Fast JND-Based Video Carving with GPU Acceleration for Real-Time Video Retargeting

Chen-Kuo Chiang, *Student Member, IEEE,* Shu-Fan Wang, *Student Member, IEEE,* Yi-Ling Chen, *Student Member, IEEE,* Shang-Hong Lai, *Member, IEEE*

***Abstract*—A recently developed image technique, seam carving, has been proved to be a useful tool for content-adaptive spatially non-uniform image resizing with the purpose of optimal display on a screen of reduced resolution or different aspect ratio. In this paper, we present a fast algorithm for real-time content-aware video retargeting based on the improved seam carving method proposed in this paper. The proposed algorithm is designed to be highly parallelizable and suitable for running on a multi-core architecture. First, two novel operators, i.e. seam update and seam split, are introduced to analyze an image for detecting the local and global seams with minimum costs very efficiently. With these operators, parallel processing can be achieved to determine multiple seams simultaneously. In addition, the saliency measure is extended with a Just-Noticeable-Distortion (JND) model which makes the resized video more consistent with human perception. We demonstrate the efficiency of the above new components with a GPU implementation. In addition, the proposed fast seam carving algorithm is extended for video retargeting. To the best of our knowledge, this is the first paper based on the seam carving method to achieve real-time video retargeting on GPU. Experimental results on video sequences of various characteristics are demonstrated to show the superior performance of the proposed algorithm in comparison with the existing content-adaptive image/video resizing methods.**

***Index Terms*—Graphics processing unit (GPU), seam carving, video retargeting.**

I. Introduction

As moderate video camcorder becomes more and more affordable, users can easily create their own media content. However, the media content is usually generated to be displayed on devices of a specific resolution. To fit it into different types of display devices, the video frames could be resized adaptively based on their content and the aspect ratio could be changed while the important part of the content can be retained as best as possible. This problem is referred to as *video retargeting* [1], which is to adapt a video for better viewing on a different display device with different resolution and aspect ratio. Fig.1 depicts an example for this problem. Some existing methodss, such as image scaling or cropping, are fast and simple solution but they may cause unpleasant viewing experiences. Scaling tends to lose detailed information and sometimes makes the region of interest too small for viewing, whereas cropping discards data outside the cropping window and it may lose important information. Therefore, it is important to develop an efficient and effective approach to retarget videos to some common resolutions used in various display devices.

{ckchiang,shufan,yilin,lai}@cs.nthu.edu.tw

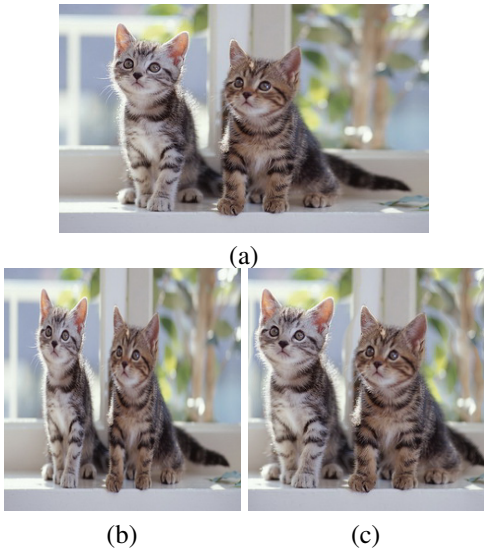


Fig. 1. (a) The input image. (b) Uniformly scaling the image in the horizontal direction. (c) Adaptively removing the less important regions according to image content.

Seam carving [2] is a content-aware image resizing technique that changes the image size by recursively removing or inserting a seam with minimal cost from top to down or from left to right in the image. The cost of a path or *seam* is defined according to an importance map estimated for all pixels. After successively removing or inserting paths with minimal costs, the resized image still retains the most perceptually important regions.

Seam carving in general provides satisfactory adaptive image resizing results, but it is computationally too expensive for video retargeting. In this paper, we present a multi-seam search scheme for efficient seam carving. The proposed method analyzes an image to detect the local and global seams with minimal costs. Local seams are determined by removing multiple seams simultaneously, thus improving the efficiency of the original seam carving algorithm. In addition, we introduce a just-noticeable-distortion (JND) model into the saliency map required in seam carving, which is computed from various types of image features, such as gradient, entropy, visual saliency, segmentation, etc. The proposed JND-based spatio-temporal model makes saliency measure more consistent with human perception.

Based on the proposed new components, we extend the proposed fast seam carving algorithm for efficient video retargeting. In addition, this algorithm combines temporal

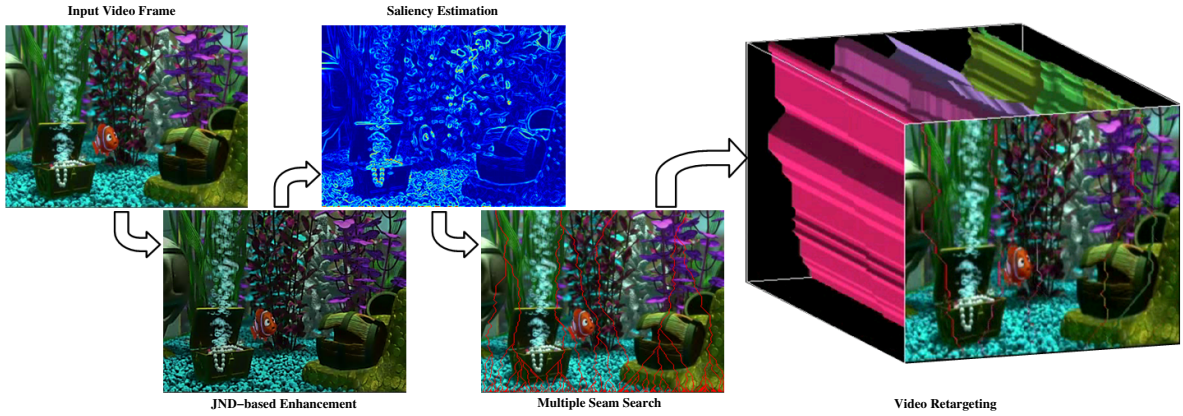


Fig. 2. Algorithmic flow of the proposed method

coherence with a combined JND-based measure in the spatial and temporal domains into the saliency measure, thus leading to smooth viewing experiences. We demonstrate that the proposed video carving algorithm is computationally efficient compared to other existing methods. On the other hand, general-purpose computing on graphics processing units (GPGPU) has attracted much attention in recent years [3]. The parallel computing power of GPUs has been employed to speed up the computationally intensive applications, which were traditionally processed with CPUs. We demonstrate that the proposed method can greatly benefit from GPU implementation in terms of computational efficiency since it is highly parallelizable. For video sequences of moderate sizes, the proposed method can easily achieve real-time performance with GPU implementation.

