Compressed-Domain Video Retargeting

Jiangyang Zhang, *Student Member, IEEE*, Shangwen Li, *Student Member, IEEE*, and C.-C. Jay Kuo, *Fellow, IEEE*

***Abstract*— In this paper, we present a compressed-domain video retargeting solution that operates without compromising the resizing quality. Existing video retargeting methods operate in the spatial (or pixel) domain. Such a solution is not practical if it is implemented in mobile devices due to its large memory requirement. In the proposed solution, each component of the retargeting system is designed to exploit the low-level compressed domain features extracted from the coded bit stream. For example, motion information is obtained directly from motion vectors. An efficient column shape mesh deformation is employed to solve the difficulty of sophisticated quad-shape mesh deformation in the compressed domain. The proposed solution achieves comparable (or slightly better) visual quality performance as compared with several state-of-the-art pixel-domain retargeting methods at lower computational and memory costs, making content-aware video resizing both scalable and practical in real-world applications.**

***Index Terms*— Video retargeting, compressed domain processing, content-aware, cropping, column mesh, warping.**

I. Introduction

THE INCREASING demand to display video contents on devices with different resolutions and aspect ratios calls for new solutions to video resizing. Traditional resizing techniques are incapable of meeting this requirement as they either discard important information (e.g. cropping) or introduce visual artifacts by over-squeezing the content (e.g. homogeneous rescaling). The goal of content-aware video resizing (or video retargeting) is to change the aspect ratio and resolution of videos while preserving the visually important content and avoiding noticeable artifacts. Recently, many pixel-domain solutions have been proposed for video retargeting, which conducts resizing either through non-uniform warping [1]–[4] or iteratively removing unimportant contents [5], [6]. Despite their promising results and real-time performance in the spatial domain [2], [4], such solutions are still impractical as they only operate on raw video data. Since most real-world video contents are stored and transmitted only in compressed format, spatial domain retargeting techniques are inevitably encapsulated by additional overheads of decompression and recompression.

In this work, we attempt to address this issue with a novel framework that performs content-aware resizing directly on the compressed video bitstream. Processing video contents directly in the compressed domain has many advantages in terms of speed, storage efficiency and quality. First, the computational time is significantly reduced as a few time-consuming modules (such as motion estimation) can be effectively avoided. Second, the data rate is highly reduced in the compressed domain, leading to significant memory saving. Lastly, extra video quality degradation can be avoided as the quantization and transform steps are not performed in the final re-encoding stage.

One of the main contributions of this work is the formulation of the retargeting problem using compressed domain features and operations. Performing video retargeting in the compressed domain is fundamentally different from that in the pixel domain, since the former is limited by many constraints. For example, all pixel-level information (*e.g.*, color, gradient, saliency), which has been extensively used for spatial-domain retargeting, is unavailable in the compressed domain. Instead, what we have is the block-level Discrete Cosine Transform (DCT) coefficients, which are not directly correlated with pixel-level features. Thus, this constraint demands us to take a very different path to the solution. Specifically, our solution computes the visual importance map using compressed-domain features obtained from the compressed video bitstream. The motion information is obtained directly from motion vectors in the coded file, and there is no need to perform expensive computation to extract this information again (*e.g.*, optical flow as conducted in previous spatial-domain methods [2]–[4]). In addition, existing warping-based approaches [2]–[4], [7] often adopt quad-shape mesh deformation. However, such geometrical modification is difficult to perform in the compressed domain. To overcome this limitation, we develop a novel column mesh deformation that is compatible with compressed domain operations without compromising the quality of resizing results.

After performing video retargeting, due to the change in the frame size and the aspect ratio, motion vectors and prediction modes of the output macroblocks have to be recomputed. Another main contribution of this work is the development of new mode decision and motion estimation modules in the re-encoding stage to allow computational saving. By exploiting block correspondences before and after retargeting, we provide a fast yet effective method to estimate motion vectors and prediction modes of the retargeted video. This saves efforts in going through the time-consuming mode decision and motion estimation procedures of a standard video encoder. The proposed fast solution can achieve a speed up factor of 90 (compared with full motion search) and 30 (compared with fast motion search) in the encoding stage. For visual quality performance evaluation, we report a subjective user test consisting of 56 subjects that compare our retargeting results

Manuscript received April 5, 2013; revised September 9, 2013 and November 19, 2013; accepted December 4, 2013. Date of publication December 11, 2013; date of current version January 9, 2014. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Charles Creusere.

The authors are with the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: jiangyaz@usc.edu; shangwel@usc.edu; cckuo@sipi.usc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2013.2294541

1057-7149 © 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

with those of state-of-the-art spatial-domain video retargeting methods. Although the exemplary video coding standard used in this work is H.264/AVC, the proposed techniques can be applied to other video coding standards in a similar fashion.

The rest of this paper is organized as follows. Related previous work is reviewed in Section II. The system overview is presented in Section III. The three key components of the system; namely, partial decoding, compressed domain re-sizing and re-encoding, are detailed in Sections IV, V-E and VI, respectively. Experimental results are shown in Section VII. Finally, concluding remarks are given in Section VIII.

II. Related Work

A. Image Retargeting

Content-aware image resizing techniques can be generally classified into two categories: the discrete and the continuous approaches [8]. For the discrete approach, content-aware resizing of an image is achieved through identifying and removing unimportant image contents. The cropping-based method [9] identifies the most prominent components in an image through saliency-based measures and cuts out a rectangular region as the retargeting result. The seam carving method [10] resizes an image through continuously removing paths of pixels with the least amount of energy. Realizing that no single retargeting operator could perform well on all images, the multi-operator method [11] that combines three different operators (namely, scaling, cropping and seam carving) was proposed. For the continuous approach, the retargeting problem [1], [12] is formulated as nonlinear warping in which shapes of salient regions should be well preserved while those of non-salient regions are allowed to be squeezed or stretched.

Homogeneous image scaling and cropping in the compressed domain has been studied in literature [13], [14]. By exploring the distributive property of the unitary orthogonal transform, image resizing methods that achieve DCT domain down-sizing/up-scaling by a factor of two were proposed in [13]. In [14], transform domain image resizing is further extended by allowing resizing with an arbitrary ratio. Most recently, content-aware image resizing in the compressed domain has been studied. For example, Fang *et al.* [15] proposed a JPEG image retargeting scheme guided by a saliency map computed using DCT coefficients. Although compressed domain features are used in this method, the seam removal procedure still requires full decoding of the JPEG image. In addition, as the overhead of compression and decompression on images is much lower than videos, the advantage of retargeting in the compressed domain is less obvious.

B. Video Retargeting

Video retargeting is different from image retargeting as temporal coherency and object motions are additional factors to consider. Most video retargeting methods extend image-based retargeting methods by adding additional constraints that enforce temporal-adjacent regions to undergo similar transforms [1], [2], [5]. Based on the seam carving image retargeting method [10], video retargeting is formulated as a graph-cut problem in a three-dimensional spatial-temporal cube [5]. In [6], the removed seams are allowed to be unconnected, and it was shown that discontinuous seams outperform continuous ones under certain scenarios. In [3], cropping is further introduced into the video retargeting framework as it could outperform warping especially when the video is over-populated with visually important contents. The issue of scalability is addressed in [4], which optimizes saliency-based resizing and temporal coherency separately to achieve reduction in both computational and memory requirements.

Most recently, a video retargeting solution that utilizes compressed-domain features was proposed in [16]. Based on the spatial-domain seam carving approach [10], this method uses compressed-domain features (e.g. DCT coefficients, motion vectors) to compute the optimum seams for removal. However, this method does not work fully in the compressed-domain, as the resizing step still operates in the pixel-domain and requires full-decoding of each video frame.

To the best of our knowledge, there is no existing solution to video retargeting in the compressed-domain. Our work provides a novel and practical solution for compressed domain video retargeting that is applicable to the great majority of today's video.

III. System Overview

The proposed system takes the H.264/AVC encoded video bitstream as the input, conducts retargeting directly on partially decoded DCT coefficients, and outputs the H.264/AVC-compliant bitstream of the retargeted video. The proposed system consists of three separate stages (or modules) as shown in Fig. 1. They are:

1) the partial decoding stage;
2) the compressed domain video resizing stage; and
3) the re-encoding stage.

In the partial decoding stage, we decode the video bitstream partially to reconstruct the non-inverse-quantized DCT coefficients of each frame. In the resizing stage, we perform video analysis and content-aware resizing directly based on compressed domain features and operations. The output of the resized image is re-encoded to be the output bitstream in the last stage. Details of each module will be described in the following sections.

