DOI:10.1145/1435417.1435437

Seam Carving for Media Retargeting

By Ariel Shamir and Shai Avidan

Abstract
Traditional image resizing techniques are oblivious to the content of the image when changing its width or height. In contrast, media (i.e., image and video) retargeting takes content into account. For example, one would like to change the aspect ratio of a video without making human figures look too fat or too skinny, or change the size of an image by automatically removing "unnecessary" portions while keeping the "important" features intact. We propose a simple operator; we term *seam carving* to support image and video retargeting. A seam is an optimal 1D path of pixels in an image, or a 2D manifold in a video cube, going from top to bottom, or left to right. Optimality is defined by minimizing an energy function that assigns costs to pixels. We show that computing a seam reduces to a dynamic programming problem for images and a graph min-cut search for video. We demonstrate that several image and video operations, such as aspect ratio correction, size change, and object removal, can be recast as a successive operation of the seam carving operator.

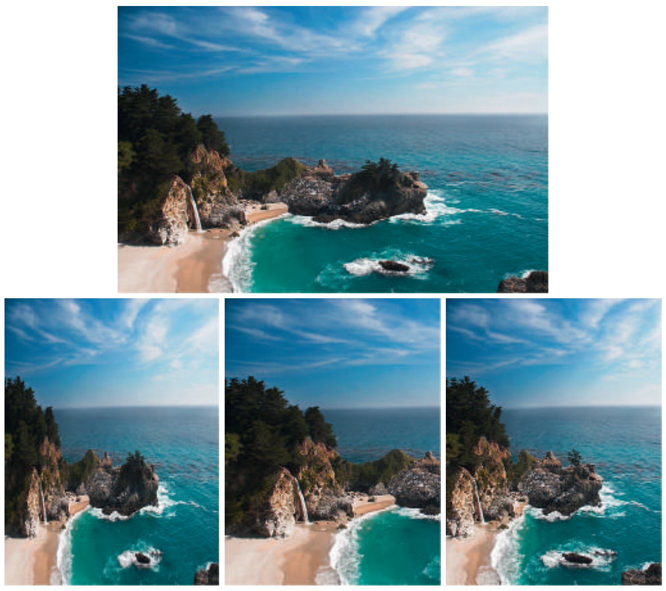
1. INTRODUCTION

The diversity and versatility of display devices today imposes new demands on digital media. Designers must create different alternatives for web-content and design different layouts, possibly even dynamic layouts, for different devices. Nevertheless, images and video, although being key elements in digital media, typically remain rigid in size and shape and cannot deform to fit different displays. To accommodate high definition TVs, computer screens, PDAs, and cell phones not only the resolution, but often the aspect ratio of the media must be adjusted. Standard image scaling is not sufficient since it is oblivious to the image content and typically can be applied only uniformly. Cropping is limited since it can only remove pixels from the image periphery and cannot support expansion (Figure 1). More effective resizing can only be achieved by considering the image content and not only geometric constraints. We propose a simple operator, we term *seam carving*, which can change the size of images and video by gracefully carving out or inserting pixels in different parts of the media. Seam carving is based on the definition of an energy function defining the importance of pixels. In images, a seam is a *connected* path of low energy pixels crossing the image from top to bottom (or from left to right), and is *monotonic*, that is, including one, and only one, pixel in each row (or column). In video, a seam is a monotonic and connected low energy 2D manifolds passing through a 3D volume cube defined by stacking the video frames (Figure 2).

By successively removing or inserting seams we can reduce, as well as enlarge, the size of media in both directions. For size reduction, seam selection ensures that we remove more of the low energy pixels and fewer of the high energy ones from the source, while also taking care to insert the least amount of energy to the target. For enlarging, the order of seam insertion ensures a balance between the original media content and the artificially inserted pixels. These operators produce, in effect, a *content-aware* resizing of media.