The main contributions of this paper are summarized as follows:

- A fast seam carving algorithm, which is based on a novel multi-seam search scheme, is presented for content-aware image resizing.
- The proposed algorithm is designed to be highly parallelizable and suitable for multi-core architecture.
- We extend the proposed fast seam carving algorithm for video retargeting, which can be accomplished in real time with GPU acceleration.

The rest of this paper is organized as follows. We first review the related works in section II. A system overview is presented in section III. The proposed fast seam carving algorithm and the JND-based image saliency measures are introduced in section IV and V, respectively. The extension of the fast seam carving to video carving is described in section VI. Then, the GPU implementation of the proposed algorithm is presented in section VII. Section VIII gives some experimental results with comparison to some other previous competing methods. We conclude this paper in the last section.

II. Related Work

Image resizing is a very useful tool in many applications. Traditionally, it uniformly scales or crops an image to produce a new image of the desired size. Recently, as the demand of changing the aspect ratios for images and videos increases, some adaptive image retargeting methods that non-uniformly scale the image based on the importance of image content have been proposed to enhance human viewing experience.

The idea of incrementally removing or inserting paths for image resizing was firstly proposed by Avidan and Shamir, which is known as the seam carving algorithm [2]. An 8-connected path of least importance pixels is incrementally removed or inserted into an image to be resized. Their algorithm showed good results on images in many cases. However, it may fail to preserve object structure, such as straight lines. Therefore, they improved the seam carving algorithm to find minimal-cost seams by computing the forward energy to reduce the effect of artifacts [4].

In the content-aware approach, warping is a common technique that has been used widely for image resizing and video retargeting. Liu and Gleicher [5] introduced a non-linear, data-dependent scaling for image warping.

In addition, Wolf et al. [1] proposed to automatically detect important regions in the image by combining saliency measure, face detector and motion estimation. It formulates the grid mapping for the image resizing as solving a large linear system of equations. Based on the similar idea, Wang et al. [6] presented a method that allows important regions to be scaled uniformly and homogeneous regions to be distorted. This method shows better capability of preserving important image regions since homogeneous regions can be safely removed or distorted.

In the past few years, we have seen great progress on GPUs. With extensive arithmetic units and large memory bandwidth, the computational power of GPUs nowadays is much higher than that of CPUs. More and more computationally intensive tasks other than traditional computer graphics problems are now migrating from CPUs to GPUs to achieve better time efficiency. Some examples include solving differential equations [7], video coding [8], stereo vision [9], image and signal processing [10], [11]. In this paper, we employ the Compute Unified Device Architecture (CUDA) recently developed by NVIDIA to accomplish the task of real-time video retargeting.

III. System Overview

The algorithmic flow of the proposed method is illustrated in Fig. 2. The proposed system takes a video sequence as

input and processes it frame by frame. To better account for human perception, when a video frame is loaded into main memory (or video memory of a graphics card), it is firstly converted into grayscale before performing the proposed JND-based image enhancement. A saliency map is then computed by applying an edge detection filter to the enhanced image. The proposed fast seam carving technique is utilized to locate multiple seams of minimal energy according to the saliency map. These seams record the pixels to be removed from the current image and also help to determine the seams in the next frame more efficiently. To further reduce the computational cost, an optional image down-sampling operation can be applied first such that all the successive steps are performed at smaller image scales. If image down-sampling is applied, the seams to be removed need to be converted to the original image scale. It is worth noting that many operations involved in the carving-based video retargeting, such as image convolution, can be carried out with parallel computing very efficiently. A modern programmable graphics hardware is thus very suitable for boosting the performance of the proposed system. In section 7, the comparison between the performance of a pure CPU implementation and the GPU accelerated version will be presented.

IV. Proposed Fast Seam Carving Scheme

A. The Seam Carving Method

Seam carving [2] is a simple and effective technique for image resizing. An path from top to down or from left to right is removed or inserted according to the associated cost computed from the image saliency. Thus, the aspect ratio of the image can be adjusted accordingly. Let I be an $n \times m$ image and a vertical seam is defined to be:

$$s^x = \{s_i^x\}_{i=1}^n = \{(x(i), i) | 1 \leq i \leq n, |x(i) - x(i-1)| \leq 1\} \tag{1}$$

where x is a mapping such that $x : [1, \cdots, n] \rightarrow [1, \cdots, m]$. That is, a vertical seam is defined to be an 8-connected path from top to bottom in the image, containing one, and only one, pixel in each row of the image. Similarly, if y is a mapping $y : [1, \cdots, m] \rightarrow [1, \cdots, n]$, then a horizontal seam is:

$$s^y = \{s_j^y\}_{j=1}^m = \{(j, y(j)) | 1 \leq j \leq m, |y(j) - y(j-1)| \leq 1\} \tag{2}$$

The pixels on the path of vertical seam s_i will be:

$$I_s = \{I(s_i) | 1 \leq i \leq n\} = \{I(x(i), i)\}_{i=1}^n \tag{3}$$

Note that removing the pixels of a vertical seam from an image makes all the pixels on the right of the seam shift left to compensate for the missing path.

