TRECVID 2006 NOTEBOOK PAPER: FEATURE BASED CUT DETECTION WITH AUTOMATIC THRESHOLD SELECTION

Anthony Whitehead
Carleton University
Ottawa, Canada

Prosenjit Bose
Carleton University
Ottawa, Canada

Robert Laganière
University of Ottawa
Ottawa, Canada

ABSTRACT

Our approach is based on feature tracking where cuts are detected when only few features can be tracked across frame pairs. This unique criterion was found sufficient to detect most cuts in most videos even in the case of short term transitions. Furthermore, we propose a method to stabilize the differences and automatically identify a global threshold to achieve a high detection rate.

For the TRECVID06 submitted runs, we tested the algorithm under the following conditions:

- CU-UO-All-Runtype:1 corresponds to the Precision set as determined by automatic threshold selection;
- CU-UO-All-Runtype:2 corresponds to the F1 set as determined by automatic threshold selection;
- CU-UO-All-Runtype:3 corresponds to the Recall set as determined by automatic threshold selection;
- The other runs correspond to various manually selected threshold values.

One important aspect of our system resides in the user control ability over the type of acceptable errors; according to this, the system will select automatically the most appropriate threshold to minimize the specified type of error (false negatives versus false positives). From the experiments, this component works as expected even though our results were less effective than expected. The exact reason for the exhibited performance is still to be determined currently being investigated.

The method was initially designed for the segmentation media types other than news (e.g. movies, television shows, cartoons, etc.). Within the news media, the existence of graphical overlays introduces a number of unique issues that includes having to adjust the feature detection to areas where the video are changing. Indeed, computer graphics introduce sharp textures where more features are detected. This also brings us to the definition of a cut in the case where there is a mix of natural background videos and graphical overlays that are changing at different rhythms.

In cinema and television it is common to have few frame shots (even just 1 or 2 frames), that situation does not occur in news so having a wider temporal search range. However, while doing so would improve the results for news, this might result in missed shot in other media types.

1. FEATURE TRACKING FOR QUANTIFYING DISSIMILARITY

We propose here an approach that uses feature tracking as a metric for dissimilarity [1]. Furthermore we propose a methodology to automatically determine a threshold value by performing density estimation on the squared normalized per-frame lost feature count. It has been reported that the core problem with all motion-based features is due to the fact that reliable motion estimation is far more difficult than detecting visual discontinuity, and thus less reliable [2]. Effectively, a simple differencing technique is replaced with a more complex one. Experimentally we have found that the proposed feature tracking method performs flawlessly on all simple[1] examples where pixel and histogram based methods did not achieve such perfect results. We continue by outlining the feature tracking method, an outlier pruning algorithm and a signal separation methodology. We follow up in the next section with a method to dynamically select a global threshold. In Figure 1 we see the entire flow chart for computing the positions of cuts in a video sequence. Each block within the diagram is detailed in this section and the next.

[1] Here we define simple to be cases of clearly obvious cuts, which were well separated over time and space.

