

Analogies Based Video Editing

Wei-Qi Yan, Jun Wang, Mohan S. Kankanhalli

<http://www.comp.nus.edu.sg/~yanwq/ACMMMSYS/VideoAnalogies/acmmmjournal.htm>

Wei-Qi Yan and Mohan S. Kankanhalli are with the School of Computing, National University of Singapore, Singapore 117543. Jun Wang is with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands. This paper was submitted to the ACM Multimedia Systems Journal on Jul. 17, 2003; was first revised on Feb. 26, 2004 and the second revision was completed on Sep. 18, 2004. The paper was accepted on Jan. 13, 2005. Corresponding author : Mohan S Kankanhalli (e-mail: mohan@comp.nus.edu.sg).

Abstract

A well-produced video always creates a strong impression on the viewer. However due to the limitations of the camera, the ambient conditions or the skills of the videographer, the quality of captured videos sometimes falls short of one's expectations. On the other hand, we have a vast amount of superbly captured videos available on the web and in digital libraries. In this paper, we propose the novel approach of video analogies that provides a powerful ability to improve the quality of a video by borrowing features from a higher quality video. We want to improve the given target video in order to obtain a higher quality output video. During the matching phase, we find the correspondence between the pair by using feature matching. Then for the target video, we utilize this correspondence to transfer some desired traits of the source video into the target video in order to obtain a new video. Thus the new video will obtain the desired features from the source video while retaining the merits of the target video. The video analogies technique provides an intuitive mechanism for automatic editing of videos. We demonstrate the utility of the analogies method by considering three applications - colorizing videos, reducing video blurs and video rhythm adjustment. We describe each application in detail and provide experimental results to establish the efficacy of the proposed approach.

Index Terms

Artifacts removal, blur removal, colorizing, video analogies, automatic video editing, video rhythm.

I. INTRODUCTION

A technically perfect video can leave a deep impression on our minds. Unfortunately such videos can be hard to produce due to the limitations of devices (handheld camcorders), techniques (of amateur videographers), or ambient conditions (nightshot videos). Many videos have shortcomings, such as being blurred or being just a gray-scale video. Figure 1 lists three video frames with these artifacts. Figure 1(a) is a blurred campus map, Fig. 1(b) is an infrared video frame from the Iraq war while Fig. 1(c) is an X-ray transparency obtained from a health check-up.

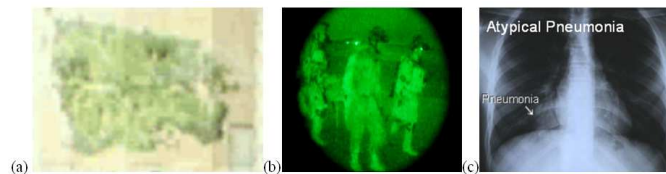


Fig. 1. Video frames with artifacts

Let us first consider the blur phenomenon in videos. Commonly listed reasons for blur are:

- Movements during the image capture process, by the camera or, when the subject uses long exposure times.
- Out-of-focus optics, use of a wide-angle lens, atmospheric turbulence, or a short exposure time that reduces the number of photons captured.
- Scattered light distortion in confocal microscopy.

Videos shot by camcorders on moving vehicles often capture blurred content due to motion. Capturing swiftly moving objects such as aircraft also leads to blurry videos. Rapid panning and zooming actions also generate blur. Another type of blur is caused by not focusing on objects clearly in time and is called the Gaussian blur. Some amount of blur is always presented in videos but too much blur negatively impacts the viewing pleasure. Moreover, it can lead to failures in retrieval and matching from video databases.

Another type of video artifact is the lack of color. The night-shot videos such as those taken during a war are usually “green-scale” which is attributed to the use of infrared rays for image capture. There exist other kinds of video artifacts as well.

On the other hand, a tremendous amount of professionally produced video is available. A lot of high-quality video material is freely accessible on the web. It would be a sheer waste if we cannot properly utilize this valuable material since they represent the collective accomplishments of some of the best videography experts. Such videos embody highly refined artistic features. Usually home videos do not possess this level of artistry because they are often produced by amateur photographers and editors. Given that most home video enthusiasts do not have the time, money or inclination to develop artistic skills, an alternative would be to transfer the refined features from professionally produced ones to the amateur videos. Our work here has been motivated by the desire to benefit from these superb quality videos. We propose a novel general scheme, namely video analogies, to transfer the designated features from one video to another so as to modify the target video.

The video analogy technique aims at learning the computable features of the source video so that the corresponding desired features can be transferred to the target video. *Computable* means that we should be able to extract and match the video features by computation (they are used in our matching step). For instance, there exist many aesthetic features in videos, some of those can be extracted, but some (e.g. mood) cannot be done as of now. The crucial steps for video analogies are feature matching and transfer.

We now provide a general description of our scheme. The video analogies method is illustrated in Fig. 2. Given a target video B (e.g. a grayscale video) and a designated source video A' (e.g. a color

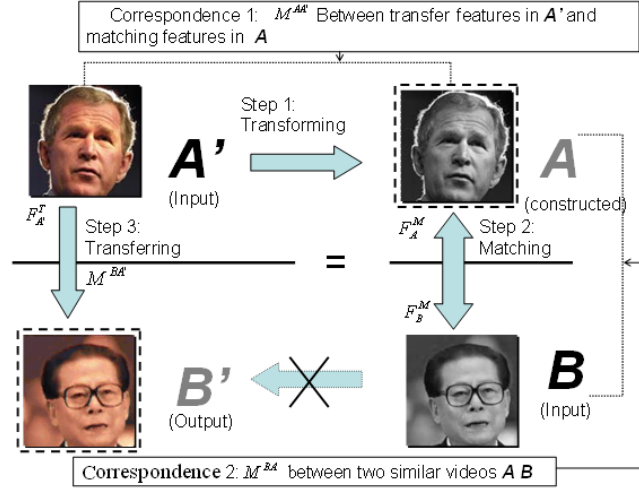


Fig. 2. The illustration of Video Analogies. B : target video; B' : target video with desired features (output result); A' : designated source video; A : constructed source video having a common feature with B

video) with similar content. The problem is how to transfer the designated feature (e.g. color) in the source video A' to the target video B (i.e. grayscale image). To this end, we first build a bridge between A' and B . The bridge should have correspondences with those two videos respectively in order to link the video B and A' . In other words, in the analogy notation: $A : B :: A' : B'$, we should construct A to make an analogy between the source and target pair, and consequently transfer features from A' to B . Thus, we achieve this by constructing the bridge (denoted by A). This is done by transforming the high quality source video A' to the one which has some features in common with B , e.g. in this case, the grayscale version of A' . By this transformation, it is clear that A has two correspondences: the relationship with A' (mapping between two different distinct features, i.e. color and grayscale) and relationship with B (mapping between a common feature of two videos, i.e. texture feature in A and B). Now we match the common feature (i.e. texture feature) between A and B in order to find the relationships between A and B . Finally, the designated feature (i.e. color) is transferred from A' to B based on the two correspondences obtained earlier.

The utilized features are typically chosen from Table I. We call the features used for matching purpose between A and B as *matching features* or *common features*. Similarly we call the features for the purpose of transferring as *transfer features* or *designated features*. For instance, we can build a texture correspondence between the video pair (A and B) and then transfer the color feature of the source video to the target video. What we would like to point out here is that the compared videos should be similar

in terms of video content at the shot level so that a proper analogy is set up. They should have *similar* objects and backgrounds for best results.

The framework of video analogies involves three phases as shown in Fig. 2: *transformation*, *matching* and *transfer*. The inputs are a source video A' and a target video B . In order to extract features, the transformed source video A' has to be constructed (by transforming) beforehand. At the *matching* stage, the corresponding computable features (matching features) in the transformed source video A and the target video B need to be extracted and compared for similarity ($A : B$). During the *transfer* stage, we establish a new function to transfer the desired features (transfer features) from the source video A' to the target video to create the new video B' by employing the analogy $A : B :: A' : B'$. The source video is assumed to be a high-quality clip that we wish to emulate. The target video possibly has the artifacts or shortcomings that we wish to overcome by mimicking the source video. We have observed that our framework of video analogies can potentially be utilized for several problems:

- *Color transfer:*

We can transfer suitable colors from a source video to a target video, such as transferring the source colors to a grayscale (X-ray or infrared) video. This can generate a more vivid and attractive video [1][2][3].

- *Texture transfer:*

By transferring the specific texture to certain areas, we can overlap and patch areas on video frames such as annoying video logos [4].

- *Motion trajectory transfer:*

By transferring the desired motions to the target video, we can actually remove or reduce shakes caused by camera vibration. Conversely, we also can borrow the shakes from a source video to add excitement to a target video.

- *Music matching:*

Music is an indispensable component of videos [5]. Automatic music matching for atmosphere enhancement is a crucial step in video editing. From the source videos, we can track features in order to add similar music to the target video.

- *Aesthetic styles transfer:*

Artistic styles in a professional movie embody the experiences and knowledge of the directors. Most of these artistic features are at the semantic level; however some of them are apparent at the low level itself [6][7], such as color, lighting, motion trajectory and rhythm. Such stylistic aspects can

be advantageously transferred.

