into a database with the structure shown in Figure 12. We chose to expose a large subset (43) of attributes in the interface, as well as three attributes which were dynamically computed from other fields (distance from center of strike zone, distance from corner of strike zone, and whether or not the batter swung at the pitch). A specific implementation could tune the number of visible attributes to the expertise level of the target user group.

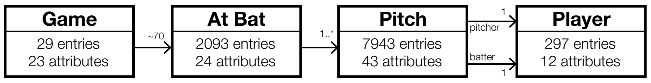

Figure 12. Object classes and relationships from the baseball dataset.

The metadata contains a timestamp for the beginning of each pitch, but no indication of the length of each event. To support rapid playback (*D3*), we need at least an estimate of how long a viewer would like to watch each play. Ideally we could manually determine an "end time" for each event, but even for a small set of events this is not practical. To estimate event lengths at a higher level, two authors independently watched a number of clips for each *at bat* type (home run, strikeout, etc.). For each play, they recorded when they considered the play to be "over". Based on these timings, nominal event lengths for each *at bat* type were determined.

Timeline Alignment/Audio Analysis

To synchronize the events with the video, we need timestamps of each event. Unfortunately our meta-data only provides the time-of-day for each event (8:32:12PM, for example). One method to coordinate the times is to manually mark the time of the first event relative to the beginning of the video. This offset could then be applied to all remaining events. However, even for a small collection of videos this is a time consuming and error prone process.

To develop an automatic process, we leveraged the audio track of the videos. Within the audio there is a noticeable spike from the "crack of the bat" when a pitch is hit, as well as when the ball hits the catcher's glove (Figure 13).

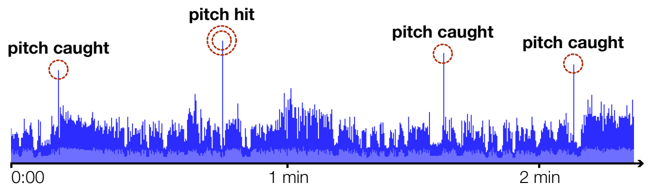

Figure 13. Audio waveform from a 2:15 long section of a baseball game showing the spikes during pitch events.

To detect these spikes we perform a sweep over the audio track at a resolution of 20ms, looking for peaks greater than 2 times the average amplitude of the previous 3 seconds. This gives a collection of "probable pitch events". The system then looks at possible offsets for the metadata file ranging between 0 and 45 minutes, in steps of 1/24th of a second (equivalent to one frame of video). At each step a simple fitness function calculates how well the metadata timing fits to the "probable pitch event" data. The fitness function looks at the time of each pitch in the metadata plus the current offset. If the set of probable pitch times contains an entry within 0.5 seconds of that time, the fitness score is incremented. At the end of each iteration, the fitness score indicates how many pitches in the metadata found a "match" in the probable pitches extracted from the audio, and the time with the highest score is used as the offset for the game (Figure 14).

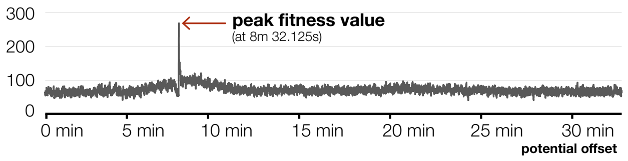

Figure 14. Graph of fitness values for a single game over a range of possible metadata offset times, with the maximum fitness value and time highlighted.

This technique worked to automatically align all 29 game videos with their metadata. The calculations took less than a second per game on a workstation computer.

Timing Micro Adjustments

Another limitation in our timing metadata is that time is rounded to the nearest second. In casual viewing situations one could start playback 2 or 3 seconds before the actual pitch event to ensure the pitch is seen. However, since one of the goals for *Video Lens* is to watch as many events as possible in the shortest amount of time (*D3*), we desired a more accurate solution.

