

Videowner

Fingerprint Based Digital Video Comparison

Paul Pucciarelli
Computer Science Honors Project
University of North Carolina at Chapel Hill
Supervised by Professor Diane Pozefsky

Introduction of Problem

The emergence and rapid expansion of digital video throughout the web has opened a largely uncontrolled medium. An era still in its infancy, the age of YouTube, podcasts, and online television is outpacing regulation. As such, copyright holders and web developers are struggling to maintain control over video content. Many entities are working to create tools to track and identify videos. Such technologies are necessary to classify, search, and protect the content of digital videos.

One specific problem is that of copyright violation. Companies like YouTube (Google) are under pressure from copyright holders to keep protected content off the web. The current system employed by YouTube is passive detection. A copyright holder can find offending videos and file a request for their removal from the site. However, this has brought into the legal limelight an important question: With whom does the burden of upholding copyright law fall? (*Viacom v. Google*) Although this question remains largely unanswered, clearly active prevention of copyright violation would benefit all parties involved. Thus there has been much work done to develop tools that will identify if a digital video contains copyrighted material.

Existing Solutions

With so much money at stake, many serious competitors have begun development of copyright detection systems. The arena has many entrants, all of whom tout their proprietary video identification systems. Among the more well known entities involved are:

- **Philips Content Identification**

“Video fingerprinting provides a digital ‘fingerprint’ of a video file by deriving unique features that can be used to identify the video content by comparing it with reference fingerprints stored in a central database.”¹

- **GUBA**

“GUBA is filtering movies and TV shows using a proprietary technology, codenamed ‘Johnny.’ Johnny analyzes video in digitized form and generates a unique fingerprint for each video. Once Johnny has scanned a video, that video is blocked from illegal file trading or distribution on GUBA’s site. GUBA plans to make Johnny available to other video sharing services to help eliminate copyright infringement on the Web... Johnny can identify a video, even if that video has been modified, cropped, reformatted, re-encoded or reposted.”²

- **Attributor**

“Attributor scans billions of web sites, blogs and social networks on a continuous basis to find copies of your content across the web... We will soon be offering image and video monitoring services.”³

- **MotionDSP**

“Ikena Copyright uses a (patent-pending) component of MotionDSP’s military-grade video enhancement technology first announced in October, 2006. As opposed to audio-matching methods, MotionDSP’s unique technology creates a ‘video fingerprint’ by tracking the motion characteristics the video.

Ikena Copyright’s video signature system stands up to many ‘attacks’ such as:

- Editing: it can identify clips as small as 20 seconds
- Low-bitrate compression: it can identify clips that have been re-encoded (i.e.: from their native format to Adobe Flash internet video)
- Aspect ratio change: 16:9 content compressed into a 4:3 display, or 4:3 stretched to 16:9
- Cropping: cropping the sides of a video
- Video quality change: Ikena Copyright can identify videos which have been converted to B+W, or had their color distorted

Ikena Copyright has been designed to scale to a very large database of content, operating on a high volume of matching (i.e.: a database of thousands of hours of copyrighted video, matching tens of thousands of uploaded videos per day).”⁴

- **MySpace & Audible Magic**

“MySpace, the world’s leading lifestyle portal, today announced the launch of Take Down Stay Down™, an innovative new feature for copyright holders that prevents users from re-posting video content in the MySpace community after that content has been removed at the request of the copyright owner... MySpace creates a digital fingerprint of the video content and adds it to its copyright filter, which is based on industry-leading Audible Magic technology.”⁵

- **YouTube Video Identification**

“YouTube Video Identification will help copyright holders identify their works on YouTube. We have worked with Google to develop one-of-a-kind technology that can recognize videos based on a variety of factors. As its Beta status indicates, our Video Identification is brand-new, cutting-edge stuff, so we will be constantly refining and improving it.”⁶

Many of these technologies are in or approaching their BETA stages, which is an indication of the immaturity of the field of video fingerprinting. The reason for this project is the apparent lack of open-source solutions for the problem of video identification.

In addition, note how nearly every system contains the same general components:

- An algorithm that takes a video as input and yields a fingerprint.
- A bit-string fingerprint that supports sub-clip matching, typically by having portions of the string correspond to small time intervals within the video clip.
- A database of video fingerprints to search against.
- An algorithm for comparing the query fingerprint against fingerprints in the database.

