Preprint

Automatic Video Editing for Sensor-Rich Videos

Wesley Taylor Faisal Z. Qureshi
Faculty of Science, University of Ontario Institute of Technology
Oshawa, ON L1H 7K4 Canada
wesley.taylor3@uoit.net faisal.qureshi@uoit.ca

Abstract

We present a new framework for capturing videos using sensor-rich mobile devices, such as smartphones, tablets, etc. Many of today's mobile devices are equipped with a variety of sensors, including accelerometers, magnetometers and gyroscopes, which are rarely used during video capture for anything more than video stabilization. We demonstrate that these sensors, together with the information that can be extracted from the recorded video via computer vision techniques, provide a rich source of data that can be leveraged to automatically edit and "clean up" the captured video. Sensor data, for example, can be used to identify undesirable video segments that are then hidden from view. We showcase an Android video recording app that captures sensor data during video recording and is capable of automatically constructing final-cuts from the recorded video. The app uses the captured sensor data plus computer vision algorithms, such as focus analysis, face detection, etc., to filter out undesirable segments and keep visually appealing portions of the captured video to create a final cut. We also show how information from various sensors and computer vision routines can be combined to create different final cuts with little or no user input.

1. Introduction

Camera-equipped mobile devices are increasingly common, and these devices have transformed how we interact with our environment and with each other. Our access to camera-equipped mobile devices has increased our ability to capture our lives in images and videos by such a large degree, it is no wonder that the Oxford English Dictionary declared 2013 the year of the *selfie*. If the current trend holds, we will only see an increase in the volume of images and videos captured by individuals using their mobile devices. The sheer volume of images and videos being captured is quickly becoming overwhelming, and we urgently need new techniques for managing, archiving, curating, and presenting the imagery captured by these devices. This paper proposes a framework for summarizing and "cleaning up" raw videos captured through mobile devices with little or no human intervention. Computer vision techniques and raw readings from accelerometer, magnetometer, and other mobile device sensors work in tandem to construct an edited final-cut of any recorded video containing only visually appealing and informative bits of the raw video. Video editing is an extremely tedious process, and this paper proposes a first-of-its-kind system that uses sensor data and image processing for the purpose of automatically editing videos captured using mobile devices.

Existing video summarization schemes to a large extent have focused on visual data, i.e., information that can be extracted from the video through image analysis and computer vision techniques. Chen *et al*. [3], for example, use image analysis to partition a video into segments based upon their "interestingness." Similarly, Hua *et al*. [6] use computer vision techniques to select "desirable" segments from a video and match these to music. Their method also adds transitions between the selected segments to construct a visually pleasing final-cut. Non-visual data—such as accelerometer, magnetometer and gyroscope readings, etc.—is either unavailable for particular videos or is ignored during video summarization using conventional methods. To the best of our knowledge, existing mobile video recording apps do not attempt to save sensor data (from accelerometers, magnetometers, gyroscopes, etc.) during video capture. We present an Android video recording app that saves sensor data in combination with relevant timing information during video capture. This timing information is later used during temporal alignment of video and sensor streams. We also propose a video summarization framework that exploits both visual (i.e., derived from the video stream using image analysis or computer vision techniques) and non-visual (sensor readings from accelerometers, gyroscopes, etc.) data to automatically edit the captured video into a shorter—and hopefully more pleasing—final-cut.

Combining other sources of data with computer vision techniques for video summarization appears beneficial and can lead to significant computational savings—computer