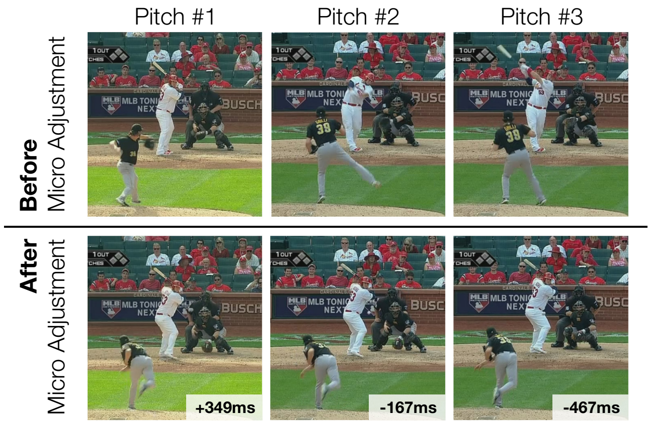

Figure 15. Frame 0.5 sec into clip for using the *one second* auto-advance mode for three pitches *before* and *after* the audio-based timing micro adjustments are applied. (Note the varied position of the pitcher and batter in the *before* row, and the consistent positions *after*.)

We again used the detection of probable pitches through the audio track to calculate the exact pitch times at a finer resolution. For example, if the metadata reports the pitch occurred 38 seconds after the first pitch, but a spike is detected at 38.375 seconds, we adjust time of the pitch in the metadata to 38.375. When using the *one second* auto-advance setting, the video begins playback 0.7 seconds before the pitch reaches the plate and continues for 0.3 seconds afterwards. Figure 15 shows the frame 0.5 seconds into the clip for three consecutive pitches before, and after, the micro adjustments are applied.

The algorithm was able to successfully find a "probable pitch event" within the appropriate time window and apply the timing micro adjustment to 94% of the pitch events in the database.

Sample Tasks

To demonstrate several ways in which the *BVL* system can be used, we now provide a walkthrough of how to complete three representative tasks.

Task 1: Watch the lowest pitches hit for a homerun

First, the user filters the "at bat event type" to only show the homeruns (Figure 16, Step 1). Then the user makes a lasso selection around the lowest points in the Multi-Attribute Grid (Figure 16, Step 2), which is by default mapped to the horizontal and vertical location of the pitch. Alternatively, the user could select the lowest values in the "V. Location" *single-attribute controller* (Figure 16, Step 2 alternate). The three homerun clips immediately play in the video player.

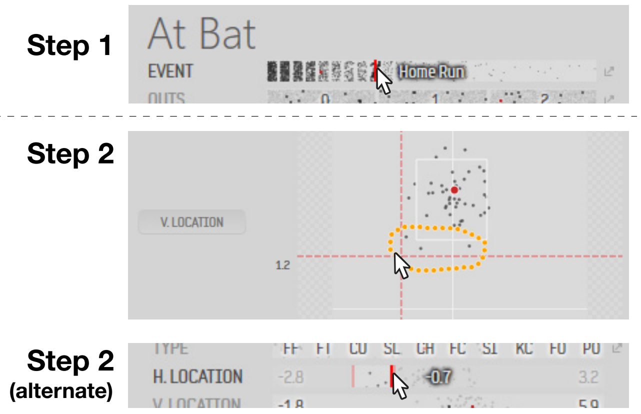

Figure 16. Watching the lowest pitches hit for a homerun.

Task 2: Watch all strikeouts in a particular game

For this task we are going to watch all strikeouts in the October 2nd game between the Tampa Bay Rays and the Cleveland Indians. The user could first click on the video timeline label for the "OCT 2 CLE vs TBA" game (Figure 17, Step 1). At this point the video player displays all pitches from this game, and using the *variable* timing auto-advance mode, would give the viewer a quick way to watch the recap of an entire game. To watch only the strikeouts, the user would filter the "at bat event type" to show only strikeouts (Figure 17, Step 2). Due to the faceted nature of the filtering interface, these two steps could be done in either order.

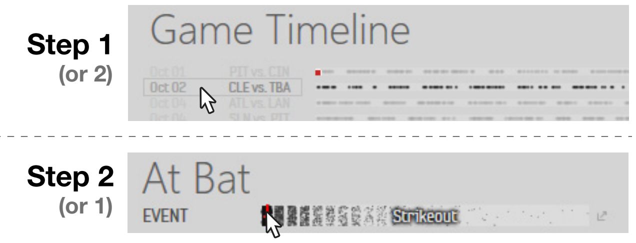

Figure 17. Watching all strikeouts in a particular game.

Task 3: Analyze the release points of Jon Lester's pitches
This is the type of task baseball teams often do for "advance scouting" of their opponent. In this case we imagine that we are playing against Jon Lester tomorrow and want to see what we can learn about how he pitches.