IV. Partial Decoding

In this stage, we partially decode the input bitstream to the proper form of compressed data such that it can be further utilized in the resizing stage. Although we try to limit the amount of operations conducted in this stage, a certain amount of decoding is still required. The input bitstream is first entropy decoded into residual DCT blocks, which are then used to compute reconstructed DCT blocks of the original video frame. Since our system operates directly in the DCT domain, no inverse DCT is required as well. To further reduce the overhead of the decoding stage, we avoid the inverse quantization step as well. Therefore, the output

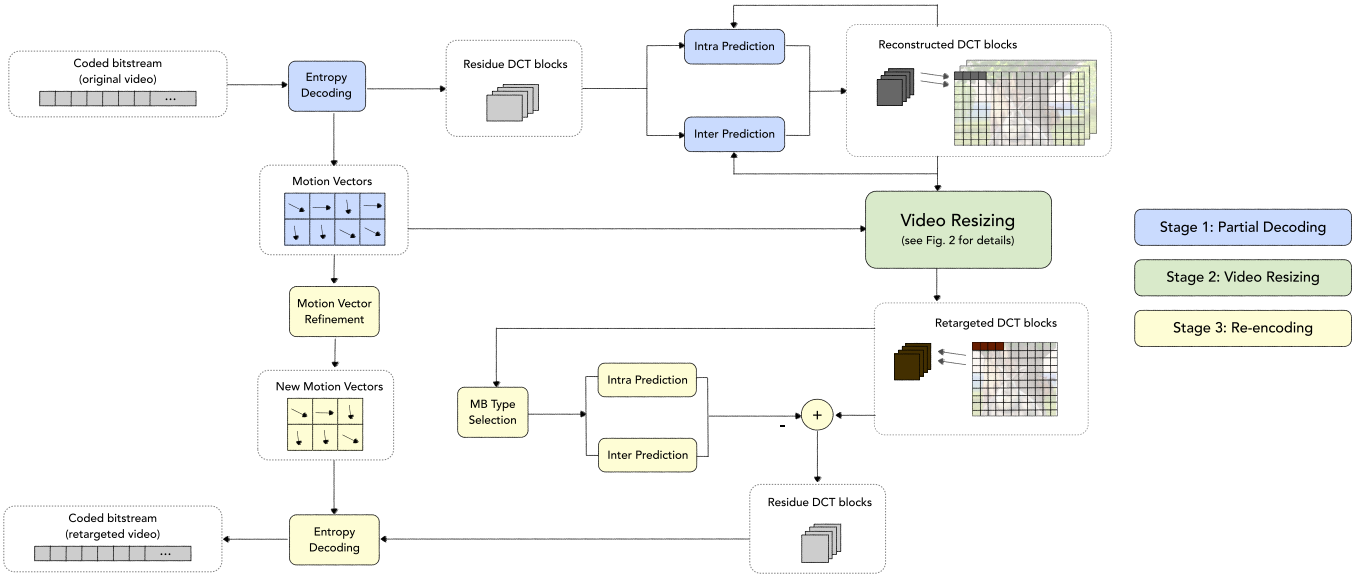


Fig. 1. The block diagram of the proposed compressed-domain video retargeting system that consists of three stages (or modules): 1) the partial decoding stage, 2) the compressed domain video resizing stage, and 3) the re-encoding stage.

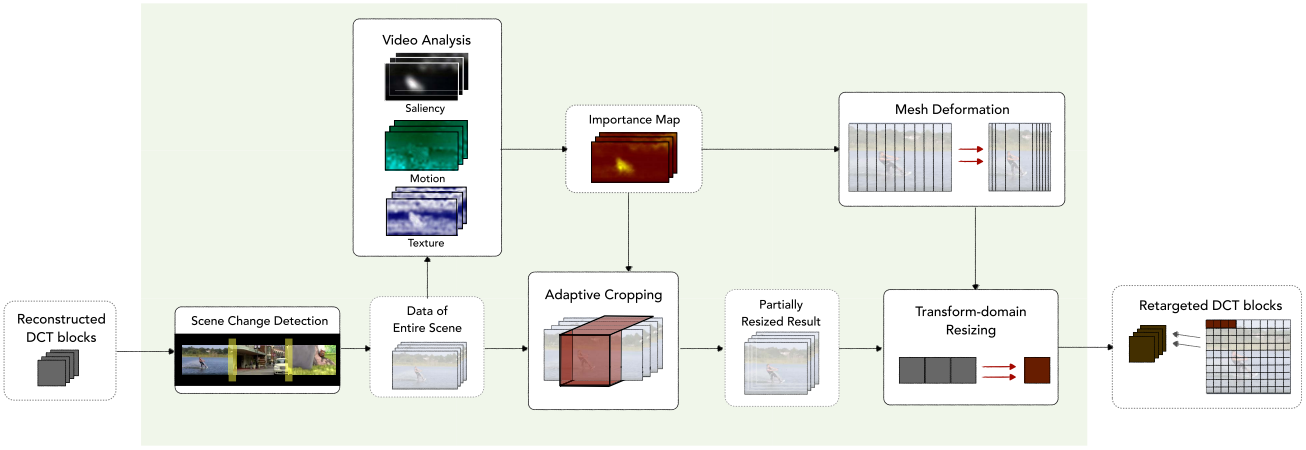


Fig. 2. The block-diagram of the compressed-domain video resizing stage. This module takes reconstructed DCT coefficients as input and outputs DCT coefficients of the retargeted video.

of the partial decoding stage are non-inverse-quantized DCT block coefficients.

To compute reconstructed DCT coefficients from the residual data, we make use of the transform domain prediction techniques proposed in [17], [18]. As the H.264/AVC standard supports both intra and inter prediction modes, two types of prediction are conducted here. For the inter-prediction mode, we use the macroblockwise inverse motion compensation (MBIMC) scheme proposed by Porwal *et al.* [17]. In this scheme, the predicted DCT block of the current frame is estimated using DCT coefficients of nine spatially-adjacent blocks in the previous frame. Although originally proposed for the MPEG standard, this method can be easily applied to the inter prediction mode of H.264/AVC as well. For the intra-prediction case, different situations have to be considered as H.264/AVC supports nine intra-prediction modes for 4×4 sub-blocks and four intra-prediction modes for 16×16 macroblocks. Here, we use the method in [18] to compute the DCT coefficients of intra-predicted blocks. By combining the intra/inter-predicted DCT coefficients and the partially decoded residual DCT coefficients, we obtain reconstructed DCT coefficients, which will be utilized in the resizing stage.

Another important task to be performed in this stage is extracting motion vectors from the input bitstream. The motion vectors are temporarily stored, and they will be processed and utilized in both the resizing and the re-encoding stages.

V. Compressed Domain Video Resizing

In this stage, we perform content-aware resizing using compressed-domain features and operations with multiple steps. The input video is first segmented into different scenes and each scene is processed separately. Then, we analyze the importance of each scene using three different measures: saliency, motion and texture. Guided by the importance map, the input video is partially resized through optimum cropping, followed by the column-mesh-based warping procedure to reach its desired size. Finally, we compute DCT coefficients of the retargeted result. The block-diagram of this procedure is depicted in Fig. 2. All steps will be detailed below.

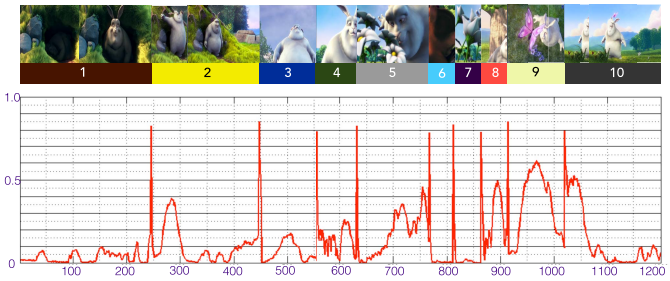


Fig. 3. Top: the manual scene segmentation result for a 1200-frame segment of the *big buck bunny* sequence. Bottom: the percentage of block change T_c of each frame. The sharp peaks in the T_c curve closely match the manual segmentation result.

A. Scene Change Detection

We use a pair-wise macroblock comparison method to detect scene changes between consecutive frames. For each video frame, we compare the DCT coefficients of each block with the average coefficient values of the same block in previous frames. The content difference for block k is computed as

$$\eta_k = \frac{1}{N^2} \sum_{i=1}^{N \times N} \frac{\left| c_k(i) - \bar{c}_k^T(i) \right|}{\mathbf{max}\{c_k(i), \bar{c}_k^T(i)\}}$$

where N is the size of block, $c_k(i)$ is the DCT coefficient of block k, and $\bar{c}_k^T(i)$ is the average DCT coefficients of block k of the previous T frames (in our case, $T = 5$).