Although video analogies techniques thus can be applied in multiple situations, in this paper we focus our investigations on three problems: colorizing videos, video blur reduction and video rhythm adjustment. To the best of our knowledge, this is the first time the concept of video analogies has been proposed, and it is also the first time that the video analogies techniques have been used to remove video artifacts.

The structure of this paper is the following: we would like to report related work in section II, and our framework with respect to video analogies is proposed in section III. Section IV will discuss our applications and section V will demonstrate our experimental results. We end with a summary and our future research intents.

II. RELATED WORK

In [8], Hertzmann et al proposed example-based image processing using image analogies. Under this framework, two phases: a design stage and an application stage are defined. This framework supports a wide variety of image effects such as filtering and texture transfer. The advantage of image analogies is that it provides a very natural means of specifying image transformations. Moreover, image analogies may learn complex image processing from the comparison. The same general mechanism could be used to cater for a wide variety of applications. This algorithm can be implemented in a real-time interactive system. Image analogies are not only employed for low level processing but can also be employed at a semantic level.

Curve Analogies [9][10] is a technique for learning statistical models for 2D curves. The target of curve analogies is to automatically learn how to generate an output curve from an input curve. Curve analogies can not only modify existing curves, but can also create new curves.

Analogies are a fundamental concept in other research areas such as artificial intelligence and machine learning [11][12]. Color transfer between images is an instance of image analogies. The inputs are a color image and a grayscale image. The color image is first transformed to a grayscale image and then the corresponding regions on the two grayscale images are matched. The matching results are retained for transferring. In the implementation phase, the color in a portion of the color image is transferred to the corresponding region on the grayscale image. The new colorized image is the output [1][2]. An interesting aspect of this color transfer is that it obtains the cue for matching from the input color image itself. The input color image provides a grayscale image for matching. We have studied this color transfer problem for video data and have introduced a new transfer technique for colorizing infrared home videos [3]. Our algorithm mainly employs correlation existing in videos to efficiently render an infrared

home video using color key frames. We basically establish a color palette for a video shot for the sake of rendering efficiency.

To the best of our knowledge, this is the first time that video analogies have been used for video processing [13][14][15]. Our work is quite different from image and curve analogies since we need to take the unique aspects of videos into account. What the most important is that a video has totally different features than other media such as images, graphics, audio and etc, therefore, we exploit both temporal and spatial correlations. Also, instead of three inputs in image and curve analogies, we often use only two inputs because a video is a rich medium possessing several potentially useful features. Also, many types of feature transfers are unique to videos. There is a growing interest in the general area of computational media aesthetics [16][17][18]. Video analogies appear to be useful in this area.

III. THE METHOD OF VIDEO ANALOGIES

Our general framework of video analogies is introduced below.

A. Features for Matching and Transferring

Features, which are extracted within a single frame, are called *local features* while features extracted over a shot are called *global features*. We now define the features used in the video analogies framework:

$F_{A'}^T(x, y, t)$: The designated features in A' that need to be transferred, where subscript A' refers to the source video, T means it is the transfer feature, t is the video frames index and x, y are the coordinates in the frame plane. For instance, the color feature in A' is used in Fig. 2.

F^M : The matching features in A and B that are used to set up the analogy between A and B . They come from both the source and target videos.

$F_A^M(x, y, t)$: Denotes the matching features in the source video A while $F_B^M(x, y, t)$ denotes the matching features in the target video B , where the superscript M implies the mapping aspect, and A and B mean features come from the source video and the target video respectively (the definition of x, y , and t being the same as earlier). One of the simple features is the texture feature in A and B as utilized in Fig. 2.

Video features, unlike image features, include temporal and spatial features. Theoretically, each individual feature of a video can be extracted; however, not all features are useful as matching and transfer features.

The general criteria for feature selection are that we require that the extracted features should be *computable* and *invertible*. These are two mandatory conditions. The concept *invertible* refers to the

ability to modify (add/subtract or overwrite) the target video by utilizing the extracted features. The transfer features (denoted as $F_{A'}^T(x, y, t)$) should be invertible since those features finally will be added into the target video B . Thus these features can be extracted from a video and they can also be inserted into a video.

TABLE I
THE RELATIONSHIP BETWEEN THE MATCHING FEATURES AND TRANSFER FEATURES

Features Type	Features for matching	Features for transfer
<i>Local features</i>	Texture	Light, Contrast, Hue, Saturation, Color, Texture, Blurring
	Motion vector	Motion vector, Motion trajectory, shakes
	Energy/Entropy in frames	Color, Lighting, Histogram
<i>Global features</i>	Motion trajectory	Motion trajectory, shakes, Rhythm
	Rhythm	Rhythm, Music, shakes

Table I lists the computable and invertible video features. Features for matching are given in the left column, while the features for transfer are shown in the right column. For example, we can transfer the color information based on texture matching, but the converse is not true.

B. Mapping Functions

As discussed earlier, the key issue is, given the feature F_B^M in the target video B , finding the suitable $F_{A'}^T$ in the source video A' in order to transfer it to the target video B . Here the term *transfer* means adding on or overwriting the feature F_B^M by using the feature $F_{A'}^T$ in order to bring the designated feature to the target video. Our video analogies technique is the precise solution to this mapping problem. We define this mapping as $M^{BA'} : F_{A'}^T \rightarrow F_B^M$. Since our aim is to transfer designated features from A' to B , we use the superscript B and A' to denote this transfer $M^{BA'}$.

$M^{BA'}$ can be obtained by solving the two step matching problem explained as follows:

Correspondence 1: By transforming the source video A' to A , we obtain the mapping function between $F_{A'}^T$ and F_A^M . Since A is constructed from A' , the relationship of the features between those two videos remains unchanged. We can easily define the mapping function as Eq.(1):

$$\begin{aligned}
 M^{AA'} : F_{A'}^T(x_{A'}, y_{A'}, t_{A'}) &\rightarrow F_A^M(x_A, y_A, t_A) \\
 \text{iff} \quad x_{A'} &= x_A, y_{A'} = y_A, t_{A'} = t_A
 \end{aligned} \tag{1}$$

Correspondence 2: The mapping feature F^M links video A with video B . We map the features from $F_A^M(x, y, t)$ to $F_B^M(x, y, t)$ by calculating their similarity. We define this mapping function as M^{BA} . It can be formulated by calculating similarities of frames and blocks as shown in Eq.(2):

$$\begin{aligned} M^{BA} : F_A^M(x_A, y_A, t_A) &\rightarrow F_B^M(x_B, y_B, t_B) \\ \text{iff} \quad S_{fr}(Fr(t_B), Fr(t_A)) &< T_{fr} \\ \text{and} \quad Sb(F_B^M(x_B, y_B), F_A^M(x_A, y_A)) &< T_b \end{aligned} \quad (2)$$

where $S_{fr}(\cdot)$ denotes the frame level similarity function and the $Sb(\cdot)$ denotes the block level similarity function. T_{fr} and T_b are their associated thresholds. $Fr(t)$ is the frame at time t . The details of the similarity measurement at both frame and block levels are described in the next section. After obtaining the two correspondences M^{BA} and $M^{AA'}$, we can easily derive $M^{BA'}$ as follows:

$$M^{BA'} = M^{BA} \cdot M^{AA'} \quad (3)$$

It is clear from Eq.(3) that the new video B' cannot be directly obtained from A' . The desired features from A' reach B via A . This three-step procedure is shown in Fig. 2.

C. Similarity Measurement

For two video shots $V_1 = \{Fr_1(1), Fr_1(2), \dots, Fr_1(m)\}$ and $V_2 = \{Fr_2(1), Fr_2(2), \dots, Fr_2(m)\}$, $Fr_i(j)$, $i = 1, 2$; $j = 1, 2, \dots, m$; are the video frames. The similarity measurement for videos can be at two levels: frame level and block level, we will describe them in the following subsections.

1) *Frame Level Similarity:* Given an arbitrary frame in video V_2 , we can find a similar frame in video V_1 in terms of a similarity function $S_{fr}(\cdot)$, i.e. $\forall Fr_2(j) \in V_2, j = 1, 2, \dots, n; \exists Fr_1(c) \in V_1, S_{fr}(Fr_2(j), Fr_1(c)) = \min(\{|g(Fr_2(j)) - g(Fr_1(i))|, i = 1, 2, \dots, m\})$; $g(\cdot)$ is a weight function which is employed to measure the similarity degree of two video frames. Note that the weight function $g(\cdot)$ (used to measure the similarity between two corresponding elements in a certain space) is a general concept. Many distance functions can be adapted as the weight function $g(\cdot)$.

The choice of the similarity function $g(\cdot)$ is dependent on the kind of feature that needs to be transferred. For transferring local features such as color, motion vector, the spatial-domain color distance, histogram distance based on statistics, energy and entropy distance, all can be used as the weight function.

As an example, the video entropy can be used as the weight function for the local features as shown in Eq.(4). It is usually used to measure the similarity of videos [19] since it reveals the information capacity of a video unit (block or frame).

$$g = - \int_{x_1}^{x_2} p(x) \ln p(x) dx \quad (4)$$

where $[x_1, x_2]$ is the color domain, $p(x)$ is the probability, which for instance in a discrete case, can be:

$$p(i) = \frac{b_i}{W \times H}, i = 0, 1, \dots, 255. \quad (5)$$

where b_i is the histogram of features (e.g. intensity), W and H are the width and height of the video units respectively. If we use Eq.(5) for measuring the frame level similarity, W and H are the width and height of the frame respectively.