It seems this system composition is a natural consequence of the problem. Videos must be fingerprinted and indexed in order to search against them. Algorithms are needed both to distill a video into a fingerprint and to determine the similarity of two or more fingerprints. The fingerprint should be such that it supports sub-clip (part of the whole) matching.

Prior Work

Before the recent video explosion on the web, much of the fingerprinting work related to audio files. Many projects exist to develop and improve acoustic fingerprinting. MusicBrainz is one of the larger open source players; they provide a “metadatabase” that powers automated ID3 tagging of MP3s - given the audio, they provide track information.⁷

Aside from the commercial endeavors mentioned above, there is much academic research taking place on video fingerprinting. Of the sampling of articles examined, there are many approaches used. All of the approaches are complex and most attempt to derive “key frames” or interpret motion vectors. Below are some examples of these complicated approaches.

- **Similarity Measurement and Detection of Video Sequences**

“Efficient technique to detect the similar video sequences on the web has become one of the most important and challenging issues in multimedia and database related areas. In this paper, an original two-phase scheme for video similarity detection is proposed. For each video sequence, we extract two kinds of signatures with different granularities: coarse and fine. Coarse signature is based on the Pyramid Density Histogram technique and fine signature is based on the Nearest Feature Trajectory technique. In the first phase, most of unrelated video data are filtered out with respect to the similarity measure of the coarse signature. In the second phase, the query video example is compared with the results of the first phase according to the similarity measure of the fine signature. Different from the conventional nearest neighbor and Hausdor distance measure methods, our proposed similarity measurement method well incorporates the temporal order of video sequences. Experimental results show that our scheme achieves better quality results than the conventional approaches.”⁸

- **Feature Extraction and a Database Strategy for Video Fingerprinting**

“We approach the concept of fingerprints as an adaptation of cryptographic hashes.”⁹

- **Video Sequence Matching**

“We present a novel scheme to match a video clip against a large database of videos. Unlike previous schemes that match videos based on image similarity, this scheme matches videos based on similarity of temporal activity, i.e., it finds similar “actions”. Furthermore, it provides precise temporal localization of the actions in the matched videos. Video sequences are represented as a sequence of feature vectors called fingerprints. The fingerprint of the query video is matched against the fingerprints of videos in a database using sequential matching. The fingerprints are computed directly from compressed MPEG videos. The matching is much faster

than real-time. We have used this scheme to find similar actions in sporting events, such as diving and baseball.”¹⁰

Approach

The primary objective of this study was to develop the groundwork for a simple open-source video fingerprinting mechanism. It was based on the theory that a complex algorithm may not be necessary. Rather, a fingerprinting algorithm can be elegantly simple since there is no need for the fingerprint to convey any meaningful information about the video.

The concept stemmed from the fact that all digital video boils down to a stream of RGB values over time. Each frame of a video can be decomposed into a mere three values; the average red, green, and blue values for that frame. That is the R, G, and B color channel values of each frame can be summed for every pixel in the frame, and then divided by the total pixel count. Now taking these color channel averages from every frame over the course of the video yields a simple visual summary of that video.

