View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Carleton University's Institutional Repository

FAST FEATURE-BASED VIDEO SEGMENTATION AND ANNOTATION

Anthony Whitehead
Carleton University

ABSTRACT

We present a method of segmenting video to detect cuts with accuracy equal to or better than both histogram and other feature based methods. As well, the method is faster than other feature based methods. By utilizing feature tracking on corners, rather than lines, we are able to reliably detect features such as cuts, fades and salient frames. Experimental evidence shows that the method is able to withstand high motion situations better than existing methods. Initial implementations using full sized video frames are able to achieve processing rates of 10-30 frames per second depending on the level of motion and number of features being tracked; this includes the time to generate the MPEG decompressed frames.

1 INTRODUCTION

Much work has been completed in the area of scene detection, shot detection and annotation and as a result, the methods and algorithms are quite mature. In 1965, Seyler [1] developed a frame difference encoding technique for television signals. The technique is based on the fact that only a few elements of any picture change in amplitude in consecutive frames. Since then much research has been devoted to video segmentation techniques based on Seyler. A variety of metrics have been suggested to work on either raw video or compressed data. These metrics are used to quantify the difference between two adjacent frames and can be further sub-classified into 3 major categories:

- Pixel-Level/Histogram Change Detection [2,3,4]
- DCT Change Detection [5,6,7]
- Subband Feature Change Detection [8]

The basic idea behind shot/scene detection is to evaluate the similarity of adjacent frames using one of the aforementioned metrics. When the similarity measure crosses a certain threshold, a scene change or shot boundary has been detected. Equations (1) and (2) below describe a pixel level change metric. Histogram change metrics utilized histogram values of the pixel data rather than the pixel values themselves.

$$DI_i(x,y) = \begin{cases} 1 \text{ if } |I_i(x,y)-I_{i+1}(x,y)| > t \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

$$\sum_{x=1}^{X}\sum_{y=1}^{Y} DI_i(x,y)/(X*Y) > T \quad (2)$$

In (1), we compute the difference between pixel values between image i and i+1 and create a difference image DI, where t is a threshold signifying individual pixel difference. We then compute the overall image difference using (2). If the percentage of image change is greater than our threshold T, we declare a shot boundary. Any of the aforementioned metrics is sufficient for this step, however they rely on certain types of video, MPEG compressed, for example. The pixel level detection metric displayed in (1) and (2) is the most basic form for raw, uncompressed video which is easily achieved regardless of the input. Suitable values for the thresholds t and T are 10 and 0.75 respectively. Depending on the precision requirements, time requirements, and characteristics of the input data one may find another metric or threshold values to be more suitable. We use this metric as a comparison baseline for the method presented in this work.

An edge based feature approach was presented in [9] and further refined in [10]. While these methods provide advances in capabilities, they suffer greatly from the complexity and increased runtimes. A recent review [11] managed to get real time capabilities of the feature-based method presented in [10] but only on micro-frames of 88x72 pixels. When the frame sizes were increased to 352x288 (standard resolutions) the frame-processing rate dropped to approximately 2 frames per second. Our method, presented next, easily achieves frame rates of 10 frames per second, including the MPEG decompression time on standard resolutions.

2 MOTION ESTIMATION AND FEATURE TRACKING

We utilize a corner-based feature tracking mechanism to indicate the characteristics of the video frames over time. As we track corner features over time, we detect production features within the video and annotate the sequence depending on the features that are successfully tracked over time versus those that are lost. Feature tracking is performed on the luminance channel (grey map) for the

0-7803-7946-2/03/$17.00 ©2003 IEEE.

video frames. The luminance channel is computed as follows:

$$\text{Luminance} = \text{Red}*0.299 + \text{Green}*0.587 + \text{Blue}*0.114 \quad (3)$$

The feature tracker we use is based on the work of Lucas and Kanade in [12]. This work was further developed by Tomasi and Kanade in [13] of which Shi and Tomasi provide a complete description in [14]. Recently, Tomasi proposed a modification to make the similarity measures symmetric with respect to adjacent images; the resulting equations are derived completely by Birchfield [15].

