Feature Tracking for Cut Detection

Anthony Whitehead, Prosenjit Bose, Robert Laganiere

Introduction

Recently, investigation into shot boundary detection schemes has gathered much momentum [20-29]. Cut detection is seemingly easily solved by an elementary statistical examination of inter-frame characteristics; however a truly accurate and generalized cut detection algorithm still eludes researchers. Reliable shot boundary detection forms the cornerstone for video segmentation applications as shots are considered to be the elementary building blocks that form complete video sequences. Applications such as video abstraction, video retrieval and higher contextual segmentation all presuppose an accurate solution to the shot boundary detection problem [8, 17, 18, 19]. Automatic recovery of these shot boundaries is an imperative primary step, and accuracy is essential.

Shot transitions can be classified into four classes based on the 2D image transformations applied during transition production [20].

- *Identity Class:* Neither of the two shots involved are modified, and no additional edit frames are added. Only hard cuts qualify for this class.
- *Spatial Class:* Some spatial transformations are applied to the two shots involved. Examples are wipe, page turn, slide, and iris effects.
- *Chromatic Class:* Some color space transformations are applied to the two shots involved. Examples are fades and dissolve effects.
- *Spatio-Chromatic Class:* Some spatial as well as some color space transformations are applied to the two shots involved. All morphing effects fall into this category.

In this work, we concentrate only on the identity class, as our goal is to improve the accuracy of cut detection by introducing a new differencing metric based on stable feature tracking from frame to frame. The basic idea behind the technique has been shown to detect fades [29], but we concentrate solely on cuts in this work.

Additional Background

In 1965, Seyler developed a frame difference encoding technique for television signals [1]. 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 the ideas of Seyler. 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. However, a truly accurate cut detection algorithm has yet to be introduced. Any improvements in cut detection will ultimately improve the applications that rely on it. Hard cuts are the most common transition between shots. A hard cut is the direct concatenation of two different sequence without the presence of transitional frames.

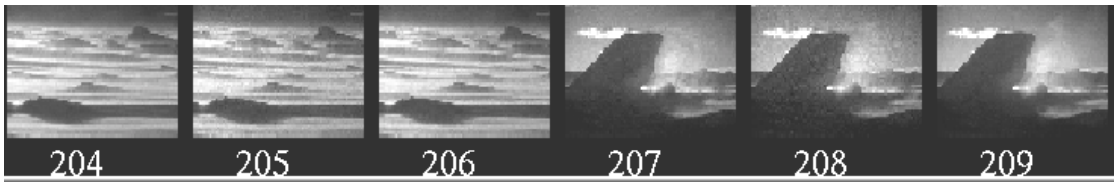


FIGURE 4.1: HARD CUT BETWEEN FRAMES 206 AND 207 OF A VIDEO SEQUENCE.

A hard cut produces a visual discontinuity in the video stream as we see in Figure 4.1. Existing hard cut detection algorithms differ in the feature(s) used to measure the inter-frame differences and in the classification technique used to determine whether or not a discontinuity has occurred. However, they almost all define hard cuts as isolated peaks in the features time series. In [21] a complete survey is given on techniques to compute inter-frame difference and classify the type of transition. A variety of metrics have been suggested to work on either raw video or compressed data and we briefly outline methods that have been used in the past, or are currently in use, forming the basis of our comparisons. We briefly outline the techniques next.

Quantifying Inter-frame Differences

The basic idea behind shot/scene detection is to evaluate the similarity of adjacent frames using some metric. When the similarity measures cross a certain threshold, a scene change or shot boundary will be classified as occurring. Selecting a better method to quantify the inter-frame differences improves the accuracy of the classification algorithm and simplifies the implementation. In this section we outline several known inter-frame difference quantification techniques.

Individual Pixel Differences

Equations (4.1) and (4.2) describe a pixel level change metric and a cut classifier respectively.

$$DI_i(x,y) = 1 \text{ if } |I_i(x,y) - I_{i+1}(x,y)| > t$$
$$0 \text{ otherwise} \quad (4.1)$$

$$\sum_{x=1}^{X}\sum_{y=1}^{Y}\frac{DI_i(x,y)}{(X*Y)} > T \quad (4.2)$$

In (4.1), we compute the difference between pixel values between images i and i+1 to create a difference image DI, where t is a threshold signifying individual pixel difference. We then compute the overall image difference using (4.2). If the percentage of image change is greater than a threshold *T*, we declare a shot boundary. The pixel level detection metric displayed in (4.1) and (4.2) is the most basic form for raw, uncompressed video.

Unfortunately, this simple metric is very susceptible to object and camera motion. Even if camera motion is compensated and pixels are transformed (via homographies for example) before being compared, object motion still poses significant difficulties. More sophisticated methods use optical flow, the number and distribution of motion vectors and the strength of the residual derived by block matching features [23, 24]. In addition, performance of the segmentations relies directly on the adequate selection of two threshold values.

Intensity/color histograms

Histogram change metrics utilize histogrammed values of the pixel data rather than the pixel values themselves. This makes the entire system more robust to noise and small object motion. There exist many different histogram possibilities based on the many different color spaces such as RGB, YUV, HSV, HIS, YIQ, Lab, Luv, Munsell and opponency colors, since the color spaces can all be easily converted from one to another. As such they can be considered equivalent. In practice, simple histogram differencing has shown to be adequate and quite efficient. Performance capabilities have been outlined in [22,23]. Specific examples of histogram techniques are presented in [2, 3, 4]. Note that in general, for histogram techniques, greater improvements in cut detection performance can be attained by making a proper choice for the categorization algorithm than can be attained by using alternate color spaces or by fine tuning the histogram difference functions [21].

Edge based features

The edges of objects between two adjacent frames in a cut cannot usually be found and appropriately put in correspondence. An edge based feature approach was presented in [9] that used the so-called Edge Change Ratio (ECR) and was further refined in [10]. The ECR is defined as number of dilated edge pixels in two adjacent frames that do not conform. Edges are detected using a Canny [30] edge detector, and in order to handle object motions a dilated edge is compared in a windowed area around the pixels rather than a single pixel. Such a method is prone to failure in the presence of very fast camera or object motion, multiple moving objects, moving cameras with moving objects, and occlusions. A comparison of histogram techniques against the edge change ratio technique has shown that the histogram techniques provide similar results without the added complexities [21].

While the ECR methods provide advances in capabilities, especially for fades and dissolves, they suffer due to increased runtimes from the added complexities. A recent review [11] managed to get real time capabilities of the edge feature-based method presented in [10] but only on micro-frames of 88x72 pixels. When the frame sizes were increased to 352x288 (a more standard resolution) the frame-processing rate dropped to approximately 2 frames per second.

Classifying Differences as cuts and non-cuts

Once a metric that quantifies the inter-frame differences has been defined, a classifier is required to separate the differences into cuts and non-cuts. Two basic classification techniques that revolve around thresholding techniques as linear discriminators have been proposed; they are global and adaptive thresholding.

Global threshold

The input to a global thresholding technique is all of the difference values for a given video, which in the ideal case is supposed to show a single large peak at hard cut locations. A hard cut is declared each time the feature difference value surpasses a globally fixed threshold. A common problem of global thresholding is that in practice it is impossible to find a single global threshold that works with all kinds of video material [21].

Adaptive threshold

The input to an adaptive thresholding technique is a windowed subset of difference values for a given video, which in the ideal case is supposed to show a single large peak at cut locations. A hard cut is detected based on the difference of the current feature values with respect to its local neighborhood. Usually a temporal sliding window of size w centered on the current frame is chosen to represent the local neighborhood [21]. A cut is classified when the ratio between the largest and the second largest value in the window surpasses a second threshold [25].