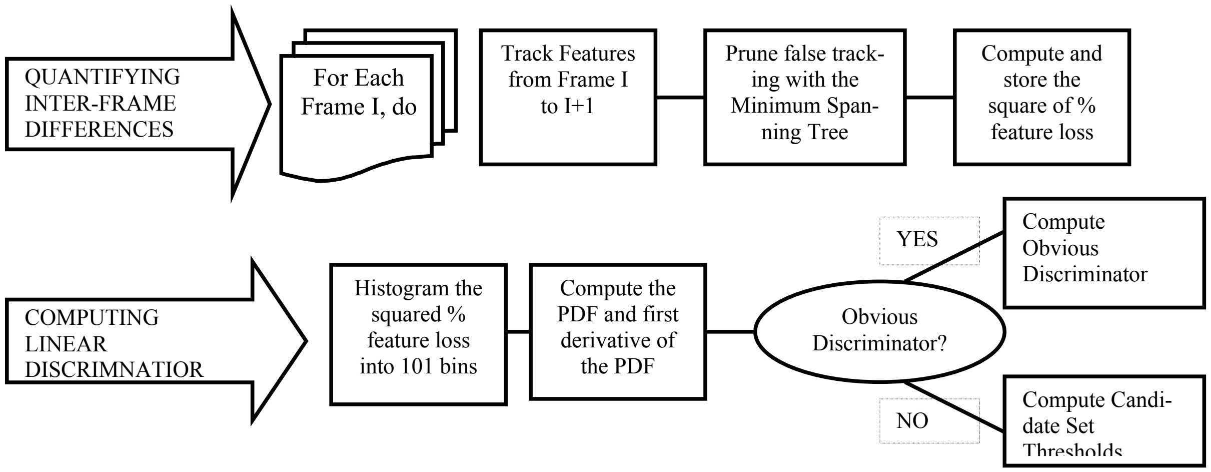

QUANTIFYING INTER-FRAME DIFFERENCES
For Each Frame I, do
Track Features from Frame I to I+1
Prune false tracking with the Minimum Spanning Tree
Compute and store the square of % feature loss
COMPUTING LINEAR DISCRIMNATIOR
Histogram the squared % feature loss into 101 bins
Compute the PDF and first derivative of the PDF
Obvious Discriminator?
YES
Compute Obvious Discriminator
NO
Compute Candidate Set Thresholds

Figure 1: Diagram of system to compute cuts

1.1 FEATURE TRACKING

Previous feature based algorithms [3,4] rely on course-grained features such as edges and do not track edge locations from frame to frame. Rather they rely on sufficient overlap of a dilated edge map and search a very small local area around the original edge locations. In contrast, the proposed method of tracking fine-grained features (corners) on a frame-by-frame basis in less constrained by the original location due to the pyramidal tracking approach. This allows the proposed method to be more robust to object and camera motions. Cuts are detected by examining the number of features successfully tracked (and lost) in adjacent frames, refreshing the feature list for each comparison.

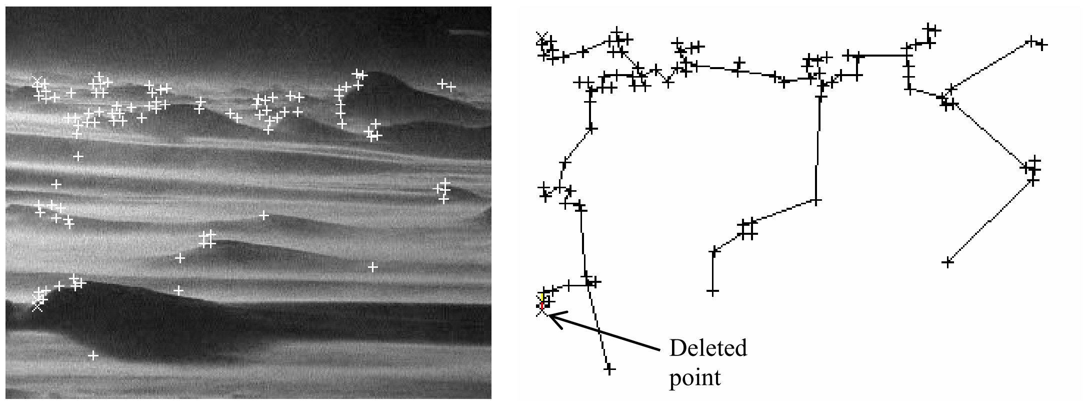
We utilize a corner-based feature tracking mechanism to indicate the characteristics of the video frames over time. As we track corner features over time, we detect production features within the video and can annotate the sequence depending on the features that are successfully tracked over time versus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

$$\text{Luminance} = \text{Red}*0.299 + \text{Green}*0.587 + \text{Blue}*0.114 \quad (4)$$

The feature tracker we use is based on the work of Lucas and Kanade in [5]. This work was further developed by Tomasi and Kanade in [6] of which Shi and Tomasi provide a complete description in [7].

1.2 PRUNING FALSE TRACKING

In the case of a cut at frame I, all features being tracked should be lost from frame I_{i-1} to I_i. However, there are often cases where the pixel areas in the new frame coincidentally match features that are being tracked. In order to prune these coincidental matches, we examine the minimum spanning tree of the tracked and lost feature sets.



