2010 4th International Conference on Distance Learning and Education (ICDLE)

Opinion Maps for Tracking and Visualizing Feedback on Digital Video Content

Nikhil Sharma
Amrita University
AIMS Campus, Cochin, India
Email: nikhilsharma@aims.amrita.edu

Raghu Raman
School of Engineering
Amrita University
Kerala, India
Email: raghu@amrita.edu

***Abstract* — An increasingly large amount of digital video content continues to become available on the Internet. This content ranges from short entertainment clips by amateurs lasting a few minutes to full-fledged, significantly longer, serious course lectures and presentations by subject matter experts. As these video bits of content continue to get pushed out one-way, to the learner, the content authors have no quick or easy way of getting an objective, machine-processable opinion on arbitrary subsections of their content.**

In this paper, we explore the design and implementation of an innovative software *visualization system* that uses *'Opinion Maps'* and a WYSIWYG *(When You See Is When You Give)* approach to bring a much needed real-time focus to the feedback aspect of the video mode of instruction. In addition to using routine graphical charting to visualize server-side tabular data, an Opinion Map employs a combination of (i) a Tukey Box$^+$ plot — a simple but powerful extension to the Tukey Box plot — to present N-point ordinal opinion scores, collected from learners potentially for every single second of the video content, (ii) a per-second opinion score histogram, and (iii) overall 'best' and 'worst' opinion score visualizations. The system is nonintrusive; it requires no special content authoring processes and bolts on to prerecorded digital video content.

Keywords: real-time asynchronous feedback; visualization; video; opinion map; Tukey Box plot; Tukey Box+ plot

I. INTRODUCTION

Ever since multimedia became mainstream, digital video has been the preferred mode of instruction of by far a vast majority of learners and students.[1] As Internet penetration deepens worldwide and as the bandwidth gets ever bigger and cheaper for the average end-user, the demand for video-based instruction is poised to only grow, and grow at an exponential rate [1][7]!

The sheer growth in the size of the video audience over the past few years has translated into a proportional increase in demand for high-quality video content. Content authors and teachers rely more and more on the video mode of instruction for reaching out to their students, which is leading in turn to a further rise in the size of this audience. A highly desirable virtuous cycle, in other words, from the point of view of all involved.

Now, in any teacher-student relationship, it is ultimately the unsolicited and ongoing feedback from the astute students as well as peers that helps improve the quality and content of the instruction. For example, the teacher may be of the highest credentials and the course content the most meticulously planned, there may still be unintentional lacunae in such nebulous things as communication, emphasis, timing, and delivery. Not only that, the teacher can and will never have an *a priori* knowledge of the *precise* background of each and every member of her audience, leading inevitably to a gap between the assumed and the actual level of this background.

II. THE PROBLEM

In traditional and still extant *'old-school'* settings, where students are all collocated in the same physical classroom, the student to teacher feedback is a nonissue: the feedback is always visual, direct, real-time, and usually *synchronous*. Synchronous, in the sense that a student can potentially stop the flow of instruction at any time by choosing to interact with the teacher.

In technology-enabled, distributed learning systems of today, however, there exists no similar mechanism to accept any type of real-time feedback as the instruction gets delivered to the student. There lurks an unfortunate asymmetry between the teacher-student and student-teacher communication channels: the teacher has no rich feedback on how well-received or otherwise each bit of the transmitted content was, and what could perhaps be done to improve its quality even further. This is especially true of video replays, since in a live-cast or in the original recording of a video, students *can* and *do* sometimes interact with the teacher. It should be noted, however, that a video-based instruction is almost never a one-time, use-and-throw artifact; it is meant to be played and replayed. Hence, the problem of the inability to provide feedback during a replay cannot be dismissed just because there occurred some, onetime student-teacher interaction in the original live-cast of a video.

While some video content portals[2] allow their viewers to provide an overall rating of a given video on a scale of 1 to 5, this rating system is not a popular or even a typical feature in

[1] Hereafter, and unless so emphasized, we shall use the terms in the following sets interchangeably: {learner, viewer, student}, {course, content, video, video content}, {course author, content author, teacher}.

[2] such as You Tube

the learning management systems of today. And, when emerging and serious knowledge dissemination portals such as NPTEL[3] decide to host their video content off You Tube servers,[4] the problem only goes unaddressed. To use a sci-fi metaphor, it is almost like broadcasting a message into the outer-space: Our message is rich and intelligent in content and we also hope that it is assimilated by an equally or more intelligent life on the other end... but devoid of any *feedback* we have no way of knowing the fate of our message. A two-way contact, of some sort, it seems, is a basic prerequisite for intelligent communication and progress, not only between civilizations but also within.