The strategy of seam carving method is to remove unnoticeable seams. Therefore, it will have the least visual impact after image resizing. This leads to use image saliency to measure the importance of pixels. Seam carving supports several types of saliency measure, such as gradient magnitude, entropy, segmentation, etc. Given a saliency function e, the cost of a seam is defined as:

$$E(s) = \sum_{i=1}^{n} e(I(s_i)) \tag{4}$$

The seam with minimal cumulative cost will be removed. The optimal seam s^* that minimizes the seam cost is:

$$s^* = arg \min_s E(s) = arg \min_s \sum_{i=1}^{n} e(I(s_i)) \tag{5}$$

Dynamic programming is used to find the optimal seam. First, traverse the image from the second row to the last row and compute the cumulative minimum cost M for all possible connected seams for each entry (i, j):

$$M(i, j) = e(i, j) + \min(M(i-1, j-1), M(i-1, j), M(i-1, j+1)) \tag{6}$$

At the end of the process, each entry in the last row of M represents the minimal cumulative cost of a path ending at the corresponding pixel. Hence, backtracking from the pixel with minimal-cost corresponds to the path of the optimal seam. The procedure for finding the minimal-cost horizontal seams is similar to that for the vertical seams and is omitted for brevity.

B. Local Seam versus Global Seam

The optimal seam, which has minimal cumulative cost in the whole image, is referred to as the *global seam*. Removing or inserting an optimal seam introduces the least visual impact on the image. However, to find a global seam and remove it from an image only reduces one pixel in width or height. After such seam is removed from the image, we need to re-compute the cumulative cost array M before determining the next optimal seam. It is too expensive to apply this procedure repeatedly for real-time applications. In Fig. 3, all seams are back-traced after the process of cost accumulation. We observe that seams are not fully chaotic. On the contrary, they tend to converge to several locally-minimal seams. Such seams are called *local seams*. A local seam is defined as

$$s_l = arg \min_{s \in \Omega} E(s) = arg \min_{s \in \Omega} \sum_{i=1}^{n} e(I(s_i)) \tag{7}$$

Ω is a set of seams in the local neighborhood. Local seams have less accumulative costs than those of other seams locally. In the seam carving operation, we remove the global seam one by one to reduce the image size. In fact, we are removing local seams from one area to another. Therefore, if we can manipulate multiple local seams simultaneously, the efficiency of the seam carving procedure can be greatly improved. Note that, Ω in the above equation is not a parameter to be determined. The proposed method will detect all local seams without knowing the size of Ω. (See subsection C.)

Using local seams to improve the efficiency of the seam carving process is practical if the following conditions hold:

1) A number of local seams are present in an image.
2) The overhead to find local seams is light.
3) To process multiple local seams simultaneously does not degrade the perceptual quality.
4) It is applicable to all types of images.

To validate these conditions, we have performed tracing of local seams on a wide variety of images contained in the dataset SIMPLIcity [12], [13]. We observed that all seams

Fig. 3. All seams are traced after the process of cost accumulation for a wide variety of test images. The results show that all seams tend to converge to several local minima.

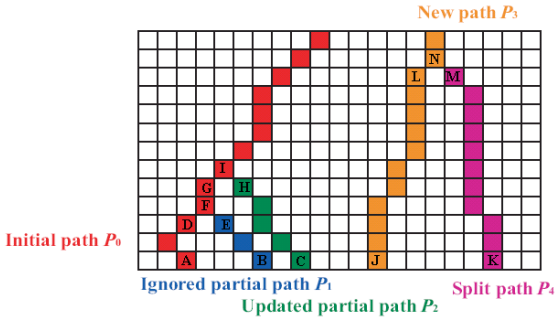


Fig. 4. Multiple seam searching. After finding the first path P_0(A to Q), tracing is restarted from the next entry of the last row in the energy map M. One path with less cumulative energy than P_0 will update P_0 with the partial path, such as P_2 (replace partial path AG with CH). Paths with higher cumulative costs will be discarded, such as P_1 (BE). If it does not meet any pixel of previous seam during the tracing, a new path is created, such as P_3 (JR) and Path P_2 is considered as a local seam. The number of the pixels in the partial path is also checked during the tracing process. Seam split is performed when it is over a threshold. Path KR may split from path JR to either path KMST or path KMSU.

converge to several local seams in all test images. In the subsequent subsections, we present a mult-seam search scheme which locates multiple seams to be removed with low costs, and a JND-based saliency measure which is consistent to human perception to improve the quality of energy estimation required in the seam carving process.

C. Multiple Seam Searching

Finding multiple local seams for removal and insertion reduces the computation involved in the update of the cumulative cost array and the management of internal data structures. We propose a mult-seam search algorithm for fast seam carving, which consists of the following two stages: *seam update* and *seam split*.

Seam Update: Seam update focuses on finding all local seams. Recall that, in the seam carving process, the optimal seam can be found by using dynamic programming. Each entry in the last row of matrix M represents the minimal cumulative cost of a path ending at the corresponding pixel. Seam update traces a path from the first entry in the last row (left-bottom pixel in the image)upward to the top of the image. Sequentially, all pixels are traced from left to right until the right-bottom pixel is completed. During the upward tracing, if the current path meets any pixel of the former path, the total cumulative cost is compared to that of the former path. If it has a higher total energy, this path is abandoned. Otherwise, the path is used to update the former path. Only partial path before the intersection pixel is required for seam update. During the process, it records only one path with the minimal cumulative cost. For any path that does not meet any pixel belonging to the recorded path, this path is created as a new initial path and the seam update starts to process for another set of seams. The currently recorded path is considered as a local seam.

Seam Split: Seam split is adopted to find more disjoint paths to remove or insert. In our observation, different paths may meet at the top of the image but come from entries far apart from the bottom row. Seam split separates paths with just a few common pixels in order to make more disjoint seams. When the current path meets any pixel in the former paths during the tracing process, this path is split from the former if the pixel number of the partial path is over a threshold S. After the splitting, the current path selects pixels among the top, top-left and top-right neighbors with the minimal cost to form a path upward to the top of the image. In the video retargeting applications, the threshold can be used as a speed control parameter because more disjoint seams usually lead to faster operations. The determination of this threshold depends on the requirements of the application. S is set as a ratio of image height when we need to resize image width. In our experiment, S is set to 0.8.