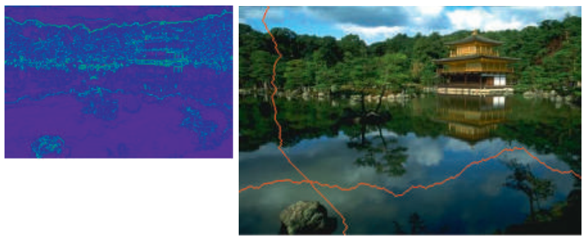
On images, the search for an optimal seam can be formulated using an efficient dynamic programming algorithm, which is linear in the number of pixels. For videos the dynamic programming approach is no longer applicable. We show how to define a graph such that running the min-cut/max-flow algorithm on it will create a cut that is a 2D

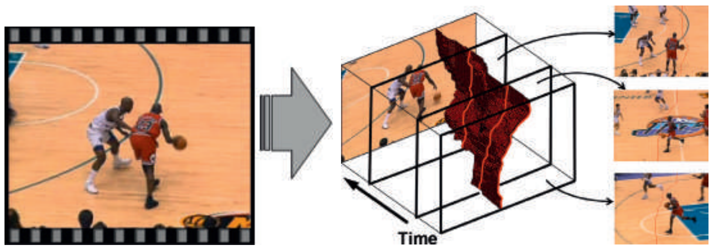
Figure 1: Content aware vs. content oblivious resizing: scaling (left) creates distortion, cropping (middle) removes important parts, while seam carving (right) preserves the image content.



This paper is a summary of two papers originally published in SIGGRAPH Proceedings: Seam carving for content-aware image resizing. *ACM Trans. Graph. 26*, 3 (2007), 10; Improved seam carving for video retargeting. *ACM Trans. Graph. 27*, 3 (2008), 10.

Figure 2: A seam is a connected and monotonic path of low energy pixels (energy shown in small) connecting the two sides of the image either horizontally or vertically (shown in red). On video (right) the sequence of frames is combined to a 3D cube and a 2D monotonic and connected manifold is sought. The intersection of the manifold with each video frame defines the seams on the frame.





manifold. The intersection of this manifold with every video frame produces a valid seam on the frame. When applied to images this graph-cut formulation is equivalent to the one created by dynamic programming.

We illustrate the application of seam carving and insertion for aspect ratio change, media retargeting, and object removal. Furthermore, by storing the order of seam removal and insertion operations, we define *multi-size* images and video that support on-the-fly media retargeting at very high frame rates. Multisize media allows a designer to create retargeted media once and then let client applications retarget the media to various displays in real time.

Seam carving can support a whole range of image energy functions, while also reducing the created artifacts by looking forward at energy inserted into the resized media. To allow interactive control, we also provide a scribble-based user interface for adding or reducing the energy of a pixel or region and guide the desired results. This tool can also be used for object removal and for authoring multisize images.

2. BACKGROUND

Media resizing is a standard tool in many image and video editing tools. It works by uniformly resizing the media to a target size. Recently, there has been a growing interest in media retargeting that is driven largely by the growing number of mobile devices used to view digital content.[4, 12–14, 18, 22, 29] These seek to change the size of the image or video while maintaining the important features intact. The common approach for all media resizing works is first to define an importance map on the pixels of the media, and then use this map to guide some operator that reduces (or enlarges) the media size.

There are numerous ways to define importance in media. Some use top-down methods such as face detectors[23] or foreground-background segmentation,[20] while others use bottom-up methods that rely on visual saliency,[9] the detection of Regions Of Interest (ROI),[13, 18, 25] or motion.[29] Similarly, numerous operators are used to change the media size in a content-aware fashion. These include cropping,[7, 18] uniform scaling,[13] virtual camera motions,[7] and recently nonuniform warping.[8, 27, 29] More recently, some new approaches to media retargeting were introduced by Simakov[21] and Wei[28] where retargeting is formulated as the problem of patch similarity between source and target images. For a more thorough overview of previous work the reader is referred to Avidan[2] and Rubinsteing.[17]

In this work we present the novel seam carving operator for resizing. We mostly use image and video edge energies as the importance map, but can support any saliency map defined on pixels. We show results for reduction as well as expansion of images and video. We demonstrate object removal, and also define the notion of multisize media.