For transferring the global features such as motion trajectory or rhythm, one has to map the entire shot from the source video to the target video. Therefore, instead of a distance function, other mapping methods have to be used. For instance, if there are N_1 frames in the source video and N_2 frames in the target video, given a frame n in the target video, the corresponding frame m in the source video can be found by using a linear scaling mechanism Eq.(6). This is the simplest mapping function between video frames or blocks.

$$m = \frac{n \cdot N_1}{N_2} \quad (6)$$

2) *Block Level Similarity*: When the corresponding frames are found, then we need handle the target video at the block level. Among the blocks belonging to the two different frames, we obtain the most similar matching blocks in the frame pair as shown in Fig. 3.

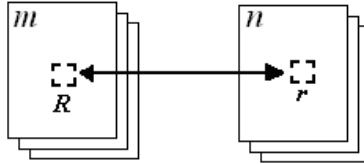


Fig. 3. Feature Matching

Frames $Fr_1(j) \in V_1$ and $Fr_2(j) \in V_2$ are divided into blocks $R_{m,n}$ and $r_{u,v}$, $m = 1, 2, \dots, M$; $n = 1, 2, \dots, N$; $u=1, 2, \dots, U$; $v=1, 2, \dots, V$; M, N, U and V are the maximum resolutions respectively. We can find the most similar blocks by using the block similarity function Sb , i.e. $\forall R_{m,n} \in Fr_1(j)$ in video V_1 , $\exists c_1, c_2$, $|g(R_{m,n}) - g(r_{c_1, c_2})| = \min_{r_{u,v} \in V_2} (\{|g(R_{m,n}) - g(r_{u,v})|\})$.

For instance, the similarity function $Sb(\cdot)$ for two blocks can be defined as the shortest distance (under the p norm) among all the candidate blocks in terms of distance according to Eq.(7):

$$Sb(R_{m,n}, r_{u,v}) = \min_{r_{u,v} \in V_2} (\{\|R_{m,n} - r_{u,v}\|^q\}), q > 0 \quad (7)$$

Alternatively, we can also use the video entropy as the weight function, instead of the Euclidean distance. The equation is shown as follows:

$$Sb(R_{m,n}, r_{u,v}) = \min_{r_{u,v} \in V_2} (\{|g(R_{m,n}) - g(r_{u,v})|\}) \quad (8)$$

where $g(\cdot)$ is replaced by Eq.(4).

The full positional correspondence between the two videos is established after all the matching blocks $R_{m,n} \in Fr_1(j)$ in video V_1 among all the frames in the target video B are found.

D. Feature Transfer

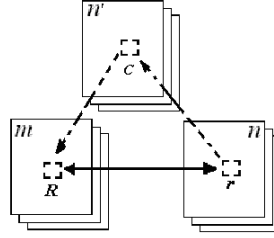


Fig. 4. Invertible feature transfer after computable feature matching

For feature transfer, we employ the map between video features denoted as $M^{BA'}$ (see Section III-B) to handle the target video B since the mapping provides heuristic clues. We use these hints to transfer the new features to the target video. These clues or hints are the corresponding values of the computable video features. The goal of the feature transfer step is to generate a new video by using these values. In the following paragraphs, we will elaborate on the details.

Figure 4 explains the feature transfer process at the block level. In Fig. 4, n' is one of frames of the source video A , m is one of frames of the target video B , n is one of frames transformed from the source video A' , n and n' have certain transform relationships. For example, n' is a color frame, n is the corresponding grayscale frame, m is another grayscale frame for color transfer. We obtain the correspondence from block based matching, namely, the positional correspondence (solid line) of block r in frame n and block R in frame m . We can then project block c in frame n' and transfer other information from block c in n' to block R in frame m according to the matching result. After all the blocks in video are transferred, a new video B' is created.

E. Framework

Input : Source Video A' , Target Video B

Output : Modified Target Video B'

Procedure:

< Transforming >

1. $A \leftarrow \text{transform}(A')$;
2. $F_A^T(x, y, t) \leftarrow \text{extract_feature}(A)$;
3. $F_{A'}^T(x, y, t) \leftarrow \text{extract_feature}(A')$;
4. $M^{AA'} = \text{mapping}(F_A^T, F_{A'}^T)$;

< Matching >

1. $F_A^M(x, y, t) \leftarrow \text{extract_feature}(A)$;
2. $F_B^M(x, y, t) \leftarrow \text{extract_feature}(B)$;
3. $M^{BA} = \text{mapping}(F_A^M, F_B^M)$;

< Transferring >

1. $M^{BA'} = M^{BA} \cdot M^{AA'}$;
2. $B' = \text{transfer}(M^{BA'}, B)$;

Algorithm 1: Video Analogies in shots ($A' : A :: B : B'$)

Our overall framework can be succinctly captured by the core analogies algorithm 1. One significant difference between our proposal and other analogies (curve analogies, image analogies) is that we use only two input videos (A and B) instead of three (A , A' and B). This is because that we utilize the mutual information of the input videos (A and A') and build a bridge between them. This is possible because video data hosts many features simultaneously. For instance, video has temporal features, spatial features and spatial-temporal features. One interesting advantage of our framework is that though a video possesses several features at the same time, we extract and learn only one computable feature (for matching purpose) from the video pair, however, we can transfer multiple features from the source video A to the target one B . This is an advantage, for instance, after getting the texture correspondence, we can not only transfer the color information to the target video, but also we can transfer the lighting/motion features to the target video. Once we find an appropriate correspondence, it can be employed to modify several features which is an advantage.

F. The selection of source video

In video analogies, the selection of the source video depends upon the particular editing task. It really depends on what possible source videos are available and what is the desired effect to be transferred. For many tasks in media synthesis, it also does depend upon the video creator (more accurately, the editor in our case). Different editors may select different videos as their template video. Our default procedure is therefore to select the source video manually. However, in order to help automate the source video selection from a collection, we propose the following approach.

In the video analogies technique, users select the source video which has the desired features and intend to transfer the designated features to the target videos. In the algorithm described earlier, the feature $F_A^{T'}$ will be transferred to the location of the matched feature F_M^B in the target video B . It therefore requires that the source video and the target video belong to the same domain and have similarities in terms of objects or scenes. This is essential in order to have a meaningful transfer. However, it is difficult to develop a similarity metric to measure the similarity between the transformed source video A and the target video B . This is defined in general as follows by using the block-wise distance function:

$$d(V_A, V_B) = \sum_{r_{u,v} \in V_B} \min_{R_{m,n} \in V_A} (\{|g(R_{m,n}) - g(r_{u,v})|\}) \quad (9)$$

where $R_{m,n}$ and $r_{u,v}$ are the blocks in the transformed video A and the target video B , respectively. $g(\cdot)$ is the entropy defined in the equation (4). The weight function $g(\cdot)$ should select the pertinent features (e.g. color) of a frame for comparison. Following this, we then can develop a criterion to help user judge whether the source video is suitable for use in transfer.

If we have a finite collection of videos at hand: $\mathcal{V} = \{A_1, A_2, \dots, A_n\}$, where n is the size of the collection. We select an arbitrary video $A_i \in \mathcal{V}$, $i = 1, 2, \dots, n$ to compare with the target video B and calculate $d(A_i, B)$. If $\exists A_m$, $1 \leq m \leq n$, $d(A_m, B) = \min\{d(A_i, B), i = 1, 2, \dots, n\}$, then we select A_m as the source video. For an example, Fig. 5 illustrates the procedure of source frame selection, when used for color transfer.

In Fig. 5, we show some source frames in the second row and use them to render target image (the image in the first column). The resultant rendered images are also shown. The comparison results using Eq.(9) is given in the bottom-most row. We see that if person A's photograph is used to render the target photo, it has the minimum distance at the block level. Thus, the conclusion is that person A's photograph is the best source among all of the candidates in this finite set.















Samples	Person A	Person B	Person C	Person D	Person E	Person F	Person G
128x128							
							
Distance	<u>1720643</u>	2302308	1898930	1841469	2906063	1808338	4241766

Fig. 5. Source video selection

Note that in this case, Eq.(9) takes this form:

$$d(fr_A, fr_B) = \sum_{r_{u,v} \in fr_B} \min_{R_{m,n} \in fr_A} (\{|g(R_{m,n}) - g(r_{u,v})|\}) \quad (10)$$

where fr_A and fr_B are video frames, $|g(R_{m,n}) - g(r_{u,v})| = \sum_{i=1}^5 \sum_{j=1}^5 |G_{R_{m,n}}(i, j) - G_{r_{u,v}}(i, j)|$, $G_{R_{m,n}}(\cdot)$ and $G_{r_{u,v}}(\cdot)$ are the grayscale blocks of $R_{m,n}$ and $r_{u,v}$ respectively, with both of their resolutions being 5×5 in our implementation.

Alternatively, the following approaches can also be used for source selection:

- We can employ the video matching technique [20] to decide the most appropriate source video. If we have many candidate clips in our collection, we select the one which has the least distance among all the given videos.
- We can use of subjective evaluation. We may choose the source video by using subjective judgement. We can utilize the service of a video professional. Or we can rely on the opinion of several non-professional subjects and select the majority choice. In this case, the subject choices are considered as evidences for judging the suitability of the source video.