In Figure 4.2, shots are easily seen to be length 2 (664-665) and length 4 (666-669). Both adaptive thresholding techniques in combination with color histogram differences between frames have been shown to lead to higher performance [25,26]. However this type of adaptive thresholding is prone to false negatives in highly edited sequences. We have found that in commercial video sequences, shots of length two, three and four frames are more common than one would expect. Figure 4.2 shows a sequence with several cuts that may be missed when the window in an adaptive thresholding technique is too large.

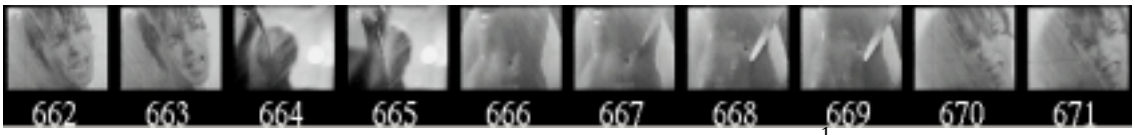


FIGURE 4.2: SEQUENCE FROM THE MOVIE PSYCHO[1].

Feature Tracking for Quantifying Dissimilarity

We propose in this thesis a new approach that uses feature tracking as a metric for dissimilarity. 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 [21]. Effectively, a simple differencing technique is replaced with a more complex one. Experimentally we have

[1] All copyrights © belong to their respective owners. Psycho is an Alfred Hitchcock movie, produced by Shamley Productions and distributed in North America by Universal Pictures.

found that the proposed feature tracking method performs flawlessly on all simple[2] examples where pixel and histogram based methods did not achieve such perfect results.

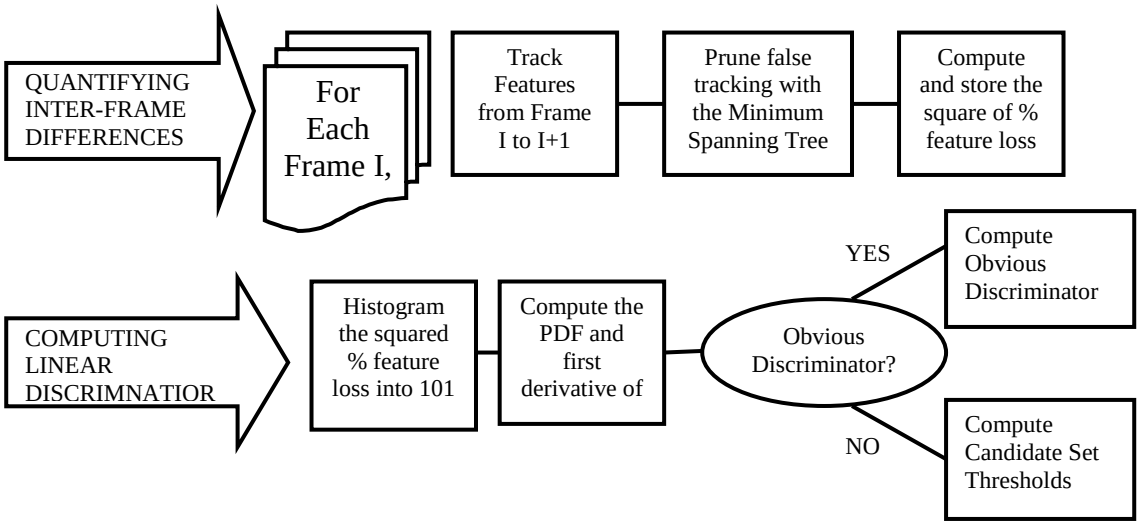


FIGURE 4.3: DIAGRAM OF SYSTEM TO COMPUTE CUTS

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 4.3 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.

Feature Tracking

Previous feature based algorithms [9, 10] 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.

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

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:

$$\text{Luminance} = \text{Red}*0.299 + \text{Green}*0.587 + \text{Blue}*0.114 \tag{4.3}$$

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].

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 [12,13,14].

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:

$$\delta = Dp + t \tag{4.4}$$

where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \tag{4.5}$$

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 the problem of 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 matrix and thus

$$\delta = p + t \tag{4.6}$$

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 a safe computation [13], leaving us with the translation vector being exactly the displacement vector. Complete details for the tracking equations and feature selection can be found in Chapter 2, Section 11.

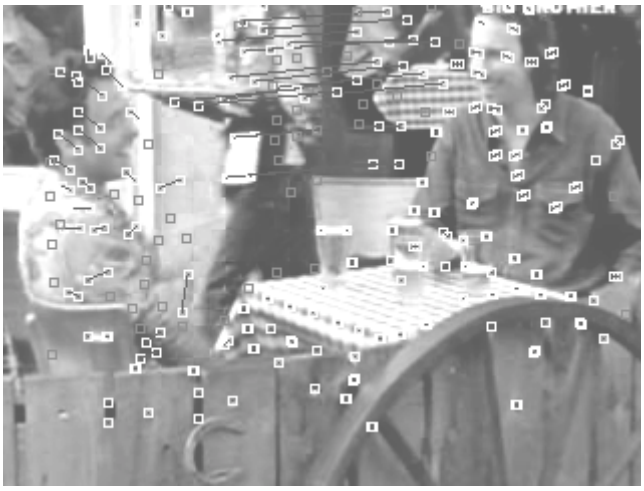


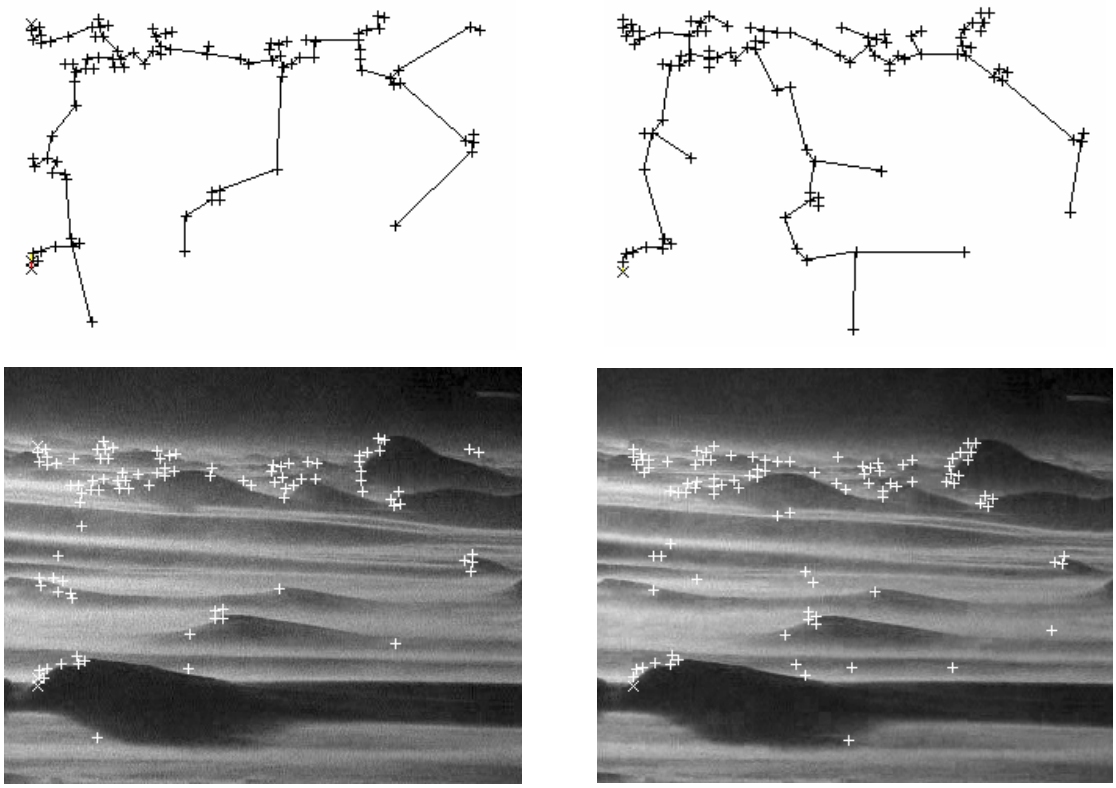
FIGURE 4.4**:** END RESULT FOR FEATURE TRACKING OVER SEVERAL FRAMES.