If this difference exceeds a preset threshold, we claim that there is a change for block k. We use D_k to denote this change:

$$D_k = \begin{cases} 1 & \eta_k > \eta_{thre} \\ 0 & otherwise \end{cases}$$

Let K be the total number of blocks in a frame. For all blocks in the current frame, when the percentage of changed blocks exceeds a preset threshold τ; namely, $T_c = \frac{\sum_{k=1}^{K} D_k}{K} > \tau$, a scene change is detected, and the entire data of this scene will be loaded for further processing. Fig. 3 illustrates the scene change detection results for a segment of 1200 frames of the *big buck bunny* sequence. The sharp peaks in the T_c curve indicates the occurrences of scene changes.

B. Visual Importance Analysis

The visual importance map is used to guide the retargeting process, since we would like to preserve the important content as much as possible while allowing the unimportant content undergo more deformation in an ideal retargeted result. Once a scene change is detected, data of the entire scene would be processed together for visual importance analysis. The challenge of conducting analysis in the compressed-domain is that the original pixel values are unknown and we need to rely on compressed-domain features (*e.g.*, DC, AC coefficients, motion vector, etc.) only. In this work, we perform content analysis based on three features: saliency, texture and motion. Each analysis generates one single map, which will eventually be combined to form the final visual importance map.

1) Saliency Map: Saliency is used to detect the region of interest in images, and has been widely used to guide both image and video retargeting [2]–[4], [7], [12]. While most saliency detection methods operate in the pixel-domain [19], [20], there is recent work on compressed-domain saliency detection for images [15] and videos [21].

In the proposed system, we adopt the spectral-residual visual attention model [19] for saliency map computing. In [19], the saliency map of an image is calculated using the spectral residual signal, derived from analyzing the log spectrum of the image. Although this saliency detection method operates in the pixel-domain, we can modify it so that it can be used in the DCT domain as well. Instead of downsampling the input image to a smaller size as done in [19], we directly use the DC coefficient of each DCT block and apply the same saliency detection algorithm to this DC-based image. For the 4×4 DCT transform, the DC coefficients of the entire image yield an equivalent image obtained by downsampling the original image to its original size by a factor of 4×4. For improved temporal coherency, the visual saliency map is temporally filtered with its neighboring T frames (in our system, $T = 5$).

2) Texture Map: Fine structures, such as textures and edges, need special treatment in video retargeting. One of the limitations of seam carving [5] is that, when the removed seams pass through edge regions, noticeable artifacts would occur. In addition, the effect of texture regularity on the retargeting result was studied in [22], and it is observed that stochastic textures are less susceptible to large deformation than regular textures.

The saliency map generated using [19] contains limited amount of texture information, as only DC coefficients were used for computation. Extracting textures and edges normally require pixel-level processing, yet it is possible to obtain some level of texture and edge information through frequency analysis on DCT coefficients as textures and edges correspond to mid-to-high frequency components in the DCT domain.

In the proposed system, we use different frequency components of DCT coefficients to generate the feature vector for each block. To classify each block into one of the three categories (texture, edge and smooth region), we compute the distance between the feature vector of a given block with a group of preset feature vectors, obtained by training on a set of test sequences from the public database [23]. The likelihood of a block belonging to a particular category ($k = tex, edge, smooth$) is given by:

$$L_k = \frac{e^{\frac{1}{1+d_k}}}{\sum_k e^{\frac{1}{1+d_k}}},$$

where d_k is the Euclidean distance between the feature vector and the preset centroid vector for category k. In this work, we pay special attention to the texture map L_{tex}.

The texture degree of a block also provides a measure on the reliability of motion vectors. It is well known that low-textured regions tend to yield larger encoding matching errors [24]. For each macroblock, we can compute a confidence score for the corresponding motion vector using its texture degree, which will be elaborated in the next section.

3) Motion Map: The saliency map generated by [19] mainly captures the visual attractive parts in an individual frame, but

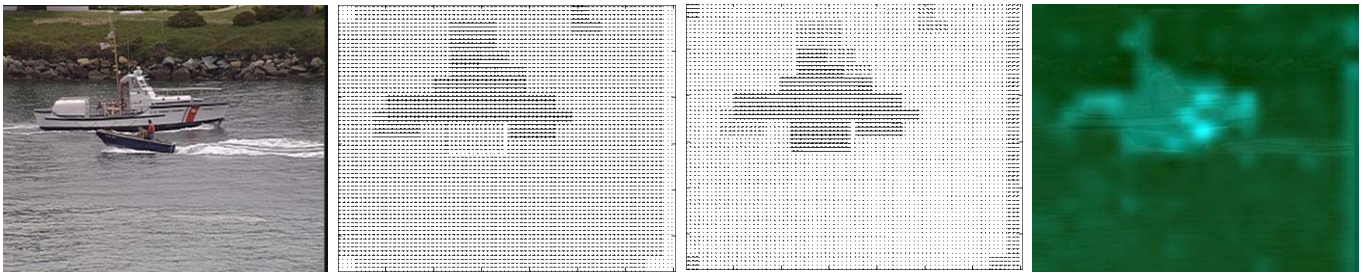


Fig. 4. Illustration of motion map generation using motion vectors from sequence *coastguard*. From left to right: one video frame, the original motion vector map, the compensated object motion, and the final motion map (after applying temporal filtering on the compensated motion map).

fails to consider the motion information, which is another critical factor for important content detection in video. For example, a fast moving object might be non-salient in a single frame, but an important content in a video sequence.

Most spatial-domain video retargeting methods [2]–[4], [7] have incorporated motion detection techniques based on the SIFT feature [25] or optical flow [26]. However, these motion detection methods are not applicable in the compressed domain. Our goal here is to detect moving regions in the video sequence using the motion vectors embedded in the video bitstream. There are two challenges for using motion vectors directly for moving object detection:

1) It is common that the video includes various types of camera motions (*e.g.* zoom, pan, tilt) and they need to be excluded to reflect true object motion.
2) Some motion vectors are unreliable as they do not agree with the true motion.

In the proposed system, the camera motion is estimated using a four-parameter global motion model [27]. In this model, the relationship between pixels of consecutive frames can be written as:

$$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} = \begin{pmatrix} z & r\\ -r & z\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} + \begin{pmatrix}p_r\\ p_d\end{pmatrix}, \tag{1}$$

where x and y are coordinates of the current frame, $\bar{x} = x - mv_x$ and $\bar{y} = y - mv_y$ are coordinates of the previous frame, z, r, p_r and p_d are the four unknown camera parameters representing zoom, rotate, pan right and pan down, respectively.

To estimate the four camera parameters, we can re-write Eq. (1) as an over-determined linear system [27] and compute the least-square estimator of the four camera parameters:

$$\mathbf{X_{LS}} = \begin{bmatrix} z & r & p_r & p_d\end{bmatrix}^T = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}, \tag{2}$$

where $\mathbf{Y}$ is the observation column vector and $\mathbf{H}$ is the spatial location matrix. As done in [24], we weigh the rows of $\mathbf{Y}$ and $\mathbf{H}$ using the confidence measure computed from the texture map so that unreliable motion vectors would have a minimum impact on the camera motion estimation result.

The estimation process given above assumes that object motion does not fit into the camera model in Eq. (1) and becomes outliers in the least square estimation. The estimated object motion is then computed as

$$\mathbf{MV_{obj}} = \begin{bmatrix} \bar{mv}_x(1)\\ \bar{mv}_y(1)\\ \bar{mv}_x(2)\\ \bar{mv}_y(2)\\ \cdots\end{bmatrix} = \mathbf{Y} - \mathbf{H}\cdot\mathbf{X_{LS}},$$

where $\bar{mv}_x(i)$ and $\bar{mv}_y(i)$ are the compensated motion vector components of block i. The final motion map is computed using the magnitude of the compensated motion vector and applying a temporal filter over the neighboring T frames (in our case, $T = 5$).

We show the motion vectors after motion compensation and the final motion map for sequence *coastguard* in Fig. 4. The original motion vectors include both the object motion (ship) and the camera motion (right pan). After motion compensation, we eliminate the camera motion from the original motion vectors, leaving only the object motion. The two camera parameters in Eq. (2), p_r and p_d will be used in the mesh deformation stage as described in Section V-D.

4) Visual Importance Map: For each video frame, the final visual importance map denoted by I is computed by combining all three maps (see Fig. 5) generated from the above analysis:

$$I = I_s \times I_t \times I_m, \tag{3}$$

where I_s, I_t and I_m are the saliency, texture and motion map, respectively. The values of all three maps are normalized to the range of $[0.10, 1.00]$. Although there are other ways (*e.g.*, weighted sum) to fuse all three maps into the final importance map, we find that the multiplication-based fusion generates a more satisfying result.