IV. APPLICATIONS

Although video analogies have many applications, we will now describe three specific applications: video colorizing, video blur reduction, and video rhythm adjustment.

A. Colorizing Videos

Grayscale videos, such as medical videos and night-shot videos, do not possess colors due to the underlying physics of imaging or due to the use of infrared and X-rays. How to colorize them is a really

interesting and challenging issue. Pseudo-colorizing algorithms lead to unnatural looking videos. In this paper, we colorize these grayscale videos by using video analogies.

It is always possible to transform the color video to grayscale and X-ray or infrared video. For example, an X-ray video is the negative of a grayscale video while an infrared video is really a green scale video. Given a source video (a color video) and a target video (grayscale or X-ray or infrared video), we first obtain the video A by transforming the source video A' into the grayscale or X-ray or infrared video. Thus, we setup the color palette with respect to grayscale and color values. In this case, the mapping function $M^{AA'}$ between features $F_{A'}^T$ and F_A^M is set up in the form of the color palette. Next, we match the texture features from the constructed grayscale video A and the target video B , and establish a correspondence between the blocks, denoted as M^{BA} . Finally we can map the palette onto the frames of the target video (as the form of $M^{BA'}$) and transfer color to the target video B to create the natural looking target video B' .

We would like to formally describe our algorithm. After transformation, we obtain the grayscale source video shot (after segmenting the source video A): $V_1 = G_{M,N,N_1}$, M and N are the width and height with a total of N_1 frames; the grayscale video shot (after segmentation to the target video B) is $V_2 = G_{W,H,N_2}$, W and H are the width and height with a total of N_2 frames. First, we perform frame matching between A and B . For an arbitrary frame $n_2 < N_2$ in V_2 , we can find a corresponding frame $n_1 < N_1$ in V_1 using Eq.(1) and Eq.(3) so that we can map the information from the frame n_1 to the frame n_2 . We also can expand the matching range of candidate frames w by giving a parameter $\delta > 0$, $w = n_1 - \delta, n_1 - \delta + 1, n_1 - \delta + 2, \dots, n_1 + \delta$ extending even to the entire shot.

Second, we divide the frames into blocks and match the similar blocks in frame n_1 in video V_1 and frame n_2 in video V_2 by using color energy. We select the block pairs with the minimum distance as the matched blocks. We define color energy as follows:

$$CE = \int_{x_0}^{x_1} \int_{y_0}^{y_1} ((r_1(x, y) - r_2(x, y))^2 + (g_1(x, y) - g_2(x, y))^2 + (b_1(x, y) - b_2(x, y))^2) dx dy \quad (11)$$

where CE is the color energy, $(r_1(x, y), g_1(x, y), b_1(x, y))$ is the color of pixel $(x, y) \in [x_0, x_1] \times [y_0, y_1]$, and $(r_2(x, y), g_2(x, y), b_2(x, y))$ is the color of pixel $(x, y) \in [x_0, x_1] \times [y_0, y_1]$. We obtain the matching blocks by computing the color energy. Given a block in frame n_1 and given candidate blocks in frame n_2 , suppose their color energy is $CE_{i,j}$, color energy of the best matching blocks is $CE_{x,y}$, then:

$$CE_{i,j} \geq CE_{x,y} \quad (12)$$

Input : A source video and a target grayscale video

Output : The colorized video

Procedure:

< Transformation >

De-colorize the source video A' to a grayscale video A .

< Matchingframes >

Map grayscale video A to grayscale video B by using the Eq.(1) and Eq.(3).

< Matchingblocks >

Find the best matching blocks between Fr_A in A and Fr_B in video B in Step 3 by using Eq.(4).

< Transferfeature >

Transfer the colors of the source to the corresponding pixels in grayscale frames of the target video.

Algorithm 2: Rendering a grayscale video

where $(x, y) \in [x_0, x_1] \times [y_0, y_1]$ in frame Fr_{n_1} is the position of the best matched block to $(i, j) \in [x_0, x_1] \times [y_0, y_1]$ in frame Fr_{n_2} .

The reason why we have selected color energy is that it reveals the real pertinent color differences between these regions. Actually, given a color frame of the source video (which is equivalent to having a color palette), the palette is used to colorize the current gray frame of the given target video. After transferring the given colors to all the corresponding pixels in grayscale frames, a color image sequence is created. Our algorithm can be summarized as following algorithm 2.

Compared to our earlier work on colorizing infrared home videos [3], the advantage of the algorithm provided in this paper is that we find the corresponding color frame related to current grayscale frame automatically. This eliminates the need for manually assigning the color key frames. This also leads to more efficient processing.

B. Video Blur Reduction

Blurs stemming from geometric optics are introduced into various optical devices. The video camera therefore also has this feature. In geometric optics, optical blur is defined as the position error of a

projection passing through a convex or concave lens that exceeds one hundredth of the lens foci.

Before, we can de-blur videos, we need to identify which frames of a video are blurred. For the sake of completeness, we first provide the algorithm for blur detection and then provide the video analogies based solution to reduce the blur.

1) *Blur Detection*: Blur detection has many solutions [21][22]. Our principle for blur detection is that the lesser the edge information in a frame, the higher is the possibility of it being a blurred frame. Thus blurred frames can be detected by counting the edges in the frames. The edges of a frame are obtained by using the first or second derivative [19][23] [24]. We use the Sobel operator to extract the edges because it can take the correlation of videos into account and does not need much computation. Our experiments also show that the results using Sobel operator is better than that of the Canny edge detector.

In Eq.(13), the Laplace operator is used to extract the edges while in Eq.(14), the Sobel operator is used to extract gradient information. Eq.(15) is used to find the edges and finally Eq.(16) is used to compute the degree of blur.

$$f_{smooth}(x, y) = \sum_{(m,n)} h_G(m, n) f(i - m, j - n) \quad (13)$$

where $f(\cdot)$ is a pixel value, $h_G(\cdot)$ is Gaussian distribution function.

$$h_G = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$f_{edge}(x, y) = \sum_{(m,n)} h_s(m, n) f_{smooth}(i - m, j - n) \quad (14)$$

where:

$$h_s = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\bar{f}_{edge}(x, y) = \max(\{|f_{edge}(x, y)|, T_1\}) - T_1 \quad (15)$$

where T_1 is a threshold along the X -axis.

$$mean_{edge} = \frac{1}{W \times H} \sum_{(x,y)} \bar{f}_{edge}(x, y) \quad (16)$$

where W and H are width and height of video frames respectively. Our algorithm can be described as following algorithm 3:

Input : A frame from a video

Output : The detected blur degree

Procedure:

1. Get a smoothed image by using Eq.(13);
2. Calculate the edge map by using Eq.(14);
3. Remove low valued edges using the threshold in Eq.(15);
4. Obtain the mean value of the threshold edge map Eq.(16);
5. If the mean value is less than the designated threshold, the frame is blurred. Otherwise the frame is a good quality frame. We have normalized the blur degree in the range $[0, 10]$.

Algorithm 3: Blur detection

In algorithm 3, the mean value in step 5 is defined as the degree of blur. This degree is helpful in deciding the extent of the blurring of a video frame. In this algorithm, two thresholds are required for blurring degree determination; one is T_1 in Eq.(15), another is for the mean value in Eq.(16). Their values determine the blurring degree of the current frame.

2) *Video Analogies Based Blur Reduction:* In order to remove the blurring in frames or blocks, traditionally Wiener filter, regularized filter and blind de-convolution filter are utilized [19][25]. For video blur reduction, super-resolution is usually employed [21][26][27]. But these approaches usually require the camera parameters and do not work well. For instance, Wiener filter based blur reduction heavily relies on the probability distribution, which we cannot precisely obtain in practice. Thus, the result is not good, even much worse than others as shown in Fig. 6, Fig. 6(a) is the original video. Fig. 6(b) is the de-blurred video using Wiener filtering.

In this section, we provide a novel solution for blur reduction. It is quite simple but very different from the classical methods since we employ a video analogies based technique. Our initial idea was inspired by our earlier work on adjusting lighting of home videos by using histogram equalization [28]. We noticed that after histogram equalization, the video quality is improved significantly and the blurry video becomes much sharper than before as shown in Fig. 7. Figure 7(a) is the original blurry video, Fig. 7(b) is the video after histogram equalization. We can see that the video quality is improved significantly.