The use of seams for image editing and composition is prevalent.[1, 10, 16] However, none of these methods discuss the problem of image retargeting and none of them consider, as we do, computing a seam on a single image. Computing a seam can be done in a variety of ways, including Dijkstra's shortest path algorithm,[5] dynamic programming[6] that we use for images, or graph cuts[3] that we use for video.

3. THE OPERATOR

For ease of exposition, we will focus on images first, then we will show how to extend the seam carving operator to handle video. We start by presenting the operator in one direction allowing a user to change either the width or the height of an image. We then show how to optimally change both width and height of a given image. Next, we investigate the use of a different objective function that leads to visually more pleasing results. We follow with an extension to video retargeting based on a graph-cut formulation and finally describe *multisize media*, which is a new representation for images and video.

3.1. The one directional case

Our approach to content-aware resizing is to remove pixels in a judicious manner. Therefore, the question is how to choose the pixels to be removed? Intuitively, our goal is to remove unnoticeable pixels that blend with their surroundings. Since human vision is more sensitive to edges, we give high value to edges using the following simple energy function (see Figure 2, later, we elaborate on the definition of energy functions in Section 3.3):

$$e_1(\mathbf{I}) = \left|\frac{\partial}{\partial x}\mathbf{I}\right| + \left|\frac{\partial}{\partial y}\mathbf{I}\right| \qquad (1)$$

Given this, or similar, energy function, assume we need to reduce the width of an image. One can think of several strategies to achieve this. For instance, an optimal strategy to preserve energy (i.e., keep pixels with high energy value) would be to remove the pixels with lowest energy in ascending order. This destroys the rectangular shape of the image, because we may remove a different number of pixels from each row (see Figure 3(e)). If we want to prevent the image from breaking we can remove an equal number of low energy pixels from every row. This preserves the rectangular shape of the image but destroys the image content by creating a zigzag effect (Figure 3(d)). To preserve both the shape and the visual coherence of the image we can use auto-cropping. That is, look for a subwindow, the size of the target image, that contains the highest energy (Figure 3(a)). Another possible strategy somewhat between removing pixels and cropping is to remove whole columns with the lowest energy. Still, artifacts might appear in the resulting image (Figure 3(b)). Therefore, we need a resizing operator that will be less restrictive than cropping or column removal, but can preserve the image content better than single pixel removals. This leads to our strategy of seam carving (Figure 3(c)).

Formally, let **I** be an $n \times m$ image and define an image *vertical seam* to be

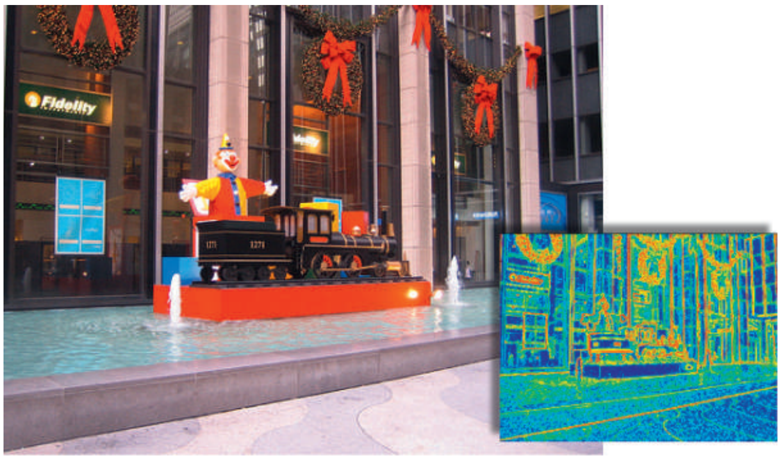
$$\mathbf{s}^{\mathrm{x}} = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, \text{ s.t. } \forall i, |x(i) - x(i-1)| \le 1 \quad (2)$$

