A framework for robust online video contrast enhancement using modularity optimization

Anustup Choudhury*, *Student Member, IEEE,* and Gérard Medioni, *Fellow, IEEE*

***Abstract*—We address the problem of video contrast enhancement. Existing techniques either do not exploit temporal information at all or do not exploit it correctly. This results in inconsistency which causes undesirable flash and flickering artifacts.**

Our method analyzes video streams and clusters frames that are similar to each other. Our method does not have omniscient information about the entire video sequence. It is an online process with a fixed delay. A sliding window mechanism successfully detects shot boundaries "on-the-fly" in a video. A graph-based technique called "modularity" performs automatic clustering of video frames without a priori information about clusters. For every cluster in the video, we extract key frames belonging to each cluster using eigen analysis and estimate enhancement parameters for only the key frame , then use these parameters to enhance frames belonging to that cluster, thus making our method robust.

We evaluate the clustering method on video sequences from the TRECVid 2001 dataset and compare it with existing methods. We show reduction of flash artifacts in enhanced videos. We show statistically significant improvement in perceived video quality and validate that by conducting experiments on human observers. We show application of our clustering process to perform robust video segmentation.

***Index Terms*—Video enhancement, Modularity optimization, Shot detection, Human Validation, Video segmentation.**

I. Introduction

CONSISTENTLY modifying video is a ubiquitous problem. For instance, low vision people benefit from viewing contrast enhanced videos [1]. Existing techniques for video contrast enhancement either trivially enhance each frame of the video, use a fixed number of frames to do temporal smoothing, or assume slow changing light intensity while performing enhancement. We argue that enhancing individual frames of the video does not exploit temporal information, which results in temporal incoherence causing flash and flickering artifacts. Also, using a fixed number of frames is undesirable because the method does not account for illumination changes or scene changes resulting in improper enhancement. Slowly changing light intensity may be an unreasonable assumption when dealing with videos with diverse content.

Our method analyzes a video stream and uses a recent image color contrast enhancement technique [2] to enhance videos. Our framework can be extended to any other image contrast enhancement technique. Successive frames of a video generally convey similar information, unless there is a shot change. We use this information to maintain temporal consistency. In order to account for the challenges mentioned earlier, we propose a novel way to achieve video contrast enhancement. A video sequence, particularly in movies, consists of many shots. In order to account for the shots, we first cluster frames that are similar to each other. Next we select a key frame that is most representative of all frames in that cluster and estimate the enhancement parameters for only that key frame. Then, we enhance all frames in a cluster with the estimated enhancement parameters of its key frame. Therefore, clustering is critical to video contrast enhancement and helps in reducing flash and flickering artifacts. This idea is quite intuitive and as can be seen in Section II, where we review previous work in video enhancement, none of the existing methods use this idea for video enhancement. We also show the effectiveness of our enhancement method by conducting experiments on human observers.

A lot of work has been done in video clustering and a review of some existing methods can be found in [3], [4], [5], [6]. In order to cluster frames of the video, we perform graph based partitioning. The graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is a weighted undirected graph where the vertices $\mathbf{V}$ correspond to the frames and edges $\mathbf{E}$ are formed between the vertices. The weight of an edge, $w(i, j)$ is the similarity measure between vertices i and j. Most of the previous graph based clustering techniques such as normalized cut [7] and techniques based on spectral clustering [8] minimize the size of the cut to produce two disjoint clusters. Recently, these techniques have been used in video clustering [9], [10], [11]. However, these methods require prior information about the number of clusters, which is not always available and varies largely across different videos. Eigengap [12] can be used to predict the number of clusters [9], [13], [14]. However, this measure is not robust. It is critical for the clustering approach to have a high recall rate because a missed detection of a cut or a gradual scene change may cause improper enhancement.

In this paper, we introduce a new approach for graph-based video clustering using *modularity* maximization. This method was introduced by Newman [15] and extensively used in community detection and network analysis [16], [15]. This method finds clusters such that the number of edges between clusters is *smaller than expected*. To the best of our knowledge, this *modularity* maximization algorithm has not been used for shot detection and a key advantage is that it can automatically find clusters in a graph without a priori information about the cluster count or size. Using eigen analysis, we can robustly find the most influential vertex in a cluster that is equivalent to finding the key frame in a sequence. As spectral clustering techniques minimize the cut, there is no intuitive way to extract

A. Choudhury and G. Medioni are with the Department of Computer Science, University of Southern California, 3737 Watt Way, PHE 204, MC-0273, Los Angeles, CA, 90089. E-mail: {achoudhu*, medioni}@usc.edu

the key frame using eigenvector analysis. Since this method has not been earlier used for clustering, we evaluate it by conducting experiments on video sequences from TRECVid 2001 dataset [17] and compare it with existing approaches.

Most existing video clustering techniques have omniscient information about the video thus making it an offline process. An online process for video clustering [3] detects shot using change in curvature of the similarity measure and detects the key frame as the middle frame amongst the frames detected so far in a shot, which may not be the most representative frame. Another online clustering method [18] calculates the rank of the feature matrix of a window of frames using SVD and chooses a frame as the key frame if its rank is more than a threshold. Since this threshold is fixed, it will not scale to videos with diverse content such as movies, resulting in improper estimation of the key frame. Since we want an online system and a shot can be described by a consecutive window of frames, we use a sliding window mechanism to detect shots and transfer clustering information of a fraction of the last frames from the previous window to the current window.

The rest of the paper is organized as follows: In Section III, we describe the details of the proposed method. We discuss details regarding similarity measure, cluster analysis, key frame detection and information transfer across sequences to maintain temporal consistency. In Section IV, we show the results of cluster analysis and key frame detection on video sequences and compare with existing methods. We also show how this analysis benefits video enhancement and video segmentation. Finally, we conclude our paper in Section V.

II. Previous Work

The technique by Fullerton and Peli [1] enhances MPEG videos one frame at a time. Hines *et al.* [19] use a DSP implementation of the retinex algorithm to enhance video on a frame by frame basis. However, these methods do not ensure temporal consistency. We ensure temporal consistency by enhancing similar frames with same enhancement parameters.

Goh *et al.* [20] use a weighted histogram in which the weights depend on the comparison of the current frame histogram with the previous frame histogram. Since only 2 frames are considered at a time and the histogram may change for every frame, it is not temporally consistent. Wang and Ward [21] use a weighted histogram where the weights depend on a fixed number of frames. This will ensure smoothing (less flicker) if all frames belong to one shot of a video but will fail in case of scene changes. The technique by Kang *et al.* [22] extends the work of Reinhard *et al.* [23] to videos. For enhancement of images, Reinhard *et al.* have defined a term called **key** that indicates if a scene is subjectively light, normal or dark. Kang *et al.* assumes the videos to have a slowly changing scene intensity and so consider the value of **key** to be a constant. They also assume a fixed number of frames to maintain temporal coherence. But this is not necessarily the case in most videos because of scene changes and shots having different lengths. Ramsey *et al.* [24] adapt the value of **key** depending on luminance changes, but they too use a fixed but small number of frames to maintain temporal consistency. Wang *et al.* [25] consider temporal information to avoid flash and flickering effects. However, their technique uses a specially designed camera to capture HDR video. Bennett and McMillan [26] use the bilateral ASTA filter (Adaptive Spatio-Temporal Accumulation) that filters the image depending on the detection of motion and it favors more spatial filtering over temporal filtering when motion is detected. However, the parameters have to be set manually in case of a change in the lighting conditions. The advantage of our method is that during video enhancement, it deals with scene changes including illumination changes using a clustering mechanism and can automatically detect and accordingly enhance arbitrary sized shots without additional hardware.

Pattanaik *et al.* [27] mimics the temporal adaptation of the human visual system to illumination changes. This adaptation is gradual, and it takes longer if one moves from a bright lighting place to a dark place than if one moves from a dark place to a bright place. As a result, multiple frames are needed to adjust to the overall luminance changes resulting in undesirable information loss during the adaptation phase, that is not present in our method.

III. Our Method

Our method is described in Algorithm 1. We first measure the similarity between frames (Section III-A). In order to cluster similar frames, we use the *modularity* measure (Section III-B) and extract a key frame from that cluster (Section III-C). We want an online system and therefore consider an incoming window of n frames at a particular time. Since successive windows can describe the same scene, in order to maintain temporal consistency we employ a sliding window mechanism to transfer information about the clustering index and the key frame, K to subsequent windows (Section III-D). We denote the overlap between windows by P and use a third of the last n frames for overlap ($p = n/3$). Finally we perform video enhancement (Section III-E).

