Improved Seam Carving for Video Retargeting

Michael Rubinstein
Mitsubishi Electric Research Lab, Cambridge

Ariel Shamir
The Interdisciplinary Center, Herzliya

Shai Avidan
Adobe Systems Inc.

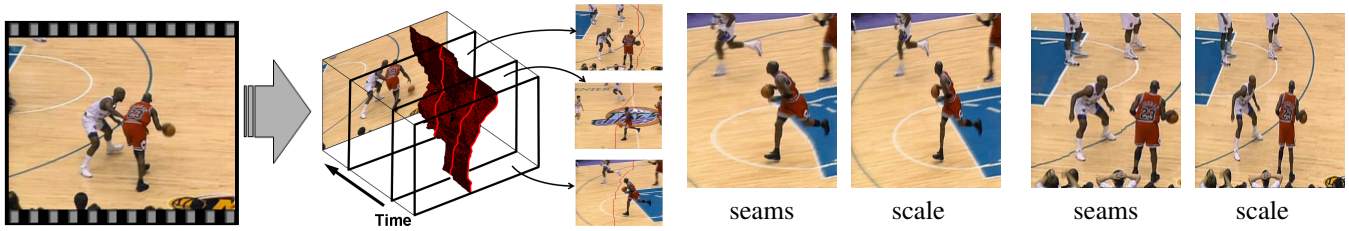

Figure 1: Improved seam carving for video sequences combines the frames of the video to form a 3D cube and finds 2D monotonic and connected manifold seams using graph cuts. The intersection of the manifolds with each frame defines the seams on the frame. The manifolds are found using a new forward-energy criterion that reduces both spatial and temporal artifacts considerably.

Abstract

Video, like images, should support content aware resizing. We present video retargeting using an improved seam carving operator. Instead of removing 1D seams from 2D images we remove 2D seam manifolds from 3D space-time volumes. To achieve this we replace the dynamic programming method of seam carving with graph cuts that are suitable for 3D volumes. In the new formulation, a seam is given by a minimal cut in the graph and we show how to construct a graph such that the resulting cut is a valid seam. That is, the cut is monotonic and connected. In addition, we present a novel energy criterion that improves the visual quality of the retargeted images and videos. The original seam carving operator is focused on removing seams with the least amount of energy, ignoring energy that is introduced into the images and video by applying the operator. To counter this, the new criterion is looking forward in time - removing seams that introduce the least amount of energy into the retargeted result. We show how to encode the improved criterion into graph cuts (for images and video) as well as dynamic programming (for images). We apply our technique to images and videos and present results of various applications.

CR Categories: I.3.0 [Computing Methodologies]: Computer Graphics—General; I.2.10 [Computing Methodologies]: Vision and Scene Understanding—Video Analysis; I.4.9 [Computing Methodologies]: Image Processing and Computer Vision—Applications

Keywords: Video retargeting, Video editing, Image retargeting, Seam carving, Forward energy

ACM Reference Format
Rubinstein, M., Shamir, A., Avidan, S. 2008. Improved Seam Carving for Video Retargeting. *ACM Trans. Graph. 27*, 3, Article 16 (August 2008), 9 pages. DOI = 10.1145/1360612.1360615 http://doi.acm.org/10.1145/1360612.1360615.

1 Introduction

Seam carving is an effective technique for content aware image retargeting. In a similar manner, video should support retargeting capabilities as it is displayed on TVs, computers, cellular phones and numerous other devices. A naive extension of seam carving to video is to treat each video frame as an image and resize it independently. This creates jittery artifacts due to the lack of temporal coherency, and a global approach is required. The approach we take is to treat video as a 3D cube and extend seam carving from 1D paths on 2D images, to 2D manifolds in a 3D volume (Figure 1). Nevertheless, because we need to build a 2D connected manifold through space-time volume, the dynamic programming approach used for image resizing is no longer applicable. In this paper we define a new formulation of seam carving using graph cuts. However, a simple cut cannot define a valid seam. A seam must be *monotonic*, including one and only one pixel in each row (or column), and *connected*. We show how to define a graph whose cut creates a monotonic and connected seam, which is equivalent to the one created by dynamic programming on images. Using this formulation, we extend seam carving to video and define a monotonic and connected 2D manifold seam inside the video cube. We also discuss a multiresolution approach to speed up the computation time of seams for video.

Seam carving also has other limitations. On images, where salient spatial structures appear, seam carving can create serious artifacts. This is magnified in video, where spatial artifacts can be amplified, and augmented by temporal ones. In fact, because of human perception, the latter may even be more disturbing in video, as the human eye is highly sensitive to movement. To address this problem, we define a novel seam carving criterion that better protects salient spatial, as well as temporal content. This improves the visual quality of the retargeted images and videos considerably. The new criterion takes into account the energy *inserted* into the image or video during retargeting, not just the energy removed from it. We show how to encode the new criterion into both the dynamic programming and the graph cut solutions.

The difficulties imposed by video resizing using seam carving can therefore be characterized as algorithmic, dimensional and cardinal. The algorithmic difficulty follows from the fact that we cannot extend the original dynamic programming method to a 3D video cube. Dimensional difficulties originate from the additional, temporal, dimension of a video, which enhances spatial artifacts and introduces new ones involving motion. Cardinal difficulties stem from the fact that a video is a sequence of frames, and hence any processing of

a video sequence involves larger amounts of data. This paper addresses these difficulties and presents results for video resizing applications such as size reduction and expansion, multi-size videos for interactive size manipulation and object removal.