In Figure 4.4, stationary objects in the foreground and a quickly moving object (noted by the long motion vectors) in the background. Gray squares are lost features; white squares represent the tracked feature and its original position.

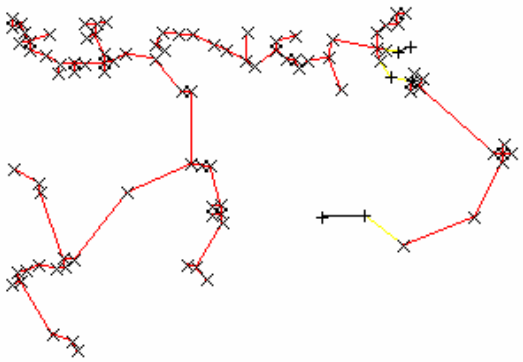
The displacement vector is computed using a pyramid of resolutions because processing a high resolution image is computationally intensive. The multi-resolution pyramid within the feature tracker reduces the resolution of the entire image, say by a factor of 2. Tracking occurs by tracking a feature's general area in the lowest resolution and upgrading the search for the exact location as it progresses back up the pyramid to the highest resolution. An example of tracked feature and displacement vectors is given in Figure 4.4.

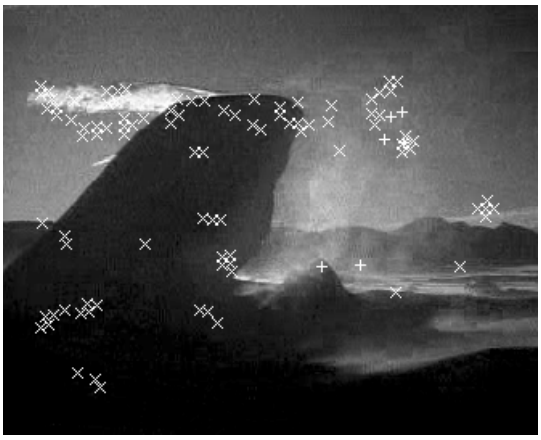
While tracking features it is possible that an extremely large object motion between frames does occur. It has been noticed that in such cases the tracking mechanism begins to fail because the disparity between adjacent frames is too large. The result, features are lost and cannot be tracked any further. 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 by increasing the pyramid dimensions or by removing the identity constraints of the matrix *D*.



(a) (b)





(c)

FIGURE 4.5: 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.

Pruning False Tracking

In the case of a cut at frame i, all features being tracked should be lost from frame i to i+1. 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. We can see from Figure 4.5 a, in the case of a cut, 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 to i+1. 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. In order to further separate the cut set from the non-cut set, we square the percent feature loss which falls in the range [0..1]. This has a beneficial property of ensuring the densities of the cut set and the non-cut set are further separated and thus ease the computation of a discriminating threshold. The idea here is that in the case of optimal feature tracking, non-cut frame pairs score 1 (all features tracked) and cut frame pairs score 0, no features tracked. Squaring, in the optimal case, has no effect as we are already maximally separated. However, in practice, squaring forces the normalized values for non-cut frame pairs closer to zero. Figure 4.6 shows the effect of stretching on the inter-frame differences.

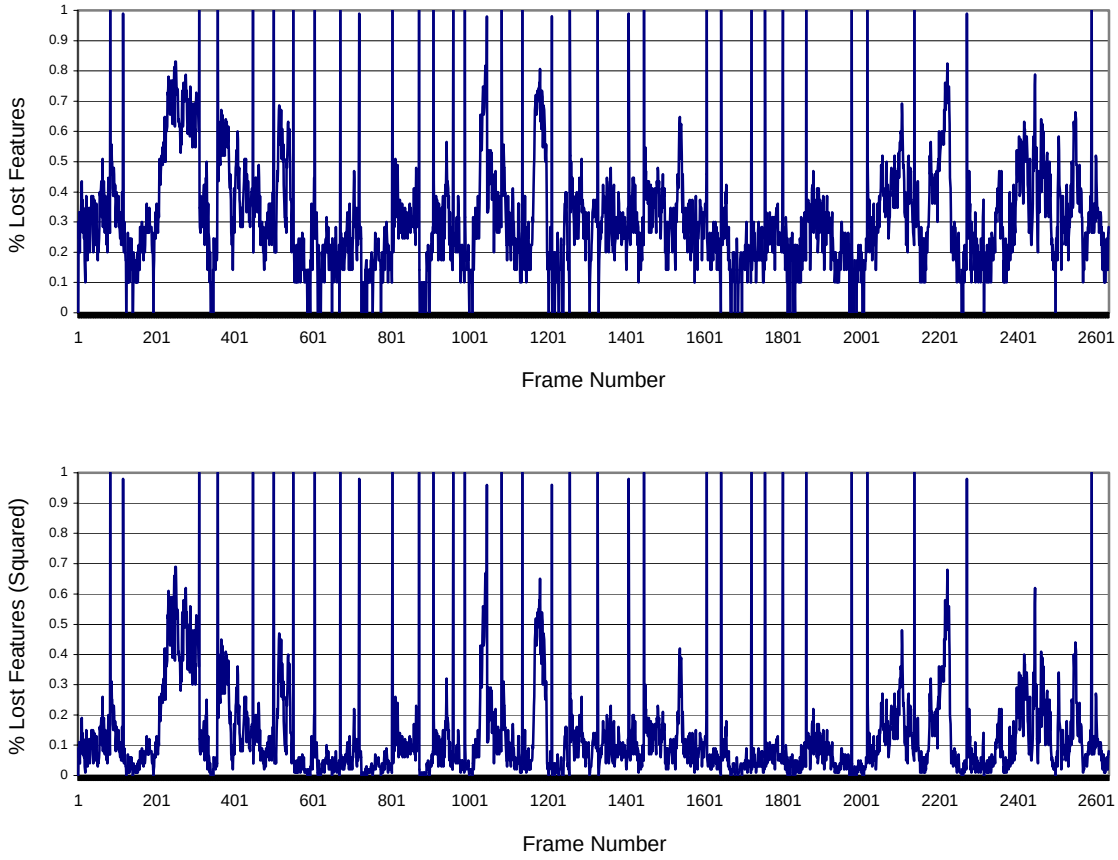


FIGURE 4.6: THE RESULTS OF SQUARING.
Top is the original signal, bottom is the squared signal. The separation between the cuts and the non-cuts has been greatly increased.

4.2 Automatically Determining a Linear Discriminator

Having a difference metric and a method to further separate the cut set from the non-cut set, we can now compute the linear discriminator for the two sets. There is no common threshold[3] that works for all types of video. Next we present an algorithm to auto-select a global threshold. 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 cut set and the non-cut set can be considered to be two separate distributions that should not overlap, however in practice they often do, as illustrated in Figure 4.7b. When the two distributions overlap a single threshold results 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. Until an optimal difference metric is proposed, the problem of optimal determination of the discriminator must be considered.