A. Similarity Measure

Given a window of n frames, we first compute the illumination (lighting conditions) and the reflectance component (properties) of each frame as described in the enhancement method proposed in [2]. We construct a weighted fully connected undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. The vertices, $\mathbf{V}$, are the frames in the feature space, and the weight of the edges, $w(i, j)$, connecting two vertices is the similarity measure between frames i and j. In order to account for lighting and scene variations, we use two different descriptors: a color histogram to account for illumination variations and an edge direction histogram to account for edge variations. We use two different descriptors to consider both the color properties as well as the structural properties of the frame. The color histogram is computed for the illumination image and is composed of 64 bins that are determined by sampling colors in the RGB color space. The edge direction histogram is computed for both the illumination and reflectance images. We use Sobel filters on the images to obtain vertical and horizontal gradients of the image. These gradients are used to compose the edge direction

histogram with 72 bins, where each bin corresponds to an interval of 2.5^o.

Algorithm 1 Algorithm (Pseudo-code) of our method

Input: Streaming Video; A window, W having n frames (F_i) and corresponding cluster labels $tempC$; Size of overlap window p; Overlap window $P \subseteq W$ with corresponding cluster labels CP

Initialize: $C = \phi$,$W = \phi$,$t = 1$,$CP = \phi$

Output: Enhanced frames and cluster labels C of all frames

while there are frames to be processed **do**
  if $CP == \phi$ **then**
    $W = [F_t, \ldots, F_{t+n-1}]$
    $tempC = \phi$
  else
    $W = [P, F_{t+n}, \ldots, F_{t+2n-p-1}]$
    $tempC = [CP \quad \phi]$
    $t = t + n - p$
  end if
  Cluster frames in W depending on similarity measure
  Assign cluster labels $tempC(j)$'s such that $W(j)$'s belonging to same cluster have same label and those belonging to different clusters have different labels {This ensures information transfer across different clusters using CP}
  $tW(j) = W(j)$ and $tC(j) = tempC(j)$ where $j = [1, \ldots, n-p]$
  for each unique cluster label c of tC **do**
    if $c \notin C$ **then**
      Find key frame, K amongst frames in tW corresponding to cluster label c
      Estimate enhancement parameters of K
      Save these enhancement parameters for c
    else
      Retrieve enhancement parameters for c
    end if
    Enhance frames in tW with label c with corresponding enhancement parameter
  end for
  print $C = [C \quad tC]$
  $P = [F_{t+n-p}, \ldots, F_{t+n-1}]$
  $CP(j-n+p) = tempC(j)$ where $j = [n-p+1, \ldots, n]$
end while

In order to compare the histograms, we compute the difference between the histogram. For the color histogram, we compute the histogram intersection [28] $d_c(i,j)$ defined as

$$d_c(i,j) = \frac{\sum_{b=0}^{63} min(c_i(b), c_j(b))}{\sum_{b=0}^{63} c_i(b)}, \tag{1}$$

where i and j are the frames from window W, c_i and c_j are the respective color histograms and b is the bin number. The intersection is normalized by the number of pixels to get a value between 0 and 1. If the frames are similar, then the value of $d_c(i,j)$ will be high (close to 1). Otherwise, if the frames are dissimilar, then the value of $d_c(i,j)$ will be low (close to 0). To compare the edge direction histogram, we compute the complement of the Euclidean distance between the histograms $d_e(i,j)$ defined as

$$d_e(i,j) = 1 - \sqrt{\sum_{b=0}^{71} (e_i(b) - e_j(b))^2}, \tag{2}$$

where i and j are the frames from window W, e_i and e_j are the respective normalized edge histograms and b is the bin number. For similar frames, the Euclidean distance between the histograms will be low. Therefore, if the frames are similar, then the value of $d_e(i,j)$ will be high (close to 1). Otherwise, if the frames are dissimilar, then it will be low (close to 0).

We tried different combinations of difference measures (such as Euclidean distance, histogram intersection, cosine distance etc.) to compare the histograms. We chose histogram intersection for the color histogram and Euclidean distance for the edge direction histogram because experimentally, we found these measures to be the best interpretation of similarity measure. This combination best accentuated dissimilarities to account for scene changes since amongst the ones we tried, it gave the highest value for frames with similar content and lowest value for frames with different content. We calculate the similarity measure, $w(i,j)$ between frames by combining the similarity measure of the histograms as shown:

$$\begin{aligned} w(i,j) \quad = \quad & d_c(i,j) d_{e_{illum}}(i,j) + d_c(i,j) d_{e_{refl}}(i,j) \\ & + d_{e_{illum}}(i,j) d_{e_{refl}}(i,j), \end{aligned} \tag{3}$$

where $d_c(i,j)$ is the color histogram similarity measure between illumination component of frames i and j, $d_{e_{illum}}(i,j)$ is the edge similarity measure between illumination component of frames i and j and likewise $d_{e_{refl}}(i,j)$ is for the reflectance component. All the distance measures are normalized to have a value between 0 and 1. Since most of the color properties of the frames are present in the illumination component, we do not compute the color histogram similarity measure for the reflectance components. However, edge information is present in both the illumination and the reflectance components of the image.

We used 2 different descriptors - color histogram and edge histogram. Using only one descriptor is not sufficient to describe content completely. For example, if we use only a color histogram descriptor then frames with similar color content but different in other attributes will be considered similar. This is undesirable. Therefore, we need to combine multiple feature attributes. One way to do this is to weigh the features. However, this technique is cumbersome and may involve sophisticated machine learning tricks to come up with the right combination of weights. We instead propose a simpler alternative as shown in Equation 3. We weigh the similarity measure of each feature with that of every other feature. This helps us to get high similarity values if each of the similarity measures have high values. Thus we need a good similarity score from all 3 features to be confident of the frames being similar.

B. Cluster Analysis

Once we have constructed the graph **G** as shown in Section III-A, we want to partition the graph into clusters and

compute the key frame for every cluster. We use the technique introduced by Newman [15] in network analysis, that tries to maximize the *modularity* instead of minimizing the cut size [16]. One key advantage of this method is that this is a recursive method and can automatically determine the number of clusters in a graph.

We first take a look at bi-partitioning the graph. Modularity function [15], M can be defined as

$$\begin{aligned} M &= (\text{number of edges within a cluster}) \\ &\quad - (\text{expected number of such edges}). \end{aligned} \tag{4}$$

For all pair of vertices i and j, the number of edges can be determined by its weighted adjacency matrix A_{ij}, where the weights are determined by the similarity measure as mentioned in Equation 3. The expected number of edges can be derived from a null model as $\frac{k_i k_j}{2m}$, where k_i and k_j are the total weights of nodes i and j. $k_i = \sum_j A_{ij}$ and therefore, $2m = \sum_i k_i$ is the total weight of all edges in the graph. Thus, the modularity function M can be represented as $M = \frac{1}{2m} \sum_{(i,j) \in \text{same cluster}} B_{ij}$, where $B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$. Once the modularity function is obtained, a matrix based approach is used and M can thus be represented as

$$M = \frac{1}{4m} \mathbf{s}^T \mathbf{B} \mathbf{s}, \tag{5}$$

where $\mathbf{s}$ is the index vector such that $s_i = +1$ if vertex i belongs to the first group and $s_i = -1$ if vertex i belongs to the second group. $\mathbf{B}$ is the modularity matrix with elements B_{ij} that is used to maximize the modularity M in order to bi-partition the graph (if possible).

The maximization problem of M is NP-hard [29], and so an approximation is done using eigen vector analysis [15]. The method is shown in Algorithm 2. Existing spectral clustering techniques must consider the trivial case of all vertices being placed in one group. This is not present in our case because the modularity is being maximized.