2 Background

The increasing need to adapt content to various displays caused a surge in the number of publications dealing with image, as well as video, retargeting.

Attention models, based on human spatiotemporal perception have been used to detect Regions Of Interest (ROIs) in image and video. The ROIs are then used to define "display paths" ([Wang et al. 2004b]) to be used on devices in which the display size is smaller than the video (or image) size. The least important content of the video is cropped, leaving the important features in larger scale, essentially creating a zoom-in-like effect ([Fan et al. 2003]). Virtual camera motions or pseudo zoom-in/out effects are used to present the content in a visually pleasing manner.

A similar system was proposed by [Liu and Gleicher 2006], where both cropping and scaling are used together with virtual camera motion to mimic the process of adapting wide screen feature films and DVDs to standard TV resolution. Their system minimizes information loss based on image saliency, object saliency and detected objects (e.g. faces). Cropping, however, discards considerable amounts of information and might be problematic, for instance, if important features are located at distant parts of the image or frame, which is common in wide or over-the-shoulder shots in videos.

An alternative approach is to segment the image into background and foreground layers, scale each one of them independently and then recombine them to produce the retargeted image. This was first proposed by [Setlur et al. 2005] for non-photorealistic retargeting of images and later extended to video by [Tao et al. 2007]. While this is an appealing approach, it relies crucially on the quality of segmentation - a difficult and complicated task in itself. For video, [Pritch et al. 2008] propose an "object-based" approach to webcam synopsis, where they segment the input video into objects and activities, rather than frames. Then they compose a short video synopsis, in response to user query. Their work only deals with retiming the video, not changing its spatial extent.

Recently, [Wolf et al. 2007] presented a system to retarget video that uses non-uniform global warping. They concentrate on defining an effective saliency map for videos that comprises of spatial edges, face detection and motion detection. Results are shown mainly for reducing video size. Our work differs since we take a discrete approach and we also show results for video expansion, object removal, and introduce multisize videos, which are not supported by their system. We mostly use image edge energies but also show results using their saliency map.

We build on and extend the work of [Avidan and Shamir 2007]. They proposed seam carving for image retargeting and used dynamic programming to find the optimal seam iteratively. We propose a graph based approach to seam carving, allowing us to handle video retargeting. This extension defines 2D surfaces to be removed from the 3D video cube. An alternative approach is to map these 2D manifolds to frames in a new video sequence [Rav-Acha et al. 2007]. This approach, termed Evolving Time Fronts, gives users the ability to manipulate time in dynamic video scenes.

Graph partitioning and graph-based energy minimization techniques are widely used in image and video processing applications such as image restoration, image segmentation, object recognition and shape reconstruction. A graph representing an image, together with some constraints, is partitioned into disjoint subsets by connecting pixels or voxels based on their similarity. Traditionally, similarity is defined by some variation of intensity change or gradients. For videos, it is often convenient to consider the sequence of frames as a 3D space-time volume [Kwatra et al. 2003; Schödl et al. 2000; Wang et al. 2004a; Wang et al. 2005]. In such cases, the extension of energy minimization from 2D images to 3D space-time video is usually straightforward. We are influenced by [Kwatra et al. 2003], that use graph cuts to seamlessly patch two 2D or 3D textures. However, there are differences in the way we construct the graph, and the terminal nodes in our method are placed differently than in theirs. The challenge we face is in designing a graph that produces only admissible cuts, that is, cuts that are monotonic so that only one pixel is removed from every row and are connected. As we will show, standard graph cut based construction do not satisfy these constraints and new ones must be defined.

Figure 2: Seam carving on each video frame independently creates locally optimal seams that can be totally different over time. This creates a jittery resized video. In this example we show the first ten seams removed. A similar illustration is shown in the accompanied video.

3 Preliminaries

A seam is a monotonic and connected path of pixels going from the top of the image to the bottom, or from left to right. By removing one seam from an image, the image size is reduced by one either in the horizontal or the vertical dimension. Seam carving uses an energy function defined on the pixels and successively removes minimum energy paths from the image. In video, we search for a resizing operator in the granularity of shots (i.e. a sequence of frames where the camera shoots continuously). Simply applying the seam carving operator separately to each frame of the video introduces serious artifacts (Figure 2).

Alternatively, one can search for regions in the image plane that are of low importance in *all* video frames. This is done by computing the energy function on every image independently and then taking the maximum energy value at each pixel location, thus reducing the problem back to image retargeting. We call the seams computed this way *static* seams, because they do not change along frames. Specifically, given a video sequence $\{I_t\}_{t=1}^{N}$ we extend the spatial L_1-norm to a spatiotemporal L_1-norm:

$$\begin{aligned} E_{\text{spatial}}(i,j) &= \max_{t=1}^{N}\{|\frac{\partial}{\partial x}I_t(i,j)| + |\frac{\partial}{\partial y}I_t(i,j)|\} \\ E_{\text{temporal}}(i,j) &= \max_{t=1}^{N}\{|\frac{\partial}{\partial t}I_t(i,j)|\} \\ E_{\text{global}}(i,j) &= \alpha \cdot E_{spatial} + (1-\alpha)E_{temporal} \end{aligned}$$

Essentially, this measure can be seen as a (maximum) projection of the spatial L_1-norm to 2D, where $\alpha \in [0,1]$ serves as a parameter that balances spatial and temporal contribution. In practice, since motion artifacts are more noticeable, it is good to bias the energy toward temporal importance, taking $\alpha = 0.3$. We use a maximum projection and not average to be conservative in the cost calculation. Figure 3 shows examples for the global energy map and static seams removal from videos.