III. The Current State-Of-The-Art

Today, the most popular — and in fact the only — ways by which a viewer can provide feedback on an online, digital video content are email, and predesigned forms and questionnaires, which in our view, are all *ad hoc* and, thus, highly subjective. They are certainly good devices to capture detailed, descriptive feedback but can neither capture nor expose, say, an aggregate, machine-processable opinion score[5] for a unit of digital video content.

The aforementioned rating mechanisms employed by some video content portals do allow an *overall* rating. This may even be more than adequate for short, informal content. However, in case of serious, long-lengthed content such as course lectures and formal presentations by subject matter experts, this approach fails, and indeed miserably so. For example, the content may have one or more segments that may be radically different in quality relative to the rest. How can a learner indicate this to the content author? As of this writing, the only way to provide this feedback is to manually observe and then type the start and end times of each such segment in the textual comments section — if one is at all available and enabled[6] — accompanied by some sort of descriptive text. This is painful as well as time-consuming even when the feedback provider happens to be articulate enough for the job. It is discouraging at the very least and requires a sufficient initiative and participation on part of the viewer. Lastly, if the video content ends up becoming unusually popular (or, unpopular), the content author has no easy way to machine-process and analyze such a descriptive feedback on the content by hundreds of thousands of its viewers.

IV. Our Solution

Having considered the motivations behind the problem that we are attempting to solve, let us now consider a few constraints and desirables that any pragmatic, real-world solution must take due cognizance of.

First, the technology used in building the solution must be mainstream. It should not be even a little incongruent with everything else the end-users are used to using.

Second, the solution built around this technology should be easy and intuitive enough to use. When put to actual use in a real-world learning situation, it should itself not get in the way of the main goal of the student, *viz.*, learning. The act of providing feedback on the content must *not* steal away from the student the primary focus and time needed to learn and assimilate the content in the first place.

Third, in order for the feedback to be amenable to machine processing and analysis at a later stage, it should be as objective as possible, and not *ad hoc* or descriptive.

Fourth, the temporal granularity of the feedback should be fine enough, but no finer. It should be possible to correlate easily, in time, the segment of the content the feedback applies to.

Finally, to emulate a physical classroom setting (or, at least its positive aspects) it should be possible to provide the feedback in real-time as the content gets delivered, and not submitted *post hoc*. One of the psychologies operating in a typical learner's mind is to learn and quickly move on to 'the next thing' without being held back or even slowed down in any way via questionnaires, opinion polls, and such quality assessment devices at the *end* of a learning session. The feedback seeking philosophy in the solution should essentially be WYSIWYG:[7] When You See Is When You Give!

A. Choice of Technology

While there may be many alternatives, equally or perhaps more superior technologically and aesthetically, we have elected to use a browser-hosted, Java/Swing-based applet system. Java is an extremely mature, stable, sophisticated, and well-known platform in the distributed computing world, so much so that we need not be extolling its merits here. As of release 7, it has even jumped on the open-source bandwagon. Its Swing package too is fairly mature and sophisticated for 2D graphics and charting, which essentially are our primary needs here. The web browser, similarly, is a ubiquitous thin-client with which the average end-user evinces a high degree of familiarity. For sake of completeness, we would like to mention that the web browser (such as Firefox) and Java are both vendor- and platform-agnostic. This would allow our solution to run unchanged with zero additional effort on a plethora of operating systems and hardware specifications. For playing video content, we have elected to consider Flash Video only for our reference implementation, though the concept proposed in this paper should apply to other formats just as well. On the server side, we have a Tomcat-hosted[8] servlet that, in addition to serving requests originating from the client-side applet, also acts as a server-side model.[9]

We shall now describe the proposed system in increasing levels of detail, starting from the user's perspective of the

[3] National Programme on Technology Enhanced Learning

[4] as of this writing

[5] Hereafter, we use the terms 'opinion', 'opinion score', and 'rating' interchangeably.

[6] Since a textual comment can potentially have objectionable language or tone, the site operators either choose to moderate user submitted feedback or enforce account login as a precondition.

[7] an unintentional but welcome coincidence with the original acronym for What You See Is What You Get

[8] Tomcat, version 6.x

[9] in the Model View Controller (aka, MVC) scheme of things

system through all the way to the very internals of one reference implementation of the system that we have built. In addition to being a fully usable piece of software that can be deployed on a real-world website today, this reference implementation has also served as a solid proof of concept for us while writing this paper.