In order to improve the contrast, histogram equalization is usually the method of choice. Histogram

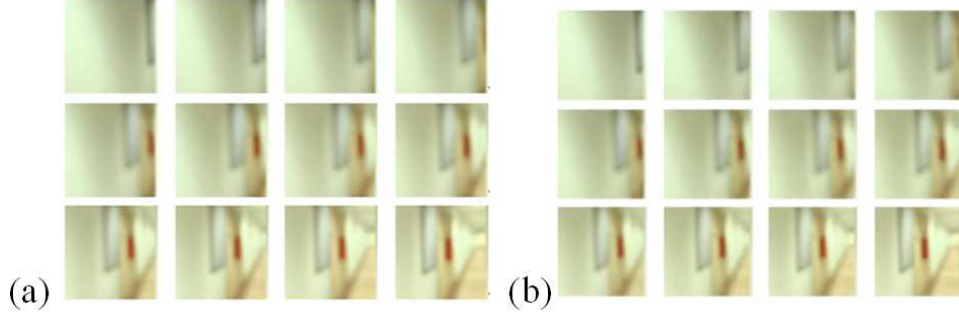


Fig. 6. Video blur reduction by using Wiener filter

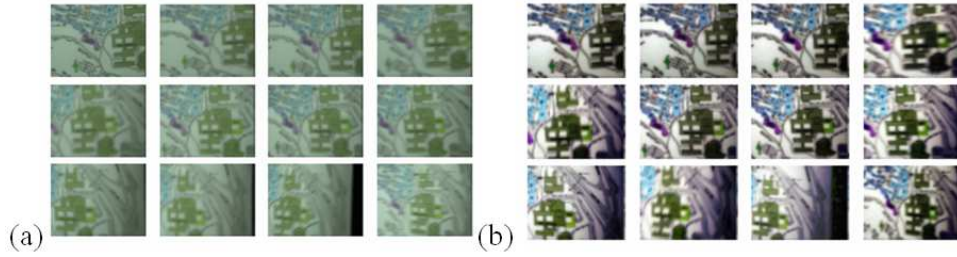


Fig. 7. Improved quality video after histogram equalization

equalization extends the histogram from a limited interval to a wider one. But this changes the color information significantly and even introduces color distortion. Moreover, histogram equalization cannot remove the blur, we therefore adopt the approach of mimicking the histogram of a similar sharp video in order to reduce the blur. Thus, in this paper, we reduce the blur by using a Bezier curve based nonlinear enhancement.

In Fig. 8, the color information is from point $(0, 0)$ to point $(255, 255)$ along the solid line while the histogram equalization method is indicated by dotted-dashed line. In order to enhance the video quality, we need to change the color distribution, and make the differences between different colors more significant. Thus we use the Bezier curve shown as dashed curves in Fig. 8 to reduce the blur. Cubic Bezier curves are described as follows:

$$B(t) = \sum_{i=0}^3 P_i B_3^i(t), B_3^i(t) = C_3^i t^{3-i} (1-t)^i, t \in [0, 1] \quad (17)$$

where P_i are the control points in color space, for example, $(0, 0)$, $(128, 0)$, $(128, 255)$ and $(255, 255)$. $B_3^i(t)$ are Bernstein polynomial, and t is the parameter. In order to describe a family of Bezier curves, rational Bezier curves are recommended:

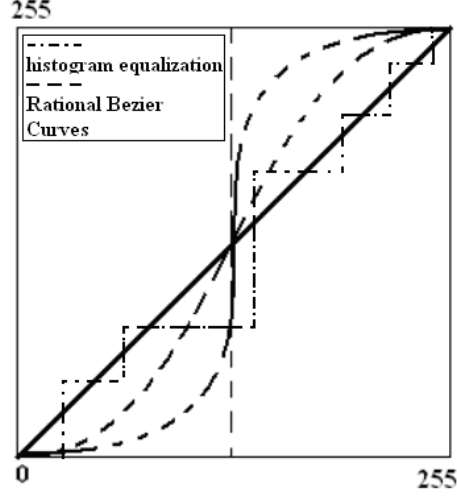


Fig. 8. Video analogies based blur reduction

$$RB(t) = \frac{\sum_{i=0}^3 w_i \cdot P_i \cdot B_3^i(t)}{\sum_{i=0}^3 w_i \cdot B_3^i(t)}, \quad (18)$$

where $w_i \in (-\infty, \infty)$ is the weight. If we increase the weight w_i to infinity, the rational curves will move closer to the control vertex P_i [29]. The advantage of using rational Bezier curves is that we just need to adjust the weights rather than move control points in order to reduce blur. If all the weights are equal to 1, the rational Bezier curve is a Bezier curve.

With the aid of rational Bezier curves, we may modify the color distribution of the video frames. We modify the luminance so as to reduce the blur of video frames. In order to achieve our goal, we should take the boundary also into account.

Suppose the luminance distribution of the source video is within $[SC_m, SC_M]$ and the luminance distribution of the target video is within $[TC_m, TC_M]$. Each pixel $c \in [TC_m, TC_M]$ in the target video will be linearly mapped onto $C \in [SC_m, SC_M]$ according to Eq.(19):

$$C = \frac{c - TC_m}{TC_M - TC_m} \cdot (SC_M - SC_m) + SC_m \quad (19)$$

In the implementation of the de-blurring adjustment based on cubic Bezier curves, end control points are given by $(SC_m, 0)$ and $(SC_M, 255)$. The middle control points are given by $(MSC, 0)$ and $(MSC, 255)$, where $MSC \in [SC_m, SC_M]$, whose histogram distribution occupies fifty percent shown in Fig. 9.

Figure 9 visually illustrates the mapping in de-blurring adjustment. The histogram plots are given

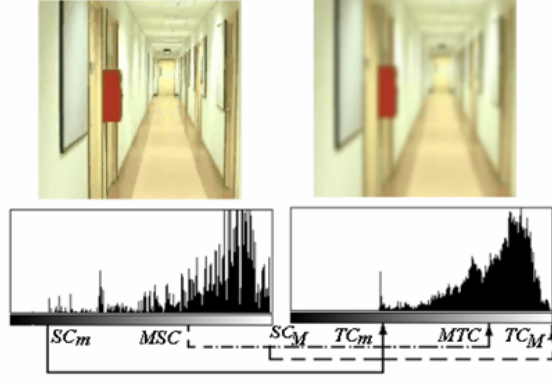


Fig. 9. Mapping in blur reduction

below the two color images. From the histogram graph, we find the transform domains $[SC_m, SC_M]$ and $[TC_m, TC_M]$, the start correspondence is indicated by dotted line; the end correspondence is given by dashed line and correspondence of the middle points is marked by dashed and dotted line.

Thus, for the sharp source video A' , we first smooth it to obtain a less sharp one which is same as the target video. We denote the transformed one as $A = V_{M,N,N_1}^T$, M and N are the width and height of the source video with total N_1 frames. $B = v_{W,H,N_2}^T$, W and H are the width and height of the blurry target video with total N_2 frames. For arbitrary $n_2 \leq N_2$ in B , we can find a corresponding frame $n_1 \leq N_1$ in A based on Eq.(6) so that we can transfer the information from frame n_1 to frame n_2 . We extract the blur information and build the corresponding relationship according to Fig. 9. We transfer the de-blurring information from A' to the B by using Eq.(19). If the work to reduce blur is performed at the block level, we need to find the similar region by using Eq.(11). Once all the regions in each frame are substituted, the new blur reduced video is created. Our algorithm for blur reduction is described as following algorithm 4.

Compared to other classical algorithms for blur reduction, our method is quite efficient. The core of our proposal is that we adjust the blur frames by using video analogies. However other algorithms need find the information (such as the PDF in Wiener filter) from the image sequence in a video first, and then synthesize a new video. If such a PDF cannot be found, the quality of blurry video cannot be improved. Without any such constraints, our method can borrow the information from the source video and transfer it to the target video. Our limitation is that an appropriate source video needs to be utilized for good results. This is a tremendous advantage of the video analogies approach.

Input : A sharp source video A and a blurry target video B

Output : The blur reduced video

Procedure:

1. $A' = \text{smooth_transform}(A)$;
2. $M^{BA} = \text{frame_mapping_texture}(F_A^M, F_B^M)$ by using Eq.(6);
3. $M^{BA} = \text{block_mapping_texture}(F_A^M, F_B^M)$ by using Eq.(11), this step only is needed when our work is performed at block level;
4. $M^{BA} = \text{block_mapping_contrast}(F_A^M, F_B^M)$; Mapping the frames of the target video according to Eq.(1);
5. $M^{BA'} = M^{BA} \cdot M^{AA'}$, $B' = \text{transfer}(M^{BA'}, B)$; Adjust the video frame according to Eq. (17) or Eq.(18);

Algorithm 4: Blur reduction

C. Video Rhythm Adjustment

Video rhythm refers to the duration and frequency of segmented events, it is subject to the pace of the events and the relationships between these events [16][30][31][32] [33]. The overall rhythm is usually determined by the transitions between video components such as shots, scenes, and sequences. The work presented in [30][31][32] has described many original techniques for characterizing rhythm and beat extraction in videos. In this paper, we would like to handle video rhythm adjustment from the video analogies point of view.

Usual home videos often do not possess a proper aesthetic rhythm. In order to emphasize special scenes, actions, characters and atmosphere in a video, rhythm adjustment can be extremely useful. It can thus be used for enhancing home videos also.

Video rhythm is a global feature built on shot segments. The mapping between the source and target video below frame level (frame mapping and block mapping) is not required. Thus, in the case, A is similar to A' but with additional meta-data in the form of associated clip lengths (to represent the rhythm). The method is described as follows.

In order to perform video rhythm adjustment or transfer, we take video shots into account. Given a source video A' and a target video B , we segment them into several shots based on the events which are subject to our needs and obtain the clip lengths (total frame number). Suppose A' has n shots, their frame numbers are S_1, S_2, \dots, S_n respectively; B has m shots, their frame numbers are s_1, s_2, \dots, s_m respectively.