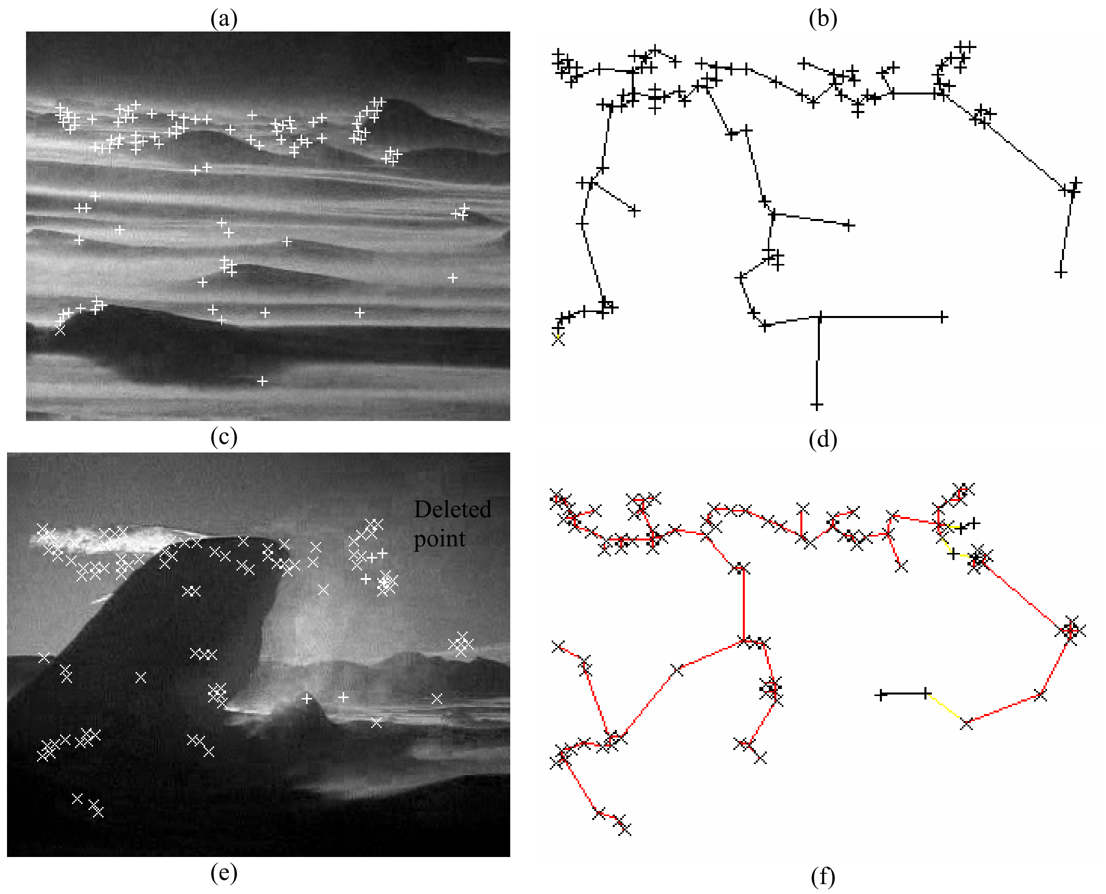


Figure 2: Three consecutive frames from a sequence. (a) shows a very high proportion of successfully tracked features from the previous frame to current frame (b) shows successfully tracked features from (a) (previous) to (b) (current) (c) shows those features cannot be found in very high proportion indicating a cut. Above each frame is the minimum spanning tree for each of the feature sets, (+) are tracked features, (X) are lost features.