Algorithm 2 Graph bi-partitioning using modularity

Input: A graph with n vertices, $\mathbf{B}$

1: Compute the eigenvectors $\mathbf{u}_i$ of modularity matrix $\mathbf{B}$. Subscript i denotes the i^{th} element of vector
2: Index vector $\mathbf{s} = \sum_{i=1}^{n} \alpha_i \mathbf{u}_i$ such that $\alpha_i = \mathbf{u}_i^T \mathbf{s}$
3: Compute the leading eigenvalue β^{max} and corresponding eigenvector $\mathbf{u}^{max}$ of $\mathbf{B}$
4: **if** $\beta^{max} \geq 0$ **then**
5: Compute $\mathbf{s}$ such that elements $s_i = +1$ if $(\mathbf{u}^{max})_i > 0$ and $s_i = -1$ otherwise
6: **if** $\mathbf{s}^T \mathbf{B} \mathbf{s} > 0$ **then**
7: **return** Division of graph into 2 clusters according to the signs of s_i
8: **else**
9: **return** Graph cannot be partitioned
10: **end if**
11: **else**
12: **return** Graph cannot be partitioned
13: **end if**

It is quite likely that a window of frames may contain more than 2 shots. So, we extend this method to find more than 2 clusters by using a recursive approach where each cluster is sub-divided. However, unlike in normalized cut [7] where each subgraph is treated independently, it is incorrect to delete edges that fall between two sub-graphs and apply the algorithm to each subgraph. This is because the modularity measure changes if edges are deleted and the wrong modularity measure is maximized. We therefore consider the additional contribution to modularity ΔM upon dividing a cluster G with n_G nodes in two, defined as

$$\Delta M = \frac{1}{4m} \mathbf{s}^T \mathbf{B}^G \mathbf{s}, \tag{6}$$

where $\mathbf{B}^G$ is the $n_G X n_G$ modularity matrix with elements indexed by labels of vertices i and j with cluster G and having values $B_{ij}^G = B_{ij} - \delta_{ij} \sum_{l \in G} B_{il}$ where δ_{ij} is the Kronecker δ function. This equation has the same form as Equation 5 and we can apply the method described above to maximize ΔM. Initially, we consider all vertices to be in one graph and therefore $\sum_l B_{il} = 0$. Consequently, while repeatedly subdividing a graph, we apply Equation 6 on each cluster. We stop when the partition does not increase the sub-modularity measure ($\Delta M \leq 0$). This happens if the generalized modularity matrix $\mathbf{B}^G$ has all eigenvalues ≤ 0 and therefore checking for $\beta^{max} \leq 0$ can give an indication about the termination of the partitioning process for that cluster. Thus, using the modularity measure we can automatically partition a graph into multiple clusters.

C. Key frame Detection

In order to find the vertex that is most representative of all the vertices in a cluster, we consider the magnitude of the elements of the leading eigenvector, $\mathbf{u}^{max}$, of modularity matrix $\mathbf{B}$ and choose the one with the largest value. This is equivalent to estimating the key frame in a cluster (video subsequence) and can be extracted as

$$K = \arg\max_i ((\mathbf{u}^{max})_i), \tag{7}$$

where K is the key frame and i are the vertices belonging to a particular cluster. This equation means the vertices corresponding to elements having the largest magnitude in the leading eigenvector make the most contribution in that cluster. For more details, please refer to [15].

D. Information Transfer

Since we are dealing with online video streams, we have access to only a window of n frames at a time (and not the entire video sequence). We first cluster these n frames and find the key frame for each cluster. Next, we assign the same cluster index label to frames that belong to the same cluster. In order to maintain temporal consistency, we transfer this information to the next window by using the last p frames from the current window and merge them with the next $n - p$ frames from the video. We transfer the cluster index and the key frame information about the overlapping P frames. We do not directly transfer information about the first $n - p$

frames. In the event that cluster information of those frames is not reflected in the P frames, temporal consistency is still not affected. We consider every new detected cluster in the subsequent windows to be different from the cluster from an earlier window. However, if the first $n-p$ frames of a window and the overlapping P frames belong to the same cluster, then there are no issues.

We transfer the key frame information to subsequent windows using P. In order to maintain consistency in enhancement, we do not compute the key frame again for subsequent windows that belong to the same cluster as frames from an earlier window. The choice of n and p may affect the estimation of the key frame, resulting in a sub-optimal estimate of the key frame. However, as shown later in Section IV-B2, key frame estimation is not critical for video enhancement.

The modularity algorithm that we have described in Section III-B has a failure case - It cannot detect clusters with just one vertex. It needs at least 2 vertices in a cluster. This is because the notion of edges within a cluster is not present when there is only one vertex. If a different frame is present in the middle of a sequence, then our algorithm will fail to detect it. This is an unlikely scenario in videos. However, it is quite possible that when we consider a window of n frames, there could be $n-1$ frames belonging to one cluster and the last frame belonging to a different cluster. Since we use a sliding window mechanism, it is highly likely that the last frame from the previous window will be grouped in a cluster with frames from the next window. Therefore, to prevent such cluster index label switching which may result in incorrect enhancement, we enhance only the first $n-p$ frames of current window and postpone processing of the last p frames to the next window.

E. Video Enhancement

Given a new cluster label, we estimate the enhancement parameters of only the key frame of that cluster using [2]. We use an enhancement method that is based on the Retinex theory. We first try to estimate the illumination component and separate it from the reflectance component of the image. While doing so, we try to achieve color constancy and enhance only the luminance of the illumination component of the image. This is done in order to preserve the color properties of the image. The enhancement parameters are computed depending on the exposure conditions of the image (underexposed, overexposed or a combination of both). Then, logarithmic curves inspired by the Weber-Fechner law are applied to enhance the luminance. This enhanced illumination is multiplied with the reflectance to obtain the enhanced frames. Next we enhance all frames in that cluster with the estimated enhancement parameters of the key frame. To enhance frames that have the same cluster index label as already seen in an earlier window, we use the enhancement parameters from the previously estimated key frame of that cluster. This maintains temporal consistency during enhancement.

IV. Results and Discussion

In this section, we evaluate the results of our online clustering process on the TRECVid 2001 dataset [17] and compare with existing methods. We considered the TRECVid 2001 dataset because the ground truth information regarding scene change is available. For video enhancement, we enhance trailers of High Definition (HD) movies from the Apple website[1]. We compare our key frame extraction with an existing method. We show that video enhancement using our technique reduces flash and flickering artifacts. We also show the significance of proper clustering for video enhancement. We perform experiments to conduct human validation to know the effectiveness of the enhancement method. Finally, we show how scene change detection helps in robust video segmentation.

A. Quantitative Evaluation of Cluster Detection

In order to evaluate our clustering process, we carry out experiments on video sequences from the TRECVid 2001 dataset [17] and compare our results with 6 existing methods [30], [3], [31], [32], [33], [7]. We consider 8 video sequences whose information is summarized in Table I. We chose these 8 videos because of fair comparison with existing methods that use a similar subset. We directly present the results from [30], [31], [32], [33]. We implemented the methods described in [3] and [7] and use the same similarity measure as our method. Since normalized cuts [7] requires prior information about the number of clusters and since the number of clusters is not fixed, we use eigengap [12] to automatically detect the number of clusters. Comparing with [7] is thus equivalent to comparing with [9], [13], [10].

The ground truth information about the different transitions is provided. The transitions are divided into 2 broad categories - cuts and gradual transitions. The gradual transitions are further sub-divided into other categories - dissolve, fade in/out and others. As ground truth, the exact frame number is provided for cuts whereas for the other gradual transitions, a range of frames are provided. For some of the videos (nad31, nad33, nad53 and nad57), the reported ground truth values are at a fixed offset from the actual frame number of sequences. We account for these discrepancies while evaluating different methods. To compare and evaluate our method we use 2 performance measures - precision and recall. Recall can be defined as

$$R = \frac{N_c}{N_c + N_m}, \tag{8}$$

where R is the recall, N_c is the number of correct detections and N_m is the number of missed detections. Precision can be defined as

$$P = \frac{N_c}{N_c + N_f}, \tag{9}$$

where P is the precision and N_f is the number of false detections.

Unlike existing methods such as [30], [31], [32], [33], our clustering method does not explicitly detect gradual transitions. For gradual transitions, we consider correct detection as the case when clustering detects a scene change in the range of the ground truth gradual transition frames. However, if more

[1]http://trailers.apple.com/

TABLE I
VIDEOS FROM TRECVID 2001 DATASET USED IN OUR EXPERIMENTS

Title	**File**	**Size** (MB)	**Run time** (mm:ss)	**Frame Count**	**Transition Count**	**Cuts**	**Gradual Transition**
NASA 25th Anniversary Show-Seg 5	anni005	66.9	6:19	11364	65	38	27
NASA 25th Anniversary Show-Seg 9	anni009	72.4	6:50	12307	103	38	65
Spaceworks - Episode 6	nad31	260.1	29:08	52405	242	187	55
Spaceworks - Episode 8	nad33	247.1	27:40	49768	215	189	26
A&S Reports Tape 4-Report 260	nad53	128.0	14:20	25783	157	82	75
A&S Reports Tape 5-Report 264	nad57	63.4	7:06	12781	67	44	23
Challenge at Glen Canyon	bor03	240.5	26:56	48451	242	231	11
The Great Web of Water	bor08	251.0	28:07	50569	531	380	151