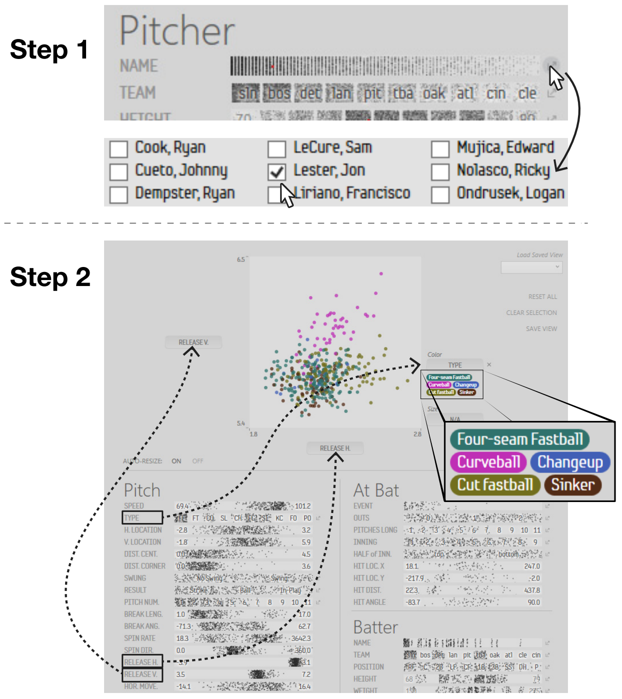

Figure 18. Analyzing the pitches thrown by Jon Lester.

The first step is to filter the "pitcher name" attribute to "Lester, Jon" (Figure 18, Step 1). Then, we drag the "Release H." and "Release V." attributes, representing the horizontal and vertical position of the ball as it leaves the pitcher's hand, up to the *MAG*. At this point we see the location where all of the pitches were released. To look for patterns, we drag the "pitch type" attribute to the *color* dimension (Figure 18, Step 2). Now we can see that the pitcher releases his curveballs from a higher position than his other pitches. This information can increase the batters chance of getting a hit [44]. At this point the clips of all Jon Lester pitches are playing, so we can watch the video and verify these visual tells.

EVALUATION

To evaluate the *BVL* system, we recruited members from each of the four target user groups (Table 1).

Qualitative User Study

We first wanted to get feedback on the *BVL* system for recreational use by serious baseball fans. We recruited 12 paid volunteers (2 women) between the ages of 27-50 (mean 32) who classified themselves as "baseball fanatics". Participants reported watching an average of 15 hours of baseball-related programming per week, and 11 of the 12 watch baseball videos on a secondary device such as a tablet or phone. Additionally, 8 of the 12 participants subscribe to MLB.TV, a paid service which provides access to all games and highlights throughout the MLB season.

The study took approximately 45 minutes and was conducted on a desktop computer driving a 42" LCD. For the first 10 minutes the proctor demonstrated the features of the system, followed by 5 minutes for the participant to explore the system and ask the proctor questions. For the final 30 minutes the participants worked through a series of 18 tasks (including the three described in the previous section) (Figure 19). Some tasks were adapted from a series of posts on a popular blog for analytics-minded baseball fans. The collection of tasks was designed to utilize the full set of functionality available in the system. Participants were timed from when they began the task until the system was set up to view the resulting clips. After the study tasks, participants were given 5 minutes to experiment with the system before completing a post-study questionnaire.

Participants were able to complete 7 of the 18 tasks in a median time less than 30 seconds, and 16 of the 18 tasks in a median completion time less than 1 minute (Figure 19). The two most difficult tasks used attributes the participants were not necessarily familiar with. The increased time for these tasks was mostly due to the participants thinking through the task and coming up with an optimal strategy.