where x is a function $x: [1, \ldots, n] \rightarrow [1, \ldots, m]$ that is continuous in a discrete sense (two consecutive values of the function do not differ by more than 1). In other words, a vertical seam is a connected path of pixels in the image from top to bottom, containing one, and only one, pixel in each row of the image (see Figure 2). Similarly, if y is a discretely continuous function $y: [1, \ldots, m] \rightarrow [1, \ldots, n]$, then an image *horizontal seam* is

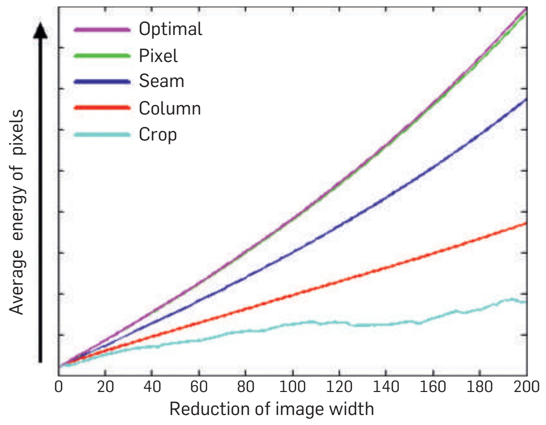
$$\mathbf{s}^{\mathrm{y}} = \{s_j^y\}_{j=1}^m = \{(j, y(j))\}_{j=1}^m, \text{ s.t. } \forall j | y(j) - y(j-1) | \le 1 \quad (3)$$

The pixels of the path of seam **s** (e.g., vertical seam $\{s_i\}$) will therefore be $\mathbf{I}_s = \{\mathbf{I}(s_i)\}_{i-1}^n = \{\mathbf{I}(x(i), i)\}_{i=1}^n$. Note that similar to the removal of a row or column from an image, removing the pixels of a seam from an image has only a local effect: all the pixels of the image are shifted left (or up) to compensate for the missing path. The visual impact is noticeable only along the path of the seam, leaving the rest of the image intact.

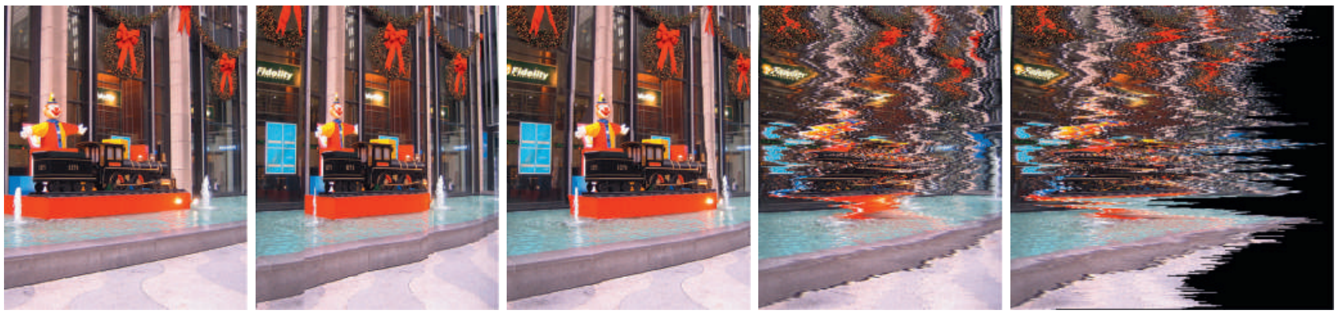
Figure 3: Results of five different strategies for reducing the width of an image. (a) The original image and its e_1 energy function, (b) average energy graph, (c) best cropping, (d) removing columns with minimal energy, (e) seam removal, (f) removal of the pixel with the least amount of energy in each row, and finally, (g) global removal of pixels with the lowest energy, regardless of their position. The graph shows the energy preservation of each strategy.