TABLE II
RESULTS FOR GRADUAL TRANSITIONS ON VIDEOS FROM TRECVID 2001 DATASET. R IS FOR RECALL AND P IS FOR PRECISION. A '-' INDICATES UNREPORTED RESULT

File	**Ours**		[30]		[31]		[32]		[33]		[3]	
	R	P	R	P	R	P	R	P	R	P	R	P
anni005	**0.888**	0.828	0.666	0.782	0.759	1.000	0.786	**0.880**	-	-	0.000	0.000
anni009	**0.907**	**0.881**	0.507	0.733	-	-	0.848	0.926	0.523	0.669	0.046	0.750
nad31	**0.818**	**0.789**	0.535	0.428	-	-	0.708	0.687	0.418	0.436	0.000	0.000
nad33	0.923	0.800	0.692	0.382	-	-	**0.943**	0.805	0.500	0.389	0.231	**0.857**
nad53	**0.970**	0.913	0.805	0.696	-	-	0.826	**0.947**	0.631	0.575	0.040	0.750
nad57	**0.956**	0.846	0.826	0.593	-	-	0.852	0.885	-	-	0.087	**1.000**
bor03	**1.000**	0.688	0.818	0.281	-	-	0.929	0.382	0.545	0.283	0.182	**1.000**
bor08	**0.960**	0.913	0.758	0.816	-	-	0.716	**0.930**	0.741	0.794	0.007	0.500

than 1 scene change is detected in the range of the ground truth of gradual transition frames, then we consider those to be false detections. The number of clusters within a dissolve transition varies according to the rate at which the transition occurs. The precision and recall rates are shown in Table II.

Due to the reasons mentioned above, the comparison of precision and recall rates for gradual transitions as shown in Table II does not convey significant information about the comparison of methods. Since our method and [3] use the same technique, we can conclude that our method is better. It is not clear how the existing methods will behave when their respective criterion for cuts is applied to gradual transition frames since neither the source code nor any confusion matrix is provided. As the recall rates are quite high when using our clustering method, we can infer that our method is able to detect scene changes across gradual transitions. An instance of the failure case occurs when the dissolve transition gradually happens from the previous scene to a zoomed-in part of the same scene. In these cases, the illumination remains almost same. Also, a lot of edge information between the 2 scenes remains the same. This results in a high similarity measure and therefore, our clustering method fails to detect such gradual scene changes. Note that such failure cases that occur in shot boundary detection are actually desirable from the video enhancement perspective for robust enhancement since the frames effectively belong to the same scene and have similar contrast. Notice that our precision rates are on the lower side This is because our method generally detects a scene change both at the start and at the end of a gradual scene transition. [3] has a higher precision rate because along with not being able to detect gradual transitions (low recall rate), it also does not detect many false scene changes amidst gradual transitions.

We use our clustering method to also detect cuts. Since we assume the clustering method to detect cuts and the ground truth values are provided, we can use them to compare different methods using both precision and recall values as shown in Table III. The false detections in the range of the ground truth of gradual transitions (depicted by precision in Table II) and the false detections detected otherwise (depicted by precision in Table III) comprise all the false detections of scene changes detected by our method for the given video sequence.

As can be seen in Table II and Table III, our clustering method has a high recall at the cost of precision. It is critical that our clustering method has a high recall rate (i.e.) all the clusters/scene changes in a sequence should be correctly detected. This is because during enhancement, a missed scene change due to either a cut or a gradual transition may cause the enhancement parameters of a previous scene to be applied to a different scene. These effects are undesirable, and an example illustrating such an undesirable effect in terms of video enhancement can be found in Section IV-B3. One of the failure cases for cuts in the TRECVid dataset was when the movie had a cut transition between 2 scenes - both scenes show different faces under similar illumination conditions. In this case both the color and edge histograms were quite similar.

Since a low precision rate implies a high false detection rate of clusters, the precision rate should also not be very low. This will result in frequent computation of enhancement parameters for different clusters that belong to the same scene causing temporal inconsistency. In the boundary condition, a very low precision rate will cause the enhancement process to perform frame-by-frame enhancement, resulting in flashing artifacts. An illustration of such an effect in video enhancement is described later in Section IV-B2. The low precision rate using normalized cut [7] is primarily because of false cluster detections using eigengap [12]. This is because for some sub-sequences of videos, eigengap considers every frame to

TABLE III
RESULTS FOR CUTS ON VIDEOS FROM TRECVID 2001 DATASET. R IS FOR RECALL AND P IS FOR PRECISION. A '-' INDICATES UNREPORTED RESULT

File	**Ours**		[30]		[31]		[32]		[33]		[3]		[7]	
	R	P	R	P	R	P	R	P	R	P	R	P	R	P
anni005	**1.000**	0.826	0.973	0.948	0.946	**1.000**	**1.000**	0.736	-	-	0.868	0.943	**1.000**	0.160
anni009	**1.000**	0.760	0.842	**0.888**	-	-	**1.000**	0.655	0.789	0.697	0.947	0.857	**1.000**	0.086
nad31	0.963	0.818	0.912	0.938	-	-	0.901	0.945	0.866	0.900	0.043	**1.000**	**0.973**	0.176
nad33	0.947	0.861	**0.989**	0.963	-	-	0.984	**0.964**	0.952	0.918	0.910	0.950	0.973	0.196
nad53	**0.988**	0.802	0.975	**0.975**	-	-	0.952	0.941	0.951	0.797	0.842	0.920	**0.988**	0.129
nad57	**0.977**	0.915	0.772	**0.971**	-	-	0.958	0.939	-	-	0.909	0.909	**0.977**	0.167
bor03	0.978	0.904	0.938	**0.954**	-	-	0.961	0.949	0.982	0.953	0.627	0.923	**0.996**	0.321
bor08	0.934	0.868	0.965	0.882	-	-	**0.973**	0.901	0.873	0.945	0.497	**0.990**	0.800	0.372

belong to a different cluster which increases the count of false detections by a large value. If the cluster count for each sliding window is calculated using our method and the count is given as an input to [7], then the results using [7] are similar to our results. Using recursive normalized cut will not work well because cut-size threshold will vary across diverse videos. Having a fixed constant cluster count is also not a good idea. Existing techniques [9], [13], [14] based on eigengap will show similar poor performance. As shown in Table II, [3] does not work well when the scene is gradually changing although the performance is reasonable for cut transition as shown in Table III.

Since we use 8 videos and 7 methods for comparison, each having 2 measures (precision and recall), the visualization of all these measures on a graph may get incomprehensible. Therefore, to compare the performance of different methods with respect to cuts transition, we calculate F_1 measure defined as

$$F_1 = 2\frac{P.R}{P+R}, \tag{10}$$

where P is the precision and R is the recall. The F_1 measure depends on both precision and recall. We calculate the F_1 measure of each method on every video, and the comparison of F_1 measures is shown in Figure 1.

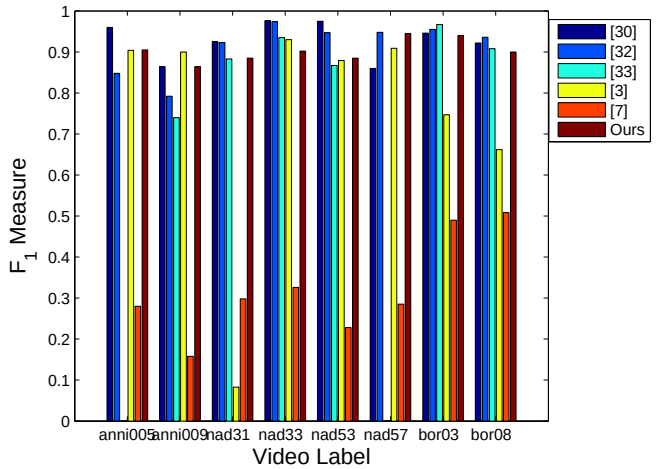


Fig. 1. Comparison of methods on the basis of F_1 measure for cuts transition

As can be seen in Figure 1 and in Table IV, our clustering method based on modularity maximization performs comparably to state-of-the-art approaches despite using very simple