To avoid the problems illustrated in Figure 4.2 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). In practice, this situation appeared about 50% of the time in our data set. We will introduce the idea of a candidate set in section 4.4, 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.

[3] Note that a threshold is a linear discriminator with the function y = some_value.

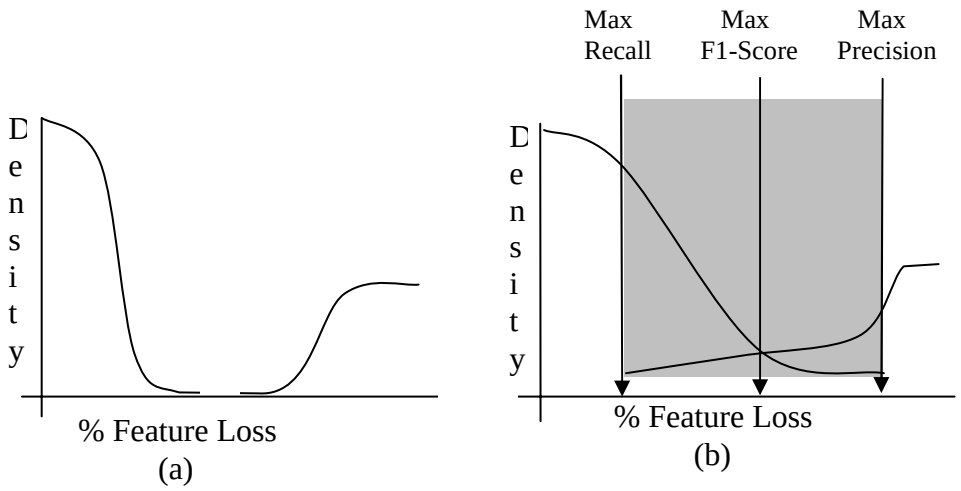


FIGURE 4.7**:** (A) NON-OVERLAPPING DISTRIBUTIONS, (B) OVERLAPPING DISTRIBUTIONS
(a) the discriminator is obvious. (b) the cut set on the right and the non-cut set on the left. The ideal discriminator lies within the overlap region.

4.2.1 Density Estimation

In order to auto-select a threshold, we examine the frequency of high and low feature loss. 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 goal of density estimation is to approximate the probability density function $f(\bullet)$ 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

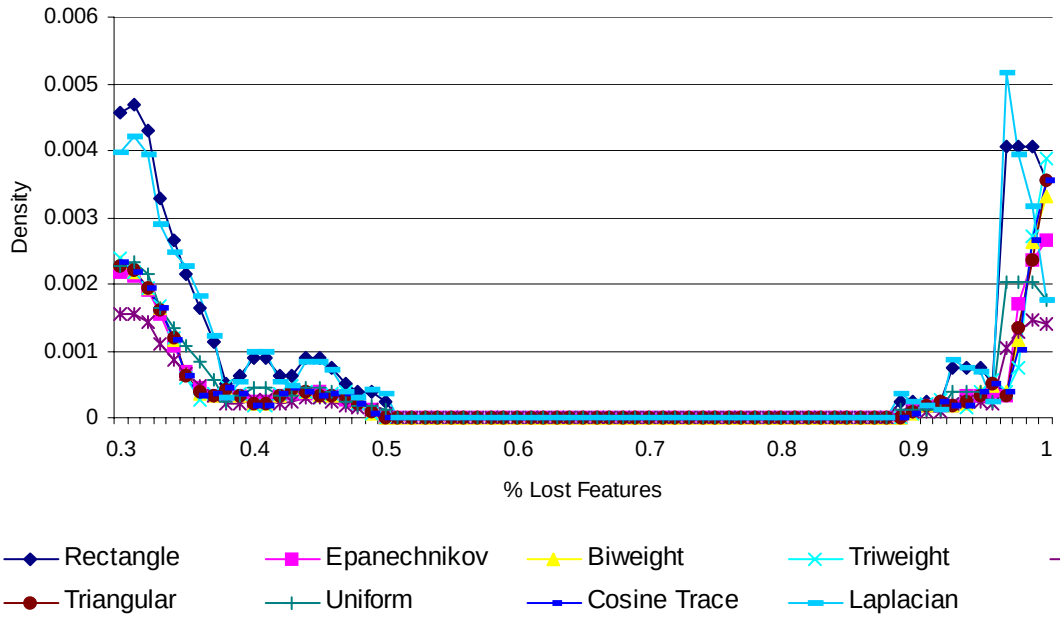
$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x_i - x}{h}\right) \tag{4.7}$$

Where $K(\bullet)$ 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 a variety of kernel functions presented in the past and we performed an empirical evaluation of the 9 kernel

functions listed in Table 4.1 to determine which kernel is the most appropriate for our problem.

Table 4.1: Density Estimation Kernel functions.

Kernel Name	Kernel Function K(α)
Uniform	½ α
Rectangle	α
Epanechnikov	¾ $(1-\alpha^2)$
Biweight	15/16 $(1-\alpha^2)^2$
Triweight	35/32 $(1-\alpha^2)^3$
Gaussian	$\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\alpha^2}$
Triangular	$\lvert \alpha \rvert$)
Cosine Trace	$\frac{\pi}{4}\cos\left(\frac{\pi}{2}\alpha\right)$
Laplacian	$\frac{1}{2}e^{\lvert\alpha\rvert}$



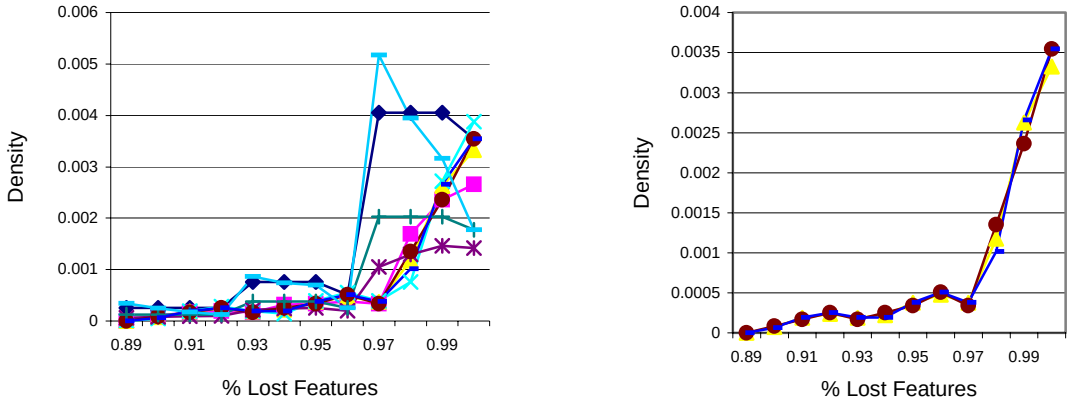


FIGURE 4.8: DETAILED EXAMINATION OF KERNEL FUNCTIONS.
(a) Examination of all 9 kernels for the same video sequence. (b) all nine kernels at the high feature loss areas. (c) Biweight, Cosine Trace, and Triangular kernels

We want a kernel estimator that will facilitate the identification of extrema in the probability density function. In Figure 4.8 we examine each of the kernels in detail to evaluate the effects the smoothing kernel has on each. It is important to select a kernel that does not over smooth, resulting in a loss of discrimination capabilities. As well, we must not select a kernel that under-smoothes, resulting in a ragged signal and thus a misrepresentation of the position of extrema. In the case where the distributions overlap, we determined that the triangular kernel provided the best smoothing properties.