We can now obtain A which has the clip proportion information $S_1 : S_2 : \dots : S_n$. From this ratio, we can determine the relative duration or frame numbers of the video shots. We use it to modify the target video by keeping the ratio invariant. Namely:

$$L_i = \frac{L_{i-1} \cdot S_i}{S_{i-1}}, i = 2, 3, \dots, m; \quad (20)$$

where L_i is the new length of video shots, and L_1 can be fixed based on the requirements, for instance, $L_1 = s_1$. After obtaining the mapping, next we determine the rhythm and transfer it to the target video.

In order to automatically detect the rhythm changes in a video shot, we take the video motion into account. We subtract two adjacent frames in the shot. i.e. $\Delta_i = |Fr_j - Fr_{j-1}|$, Fr_j is j -th video frame, $2 \leq j \leq S_i$. If the motion in a clip is acute, the difference will be significant, or else the distinction is minor. Thus we can find these minor differences by calculating the density of minor difference in this portion. i.e. $\Omega(\Delta_j) = \frac{\sum_{T > \Delta_i}}{\sum_{j < S_i}}$, where \sum denotes as the sum of frame numbers under a condition, If a density is the highest one among all segmentations of this clip, we think the rhythm in this portion is slow.

In practice, we need to add frames or drop frames from the shots of target video according to Eq.(21):

$$d_i = L_i - s_i, i = 1, 2, \dots, m \quad (21)$$

If $d_i > 0$, then we need add d_i frames in the i -th clip. New frames can be created by frame interpolation or replication. If $d_i < 0$, then we need drop d_i frames from the i -th clip. We add or drop the j -th video frame according to Eq.(22).

$$l_{i,j} = \lfloor \frac{j \cdot L_i}{s_i} \rfloor, j = 1, 2, \dots, s_i \quad (22)$$

where function $\lfloor \cdot \rfloor$ is the floor function.

The procedure to drop frames from a clip is rather easy, we only need to subtract the frames according to a uniform step; however the procedure to add frames for a clip is difficult, requiring interpolation and synthesis. In our implementation, we just replicate the adjacent frames to expand the sequence which is sufficient in many cases. This is an uniform replication with the equal interleaving. After this replication, the tempo is changed, which will greatly influence the overall emphasis of this video. Viewers can sense a changed tempo to that of the exemplar source video. Our algorithm can now be described as following algorithm 5.

Input : A source video A' and a target video B

Output : The new target video B

Procedure:

1. Segment the source video A' into several shots according to its events and rhythm. This rhythm feature labeling produces A ;
2. For automatic detection of rhythm in a video clip, calculate the density of minor difference of video frames. Determine the length proportion of the two shots A and B ;
3. Calculate the new proportions of the target video by using video analogies according to Eq.(20);
4. Calculate the new frames of the target video using Eq.(21), to add frames, replication/interpolation of adjacent frames is used, to drop video frames, frames at uniform intervals are subtracted from the current sequence;
5. Output the new target video B .

Algorithm 5: Video rhythm adjustment

Essentially, our algorithm for video rhythm adjustment is for frame number adjustment in video shots; however with the changing of video clip lengths, we can get a surprisingly powerful effect.

V. EXPERIMENTS

In this section, we provide results of our experiments. Since it is difficult to accurately convey the quality of results of video experiments on paper, the reader is encouraged to view the results on our website [<http://www.comp.nus.edu.sg/~yanwq/ACMMMMSYS/VideoAnalogies/acmmmjournal.htm>].

A. Colorizing Videos

In this section, we provide the experiment about colorizing by using texture mapping. We colorize videos by using video analogies. We transform the target video into grayscale first and then render the grayscale video by finding the best matched frames in another color video. Thus, we transfer the colors to the grayscale frames.

Figure 10, Fig. 11, and Fig. 12 provide three groups of results, (a) is a source video, (b) is a grayscale video, (c) is the colorized video and (d) is the comparison result between grayscale video and colorized video, we use chroma to find the chroma degree. Chroma is calculated as following Eq.(23):

$$c = \sqrt{C_r^2 + C_g^2} \quad (23)$$

where $C_r = -0.1687 \cdot r - 0.3313 \cdot g + 0.5 \cdot b$; $C_g = 0.5 \cdot r - 0.4187 \cdot g - 0.0813 \cdot b$; (r, g, b) is the pixel color coordinates. Chroma fundamentally reflects the colorizing degree of the video. The chroma degree of a grayscale video is equal to zero.

Figure 10 is a video about jellyfish shot at the Underwater World in Singapore. Figure 11 is a video shot from the balcony of an apartment. Figure 12 is a video of the corridor in an office building. The experiment demonstrates that the grayscale video can be colorized. Furthermore, if a video without color (e.g. infrared video) can be transformed to a proper grayscale video, we can colorize the grayscale video.

B. Video Blur Detection and Reduction

Figure 13 shows the results of blur detection. The two thresholds for edge detection and blur degree are taken as 128 and 1.5 respectively in the algorithm for blur detection. The results show that our proposed method can detect both motion blur and Gaussian blur robustly. Figure 13(a) is the plot of the blur degree and the frames of the video. Figure 13(b) is the blur degree curve and the video frames of the video. In Fig. 13(a), the frames whose blur degree is less than 1.5 are Gaussian blur frames, we see the frames around 40 or so are the blurry frames. In Fig. 13(b), the frames from 0 ~ 20 are all blurry frames.

Figure 14 and Fig. 15 show the results from two groups of experiments about video analogies based blur reduction, (a) is the blurry video, (b) is the source video, (c) is the improved video, (d) is the difference between blur video and the video with blur reduction. We use the blur degree to measure the effectiveness.

Figure 14 is a video about corridor in NUS while Fig. 15 is a campus car park. It is obvious from the plots that the target videos have become significantly sharper after the transfer.

C. Video Rhythm Adjustment

Figure 16 and Fig. 17 depict one group for video rhythm adjustment. Given a video clip from a movie (a), we map the proportion of clip length to the target video (b), we get an exciting video clip (c). Figure 16 is a video about Michael Jackson's dance performance to which we transfer the rhythm of a piece of clip of the movie "Hero". This transfer emphasizes the dancing skills of Michael Jackson. Figure 17 is the video about creaming of Bill Gates, to which we transfer the rhythm of one clip of the movie of Jackie Chan (Flying Motorcycle) and obtain a new clip that accentuates the creaming effect.

VI. CONCLUSION

The paper presents a methodology that exploits feature correlation of segments of different videos for automatic video colorization, blur reduction and rhythm adjustment. This three-step solution consisting

of transformation, matching & transfer phases has many potential applications. We colorize videos by using video analogies based color transfer. For blur detection, we use the Sobel edge operator to detect the blurring degree while for blur reduction, we use a video analogies based Bezier curve histogram adjustment technique. We transfer the rhythm of video shots by using video analogies based frame number adjustment. The advantage of video analogies based handling is that we are able to provide an “ideal” for video editing which is the source video. We emulate the desired trait of this ideal video. Thus we are able to transfer features like the rhythm of professional movies and sitcoms to any video. This greatly helps bridge the gap between the low quality videos and professional videos. We strongly believe that the video analogies technique has many other applications which we plan to work on in the future:

- Lighting transfer: The idea is to use the lighting of a professional video in order to improve the lighting of a video with lighting artifacts.
- Motion transfer: Shaking artifacts occur frequently in videos which need to be removed. The video analogies technique provides an alternative to motion trajectory smoothing. Also, motion transfer can be used as a basis for music matching.

Our future work in general will focus on automatically and efficiently editing videos by using the video analogies based techniques.

ACKNOWLEDGEMENT

Wei-Qi Yan’s work in this paper is sponsored by a fellowship from the Singapore Millennium Foundation (SMF). We deeply appreciate the numerous constructive suggestions from the anonymous reviewers.

REFERENCES

- [1] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, “Color transfer between images,” *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34–41, 2001.
- [2] T. Welsh, M. Ashikhmin, and K. Mueller, “Transferring color to grayscale images,” in *Proc. of ACM SIGGRAPH’02*, vol. 20, no. 3, 2002, pp. 277–280.
- [3] W.-Q. Yan and M. S. Kankanhalli, “Colorizing infrared home videos,” in *Proc. of IEEE ICME’03*, Baltimore, USA, Jul. 2003.
- [4] —, “Erasing video logos based on image inpainting,” in *Proc. of IEEE ICME’02*, Lausanne, Switzerland, Aug. 2002, pp. 521–524.
- [5] P. Mulhem, M. S. Kankanhalli, H. Hasan, and Y. Ji, “Pivot vector space approach for audio-video mixing,” *IEEE Multimedia*, pp. 28–40, Apr. 2003.
- [6] S. Chang and H. Sundaram, “Structural and semantic analysis of video,” in *Proc. of IEEE ICME’00*, vol. 2, New York, USA, Jul. 2000, pp. 687–690.