C. Optimum Cropping

The importance of incorporating cropping into video retargeting was extensively discussed in [3]. When the video sequence is densely populated with salient contents, it is difficult to preserve all salient contents while maintaining temporal coherence, because the retargeting result would be close to uniform scaling in this scenario. Instead of performing nonuniform warping on the entire frame, a better solution is to allow some of non-salient regions to be discarded.

In the proposed system, we partially resize the video through cropping first, and then perform warp-based deformation to resize the video to its desired size. We define the

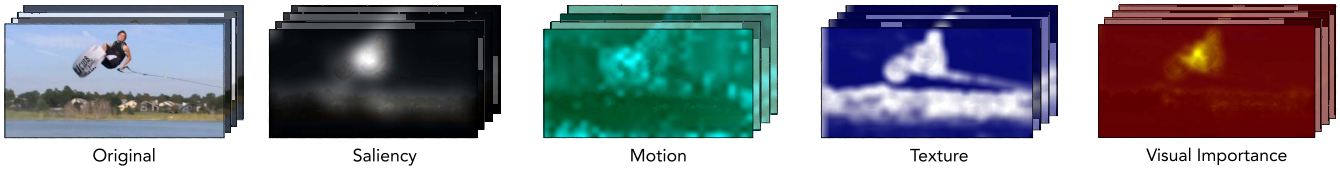


Fig. 5. Illustration of the visual importance analysis procedure: frames of the entire scene are analyzed and three maps (saliency, texture and motion) are generated. Being fused together, they form the final visual importance map.

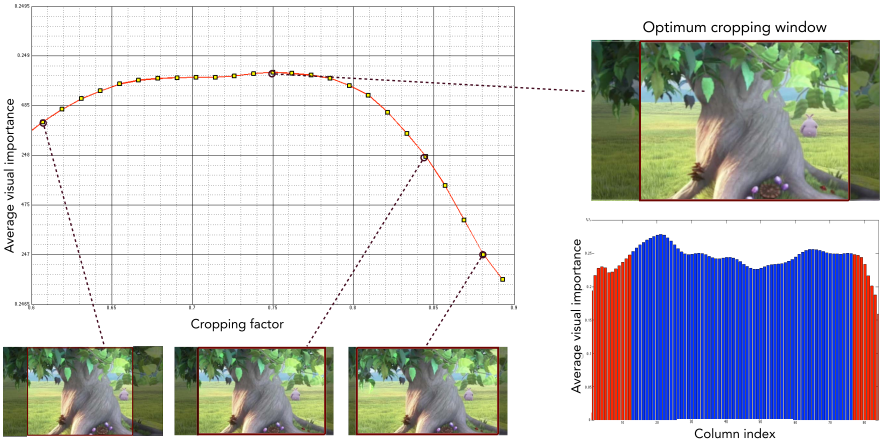


Fig. 6. Left: the average column importance curve for different cropping factors in the cropping range. Each point in this curve corresponds to the best window of a given length. The optimum cropping factor and its corresponding window maximize the average column importance within the cropping window. Right: the optimum cropping window and the average visual importance for each column. Columns marked by blue color represent the region that fall inside the cropping window while columns marked by red color would be discarded after cropping.

cropping factor as the percentage of block columns to discard during cropping. The optimum cropping factor, which balances the amount of cropping and warping, needs to be determined at the first place. Since the importance map is computed at the DCT block level, we perform cropping through discarding unimportance columns of DCT blocks (size of 4×4 in our case). In the following, we assume the resizing is conducted along the horizontal direction. The same process can be applied to resizing in the vertical direction.

To compute the optimum cropping factor, we first determine the minimum and the maximum cropping factors, which are precomputed values based on the desired resizing factor. For example, if the resizing factor is 0.50 (resized to half width), the minimum cropping factor is set to be $0.50 + (1.0 - 0.50) \times 20\% = 0.60$, while the maximum cropping factor is $0.50 + (1.0 - 0.50) \times 80\% = 0.90$. Within this cropping range, we compute the average visual importance value for all possible cropping options. Our goal is to find a rectangular region in the input video that contains the maximum average visual importance value. Since we are only dealing with a limited number of options, we use exhaustive search to compute the best cropping window of a given window length. After computing the best cropping window for each window length (see Fig. 6), we look for the optimum window length that contains the maximum average visual importance value. Once we have determined the optimum cropping window of the given video scene, we discard the columns of DCT blocks that fall outside of this window. This cropping window will be applied to all the frames of the same scene.

D. Column Mesh Deformation

After partially resizing through cropping, the video will be further resized to its desired result through nonuniform warping. For spatial domain retargeting methods [2]–[4], [7], a quad-shape mesh is often used to guide the warping operation. However, conducting a quad-to-quad deformation is a challenging task in the compressed domain since it is difficult to compute the corresponding DCT coefficients of a block after it is warped to an arbitrary-shape quadrilateral. Instead of adopting the conventional quad-shape mesh, we use a column-shape mesh to guide the warping process.

For mesh warping, we extend the formulation in [22] and adjust it to our proposed column-mesh structure. In addition, two new energy terms are added to preserve motion and temporal coherency.

Consider a column mesh, represented by $\mathbf{M}^t = \{\mathbf{V}^t, \mathbf{C}\}$, as shown in Fig. 7, where $\mathbf{C} = \{c_1, c_2, ..., c_n\}$ denotes the set of columns, and $\mathbf{V}^t = \{v_0^t, v_1^t, v_2^t, ..., v_n^t\}$ is the set of vertex positions, with v_i^t representing the horizontal coordinate of vertex i. The width between consecutive mesh vertices is set to 4, which is the same as the length of transform block. We place such a mesh on each input video frame. We take the initial vertex positions of each frame, $\mathbf{V}^t$, as the input and solve for their new positions $\mathbf{V}_{new}^t = \{\bar{v}_0^t, \bar{v}_1^t, \bar{v}_2^t, ..., \bar{v}_n^t\}$ by minimizing an objective function as described below.

1) Shape Deformation: During the resizing process, we want to preserve the shape of columns with high saliency while allowing columns of lower saliency to be squeezed or stretched more. For each each frame t, we measure the amount of shape

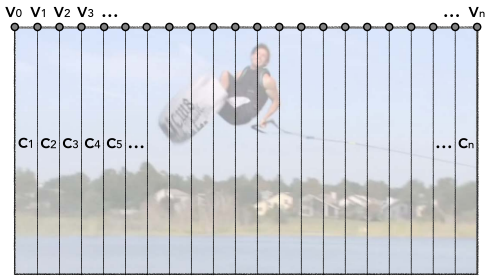


Fig. 7. The column mesh used for compressed-domain video resizing. The mesh $\mathbf{M} = \{\mathbf{V}, \mathbf{C}\}$ includes a set of vertices $\mathbf{V}^t = \{v_0^t, v_1^t, v_2^t, ..., v_n^t\}$ and a set of columns $\mathbf{C} = \{c_1, c_2, ..., c_n\}$.

deformation of column i as

$$D^t(c_i) = \left\| (\bar{v}_i^t - \bar{v}_{i-1}^t) - l_i (v_i^t - v_{i-1}^t) \right\|^2, \quad i = 1, 2, 3 \cdots, n,$$

where l_i is the optimum scaling factor for C_i and is updated at each iteration as:

$$l_i^{(k)} = \frac{\tilde{l}_i}{|v_i^{t(k)} - v_{i-1}^{t(k)}|}$$

where $\tilde{l}_i$ is the original width of column i before deformation. $v_i^{t(k)}$ is the vertex position of column i at iteration k.

The shape deformation energy of all columns is given by

$$E_d = \sum_t \sum_{c_i \in \mathbf{C}} I^t(c_i) \cdot D^t(c_i), \tag{4}$$

where $I^t(c_i)$ is the average visual importance of column c_i at frame t.

2) Vertex Order Preservation: It is possible that some vertices may flip over each other after mesh deformation, leading to unwanted artifacts. To avoid it, we preserve their relative positions with respect to their immediate neighboring vertices by maintaining their relative barycentric coordinate. For vertices v_i, we minimize:

$$E_v = \sum_t \sum_i \left\| \bar{v}_i^t - \sum_{v_j \in \mathbf{N}(v_i)} m_{ij} \cdot \bar{v}_j^t \right\|^2, \tag{5}$$

where m_{ij} is the barycentric coordinate of v_j with respect to v_i, and $\mathbf{N}(v_i)$ represents the set of neighboring vertices of v_i.