Note that, if only one local seam found in the seam update process, the method degenerates to the original seam carving method. Thus, the seam spilt is proposed to boost the number of disjoint paths. Fig. 4 depicts an example of the seam update and split process. Note that, as in our observation, most paths meet near the bottom of the image. Tracing just a few pixels is sufficient to decide if it is qualified for update. Therefore, it does not introduce much computational overhead in these steps. After all local seams are discovered, K minimal-cost seams are chosen among them to perform seam removal or insertion.

The complexity of saliency measure computation for an m by n image is $O(nm)$. The complexity of dynamic programming to find the optimal seam is $O(nm)$. To trace back the optimal seam is $O(m)$. If p seams are to be removed, the total complexity is $p \times (O(nm) + O(nm) + O(m)) \simeq O(pnm)$. In multi-seam carving scheme, if the average number of local minimal seams to be removed is K, the trace back takes $O(nm)$. Then, the overall complexity is $(p/K) * (O(nm) + O(nm) + O(nm)) \simeq O(pnm/K)$. Theoretically, if n and m are large enough, the computational reduction can approximately reach K times.

V. JND-Based Energy Enhancement

A. Importance Measure

The main idea of seam carving is to find the seams with the minimal costs. Removing or inserting such seams will cause least visual impact on the image, thus yielding good viewing experiences after resizing. To preserve good visual quality and reasonable image content structure after resizing, seam carving relies on the choice of the importance measure. Poor saliency measure usually causes flaws in the resized image, such as edge discontinuity, boundary missing or object distortion. Therefore, the effectiveness of importance measure is the key to achieve satisfactory visual quality.

Since the ultimate receiver of video signals is the human visual system (HVS), the goal of video retargeting should be aimed at the preservation of visually important regions, such as objects or human faces, at certain level of perceptual quality. In this section, we introduce a Just-Noticeable-Distortion (JND) model for the spatio-temporal image enhancement which makes importance measure more consistent with human perception for video carving.

B. JND-Based Spatio-Temporal Enhancement

A JND model is measured based on the observation that human eyes are insensitive to the intensity changes around a pixel below the JND threshold due to the spatial/temporal sensitivity and masking properties [14]. An appropriate JND model can significantly improve the measure of video saliency. To preserve the important areas, we propose a JND-based image enhancement for saliency measure.

The main idea of the proposed JND-based saliency measure is to first apply the JND-based image enhancement, followed by employing a saliency measure on this enhanced image. The local areas that HVS can perceive changes should contribute more to the cost than those areas that HVS can not perceive changes. Less important areas should be adjusted to contribute less to the total cost. Thus, the proposed JND-based importance enhancement is defined as follows:

$$\tilde{R}(x,y,t) = \begin{cases} R(x,y,t)-\rho, & if \quad R(x,y,t)-\bar{R}_B < -\rho \\ \bar{R}_B, & elseif \left|R(x,y,t)-\bar{R}_B\right| \le \rho \\ R(x,y,t)+\rho, & otherwise \end{cases} \tag{8}$$

where $R(x,y,t)$ is the intensity at location (x,y) and time t, ρ stands for $\lambda \cdot JND(x,y,t)$. Note that $JND(x,y,t)$ is the final model that takes luminance, texture and temporal masking into consideration, $\bar{R}_B$ is the average intensity of local background, and λ is a control parameter.

$\tilde{R}(x,y,t)$ contains the adjusted intensity. Take $\lambda = 1$ for example, if the difference between $R(x,y,t)$ and the background is below the JND threshold, HVS can not perceive the difference, and $R(x,y,t)$ is adjusted to the background. On the contrary, if the difference is above the threshold, the difference is further emphasized by adding or subtracting the JND value from $R(x,y,t)$. Thus, if HVS can not sense the difference, the difference will be suppressed. Otherwise, the difference will be enhanced to denote a more important region. After such adjustment, we may apply any other image operator $e(x,y,t)$ to the adjusted image. This leads to a saliency measure that is more consistent with human perception by considering luminance, texture and temporal masking.

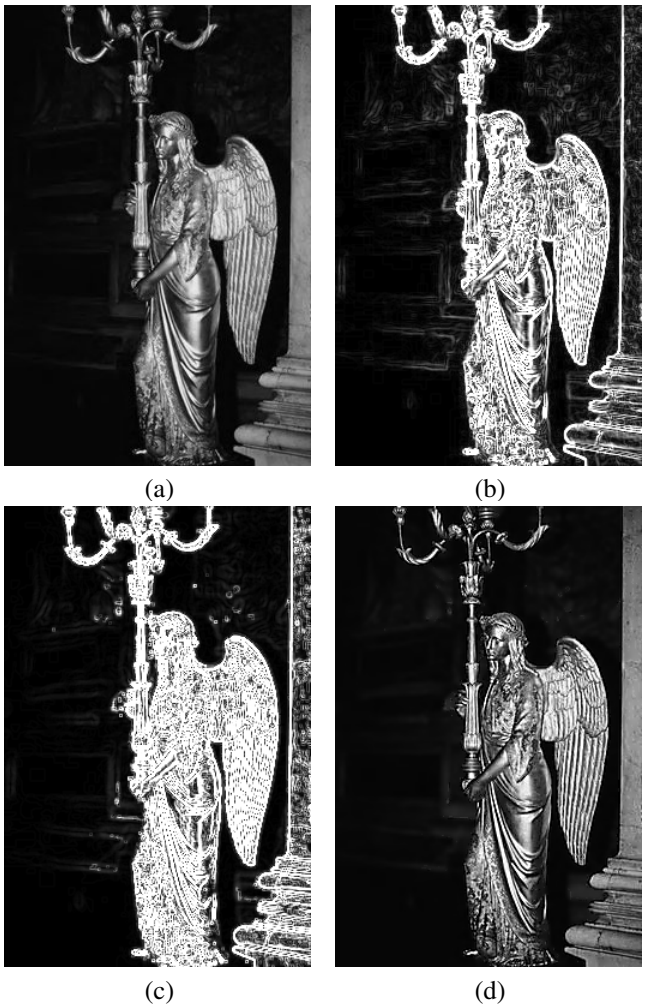


Fig. 5. (a) The source image. (b) The gradient map obtained by Sobel operator. (c) The energy map after applying JND-based importance enhancement scheme. (d) The enhanced image of (a).