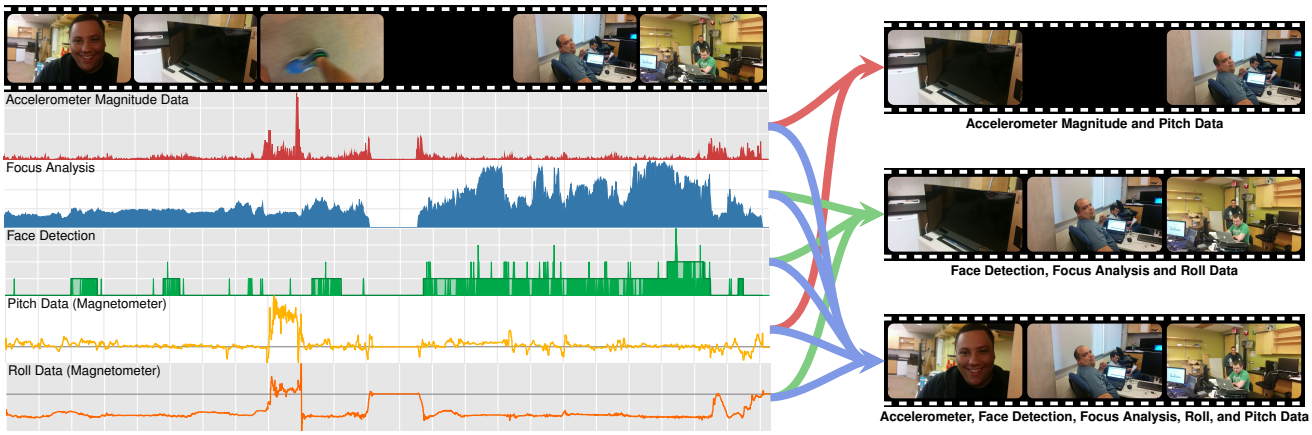

Figure 1. A high-level overview of the capabilities of the proposed system. Frames from the original video, along with time-series plots of both image and sensor data recorded over the course of the video, are shown on the left. The images on the right represent three possible automatically-obtained final-cuts of the same video. These cuts are constructed using different combinations of sensor and visual data streams.

vision techniques typically have high computational requirements. Using computer vision methods to detect camera motion, for example, requires optical flow estimation, while similar results can be obtained at a fraction of the computational cost by using accelerometer data. Computational savings aside, sensor data can also be used to improve both the performance and the accuracy of image analysis routines. Advantages obtained using sensor data readily available from mobile devices make a strong case for exploring video summarization methods that can leverage data available from the many sensors embedded in these devices and image analysis techniques typically used for these purposes. The work presented here is a step in that direction.

1.1. Contributions

We present an Android video recording/summarization app that combines computer vision techniques with accelerometer, magnetometer and gyroscope data for the purposes of video summarization and "clean up." Figure 1 provides a high-level overview of our system. The proposed system is able to extract device context—i.e., its orientation, movement, etc.—from the sensor data that is recorded alongside the video. The sensor readings can be used to segment the recorded video into undesirable and desirable regions. For example, accelerometer data can identify times when the mobile device is undergoing extreme, uncontrolled motion. There is a good chance that any video recorded under these conditions will exhibit motion blur. We can, of course, use computer vision techniques to identify segments exhibiting strong motion blur; however, image-based motion blur analysis is computationally expensive. The availability of sensor data also enables our system to apply computationally costly computer vision analysis to more promising regions of the recorded video.

Our system comprises a number of video segmentation routines capable of splitting a recorded video into segments based upon their desirability. We define desirability loosely as the visual appeal of a video segment, so a video segment with high desirability score has a greater chance of making it into the final cut. Our video segmentation routines use one or more visual and non-visual (sensor) streams during video segmentation. For example, an accelerometer video segmentation routine uses accelerometer readings for video segmentation. Similarly, a magnetometer video segmentation routine relies upon magnetometer sensor data for constructing and ranking video segments. We have also implemented a face detection video segmentation routine that relies upon computer vision techniques to segment and rank video based upon whether or not a face is visible in a particular segment of the video. Video segmentation routines can be easily added or removed from our system, making the proposed system both flexible and extensible. Each video segmentation routine returns a list of video segments ranked according to their desirability. We also propose a segment selection method that selects the top ranked segments to construct the (shorter) final-cut (summary of the video). Our system allows both interactive and fully automated operation. In interactive mode, the list of ranked video proposals is presented to the user, who can decide which of the regions should make it into the final-cut. The app also exposes a fully manual mode where the user is responsible for "editing" the video, manually placing the split points for each desired segment in the final video.

We demonstrate the proposed system, i.e., our Android app (Figure 2), on a variety of videos and show that the system is able to discard "undesirable" regions of the recorded video by combining 1) accelerometer & magnetometer data, 2) gyroscope readings, 3) face detection, 4) focus analysis,

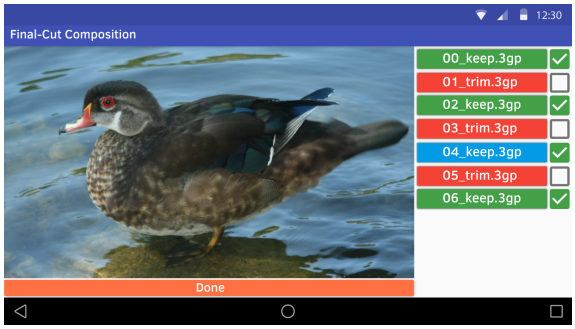

Figure 2. A screenshot of our Android app. The recorded video is segmented into 7 portions (seen on the right). The red segments will be discarded in the the final-cut. The green segments will be used to construct the final-cut. The blue segment is currently being displayed.

and 5) optical flow computation. It is worth bearing in mind that we cannot use our system on pre-recorded videos where the sensor data is not available. The primary focus of this work is using sensor data that is saved at the same time that the video is being recorded, and we were unable to find any dataset suitable for our purposes.