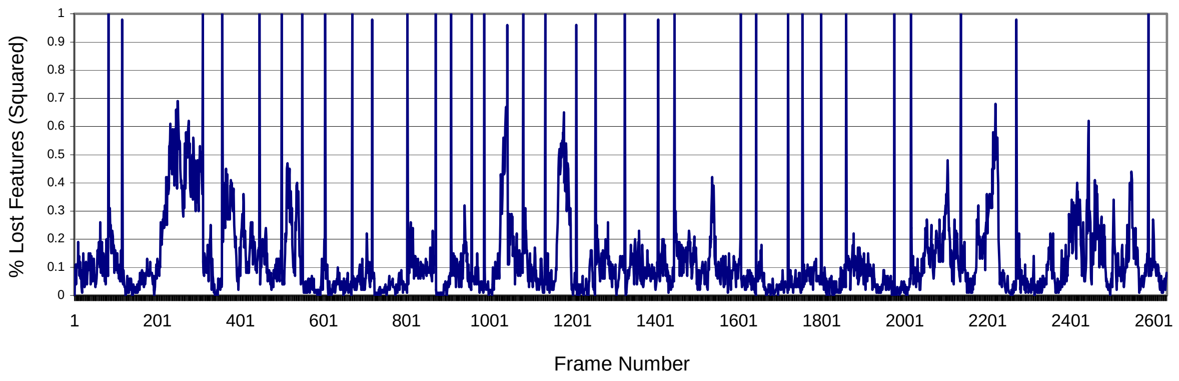


Figure 3: Percentage of lost features (squared) when tracking feature points between previous and current frames. Peaks should correspond to cuts in the sequence.

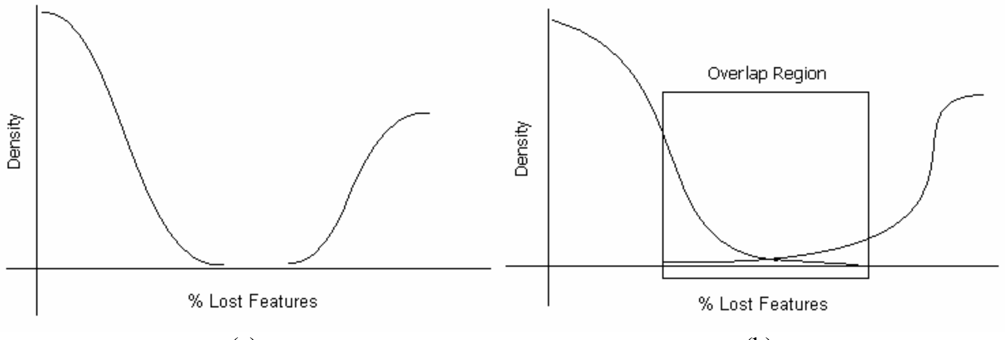
We can see from Figure 2 (c), that there is a very small percentage of features that are tracked. This is clearly an erroneous situation because the two consecutive frames are so obviously different. We can remove some of these erroneous matches by examining properties of the minimum spanning tree of the tracked and lost feature sets. By severing edges that link tracked features to lost features we end up with several disconnected components within the graph. Any node (feature) in the graph that becomes a singleton (not connected to any other feature) has its status changed from tracked to lost, and is subsequently included in the lost feature count. The property we are exploiting here is the fact that erroneously tracked features will be minimal and surrounded by lost features. Clusters of tracked (or lost) features have localized support that we use to lend weight to our assessment of erroneous tracking.

Our inter-frame difference metric is the percentage of lost features from frames I_{i-1} to I_i. This corresponds to changes in the minimum spanning tree, but is computationally efficient. Because we are looking to automatically define a linear discriminator between the cut set and the non-cut set, it is advantageous to separate these point sets as much as possible. We therefore decided to square the percent feature loss which falls in the range [0..1]. Figure 3 the result obtained when tracking the consecutive frames of a video sequence.

2. AUTOMATICALLY DETERMINING A LINEAR DISCRIMINATOR

Having a difference metric, we can now compute the linear discriminator for the two sets. There is no common threshold that works for all types of video. Next we present an algorithm to auto-select a global threshold for a given sequence. There are two classes of frame differences, cuts and non-cuts; and our goal is to find the best linear discriminator that maximizes the overall accuracy of the system. The tracking result of a given sequence can be considered to be the observation of two separate distributions. The first one is generated by the frames of a same shot for which the percentage of lost features is normally low (with an extension that usually depends on the level of action in the scene). The second distribution is generated by the cuts; in this case all features should be lost but as mentioned previously, because of the unavoidable incidental match, there will remain a small portion of tracked points inside cuts. The cut set and the non-cut set should ideally be two separate distributions that should not overlap, however in practice they often do, as illustrated in Figure 4b. When the two distributions overlap a single threshold will result in false positives and false negatives. An optimal differencing metric would ensure that these two distributions do not overlap; in such a case the discriminating function is obvious and accuracy is perfect. The quality of the difference metric directly affects the degree to which the two distributions overlap, if any.