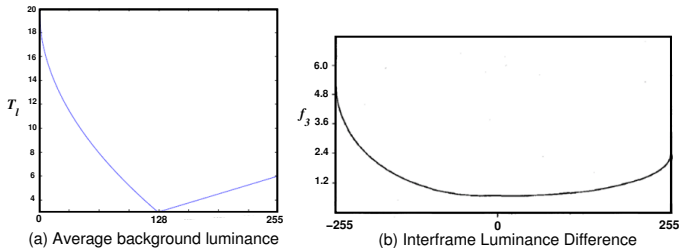
$$e(x,y,t) = \left|\frac{\partial}{\partial x}\tilde{R}(x,y,t)\right| + \left|\frac{\partial}{\partial y}\tilde{R}(x,y,t)\right| \tag{9}$$

Compared to a gradient-based energy map, a JND-based saliency map hides more intensity change that HVS can not perceive in smooth or background areas based on the JND model. Furthermore, it enhances the details, like the edge and object boundary, that HVS can perceive to a certain degree, as shown in Fig. 5.

A JND model which considers both spatial and temporal effects can be denoted as

$$JND\,(x,y,t) = JND_s(x,y) \cdot JND_t(x,y,t) \tag{10}$$

There are two major factors of spatial JND in image domain: 1) background luminance masking, 2) texture masking. In [15] and [16], the relationship between luminance (x,y) and the average background luminance is modeled by a root equation for low background luminance (below 127) while the part over 127 is approximated by a linear function, as illustrated



(a) Average background luminance (b) Interframe Luminance Difference

Fig. 6. (a) Illustration of luminance modeling. T_l is a function of average background luminance. (b) Temporal masking function.

in Fig. 6(a):

$$T_l(x,y) = \begin{cases} 17(1-\sqrt{\frac{\bar{I}(x,y)}{127}})+3, & \text{if } \bar{I}(x,y) \le 127, \\ \frac{3}{128}(\bar{I}(x,y)-127)+3, & \text{otherwise.} \end{cases}$$

$$\bar{I}(x,y) = \frac{1}{32}\sum_{i=1}^{5}\sum_{j=1}^{5} I(x-3+i, y-3+j)\cdot B(i,j) \quad (11)$$

where $T_l(x,y)$ is the visibility thresholds for the luminance masking, $I(x,y)$ is the pixel intensity, and $B(i,j)$ is a weighted low-pass filter.

Texture masking can be determined with local gradients around the pixel. However, edge and non-edge regions should be distinguished for more accurate JND estimation. This is because edge structure attracts more attention from HVS. It is much easier for HVS to detect distortion around edge than that in smooth and textured regions. In [17], $T_t(x,y)$ takes the difference for edge into account:

$$\begin{aligned} T_t(x,y) &= \max_{k=1,2,3,4}\{|grad_k(x,y)|\} \\ grad_k(x,y) &= I(x,y)\otimes g_k(x,y) \end{aligned} \quad (12)$$

where $T_t(x,y)$ denotes the maximal weighted average of gradients around pixel (x,y), $\otimes$ is the convolution operator and $g_k(x,y)$ is the k-th directional high-pass filter.

The combination model in [15] is used to form the spatial JND:

$$JND_s(x,y) = \begin{cases} T_1(x,y), & if \quad T_1(x,y) \ge T_t(x,y) \\ \beta+3, & if \quad T_t(x,y) > T_1(x,y) \end{cases} \quad (13)$$

The minimal luminance JND is 3 according to statistics in Fig. 6(a), and β is a parameter to control the degree of texture magnitude.

From temporal consideration, video motion usually introduces a lot of inter-frame difference. This leads to larger temporal masking. In [18], temporal masking is shown as the empirical curve in Fig. 6(b).

$$JND_t(x,y,t) = f_3(ild(x,y,t)) \quad (14)$$

$$\begin{aligned} &ild(x,y,t) = 0.5\times \\ &(I(x,y,t) - I(x,y,t-1) + \overline{I(x,y,t)} - \overline{I(x,y,t-1)}) \end{aligned} \quad (15)$$

where $ild(x,y,t)$ represents the average inter-frame luminance difference between the t-th and $(t-1)$-th frame. $\overline{I(x,y,t)}$ is the average intensity and is the empirical function defined in Fig. 6(b). In the case of small inter-frame changes, the scaling factor is around its minimum. The scaling factor increases with the increase of inter-frame luminance difference.

VI. Video Carving

The seam carving technique for image resizing can be extended to video carving for video resizing by exploiting the temporal coherence and the importance measures of seams. In our fast seam carving scheme, K minimal-cost local seams are chosen for the first frame. The temporal coherence can be consistently maintained as follows. The k-th vertical local seam in frame $t-1$, $k=1,\cdots,K$, is denoted by ${}^kS^x(t-1)$, which is defined by

$$\begin{aligned} &{}^ks^x(t-1) = \\ &\{(x^k(i), i, t-1) \mid 1 \le i \le n, |x(i)-x(i-1)| \le 1\} \end{aligned} \quad (16)$$

where $x^k(i)$ is the x-coordinate of i-th pixel in the k-th local seam. Three candidate seams in frame t for local seam k is defined as follows:

$$\begin{aligned} s^{x_0}(t) &= \{(x^k(i)-1, i, t-1)\}_{i=1}^n \\ s^{x_1}(t) &=^k S^x(t-1) \\ s^{x_2}(t) &= \{(x^k(i)+1, i, t-1)\}_{i=1}^n \end{aligned} \quad (17)$$

The best seam ${}^kS^x$ in frame t is chosen to be the one that minimizes this seam cost among the three candidate seams, i.e.

$${}^kS^x(t) = \min_{s\in\{s^{x_0}(t), s^{x_1}(t), s^{x_2}(t)\}} \left(\sum_{i=1}^{n} e\left(I(S_i)\right)\right) \quad (18)$$

Local seam k and all its extensions in temporal space form a surface in the 3D video cube, as depicted in the right most of Fig. 2. as depicted in the right most of Fig. 2. After picking the best K local seams, the carving process removes K seams and their surfaces to reduce K in width. Applying this carving process recursively yields a resized video. The expanding problem is similar to the reducing problem. A detailed description is omitted for brevity.