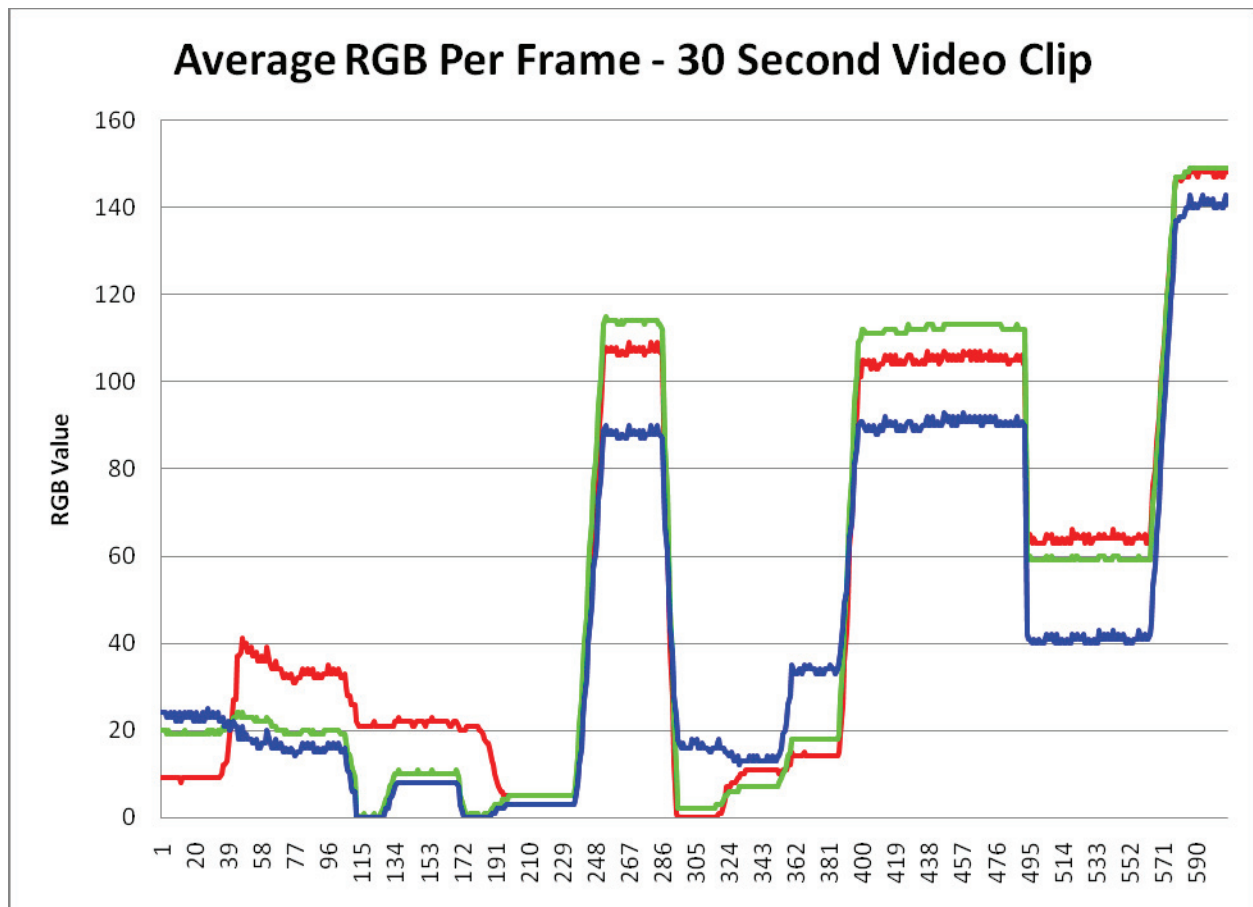


Figure 1. Average values of red, green, and blue channels per frame over the course of a 30 second video clip - NASA.mpeg.

Given the proper codec, we can create such a summary for any video regardless of format. Although there is significant loss of information when converting a video into such a simplified summary, it does provide the basis for a computationally lightweight analysis. Furthermore, the method of trend analysis should provide a robust comparison technique as many video alteration techniques will not alter the fundamental shape of these average RGB trend lines. Sub clips will align with a certain portion of the original; and other modifications like color and brightness changes, aspect ratio changes, and text or logo overlays will not destroy the general shape and critical points of these trend lines.

Fingerprint Creation

Fingerprints are based on two primary criteria: perceived scene changes and average RGB samples. Note that the “scene changes” mentioned here may not always be reflective of true scene changes as a human viewer might determine but rather a more rigid mathematical concept. The start and end time pairings of these “scenes” will serve as an initial matching criteria while average RGB samples at specified intervals will serve as a secondary match confirmation and accuracy criteria.

The fingerprint bit-string is composed as follows:

```
[long] Sampling Rate for Average RGBs
[integer] Number of Scene Changes in Red Channel
[integer] Number of Scene Changes in Green Channel
[integer] Number of Scene Changes in Blue Channel

For each red scene:
    [long, long] Scene Start, Scene End

[long] Flag to Delimit End of Red Scene Listings

For each green scene:
    [long, long] Scene Start, Scene End

[long] Flag to Delimit End of Green Scene Listings

For each blue scene:
    [long, long] Scene Start, Scene End

[long] Flag to Delimit End of Blue Scene Listings

For each sample:
    [integer] Sample RGB Averages
```

Figure 2. Prototype for two part video fingerprint. The “header” contains scene change information while the body is average RGB samples taken at the rate specified in the header.