Generally speaking, we found that the Laplacian, Uniform and Rectangular kernels under smoothed the signal, leaving too many extrema for reliable subsequent analysis. We also found that the Gaussian, Triweight, and Epanechnikov kernels over smoothed the signal, making accurate determination of extrema difficult. The remaining 3 kernels, Biweight, Cosine Trace and Triangular appeared to have a very similar effect on the original signals. The triangular kernel was selected from the remaining three because it did not over-smooth locally, making the determination of extrema easiest of the three. In Figure 4.8 we see an analysis of all the kernels for a single data source with non-overlapping distributions. The results presented were consistent across many different samples from our data set. From Figure 4.8 we draw the conclusion that in the case that the two distributions do not overlap, a smoothing of the function will not destroy the obvious discriminating threshold.

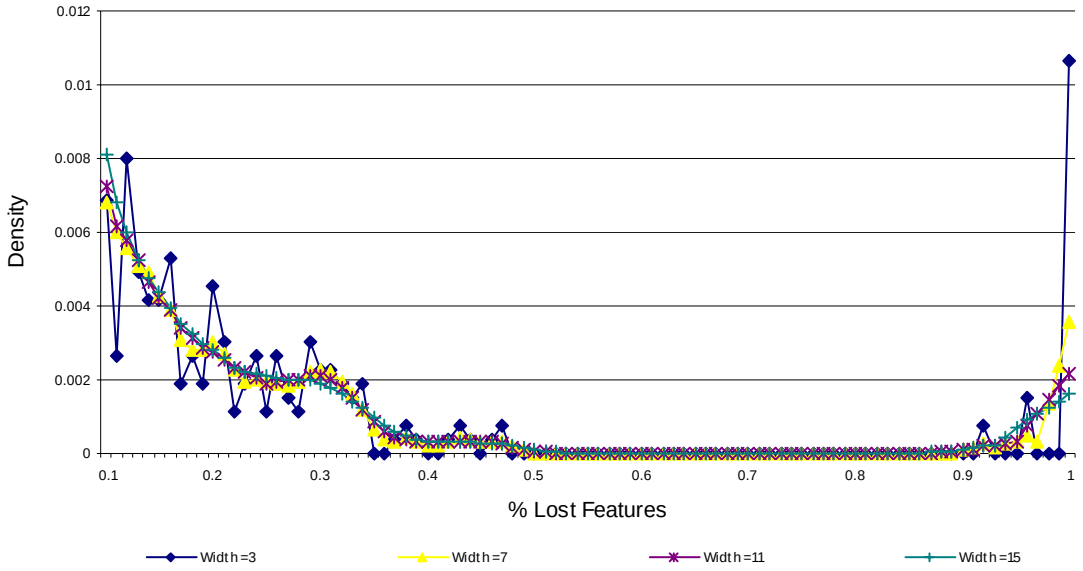


FIGURE 4.9: TRIANGULAR KERNEL AT VARIOUS BANDWIDTHS (WINDOW SIZES)

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 4.9 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.

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 [15]. 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 i 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 $O(h \cdot r/d)$ plus the initial pass to histogram the data being $O(n)$. In total, the number of steps is $O(n) + O(h \cdot r/d)$ where h is our window width (7), the range I is 101 and the bin length is 1. This reduces the number of steps to $O(n) + O(h \cdot r)$. Since n greatly exceeds $h \cdot r$ (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.

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 non-cut set) for the threshold has yielded the correct result on all cases of non-overlapping distributions in our test suite.

The Candidate Sets (Overlapping Distributions)

We now introduce the ideas around what we term the candidate sets. We define 3 candidate sets, where each set contains the frames 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 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 section 4.5. 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 I. These sets have the following property:

- $R \subseteq F \subseteq P$

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:

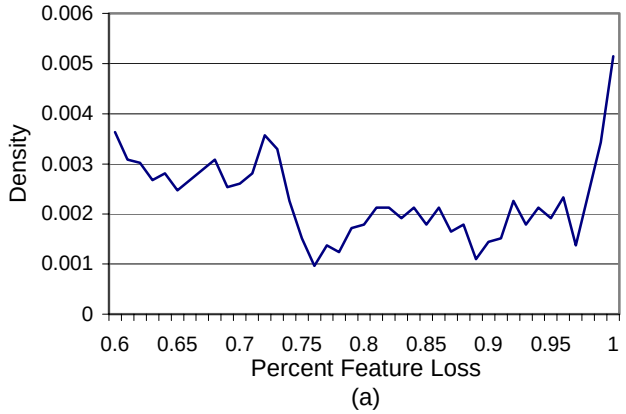
$$\begin{aligned} &G(x) = g(x) + g(x+1): \\ &\quad \text{if } \hat{f}'(x) \le 0 \text{ then } g(x) = 0 \\ &\quad \text{if } \hat{f}'(x) > 0 \text{ then } g(x) = 1 \end{aligned} \tag{4.8}$$

In Figure 4.10, we see the original first derivative function and the modified function. 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.
- The next subsequent zero crossing

The arrows in Figure 4.10(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.



(a)

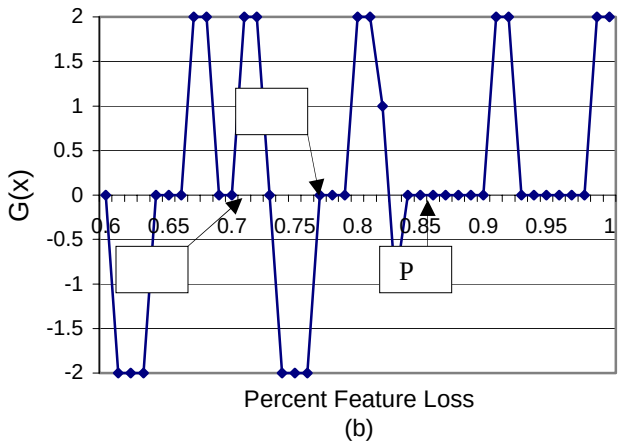


FIGURE 4.10: (A) ORIGINAL PDF (B) MODIFIED FIRST DERIVATIVE (G(X))

Experiments

In this section we perform a variety of experiments on data sources that were deemed to be difficult[4]. We outline our metric for comparing the proposed method against other methods followed by the experimental results. We conclude this section with some information on running time and how feature count and selection will affect the system.

Comparison Metrics

Contingency tables are often used in the quantification of the results categorization systems. Considering the system to be a 2-category classifier (cuts and non-cuts) we can do the evaluation of the effectiveness using a contingency table and its associated statistics. Commonly, precision and recall are used, however we will also address accuracy, error and the so-called F1 score as means of evaluating the effectiveness of the cut detection.