3) Temporal Coherency: To preserve temporal coherency and avoid jittering artifacts, we enforce the temporal smoothness of vertex positions across neighboring frames. Specifically, we try to minimize

$$E_c = \sum_t \sum_i \left\| \bar{v}_i^{t+1} - \bar{v}_i^t \right\|^2 \tag{6}$$

where v_i^t are vertex positions of the previous frame.

4) Motion Preservation: As noted in [3], simply enforcing per-pixel smoothing along the temporal dimension, which does not take object or camera motion into account, yields poor re-sizing results. Under this scenario, an object that moves from the left to right of the frame may be resized differently throughout the whole scene. For example, as shown in Fig. 8 (top), the scene consists of camera panning from right to left and the tree size has changed across the frames without motion preservation.

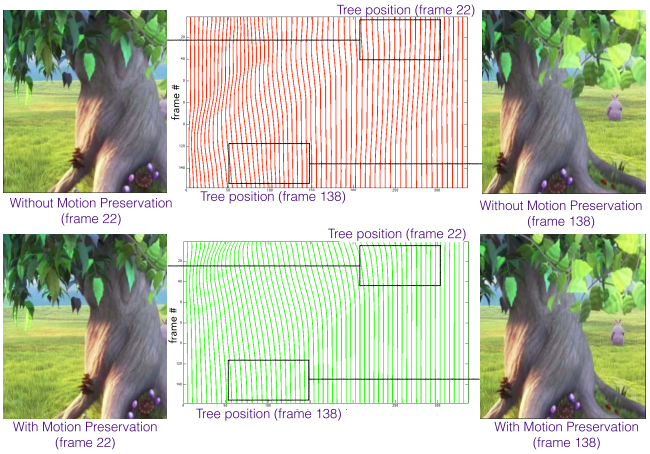


Fig. 8. The impact of using motion preservation in the column mesh deformation. Top: resizing results without considering motion preservation and the corresponding column vertex movement paths. The tree is resized inconsistently at different frames. Bottom: resizing results that considers motion preservation and the corresponding column vertex movement path. The tree size undergoes more consistent transformation throughout the entire video sequence.

To account for object motion and camera motion, we exploit the camera parameters estimated in Section V-B. Specifically, we utilize the camera right panning parameter, p_r, since we resize the input video along the horizontal direction. Similarly, the down panning parameter, p_d, will be used if we perform resizing along the vertical direction. To achieve motion-aware resizing, we minimize the following energy:

$$E_m = \sum_t \sum_i \left\| (\bar{v}_i^{t+1} - \bar{v}_{i-1}^{t+1}) - (\bar{u}_i^t - \bar{u}_{i-1}^t) \right\|^2, \tag{7}$$

where $u_i^t = v_i^{t+1} - p_r^t$, and p_r^t is the right panning parameter of frame t. $\bar{u}_i^t$ represents the corresponding position of u_i^t after mesh deformation. Since u_i^t may not align with any of v_i^t, we represent it with a linear combination of the column mesh vertices in its immediate vicinity as

$$u_i^t = \sum_{v_j \in \mathbf{N}(u_i)} m_{ij} \cdot v_j^t,$$

where m_{ij} is the barycentric coordinates of u_i^t w.r.t. column vertices v_j^t of its immediate vicinity $\mathbf{N}(u_i)$. Note that we only consider columns whose corresponding positions in the previous frames are still within the frame boundary. For other columns, their temporal coherency will be preserved by the temporal coherency energy defined in Eq. (6). As shown in Fig. 8 (bottom), after incorporating the energy function for motion preservation, the tree size has been preserved throughout all frames.

5) Joint Optimization for Column Mesh Deformation: Combining all energy terms in Eqs. (4)–(7), we solve for the deformed column mesh by minimizing the following objective function:

$$E = E_d + \alpha \cdot E_v + \beta \cdot E_c + \gamma \cdot E_m, \tag{8}$$

subject to the boundary constraint. The weighting coefficients are empirically set to $\alpha = 1.0$, $\beta = 50.0$ and $\gamma = 10.0$ in our experiments. We can represent the objective function in Eq. (8) and its constraint in matrix format as:

$$\begin{aligned} &\underbrace{\|\mathbf{DV} - l(\mathbf{V})\|^2}_{\text{Eq.(4)}} + \underbrace{\|\mathbf{CV}\|^2}_{\text{Eq.(5)}} + \underbrace{\|\mathbf{TV} - \mathbf{S}\|^2}_{\text{Eq.(6)}} \\ &\quad + \underbrace{\|\mathbf{MV} - \mathbf{N}\|^2}_{\text{Eq.(7)}} + \|\mathbf{PV} - \mathbf{Q}\|^2, \end{aligned} \tag{9}$$

where the last term, $\|\mathbf{PV} - \mathbf{Q}\|^2$, denotes the position constraint imposed by the target video size. The matrix expression given above can be rewritten as

$$\min_{\mathbf{V}} \|\mathbf{AV} - \mathbf{b}(\mathbf{V})\|^2, \tag{10}$$

where

$$\mathbf{A} = \begin{pmatrix} \mathbf{D} \\ \mathbf{C} \\ \mathbf{T} \\ \mathbf{M} \\ \mathbf{P} \end{pmatrix}, \quad \mathbf{b}(\mathbf{V}) = \begin{pmatrix} l(\mathbf{V}) \\ 0 \\ \mathbf{S} \\ \mathbf{N} \\ \mathbf{Q} \end{pmatrix}.$$

The nonlinear least-squares optimization problem in Eq. (10) can be solved through an iterative Gauss-Newton method. The vertex positions are initialized with a homogenous resizing condition and updated iteratively via

$$\mathbf{V}^{(k)} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}(\mathbf{V}^{(k-1)}) = \mathbf{H} \cdot \mathbf{b}(\mathbf{V}^{(k-1)}), \tag{11}$$

where $\mathbf{V}^{(k)}$ is the vector of vertex positions after the k-th iteration. As $\mathbf{H}$ is only dependent on $\mathbf{A}$, it can be precomputed and stay fixed during the iteration process.

E. Transform Domain Block Resizing

Once the deformed mesh is computed, we use this information to resize the video frame in the compressed domain. We employ the DCT domain resizing method in [14], which supports resizing with arbitrary factors. In [14], each block in the resized frame is downsized from a rectangular region (called the supporting area) in the original frame as illustrated in Fig. 9. The resizing is conducted in two separate steps: 1) extracting the supporting area from the original frame, and 2) downsizing the supporting area to a square-size output block.

In the proposed system, the block resizing task is conducted at the transform block level. With the information of the deformed mesh, we first compute the supporting area of each output macroblock through reverse mapping. Specifically, for every vertex $\bar{v}_i$ in the retargeted frame, we compute its corresponding coordinates in the original frame through interpolation. Then, every $N \times N$ block ($N = 4$ in our case) in the resized frame is reverse-mapped to an $N \times N'$ block in the original frame. The height of the supporting area equals to the output block as we only consider resizing along the horizontal direction. It should be noted that the supporting area may cover multiple transform blocks in the original frame, and some blocks may be only partially covered by the supporting area.

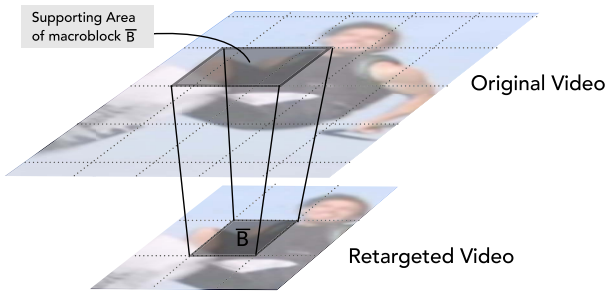


Fig. 9. Illustration of the supporting area, which each macroblock of the output video frame is resized from its corresponding supporting area in the original frame.

By following [14], the supporting area is resized to the output block via

$$\bar{B} = \begin{bmatrix} I_N & 0 \end{bmatrix}_{N \times N'} \cdot M_L \cdot \sum_{B_i \in \mathbf{S}(\bar{B})} B_i \cdot M_R \cdot \begin{bmatrix} I_N \\ 0 \end{bmatrix}_{N' \times N}, \tag{12}$$

where I_N is an identity matrix of size $N \times N$, and B_i are DCT coefficients of blocks that are covered by the supporting area of $\bar{B}$, M_L and M_R are the DCT transforms of shifting matrices, and $\mathbf{S}(\bar{B})$ denotes the supporting area of block $\bar{B}$.