The sampling rate for the averages is variable (FINGERPRINT_SAMPLE_RATE) but for the purposes of this initial investigation was set at one sample per second. This means that the frames closest to 1 second, 2 seconds, etc. were found and their average RGB's and timestamp were appended to the fingerprint. Additionally, "scene changes" were calculated for each color channel separately. Because we are using an arbitrary mathematical definition for a scene, there is no purpose in attempting to reconcile the different values determined for each color channel. Consequently, we avoid unnecessary complexity by simply allocating to each color channel its own set of scenes. Finally, one should observe that this fingerprint format will yield fingerprints of a very small size. Assuming the number of scene changes remains reasonable, we can consider the scene portion of the fingerprint will be constant. At that point the fingerprint reduces to a 32 bit / sample rate encoding. An hour of video fingerprinted at 1 sample / second will yield a fingerprint size of 14.06 kilobytes. As shown in figure 3, one could fingerprint 10,000 feature films at a more accurate sample rate of 4 samples / second and use less than 2 gigabytes of space.

```
10,000 films * 2 hour average length * 60 minutes / hour * 60 seconds / minute =  
72,000,000 seconds  
  
72,000,000 seconds * 4 samples / second * 32 bits / second = 9,216,000,000 bits  
  
9,216,000,000 bits * 1 byte / 8 bits = 1,152,000,000 bytes  
  
1,152,000,000 bytes * 1 gigabyte / 1,073,741,824 bytes = 1.07 gigabytes  
  
2 GB - 1.07 GB = .93 gigabytes left for scene interval headers
```

Figure 3. Storage requirements for a large collection of fingerprints.

Clearly this technique is a realistic approach when considering storage requirements. Computation requirements will be discussed after the algorithms are introduced.

Fingerprint Creation: Scene Detection

In order to detect a “scene”, we must first define what we will consider a scene to be. We will consider a scene to be those portions of the video where the average RGB value stays within a certain range. Therefore the main objective of this algorithm is to identify portions of relatively stable average RGB values. Essentially, if the channel is flat for certain period of time, then we will consider that flat portion to be a scene.