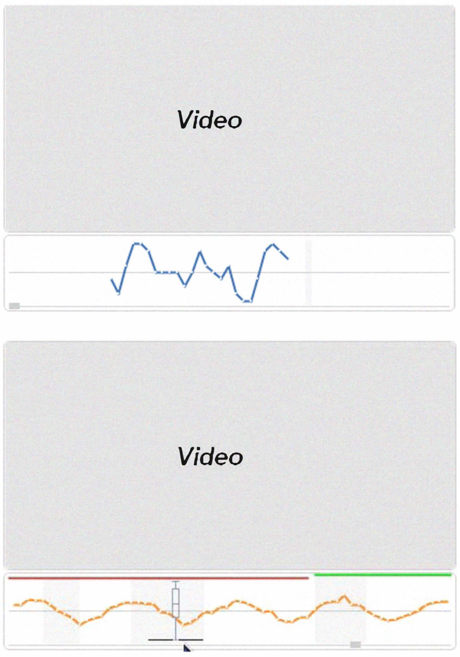

Figure 1. The X-axis represents time and the Y-axis opinion levels. While a student sees her own opinion map only, the teacher sees an aggregated opinion map of all students with interactive Tukey Box$^+$ plots for every second of the video content. The student console (top) allows its user to provide real-time feedback via a mouse or keyboard as the video plays. The teacher console (bottom) presents to its user easy-tovisualize statistics on the aggregate opinion map for the video. Also shown in the teacher console are three time-spans (in light gray), the Tukey Box$^+$ plot for these time-spans, and the best and the worst opinion lines (in green and maroon, respectively).

B. *The User Perspective*

The system proposed in this paper assumes two types of users: students and teachers. The student view, in Fig. 1(top), is composed of two subviews: (i) the player view, which plays the video content, and (ii) the student console. The student uses the student console to provide real-time feedback as the video plays.

The teacher view in Fig. 1(bottom) is, similarly, composed of two subviews: (i) the player view, and (ii) the teacher console. The teacher console presents to its user, the teacher, easily visualizable statistics via an interactive and intuitive user-interface. It is the job of the site administrator to make the right view available to the right user. We do not delve into the problem of view delivery in this paper; depending on the usage policy of a site, this could be done either via login authentication upon each visit of the user or via a nonexpiring cookie that needs a one-time authentication only. What is crucial to note is that the student and teacher views provide very different sets of features and, thus, best exist disparately in line with the second desirable listed in Section IV.

The student accesses a video of interest and is presented with the student view. The moment the video is played (either manually by clicking the player's play button, or implicitly after the HTML page loading completes), the student console becomes 'active' and is ready to receive opinions from its user. An opinion, in general, is an N-point integral score on an ordinal scale in the inclusive range [-$\lfloor N/2 \rfloor$, + $\lfloor N/2 \rfloor$] with N being odd. It is up to the site administrator to choose a suitable value for N and the meaning or description to ascribe to each of these values. In this paper, we have chosen $N = 9$, giving us an opinion range of [-4, +4]. Though we have elected to name these opinions as 'Unacceptable,' 'Very Bad,' 'Bad,' 'Poor,' 'Normal,' 'Fair,' 'Good,' 'Very Good,' and 'Excellent' in the increasing order of their ordinal values, these names *per se* are irrelevant for the purpose of machine aggregation and analysis; they exist only to serve as good mnemonics, say, when publishing site feedback guidelines for the users. (See Section V for a few caveats on the value of N.)

The student console remains active while the video plays. If the video is paused, the console becomes inactive again. This is readily seen via a visual, temporal cursor that moves forward (to the right) with the passage of time. The student gives her opinion with the clicks of a mouse: a left-button click changes the current opinion value by -1, a right-button click by +1, and a middle-button click carries over the previous value of the opinion. For $N = 9$, an opinion can also be given numerically via the keyboard. This opinion is applicable only to the current second, and this is indicated by the temporal cursor. Because the console cannot aesthetically or even usefully display the time-span of arbitrarily long videos, the contents of the console automatically scroll to the left, enough to make room for new opinions to be submitted.

The opinions submitted by the user are connected with straight lines which, taken all together, comprise the student's 'opinion map[10]' for the video. If the student seeks to an earlier time in the video (via the player controls), the console too syncs itself up with this time instant and allows the student to alter previously given opinions. When the video finishes playing, this opinion map gets automatically submitted to the server where it gets aggregated, in real-time, with opinion maps submitted by other students in the past.

In the teacher console, the teacher sees the mean opinion map for all prior opinion maps submitted by the students. Because the teacher need not submit opinion on her own video, this ability (of submitting opinion maps) is absent by design in the teacher console. Since a few extreme values can easily skew the statistical mean of a dataset and lead to unsound conclusions, we elected to present, on a mouse-hover event, the summary statistics of each second via Tukey Box$^+$ plots. A Tukey Box$^+$ plot, in addition to presenting the

[10] We use the initial-caps 'Opinion Map' style of spelling to refer to overall opinion tracking and visualization system presented in this paper; when spelled in lower-case, as here, it is equivalent to and is to be read as 'set of opinions'.

five-member set[11] summary of a dataset, gives a visual indication of their relative frequencies against the backdrop of the dataset size. Note that, unlike traditional applications of a boxplot, our solution does *not* employ this device to juxtapose and size up a set of two or more comparable surveys (or, batches) of a domain; it rather invokes them individually to display useful summary statistics of what essentially are logically different and disconnected survey domains. We trivially intuit that each second of content is independent of and different from any other second and hence not sensibly comparable via boxplots.