We can, therefore, design an online system for video carving. This means we do not have to build a system to calculate the temporal seams for the whole shot. Instead, an order of seam removal is maintained. After resizing the current frame to the target size, choosing the best candidate for each seam can be processed according to the order. Note that, for efficiency, not the entire frame needs the computation of the spatio-temporal JND saliency cost. Instead, it is computed only at the pixels on the candidate seams.

VII. Video Retargeting on GPU

In this section, we introduce the GPU implementation of our video retargeting system with CUDA. Under the CUDA architecture, GPUs are designed to be a single instruction multiple data (SIMD), multi-threaded manycore processor, which integrates the vertex and pixel shader into a unified computing unit and provides a software environment that enables programmers without strong knowledge of graphics concepts to utilize them as a coprocessor of CPU. In CUDA programming, parallelism is achieved by partitioning the computation task into finer sub-problems that can be solved independently in

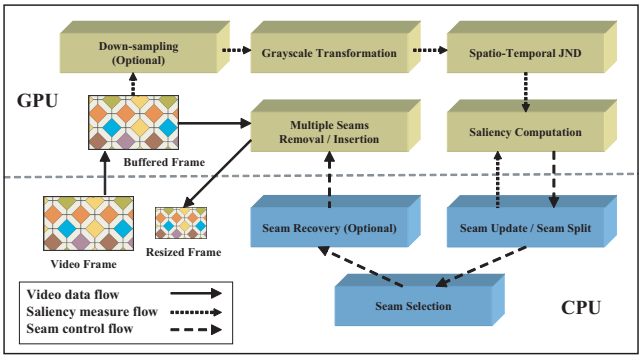


Fig. 7. Block diagram of the proposed system.

parallel by the so-called *kernels*. CUDA kernel functions are executed by threads organized into *thread blocks*, which are then dispatched to one of the multiprocessors of GPU.

Fig. 7 illustrates the system components of the proposed system, which also indicates the components that have been implemented on GPU. Some system components that present data dependency and more complex control flow are accomplished on CPU. For example, the determination of local seams requires the partial results of previous seams, which is carried out sequentially on CPU. As shown in Fig. 7, the input frames are buffered in video memory on GPU and resized. This refers to video data flow. For the buffered data on GPU memory, parallel operations are performed. These are included in saliency measure flow. Then, we perform the multi-seam search on CPU side and instruct the GPU to perform the image resizing operations to remove the seams of least cost. This refers to seam control flow. The previous results are utilized to accomplish the resizing process of subsequent frames.

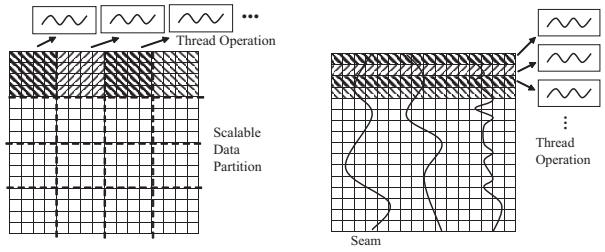


Fig. 8. Two types of parallelization operations: image convolution operation (left) and image resizing operation (right).

The parallelization in our system contains two types of operations: *image convolution operation* and *image resizing operation*. The convolution-based operations, such as grayscale translation, JND-based image enhancement and saliency estimation, mostly involves in applying a convolution mask to each of the image pixels to form a new pixel value. In our CPU implementation, each video frame is partitioned into a number of disjoint blocks that are independently processed. Under the CUDA environment, these pixel-wise computation can be more efficiently realized by creating a thread for each single pixel. Fig. 8 depicts the parallelization of the image convolution operation. The image resizing operation compacts a video frame in horizontal or vertical direction once a seam is ready for removal. The main operation of image resizing involves in shifting the pixels lying on one side of the removed pixel forward or backward to compact the fragmentation of the image, which can only be realized sequentially for each row of column of pixels (see Fig. 8).

VIII. Experimental Results

We conduct several experiments for comparing the proposed algorithm to the original seam carving [2] and the warping-based algorithm by Wolf et al. [1]. In our implementation, the parameter K for fast seam carving is 5, $\beta = 2.5$ in spatial JND and $\lambda = 0.5$ in JND-based enhancement. We apply image down-sampling by a factor of 4 in our experiments. Note that different high-level image saliency measures can be included to improve the saliency measure, such as face detection and motion estimation used in [1]. For a fair comparison of computational efficiency, we do not add these components to the method by Wolf et al. [1] for efficiency comparison.

We have implemented the proposed method in two versions: pure CPU-based and GPU accelerated. To validate the impact of parallel processing, the CPU-based program is multi-threaded and tested on a single- and dual-core Intel PC, respectively. The GPU-based implementation is accomplished on the same single-core PC equipped with a GeForce 8800GT GPU and 512 MB video memory. Without specifically mentioned, all experimental results reported in this paper are gathered on the single-core platform. Most system components implemented on the GPU, such as grayscale conversion and JND-based image enhancement, are boosted in terms of time efficiency by over 10 times compared with their counterparts in CPU-based implementation. The bottleneck of our current system is the resizing of each video frame, which usually consumes over 60% of the overall execution time in contrast to $6\% \sim 8\%$ of the JND computation and $10\% \sim 12\%$ of the edge detector convolution. This is because the compaction of an image after removing a vertical or horizontal seam is only row-wise or column-wise parallelizable but not pixel-wise. Besides, the GPU also suffers from the fact that the memory transactions of moving data to fill the removed seam are hardly *coalesced* such that its Single-Instruction-Multiple-Data (SIMD) feature can not be fully utilized. We believe that a more sophisticated implementation may overcome this bottleneck by *virtually* resizing the images with a mask indicating the new positions of surviving pixels.