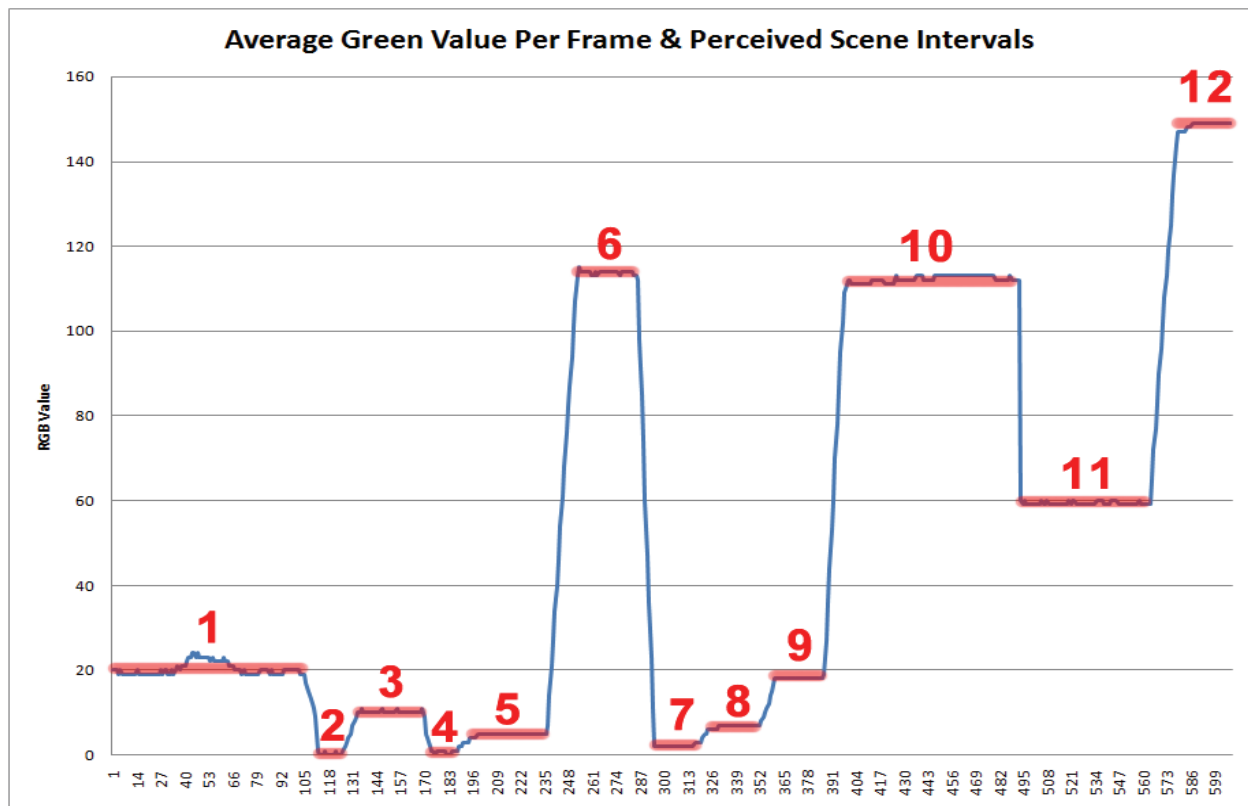


Figure 4. An example of sample intervals the scene detection algorithm would consider scenes.

The technique for scene detection evolved partially out of trial and error. Various concepts were tested involving slope, change in slope, magnitude of value fluctuation, etc. The trials revealed the problem of discerning between in-scene fluctuations and actual scene change transitions. Merely comparing adjacent samples yielded insufficient information to make this determination. The solution was to store a window of samples for a specified time length. When value fluctuations occurred, we could see if the window returned to stability near its previous value (same scene) or moved into another disparate range of values (new scene). The head of the window can traverse rough patches within the scene as long as it eventually returns to the same value range as the window tail (or vice versa). If there is a true scene change, both the head and the tail will eventually depart the flat segment of the trend line. In order to detect these flat segments an algorithm runs on each color channel as follows:

1. Create a window, a collection of RGB samples.
 - a. The window is essentially every sample taken between time A and time B. This time gap is the size of window known as `WINDOW_LENGTH` (measured in seconds).
 - b. The window dynamically resizes itself to maintain this constant time-length.
2. Move the window across the samples from beginning to end.
 - a. As the window moves, examine the samples at the head and tail of the window.
 - i. Once head and tail have values within `WINDOW_TOLERANCE` of each other, log the average of those values as `stabilityLocation`. This begins a condition known as stability (i.e. we believe we're on a flat interval).
 - ii. During stability, either the head or the tail must be within `WINDOW_TOLERANCE` of `stabilityLocation`.
 - iii. If at any time both the head and the tail have values too distant from `stabilityLocation` (beyond the `WINDOW_TOLERANCE`), we are no longer in stability and return to step i.
 - iv. If stability exists for at least `MINIMUM_SCENE_LENGTH`, we consider the current interval a scene and wait for stability to end to determine the ending time of this scene.
 - v. Every scene detected has its start and end times log.

The algorithm as is exhibited the following shortcoming: It would wrongly divide one scene into two. That is short periods of instability should not divide a scene if the values do not significantly change.



Figure 5. One scene that might wrongly be perceived as two.

This problem was addressed the following addition to the algorithm:

3. For each scene logged, decide if it should be joined with the previous scene.
 - a. Is the beginning of this scene within `SCENE_JOIN_TIME_GAP` of the ending of the previous scene?
 - b. Is the `stabilityLocation` value from this scene within `SCENE_JOIN_VALUE_GAP` of the `stabilityLocation` from the previous scene?
 - i. If both conditions are satisfied combine the two scenes into one.

Fingerprint Creation: Average RGB Sampling

While the first half of the fingerprint summarizes scene intervals, the second half is the average RGB samples taken at the interval specified by `FINGERPRINT_SAMPLE_RATE`. The sampling process is very straightforward. Based on the sample rate, we determine at what times we would like samples. For instance, a rate of 4 samples / second means we want samples at 0s, .25s, .50s, .75s, 1.00s, and so on. The first average RGB value becomes the sample at 0 seconds. We then calculate time between values by using the samples' timestamps. We simply use the RGB value nearest to .25 seconds as the sample for that time. We continue using the sample nearest to the desired time until the end of the movie.