TABLE IV
AVERAGE F_1 MEASURE FOR VIDEOS SHOWN IN FIGURE 1

Method	Average F_1 measure
[30]	**0.928**
[32]	0.915
[33]	0.883
[3]	0.752
[7]	0.322
Ours	0.903

similarity measures. We performed statistical analysis on the different methods shown in Table IV using the Wilcoxon-signed rank test and found that the F_1 measure of the best method [30] is not statistically significantly different from our method. Our method is not specifically designed for shot detection, and the failure to detect some shot boundaries (Figure IV-A(a) and (b)) or extra detection of cluster boundaries (Figure IV-A(c) and (d)) is counter-intuitive and is in fact conducive to video enhancement. Illustrations of such cases can be found in Figure IV-A. Some of the cases when our method detects a false cluster boundary is when there is a change in the illumination of the scene due to flash/explosion/light of the scene turning on/off, when the appearance of an object changes in the scene (due to camera motion or object motion), when there is a zoom-in or zoom-out of parts of the scene, or when text(movie credits) gradually appears and covers substantial part of the scene. Although in such cases the ground truth considers the events to belong to the same scene, the contrast of the scene changes, and therefore it is reasonable to assume a scene change for the purpose of video enhancement. Consequently, the enhancement parameters should also change with change in contrast of the scene. Similarly for events as shown in Figure IV-A(a) and (b), the ground truth considers them to be from different scenes. However, our method considers them to be from the same scene, and this matches with the intuition of video enhancement because we would want those scenes to be enhanced using the same parameters. This is why our method uses simple features such as color and edge histograms to compute similarity measures as these features fit well in our contrast enhancement framework. Existing methods [3], [5] use wavelet analysis, luminance values, texture analysis, shape features, flash detection, motion estimation, adaptive thresholds etc. to compute similarity measures between frames. Using machine learning techniques, we can find the best combination of features to make the similarity measure more robust and con-

Fig. 2. Failure cases using our clustering method with true cluster boundary shown in red and false positive in magenta. It could not detect cut transition in (a) where the illumination and edge information is similar. It also could not detect dissolve transition in (b) where it shows the same surface of moon after zooming in. It detects extra cluster boundaries in (c) and (d) as there is a significant difference in illumination. Such failure cases in terms of shot detection are required for video enhancement

sequently improve the performance of the clustering method for the purpose of shot boundary detection at the potential cost of stable video enhancement.

B. Video Enhancement

In this section, we show results of clustering on some HD movie trailers. We show how clustering results in temporal consistency by removing flashing artifacts thus improving the process of video enhancement. We also show why proper estimation of clusters is important for the video enhancement process.

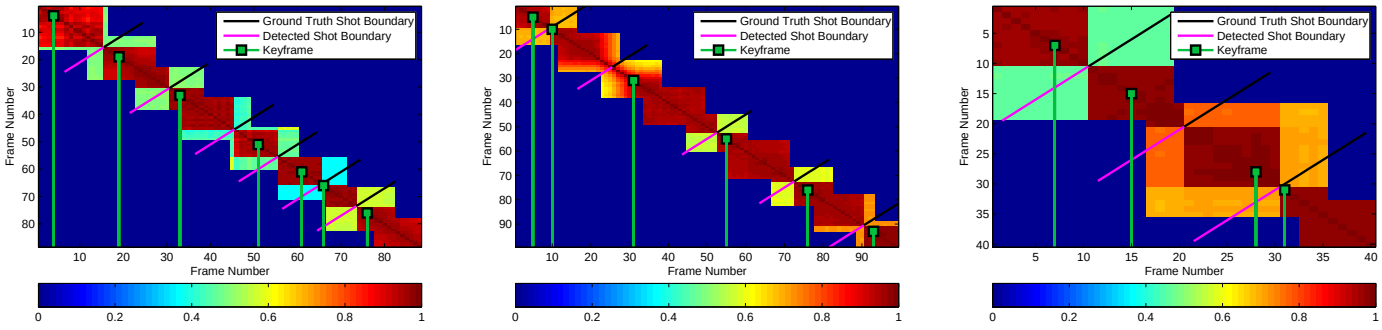
The most obvious way to measure the effectiveness of contrast enhancement is to ask an observer to indicate their preference in a side-by-side comparison of the original and the enhanced videos. The preferences can then be quantified to know the effectiveness of the method. Choudhury and Medioni have shown the effectiveness of the proposed image contrast enhancement technique in [2]. However, Weber *et al.* [34] have demonstrated that static enhanced images are perceived as less enhanced or not enhanced after a short period of adaptation. This adaptation to enhancement may diminish the benefits of image enhancement (due to perceiving the enhanced images as normal) and thus affect the subjective evaluation of enhanced videos. We therefore conduct experiments, present the results, and discuss them.

1) Cluster Analysis and Key frame Extraction: We consider 3 movie trailers and plot the frame similarity graph for those trailers in Figure 3. We consider a sliding window mechanism with window size of $n = 16$ frames and overlapping set of $p = 5$ for all results in the paper except Figure 3(c). For Figure 3(c), we use $n = 19$ and $p = 3$ to show robustness of our method to n and p and to illustrate that the modularity algorithm automatically detects more than 2 clusters (3 clusters) as can be seen in the 2^{nd} window. The frame similarity graph indicates the similarity between frames (Red is the most similar and blue is the least). Since we transfer information across windows using an overlapping set, P, we are able to correctly detect clusters (shots) that are spread across consecutive windows in each of the videos shown in Figure 3. The ground truth values of shot boundaries are obtained by manual inspection of the trailer sequences. There are 2 distinct groups that we can visualize. One is the sliding window of frames, W with overlapping frames, P, and the other is the cluster that may/may not be spread across different windows. We have also shown the results of key frame extraction for every unique cluster. One important aspect to note is that this key frame is locally the most representative key frame of a cluster. It does not need to be globally the most similar frame to other frames in that cluster (that may be spread across different windows).

A potential concern regarding the clustering mechanism is its behavior in a slowly changing scene. One such example is shown in Figure 3(b). Consider the 2^{nd} and 3^{rd} window. That is an example of a *dissolve* transition where one shot gradually blends into another shot. The images from this transition period are shown in Figure IV-B1. Even in such instances, the modularity clustering algorithm is able to detect shot boundaries.

Fig. 4. Slowly changing scene where we estimated the shot boundary (black rectangle). Images in sequence are from left to right and top to bottom

Notice that for different values of n, the results of key frame detection may change. However, the shot boundary detection will be consistent for a *cut* transition as shown in



(a) Video 1 (b) Video 2 (c) Video 3

Fig. 3. Frame similarity graph along with cluster and key frame information for 3 movie trailers. The ground truth shot boundaries are shown with black lines on the upper right of the diagonal and the detected shot boundaries (indicated by change of cluster label index) are shown with pink lines on the lower left of the diagonal. The key frame information that is being used for enhancement for every cluster is shown with a green square marker

Figure 3(c). We discuss the effects of choosing key frame in Section IV-B2. In order to check for correctness of the key frame, we consider the video whose similarity graph is shown in Figure 3(a). We consider the 1^{st}, 4^{th} and 5^{th} clusters and the respective key frames for that cluster. We compute how similar each frame is to the other frames in that cluster, and then find the mean of that similarity measure. As can be seen in Figure 5, for all 3 clusters, the key frame is locally the most similar to other frames in that cluster. We also plotted the similarity measure of the frame that is the mid-point in the sequence of that cluster. This frame is considered to be the key frame by an online clustering process [3]. However as seen in Figure 5, the detected key frame by [3] does not need to be the most representative frame of the cluster. Apart from video summarization [35], key frame detection can also find application in video enhancement as shown in Section IV-B2.

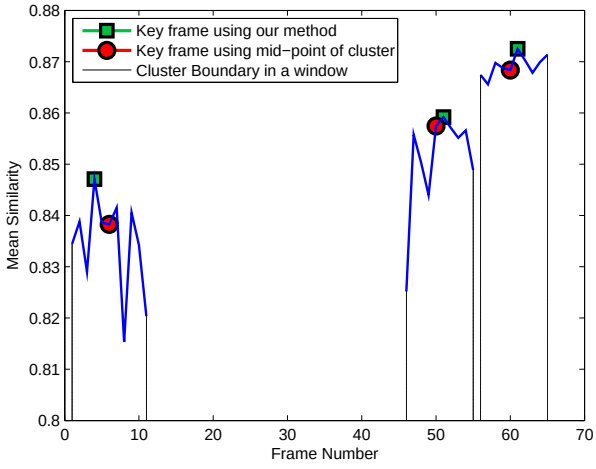


Fig. 5. Similarity measure of key frame

2) Removal of flashing artifacts: In order to demonstrate video enhancement, we consider a part of the video whose similarity graph is shown in Figure 3(b). We consider the 3^{rd} cluster. We enhance frames from that video sub-sequence using 2 techniques - 1) On a frame-by-frame basis 2) Using our online method (Enhance all frames using parameters of local key frame). For that video sub-sequence, we track an object and calculate the mean intensity of the tracked object through the length of the sequence.