Figure 3: Static seams for the golf video and ape animation. The global energy function is shown using color mapping from violet (low) to red (high). The actual static seams are shown for the golf sequence at the top. Some representative resized frames are also shown for both videos (example results can be seen in the accompanied video).

The main appeal of such a static method is its simplicity and speed. It gives good results when the video is created by a stationary camera, and the foreground and background are separated (Figure 3). However, in more complex video scenes where the camera is moving or when multiple motions are present, seams must be allowed to adapt over time.

Towards this end, we define a video seam as a connected 2D manifold "surface" in space-time that cuts through the video 3D cube. The intersection of the surface with each frame defines one seam in this frame. Hence, removing this manifold removes, in effect, one seam from each video frame. On the one hand, because the surface is flexible, the seams can change adaptively over time in each frame (Figure 1). On the other hand, because the surface is connected, the seams preserve temporal coherency. Unfortunately, there is no simple extension of the dynamic programming algorithm of 2D images to a 3D space-time volume, and we must employ another algorithm, namely graph cut.

4 Seam Carving using Graph Cuts

We first discuss a formulation of the seam carving operator as a minimum cost graph cut problem on images and then extend the discussion to video. We will further assume that we are searching for vertical seams in the image. For horizontal seams all constructions are the same with the appropriate rotation. We refer to graph edges as *arcs* to distinguish them from *edges* in the image. We construct a grid-like graph from the image in which every node represents a pixel, and connects to its neighboring pixels. Virtual terminal nodes, S (source) and T (sink) are created and connected with infinite weight arcs to all pixels of the leftmost and rightmost columns of the image respectively.

An S/T *cut* (or simply a *cut*) C on such a graph is defined as a partitioning of the nodes in the graph into two disjoint subsets S and T such that $s \in S$ and $t \in T$. The *cost* of a cut $C = \{S, T\}$ is defined as the sum of the cost of the 'boundary' arcs (p, q) where $p \in S$ and $q \in T$. Note that a cut cost is *directed* as it sums up the weights of directed arcs specifically from S to T. That is, arcs in the opposite direction do not affect the cost. To define a seam from a cut, we consistently choose the pixels to the left of the cut arcs. The optimal seam is defined by the *minimum cut* which is the cut that has the minimum cost among all valid cuts.

Converting dynamic programming to graph cuts was already done in the past for the purpose of texture synthesis [Kwatra et al. 2003]. However, there is a crucial difference between our work and theirs. The reason is that a general cut does not define a valid seam for seam-carving, as it must satisfy two constraints:

Monotonicity the seam must include one and only one pixel in each row (or column for horizontal seams).

Connectivity the pixels of the seams must be connected.

More formally, a vertical seam can be thought of as a (discrete) mapping $S\colon Y \times T \rightarrow X$ (where $T = \{0\}$ for images) from (row, time) to column. The monotonicity constraint requires this mapping to be a *function*, while the connectivity constraint forces this function to be *continuous*. Hence, the challenge is to construct a graph that guarantees the resulting cut will be a continuous function over the relevant domain.

4.1 Graph Cuts for Images

In a standard grid graph construction, every internal node $p_{i,j}$ is connected to its four neighbors $Nbr(p_{i,j}) = \{p_{i-1,j}, p_{i+1,j}, p_{i,j-1}, p_{i,j+1}\}$. Following the L_1-norm gradient magnitude E_1 energy that was used in [Avidan and Shamir 2007], we define the weight of arcs as the forward difference between the corresponding pixels in the image either in the horizontal direction: $\partial x(i,j) = |I(i,j+1) - I(i,j)|$ or in the vertical: $\partial y(i,j) = |I(i+1,j) - I(i,j)|$. Under this formulation, Figure 4(a) shows an optimal partition of the waterfall image into source and target parts. This cut does not satisfy the seam carving constraints.

To impose the monotonicity constraint on a cut, we use different weights for the different directions of the horizontal arcs. For forward arcs (in the direction from S to T), we use the weight as defined above, but for backward arcs we use infinite weight. Appendix A gives the proof why the monotonicity constraint is maintained under this construction (Figure 4(b)).

The main difference between this graph cut construction and the original dynamic programming approach is that there is no explicit constraint on the cut to create a *connected path*. The cut can pass through several consecutive vertical arcs, in effect creating a piecewise-connected seam. Although this behavior is penalized as more vertical arcs are cut, it does happen in practice. Our empirical results show that connected seams are important to preserve both spatial and temporal continuity and to minimize visual artifacts. To constrain cuts to be connected we use infinite weight diagonal arcs going "backwards". Using similar arguments, Appendix A shows why this construction imposes the connectivity constraint.

In fact, by combining the weights of the vertical and horizontal arcs together, we can create a graph whose cut will define a seam that is equivalent to the one found by the original dynamic programming algorithm. For example, we assign the weight $E_1(i,j) = \partial x(i,j) + \partial y(i,j)$ to the horizontal forward arc and remove the vertical arc altogether (Figure 4(c)). A cut in this graph is monotonic and connected. It consists of only horizontal forward arcs (the rest are infinite weight arcs that pose the constraints and cannot be cut), hence its cost is the sum of $E_1(i,j)$ for all seam pixels, which is exactly the cost of the seam in the original seam carving operator. Because both algorithms guarantee optimality, they must have the same cost, and (assuming all seams have different costs) the seams must be the same.