Briefly, features are located by examining the minimum eigenvalue of a 2x2 image gradient matrix that is noticeably similar to the Harris corner detector [16]. The features are tracked using a Newton-Raphson method of minimizing the difference between the two windows around the feature points. We continue by presenting a very brief outline of the work by Tomasi et al.

Given a point p in an image I, and its corresponding point *q* in an image J, the displacement vector δ between *p* and *q* is best described using an affine motion field:

$$\delta = Dp + t \quad (4)$$

where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \quad (5)$$

is a deformation matrix and t is the translation vector of the centre point of the tracked feature window. The translation vector t is measured with respect to the feature in question. Tracking feature *p* to feature *q* is simply determining the six parameters that comprise the deformation matrix D and the translation vector t. In the case of pure translation, D will be the identity and thus

$$\delta = p + t \quad (6)$$

Because of this, the case of pure translation is computationally simpler and thus preferable. Since the motion between adjacent frames of standard video is generally quite small, it turns out that setting the deformation matrix to identity is the safest computation [13], leaving us with the translation vector being exactly the displacement vector. While tracking features, it is possible, that large motion between frames does occur. It has been noticed that in such cases the tracking mechanism begins to fail because the disparity between adjacent frames is too large. The result, features are lost and cannot be tracked any further. This fact indicates that some large shift in the adjacent frames has occurred and can be handled at the cost of substantially higher processing time or by removing the identity constraints of the matrix *D*.

In the sections that follow, we present heuristics for the detection of production effects such as cuts, fades and salient frames for storyboarding. Finally we note that C code for the Kanade, Lucas, Tomasi tracker can be found here [17] to simplify implementation of this algorithm.

3 SEGMENTATION, SALIENT FRAME EXTRACTION AND ANNOTATION

By tracking features, on a frame-by-frame basis, we are able to detect cuts, fades and salient frames. Cuts are defined as abrupt changes of content in adjacent frames, commonly termed razor edits. Fades are gradual transitions to or from a solid colour (usually black). Finally salient frames are interim frames between cuts that help build a storyboard so that the content of the missing frames can be easily interpolated.

3.1 Detecting Cuts

Cuts are easily detected by examining the number of features successfully tracked in adjacent frames, refreshing the feature list for each comparison. In algorithmic form, cuts are detected as so:

```
1. Select good features for frame 1,
   place in feature list FLcut
2. For each frame x in the video
  a. Track features from FLcut in current
      frame x
  b. Count number of lost features
     i. If(features lost > t (75%))
           Cut Detected
  c. Refresh Flcut using frame x
4. Endfor (2)
```

3.2 Detecting Fades

Fades are simply detected by a string of frames where all features are lost. In the case of a fade to black (or white), the luminance of the frames increase or decreases enough to cause the similarity measured in the windows around corresponding features to fail.

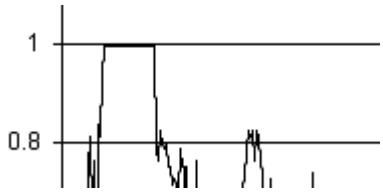

Figure 1. Fade to black indicated as a plateau in the first derivative map.

As figure one shows, this manifests itself as a plateau at 1 (100% feature loss) in the graph of frame differences, and can be simply computed when the 2nd derivative of the frame differences is 0 over a number of consecutive frames.

3.3 Detecting Salient Frames

Segmentation algorithms often ignore salient frames, however they represent important contextual information from the video stream. Because of the contextual value of the salient frames, there is a relationship between the initial frame and the current frames tracked features. As we track features from frame i to i+n, a salient frame is indicated when enough features are lost between frames i and

i+n plus those features that have moved a sufficient distance.

Algorithmically, salient frame extraction:

```
1. Select good features for frame 1
   place in feature list FLsal
2. For each frame x in the video
   a. Track features from FLsal in x
   b. Count number of lost features
   c. Count number of features passing
      the distance threshold
   d. If(lost + passing) > threshold
      i. Signal salient frame
      ii. Refresh FLsal using frame x
3. Endfor(2)
```

3.4 Annotating Video

By examining the frame-to-frame feature loss, we can make determinations of how much motion or action is happening in the scene. As a result, we can annotate the scenes between cuts as active or passive. Annotations such as these will allow searching of video databases via action labels to further assist genre labelling and searching. In figure 2 we see a series of consecutive scenes taken from the film True Lies.