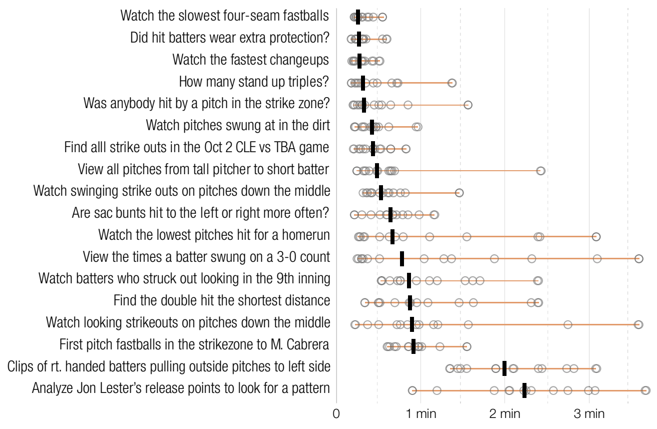

Figure 19. Completion times for each of the study tasks. Median time shown with black bar.

The subjective results are shown in Figure 20. Participants were positive about the system, all saying they found the system enjoyable to use, and 11 out of 12 said that they would use the system if it were available to them; *"This is amazing! If I had this I would lose entire days using it.* (P2)*"*.

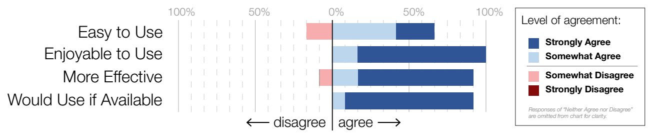

Figure 20. Subjective results from participant survey.

Participants were generally impressed with the set of video-browsing tasks which were made possible by our system, with one participant saying he *"...would have no idea how to even begin finding these clips without this system* (P6)*"*.

Despite the large number of UI elements in the interface, participants found the interface generally easy to use: *"It is* *remarkable how easy it is to go from the huge collection of pitches down to the couple of things you want* (P3)*"*. They were also genuinely interested viewing the clips they found during the study, even though they were free to move on to the next task once the query was set up. In fact, 4 of the 12 participants asked in the middle of the study if they could try a query of their own, sparked by one of the study tasks. While they were asked to wait until the study was over, this reinforces the idea that they really would like to have access to such a system.

Professional User Feedback

To get feedback on the suitability of the *BVL* system from users in a professional context, we recruited members from the three identified professional user groups. We recruited baseball analysts from two MLB teams (MLB1, MLB2), a writer/editor from a popular baseball analytics website (WR1), and a researcher from a major sports broadcasting organization (BR1). We sent each a copy of the system to experiment with on their own time, and a short video and document explaining how the system worked. After they had a chance to use the system, feedback was provided to us through a phone interview or written questionnaire.

Overall, the response from the professional users was encouraging. After seeing a two minute video of the system in action, WR1's initial response was *"holy god damn that's incredible."* Given the chance to use the program, WR1 appreciated the ease of which he could try new things, *"The tool essentially lets me play. I can go exploring, like you would through the woods. It would prompt questions I might not have ever even thought of asking."* Many of WR1's articles contain animated .gifs for a collection of events. His current workflow involves finding the events from a database, loading up MLB.tv, playing the appropriate game, and manually finding the event. WR1 stated that *BVL* could *"conceivably save [me] hours a week."*

From a broadcasting perspective, BR1 said that finding video clips based on event metadata is something that he currently needs to do, and that he could see the *BVL* system being useful for broadcasting organizations. His organization currently uses a custom-made, internally-designed tool for these sorts of tasks, but he liked the speed in which *BVL* was able to access and display the clips. He suggested that in broadcasting the key is to get to the clips quickly (rather than exploring the data), and that a version of *BVL* with less attributes might be more suitable in a broadcasting scenario.

From the perspective of professional baseball teams, both MLB1 and MLB2 felt the *BVL* system was something that would be useful for MLB teams, and without any hands-on training, were comfortable using the system within a couple of minutes. Both organizations have existing proprietary solutions they currently use for these types of tasks.

In particular, MLB1 liked the *"flow"* of the *BVL* system and mentioned *"The UI is really nice. Visualizing the ranges of the variables makes it easy to eliminate the things I don't want."* MLB1 also liked the immediacy in which he could get results, as the system his team currently uses requires him to set up the entire query up front, and then wait up to a minute before the results are presented to him. MLB2 suggested that for maximum utility for MLB teams, it would need to be integrated with existing scouting and statistical databases.

DISCUSSION AND FUTURE WORK

The main function of the *Video Lens* is to allow users to find and view relevant video clips. Visualizing the attribute values in the various UI components are primarily meant to aid in the workflow of selecting a set of events to watch. However, several participants saw on opportunity to extend the system to also answer more analytical questions (e.g. "What percentage of curve balls are hit for home runs?").