This suggests we can use any energy function defined on the pixels as the weight of the forward horizontal arcs and achieve the same results as the original dynamic programming based seam carving. Moreover, high level functions such as a face detector [Viola and Jones 2004], or a weight mask scribbled by the user, can be used

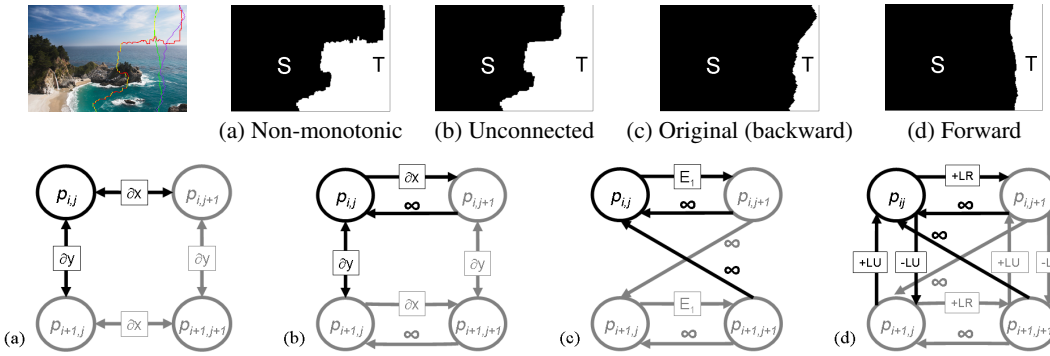

Figure 4: Minimum cut on the waterfall image (top left) for various graph constructions. The seam is composed of the pixels to the left of the cut. The different graph constructions are illustrated by four nodes representing four pixels in the image. The actual image graph is created by tiling these sub-graphs across the image (see text for details). Graph (a) creates a general path and not a valid seam, while (b) creates a monotonic but piecewise-connected seam. The construction at (c) is equivalent to the original seam carving algorithm (with E_1). The construction at (d) represents the new forward energy we present in Section 5.

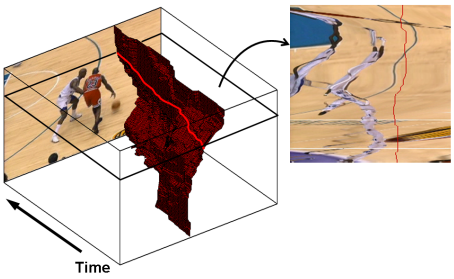

Figure 5: The intersection of every $X \times T$ plane with the seam surface defines a spatiotemporal seam.

in any of the graph constructions we present. We simply add the pixel's energy to the horizontal arc going out of the pixel.

4.2 Graph Cuts for Video

The extension to video is straightforward. Assuming we are searching for a vertical seam, we consider the $X \times T$ planes in the video cube and use the same graph construction as in $X \times Y$ including backward diagonal infinity arcs for connectivity. We connect the source and sink nodes to all left and right (top/bottom in the horizontal case) columns of all frames respectively. A partitioning of the 3D video volume to source and sink using graph cut will define a manifold inside the 3D domain (Figure 5). Such a cut will also be monotonic in time because of the horizontal constraints in each frame that are already in place. This cut is globally optimal in the cube both in space and time. Restricted to each frame, the cut defines a 1D connected seam.

The graph cut algorithm runs in polynomial time, but in practice was observed to have linear running time on average [Boykov and Kolmogorov 2004]. For the full video volume, the computation time depends on the number of nodes times the number of arcs in the graph, which is quadratic in the number of voxels. Solving minimal cut on a graph in which every voxel is represented by a node is simply not feasible. In fact, performance issues are encountered already for high resolution images. To improve efficiency, we employ a banded multiresolution method, similar to the one described in [Lombaert et al. 2005]. An approximate minimal cut is first computed on the coarsest graph, and then iteratively refined at higher resolutions. Coarsening is performed by sampling the graph both

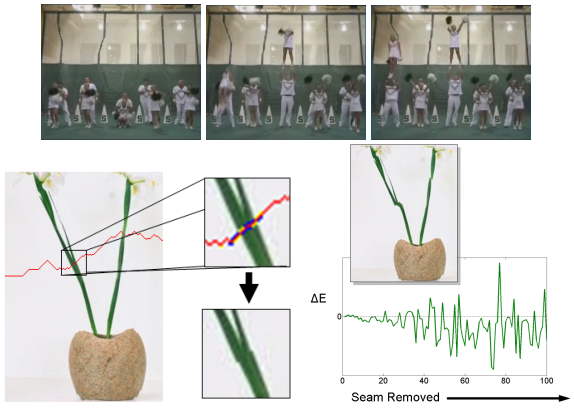

Figure 6: The artifacts seen in video retargeting (top) can also be seen on a static vase image (bottom). We show an example of the change in energy after a specific seam is removed (a). In some pixels (blue) energy is reduced and in others (yellow) increased. This seam inserts more energy to the image than removes, creating a step artifact in the stem of the flower. The actual change in energy ΔE after each seam removal is shown in (b).

spatially and temporally, while refinement is done by computing graph cut on a narrow band induced by the cut that was computed at the coarser level. The band in our case takes the form of a "sleeve" cutting through the spatiotemporal volume.

The graph cut approach to seam carving allows us to extend the benefits of content-aware resizing to video. Still, the method is not perfect and no single energy function was shown to perform properly in all cases [Avidan and Shamir 2007]. Therefore, we introduce a new energy function that better protects media content, and improves video results.