2. Previous Work

Current video summarization schemes mostly rely upon computer vision and image analysis [3, 6]. These approaches typically have high computational requirements, which render these approaches unsuitable for mobile deployment. Otsuka *et al*. [10] presented a method for classifying sports video using audio information. This method avoids the computationally expensive processing associated with image analysis. The primary drawback of their approach is that in general it is not always possible to identify the interesting bits of a video by relying purely on the audio information. Still, this approach shows that it is indeed possible to summarize a video by identifying its interesting bits using non-visual data.

Within the context of mobile devices, Siewiorek *et al*. [16] have shown that it is possible to extract the "context" of a mobile devices—i.e., is it lying face down on a flat surface, is it being used to capture an image, etc.—by using onboard sensors. Similarly, Teng *et al*. [17] are able to determine the context by using only GPS and "shake" sensors available on mobile devices. Others have explored the use of non-visual data for annotation purposes. Shen *et al*. [15] and Zhang *et al*. [20], for example, use GPS data to tag different portions of the video. We currently do not use GPS information. It is, however, straightfoward to add a routine that will use location information for video segmentation and tagging.

Ahanger and Little have studied video composition methods for constructing a video summary by combining video segments [1]. This approach bears resemble to the method presented here. Our method composes the final-cut by selecting from a ranked set of video segments. These video segments are computed by video segmentation routines that use both visual and non-visual data. Unlike previous approaches, the method presented here uses the "context" of a mobile device (as inferred by relying upon onboard sensors), the information value of a video segment (say the existence of a face), and the visual quality of a segment (whether or not it is in focus, etc.) to summarize the recorded video into a final-cut. Our method is both flexible and extensible, and we demonstrate our method on a mobile device.

3. System Overview

Our system is realized as a collection of 1) data processing, 2) video segmentation, and 3) video composition routines. New routines can be easily added to the system to enhance its functionality. Data processing routines operate upon one or more data streams and return a data stream. Video segmentation routines are responsible for dividing the recorded video into non-overlapping segments. The input to video segmentation routines is a video and zero or more data streams. We allow for 0 data streams to account for situations where a video segmentation routine relies solely upon data that can be extracted from the input video stream. Video composition routines are responsible for picking the "best" segments to compose the final-cut. We now describe the five aspects of our system: 1) sensor-rich video recording, 2) sensor stream processing, 3) video processing, 4) segmentation, and 5) final-cut composition.

3.1. Sensor-Rich Video Recording

We have implemented our system on a Google Nexus 5 smartphone, which runs Google's Android operating system. Android has no built-in support for storing both videos and other sensor readings at the same time. Consequently, we wrote custom routines capable of recording this sensor data as videos are being saved. Our method assumes that the video and sensor data is synchronized. The Android API does not support sensor polling, but rather sensors push a stream of data whenever new readings become available. Similarly, there is no way to execute a piece of user code for each frame of the video during capture; videos are encoded on-the-fly and it is not possible to tie an event to "frame capture." Our method is able to achieve frame-level synchronization by relying upon start and end times associated with the recorded sensor streams and video clips.

3.2. Data Processing Routines

These routines deal with readings from sensors, such as accelerometers, magnetometers, gyroscopes, and cameras.

Figure 3. Accelerometer. Raw x, y, and z readings over time. Time is along the horizontal axis.

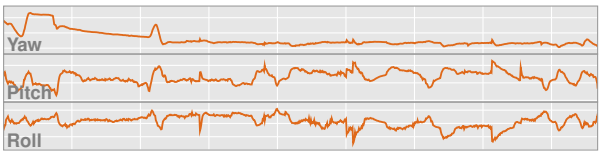

Figure 4. Orientation data. These plots show the raw yaw, pitch, and roll values over time, with time along the horizontal axis.

We assume that different data streams are already temporally aligned (see above). We support a number of common (frequency and time based) time-series analysis operations, such as moving averages estimation, derivative and integration computations, wavelet transforms, and Fourier space decomposition.

3.2.1 Accelerometer Data

Readings from an accelerometer, which measures forces acting on a device, can help us locate portions of videos exhibiting large motions (see Figure 3). These regions are tagged as candidates for removal. Accelerometer readings are also combined with other sensors to determine the orientation of the device with respect to the gravity vector.

3.2.2 Device Orientation and Gravity Vector

Figure 4 shows Roll (R), Pitch (P), and Yaw (Y) values (for the mobile device) calculated using readings from accelerometer, magnetometer, and gyroscope data. RPY values can be used to infer the orientation of the device and the direction of the gravity vector. This data can then detect regions of videos where the device has gone through large orientation changes. These regions sometimes correspond to events, such as placing the mobile device on a flat surface or putting it in one's pocket. Orientation information is also used to determine whether or not a video segment is recorded in portrait or landscape mode.