Fingerprint Matching

In order to compare 2 videos, one must simply compare their respective fingerprints. In theory one would maintain a database of video fingerprints and query against it with a “query fingerprint.” Fingerprint matching consists of two steps: scene information comparison and average RGB sample comparison. The intended benefit of the scene interval header found within the fingerprint is to avoid average RGB sample comparison when the scene intervals of the two videos are clearly incongruent. We can theoretically use the fingerprint header to rule out obvious negative match results. This significantly speeds up the time required to search through a database of fingerprints.

Fingerprint Matching: Scene Comparison

The scene comparison algorithm and RGB sample comparison algorithm are actually quite similar. They both boil down to the longest common substring problem. They also both return individual results for each color channel. I will reserve judgment in determining how many channels are needed to qualify as a match.

Scene comparison steps through the three scene interval summaries for each channel. For each channel it determines a `sceneMatchScore` as follows:

1. Compare the scene counts of the two fingerprints.
 - a. The fingerprint with fewer scenes becomes the “little” fingerprint.
 - b. The fingerprint with more scenes becomes the “big” fingerprint.
2. For each scene length in the little fingerprint:
 - a. For each scene length in the big fingerprint:
 - i. If the difference between the two scene lengths is less than `FINGERPRINT_SCENE_INTERVAL_TOLERANCE`, we have a match.
 - ii. If there’s a match, compare the next two scene lengths.
 - iii. Continue doing this until we find two scene lengths that don’t match (or until the end of the scenes).
 - iv. Log this substring length.
 - v. If substring length is the longest found so far, log it as the `maxSceneMatchCount`.
3. Now we have the instance where the most consecutive scene lengths in the little fingerprint matched up with consecutive scene lengths in the big fingerprint.
4. Compute the `sceneMatchScore`:

$$\text{SceneMatchScore} = .5 * (\text{maxSceneMatchCount} / \text{total \# of scenes in little fingerprint}) + .5 * (\text{percent accuracy of matches})$$

To get the “percent accuracy of matches”, we determine a percent error for each scene length match.
 Percent error = total difference of matched scene lengths / average of matched scene lengths
 Percent accuracy of matches = 1 - (average percent error)

The `sceneMatchScore` that results from the scene comparison algorithm is one part the amount of scenes matched and one part the accuracy of those matches. Note that we actually derive `rSceneMatchScore`, `gSceneMatchScore`, and `bSceneMatchScore` by running the algorithm for each color channel’s scene interval summary.

Fingerprint Matching: Average RGB Sample Comparison

Comparing the average RGB samples of the fingerprints is another instance of longest common substring determination. Although the true algorithm runs through the two fingerprints once and compares all three color channels separately at each

sample, the algorithm summary will just handle one channel. The other two channels are compared in the same manner.

1. Compare the sample counts of the two fingerprints.
 - a. The fingerprint with fewer samples becomes the “little” fingerprint.
 - b. The fingerprint with more samples becomes the “big” fingerprint.
2. For each RGB sample in the little fingerprint:
 - a. For each RGB sample in the big fingerprint:
 - i. Do the two R samples match?
 1. A match occurs if the difference between the R values is less than `FINGERPRINT_SAMPLE_MATCH_TOLERANCE`.
 2. A match can also occur if the little R value is between the current big R value and the previous big R value. In this way a match can still occur on rapid changes of R.
 - ii. If the samples match, compare the next two samples.
 - iii. Continue doing this until we find two samples that don’t match (or until the end of the samples).
 1. We actually wait until multiple consecutive samples fail to match. Only until samples remain different for `FINGERPRINT_SAMPLE_TIME_MISS_TOLERANCE` do we consider the substring to have ended.
 - iv. Log this substring length.
 - v. If substring length is the longest found so far, log it as the `maxRMatch`.
3. Now we have the instance where the most consecutive R samples in the little fingerprint matched up with consecutive R samples in the big fingerprint.
4. Compute the `rSampleMatchScore`:

`rSampleMatchScore = maxRMatch / total # of samples`

In practice, the algorithm yields one sample match score for each channel. This score is just the percent of the smaller video found within the larger video. Although false matches are possible, it is quite unlikely that a video’s three color channels will vary along the same values as another video.