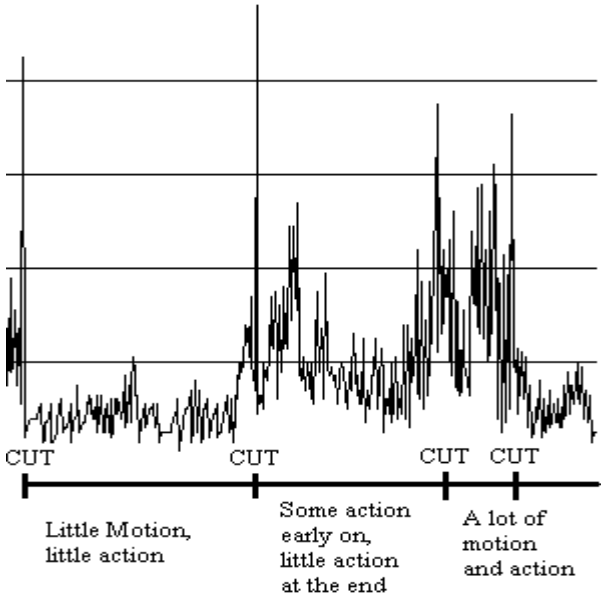

Figure 2. Action annotation for scenes from True Lies; annotations accurately describe the film content

Annotations of action simply relate to the average magnitude of feature loss over a series of frames, the higher the average the more motion/action exist in the scene. More sophisticated annotations are possible by looking at a smaller window, rather than between cuts.

3.5 General settings to achieve good results

The number of features to track, the lost feature and distance thresholds will vary depending on frame dimensions, content type and compression ratio and speed requirements, however a good rule of thumb is the following: Between 1.5 features for every 1000 pixels, the distance threshold should be 1/8 of the smallest dimension, and a sensitivity threshold of 75 percent. For example, a video that is 320x240 would have 96 features to track, a distance threshold of 30 pixels (240/8), and a sensitivity threshold of 75 percent.

4 EXPERIMENTAL RESULTS

In the experiments that follow, a selection of video clips that represent a variety of different video genres are used for cut detection. In table 1, we present a list of data sources, the dataset that we have selected accentuates known trouble areas for feature-based scene detection algorithms, namely[11]:

- Close up moving scenes (F)
- Zooming, Panning (Camera Motion) (C,F)
- Animated video (A)
- Fast motion, or high action (B,C,D)
- An object occluding the majority of the scene (E)
- Credits (C and F)

Source Label	Characteristics of video data
A	Cartoon clip. Substantial object motion.
B	Action movie clip. Action scenes intermixed with scenes that have little action. From the film: True Lies
C	Movie credits with high motion, quick pans and zooms. A very difficult example. From the film: The Santa Clause
D	Sports news, clips of very high action with clips of commentary with no action.
E	Commercial, no cuts, quick motion, many production effects.
F	Film (Groundhog Day), no production effects, just cuts. Easiest test case in the set.

Table 1: Experimental data set

Data	# of cuts	True +'ves	False +'ves	False –'ves	Precision	Recall
A	7	7	0	0	1	1
B	8	8	6	0	.57	1
C	35	35	142	10	.198	1
D	40	38	15	2	.717	.95
E	0[1]	0[1]	7	0	0	0
F	39	39	23	0	.629	1

Table 2: Results of pixel-based algorithm

For a comparison metric, we choose precision and recall rates on the detection of cuts. We compare a pixel-based method against our presented feature-based method. In our experiments, we set the thresholds for both methods to be equivalent, allowing for a comparison of methods that would be unaffected by threshold settings.

[1] We use 1 when computing the precision and recall.

Data	# of cuts	True +'ves	False +'ves	False –'ves	Precision	Recall
A	7	6	0	0	1	.857
B	8	8	0	0	1	1
C	35	35	53	5	.398	1
D	40	40	10	0	.80	1
E	0[1]	0[1]	0	0	1	1
F	39	36	2	3	.947	.923

Table 3: Results of feature-based algorithm

To determine the effectiveness of the salient frame detection, we assembled a test group to examine the created storyboards and subsequently "re-tell the story" as best as they could. In most cases the test subject's explanation of the video content from a storyboard was accurate. One notable area of confusion came from scenes where two characters from the video were engaged in a discussion and the storyboard resulted in a series of headshots of the characters. In a few cases, the test subject recognized the content as a movie or commercial that they had seen before. In figure 3, we present a sample storyboard generated from 191 frames (6.33 seconds) of a baseball game.