5 Forward Energy

The artifacts created in video frames can actually be seen on static images as well (Figure 6). They are created because the original algorithm chooses to remove the seam with the least amount of energy from the image, ignoring energy that is *inserted* into the re-

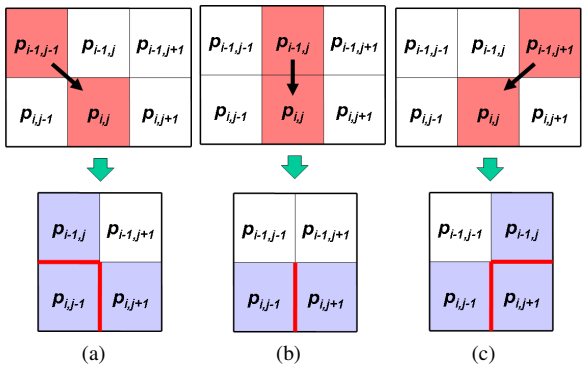

Figure 7: Calculating the three possible vertical seam step costs for pixel $p_{i,j}$ using forward energy. After removing the seam, new neighbors (in gray) and new pixel edges (in red) are created. In each case the cost is defined by the forward difference in the newly created pixel edges. Note that the new edges created in row $i-1$ were accounted for in the cost of the previous row pixel.

targeted image. The inserted energy is due to new edges created by previously non adjacent pixels that become neighbors once the seam is removed (see e.g. the steps artifacts in Figure 6(a)). Assume we resize an image $I = I_{t=1}$ using k seam removals ($t = 1 \dots k$). To measure the real change in energy after a removal of a seam, we measure the difference in the energy of the image after the removal ($I_{t=i+1}$) and the energy of only those parts that were not removed in the previous image $I_{t=i}$ (i.e. the image energy $E(I_{t=i})$ minus the seam energy). In our new graph cut formulation, the energy of the image is no longer an attribute of the pixels, but rather an attribute of the arcs in the graph. Hence, the energy of an image $E(I)$ is given by the sum of all finite arcs of its induced graph, and the energy of a seam $E(C)$ is simply the cost of the cut C. The energy difference after the i^{th} seam carving operation is:

$$\Delta E_{t=i+1} = E(I_{t=i+1}) - [E(I_{t=i}) - E(C_i)] \qquad (1)$$

As can be seen in Figure 6(b), ΔE_t can actually increase as well as decrease for different seam removals using the original seam carving approach (the energy measured in this case is E_1). The figure also shows a specific example of a seam that inserts more energy to the image than it removes.

Following these observations, we propose a new criterion for choosing the optimal seam. The new criterion looks *forward* at the resulting image instead of backward at the image before removing the seam. At each step, we search for the seam whose removal inserts the minimal amount of energy into the image. These are seams that are not necessarily minimal in their energy, but will leave less artifacts in the resulting image, after removal. This coincides with the assumption that natural images are piece-wise smooth intensity surfaces, which is a popular assumption in the literature. We will show how to define forward energy on images and then discuss the extension to video.

As the removal of a connected seam affects the image, and its energy, only at a local neighborhood, it suffices to examine a small local region near the removed pixel. We consider the energy introduced by removing a certain pixel to be the new "pixel-edges" created in the image. The cost of these pixel edges is measured as the forward differences between the pixels that become new neighbors, after the seam is removed. Depending on the direction of the seam, three such cases are possible (see Figure 7).

5.1 Forward Energy in Dynamic Programming

For each of the three possible cases, we define a cost respectively:

$$(a)\ C_L(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j-1)|$$
$$(b)\ C_U(i,j) = |I(i,j+1) - I(i,j-1)|$$
$$(c)\ C_R(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j+1)|$$

We use these costs in a new accumulative cost matrix M to calculate the seams using dynamic programming. For vertical seams, each cost $M(i,j)$ is updated using the following rule:

$$M(i,j) = P(i,j) + \min \begin{cases} M(i-1,j-1) + C_L(i,j) \\ M(i-1,j) + C_U(i,j), \\ M(i-1,j+1) + C_R(i,j) \end{cases} \qquad (2)$$

where $P(i,j)$ is an additional pixel based energy measure, such as the result of high level tasks (e.g. face detector) or user supplied weight, that can be used on top of the forward energy cost.

5.2 Forward Energy in Graph Cut

To define the forward energy cost in graph cut, we need to create a graph whose arc weights will define the cost of the pixel removal according to the three possible seam directions. Figure 4(d) illustrates this construction. A new horizontal pixel-edge $p_{i,j-1}p_{i,j+1}$ is created in all three cases because $p_{i,j}$ is removed. Hence, we assign the difference between the **L**eft and **R**ight neighbors $+LR = |I(i,j+1) - I(i,j-1)|$ to the graph arc between the nodes representing $p_{i,j}$ and $p_{i,j+1}$. To maintain the seam monotonicity constraint as before, we connect $p_{i,j+1}$ and $p_{i,j}$ with a (backward) infinite weight arc. We also add diagonal backward infinite arcs to preserve connectivity.

Next, we need to account for the energy inserted by the new vertical pixel-edges. In the case of a vertical seam step (Figure 7(b)), there are no new vertical edges so no energy is inserted. From the corollary in appendix A we have that all nodes to the left of the cut must be labeled S and all nodes to the right of the cut must be labeled T. By definition, the cost of a cut will only consider arcs directed from nodes labeled S to nodes labeled T. It therefore follows that only upward vertical arcs will be counted in right-oriented cuts (Figure 7(a)), and only downward vertical arcs will be counted in left-oriented cuts (Figure 7(c)). Hence, we assign the difference between the **L**eft and **U**p neighbors $+LU = |I(i-1,j) - I(i,j-1)|$ to the upward vertical arc between $p_{i,j}$ and $p_{i-1,j}$, and the weight $-LU = |I(i+1,j) - I(i,j-1)|$ to the downward vertical arc between $p_{i,j}$ and $p_{i+1,j}$ ($-LU$ means the difference between the **L**eft and **U**p neighbors with respect to the end point of the arrow).