Algorithm Summary

The idea that the algorithms should be as simple as possible is aimed at keeping the technology fast and robust. The magnitude of the problem it addresses mandates computational efficiency. The process should also be readily adaptable and thus easy to understand.

The scene detection algorithm makes three passes over the sample data and is thus of the Order $(3n)$ or $O(n)$ where n is representative of video length. The scene comparison algorithm and sample comparison algorithms are both longest common substring algorithms. Although this initial investigation used simple inefficient algorithms, dynamic programming has been shown to reduce this problem to $O(nm)$ where n and m are the two string lengths. Let us consider the two fingerprints to be of similar length yielding $O(n^2)$ algorithms for both scene comparison and sample comparison. Keep in mind that only scene comparison will run in a fingerprint comparison of two significantly different fingerprints. We can conclude that the algorithms necessary to implement this system are efficient enough for real world deployments.

Results

What follows are some video analysis scenarios with their corresponding fingerprint match scores.

1. Comparing a video to itself.

Video A: nasa.mpeg

Video B: nasa.mpeg

RGB SCENE MATCH SCORES = 99%, 96%, 97%

RGB SAMPLE MATCH SCORES = 100% 100% 100%

2. Comparing two different videos.

Video A: nasa.mpeg

Video B: videogame.mpg

RGB SCENE MATCH SCORE S = 54%, 58%, 64%

RGB SAMPLE MATCH SCORES = 7%, 7%, 7%

Video A: videogame.mpeg

Video B: groundbreaking.mpg

RGB SCENE MATCH SCORE S = 43%, 49%, 49%

RGB SAMPLE MATCH SCORES = 0%, 0%, 0%

3. Comparing a sub-clip to its original source

Video A: nasa.mpeg

Video B: nasa_short.mpg

RGB SCENE MATCH SCORES = 82%, 83%, 82%

RGB SAMPLE MATCH SCORES = 100%, 100%, 100%

4. Comparing a video to a version with 4x the original brightness

Video A: nasa.mpeg

Video B: nasa_bright.mpg

RGB SCENE MATCH SCORES = 70%, 64%, 50%

RGB SAMPLE MATCH SCORES = 66%, 68%, 68%

Interpretation

The system works consistently enough to yield near 100% matches when the same video is run twice. This is a good indication that different sampling offsets will not have much impact on results. It shows that the system is calibrated accurately.

On two totally different videos, the system yielded a surprisingly high scene match score. This is probably due in part to both videos being short and thus the odds of both having one or two second scenes are very high. Considering the scene length tolerance is .6 seconds, any short scene will match given this margin of error. Luckily, these results were negated by the 7% sample match score, which clearly indicates the videos are different. Additionally, longer videos will yield lower scene match scores.

The sub-clip matching worked nearly perfectly. The scene scores were in the 80% since the first and last scenes in the sub-clip were not cut mid-scene. The 100% sample match on this test shows sub-clip matching will definitely work.

Finally, the brightness increase on the final test was huge. The change destroyed the visual integrity of the video from a human standpoint. Yet scene matching was between 50% and 70% which would be sufficient to trigger sample comparison. The sample match score was about 67%, which is beyond anything that might happen accidentally. Keep in mind this is from an algorithm that makes no attempt to handle brightness changes. It is definitely feasible to develop modifications that would handle linear shifts and/or magnitude changes. For example, the algorithm could recognize that if the RGB values change to be $R+x$, $G+y$, $B+z$ or xR , yG , zB , these changes would not alter the fundamental patterns created by

the average RGB samples. The modified samples would still increase, decrease, and stabilize during the same time intervals as the original.

Evaluation

This initial investigation can be considered a success for the proposed concept. The first round of testing has proven that average RGB samples can indeed provide a relatively unique summary of a video. It is a straightforward concept that makes sense conceptually and as we have just demonstrated, in practice. We believe developers could build upon this foundation to improve upon the accuracy and robustness of the current prototype. The accuracy of the scene matching algorithm seems to need some improvement. However, the lengths of scenes were never intended as a definitive video summary. Rather, the only requirement for this component was that it allowed us to separate the wheat from the chaff. Its purpose is merely to provide criteria for making the first round of cuts during a video search. While the scene matching may suffer from false positives, it will not yield false negatives. Consequently, we can conclude that the use of this information as a header within the video fingerprint is worth serious consideration. All in all, the initial prototype functions well enough to warrant further investigation.