Another common request was the ability to see "what happens next". For example, P1 wanted to see all the pitches Clayton Kershaw threw *after* he threw a curveball. This type of functionality, along with options for sorting the clip order during playback would be useful additions.

To represent the distribution of values within the *Single Attribute Controllers* we looked at several other visual encodings besides the overplotted dots, including area charts, line graphs, and 1D colored histograms. Besides the benefit of maintaining continuity with the dot/event representation of the *Video Timelines* and *Multi Attribute Grid* components, the dots also provided the clearest representation of outlying data values, which are often the most interesting ones to watch. We would like to continue exploring the benefits of this style of visual representation.

Our implementation of the baseball video lens raised several interesting domain specific challenges. We believe *Video Lens* could be adapted to other usage domains, but similar domain-specific issues should be expected. For example, we addressed the timing discrepancies using audio information specific to baseball. It would be interesting to explore what audio/visual clues could be used to perform the task of aligning metadata in other domains. Perhaps for soccer we could use the increased volume of the fans and announcers [35], or for software tutorials we could analyze the position and appearance of the cursor and buttons [1].

Overall, we believe the *Video Lens* framework represents a valuable solution for exploring a large collection of metadata rich video content, and the *Baseball Video Lens* system can serve as a baseline example for future work in this area.

REFERENCES

1. Banovic, N., Grossman, T., Matejka, J., and Fitzmaurice, G. Waken: Reverse Engineering Usage Information and Interface Structure from Software Videos. *ACM UIST*, (2012), 83–92.
2. Becker, R.A. and Cleveland, W.S. Brushing Scatterplots. *Technometrics 29*, 2 (1987), 127–142.
3. Buja, A., McDonald, J.A., Michalak, J., and Stuetzle, W. Interactive Data Visualization Using Focusing and Linking. *IEEE VIS*, (1991), 156–163.