Figure 8 illustrates the difference between removing seams using the original algorithm with E_1, and removing seams using the new forward energy we propose. In the original cost map the cost is increased with every crossing of a bar in the bench, as it defines an edge in the image. This drives the seams to the image sides while creating disturbing artifacts. In the improved criterion, vertical seams can intersect the bars without inserting energy to the image, resulting in almost no increase in the cost map in these areas and a more plausible result. More examples are given in Figure 9 and in the supplemental material. Figures 10 and 13 show some frames from video sequences retargeted with graph cuts using the improved forward energy.

For video, we examine slices in the 3D video-cube depending on the seam direction. For vertical seams (Y-direction), the intersection of every slice on the ($X \times T$) dimension with the seam creates a seam on that plane (Figure 5). We use the same formulation in ($X \times T$)

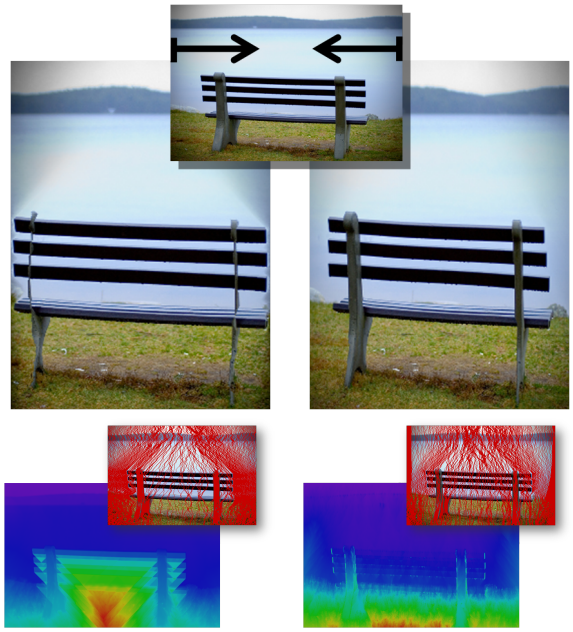

Figure 8: Comparison between the original seam carving backward energy (left) and the new forward energy (right) for resizing an image (original shown in small at the top). At the bottom are the respective cost maps M of both techniques and the seams removed from the image. The new results suffer much less from the artifacts generated using backward energy such as the difference in water color and the distortions of the bench bars and skeleton.

as we did in $(X \times Y)$. Hence, we define the cost of every pixel removal as the new *temporal* pixel-edges created between frames in the temporal direction, that are introduced to the video when this pixel is removed. We then create arcs between nodes in the graph between time-steps with the appropriate costs exactly as in the spatial $X \times Y$ domain.

6 Results

In the accompanied video, we present results for aspect ratio changes of videos by removing, as well as inserting seams (see also Figure 10 and Figure 13). We also support multisize videos for interactive resizing (Figure 11, top, and the accompanied video). We extend the method suggested by [Avidan and Shamir 2007] of precomputing seam index maps for images, to each frame in the video. As we cannot hold the entire index structure in memory, these maps are stored on disk, and are loaded on demand before the frame is displayed.

As discussed, we also support other energy functions for retargeting. For example, Figure 12 shows the result of our method on the football video using the saliency map of [Wolf et al. 2007]. Our system also supports other energy functions such as object detectors and manually inserted weights. As our approach is global, the algorithm is relatively robust to cases in which the energy function is not given for every frame, and to occasional false positive or false negative detections. An example using face detector is shown in the accompanied video. By marking pixels with positive weights, the user can protect certain parts of a video during the retargeting process. The user need not mark every frame, but only once every k frames (in practice we use $k \simeq 10$). By supplying negative weights, the user can also attract seams to desired parts of the video, for example, for object removal (Figure 11).

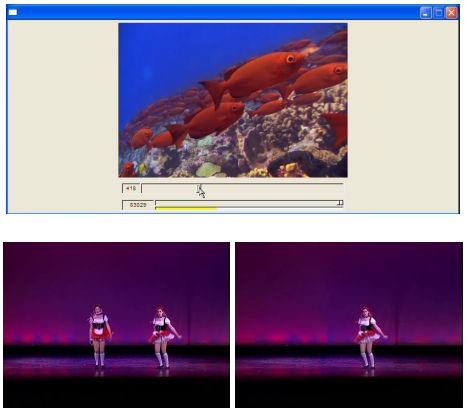

Figure 11: A snapshot of the multisize video interface is shown at the top. After pre-computation, the user is able to resize the video interactively while it plays. Below, on the left, is a frame from the dancers video. On the right is the corresponding frame from the video in which the left dancer was removed using user markings. Actual results can be viewed in the accompanied video.

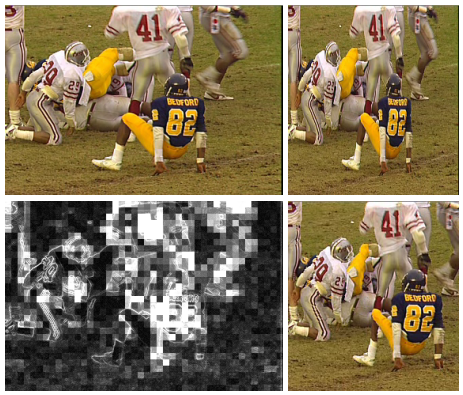

Figure 12: Retargeting using given energy (saliency) function. In the left column a frame from the football video is shown, and underneath is its saliency map. On the right column, on top is the rescaled frame, and at the bottom the retargeted frame.