The teacher console offers three additional, highly useful visualization devices for aiding the interpretation of the aggregate opinion map for a video, especially the one aggregated with a large number of individual opinion maps. First, on a mouse-hover event,[12] the teacher can see an opinion histogram for the second of time in consideration. In this histogram, the X-axis represents the frequency of opinions and the Y-axis borrows the opinion value dimension from the background opinion map. This can be important as the Tukey Box$^+$ plot shows only the five main summary values and gives no information on the frequencies of the rest of the *N* opinion values.

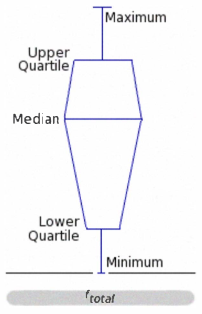

Figure 2. In a Tukey Box$^+$ plot, with the sampling process guaranteeing the absence of outliers, the five-number summary lines are drawn in proportion to their frequencies. Additionally, a line proportional to the total sample count, f_{total}, is partly overlaid on the minimum value. This obviates the need for an independent histogram by helping pack even more information about these crucial parameters *while also* retaining the aesthetics and most of the semantics of the original Tukey Box plot.

The second visualization device that we employ in the teacher console is the *Best* and *Worst Opinion Lines*. The best opinion line, shown in green in Fig. 1(right), is a line that spans the largest continuous extent of time during which the total opinion was the maximum in the opinion map. Likewise, the worst opinion line, shown in maroon, represents the largest continuous extent of minimum total opinion. Since there may be multiple such extents with identical values for maximum and minimum total opinions, there may be multiple instances of best and worst opinion lines present in the console. For ease of use, especially for long-lengthed videos, the teacher is allowed to quickly jump between these lines instead of having to painfully scroll a pageful of time-span at a time.

The third visualization device offered by the teacher console is Tukey Box$^+$ plotting and opinion histogramming for *arbitrary* spans of time. The teacher is allowed to select multiple, noncontinuous time-spans with simple drag actions of the mouse[13]. A subsequent mouse-hover over any of these individual, drag-selected time-spans yields the Tukey Box$^+$ plot (or, the opinion histogram[14]) for the entire set of drag-selected time-spans.[15] This device empowers the teacher to analyze any 'dubious' or 'interesting' regions that earlier were either too fine or too coarse or were arbitrarily spread out to admit of an easy analysis.

C. The Tukey Box$^+$ Plot

To recall, a Tukey Box plot [2][3] is a visualization device that is used for *"graphically depicting groups of numerical data through their five-number summaries: the smallest observation (sample minimum), lower quartile (Q1), median (Q2), upper quartile (Q3), and largest observation (sample maximum).* [6]" Also known as a box-and-whisker plot, in some of its variations [4], instead of displaying the minimum and maximum at its two whisker-ends, it can, for example, show the 9th and the 91st percentiles, respectively. In specifically these variations, any outliers in the dataset tend to get shown very elegantly.

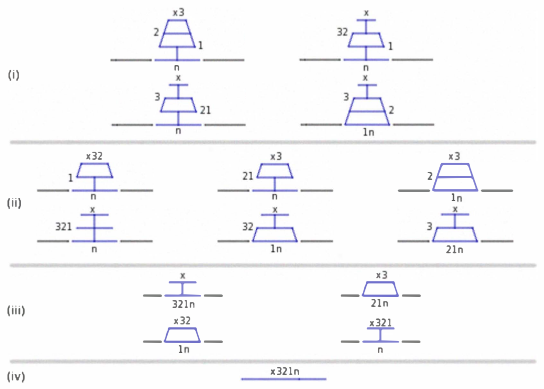

Figure 3. Some of the degenerate cases of a Tukey Box$^+$ plot. The maximum, upper quartile, median, lower quartile, and minimum parameters are denoted by *x*, *3*, *2*, *1*, and *n*, respectively.

However, in situations where sample values can be guaranteed to be sound by virtue of their being delimited, by a process, to a well-known range, outliers cannot be deemed to be present; at least, they cannot be present at the 'physical,' sampling level. This allows for the maximum and minimum values to be justifiably used as the whisker-ends.

[11] Consisting of the minimum, lower quartile, median, upper quartile, and maximum values. Because the opinions scores are based on an ordinal scale, we less-strictly select the $\lfloor n/4 \rfloor$th, $\lfloor n/2 \rfloor$th, $\lfloor 3n/4 \rfloor$th samples as the lower quartile, median, and upper quartile values, respectively, regardless of whether the size of the dataset is even or odd.