VI. Re-Encoding

In the last stage, we re-encode the block DCT coefficients of the retargeted result to an H.264/AVC-compliant bitstream. The re-encoding step is essentially the reverse process of that in the partial decoding stage. It should be noted that quantization is not required here since we did not perform inverse quantization in the partial decoding stage. The encoder forms a prediction of each macroblock based on previously-coded data either from the current frame using intra prediction [18] or from other frames that have already been coded [17]. The prediction is then subtracted from the current macroblock to form a residual and exported to the output bitstream. However, since each macroblock is modified during the retargeting process, there are two additional issues to be addressed: 1) macroblock type selection, and 2) motion vector re-estimation. We will describe our solutions below.

A. Macroblock Type Selection

For selecting the macroblock types, we employ the MTSS scheme [28], which was originally proposed for the compressed-domain video downsizing system. This scheme can be modified to fit our retargeting framework. For the H.264/AVC video coding standard, a macroblock in a P frame can be intra-coded, forward predicted, or skipped. A predicted B-frame macroblock can be intra-code, forward, backward, or bi-directionally predicted, or skipped. We use the area proportion of each macroblock w.r.t. the supporting area to determine the prediction mode of output macroblocks.

A retargeted macroblock is intra-coded if and only if more than 50% of its supporting area covers macroblocks that are originally intra-coded. For B frames, a retargeted macroblock is to be forward predicted if more than 70% of its supporting area covers original macroblocks that have forward prediction.

TABLE I
SCENE CHANGE DETECTION RESULTS ON TWO TEST SEQUENCES: BIG BUCK BUNNY AND ELEPHANTS DREAM

Sequence	Total frame #	Abrupt change			Gradual change			Total			Recall	Precision
		N_C	N_m	N_f	N_C	N_m	N_f	N_C	N_m	N_f		
big buck bunny	14,315	127	0	7	1	4	0	128	4	7	96.97%	94.81%
elephants dream	15,691	99	14	22	5	3	0	104	17	22	85.95%	85.33%

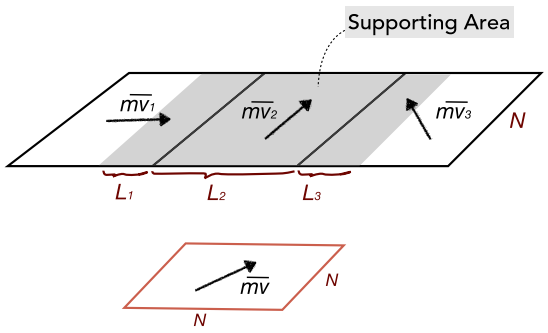


Fig. 10. Motion vector of a retargeted block, $\bar{mv}$, is estimated using the motion vectors in its supporting area: $\bar{mv}_1$, $\bar{mv}_2$ and $\bar{mv}_3$.

Similarly, it is backward predicted if 70% of its supporting area covers original macroblocks that have backward prediction. In the case where the supporting area covers both forward and backward predicted macroblocks while both are lower than 70%, then the prediction type that has a higher percentage will determine the retargeted macroblock type. In case of a tie (50% backward, 50% forward) the macroblock is bi-directionally predicted.

B. Motion Vector Refinement

The conventional way to generate an H.264/AVC bitstream of a resized video requires decompressing it and then applying a spatial-domain motion estimation technique to recompute motion vectors in the pixel domain. However, recomputing motion vectors is a computationally intensive procedure and it typically takes 60% or higher of the workload of a video encoder [29]. To remedy this, we propose a motion vector refinement technique that works directly in the compressed domain and re-estimate the new motion vector using motion vectors of the original macroblocks. The proposed refinement technique is intended only for inter-frame coding, as intra frames are coded independently and do not contain any motion information.

Consider the case of resizing a supporting area of size $N \times N'$ ($N' = L_1 + L_2 + L_3$) to the output macroblock of size $N \times N$, as shown in Fig. 10. The supporting area covers three macroblocks in the original frame with motion vectors $\bar{mv}_1$, $\bar{mv}_2$ and $\bar{mv}_3$, respectively. The motion vector $\bar{mv}$ of the resized macroblock is estimated as

$$\overline{mv} = \frac{N}{N'} \frac{\sum_i \bar{mv}_i \cdot L_i}{\sum_i L_i}, \tag{13}$$

where $\bar{mv}_i$ is the motion vector of original macroblock i and L_i is the length of macroblock i in the supporting area. It should be noted that intra macroblocks are considered as blocks with zero motion vectors. The effectiveness of our motion vector refinement scheme is validated through experiments in the next section.

VII. EXPERIMENTAL RESULTS

In this section, we demonstrate the effectiveness of the proposed compressed-domain video retargeting solution by comparing its results with previous spatial domain video retargeting methods [2], [4], [5]. We begin with evaluating the scene change detection method in our proposed system, followed by visually comparing our retargeting results with other spatial-domain techniques. The effectiveness of our motion vector refinement technique is then validated, followed by computation complexity analysis on the proposed system versus spatial-domain retargeting methods. Finally, we present subjective quality evaluation results conducted on 56 subjects.

A. Scene Change Detection Evaluation

In Table I, we evaluate the performance of our scene change detection algorithm proposed in Section V-A. The two test sequences used in this experiment, *big buck bunny* and *elephants dream*, contain various types of scene changes and camera motions. In our experiment, abrupt and gradual scene changes are evaluated separately.

The performance of a scene-change-detection algorithm is measured in terms of recall and precision rate. The recall and prevision rate are defined as:

$$\text{Recall} = \frac{N_c}{N_c + N_m} \times 100\%$$
$$\text{Precision} = \frac{N_c}{N_c + N_f} \times 100\%$$

where N_c, N_m and N_f represent the number of correct, miss and false detections, respectively.

Our proposed scene change detection algorithm performs relatively well on *big buck bunny* sequence, with recall rate of 96.97% and precision rate of 94.81%. The algorithm has better performance in detecting abrupt changes than gradual changes. While none of the abrupt changes are missed by our method, the number of missed gradual changes is relatively high (4 out of 5 were missed). On the other hand, our method has relatively lower recall and prevision rates on *elephants dream* sequence. This is mainly due to the existence of non-static background and fast camera motions in the video content. This implies that our detection method is not robust enough to perform well on all types of video sequences and there is still room for further improvement.

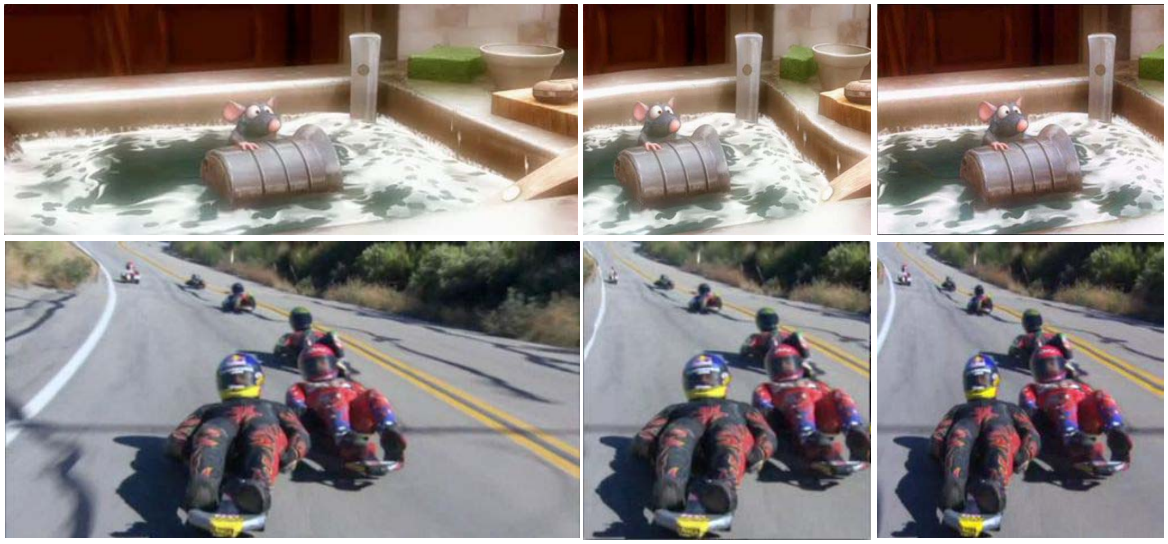


Fig. 11. Performance comparison of the proposed solution versus the seam carving method [5] for sequence *rat* and *roadski*. From left to right: the original video sequence, the result of seam carving [5] and our result. The seam carving method is incapable of preserving the shape of prominent edges, as can be observed at from the distortions on the curb-line in the *rat* sequence and the road-lines of the *roadski* sequence.

Fig. 12. Performance comparison of the proposed solution versus the pixel-warp method [2] for sequence *waterski*. From left to right: the original video sequence, the result of [2] and our result. The pixel-warp method over-squeezes the water wave region of the *waterski* sequence, leading to noticeable artifacts. In contrast, our method incorporates cropping into the whole procedure and performs better in preserving the original content.