We continue by defining each statistic available for use in our evaluation. All of the statistics are calculated based on a so-called contingency table, where our classifier (cut detector) detects cuts (positives) or non-cuts (negatives). The cut detector can properly detect a cut (true positive), improperly detect a cut (false positive), properly detect a non-cut (true negative), or improperly detect a non-cut (false negative). The

[4] By difficult we mean frames with quick motion, of both the camera and objects, as well as many cuts in short succession.

elements in the set of cuts and non-cuts will never intersect because a frame cannot be double classified as both a cut and a non-cut. Our contingency tables have this form:

	True Cut	True Non-Cut
Classified Cut	True Positive (T^+)	False Positive (F^+)
Classified Non-Cut	False Negative (F^-)	True Negative (T^-)

T^+, F^+, F^-, and T^- are the counts that reflect how the classified categories matched the correct categories. From the contingency table we can compute the following statistics:

Accuracy – This measures the percentage of all decisions that were correct decisions. Range: 0 to 1, with 1 being the best score. It is defined as:

$$Accuracy = \frac{T^+ + T^-}{T^+ + T^- + F^+ + F^-} \tag{4.9}$$

Error – This measures the percentage of all decisions that were incorrect decisions. Range: 0 to 1, with 1 being the best score. It is defined as:

$$Error = \frac{F^+ + F^-}{T^+ + T^- + F^+ + F^-} \text{ or: 1-Accuracy} \tag{4.10}$$

Precision – This measures the percentage of the classified categories that were correct. Range: 0 to 1, with 1 being the best score. It is defined as:

$$Precision = \frac{T^+}{T^+ + F^+} \tag{4.11}$$

Recall – This measures the percentage of the correct categories that were classified. Range: 0 to 1, with 1 being the best score. It is defined as:

$$Recall = \frac{T^+}{T^+ + F^-} \tag{4.12}$$

F1-score – This measures a combination of precision and recall. Range: 0 to 1, with 1 being the best score. It is defined as:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.13}$$

In terms of T^+, F^+, and F^-:

$$F1 = \frac{2 \times T^+}{2 \times T^+ + F^+ + F^-} \tag{4.14}$$

The F1 score is the only statistic that is worth trying to maximize on its own. Perfect precision can be achieved by never detecting a cut and perfect Recall by always

detecting a cut. A truly accurate system will assign the correct categories and only the correct categories, maximizing precision and recall at the same time, and therefore maximizing the F1 score.

Attempting to maximize the accuracy score, thus minimizing error, is an inappropriate measure in the case because the size of the non-cut set is so large in comparison to the cut set. Simply declaring all frames as non-cuts will result in a high accuracy. Any metric for cut detection that uses the True negative number in its evaluation is not a good indicator of quality because of the distribution of cuts to non-cuts in video data. This is casually confirmed if we consider the redundancy that video data contains.

In [28], the authors suggest an alternate computation for accuracy as:

$$\text{Accuracy} = 1 - \frac{F^-}{T^+ + F^-} - \frac{F^+}{T^+ + F^+} \tag{4.15}$$

With some algebraic manipulation, we can see their definition of accuracy is simply

$$\text{Accuracy} = \text{Precision} + \text{Recall} - 1 \tag{4.16}$$

As Matter and Robinson's metric is a function of precision and recall, we will omit using it and rely on the more standard metrics of precision, recall and the F1 score.

Experimental Results

To start, we perform a set of experiments to compare the proposed method against a histogram-based method, specifically 'cutdet' from the MOCA project [66]. We used the precompiled version of cutdet and treated the internals as a black box. We ran a sequence through cutdet with various threshold settings to determine its characteristics. We examined the precision, recall, and F1 score. The sequence is from a television show and the quality of the capture is quite high. The action and motion is not extreme, and the colours are vibrant and distinct. It was assumed, based on the ideas behind histogram comparisons, that this sample would highlight the capabilities of the cutdet system. In Figure 4.11 we see that the F1 score has platitude at threshold 0.45 which indicates the best threshold for cutdet on this particular sequence. The exact values of precision, recall and F1 score are: 1, 0.941, and 0.969 respectively. In the case of perfect detection,

precision, recall and the F1 score will all be 1. As the F1 score hit a platitude at 0.969, the cutdet method would be unable to achieve a perfect score in this example.

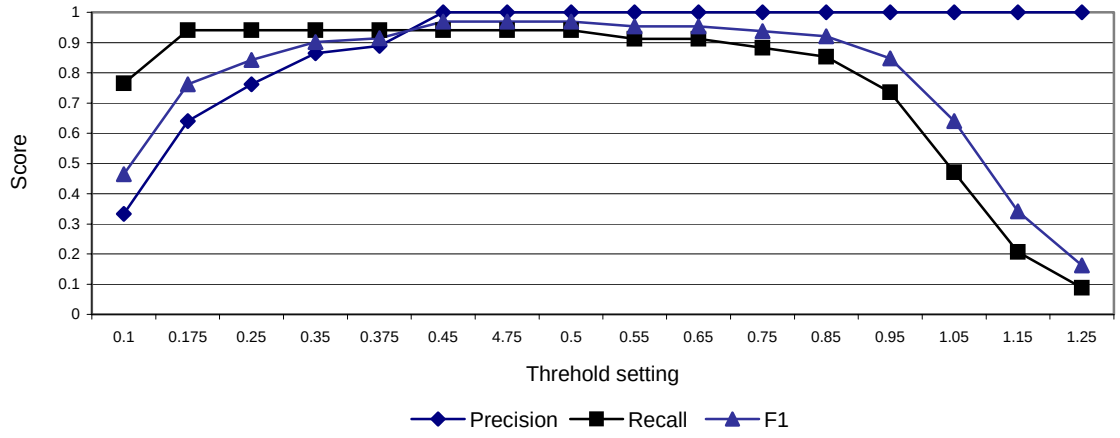


FIGURE 4.11: GRAPH OF PRECISION, RECALL AND F1 SCORES FOR A SEQUENCE RUN THROUGH THE CUTDET SYSTEM.

When the proposed method was run against the same sequence, it received a perfect detection rate. The clear selection of threshold can be seen in Figure 4.12 as the feature tracking was clearly working very well and the separation between the cut set the non-cut set is obvious due the large space of zero density between the cut and non-cut sets.

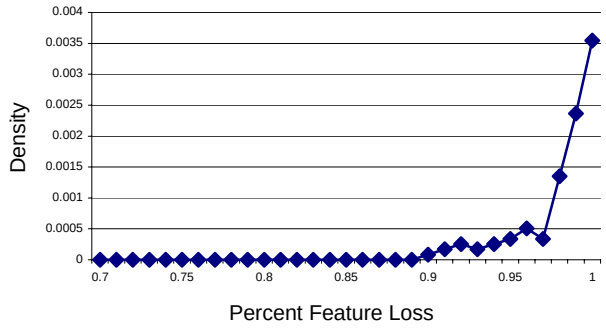


FIGURE 4.12: DENSITY ESTIMATION GRAPH.

In the experiment that follows, a selection of video clips that represent a variety of different video genres are used for cut detection. In Table 4.2, we present the data set with explanation why it is being used in the experiments. We compare the results of the proposed method against a pixel-based method with localization information and a histogram based method (specifically cutdet, from the MOCA project [27]). The localization information in the pixel based method relies on the statistical improbability

that the positions of pixel values will remain the same over a cut frame pair. In this experiment, we are attempting to maximize the F1 score. For the proposed method, we ran each sample through the system once and computed the F1 candidate set threshold. For cutdet, and the pixel based method we ran 13 different thresholds and computed the precision, recall and F1 scores for each of the runs. The best F1 score was selected for comparison. This effectively required the two methods to be executed 13 times in comparison to the 1 time for the proposed method.

In Table 4.3, we present the results of running the 3 methods on the dataset. The proposed method outperforms both the histogram-based method and the pixel based methods. In most cases (8 of 10) the proposed method provides the maximal F1 score. A simple statistical analysis of the overall capabilities is given at the end of Table 4.3. The average, variance and standard deviation for the 10 samples were computed. On average, the proposed method significantly outperforms the other two methods. The variance and the standard deviation show that the results offered by the proposed method are also more stable across a variety of different video genre.