The Android platform computes RPY values using readings from the accelerometer and magnetometer. We noticed that by using only accelerometer and magnetometer—which measures Earth's magnetic field and acts as a compass—results in jittery estimates for the device's orientation; small forces registered by the accelerometer throw off the computations. An alternate approach is to use readings from a gyroscope, which measures the instantaneous angular velocity of the device, to compute the orientation

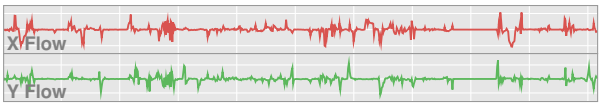

Figure 5. Optical flow data computed over the same time frame as the accelerometer data shown in Figure 3. Optical flow can only be used to compute the x and the y component. Time is along the horizontal axis.

of the device—readings from the gyroscope are integrated over time to estimate the current orientation of the device. This too leads to inaccurate estimates, as small errors at each instant accumulate over time.

We combine readings from accelerometer, magnetometer, and gyroscope to compute the orientation and gravity direction vector of the mobile device [19]. Specifically, we use a *complementary filter* to filter high frequency signal from gyroscope data and filter low frequency component from accelerometer data [14]. A low-pass filter on gyroscope readings accounts for the drift due to integration over time; whereas, a high-pass filter on accelerometer data is able to account for sudden, short-duration forces applied on the mobile device.

3.3. Video Processing

We have implemented the following state-of-the-art computer vision routines for identifying promising segments of the recorded video: 1) frame brightness estimation; 2) optical flow estimation; 3) face detection; 4) salient face recognition; and 5) focus analysis. We now briefly explain the role of these routines in our system. For technical details, we refer the reader to the relevant computer vision literature.

3.3.1 Optical Flow

Optical flow estimation is a first step in many subsequent computer vision routines, such as activity recognition [5]. In our case, it is possible to use optical flow to determine the degree of "shake" of the device during video recording. When using optical flow, however, special care has to be taken to separate camera motion from the movement present in the scene. In our experiments we have observed that accelerometer data is a better indicator of camera movement than optical flow data (average optical flow vector for a given frame). Optical flow data, on the other hand, is better suited to estimate fine-grained scene motion. Figures 5 and 3 show optical flow (x and y) and accelerometer data (x, y and z) data for the same video, respectively.

3.3.2 Face Detection

Face detection is used to identify video portions containing faces. We have implemented both HARR and LBP [2]

Figure 6. Focus analysis results for a video using the method described in [11]. Time is along the horizontal axis.

based face detection routines. Face detection routines tag each frame I_t of the recorded video V with the set of bounding boxes B_t containing faces. The set B_t is empty if no faces are found in I_t. Bounding boxes with areas less than a predefined threshold are also discarded and not considered for B_t.

3.3.3 Focus Analysis

Focus analysis is commonly used to determine the "visual quality" of an image. An image that is completely out-of-focus is typically deemed of inferior quality. On the contrary, images that are in sharp focus or where the main object is in focus are considered aesthetically pleasing. The focus analysis routine assigns a value (between 0 and 1) to each frame I_t of the video. Videos recorded with mobile devices exhibit a variety of visual artifacts, such as extreme motion, increased noise due to the lack of proper lighting, high contrast, sudden changes in brightness, etc. To account for these artifacts, we have implemented a large selection of focus analysis algorithms for our purposes [11, 12, 13, 18] (see Figure 6).

3.3.4 Salient Face Detection

Our system is able to detect salient faces in the video. Video segments containing salient faces are then given more priority when constructing the final-cut. We have implemented an unsupervised salient face detection method that does not rely upon the existence of a face database. The detected faces are encoded as Local Binary Patterns (LPB) [2] and clustered using a combination of OpenCV's face recognition code and hierarchical clustering. Cluster centres represent unique faces seen in the video. The membership size of a cluster is used to identify the salient faces. By default, the number of salient faces is determined automatically based on membership size, but it can be provided as a parameter in the case that a user has prior knowledge as to the expected number of unique clusters. This approach is similar to the one presented in [9]; however, we decided to use LBP instead of SIFT features for encoding faces. LBP, we found, were more robust to brightness changes observed in our videos. The salient face detection routine tags frames I_t (of the recorded video) containing one or more salient faces.