[12] along with a *Control+Shift* key combination

[13] along with a *Control* key combination

[14] with an appropriate key combination

[15] with an appropriate key combination

Now, once we do decide to show the maximum and minimum values at the whisker-ends of a Tukey Box plot, the question that arises is: How frequently did these extreme values occur in the dataset? As we argued earlier, physical outliers can easily be ruled out via the use of an explicit process. But what about 'logical' outliers... that have entered the dataset legally but still happen to be highly infrequent relative to the rest of their sibling samples? Failing to present their frequencies may be a nonissue in certain situations, it may be grossly misleading in many others! In our case, for example, if a video has been in use over a period of time, and if a portion of its opinion map shows an unusually high (or, an unusually low) opinion aggregated over this period of time, the teacher would have no way of knowing the number of students behind that opinion within a typical Tukey Box plot.

Among the variations of the Tukey Box plot that come closest in their ability to displaying the frequency component of the summary data are the histplot and the vaseplot [3]. However, both of these boxplots ignore the frequencies of the sample maximum and the sample minimum. Further, they give no indication of the size of the underlying dataset.

Another classic device that could perhaps be employed is the histogram. However, a histogram is typically employed to show the frequency distribution for *all* of the sample values, whereas we are interested in seeing only a subset of them. Further, even a histogram would give no easy indication of the size of the underlying dataset.

We, therefore, propose a small but powerful (and even an aesthetic) extension to the Tukey Box plot called the *Tukey Box$^+$ plot*. In a Tukey Box$^+$ plot (Fig. 2), instead of presenting the horizontal lines via an arbitrarily-chosen constant length merely for namesake, we make these line lengths proportional to the frequencies of the sample values they represent, similar to the upper quartile, median, and lower quartile lines of a histplot. Further, we show the size of the underlying dataset via a proportionally long line that is collinear with the sample minimum line and symmetrically positioned about the central, vertical whiskers. Because the size of the dataset, f_{total}, would, in most typical of the cases, be greater than the frequency of the sample minimum, we introduce a small gap on either side of the latter to prevent its complete occlusion by a longer, collinear presence.

Although degenerate[16] cases such as the ones shown in Fig. 3 would be very rare in practice, we nevertheless propose, for the sake of completeness, a simple scheme that employs textual monikers to highlight the presence of any and all occluded frequency lines: we denote the sample maximum, upper quartile, median, lower quartile, and minimum by *x*, *3*, *2*, *1*, and *n*, respectively.

D. Internals

In this section, we briefly cover the key elements of one reference implementation of the visualization system proposed in this paper. This should also give an indirect idea of some of the difficulties we encountered in this exercise.

On the client side, we have a browser that loads an HTML page with an embedded video player. Along with this player gets loaded an applet with the appropriate console for the student or the teacher. Depending on the settings, the player may either cue up the video requiring a manual or explicit play by the user, or it may start an autoplay as soon as it has received enough video bits. In either case, the player[17] fires a 'Ready' event. This gives any interfacing code a chance to initialize itself in whatever way it deems fit. In our interfacing Javascript code, we use this event to initialize the applet with the time duration of the video; the applet internally uses this to initialize and allocate the necessary view- and model-related[18] structures it would need later once the video starts playing and the user interacting.

The Ready event is followed by a 'Playing' event, signifying that the player is playing the video. Likewise, there is a 'Paused' event that is fired when the user manually pauses the video, and a 'Finished' event that fires (on its own) when the video has played its entire length. The player interfacing code registers handlers with each of these events which notify the applet in turn. For example, when the player is paused, the applet's visual temporal cursor needs to stop scrolling too, and when the video finishes, the applet needs to send its collected opinion map to the interfacing code, so that it can be further posted to the server for aggregation.

We would like to note a few subtleties associated with the player and its event model. First, the player broadcasts these events on a blocking thread! Had the player exposed an asynchronous mechanism for the same, the interfacing code would have enjoyed a little more freedom to, say, queue up the events and act upon them at its earliest leisure.[19] However, as things stand, the programmer must now be as efficient as possible in her event handling logic; a carelessly coded handler can easily block the player long enough to (perceptually) degrade the playback quality.

Second, due to player idiosyncrasies, not all of them may allow separate handlers for their events. They, for example, may have only one 'State Changed' event (with perhaps additional arguments describing the new state) forcing the interfacing code to put all event-handling logic in a single handler! Some may have more states, some may have less, and some may come with subtly differing semantics! We elected to hide all such idiosyncrasies (to the extent possible) behind a thin, wrapper layer of our own, allowing for a uniform API to be exposed to the rest of the interfacing code. When supporting a new player, the interfacing code need only include a player-specific 'binding' of this API and write player-agnostic code instead of code that is player-aware. A detailed description of this wrapper layer and all the

[16] in which two or more members of the five-member Tukey Box+ plot set have identical values

[17] We have chosen to consider only two of the relatively most popular players in this paper: the YouTube Flash player and the Flowplayer Flash player.