Using our multiresolution graph cut technique, computation times for retargeting videos are still significant. Precalculating multisize videos that enable between 50 to 150 percent change in aspect ratio takes 10 to 20 minutes on average. Typical videos have a resolution of 400×300 and 400 frames. We used a 1.8 GHz dual core laptop with 2GB memory. The memory consumption for such videos averages 300MB, which is reasonable for this kind of processing. The running time of forward energy dynamic programming on images is compatible to backward energy.

7 Limitations

The forward energy criteria we propose is designed to protect the structure of media. However, maintaining the structure can sometime come at the expense of content. For example, important objects that can be resized without noticeable artifacts (i.e. inserted energy) may be jeopardized during resizing. In such cases, a combination of the forward criteria with E_1 energy can help to achieve better results. This is because E_1 can better protect content. There are other situations on video and images where forward energy fails

Figure 9: Several comparisons between the original seam carving algorithm (left image of pairs) and forward energy (right image of pairs). At the top the car image (first on the left) was first condensed and then extended. Note how the artifacts on the car and background building are greatly reduced. In the middle, both height and width reductions are shown. At the bottom, one frame from a video resizing is given for comparison. The sequence can be seen in the accompanied video.

Figure 10: Examples of video retargeting. Top row, an original frame. In the following rows we show a rescaled frame on the left and a retargeted one on the right.

Figure 13: Each row shows a different frame from a 100 frames long video sequence. From left to right, the original image, a scaled down image, a targeted down image, a scaled up image and targeted up image.

Figure 14: Cases where forward energy can fail. On the upper left, a snapshot from vertical resizing of a bicycle video sequence is shown, together with a zoom in on the bicycle rider. The bicycles are shrunk as the algorithm abstains from cutting the textured rocks. On the upper right, a grainy background texture is considered as important content, while the matchbox is distorted. On the second row, a frame from the highway video is shown with its corresponding frame from the retargeted video. Forward energy fails to achieve plausible result in this case due to the nature in which the camera and objects are moving. The actual result can be viewed in the accompanied video.

to achieve plausible results. Some are illustrated in Figure 14. In general, due to motion and camera movement, the problem of video resizing is more challenging than image resizing. To solve some of those challenges, it may be better to revert to other methods of resizing such as scaling or cropping or combine them together with seam carving. Lastly, our current method runs on the video in batch mode. In contrast, online techniques could also support resizing while streaming the video.

8 Conclusions and Future Work

We propose an improved seam carving operator for image and video retargeting. Video retargeting is achieved using graph cuts and we have shown a construction that is consistent with the dynamic programming approach. Furthermore, we offered new insight into the original seam carving operator and proposed a forward-looking energy function that measures the effect of seam carving on the *retargeted* image, not the original one. We have shown how the new measure can be used in either graph cut or dynamic programming and demonstrated the effectiveness of our contributions on several images and video sequences.

We have outlined some future extensions in the Limitation section. Also, by switching to graph cut based representation we could rely on some advances to speed up computations. For example, [Kohli and Torr 2007] proposed a method for computing minimum cuts on an updated graph, which can hopefully yield speed gains of up to two orders of magnitude.

Our methods can also be adapted to resize videos *temporally*. By rotating the video cube to $Y \times T$ view, we can find seam manifolds that cut through the temporal domain. Each manifold, when removed, will decrease the length of the video by one, thus resulting in a shorter video. A similar method was recently proposed also by [Chen and Sen 2008]. They too use graph cuts for finding low gradient sheets to remove. A basic difference between their method and ours is that they remove an approximation to the minimal energy surface, while our method guarantees optimality under the seam constraints (Section 4.2). Their graph construction is similar to the one described in Figure 4(a), which yields non-monotonic and unconnected cuts. Moreover, they counter the cardinality problem by splitting the input video into smaller pieces and removing one frame at a time from each piece. By using a multiresolution scheme, we target a more global solution. We are currently experimenting with the application of our method for video summarization.

Finally, another future issue we plan to investigate is the relationship between seam carving, scaling and cropping. These all address the problem of fitting content to display, but take different approaches to solve it. It would be interesting to try and combine all three into a single framework.

Acknowledgements

We thank Mistubishi Electric Research Labs (MERL) for their support of this research. Shamir and Avidan were researchers at MERL when this work began. We would like to thank Matthew Brand for his comment on energy inserted to the image during the retargeting process, that led to the formulation of forward energy. We would like to thank Fatih Porikli and Michael Jones for useful discussions on optical flow, and Raphael Pelossof for general feedback. We would also like to thank Frédo Durand and the computer graphics group at CSAIL for reviewing this work, and the SIGGRAPH reviewers for their comments.
We thank Angelo Garcia for narrating our video. We thank Mammoth HD library (www.mammothhd.com) for allowing us to use their royalty free demo reel (road ski, water ski, kayak, fish, bicycle, nature). We thank Wolf et al. for letting us use their video samples (basketball, football) and saliency maps (football). We thank the members of the following communities for publicly sharing their media: youtube (www.youtube.com): Nmbr5 (golf). blip.tv (http://blip.tv): Detroit Free Press (cheerleaders), aaron (cheerleader shaky camera), mindcaster (ape animation), Mike Krumlauf

(highway), cuecast (interview). stage6 (stage6.com): dancers, Osaka hall image. flickr (www.flickr.com): Ben McLeod (bench), Thomas Hawk (rain). Other images were borrowed from Avidan and Shamir (waterfall, car, vase, umbrella, matches, snow). The footage from RATATOUILLE is courtesy of Disney/Pixar. The SIGGRAPH evolve sample was taken from the ACM SIGGRAPH 2008 demo video.