B. Visual Quality Comparison

We show the retargeting results of the proposed solution with three state-of-the-art spatial domain methods [2], [4], [5] in Figs. 11–13 for visual comparison.

The performance comparison with the seam carving method [5] is given in Fig. 11. The seam carving method resizes a video through continuously removing seams. In some cases, it is incapable of preserving prominent edges. As shown in Fig. 11, while seam carving introduces noticeable artifacts to the edge regions in both sequences (*rat* and *roadski*), the results of our solution contain fewer visual artifacts as the shape of prominent edges is better preserved.

We compare the performance of the proposed solution with that of the pixel-warp retargeting method [2] in Fig. 12. Our method differs from the pixel-warp retargeting method in that we have incorporated cropping in the resizing procedure, thereby avoiding over-squeezing the original video content. When the change in the aspect ratio is significant, as shown in the example of Fig. 12, the method entirely based on warping [2] over-squeezes the relative non-important video content, leading to noticeable visual distortion.

Finally, we compare the performance of our solution with the method proposed by Wang *et al.* [4], which is a state-of-the-art method with optimized computational efficiency. Being similar to our method, the method in [4] incorporates both cropping and warping. For most test sequences, our method achieves comparable performance as that of [4]. One example is shown in Fig. 13. On the other hand, since our method operates directly in the compressed domain, it has an advantage in terms of computational and memory cost saving. This will be analyzed in Sec. VII-D.

C. Effectiveness of Motion Vector Refinement

In the re-encoding stage, to avoid the computationally expensive procedure of motion search, we proposed a motion vector refinement scheme that computes the new motion vector of each retargeted macroblock using original motion vectors. We evaluate the effectiveness of this approach by comparing it with full motion search in terms of encoding PSNR. In the experimental setup, the retargeted sequence is encoded into an H.264/AVC (baseline profile) bitstream using the JM reference software [30]. All test sequences are encoded at a bit rate of 2 Mbps and a frame rate of 15 fps. The results of six different test sequences are listed in Table II.

As listed in Table II, the motion vectors generated by our refinement scheme offer comparable encoding PSNR values as

Fig. 13. Performance comparison of the proposed solution versus the approach by Wang *et al.* [4] for sequence *big buck bunny*, *car* and *building*. From left to right: the original video sequence, the result of [4] and our result. Our method achieves comparable results in terms of visual quality, yet it has a lower computational cost and memory consumption.

TABLE II
PERFORMANCE COMPARISON OF RE-ENCODING USING THE PROPOSED MOTION VECTOR REFINEMENT APPROACH VERSUS FULL SEARCH

Sequence name	Full Motion Search	Our approach	Difference
car	41.62 dB	39.87 dB	1.75 dB
big buck bunny	43.83 dB	43.28 dB	0.55 dB
waterski	46.66 dB	45.01 dB	1.65 dB
rat	45.46 dB	44.81 dB	0.65 dB
roadski	45.91 dB	44.70 dB	1.21 dB
building	40.44 dB	36.91 dB	3.53 dB

that of full search, which can be viewed as the upper bound. On the average, the PSNR value of the proposed scheme is about 1 dB lower than that of the full search. For sequences with less movement such as *big buck bunny* and *rat*, the PSNR difference can be as low as 0.60 dB. However, for sequences with significant moving background (such as *building*), the estimated motion vectors using our approach may become less reliable, leading to relatively larger difference (3.53 dB difference for *building* sequence). In all, the proposed solution achieves fast and accurate re-estimation of the motion vector for the output target video while significantly reducing the complexity of full search.

D. Computational Complexity Analysis

In Table III, we further compare the computational complexity of the proposed solution with spatial domain video retargeting algorithms. The experiments were conducted on a segment of *big buck bunny* sequence (158 frames, size: 672×384) encoded using the H.264/AVC baseline profile coding standard. It should be noted that the computational complexity for each frame may be different, as different prediction modes are used for each frame. Table III shows the per-frame total operation cost averaged over all frames. In the encoding stage, the EPZS approach [31] is used for fast motion search.

In the decoding stage, spatial-domain methods demand the entire full decoding process, including inverse DCT, inverse quantization, motion compensation and intra prediction. Our solution operates directly in the compressed domain, thereby avoiding both inverse DCT and inverse quantization, leading to 13.72% savings in the total operation cost (see Table III). For the encoding stage, with the proposed motion vector re-estimation scheme, our proposed system results in 30.17% and 99.92% savings in the total operation costs as compared with the fast [31] and full motion search approach, respectively.

In Table IV, we show the computation complexity analysis for two other test sequences: *roadski* (99 frames, size: 540×280) and *building* (104 frames, size: 720×376). Experimental results on these two sequences also demonstrates that our proposed systems leads to significant cost savings in both encoding and decoding stage.

TABLE III

COMPLEXITY ANALYSIS FOR RETARGETING THE BIG BUCK BUNNY SEQUENCE FOR DCT DOMAIN VERSUS THE SPATIAL DOMAIN

Our partial decoding module				**Full decoding**			
Procedure	Add	Multiply	Shift	Procedure	Add	Multiply	Shift
Compressed-domain inverse motion compensation (MBIMC[17]/ TDIP[18])	1,706,693	287,533	-	IDCT	1,032,192	-	258,048
				Inverse quantization	-	474,473	-
				Inverse motion compensation	241,732	-	-
				Intra prediction	12,892	1,109	6,384
Total operations count (per frame)	2,569,353 (Savings: 13.72%)			Total operations count (per frame)	2,977,996		
Our re-encoding module				**Full encoding**			
Procedure	Add	Multiply	Shift	Procedure	Add	Multiply	Shift
Compressed-domain motion compensation (MBIMC[17]/ TDIP[18])	853,374	143,777	-	DCT	516,096	-	129,024
				Quantization	-	237,237	-
Motion vector refinement	944	2,361		Motion compensation	120,866	-	-
				Intra prediction	6,446	555	3,192
				Motion estimation (full search)	1,529,982,109	-	-
				Motion estimation (fast search)	362,126	-	-
Total operations count (per frame)	1,292,703 (Savings - full search: 99.92%) (Savings - fast search: 30.17%)			Total operations count (per frame)	1,531,471,108 (full search) 1,851,124 (fast search)		

TABLE IV

COMPLEXITY ANALYSIS FOR RETARGETING THE ROADSKI AND BUILDING SEQUENCE FOR DCT DOMAIN VERSUS THE SPATIAL DOMAIN

roadski sequence

Procedure	**Total operations count**	**Procedure**	**Total operations count**	**Savings**
Our partial decoding module	707,750	Full decoding	1,015,718	30.32%
Our partial re-encoding module	355,970	Full encoding (full search)	399,743,392	99.91%
		Full encoding (fast search)	602,353	40.90%

building sequence

Procedure	**Total operations count**	**Procedure**	**Total operations count**	**Savings**
Our partial decoding module	1,310,533	Full decoding	1,704,394	23.11%
Our partial re-encoding module	659,493	Full encoding (full search)	500,202,682	99.87%
		Full encoding (fast search)	1,042,871	36.76%

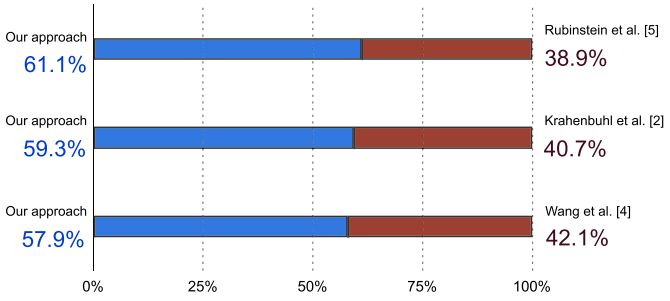


Fig. 14. Pairwise comparison results of 56 user study participants, which show that users have a preference on the visual quality of our solution over the other three benchmarking methods proposed in [2], [4], [5].

E. Subject Visual Quality Test

Lastly, we report the subject test results on the visual quality of the proposed retargeting solution via a user study conducted on 56 participants (27 female and 29 male, aged between 22 and 54). The experiments were conducted in the typical laboratory environment. We used 6 different videos in the experiment and retargeted each video to 50% width using the method in [2], [4], [5] and our method.

In the subject test, we presented the output video sequences obtained by two retargeting methods side-by-side to the observer, who is then asked to choose the better one among the two. Among each pair, one is our own result while the other is the result from one of the state-of-the-art spatial domain methods in [2], [4], [5]. The entire user study consists of $6 \times 4 = 24$ video pairs and we received $24 \times 56 = 1344$ answers overall. It took on average 15-20 minutes for each participant to complete the user study. To minimize user bias, we randomized the order of test pairs and hid all technical details from the participants.