[18] Again, using the MVC terminology. Note that the MVC design pattern can occur at many levels within a software system. Even though an applet is a view from one standpoint, being a sizable piece of code, it too, just like the server-side code, can benefit from using MVC in its internal design.

[19] This would have also made the interfacing code more complex, and perhaps this is why the player developers decided to expose a simpler, blocking API.

reasoning that went into its design and implementation is simply beyond the scope of this paper, we can only defer it to a future work.

Third, if the player tag in the HTML happens to be earlier than the applet tag, the player may win the 'initialization race' before the applet and, thus, fire its Ready event before the applet is even ready to catch it... resulting in the event getting lost! This would never be a problem if the applet tag were to be placed before the player tag *and* if the browser were running on a single-core CPU with a single-threaded page rendering logic. To this end, the player interfacing code must be able to defend itself against this nondeterminism in the player- and applet-readiness.

A common feature of the players, it seems, is that they do not notify the interfacing code of the current time (in the video) being played, say, via an event handler; the interfacing code rather must poll the player at an interval it deems fit. Once again, because the name, the syntax, and the semantics of this polling function can vary between players, our wrapper layer implements a polling service and exposes itself via an 'On Poll' event for which the interfacing code could register a handler. We, for example, employ this handler to receive subsecond notifications of the current player time, which are then fed to the applet. The applet uses the current player time mainly to reckon the part to which the user opinion applies. Further, because the applet always has a notion of the current video time, all time-dependent application invariants hold true whether it is a case of progressive downloading, or the user dragging the player scrollbar randomly at whim.

The teacher console, unconnected with the player as such, needs to display the current aggregate opinion map for the video. It does this periodically by first querying if an update is available from the server, and, if it is, then fetching the actual aggregate opinion map. This is implemented via a 'token' system wherein every opinion map submission from the student(s) results in the server-side copy of this token to be incremented. The teacher console compares this token from its local copy to determine if it must fetch any updates from the server. Note that because this is an application-level feature and not inherently related to players and their idiosyncrasies, we elected to keep this polling separate from the polling that goes on in the wrapper layer of the interfacing code. This completes our discussion of the client-side internals.

On the server side, we have a Tomcat-hosted Java servlet that implements our server-side application.[20] Simply put, this server application receives opinion maps from various students, aggregates them, and presents the current aggregated opinion map to the querying teacher. The RDBMS schema is composed primarily of the following two tables: *video* (*video-id*, *video-title, video-description, update-token*) and *opinion-map-xxx* (*time-instant*, $l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9$). A video is identified by a unique, alpha-numeric identifier, *video-id*. It may also have such supplementary attributes as title, description, date of creation... that are not of much technical consequence here. Every video has a unique row in the *video* table. For a row *xxx* in the video table, there exists a unique table *opinion-map-xxx* containing a record for every integral time instant starting with 1. Thus, if the video V is of duration D seconds, there would exist a table *opinion-map-V* with $\lfloor D \rfloor$ rows in it, whose *time-instant* column would have the values $\{1.. \lfloor D \rfloor\}$. The columns $l_1..l_9$ correspond to the opinion levels [-4, +4], with each l_i indicating the total number of times the opinion level occurred for a time-instant in the video. The *opinion-map-xxx* table gets created dynamically at the time the first opinion map gets posted to the server. Because most RDBMSs support a very large number of tables,[21] having a table per video leads to faster response times compared to a design that has all rows of all videos existing in a single table.[22]

In our relational schema, we store in the RDBMS only minimal data necessary to render the client-side consoles functional. While we could have apportioned some of the graphical calculations from the client-side to the server-side, we believe that these calculations are not all that intensive to require assistance from server. Further, by offloading the graphical calculations from the server and without significantly straining the intervening bandwidth any more, the client workstations very naturally help the system scale out.

V. Known Limitations And Future Directions

In this section, we discuss, in no particular order, some known limitations, both in the ideas proposed in this paper as well as in the reference implementation. None of these limitations is anywhere near drastic. Wherever possible, we suggest possible alternatives and directions an alternate implementation can elect to take.

Boxplots do their job of presenting summary information remarkably well. However, they can be deceptive [8] too when the samples in the dataset happen to be distributed into two clear 'lumps' instead of a 'typical' stretch. While a Bee Swarm plot [8] can better capture this scenario, due to potentially a large number of samples that would need to be drawn individually[23], we chose not to use bee swarms in our solution. Instead, we elected to address this problem via opinion histograms which we believe can more than adequately reveal the relative frequencies of the samples.