4. Capra, R.G. and Marchionini, G. The Relation Browser Tool for Faceted Exploratory Search. *ACM JCDL '08*, (2008), 420–420.
5. Derthick, M. Interactive Visualization of Video Metadata. *ACM JCDL '01*, (2001), 453–.
6. Fast, M. What the heck is PITCHf/x. *The Hardball Times Annual*, (2010), 153–8.
7. Grossman, T., Matejka, J., and Fitzmaurice, G. Chronicle: Capture, Exploration, and Playback of Document Workflow Histories. *ACM UIST '10*, (2010), 143–152.
8. Hearst, M. UIs for Faceted Navigation: Recent Advances and Remaining Open Problems. *Int. Journal of Machine Learning and Computing*, (2008), 337–343.
9. Jackson, D., Nicholson, J., Stoeckigt, G., Wrobel, R., Thieme, A., and Olivier, P. Panopticon: A Parallel Video Overview System. *ACM UIST '13*, (2013), 123–130.
10. Kim, J., Nguyen, P., Weir, S., Guo, P.J., Miller, R.C., and Gajos, K.Z. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. *CHI,* (2014).
11. Koren, J., Zhang, Y., and Liu, X. Personalized Interactive Faceted Search. *ACM WWW '08*, (2008), 477–486.
12. Lafreniere, B., Grossman, T., Matejka, J., and Fitzmaurice, G. Investigating the Feasibility of Extracting Tool Demonstrations from In-Situ Video Content. *CHI, 2014.*
13. Lee, B., Smith, G., Robertson, G.G., Czerwinski, M., and Tan, D.S. FacetLens: Exposing Trends and Relationships to Support Sensemaking Within Faceted Datasets. *ACM CHI*, (2009), 1293–1302.
14. Matejka, J., Grossman, T., and Fitzmaurice, G. Ambient Help. *ACM CHI*, (2011), 2751–2760.
15. Matejka, J., Grossman, T., and Fitzmaurice, G. Swift: Reducing the Effects of Latency in Online Video Scrubbing. *ACM CHI*, (2012), 637–646.
16. Matejka, J., Grossman, T., and Fitzmaurice, G. Swifter: Improved Online Video Scrubbing. *CHI*, (2013).
17. Monserrat, T.-J.K.P., Zhao, S., McGee, K., and Pandey, A.V. NoteVideo: Facilitating Navigation of Blackboard-style Lecture Videos. *ACM CHI*, (2013), 1139–1148.
18. Olsen, D.R., Partridge, B., and Lynn, S. Time Warp Sports for Internet Television. *ACM Trans. Comput.-Hum. Interact. 17*, 4 (2010), 16:1–16:37.
19. Perin, C., Vuillemot, R., and Fekete, J.-D. SoccerStories: A Kick-off for Visual Soccer Analysis. *IEEE Trans. on Vis. and Computer Graphics 19*, 12 (2013), 2506–2515.
20. Pileggi, H., Stolper, C.D., Boyle, J.M., Stasko, J.T., and Member, S. SnapShot: Visualization to Propel Ice Hockey Analytics. *IEEE Vis* (2012).
21. Pongnumkul, S., Dontcheva, M., Li, W., et al. Pause-and-play: Automatically Linking Screencast Video Tutorials with Applications. *ACM UIST*, (2011), 135–144.
22. Rao, R. and Card, S.K. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information. *ACM CHI '94*, (1994), 318–322.
23. Schoeffmann, K., Taschwer, M., and Boeszoermenyi, L. The Video Explorer: A Tool for Navigation and Searching Within a Single Video Based on Fast Content Analysis. *ACM MMSys '10*, (2010), 247–258.
24. Schumaker, R.P., Solieman, O.K., and Chen, H. Web Sports Data Extraction and Visualization. In *Sports Data Mining*. Springer, 2010, 71–87.
25. Smeaton, A.F., Over, P., and Kraaij, W. Evaluation campaigns and TRECVid. *MIR '06:* (2006), 321–330.
26. Song, Y., Hua, X.-S., Dai, L.-R., and Wang, M. Semi-automatic Video Annotation Based on Active Learning with Multiple Complementary Predictors. *MIR* (2005).
27. Stone, B. MLB.TV Adds Enhanced Video and User-Selected Replays. *The New York Times*, 2009. http://www.nytimes.com/2009/02/09/technology/internet/09mlb.html.
28. Stuart-Moore, J., Evans, M., and Jacobs, P. Interface Design for Browsing Faceted Metadata. *JCDL* , (2006).
29. Vandic, D., Frasincar, F., and Kaymak, U. Facet Selection Algorithms for Web Product Search. *CIKM*, (2013).
30. Voigt, M., Werstler, A., Polowinski, J., and Meißner, K. Weighted Faceted Browsing for Characteristics-based Visualization Selection Through End Users. *EICS,* 2012.
31. Worring, M., Nguyen, G.P., Hollink, L., Van Gemert, J.C., and Koelma, D.C. Accessing video archives using interactive search. *ICME*, (2004), 297–300 Vol.1.
32. Yee, K.P., Swearingen, K., Li, K., & Hearst, M. Faceted metadata for image search and browsing. *CHI*, 2003.
33. Ben-Yitzhak, O., Yogev, S., Golbandi, N., et al. Beyond basic faceted search. *WSDM*, (2008).
34. Van Zwol, R., Ng, P., Ramani, A., et al. Faceted exploration of image search results. *WWW*, (2010).
35. Efficient Multimodal Features for Automatic Soccer Highlight Generation. *IEEE ICPR*, (2004), 973–976.
36. selfiecity. *selfiecity*, 2014. http://selfiecity.net/.
37. NBA Stats - Stats Video. http://stats.nba.com/statsVideo.html.
38. TVplus | Synced Metadata. http://www.tvplus.com/#!synced-metadata/c11fw.
39. Baseball Savant: Your Source For Advanced MLB Statistics. http://baseballsavant.com/pitchfx_search.php.
40. BATS, Sydex Software. http://www.sydexsports.com/html/baseball.html.
41. College Football 2012-2013 Season Highlights (Part 1) | Catches, Hits, Fails, Amazing Plays Etc. - YouTube. https://www.youtube.com/watch?v=PlDtqO1hR3k.
42. Kramer's Entrance - YouTube. https://www.youtube.com/watch?v=XXSGV5wEv1o.
43. David Letterman: "Are Those Your Drums?" - YouTube. https://www.youtube.com/watch?v=M65Dx0STe2M.
44. Interpreting PITCHf/x Charts | FanGraphs Sabermetrics Library. http://www.fangraphs.com/library/interpreting-pitchfx-charts/.