Table 4.2: Experimental data set

Source Label	**Characteristics of video data**	**Genre**
A	Cartoon clip. Substantial object motion.	Cartoon
B	Substantial object motion. This clip is taken from a film where a blue filter was used to simulate low lighting conditions.	Action
C	Black and white movie. Substantial action and motion. Many close proximity cuts. This clip is the murder scene from the movie psycho.	Horror
D	High quality digitization of a television show.	Drama
E	Low quality digitization of a television show.	Sci-Fi
F	Commercial, no cuts, quick motion, many production effects. Meant to show that dissolves are not mistakenly classified as cuts.	Commercial
G	Commercial sequence from the MOCA Project	Commercial
Q	Video abstract from the MOCA Project	Comedy/Drama
I	News Sequence from the MOCA Project	News/Documentary
J	Trailer for a film. This clip has many computer generated features, many close proximity cuts. Trailer for the movie Lawnmower Man.	Trailer/Sci Fi/Action

Table 4.3: Results on data set.

	Proposed feature tracking method			**Pixel Based method with localization**			**Histogram MethodCut Det (MOCA)**		
Data Source	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
A	1	1	1	1	1	1	1	1	1
B	1	1	1	.825	.825	.825	1	.375	.545
C	.595	.870	.707	.764	.778	.771	.936	.536	.682
D	1	1	1	1	1	1	1	.941	.969
E	.938	1	.968	.867	.867	.867	.955	.700	.808
F	1	1	1	0	0	0	1	1	1
G	.810	.944	.872	.708	.994	.809	1	.667	.800
H	.895	.895	.895	.927	1	.962	.971	.895	.932
I	1	1	1	1	1	1	1	.500	.667
J	.497	.897	.637	.623	.540	.591	.850	.395	.540
AVG	**.874**	**.961**	**.908**	**.774**	**.800**	**.783**	**.971**	**.701**	**.794**
VAR	**.034**	**.003**	**.018**	**.090**	**.101**	**.093**	**.002**	**.060**	**.036**
STD. DEV	**.185**	**.054**	**.134**	**.301**	**.318**	**.304**	**.048**	**.246**	**.190**

It is not surprising that ‘cutdet’ out performs the proposed system in H, because the abstract was created by the MOCA project from which cutdet originates. However, it is surprising that the pixel based method outperformed both. In examples C and J, the F1 score was not maximized as the heuristic to determine the F1 candidate set threshold did not achieve the best value, rather a good value. Within the range of the F1 candidate set threshold plateau, maximum F1 was achievable.

Processing Speed

For all the experiments given in Table 4.3, we tracked 100 features with a minimum inter-feature distance of five pixels. The processing time for a frame pair is approximately 70 milliseconds on a 2.2 GHz Intel processor on frames sized 320x240. This is significantly faster than [66] without a loss in granularity. Unlike the pixel and histogram based methods, the running time of the tracker is not a function of the size of the video frames and remains constant regardless of the video size. Tracking over an n-

frame interval can cut this time by a factor of n. By skipping every other frame, we still maintain the 70ms processing time per frame pair, but have reduced the number of frames by a factor of 2. By increasing the number of frames skipped, we decrease the frame accuracy to which we can report, and risk losing tracking context in the presences of large object and camera motions. We ran several data samples through the system set to skip one and two frames between tracking. As expected the running times reduced to approximately half and a third respectively. However, we experienced a reduction in the F1 scores in half the samples and all the samples for skipping one and two frames respectively. Playing with the tracking parameters may reduce the error, but is beyond the scope of this work.

The Effect of Feature Selection

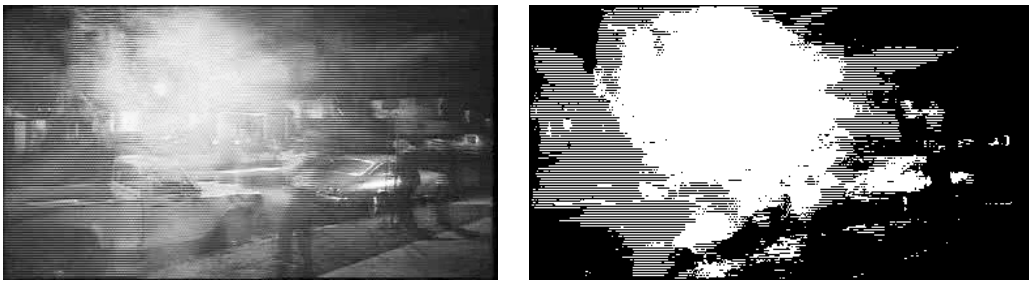
We have chosen the number of features to track to be 100. However it should be noted that the optimal number of features to track has yet to be determined, if at all possible. Logically, there must be enough features to track and ensure a certain amount of coverage of the frame, however selecting too many features will result in features that are less than ideal being selected for tracking and therefore increase the likelihood that those features will not be reliably tracked from frame to frame. We have performed some rudimentary experiments that show as the number of features selected to be tracked decreases, and thus the overall quality of those features (for tracking purposes) increases, the percentage of lost features in a cut situation increases overall. Specifically, when the number of features tracked decreases, the threshold that is automatically selected approaches 100 on simple sequences. However, when the number of features tracked decreases, the system becomes more susceptible to object motion, occlusions and very quick camera motion because we lack spatial coverage of the frame, therefore reducing the accuracy on more difficult sequences.

Another selection option that needs consideration is how wide the non-maxima suppression window should be. This option is effectively selecting the best corner feature within a given radius. By selecting a radius that is too large will result in features being selected that are less optimal for tracking while a radius that is too small results in feature clustering that is prone to loss due to object motion and occlusions. We have

found empirically that a radius of five pixels provides good results; a radius of fifteen pixels provides worse results.