Strengths and Weaknesses of the System

If this system were to be used for real world applications, there are certain strengths and weaknesses that would need to be considered. Some of these issues are

necessarily inherent in the design as a whole while others may be fixed with additional code.

Strengths

- + The system can match sub-clips assuming the original was fingerprinted in its entirety.
- + The fingerprint represents the video stream allowing various parts to link directly to different portions of the movie.
- + Destroying the integrity of one color channel will not undermine the system.

Potential System Enhancements

- The system could be modified to handle brightness adjustments.
- The system could be modified to handle the introduction of dead pixel zones (in the event a logo, watermark, or text is overlayed on a video). The average RGB values would still follow the same patterns of change, just to less extreme magnitudes.
- The system may be capable of withstanding a change in aspect ratio as the average RGB values will still follow similar change patterns.
- Any change to color balance that would not destroy human viewability would most likely not change the general pattern followed by the RGB averages.

Weaknesses

- The system cannot readily deal with a video that is significantly cropped.
- The scene definition is arbitrary and may not function well on videos with constant change (fast paced music videos for example).

Recommendations

The methods proposed here should be further examined. If they provide accurate results on a larger scale of testing, this prototype should be expanded.

There is room for various improvements, including but not limited to:

- Support comparing two different sample rates.
- Dynamic sample rate determination based on video length (faster rates for shorter videos).
- Incorporation of audio as an additional information channel.

While the demand for automated video identification tools continues to increase, the ideas presented here will continue to undergo testing and improvement until the true capabilities of this design are exposed and until we have exhausted their potential.

Works Cited

- ¹ "Content Identification." Royal Philips. 5 Dec. 2007
<<http://www.business-sites.philips.com/contentidentification/home/index.html>>.
- ² "GUBA and MPAA Team Up to Crack Down on Movie Piracy." GUBA Press Blog. 20 July 2006. 5 Dec. 2007
<http://blog.guba.com/press/archives/2006/07/guba_and_mpa_a_team_up_to_crack_down_on_movie_piracy.html>.
- ³ "How It Works." Attributor. 5 Dec. 2007 <http://www.attributor.com/how_it_works/overview.php>.
- ⁴ "MotionDSP Announces Ikena Copyright New Video Copyright Detection Technology." MotionDSP. 13 Mar. 2006. 5 Dec. 2007 <<http://www.motiondsp.com/release.php?id=2>>.
- ⁵ "MySpace Launches Take Down Stay Down Copyright Protection." Business Wire. 11 May 2007. 5 Dec. 2007
<http://www.businesswire.com/portal/site/google/index.jsp?ndmViewId=news_view&newsId=20070511005160&newsLang=en>.
- ⁶ "YouTube Video Identification Beta." YouTube. 5 Dec. 2007 <http://www.youtube.com/t/video_id_about>.
- ⁷ "AudioFingerprint." MusicBrainz Wiki. 11 Sep. 2007. 5 Dec. 2007
<<http://wiki.musicbrainz.org/AudioFingerprint>>.
- ⁸ Chu-Hong, Hoi. "Similarity Measurement and Detection of Video Sequences." Department of Computer Science and Engineering, The Chinese University of Hong Kong. 1 April 2003
<<http://www.cs.cuhk.hk/~lyu/student/mphil/steven/term2.pdf>>.
- ⁹ Oostveen, Job; Kalker, Ton & Haitsma, Jaap. "Feature Extraction and a Database Strategy for Video Fingerprinting." Proceedings of the 5th International Conference on Recent Advances in Visual Information Systems. 11 Mar. 2002 <<http://portal.acm.org/citation.cfm?id=647062.714612#>>.
- ¹⁰ Mohan, R. "Video Sequence Matching." Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on Volume 6. 12 May 1998
<<http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel4/5518/14898/00679686.pdf?tp=&isnumber=&arnumber=679686>>.