To avoid the problems that may occur with a windowed adaptive threshold, we have opted to examine the density of the recorded inter-frame difference values for an entire sequence. The idea here is that there should be two distinct high-density areas, those where tracking succeeded (Low feature loss) and those where tracking failed (high feature loss). We will introduce the idea of a candidate set in Section 2.4, which is the set of features that can be discriminated by zero crossings of the probability density function that characterizes the densities of the inter-frame differences. It needs to be noted here that while we examine the density for the entire sequence to determine a global threshold, it is possible to apply the method outlined next in a windowed manner to determine localized thresholds. The density is also a characteristic of the content nature of the video; it has therefore to be re-estimated for each new sequence as no universal threshold would work effectively on all videos.



(a) (b)

Figure 4: (a) Non-overlapping distributions, the discriminator is obvious. (b) Overlapping distribution of the cut set on the right and the non-cut set on the left. The ideal discriminator lies within the overlap region.

2.1 DENSITY ESTIMATION

The auto-selection of a threshold will be done by examining the density distribution of the lost features over the entire sequence. We are looking to exploit the fact that the ratio of non-cuts to cuts will be high, and therefore the ratio of low feature loss frame pairs to high feature loss frame pairs will also be high. As the frame to frame tracking of features is independent of all other video frames, we have n independent observations from an n+1 frame video sequence. The extrema of the probability density function can be used to determine the threshold to use. We can use the statistical foundations of density estimation to estimate this function.

The intention of density estimation is to approximate the probability density function *f(●)* of a random variable *X*. Given that we have *n* independent observations $X_1, \ldots, X_n$ (our tracked feature percentage squared) from the random variable *X* (our video sequence). The kernel density estimator for the estimation of the density value *f(x)* at point *x* is defined as

$$\hat{f}(h,x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \tag{8}$$

Where K(●) is the so-called kernel function, h is the bandwidth (window size), and n is the number of samples (number of frames-1). There have been variety of kernel functions presented in the past and we performed an empirical evaluation of the 9 kernel functions.

Now that the triangular kernel has been selected to smooth our function, we need to determine the bandwidth (window size) for the kernel. We performed a analysis with kernel widths 3,5,7,9,11,13,and 15. In Figure 5 we show a triangular kernel for window sizes 3,7,11 and 15. Widths 9 through 15 represented almost identical curves and thus any kernel width over 7 provided no further information and likely is over smoothing. The apparent extrema that appeared in widths 3 and 5 indicated under smoothing. By elimination, we were left with a kernel width of 7, which has provided good results in our experiments.