References

AVIDAN, S., AND SHAMIR, A. 2007. Seam carving for content-aware image resizing. *ACM Trans. Graph. 26*, 3, 10.

BOYKOV, Y., AND KOLMOGOROV, V. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 9, 1124–1137.

CHEN, B., AND SEN, P. 2008. Video carving. In *Short Papers Proceedings of Eurographics*.

FAN, X., XIE, X., ZHOU, H.-Q., AND MA, W.-Y. 2003. Looking into video frames on small displays. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, ACM, 247–250.

KOHLI, P., AND TORR, P. H. S. 2007. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI) 29*, 12, 2079–2088.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3, 277–286.

LIU, F., AND GLEICHER, M. 2006. Video retargeting: automating pan and scan. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, ACM, 241–250.

LOMBAERT, H., SUN, Y., GRADY, L., AND XU, C. 2005. A multilevel banded graph cuts method for fast image segmentation. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)*, vol. 1, 259–265.

PRITCH, Y., RAV-ACHA, A., AND PELEG, S. 2008. Nonchronological video synopsis and indexing. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, to appear.

RAV-ACHA, A., PRITCH, Y., LISCHINSKI, D., AND PELEG, S. 2007. Dynamosaicing: Mosaicing of dynamic scenes. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI) 29*, 10, 1789–1801.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 489–498.

SETLUR, V., TAKAGI, S., RASKAR, R., GLEICHER, M., AND GOOCH, B. 2005. Automatic image retargeting. In *In the Mobile and Ubiquitous Multimedia (MUM)*, ACM Press.

TAO, C., JIA, J., AND SUN, H. 2007. Active window oriented dynamic video retargeting. In *Proceedings of the Workshop on Dynamical Vision, ICCV 2007*.

VIOLA, P., AND JONES, M. J. 2004. Robust real-time face detection. *Int. J. Comput. Vision 57*, 2, 137–154.

WANG, J., XU, Y., SHUM, H.-Y., AND COHEN, M. F. 2004. Video tooning. *ACM Trans. Graph. 23*, 3, 574–583.

WANG, J., REINDERS, M., LAGENDIJK, R., LINDENBERG, J., AND KANKANHALLI, M. 2004. Video content presentation on tiny devices. In *IEEE International Conference on Multimedia and Expo (ICME)*, vol. 3, 1711–1714.

WANG, J., BHAT, P., COLBURN, R. A., AGRAWALA, M., AND COHEN, M. F. 2005. Interactive video cutout. *ACM Trans. Graph. 24*, 3, 585–594.

WOLF, L., GUTTMANN, M., AND COHEN-OR, D. 2007. Non-homogeneous content-driven video-retargeting. In *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV '07)*, 1–6.

A Seam Constraints Proof

We show that the graph construction introduced in section 4 using horizontal backward infinite arcs induces a minimal cut which necessarily maintains monotonicity.

The optimal cut must pass all rows: This follows directly from the definition of a cut and from the construction. As S is connected to all pixels in the leftmost column, and every pixel in the rightmost column is connected to T, every row has to be cut in some place in order to create disjoint subsets.

The optimal cut passes each row only once: W.l.g. assume that there exists a row j in the grid in which the cut passes twice (in fact it must then cut the row an odd number of times). Let us examine two consecutive cuts in row j. Let node $p_{i,j}$ be labeled S, the nodes $p_{i+1,j}$ to $p_{k-1,j}$ will be labeled T and the nodes $p_{k,j}$ will be labeled S again. However, this also means that the arc $p_{k,j} \rightarrow p_{k-1,j}$, which is an infinite weight arc, must be included in the cut (figure 15(a)). This makes it an infinite cost cut, which contradicts optimality since it is always possible to cut only horizontal arcs at some column of the grid and achieve a finite cost cut.

Corollary: if the source node is connected to the left column of the image and the target node to the right column, then all nodes on the left of the minimal cut must be labeled S, and all nodes on the right of the cut must be labeled T.

If we want the cut to be connected as well (as shown in Figure 4(c-d)), we use backward-going diagonal arcs. The same argument as above can prove connectivity as illustrated in Figure 15(b-c).

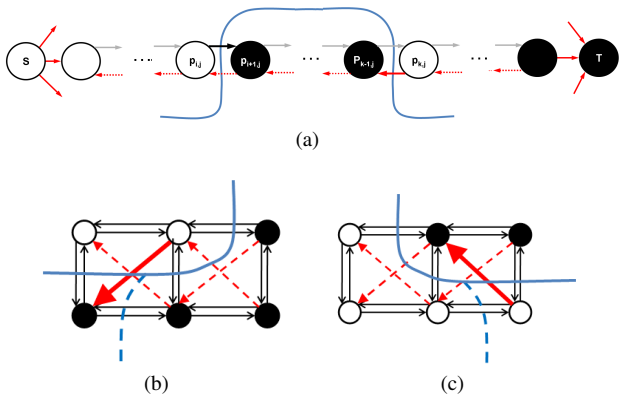

Figure 15: Using infinity edges (red) in the graph construction maintains the seam constraints. Horizontal infinity arcs maintain monotonicity (a) - see details in text. Diagonal infinity arcs maintain connectivity. If the cut skips more than one pixel to the left (b) or right (c) - a diagonal infinity arc from a source node (white) to a target node (black) must be cut.