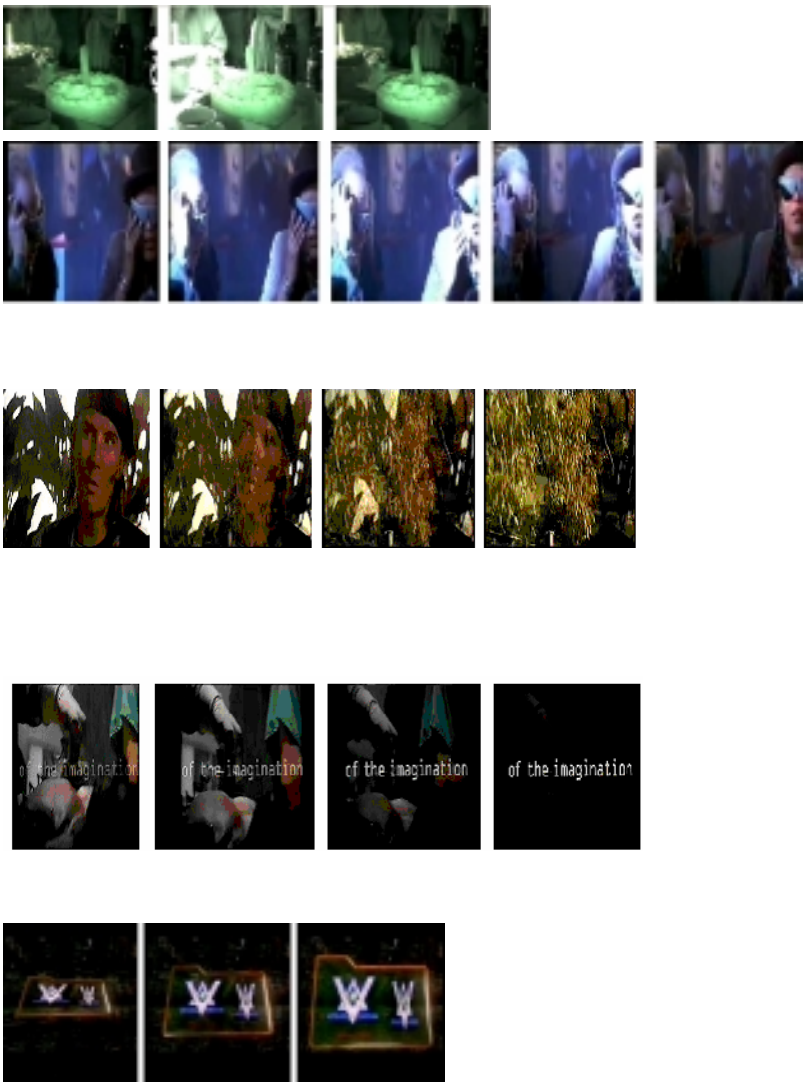
Known Problem Areas

There are some very clear restrictions on this feature-based method for detecting cuts. Primarily, the quality of the digital video plays a fundamental role because we are tracking fine-grained features (corners) rather than coarse-grained features such as edges, color blobs, and histograms. Overly noisy digitization will result in the feature tracking having a difficult time properly tracking selected features and will result in a higher overall percentage loss of tracked features. This will result in an overlapping of the cut set distribution and the non-cut set distribution and therefore result in an overall lower accuracy. As well, dropped frames in the digitization process may result in unnatural spaces between frames. This will also cause the feature tracker to potentially lose features and categorize a cut. To complicate the matter, it is not clearly defined whether or not these situations should be classified as cuts. Other anomalous glitches in the digitization process will result in feature loss as well. The figure below illustrates digitization artifacts that can cause problems for our fine-grained feature tracker because of the high gradient changes on the digitization scan lines.



(a) (b)

FIGURE 4.13: DIGITIZATION ARTIFACTS
(a) original frame (b) exemplary amplification of artifact errors. Notice the lines around the explosion.



a) Sudden flash, causing an abrupt change in luminance, improperly declared a cut.

b) A longer sequence of changed illumination, returning to original illumination.

c) A quick dissolve. Changes in luminance declared as cuts, or fade to black because the large change in pixel values caused the tracking systems correlation checks to fail.

d) A quick dissolve over two frames. Often declared as two consecutive cuts because the pixel changes cause the tracking systems residual checks to fail.

e) Computer generated graphics that move sharply. There is also little texture allowing inferior selection of features for tracking.

FIGURE 4.14: EXAMPLES[5] OF PROBLEM SITUATIONS.

Each of the examples in Figure 4.14 represents problems that the proposed cut detection method faces. Because we are tracking features in the luminance space, we are subject to feature loss when large changes in brightness occur. The system is robust to gradual dissolves, however high speed dissolves (over a small number of frames) cause problems because of the large the number of pixels that change greatly. This results in the tracking windows being very different and the correlation computations result in residuals that are simply too large. Finally, we have noticed in several cases that

[5] All copyrights © belong to their respective owners. Lawnmower Man is a Brett Leonard movie, produced and by distributed in North America by New Line Cinemas.

problems occur with computer-generated video data. This is usually due to the fact that the objects in computer generated video move at rates that are unusually quick. Take for example the 3 consecutive frames from Figure 4.14, part e: The motion exceeds what a normal object videotaped at 30 frames per seconds (often because animation is at 10 frames per second), and failure occurs.

Conclusions and Discussion

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. Seyler, “Probability distribution of television frame difference”, Proc. Institute of Radio Electronic Engineers of Australia 26(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 Systems. 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 Con-ference on Multi media, pp 267-272, 1993

[6] H. Zhang, A. Kankanhalli, and S. Smoliar, “Video parsing using com-pressed 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 Proceedings of 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 Detect-ing and Classifying Scene Breaks", Proc. ACM Multimedia, pp. 189-200, 1995
[10] R Zabih, J. Miller, and K. Mai,. “A Feature Based Algorithm for detect-ing and Classifying Production Effects”, Multimedia Systems, Vol 7, p 119-128, 1999.
[11] A Smeaton et al., “An Evaluation of Alternative Techniques for Auto-matic 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-91-132, 1991
[14] Jianbo Shi and Carlo Tomasi. “Good Features to Track”. IEEE Confer-ence on Computer Vision and Pattern Recognition, pp 593-600, 1994.
[15] W. Hardle and D. Scott. “Smoothing in by weighted averaging using rounded points”, Computational Statistics Vol. 7: 97-128, 1992.
[16] C. Harris and M. Stephens.”A combined corner and edge detector”. Proceedings of the 4th Alvey Vision Conference, pp 147--151, 1988.
[17] R. Lienhart. Dynamic Video Summarization of Home Video. SPIE Storage and Retrieval forMedia Dat bases 2000, Vol. 3972, pp. 378-389, Jan. 2000.
[18] R. Lienhart, S. Pfeiffer, and W. Effelsberg. Video Abstracting. Commu-nications of the ACM, Vol. 40, No. 12, pp. 55-62, Dec. 1997.
[19] M. M. Yeung and B.-L. Yeo. Video Visualization for Compact Presen-tation and Fast Browsing of Pictorial Content. IEEE Transactions on Circuits and Systems for Video Technology,Vol.7,No. 5, pp. 771-785, Oct. 1997.
[20] A. Hampapur, R. Jain, and T. E. Weymouth. Production Model Based Digital Video Segmentation. Multimedia Tools and Applications, Vol.1, pp. 9-45, 1995.
[21] 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.
[22] U. Gargi, R. Kasturi, S. H. Strayer. Performance Characterization of Video-Shot-Change Detection Methods. IEEE on Circuits and Systems for Video Technology,Vol.10,No.1,Feb. 2000.
[23] G. Lupatini, C. Saraceno, and R. Leonardi. Scene Break Detection: A Comparison. Research Issues in Data Engineering, Workshop on Con-tinuos Media Databases and Applications, pp. 34-41.1998.
[24] B. Shahraray. Scene Change Detection and Content-Based Sampling of Video Sequences. SPIE Digital Video Compression, Algorithm and Technologies, Vol. 2419, pp. 2-13, 1995.
[25] B.-L.Yeo and B. Liu. Rapid Scene Analysis on Compressed Video. IEEE Transactions on Circuit and Systems for Video Technology,Vol.5,No.6,Dec.1993.
[26] M. M. Yeung and B.-L. Yeo. Video Visualization for Compact Presen-tation and

Fast Browsing of Pictorial Content. IEEE Transactions on Circuits and Systems for Video Technology,Vol.7, No. 5, pp. 771-785, Oct. 1997.

[27] S. Pfeiffer, R.Lienhart, G. Kühne, W. Effelsberg. The MoCA Project - Movie Content Analysis Research at the University of Mannheim. In-formatik '98, pp. 329-338, 1998.

[28] J. Mateer, J. Robinson, Semi-Automated Logging for Professional Media Applications. Video, Vision and Graphics (VVG) 2003, Bath, UK, July, 2003.

[29] A Whitehead. Fast Feature Based Video Segmentation and Annotation. Proc. 7th International Symposium on Signal Processing and its Appli-cations (ISSPA), Paris, 2003.

[30] J. Canny A Computational Approach to Edge Detection, IEEE Transac-tions on Pattern Analysis and Machine Intelligence, Vol 8, No. 6, Nov 1986.