For the tracked object whose plot can be seen in Figure 6, the original intensity (shown in blue) is fairly consistent throughout the sequence. Enhancement on a frame-by-frame basis (shown in green) results in intensity fluctuations due to differences in the estimation of parameters. This results in undesirable flash and flickering artifacts that can be seen as unnatural peaks (at frame numbers 2, 13 and 21). Similar effects can also be observed when the clustering method has a very low precision rate. This will cause multiple estimation of enhancement parameters for the same scene resulting in flashing artifacts. In the degenerate case this is analogous to frame-by-frame enhancement.

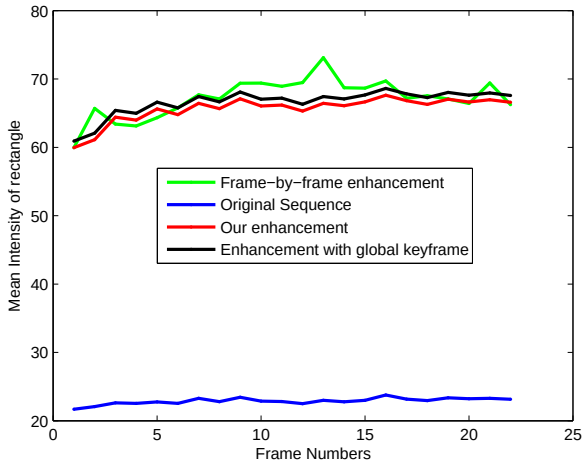


Fig. 6. Comparison of enhancement techniques for the tracked object. (Temporal consistency and significance of key frame)

We can see that using our online method (enhance all frames

using parameters of just the key frame) reduces fluctuations (shown in red). The standard deviation of the mean intensity of the tracked object across the length of the sequence, σ is 0.46 for the original sequence and is 2.81 for the frame-by-frame enhancement approach. The σ using our approach (1.88) is less than the σ of frame-by-frame enhancement approach. This gives us an indication of reduction of artifacts by our approach.

Since we want an online system, we use parameters of the local key frame to enhance the frames belonging to that cluster instead of delaying the enhancement process to account for the global key frame. We also enhanced this sub-sequence using parameters of the global key frame of the cluster. The global key frame was obtained offline by considering the entire shot and then estimating the key frame. The estimation of global and local key frames was different. As shown in Figure 6, using parameters of the global key frame also results in consistent enhancement (shown in black). Since the global or local key frame does not result in a significant difference in estimation, we can say that our enhancement process is not dependent on the estimation of the key frame. Using any other frame from the same cluster instead of the key frame also results in stable results. However given a cluster, using the key frame quantitatively gives us lower standard deviation than using other frames.

We also computed the average of the estimated parameters from all frames in a cluster and used that to enhance that cluster. However, there was not a significant difference in the visual quality of the enhanced video. Moreover, parameter estimation is a costly process. Since we aim to have a real-time system, we want low computational cost. The current piece-wise linear curve that is shown in Figure 6 is sufficiently smooth for the human eye as there are no visible temporal artifacts, thus making our enhancement perceptually temporally consistent. Achieving a smoother curve at the cost of speed without a visible change in human perception is undesirable in a real-time system. Thus key frame estimation makes our method robust and fast and helps in achieving stable video enhancement.

3) Significance of Clustering: Here we discuss how proper cluster estimation is critical to the enhancement process. We consider a sub-sequence consisting of 2 clusters. The first cluster is an overexposed set of frames, and the second one is an underexposed set with a *dissolve* transition from frames 9 to 12 as can be seen in Figure 7. Our method successfully detects 2 clusters, estimates the key frame for each cluster, and then enhances the sequence accordingly. As a result, our framework correctly enhances each cluster, resulting in consistent enhancement. It correctly reduces the intensity of overexposed frames and increases the intensity of underexposed frames. While doing so, an interesting phenomenon is observed. There is a sudden change in the mean intensity of the frame from frame 9 to frame 10. This happens for the intermediate transition frames. Since the transition occurs from an overexposed sub-sequence to an underexposed sub-sequence and the enhancement parameter that is calculated from the key frame of the underexposed sub-sequence is applied to the transition frames that contain content from both the over-exposed and the under-exposed parts, there is a jump in the mean intensity of the frame. This jump in the intensity is not noticeable during the normal viewing of the video and we verified this by playing the video at different frame rates (3 fps to 24 fps) and getting responses from 5 different human observers. Moreover, during transition frames, there is no coherent information and such effects go unnoticed by human users.

However, applying [7] with eigengap [12] results in undesirable enhancement. Eigengap only estimates 1 cluster and therefore, normalized cut [7] incorrectly enhances the underexposed cluster using the key frame from the overexposed frames. This results in a decrease in intensity for the already underexposed frames. These effects can be observed in a clustering method having a low recall rate. Thus proper clustering is extremely critical for appropriate video enhancement. We compared these two enhancements (ours and using [7]) by conducting preference tests on the same 5 human observers as above and all of them preferred our enhancement.

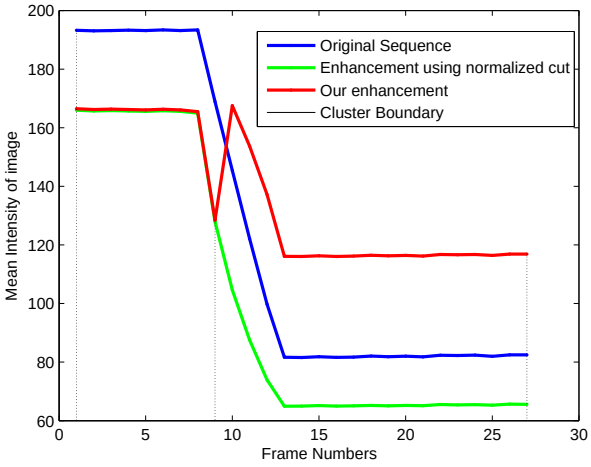


Fig. 7. Significance of clustering. Failure to detect clusters using existing method results in improper enhancement as shown by the green curve that reduces contrast of underexposed frames(Frames 13-27). This problem is not present using our method as shown by red curve

4) Human Evaluation: To establish the effectiveness of the enhancement method, we enhanced 24 High Definition(HD) videos using our framework. On an average, each video is approximately 30 seconds long. All the videos have been enhanced offline. We have uploaded the YouTube version of some of the original and enhanced videos at "http://iris.usc.edu/people/achoudhu/video.html". As mentioned earlier, all these videos are movie trailers from the Apple website. Each video has multiple transitions. The number of transitions and the combination of transitions are different for every video. Specifically, 16 videos have at least 1 dissolve transition and 9 videos have at least 1 other gradual transition. Other gradual transitions may include fade-in/fade-out/wipe transitions.

Since the videos are in the '.mov' format, we present the original and the enhanced version simultaneously using Quicktime Pro player. The placement of the original and the

enhanced version were random(either the top or the bottom of the screen) to remove any bias that may exist while selecting a video. We asked each of the 16 human observers independently to rate the video on the "top" as "**Better**", "**Same**" or "**Worse**" relative to the video on the "bottom" of the screen and recorded their responses. Scores were assigned as 1 for **Better**, 0 for **Same** and -1 for **Worse**. These responses were used to evaluate the **preference** for enhanced video. Since each video has very diverse content, the observers were asked to view the entire video trailer and were also given the liberty to view the video multiple times (if required) before rating the video. This was repeated for all 24 videos in the database.

The experiments were conducted independently on the subjects. Each subject was seated at a distance of roughly 2 feet from the computer monitor. The subjects were given the freedom to move closer to the screen or farther away from the screen according to their convenience. The screen is around 24"(diagonal) and the subjects were seated perpendicular to the center of the screen.

Figure 8 shows the **preference** ratings for the enhanced videos when compared with the original videos. The histogram in Figure 9 gives a better visualization and we can see that the observers show a strong **preference** for the enhanced videos compared with the original videos (**preference** > 0). We use the Wilcoxon signed-rank test [36] to check for statistical significance and infer that the **preference** for enhanced videos is statistically **significantly different** from that for the original videos (The null hypothesis of zero median of the difference is rejected at 5% level). By comparing the ranks we conclude that the enhanced videos have a higher rank than the original videos. This implies that observers have a **significant preference** for the enhanced videos.