A. Progressive Image Resizing

We first test the performance of the three resizing methods with progressive image retargeting. This experiment progressively resizes image from 640×512 to 512×409, 512×409 to 409×327, 409×327 to 327×216, 327×216 to 216×208, 216×208 to 208×166, 208×166 to 166×132, 166×132 to 132×105, and 132×105 to 105×84, respectively. Fig. 9 shows the computational performance of progressive image resizing of the original seam carving, and the proposed method. The execution time of the original seam carving ranges from 1.97s to 0.094s, whereas the proposed method ranges from 0.53s to 0.046s in a CPU-based implementation.

TABLE I
PERFORMANCE OF RETARGETING VARIOUS TEST VIDEO SEQUENCES.

	Akiyo(176x144)		Nemo(320x240)		Panda(400x300)		Panda(512x384)		Sport(608x336)	
	88x144	117x144	160x240	213x240	200x300	267x300	256x384	342x384	304x336	405x336
Seam Carving (sec)	18.91	12.81	53.91	35.78	89.17	60.29	151.1	101.04	166.31	113.33
Proposed (sec)	7.55	4.97	13.94	9.14	20.94	12.67	31.22	20.23	29.11	19.75
Proposed+Dual (sec)	5.74	3.72	10.63	7.08	16.25	9.94	23.99	15.24	28.2	18.63
Proposed+GPU (sec)	1.14	0.81	3.05	2.2	5.34	3.81	9.03	6.66	9.36	6.81
Proposed+GPU (fps)	87.72	123.46	32.79	45.45	18.73	26.25	11.07	15.02	10.68	14.68

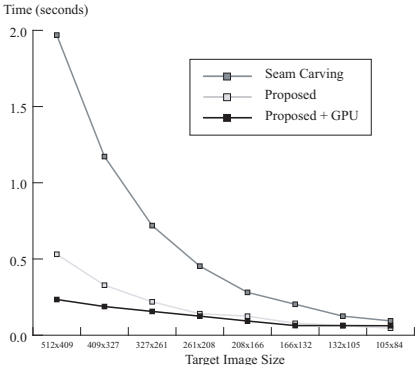


Fig. 9. Progressive image resizing.

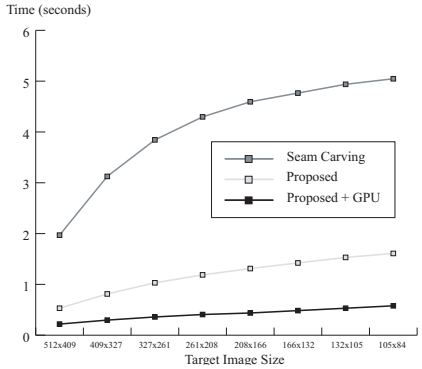


Fig. 10. Computation time of one-step image resizing.

With GPU acceleration, the computational time can be improved to the range from 0.234s to 0.062s. It is obvious that the proposed algorithm significantly improves the performance of the original seam carving algorithm. For large-size images, the GPU implementation can speed up more than twice. The execution time of the warping-based method ranges from 42.08s to 0.58s, which are evaluated by the Matlab program of our implementation. Although the development tool and implementation details may affect the time efficiency, the warping-based method is generally slower than carving-based methods since it involves in solving a large and sparse linear system.

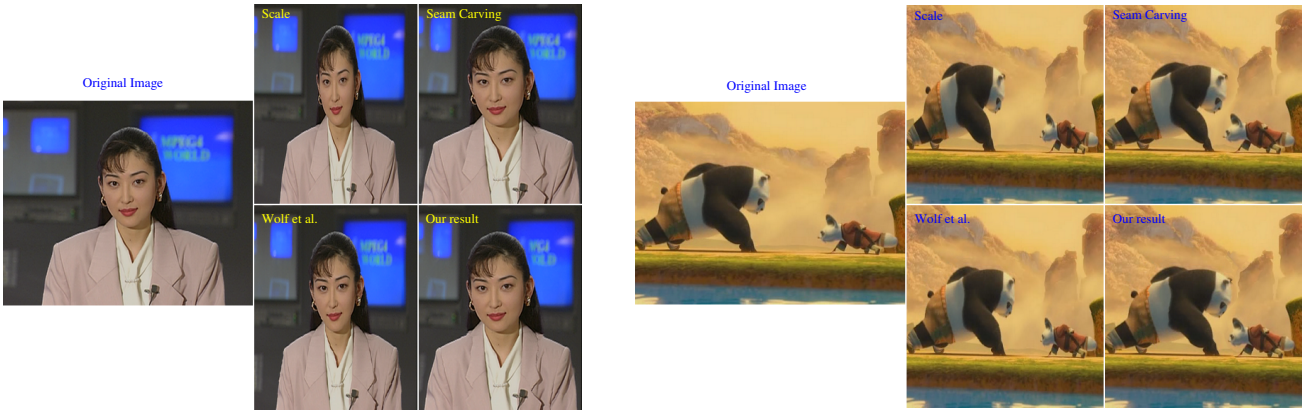
B. One-Step Image Resizing

In our experiment of one-step image resizing, a 640×512 image is resized to 512×409, 409×327, 327×216, 216×208, 208×166, 166×132, 132×105 and 105×84 directly, as illustrated in Fig. 10. The execution time of the original seam carving method ranges from 1.97s to 5.05s, whereas the proposed method is from 0.53s to 1.61s. With GPU implementation, the execution time is improved to be from 0.22s to 0.58s. The warping-based method took about 42s to resize the test image to different resolutions. Since the size of the sparse linear system is determined by the original image size, it takes the same computational effort no matter what the target size is. One can easily see that the seam carving based methods are much more efficient than the warping-based method, especially for small-scale image resizing. The proposed method benefits in time efficiency with the aid of GPU implementation. However, it is not straightforward for warp-based method to exploit parallel computing.