3.4. Video Segmentation

Each data and video processing routine returns an annotation stream, which stores the results of data/video processing for each frame in the video. An orientation stream, for example, will include device orientation data for each frame of the recorded video. A face detection stream, on the other hand, will include the number of faces found in each frame of the video. Each stream is processed separately to construct a non-overlapping partition of the recorded video. Each stream uses a finite state machine to model *friction* when partitioning the recorded video. Video frames are processed in sequence, and stream data is used to decide whether or not a video frame belongs to the current partition; a new partition is initiated if this test fails. Friction solves the problem of constructing a large number of (short-duration) partitions in the case of a noisy source data stream.

To summarize, video segmentation generates a set of non-overlapping partitions, one for data each stream. Each partition defines a segmentation over the recorded video. Each segment is then classified as "undesirable" or "desirable," or assigned a score by using relevant stream information. This information plays an important role during final-cut composition.

3.5. Final-Cut Composition

Video segmentation returns a collection of sets of video segments $\mathcal{S} = \{S_1, S_2, \cdots, S_n\}$, where S_i represents the set of video segments from the i^{th} data/video stream. Each segment $s \in S_i$ is assigned a label "desirable" or "undesirable" or a desirability score. It is easy to pull out subsets of segments from S_i consisting of all s that match a user-defined criteria. For example, it is possible to pick out all video segments from optical flow estimation stream that exhibit small motion. Similarly, we can pick out in-focus segments from the focus stream. The salient face detection stream allows us to select all segments that contain one or more salient faces.

Set operations, like union and intersection, can be used to compose the final-cut given a collection of sets of video segments that meet the user-defined criteria. Our Android app supports three modes of operation. In the completely automated mode, the system is able to compose the final-cut by selecting highly ranked (or classified as "desirable") segments from $\mathcal{S}$. In the semi-automated mode, the system presents the list of segments along with their classification or scores to the user. In this case the user decides which of the segments should make it to the final-cut. It is possible to restrict the list of video segments presented to the user based upon the streams used to construct those segments. For example, a user may only wish to use face detection and focus streams for final-cut composition. In that case, the user can choose to view segments constructed from face detection and focus routines. The last and fully manual mode requires the user to identify the partitions, construct the segments, and select the segments for the final-cut.

4. Results and Evaluation

The primary hypothesis of this work is that non-visual information recorded alongside a video can be exploited to construct a shorter, cleaned-up final-cut of the video. As mentioned previously, existing video summarization benchmarks are not suitable for our purposes, since these datasets do not contain readings from sensors (found in most of today's mobile devices), such as accelerometer, magnetometer, and gyroscope. To test our hypothesis, we implemented an Android video recording app that supports what we call sensor-rich recording, meaning that data from available sensors is also recorded alongside the video. We are able to use this app to collect test data for our purposes.

Here, we evaluate the performance of our method on three videos that are recorded using the Android app that we have developed. We manually constructed visually pleasing final-cuts for these videos. These cuts serve as our ground truth. Next, we use our method to automatically construct final-cuts by using one or more data (i.e., non-visual) and video (visual) streams. For these results we use the following five data streams: 1) D_a (accelerometer x, y and z readings); 2) D_o (orientation roll, pitch and yaw values); 3) D_{fd} (salient face detection); 4) D_f (focus analysis); and 5) D_{of} (optical flow). We provide accuracy ϕ for the final-cuts, defined as

$$\phi = \frac{t_p + t_n}{t_p + f_p + t_n + f_n},$$

where t_p, t_n, f_p and f_n denote *true positive*, *true negative*, *false positive* and *false negative* segment lengths respectively. The results demonstrate that the quality of the summarization increases when we combine visual processing with information gleaned from non-visual sensors.

We are cognizant of the fact that the ground truth for these evaluations reflects our personal biases as to what constitutes a visually appealing video segment. We plan to undertake larger scale human studies in the future.

4.1. Video 1: Lab Video

The first video is recorded in the lab and a number of individuals are visible in this video. Figure 1 (top-left) shows a selection of frames from this video. Faces are visible in some frames of the video. On the other hand a number of frames are either out-of-focus or were recorded when the mobile device was facing downwards. This video showcases the use of (salient) face detection when constructing the final-cut of the video. Figure 1 also shows selection of frames from 3 different final-cuts, each of which was constructed using one or more sensor and video streams.

We constructed ground truth by manually constructing a final-cut, primarily focusing on high-quality segments (i.e., frames that are in focus) that contain faces. The accuracy results for different final-cuts are listed in Table 1. The final-cut obtained by using accelerometer and magnetometer data has the lowest accuracy as expected. The second cut combines face detection, focus analysis, and roll data (to determine the attitude of the recording device). This cut has higher accuracy than that of the first cut. Still this cut contains some undesirable frames. The last cut also uses accelerometer information, and this cut boasts the highest accuracy. These results show that it is indeed advantageous to combine sensor (non-visual) with video data to construct high-quality final-cuts.