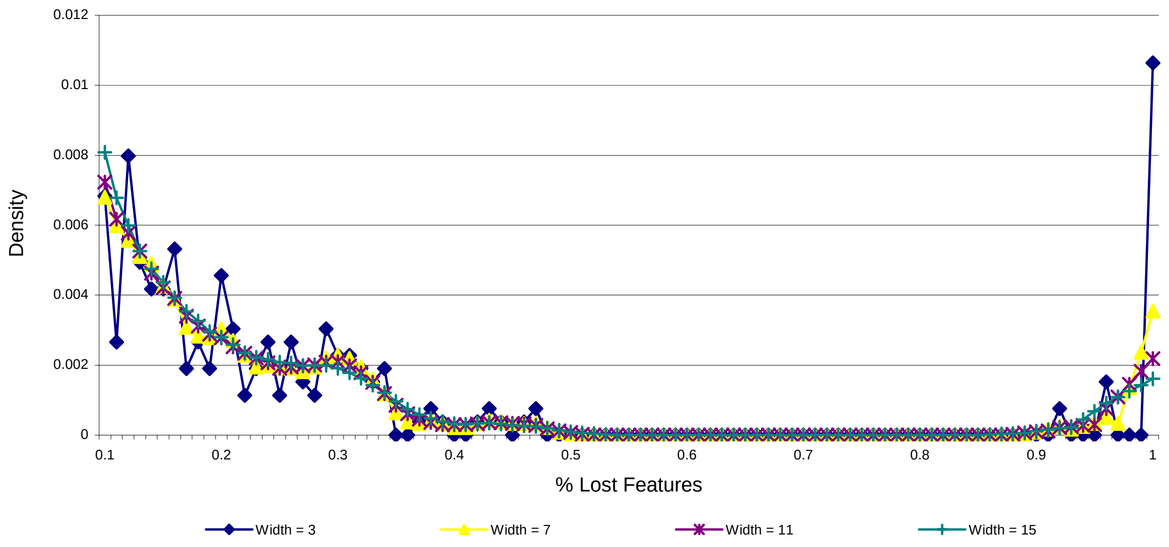


Figure 5: Triangular kernel at various bandwidths (window sizes)

2.2 COMPUTATIONAL CONSIDERATIONS

In the case of an exact computation of the density estimates, the kernel function must be evaluated O(hn^2) times. This increases the computation time as the number of frames becomes large (keep in mind that a 2 hour movie contains 216,000 frames). An alternative, and faster, way is to approximate the kernel density estimate is to use the WARPing (Weighted Average of Rounded Points) method [8] The core concept behind WARPing is to effectively histogram the data into bins of length *d*. The bin centre of its corresponding bin then replaces each observation point for subsequent computation. A typical choice for *d* is to use *h*/5 or (x_{max}-x_{min})/100. In the latter case, the effective sample size (*r*) can be at most 101. This property nicely reflects our situation, where we are keeping track of the percentage of features tracked per frame pair, which is in the range of 0 to 100 percent, or 101 bins.

For the WARPing method, the kernel function needs to be evaluated only at *O(h·r/d)* plus the initial pass to histogram the data being *O(n)*. In total, the number of steps is *O(n)* + *O(h·r/d)* where h is our window width (7), the range *(r)* is 101 and the bin length is 1. This reduces the number of steps to *O(n)* + *O(h·r)*. Since *n* greatly exceeds *h·r* (707 in our experiments), we are have an upper bound of *O(n)*. This is considerably faster than the exact computation, when the sample size is large.

2.3 NON-OVERLAPPING DISTRIBUTIONS

Non-overlapping sets of distributions are very easily determined by looking for a large plateau of zero density. The first appearance (traveling the curve from 0 to 100) of a large plateau indicates the range of the separation point. Selecting the extreme end point (closest to the cut set) for the threshold has yielded the correct result on all cases of non-overlapping distributions in our test suite.

2.4 OVERLAPPING DISTRIBUTIONS

We now introduce the ideas around what we term the candidate sets. When a threshold is selected, a set of cuts is produced. Three such sets are of particular interest; they are the ones that maximize the precision, F1 and recall rates. Precision is the portion of the declared cuts that were correct. Recall is the portion of the true cuts that were declared correctly. F1 is a combination of precision and recall. A complete description of these terms and their formulae are given in the experiments section. Depending on user need, precision, recall or best overall performance, these candidate set thresholds are now able to be determined.

The candidate sets are 3 sets that for convenience we will call the precision set (*P*), the F1 set (*F*) and the recall set (*R*). These sets have the following property:

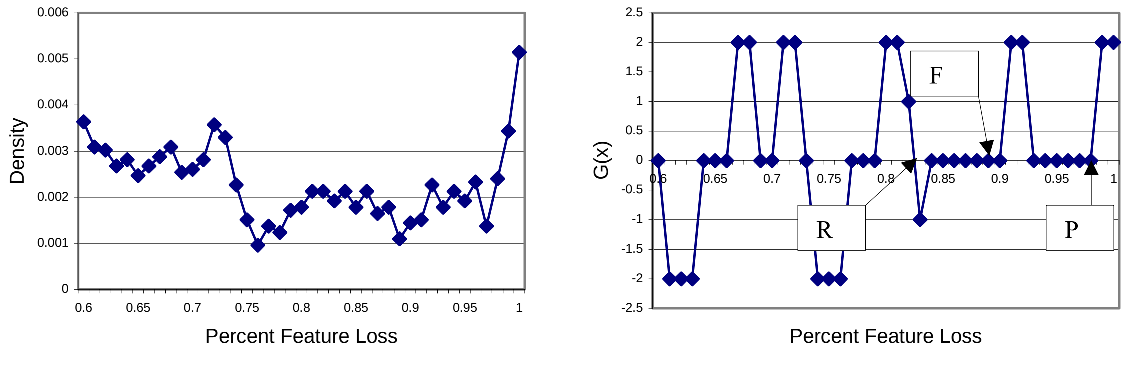
$$R \subseteq F \subseteq P \qquad (9)$$