C. Video Retargeting

For the execution time of video retargeting, we compare our algorithm with the image-based seam carving scheme. Experiments on a dual-core PC platform are also examined to compare with the results of GPU implementation. Table I shows the results of the execution time of different sequences which all contain 100 frames. For most sequences, the proposed algorithm is about four times faster than the original seam carving for video retargeting. The performance on dual-core is about 1 to 2 second faster than the same method on a single-core platform. The overall performance in the GPU implementation is further improved to be 2 to 4 times faster than the same algorithm on a dual-core platform. For a video of size 320×240, to reduce to a half size in one direction, it can operate at 32.79 frames per second (fps). For an video of size 400×300, to reduce the width by one-third also can be accomplished in real-time with 26.25 fps. According to the results shown in Rubinstein's work [4], the precalculation of video with resolution 400×300 and 400 frames takes about 10 to 20 minutes for 50 to 150 percent resizing. This means the computational performance of the work in [4] is at most 0.7 fps.

Fig. 11(a) shows the results from the sequence "Akiyo". In the original method by Wolf et al. [1], they combine face detection and motion estimation to compensate for the shortage of the gradient saliency measure. Without additional high-level image cues used in [1], the proposed method maintains better foreground object shape, like human face and wider body area. In the proposed method, we did not use other high-level image saliency measures. With the JND-based importance enhancement, foreground objects can be well retained, thus making the whole frame more consistent with HVS. Fig. 11(b) shows a frame from "Kung Fu Panda" and its resized images. With the proposed method, the distance between two animals



(a) (b)

Fig. 11. Video retargeting results of (a) "Akiyo" and (b) "Kung Fu Panda". For both examples, the top-left images are the result of bicubic interpolation. The bottom-left images are the results of Wolf et al [1]. The image-based seam carving results are depicted in the top-right and the results of the proposed fast video carving algorithm are shown in the bottom-right.

shrinks and larger foreground object are well kept in the frame than those obtained by using the bicubic interpolation and the algorithm by Wolf et al [1]. Note that, the image-based seam carving works well for image resizing but creates serious artifacts without considering the temporal coherence in video retargeting. The work by Rubinstein et al. [4] improved the quality of video retargeting, but their algorithm is not efficient for real-time applications. Please see the video carving process and more comparisons in the supplemental material available at http://cv.cs.nthu.edu.tw/research/Video/CSVT-demo.wmv.

IX. Conclusion

To summarize, we present a fast seam carving algorithm for real-time video retargeting in this paper, which contains the following three contributions. Firstly, a fast seam carving algorithm that exploits local minimal property of multiple local seams is proposed to greatly speedup the original algorithm. Secondly, we introduce a JND model to enhance important pixels in an image based on HVS. Thirdly, we extend the proposed fast seam carving algorithm to video retargeting by considering temporal coherence.

We demonstrated that the proposed algorithm is highly parallelizable and can be further improved with a GPU implementation. Although we adapted GPU to evaluate the performance of the proposed method in this paper, we believe that the proposed method is also very suitable to be used on other embedded systems. The highly parallelizable image resizing operator and data independent property of our method can benefit from the SIMD functionality found in many of the existing embedded platforms. There are many possible extensions based on this work. Since the warping-based and seam-based approaches have their own advantages, we would like to explore the possibility of combining warping, seam carving and scaling appropriately to achieve better image/video resizing in the future.

References

[1] L. Wolf, M. Guttmann, and D. Cohen-Or, "Non-homogeneous content-driven video-retargeting," in *Proceedings of International Conference on Computer Vision (ICCV'07)*, 2007.

[2] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM Trans. Graph.*, vol. 26, no. 3, p. 10, 2007.

[3] Owens, D. John, Luebke, David, Govindaraju, Naga, Harris, Mark, Kruger, Jens, Lefohn, E. Aaron, Purcell, and J. Timothy, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007.

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

[5] F. Liu and M. Gleicher, "Automatic image retargeting with fisheye-view warping," in *Proceedings of ACM symposium on User interface software and technology*, 2005, pp. 153–162.

[6] Y.-S. Wang, C.-L. Tai, O. Sorkine, and T.-Y. Lee, "Optimized scale-and-stretch for image resizing," *ACM SIGGRAPH*, pp. 1–8, 2008.

[7] N. Goodnight, C. Woolley, G. Lewin, D. Luebke, and G. Humphreys, "A multigrid solver for boundary value problems using programmable graphics hardware," in *Proc. of Graphics hardware*, 2003.

[8] G. Shen, G.-P. Gao, S. Li, H.-Y. Shum, and Y.-Q. Zhang, "Accelerate video decoding with generic gpu," *IEEE Trans. on CSVT*, vol. 15, pp. 685–693, 2005.

[9] M. Gong and Y.-H. Yang, "Near real-time reliable stereo matching using programmable graphics hardware," in *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 924–931.

[10] K. Moreland and E. Angel, "The fft on a gpu," in *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 2003, pp. 112–119.

[11] Y. Luo and R. Duraiswami, "Canny edge detection on nvidia cuda," in *Proc. of CVPR Workshops*, 2008, pp. 1–8.

[12] J. Li and J. Z. Wang, "Automatic linguistic indexing of pictures by a statistical modeling approach," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1075–1088, 2003.

[13] J. Z. Wang and G. Wiederhold, "Simplicity: Semantics-sensitive integrated matching for picture libraries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 947–963, 2001.

[14] N. S. Jayant, J. D. Johnston, and R. J. Safranek, "Signal compression based on models of human perception," in *Proc. of the IEEE*, 1993.

[15] C.-H. Chou and Y.-C. Li, "A perceptually tuned subband image coder based on the measure of just-noticeable-distortion profile," *IEEE Trans. on CSVT*, pp. 467–476, 1995.

[16] R. J. Safrenek and J. D. Johnson, "A perceptually tuned sub-band image coder with image dependent quantization and post quantization data compression," in *Proc. of Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP89)*, 1989, pp. 1945–1948.

[17] X. Yang, W. Lin, Z. Lu, E. P. Ong, and S. Yao, "Motion-compensated residue pre-processing in video coding based on just-noticeable-distortion profile," *IEEE Trans. on CSVT*, vol. 15, pp. 742–750, 2005.

[18] C.-H. Chou and C.-W. Chen, "A perceptually optimized 3-d subband image codec for video communication over wireless channels," *IEEE Trans. on CSVT*, vol. 6, pp. 143–156, 1996.