Though we propose N-level opinion maps, with a fine temporal granularity (of 1 second) it may be impractical to have more than 9 levels at this fine a granularity. Assuming it takes an average of 1 second for a user to give an opinion on a 9-point scale,[24] it appears that the only way we can support a richer range of opinions is by decreasing the temporal granularity of the feedback. The formula appears to be *OpinionGranularity / TemporalGranularity* = K, K being a constant. Thus, one could be improved only at the expense

[20] Because our needs are simple enough, we did not wish to increase serverside complexity by employing a full-fledged application server such as JBoss.

[21] MySQL 6.x, for example, supports up to two billion tables.

[22] even with row-level locking

[23] a computationally intensive affair

[24] either via the mouse of via the keyboard interface

of the other. In our reference implementation, because we have satisfactorily tried *9 opinionlevels / 1 second*, we would suggest '9 opinion levels per second' as the value of *K*. Thus, to be able to support, say, 100 opinion levels, we would need to limit the temporal granularity to *100 / 9 = 11 seconds* (approx). This topic appears to be beyond the scope of this paper, and could be a candidate for an independent study.

We elected to go with automatic submission of the opinion map on the completion of a video play. The other alternative, of course, is to have an explicit 'nag-dialog' that would first confirm the act with the student, part as a convenience and part as a courtesy: convenience, because the student can edit her opinion map before its final submission, and courtesy, in low-bandwidth, pricey Internet connections. However, as we just revealed our opinion on these dialogs, they would tend to nag most users. Secondly, good-bandwidth and cheap Internet connections are the two central assumptions with which we start out in this paper. Finally, the transparent submission (see Section III) of the opinion map helps keep track of hard or low-level statistics. We intuit that this is relatively a more useful approach. It should also be noted that this issue is easily addressable via a site-wide configuration parameter, allowing each site to decide what works best for its needs.

In our reference implementation, we do not allow for multiple temporal granularities and multiple opinion levels within the same instance of a system deployed on a site. That is, *all* videos served off a site would have a 1 second temporal granularity and 9 opinion-levels. To our knowledge, other than slightly more sophisticated core logic and database schema, there stand no fundamental technical limitations in achieving a per-video configuration of these parameters. What we are also ignoring here is the additional work required (i) by page designers for serving custom opinion map consoles per video, and (ii) by site administrators for the maintenance of this relatively more sophisticated schema. More work certainly, but not impossible.

In our user consoles, we show only a limited time-span of 60 seconds[25] instead of the full video duration. While there were other alternatives such as a semilogarithmic scale, fish-eye views, we chose to go ahead with the viewport-window approach of classical Computer Graphics. We intuit here that though a semilogarithmic scale in the teacher console can display values orders of magnitude apart, it would completely ignore the intervening values which happen to be just as important in our use-case as those that do get considered. Though, a fish-eye view would fare much better than a semilogarithmic scale, even here it would be impossible to pack up, say, an hour-long video (3600 seconds) into a console 400 or so pixels wide.[26] A quick calculation reveals that this comes to *3600 seconds / 400 pixels = 9 seconds* of time-span per pixel. Which means that even if the user has a fine grain control over her hand movements and can choose to dexterously drag the mouse left or right 1 pixel at a time every single time, this would still result in 8 seconds worth of time-span (per pixel movement) remaining intractable!

The student and teacher consoles currently provide a disjoint set of features. It could be argued equally well that providing some of the teacher console features[27] to the student may equip the student with advanced query capabilities which may possibly help her in distributed group learning settings in addition to enhancing her overall user experience.

The teacher console could perhaps support more sophisticated queries with arbitrary predicates.

We have not carried out any space-speed tradeoff studies in the use of an RDBMS versus a file-based, custom persistence mechanism on the server side. Here, we merely intuit that a carefully designed and implemented version of the latter would yield orders of magnitude more space as well as speed benefits. Because we do not anticipate too many *ad hoc* queries, we see no major benefit of employing a general-purpose, relational overkill and unnecessarily sacrifice space and speed gains that would otherwise come with a custom persistence approach.

In our reference implementation, we are performing neither raw, payload-level compression nor any at the application level. We easily see a good scope for employing either one of these to avoid bandwidth wastage.

One useful feature to support in the teacher console would be the 'pinning' of the Tukey Box$^+$ plot display either to a single second, or to a single drag-selected time-span, or to a set of them. The Tukey Box$^+$ plot could then display changes in *real-time* as opinion maps from students reach the server… again, in real-time! This would also reinstate one of the original uses of the Tukey Box plot in our system, *viz.* of showing juxtaposed boxplot histories of the video segment in question. The teacher could, for example, use this feature to limit her analysis to a part and not the whole of the video.