- [7] J. Kender and B. Teo, "Video scene segmentation via continuous video coherence," in *Proc. of IEEE CVPR'98*, Santa Barbara, USA, Jun. 1998.
- [8] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin, "Image analogies," in *Proc. of ACM SIGGRAPH'01*, 2001.
- [9] T. Evans, *Semantic Information Processing*. MIT Press, 1968, A program for the solution of geometric analogy intelligence test questions.
- [10] A. Hertzmann, N. Oliver, S. Seitz, and B. Curless, "Curves analogies," in *Proc. of Eurographics Workshop on Rendering'02*, 2002.
- [11] D. Gentner, "Structure mapping: a theoretical framework for analogy," *Cognitive Science*, vol. 7, no. 2, pp. 155–170, 1983.
- [12] P. Winston, "Learning and reasoning by analogy," *Communication of ACM*, vol. 23, no. 12, Dec. 1980.
- [13] J. Casares, B. A. Myers, A. C. Long, R. Bhatnagar, S. M. Stevens, L. Dabbish, and A. Corbett, "Simplifying video editing with intelligent interaction." [Online]. Available: <http://www-2.cs.cmu.edu/silver/publications.html>
- [14] A. Girgensohn, S. Bly, F. Shipman, J. Boreczky, and L. Wilcox, "Home video editing made easy-balancing automation and user control," in *Proc. of INTERACT'01*. IOS Press, 2001, pp. 464–471.
- [15] A. Girgensohn, J. Boreczky, P. Chiu, J. Doherty, J. Foote, G. Golovchinsky, S. Uchihashi, and L. Wilcox, "A semi-automatic approach to home video editing," in *CHI letters*, vol. 2, Feb. 2000, pp. 81–89.
- [16] M. Davis, "Editing out video editing," *IEEE multimedia*, vol. 10, no. 2, pp. 54–64, Apr.-Jun. 2003.
- [17] C. Dorai and S. Venkatesh, "Computational media aesthetics: finding meaning beautiful," *IEEE multimedia*, vol. 8, no. 4, pp. 10–12, Oct. 2001.
- [18] —, "Bridging the semantic gap with computational media aesthetics," *IEEE Multimedia*, vol. 10, no. 2, pp. 15–17, Apr. 2003.
- [19] M. Tekalp, *Digital video processing*. New York: Prentice Hall, 1995.
- [20] P. Sand and S. Teller, "Video matching," *ACM Transactions on Graphics*, 2004.
- [21] M. Chiang and T. Boulton, "Local blur estimation and super-resolution," in *Proc. of IEEE CVPR'97*, Jun. 1997.
- [22] W.-Y. Ma and H.-J. Zhang, "Blur determination in the compressed domain using DCT information," in *Proc. of IEEE ICIP'99*, vol. 2, Kobe, Japan, 1999, pp. 386–390.
- [23] R. Gonzalez and R. Woods, *Digital image processing*. Addison-Wesley, 1992-1993.
- [24] X.-D. Sun, A. Divakaran, and B. S. Manjunath, "A motion activity descriptor and its extraction in compressed domain," in *Proc. of IEEE PCM'01*, Beijing, China, Oct. 2001, pp. 450–457.
- [25] A. Pattiand, M. Sezan, and A. Tekalp, "Super resolution video reconstruction with arbitrary sampling lattices and nonzero aperture time," *IEEE transactions on image processing*, vol. 6, no. 8, pp. 1067–1076, Aug. 1997.
- [26] N. Nguyen and P. Milanfar, "A computationally efficient super resolution image reconstruction algorithm," *IEEE transactions on image processing*, vol. 10, no. 4, pp. 573–583, 2001.
- [27] T. William, R. Touis, and C. Egon, "Example-based super-resolution," *IEEE computer graphics and application*, pp. 56–65, Mar./Apr. 2002.
- [28] W.-Q. Yan and M. S. Kankanhalli, "Detection and removal of lighting and shaking artifacts in home videos," in *Proc. of ACM Multimedia'02*, Juan Les Pins, France, Dec. 2002.
- [29] G. Farin, *Curves and surfaces for computer aided geometric design: a practical guide (3rd Edition)*. Imprint Boston: Academic Press, 1992.
- [30] B. Adams, C. Dorai, and S. Venkatesh, "Formulating film tempo - the computational media aesthetics methodology in practice," in *Media computing: computational media aesthetics*. Kluwer Academic Publishers, 2002, pp. 57–79.

- [31] B. Adams, S. Venkatesh, and C. Dorai, "Finding the beat: an analysis of the rhythmic elements of motion pictures," *International Journal of Image and Graphics*, vol. 2, no. 2, pp. 215–245, 2002.
- [32] B. Adams, C. Dorai, and S. Venkatesh, "Automated film rhythm extraction for scene analysis," in *Proc. of IEEE ICME'01*, Tokyo, Japan, Aug. 2001, pp. 1192–1195.
- [33] Z. Herbert, *Sight, sound, motion: applied media aesthetics (the 3rd edition)*. Wadsworth Publishing Company, 1999.

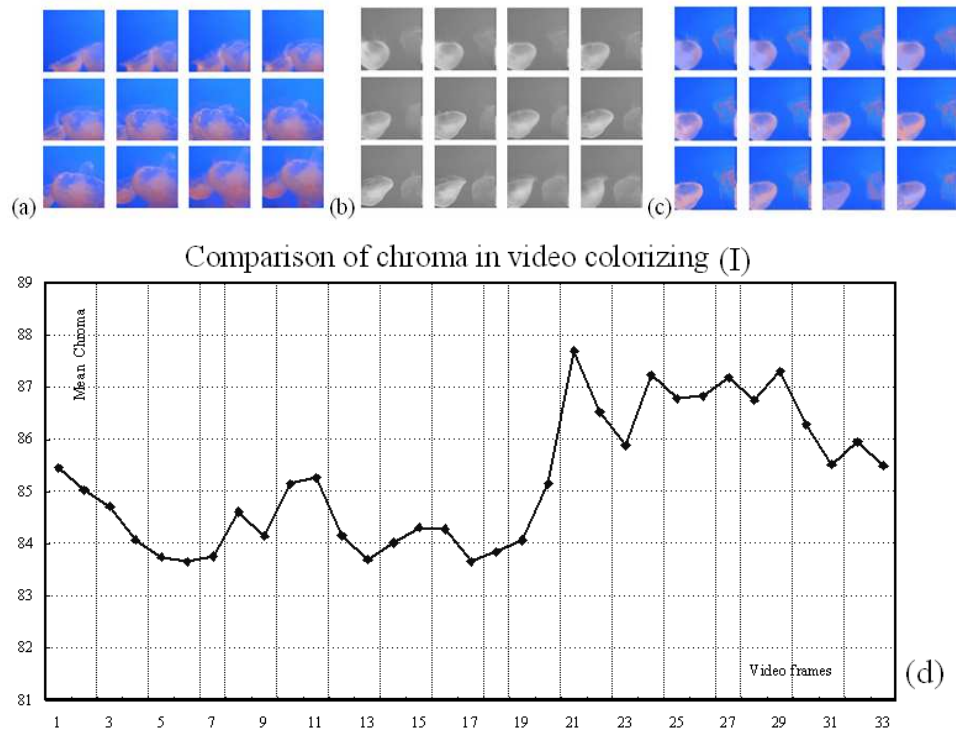


Fig. 10. Colorizing grayscale video (Group 1)

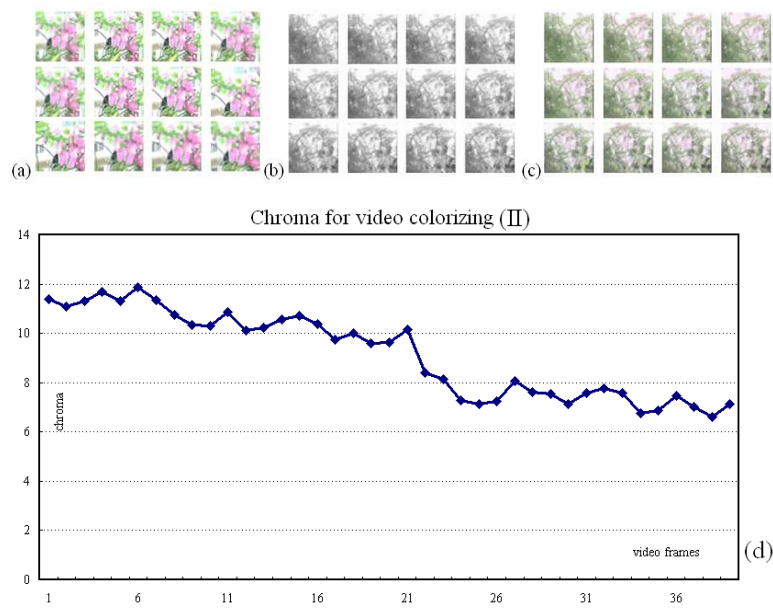


Fig. 11. Colorizing grayscale video (Group 2)