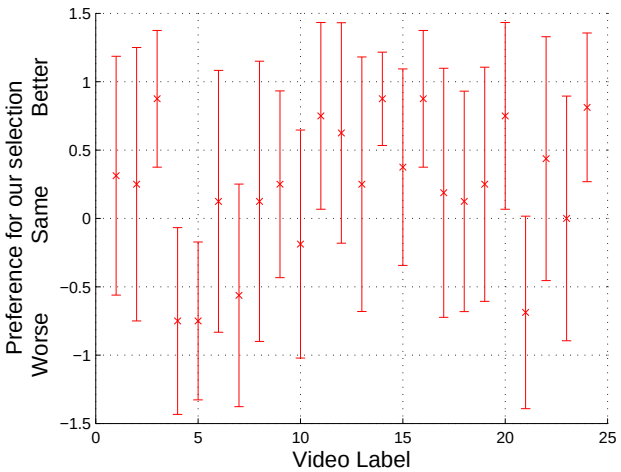


Fig. 8. **Preference** ratings for enhanced videos

The evaluation of the **preference** for the enhanced videos can be analyzed using a variation of the ROC approach as described by Peli *et al.* [37] to give an impression of perceived quality of the enhanced videos with respect to the original videos. Unlike in regular ROC analysis, where ground-truth information is provided, no such information is present in this

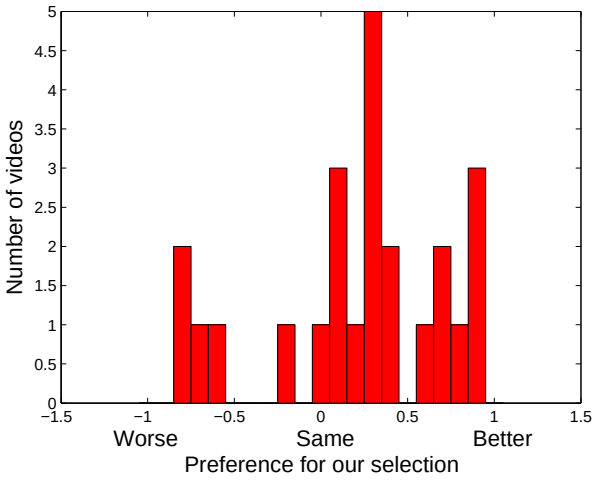


Fig. 9. Histogram of mean **preference** ratings for all observers

case. The raw data consists of the **preference** of observers regarding perceived quality of the enhanced videos with respect to the original videos. In this case, the axes of the ROC plots have a different interpretation. The Y-axis, that originally corresponded to the true positive rate, can be considered to be equivalent to the proportion of the enhanced video with higher perceived video quality. The X-axis, that originally corresponded to the false positive rate, can be considered to be equivalent to the proportion of original video with higher perceived video quality.

We consider the area under the ROC curve [38] (A_z) as a measure of the perceived quality of the enhancements. For the original videos, the value of $A_z = 0.5$. If $A_z > 0.5$ then the perceived quality of the enhanced video can be considered to be better than that of the original videos. On the other hand, if the perceived quality of the original videos is better than that of the enhanced videos, then $A_z < 0.5$. The empirical values of A_z for different observers are shown in Figure 10. For all the observers, since average $A_z = 0.6108 \pm 0.0968$, the enhanced videos are deemed to have better quality than the original videos. Using the Wilcoxon signed-rank test [36], our enhanced videos was reported as having **significantly better** quality than the original videos.

Although **preference** for our enhancements is statistically significant, in some cases (6 videos) as shown in Figure 8 the observers prefer the original video (**preference** ≤ 0). Apart from personal preferences, this is due to the characteristics of the video. Some of these videos have good quality and the observers prefer less or no enhancement of such videos. Also, since we enhance High Definition(HD) trailers of popular movies, in some cases the observers have an idea about the content of the original movie and use that information to make a judgment regarding their **preference**. The observers also did not report flashing artifacts in our enhanced videos.

5) Computational Cost: Estimating the illumination component of a video frame that is a part of the image enhancement process is computationally expensive. We therefore use a GPU (Graphics Processing Unit) and use an implementation by Kharlamov and Podlozhnyuk [39] on a PC with Intel

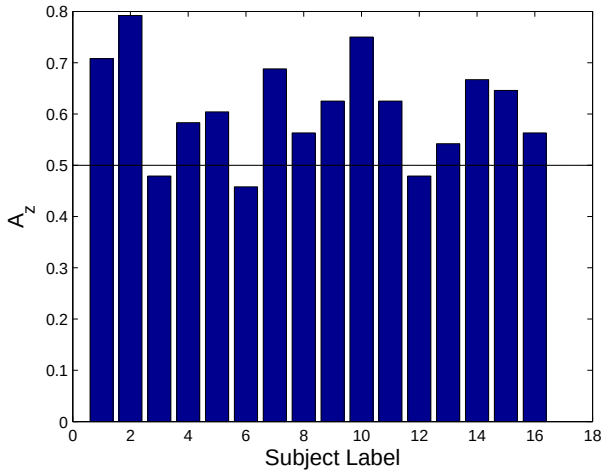


Fig. 10. A_z's for all observers

i5-680 processor with NVidia GeForce GTX 480 GPU. We build our framework in Visual C++ environment. We use GPU only for illumination estimation. For a 1920 X 1080 image, illumination component estimation runs at around 10 fps. At the cost of potential minor artifacts, the speed can be further increased by using approximations where only one distance per patch is computed. Due to this bottleneck, for a 1080p video (each frame has 1920 X 1080 resolution), our enhancement framework runs at approximately 3.5 fps and for a 720p video our enhancement framework runs at approximately 6.4fps. This time includes the computation time required for first decoding the video to get frames and then encoding the frames again to get back the original video for which we use the FFMPEG library.

C. Video Segmentation

Clustering a video sequence into shots finds interesting applications in video segmentation. We consider the video segmentation method proposed by Grundmann *et al.* [40]. This method considers a video volume and tracks segments through a spatio-temporal volume. It uses a hierarchical graph based technique where increasing levels have decreasing number of regions that are obtained by merging regions from previous levels. The authors have shown consistent results over long video sequences that also include movies. In order to maintain temporal consistency, the authors partition video into $n = 25$ frames and use $p = 8$ overlap frames from previous sub-sequence to the current sub-sequence[40]. The performance of the method is not critically dependent on the values of n and p.

Movies are a combination of a diverse range of frame sequences. Using a fixed number of frames to partition the video decreases the robustness of the existing method because of information mismatch across different shots of the video. We propose to analyze the incoming video stream by dividing the sequence into clusters of similar frames and using this information to segment the video. While choosing n and p, the method should be modified such that frames from the same cluster are chosen. This will ensure that segmentation labels remain consistent through the length of the shot. A change of segmentation label after the shot may be acceptable. The results of using cluster information are shown in Figure IV-C. As shown in the bottom row left image of Figure IV-C, we see that our results are more robust to illumination changes. At no level of the hierarchy is the original method (middle row) able to segment the face of the image. As shown in the right image of Figure IV-C, the existing method is not able to retrieve a structure (merged with background water) that was segmented by our method. For lower levels of the hierarchy, it gave a different label to that structure than other similar structures, which is not desirable. The maximum hierarchical level is set when the maximum number of regions is below a threshold(10).

Fig. 11. Segmentation Results. Top row: Original images. Middle row: Segmentation results using [40]. Bottom row: Segmentation results [40] using cluster information. The left column shows results at 90% of the maximum hierarchical level and the right column shows results at 70% of the maximum hierarchical level

V. Conclusion and Future Work

We have proposed a novel approach to perform video enhancement. Our system is online, automatically clusters video sequences, and extracts a key frame from each cluster. In order to do clustering, we have used a new graph partitioning technique called *modularity* that has been used in network analysis. We quantitatively evaluated this clustering method on TRECVid 2001 dataset and compare with existing methods. In order to extract key frames,we perform eigen analysis on the modularity matrix. We enhance video sequences using the estimated parameters from the key frame. Using clustering information results in robustness and significant reduction of flash and flickering artifacts as compared to frame-by-frame enhancement. Correct cluster estimation is critical to the enhancement process. Extraction of the key frame increases efficiency of our method though it is not critical to the enhancement process. Enhancing a scene using any frame from that scene also gives stable results. We have shown that

our technique indeed results in enhanced videos and validated our argument by conducting experiments on human observers. We have also used our framework to produce stable video segmentation.

We would also like to see the performance of our clustering method using other similarity measures such as wavelet or motion estimation. Our technique improves the quality of the video for people with normal vision. We would also like to explore whether this technique helps patients with Age-related Macular Degeneration. We would also like to further improve the speed of our framework in order to make it a real-time system. Also, it would be interesting to see if clustering in video actually helps in other problems such as video denoising. It will also be interesting to see if video contrast enhancement further improves the performance of video segmentation.

ACKNOWLEDGMENT

This research was supported by the National Institutes of Health Grant EY016093. The authors would also like to thank Hao Wang for helping them with the implementation of the framework.