In the case of non overlapping sets, precision, recall and F1 scores are all 1.0 and the frames in each set are the same. In the case of overlapping sets, the frames in the precision set appear in the F1 set, and those in the F1 set appear in the recall set.

The candidate sets are determined by examining zero crossings of the first derivative of the computed probability density function (PDF). There are often many consecutive zero crossings of the function over time, so we use a modified function G(x) to make the large changes in density more apparent. The first derivative of the PDF *f(x)*, is modified to a function G(x) using the following rules:

$$\begin{aligned} &G(x) = g(x) + g(x+1): \\ &\quad \text{if } \hat{f}'(x) < 0 \text{ then } g(x) = -1 \\ &\quad \text{if } \hat{f}'(x) = 0 \text{ then } g(x) = 0 \\ &\quad \text{if } \hat{f}'(x) > 0 \text{ then } g(x) = 1 \end{aligned} \qquad (10)$$

In Figure 6, we see the original first derivative function and the modified function.



(a) (b)

Figure 6: (a) original PDF (b) Modified first derivative (G(x)) with marked zero crossings for P,F and R.

The zero crossings, starting from 1.0 and following G(x) as x decreases are used to determine the thresholds for each of these sets using the criteria listed here:

- (*P*) is The first zero crossing
- (*F*) The position of the minimum of PDF corresponding to the plateau of G(x) given:
    - If the next zero crossing has opposing direction as the first (i.e. is not u or n shaped) and is part of the plateau of first zero crossing use this plateau, otherwise use the next plateau.
- (*R*) The next subsequent zero crossing

The arrows in Figure 10(b) point to the zero crossings used. The first zero crossing is at 0.98 (P) and because the next zero crossing at 0.93 is also an upwards direction (u shaped), we skip to the next plateau to determine F. The next zero crossing (not on the plateau) is used for R.

3. CONCLUSIONS

We have presented a fine-grained feature-based method for video segmentation, specifically cut detection. By utilizing feature tracking and an automatic threshold computation technique, we were able to achieve F1, recall and precision rates that generally match or exceed current methods for detecting cuts. The method provides significant improvement in speed over other feature-based methods and significant improvement in classification capabilities over other methods. The application of feature tracking to video segmentation is a novel approach to detecting cuts.

Due to problems associated with a window based adaptive thresholding, we have introduced the concept of candidate sets that allow the user to prejudice the system towards results that are suitable to individual needs. This kind of thresholding is a novel approach to handling the overlapping region of two distributions, namely the cut set and the non-cut set in video segmentation.

REFERENCES

[1] A. Whitehead, J. Bose, R. Laganiere. Feature based cut detection with automatic threshold selection. *IEEE Workshop on Motion and Computing*. Breckenbridge, CO, pp. 132-139, 2005.
[2] R. Lienhart. Reliable Transition Detection In Videos: A Survey and Practitioner's Guide. International Journal of Image and Graphics (IJIG), Vol. 1, No. 3, pp. 469-486, 2001.
[3] R Zabih, J. Miller, and K. Mai, "A Feature-Based Algorithm for Detecting and Classifying Scene Breaks", Proc. ACM Multimedia, pp. 189-200, 1995
[4] 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.
[5] 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.
[6] Carlo Tomasi and Takeo Kanade. "Detection and Tracking of Point Features". *Carnegie Mellon University Technical Report CMU-CS-91-13*2, 1991.
[7] Jianbo Shi and Carlo Tomasi. "Good Features to Trac*k". IEEE Conference on Computer Vision and Pattern Recognitio*n, pp 593-600, 1994.
[8] W. Hardle and D. Scott. "Smoothing in by weighted averaging using rounded points", *Computational Statistics* Vol. 7: 97-128, 1992.