Streams	**Accuracy** (ϕ)	**Duration** (s)
$D_a D_o$	0.43	87.3
$D_o D_{fd} D_f$	0.77	52.3
$D_a D_o D_{fd} D_f$	0.90	39.5

Table 1. Quality measures and durations for each of final videos automatically generated from the raw lab video.

Streams have different strengths and together these contribute to the overall accuracy (and quality) of the final-cut. For this video the face detection stream, for example, results in the largest gain in accuracy; however, it cannot handle every case. Consider the $D_o D_{fd} D_f$ final-cut in Table 1 where the TV is incorrectly categorized as a face, leading to the inclusion of a fairly large uninteresting region in the final cut. Focus analysis combined with accelerometer is able to resolve this issue.

4.2. Video 2: Ferris Wheel Video

The second video is recorded using our Android app while sitting on a Ferris wheel. Various views of the fairgrounds are seen in the video for the duration of the ride. A selection of frames from this video can be seen in Figure 7. Some frames are over-exposed, while others capture moments when the mobile device was not oriented correctly to capture any meaningful video.

Figure 7 shows three final cuts constructed from the recorded video using different combinations of sensor and video streams. The first cut uses pitch, roll, and yaw sensor streams (Right, top row, Figure 7). This cut does not use any visual data. Notice that this cut contains some over-exposed frames. The second cut uses focus information (inferred using the visual stream) in addition to accelerometer data (Right, middle row). The last cut is constructed using focus analysis, accelerometer, roll, pitch, and yaw sensor streams (Right, bottom). Roll, pitch, and yaw readings are used to determine if the camera is pointing in roughly the "forward" direction. For this case, our intent is clearly to avoid segments that are captured when the mobile device is facing downwards.

In order to ascertain the effects of using sensor streams, video streams, and a combination of the two for constructing final cuts, we use our method to automatically construct

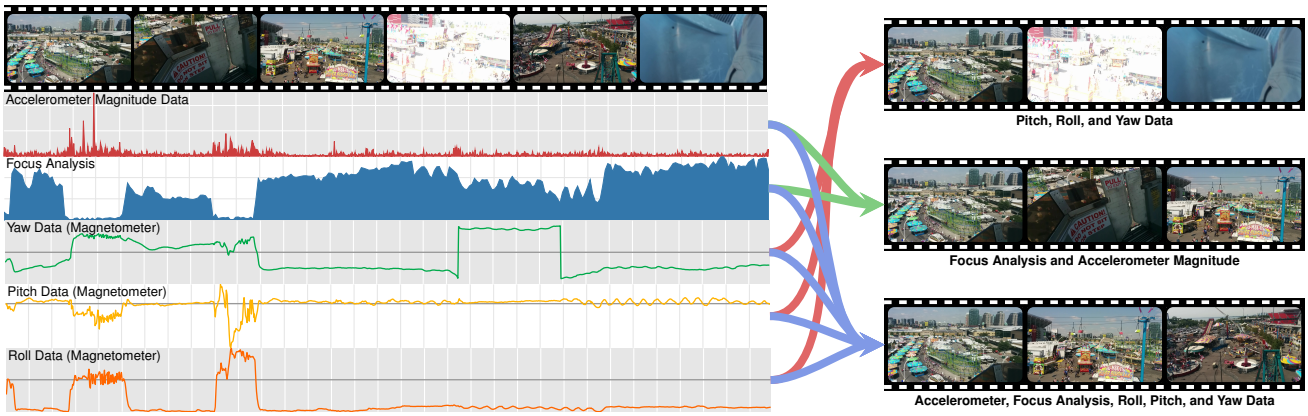

Figure 7. Example frames extracted from the raw ferris wheel video (top left) along with frames from final cuts constructed using different combinations of visual and sensor stream data (right). Arrows are used to show which streams contribute to each final cut.

Streams	**Accuracy** (ϕ)	**Duration** (s)
D_{of}	0.91	150.6
D_f	0.96	142.2
D_aD_o	0.96	141.6
$D_aD_oD_f$	0.99	134.6

Table 2. Quality measures and durations for each of final videos automatically generated from the raw ferris wheel video.

Streams	**Accuracy** (ϕ)	**Duration** (s)
D_a	0.84	105.1
D_o	0.83	107.5
D_f	0.77	80.2
D_aD_f	0.87	59.4
$D_aD_oD_f$	0.92	68.7

Table 3. Quality measures and durations for each of final videos automatically generated from the raw concert video.

four different final cuts of the Ferris Wheel video (see Table 2). The first cut uses only optical flow stream, and it is able to obtain an accuracy of 0.91. The second cut uses the focus analysis stream and achieves an accuracy of 0.96. This is an improvement over the first cut. The second cut also uses much less computational resources, since it avoids computationally expensive optical flow estimation. The third cut, which uses accelerometer and orientation streams, also achieves 0.96 accuracy. The third cut uses no visual processing. The last cut combines accelerometer, orientation and focus analysis streams and achieves an accuracy of 0.99. These results confirm our hypothesis that final cuts that combine both sensor and video streams boast higher accuracy numbers than those cuts that use only sensor or video streams.