We show the results of our conducted user study in Fig. 14. Our results were favored in 61.1% (821 of 1344) of the comparisons with Rubinstein et al. [5], in 59.3% (797 of 1344) of the comparisons with Krahenbuhl et al. [2], and in 57.9% (778 of 1344) of the comparisons with Wang et al. [4]. The study results show that users have a stronger preference on the visual quality of our solution over the other three benchmarking methods.

VIII. CONCLUSION

In this paper, we proposed a practical video retargeting system that operates directly on DCT coefficients and motion vectors in the compressed domain. This solution avoids the computationally expensive process of de-compressing, processing, and recompression. As the system uses the DCT coefficients directly for processing, only partial decoding of video streams is needed. The proposed solution achieves comparable (or slightly better) visual quality performance as that of several state-of-art spatial domain video retargeting methods, yet it significantly reduces the computational and storage costs. Although the proposed system uses the latest H.264/AVC coding standard as an example, the general methodology is applicable to other video coding standards as well.

This study can be extended along the following directions:

1) The scene change detection method in our proposed system, may not be robust enough for scenes that contain fast camera motions and large moving foregrounds. More sophisticated compressed-domain scene change detection algorithms (such as [32]) can be adopted to account for more dynamic scenarios.
2) The column-mesh structure in our proposed solution only squeezes or stretches video content along one direction. A more sophisticated mesh structure (e.g. axis-aligned deformation [33]) can be utilized to allow homogenous scaling of important objects.
3) For motion vector refinement, we assigned zero motion vector to intra-coded blocks. More reliable estimation of motion vectors for intra blocks is also our future effort.

REFERENCES

[1] L. Wolf, M. Guttmann, and D. Cohen-Or, "Non-homogeneous content-driven video-retargeting," in *Proc. 11th IEEE ICCV*, Oct. 2007, pp. 1–6.

[2] P. Krähenbühl, M. Lang, A. Hornung, and M. Gross, "A system for retargeting of streaming video," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 126:1–126:10, Dec. 2009.

[3] Y.-S. Wang, H.-C. Lin, O. Sorkine, and T.-Y. Lee, "Motion-based video retargeting with optimized crop-and-warp," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 90:1–90:9, Jul. 2010.

[4] Y.-S. Wang, J.-H. Hsiao, and T.-Y. Lee, "Scalable and coherent video resizing with per-frame optimization," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 88:1–88:8, Aug. 2011.

[5] M. Rubinstein, A. Shamir, and S. Avidan, "Improved seam carving for video retargeting," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–9, Aug. 2008.

[6] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Discontinuous seam-carving for video retargeting," in *Proc. IEEE CVPR*, Jun. 2010, pp. 569–576.

[7] Y.-S. Wang, H. Fu, O. Sorkine, T.-Y. Lee, and H.-P. Seidel, "Motion-aware temporal coherence for video resizing," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 127:1–127:10, 2009.

[8] A. Shamir and O. Sorkine, "Visual media retargeting," in *Proc. ACM SIGGRAPH ASIA*, New York, NY, USA, 2009, pp. 11:1–11:13.

[9] L.-Q. Chen, X. Xie, X. Fan, W.-Y. Ma, H.-J. Zhang, and H.-Q. Zhou, "A visual attention model for adapting images on small displays," *Multimedia Syst.*, vol. 9, no. 4, pp. 353–364, 2003.

[10] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," in *ACM SIGGRAPH*, New York, NY, USA, 2007, pp. 1–10.

[11] M. Rubinstein, A. Shamir, and S. Avidan, "Multi-operator media retargeting," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–23, Jul. 2009.

[12] Y.-S. Wang, C.-L. Tai, O. Sorkine, and T.-Y. Lee, "Optimized scale-and-stretch for image resizing," in *ACM SIGGRAPH Asia*, New York, NY, USA, 2008, pp. 1–8.

[13] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 4, pp. 461–474, Apr. 2001.

[14] H. Shu and L.-P. Chau, "An efficient arbitrary downsizing algorithm for video transcoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 6, pp. 887–891, Jun. 2004.

[15] Y. Fang, Z. Chen, W. Lin, and C.-W. Lin, "Saliency detection in the compressed domain for adaptive image retargeting," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 3888–3901, Sep. 2012.

[16] H.-M. Nam, K.-Y. Byun, J.-Y. Jeong, K.-S. Choi, and S.-J. Ko, "Low complexity content-aware video retargeting for mobile devices," *IEEE Trans. Consum. Electron.*, vol. 56, no. 1, pp. 182–189, Feb. 2010.

[17] S. Porwal and J. Mukhopadhyay, "A fast DCT domain based video downscaling system," in *Proc. IEEE ICASSP*, vol. 2. May 2006, pp. 1–2.

[18] C. Chen, P.-H. Wu, and H. Chen, "Transform-domain intra prediction for H.264," in *Proc. IEEE ISCAS*, vol. 2. May 2005, pp. 1497–1500.

[19] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in *Proc. IEEE CVPR*, Jun. 2007, pp. 1–8.

[20] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 11, pp. 1254–1259, Nov. 1998.

[21] Y. Fang, W. Lin, Z. Chen, C.-M. Tsai, and C.-W. Lin, "Video saliency detection in the compressed domain," in *Proc. 20th ACM Int. Conf. Multimedia*, 2012, pp. 697–700.

[22] J. Zhang and C.-C. Kuo, "Region-adaptive texture-aware image resizing," in *Proc. IEEE ICASSP*, Mar. 2012, pp. 837–840.

[23] (2013). *Video Trace Library* [Online]. Available: http://trace.eas.asu.edu/yuv/

[24] R. Wang, H.-J. Zhang, and Y.-Q. Zhang, "A confidence measure based moving object extraction system built for compressed domain," in *Proc. IEEE ISCAS*, vol. 5. May 2000, pp. 21–24.

[25] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[26] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof, "Anisotropic huber-L^1 optical flow," in *Proc. BMVC*, Sep. 2009, pp. 1–11.

[27] R. Wang and T. Huang, "Fast camera motion analysis in MPEG domain," in *Proc. ICIP*, vol. 3. 1999, pp. 691–694.

[28] M. Hashemi, L. Winger, and S. Panchanathan, "Macroblock type selection for compressed domain down-sampling of MPEG video," in *Proc. IEEE Can. Conf. Electr. Comput. Eng.*, vol. 1. May 1999, pp. 35–38.

[29] B. Shen, I. Sethi, and B. Vasudev, "Adaptive motion-vector resampling for compressed video downscaling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 6, pp. 929–936, Sep. 1999.

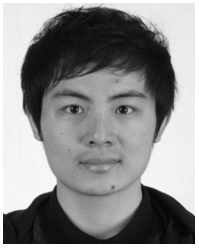
[30] (2013). *H.264/AVC JM Reference Software* [Online]. Available: http://iphome.hhi.de/suehring/tml/

[31] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," *Proc. SPIE*, vol. 4671, pp. 1069–1079, Jan. 2002.

[32] J. Meng, Y. Juan, and S.-F. Chang, "Scene change detection in an MPEG-compressed video sequence," *Proc. SPIE*, vol. 2419, pp. 14–25, Apr. 1995.

[33] D. Panozzo, O. Weber, and O. Sorkine, "Robust image retargeting via axis-aligned deformation," *Comput. Graph. Forum*, vol. 31, no. 2, pp. 229–236, Jul. 2012.

Jiangyang Zhang received the B.S. degree in telecommunications engineering from Zhejiang University, China, in 2008. He is currently pursuing the Ph.D. degree with the University of Southern California, Los Angeles. He joined the Media Communications Laboratory in 2010. His main research interest lies in image processing.



Shangwen Li received the B.S. and M.S. degrees in electrical engineering from Zhejiang University, Hangzhou, China, in 2008 and 2011, respectively. Since 2012, he has been with the Media Communications Laboratory, University of Southern California, Los Angeles. His research interests include computer vision, machine learning, and video coding.

C.-C. Jay Kuo (F'99) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1980, and the M.S. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1985 and 1987, respectively, all in electrical engineering. He is currently the Director of the Media Communications Laboratory and a Professor of electrical engineering, computer science, and mathematics with the University of Southern California, Los Angeles, and the President of the Asia-Pacific Signal and Information Processing Association. His current research interests include digital image/video analysis, multimedia data compression and information forensics, and security. He is the co-author of over 210 journal papers, 850 conference papers, and ten books.

He is a fellow of the American Association for the Advancement of Science and the International Society for Optical Engineers.