Another extremely useful feature to support would be 'tagging' of opinion maps. Here, the teacher of a course may decide to hold the opinion maps being posted after a particular date and time separately from any and all previously aggregated ones and mark them all with a single, mnemonic label or tag. Multiple such tags could be created, each being stored in its own 'holding area' on the server. For the sake of feature completeness, operations allowing the teacher to rename, copy, merge, and remove these tagged opinion maps would also be supported. One excellent use-case of this feature would be when the teacher, having seen a flaw in the original video, has issued an additional instruction, addendum, or errata via means external to and independent of this video and wishes to track opinions subsequently received separately from the previously stored ones. Another companion operation in this theme could be one that allows the teacher to reset the opinion maps in one or more segments of the video to either 0 (the 'normal' value) or to any other allowed value.

[25] a site-wide configurable parameter in our reference implementation

[26] A reasonable choice that is neither too small nor too big. For aesthetic reasons, most site administrators would make the console width identical to the player width.

[27] except for ones that mutate any server-side state

The server side of our reference implementation could benefit from a custom cache that prevents a relatively expensive database lookup if the information it needs can be served from the cache. Since the core of our relational schema is very small (a mere two tables) *and* also since there is no use-case for *ad hoc* database queries, the RDBMS-based implementation of ours could be considered as an overkill with the system benefiting more from a custom, file-based persistence solution of even an average sophistication.

The client-side API designed and implemented may not fully support other Flash players. While we have done our best to keep our API generic and simple, there can (and will) always be differences in the base API exposed by players. The best we could do is go the iterative way, which we already have in this paper.

Acknowledgments

This paper and the accompanying reference implementation of the software derive an unquantifiable but significant amount of inspiration and energy from Sri Mata Amritanandamayi Devi, the Chancellor of Amrita University. The work has been funded by the Ministry of Human Resource Development (MHRD) through the NMEICT ERP project, India.

The authors are thankful to Prema Nedungadi, Pradeep Achan, Rajitha Nair, and Dr P Venkat Rangan of Amrita University for reviewing this paper and for providing their valuable inputs on the initial few revisions of the reference implementation.

Conclusion

In this paper, we make the following contributions. We solve, for the first time, the problem of providing real-time feedback on a video with a fine temporal granularity via Opinion Maps. An Opinion Map enables a teacher or a content author to see, at a glance, a per-second aggregated mean opinion score of a video over a period of calendar time. It allows a very simple, interactive exploration of *extended* boxplot summaries and histograms of opinion scores for arbitrarily long segments of the video. Lastly, it highlights the overall 'best' and 'worst' segments in the video. This approach enables a student to convey and a teacher to know — without any manual and explicit devices of feedback questionnaires, and *ad hoc* and unstructured email — which segments of the video are good and which need more improvement. Because the solution is nonintrusive, it requires no special content-authoring or content-editing processes and, thus, renders massive amounts of legacy digital video content over the Internet 'instantly' ready for opinion mapping.

We extend the Tukey Box plot technique of information visualization by graphically incorporating the frequency components of the five summary parameters along with the population size in the same diagram, and by suggesting a text-assisted scheme to handle any degenerate cases. To avoid inventing a completely new name and to also give sufficient credit to its *original* inventor, we have elected to call this new information visualization device a Tukey Box$^+$ plot.

To our knowledge, no known solution exists for the problem considered at length in this paper. While the solution has been conceived, designed, and implemented in the context of digital video content, it should be possible to easily extend it to digital audio content as well. In Section V, we identify a fairly elaborate list of future directions this work could take... things that we either did not or could not address under the current scope of work.

References

[1] Zhang Guo-Qiang; Yang Qing-Feng; Cheng Su-Qi; Zhou Tao, *"Evolution of the Internet and its Cores."*, New Journal of Physics, Vol 10, (2008)

[2] John W. Tukey, *"Exploratory Data Analysis"*, Addison-Wesley, Reading MA, 1977, ISBN-13: 978-0201076165

[3] Yoav Benjamini, *"Opening the Box of a Boxplot"*, The American Statistician, Vol 42 (4), 257-262, (November 1988)

[4] Michael Frigge; David C. Hoaglin; and Boris Iglewicz, *"Some Implementations of the Boxplot"*, The American Statistician, Vol. 43 (1), 50-54, (February 1989)

[5] Peter J. Rousseeuw; Ida Ruts; and John W. Tukey, *"The Bagplot: A Bivariate Boxplot"*, The American Statistician, Vol 53 (4), 382-387, (November 1999)

[6] World Wide Web, *"Tukey Box Plot"*, Wikipedia, http://en.wikipedia.org/wiki/Tukey box plot

[7] Lisa Zyga, *"Internet Growth Follows Moore's LawToo."*, PHYSORG.com, http://www.physorg.com/news151162452.html, (2009)

[8] World Wide Web, *"Box Plot Distribution"*, Box Plot: Display of Distribution, http://www.physics.csbsju.edu/stats/box2.html