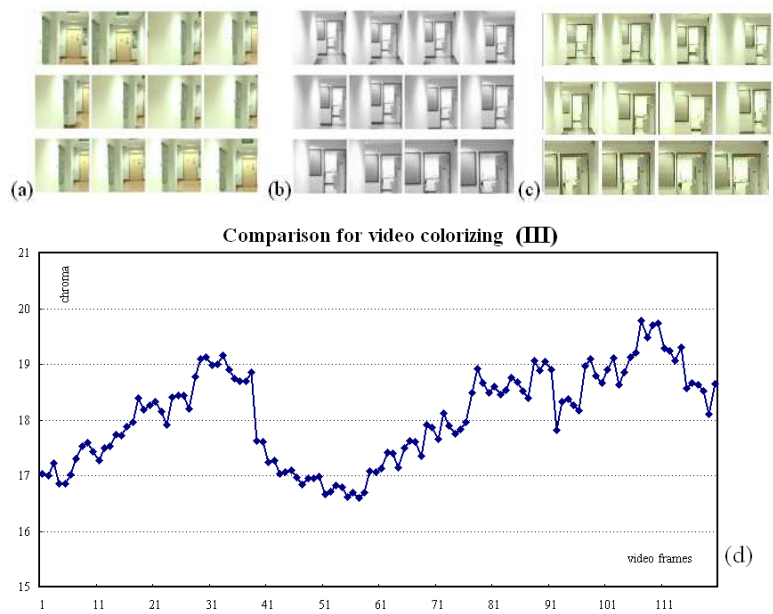
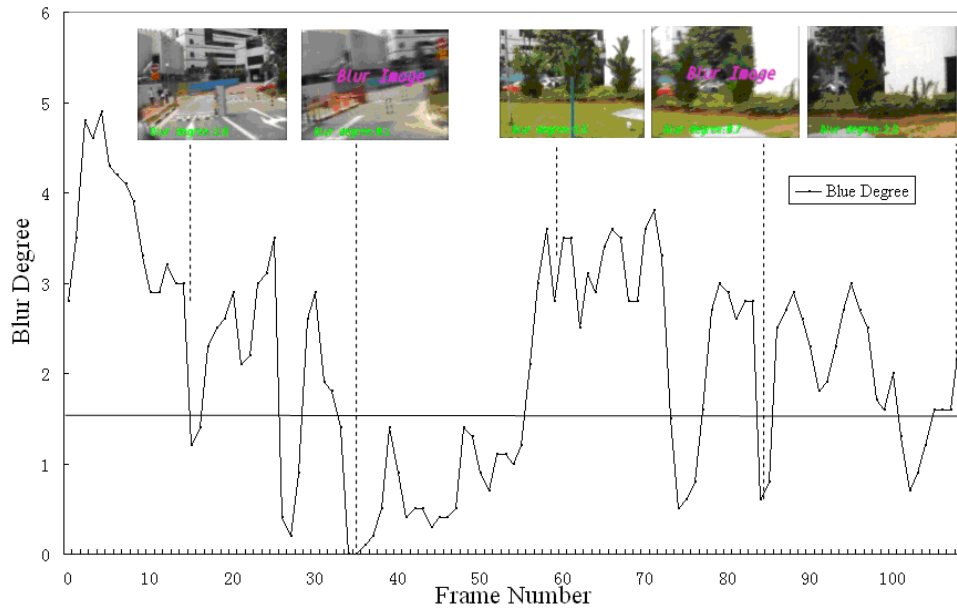
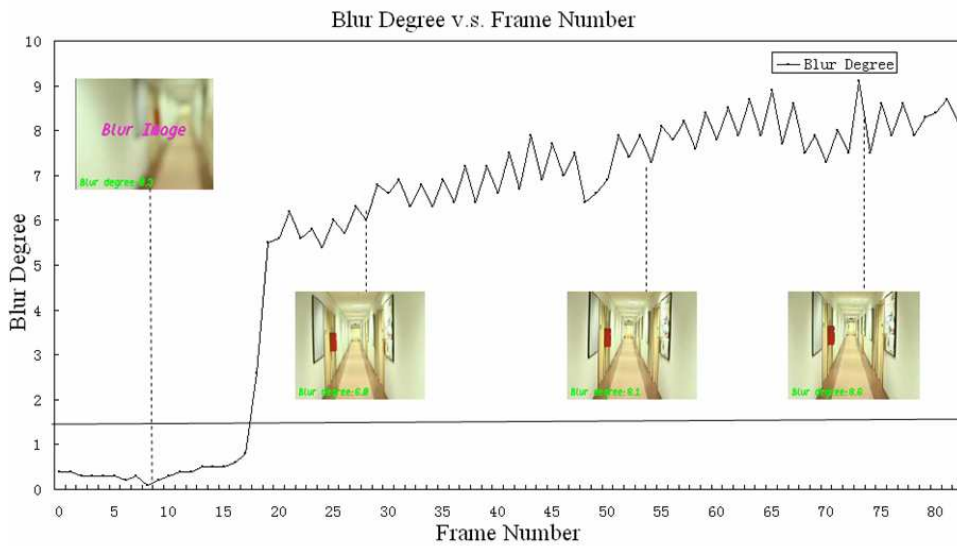


Fig. 12. Colorizing grayscale video (Group 3)



(a)



(b)

Fig. 13. Blurry frames detection

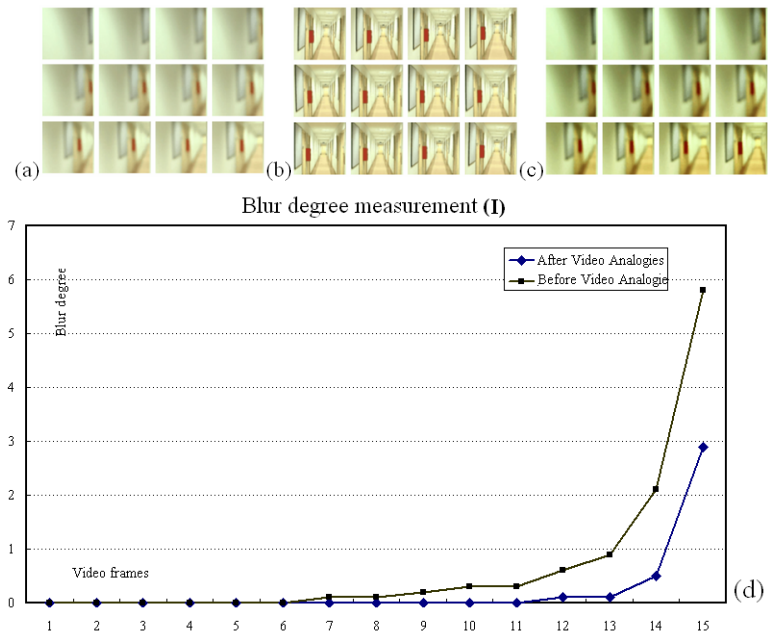


Fig. 14. Video blur reduction (Group 1)

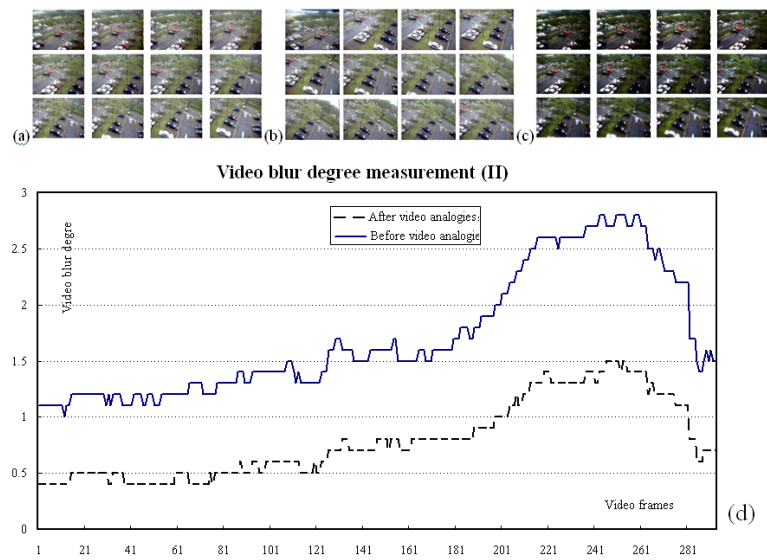


Fig. 15. Video blur reduction (Group 2)

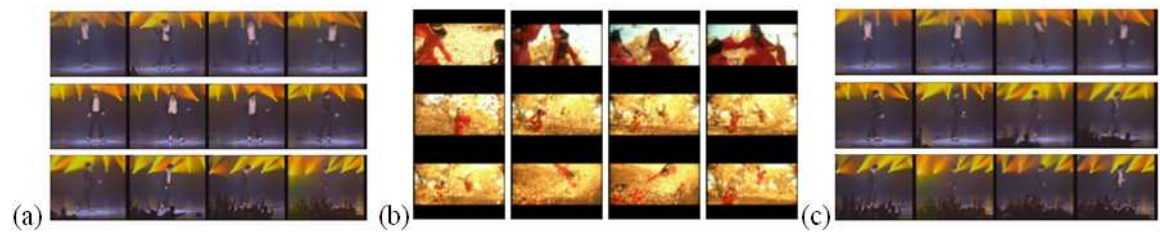


Fig. 16. Video rhythm transfer (Group 1).

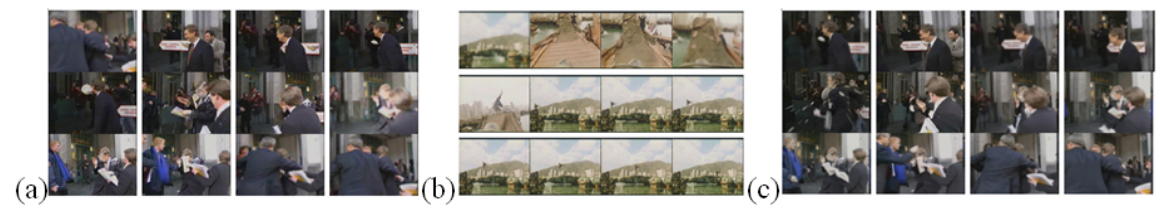


Fig. 17. Video rhythm transfer (Group 2).