4.3. Video 3: Concert Video

The last video was recorded during a music concert. This video is of particular interest since camera equipped mobile devices are perfectly suited to capture such personal moments. The captured video exhibits a variety of video effects. Some portions are captured using landscape mode, while others are recorded in portrait mode. At times the device is facing the ground. Due to extreme motion and the dynamic nature of the event being recorded, more than a few frames are out-of-focus.

We used our method to construct five different final cuts (see Table 3). The first two cuts use accelerometer and magnetometer sensor data streams, respectively. The third cut uses focus analysis stream. The accuracy for this cut is lower than that of the first two. The fourth cut combines focus analysis and accelerometer streams. This cut achieves better accuracy than the first three cuts. The last cut that uses focus analysis, accelerometer and orientation streams boasts the highest accuracy. It is also possible to compensate for portrait and landscape recording when creating a cut.

The results obtained for this video paints a similar picture: cuts that use both sensor and video streams achieve better quality than those that rely upon only sensor or video streams. Accelerometer data is able to select video segments exhibiting little or no motion blur effects and focus analysis can further refine this selection. Notice that cuts that combine accelerometer and focus analysis streams have a higher quality than cuts that use either accelerometer or focus analysis streams, but not both. The (fifth) cut that also uses orientation data in addition to accelerometer and focus analysis streams achieves even better results. Orientation data is used to discard video segments where the camera is facing downwards (thus not pointing at the stage).

Portions of this video were captured in landscape mode, while other parts were captured in portrait mode. Although our automatically edited cuts consist of segments from both of these orientations, the orientation stream data provided by the magnetometer sensor makes it a trivial task to create

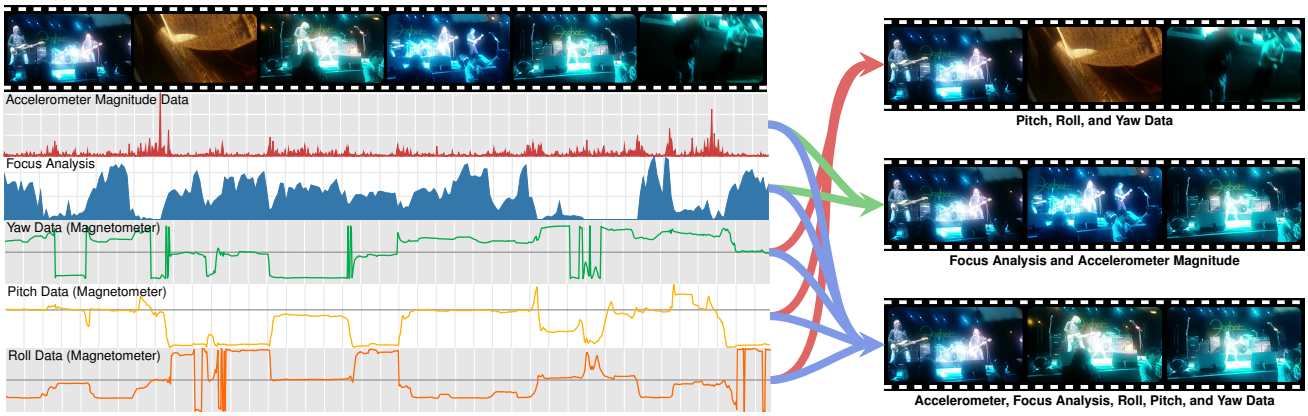

Figure 8. Example frames extracted from the raw concert video (top left) along with frames from final cuts constructed using different combinations of visual and sensor stream data (right). Arrows are used to show which streams contribute to each final cut.

a final cut consisting of segments from only a single orientation (portrait or landscape).

5. Conclusion

This paper explores the idea of automatically editing videos using non-visual sensor streams and video analysis. Our system is designed for a camera-equipped mobile device supporting sensors, such as gyroscope, accelerometer, and magnetometer. These sensors are commonplace in today's mobile devices. We implemented an Android app capable of storing gyroscope, magnetometer, and accelerometer streams while recording videos. The app also implements our video editing framework described herein. Consequently, it is possible to use this app to construct multiple final cuts of the recorded video. The app supports automatic, semi-automatic, and manual modes for constructing the final cut of the recorded video.