Figure 3: Storyboard with salient frames. Cut frames are 1,3 and 4, salient frames are 2,5 and 6.

It is clear from this short sequence (to baseball fans) that Biggio is up at bat, and the pitcher is receiving a signal from the coach. Finally, fades in all test situations reliably hit a plateau at 100% feature loss in all test examples.

5 SUMMARY

We have presented a feature-based method of video segmentation that greatly improves upon the time requirements of previous feature-based segmentation algorithms. While not yet real time, there are still areas for basic code optimisations that will increase the frame rate even further. By utilizing feature tracking and thresholding techniques, we were able to achieve recall and precision rates that match or exceed current methods for detecting cuts, yet also detects fades. As well, the method allows for some basic automatic annotation of video content as well as the extraction of salient contextual information to be used for populating video database fields.

Future areas of research that indicate promising results include a more sophisticated metric for segmenting the video that does not rely on a threshold technique. As well, feature tracking also produced motion vectors for points within frames that may also be further used in the annotation process.

6 REFERENCES

[1] A. Seyler, "Probability distribution of television frame difference", *Proc. Institute of Radio Electronic Engineers of Australia* **2**6(11), pp 355-366, 1965

[2] R. Kasturi and R. Jain, "Dynamic vision", *Computer Vision: Principles* (Kasturi and Jain Eds), IEEE Computer Society Press, pp 469-480, 1991

[3] A. Nagasaka and Y. Tanaka, "Automatic video indexing and full-video search for object appearances", in *Visual Database Systems II*, pp 113-127, 1992

[4] H. Zhang, A. Kankanhalli, S. Smoliar, "Automatic partitioning of full-motion video", *ACM/Springer Multimedia Syst.* 1(1), pp 10-28,1993

[5] F. Arman, A. Hsu, and M. Chiu, "Image processing on compressed data for large video databases", in *Proceedings 1st ACM International Conference on Multimedia*, pp 267-272, *1993*

[6] H. Zhang, A. Kankanhalli, and S. Smoliar, "Video parsing using compressed data", in *Proc. IS&T/SPIE, Image and Video Processing II*, pp 142-149, 1994

[7] B. Shahraray, "Scene change detection and content based sampling of video sequences", *Proceedings IS&T/SPIE* **2419,** 1995

[8] J. Lee and B. Dickinson, "Multiresolution video indexing for subband coded video databases", in *Proc. IS&T/SPIE, Conference on Storage and Retrieval for Image and Video Databases,San Jose, CA*, 1994.

[9] R Zabih, J. Miller, and K. Mai, "A Feature-Based Algorithm for Detecting and Classifying Scene Breaks", Proc. ACM Multimedia, pp. 189-200, 1995

[10] R Zabih, J. Miller, and K. Mai,. "A Feature Based Algorithm for detecting and Classifying Production Effects", *Multimedia Systems*, Vol 7, p 119-128, 1999.

[11] A Smeaton et al., "An Evaluation of Alternative Techniques for Automatic Detection of Shot Boundaries in Digital Video" in *Irish Machine Vision and Image Processing Conference*, 1999.

[12] B. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". *Int. Joint Conf. On A.I.* pp 674-679, 1981.

[13] Carlo Tomasi and Takeo Kanade. "Detection and Tracking of Point Features". *Carnegie Mellon University Technical Report CMU-CS-91-13*2, 1991.

[14] Jianbo Shi and Carlo Tomasi. "Good Features to Trac*k". IEEE Conference on Computer Vision and Pattern Recognitio*n, pp 593-600, 1994.

[15] S Birchfield. "Derivation of Kanade-Lucas-Tomasi Tracking Equation". *Unpublished*, 1996.

[16] C. Harris and M. Stephens."A combined corner and edge detector". *Proceedings of the 4th Alvey Vision Conference*, pp 147--151, 1988.

[17] http://robotics.stanford.edu/~birch/klt/