REFERENCES

[1] M. Fullerton and E. Peli, "Post transmission digital video enhancement for people with visual impairments," *Journal of the Society for Information Display*, vol. 14, no. 1, pp. 15–24, 2006.

[2] A. Choudhury and G. Medioni, "Perceptually motivated automatic color contrast enhancement based on color constancy estimation," *EURASIP Journal on Image and Video Processing*, 2010.

[3] C. Gianluigi and S. Raimondo, "An innovative algorithm for key frame extraction in video summarization," *Journal of Real-Time Image Processing*, vol. 1, pp. 69–88, 2006.

[4] R. Lienhart, "Comparison of Automatic Shot Boundary Detection Algorithms," in *SPIE SRIVD*, 1999, pp. 290–301.

[5] A. F. Smeaton, P. Over, and A. R. Doherty, "Video shot boundary detection: Seven years of trecvid activity," *Computer Vision and Image Understanding*, vol. 114, no. 4, pp. 411 – 418, 2010.

[6] H. Denman and A. Kokaram, "A multiscale approach to shot change detection," in *Irish Machine Vision and Image Processing*, 2004.

[7] J. Shi and J. Malik, "Normalized cuts and image segmentation," in *CVPR*, 1997, p. 731.

[8] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in NIPS*. MIT Press, 2001, pp. 849–856.

[9] V. Chasanis, A. Likas, and N. Galatsanos, "Video rushes summarization using spectral clustering and sequence alignment," in *TRECVid Video Summarization Workshop*, 2008, pp. 75–79.

[10] Y. Gao, J. Tang, and X. Xie, "Key frame vector and its application to shot retrieval," in *IMCE*, 2009, pp. 27–34.

[11] L. Zhong, C. Li, H. Li, and Z. Xiong, "Unsupervised clustering algorithm for video shots using spectral division," in *ISVC*, 2008, pp. 782–792.

[12] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory*. Academic Press, 1990.

[13] U. Damnjanovic, T. Piatrik, D. Djordjevic, and E. Izquierdo, "Video summarisation for surveillance and news domain," in *SAMT*, 2007, pp. 99–112.

[14] J.-M. Odobez, D. Gatica-Perez, and M. Guillemot, "Video shot clustering using spectral methods," in *CBMI*, 2003, pp. 94–102.

[15] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, no. 3, p. 036104, Sep 2006.

[16] R. Ghosh and K. Lerman, "Community detection using a measure of global influence," in *SNA Workshop, KDD*, Washington DC, USA, 2008.

[17] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and trecvid," in *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*. New York, NY, USA: ACM Press, 2006, pp. 321–330.

[18] W. Abd-Almageed, "Online simultaneous shot boundary detection and key frame extraction for sports videos using rank tracing," in *ICIP*, Oct 2008, pp. 3200 –3203.

[19] G. Hines, Z.-U. Rahman, D. Jobson, and G. Woodell, "Dsp implementation of the retinex image enhancement algorithm," in *Visual Information Processing XIII*, 2004, pp. 13–24.

[20] K. Goh, Y. Huang, and L. Hui, "Automatic video contrast enhancement," in *ICCE*, Reading, UK, sep. 2004, pp. 359–364.

[21] Q. Wang and R. Ward, "Fast image/video contrast enhancement based on weighted thresholded histogram equalization," *IEEE Trans. on Consumer Electronics*, vol. 53, no. 2, pp. 757 –764, May 2007.

[22] S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High dynamic range video," in *ACM SIGGRAPH 2003*, San Diego, USA, 2003, pp. 319–325.

[23] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 267–276, 2002.

[24] S. Ramsey, J. T. Johnson III, and C. Hansen, "Adaptive temporal tone mapping," in *CGIM*, Kauai, Hawaii, 2004.

[25] H. Wang, R. Raskar, and N. Ahuja, "High dynamic range video using split aperture camera," in *OMNIVIS Workshop, ICCV*, Beijing, China, 2005.

[26] E. P. Bennett and L. McMillan, "Video enhancement using per-pixel virtual exposures," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 845–852, 2005.

[27] S. N. Pattanaik, J. Tumblin, H. Yee, and D. P. Greenberg, "Time-dependent visual adaptation for fast realistic image display," in *ACM SIGGRAPH 2000*, New Orleans, USA, 2000, pp. 47–54.

[28] M. J. Swain and D. H. Ballard, "Color indexing," *IJCV*, vol. 7, no. 1, pp. 11–32, 1991.

[29] O. Goldschmidt and D. S. Hochbaum, "Polynomial algorithm for the k-cut problem," in *SFCS*, 1988, pp. 444–451.

[30] Y. fei Ma, J. Sheng, Y. Chen, and H. jiang Zhang, "Msr-asia at trec-10 video track: Shot boundary detection," in *TREC*, 2001.

[31] C.-R. Huang, H.-P. Lee, and C.-S. Chen, "Shot change detection via local keypoint matching," *IEEE Transactions on Multimedia*, vol. 10, no. 6, pp. 1097 –1108, Oct. 2008.

[32] W.-K. Li and S.-H. Lai, "A motion-aided video shot segmentation algorithm," in *Pacific Rim Conference on Multimedia*, 2002, pp. 336–343.

[33] M. J. Pickering and S. M. Rüger, "Multi-timescale video shot-change detection," in *TREC*, 2001.

[34] M. A. Webster, M. A. Georgeson, and S. M. Webster, "Neural adjustments to image blur," *Nature Neuroscience*, vol. 5, no. 9, pp. 839–840, 2002.

[35] G. Ciocca and R. Schettini, "Dynamic key-frame extraction for video summarization," in *Internet imaging VI*, 2005, pp. 137 – 142.

[36] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945. [Online]. Available: http://dx.doi.org/10.2307/3001968

[37] E. Peli, J. Kim, Y. Yitzhaky, R. B. Goldstein, and R. L. Woods, "Wideband enhancement of television images for people with visual impairments," *JOSA*, vol. 21, pp. 937–950, Jun. 2004.

[38] J. A. Hanley and M. B. J., "The meaning and use of the area under a receiver operating characteristic (roc) curve," *Radiology*, vol. 143, pp. 29–36, 1982.

[39] A. Kharlamov and V. Podlozhnyuk, "Image denoising," *NVIDIA Technical Report*, June 2007.

[40] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph-based video segmentation," in *CVPR*, 2010.

Anustup Choudhury received the B.E. degree in Computer Engineering from the University of Mumbai, Mumbai in 2004 and the M.S. degree in Computer Science from the University of Southern California in 2007. He is currently working towards the Ph.D. degree in Computer Science in the Institute for Robotics and Intelligent Systems, Viterbi School of Engineering, University of Southern California, Los Angeles. His research interests include image/video analysis and processing, pattern recognition, computer vision and machine learning.

Gérard Medioni received the Diplôme d'Ingenieur from ENST, Paris in 1977, a M.S. and Ph.D. from the University of Southern California in 1980 and 1983 respectively. He has been at USC since then, and is currently Professor of Computer Science and Electrical Engineering, co-director of the Institute for Robotics and Intelligent Systems (IRIS), and co-director of the USC Games Institute. He served as Chairman of the Computer Science Department from 2001 to 2007. Professor Medioni has made significant contributions to the field of computer vision. His research covers a broad spectrum of the field, such as edge detection, stereo and motion analysis, shape inference and description, and system integration. He has published 4 books, over 50 journal papers and 150 conference articles, and is the recipient of 8 international patents. Prof. Medioni is on the advisory board of the IEEE Transactions on PAMI Journal, associate editor of the International Journal of Computer Vision, associate editor of the Pattern Recognition and Image Analysis Journal, and associate editor of the International Journal of Image and Video Processing.

Prof. Medioni served at program co-chair of the 1991 IEEE CVPR Conference in Hawaii, of the 1995 IEEE Symposium on Computer Vision in Miami, general co-chair of the1997 IEEE CVPR Conference in Puerto Rico, conference co-chair of the 1998 ICPR Conference in Australia, general co-chair of the 2001 IEEE CVPR Conference in Kauai, general co-chair of the 2007 IEEE CVPR Conference in Minneapolis, general co-chair of the 2009 IEEE CVPR Conference in Miami, program co-chair of the 2009 IEEE WACV Conference in Snowbird, Utah, general co-chair of the 2011 IEEE WACV Conference in Kona, Hawaii, and general co-chair of the upcoming 2013 IEEE CVPR Conference in Portland, Oregon. He is a Fellow of IAPR, a Fellow of the IEEE, and a Fellow of AAAI.