Video summarization is an active area of research; however, none of the existing video summarization benchmarks are suitable for our needs. These benchmarks do not include any sensor information. Therefore, we have demonstrated the system on three videos that we captured using the app, which was developed as a part of this project. The results support the primary hypothesis of this work; it is advantageous to exploit non-visual sensor streams (where available) during video summarization.

In the future, we plan to include support for a greater range of video based techniques, e.g., degree of partial blur [8], aesthetic appeal of a video segment [4, 7], etc. We also plan to conduct a larger user-study to fully understand the perceptual quality of the final cuts constructed from the raw video.

References

[1] G. Ahanger and T. D. C. Little. Automatic composition techniques for video production. *IEEE Transactions on Knowledge and Data Engineering*, 10(6):967–987, Nov 1998.

[2] T. Ahonen, A. Hadid, and M. Pietikäinen. Face recognition with local binary patterns. In *Computer Vision - ECCV 2004*, volume 3021 of *Lecture Notes in Computer Science*, pages 469–481. Springer Berlin Heidelberg, May 2004.

[3] Q. Chen, M. Wang, Z. Huang, Y. Hua, Z. Song, and S. Yan. Videopuzzle: Descriptive one-shot video composition. *IEEE Transactions on Multimedia*, 15(3):521–534, Apr 2013.

[4] S. Dhar, V. Ordonez, and T. L. Berg. High level describable attributes for predicting aesthetics and interestingness. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1657–1664, Washington, DC, USA, Jun 2011.

[5] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, volume 2749 of *Lecture Notes in Computer Science*, pages 363–370. Springer-Verlag, Jul 2003.

[6] X.-S. Hua, L. Lu, and H.-J. Zhang. Optimization-based automated home video editing system. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(5):572–583, May 2004.

[7] C. Li, A. C. Loui, and T. Chen. Towards aesthetics: A photo quality assessment and photo selection system. In *Proceedings of the 18th International Conference on Multimedia*, pages 827–830, Firenze, Italy, Oct 2010.

[8] R. Liu, Z. Li, and J. Jia. Image partial blur detection and classification. In *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Anchorage, AK, USA, Jun 2008.

[9] A. Mian. Unsupervised learning from local features for video-based face recognition. In *Proceedings of the 8th IEEE International Conference on Automatic Face Gesture Recognition*, pages 1–6, Amsterdam, Netherlands, Sep 2008.

[10] I. Otsuka, R. Radhakrishnan, M. Siracusa, A. Divakaran, and H. Mishima. An enhanced video summarization system using audio features for a personal video recorder. *IEEE Transactions on Consumer Electronics*, 52(1):168–172, Feb 2006.

[11] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martinez, and J. Fernández-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In *Proceedings of*

the 2000 International Conference on Pattern Recognition, pages 314–317, Barcelona, Spain, Sep 2000.

[12] S. Pertuz, D. Puig, and M. A. Gacia. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, 46(5):1415–1432, May 2013.

[13] M. Qadri, K. Tan, and M. Ghanbari. Frequency domain blockiness and blurriness meter for image quality assessment. *International Journal of Image Processing*, 5(3):352–360, Sep 2011.

[14] J. Roberts, P. Corke, and G. Buskey. Low-cost flight control system for a small autonomous helicopter. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, pages 546–551, Taipei, Taiwan, Sep 2003.

[15] Z. Shen, S. Arslan Ay, S. H. Kim, and R. Zimmermann. Automatic tag generation and ranking for sensor-rich outdoor videos. In *Proceedings of the 19th ACM International Conference on Multimedia*, pages 93–102, Scottsdale, AZ, USA, Nov 2011.

[16] D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and F. L. Wong. Sensay: A context-aware mobile phone. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, page 248, White Plains, NY, USA, Oct 2003.

[17] C. Teng, C. Wu, and Y. Chen. Design and evaluation of mproducer: a mobile authoring tool for personal experience computing. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 141–148, College Park, Maryland, USA, Oct 2004.

[18] H. Tong, M. Li, H. Zhang, and C. Zhang. Blur detection for digital images using wavelet transform. In *Proceedings of the 2004 IEE Conference on Multimedia and Expo*, pages 17–20, Taipei, Taiwan, Jun 2004.

[19] N. Xiong and P. Svensson. Multi-sensor management for information fusion: issues and approaches. *Information Fusion*, 3(2):163–186, Kun 2002.

[20] Y. Zhang, G. Wang, B. Seo, and R. Zimmermann. Multivideo summary and skim generation of sensor-rich videos in geo-space. In *Proceedings of the 3rd Multimedia Systems Conference*, pages 53–64, New York, NY, USA, Feb 2012.