(a) Original image and its energy



(b) Average energy



(c) Crop (d) Column (e) Seam (f) Pixel (g) Optimal

We define the cost of a seam as the sum of energy of its pixels $E(\mathbf{s}) = E(\mathbf{I}_{\mathbf{s}}) = \Sigma_{i=1}^{n} e(\mathbf{I}(s_i))$, and look for the optimal seam s^* that minimizes this cost:

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

Although there seems to be an exponential number of possible seams, the optimal seam can in fact be found using dynamic programming in linear complexity. The first step is to traverse the image from the second row to the last row and compute the cumulative minimum energy M for all possible connected seams for each entry (i, j):

$$\begin{aligned} M(i,j) = e(i,j) + \min(&M(i-1,j-1), M(i-1,j),\\ &M(i-1,j+1)) \end{aligned} \tag{5}$$

At the end of this process, the minimum value of the last row in M will indicate the end of the minimal connected vertical seam. Hence, in the second step we backtrack from this minimum entry on M to find the path of the optimal seam. The definition of M for horizontal seams is similar.

Reducing the width of an image by n pixels boils down to applying the seam carving operator n times. That is, in each time step we find the optimal seam in the image, remove the pixels associated with it and repeat the process for n times. It is worth noting that this approach resembles the dynamic shortest paths problem,[15] where finding the seam is equivalent to finding a shortest path on a graph, and removing the seam modifies the graph for the next iteration of shortest path search.

3.2. The two directional case

Image retargeting generalizes aspect ratio change by modifying the image size in both directions, such that an image $\mathbf{I}$ of size $n \times m$ can be retargeted to size $n' \times m'$. For the time being, we assume that $m' < m$ and $n' < n$. Using seam carving, this raises the question of what is the correct order of seam removal? Vertical seams first? Horizontal seams first? Alternate between the two? Or any other order? We define the search for the optimal order of seam removal as an optimization of the following objective function:

$$\min_{\mathbf{s}^x, \mathbf{s}^y, \delta} \sum_{i=1}^{k} E(\delta_i \mathbf{s}_\mathbf{i}^\mathbf{x} + (1-\delta_i)\mathbf{s}_\mathbf{i}^\mathbf{y}) \tag{6}$$

where $k = r + c$, $r = (m - m')$, $c = (n - n')$ and δ_i is used as a parameter that determines if at step i we remove a horizontal or vertical seam: $\delta_i \in \{0, 1\}$, $\Sigma_{i=1}^{k}\delta_i = r$, $\Sigma_{i=1}^{k}(1-\delta_i) = c$.

We find the optimal order using a transport map $\mathbf{T}$ that specifies, for each desired target image size $n' \times m'$, the cost of the optimal sequence of horizontal and vertical seam removal operations. That is, entry $T(r, c)$ holds the minimal cost needed to obtain an image of size $n - r \times m - c$. We compute $\mathbf{T}$ using dynamic programming. Starting at $\mathbf{T}(0, 0) = 0$ we fill each entry (r, c) choosing the best of the two options—either removing a horizontal seam from an image of size $n - r \times m - c + 1$ or removing a vertical seam from an image of size $n - r + 1 \times m - c$:

$$\begin{aligned} \mathbf{T}(r,c) = \min(&\mathbf{T}(r-1,c) + E(\mathbf{s}^x(\mathbf{I}_{\mathbf{n\text{-}r\text{-}1\times m\text{-}c}})),\\ &\mathbf{T}(r,c-1) + E(\mathbf{s}^y(\mathbf{I}_{\mathbf{n\text{-}r\times m\text{-}c\text{-}1}}))) \end{aligned} \tag{7}$$

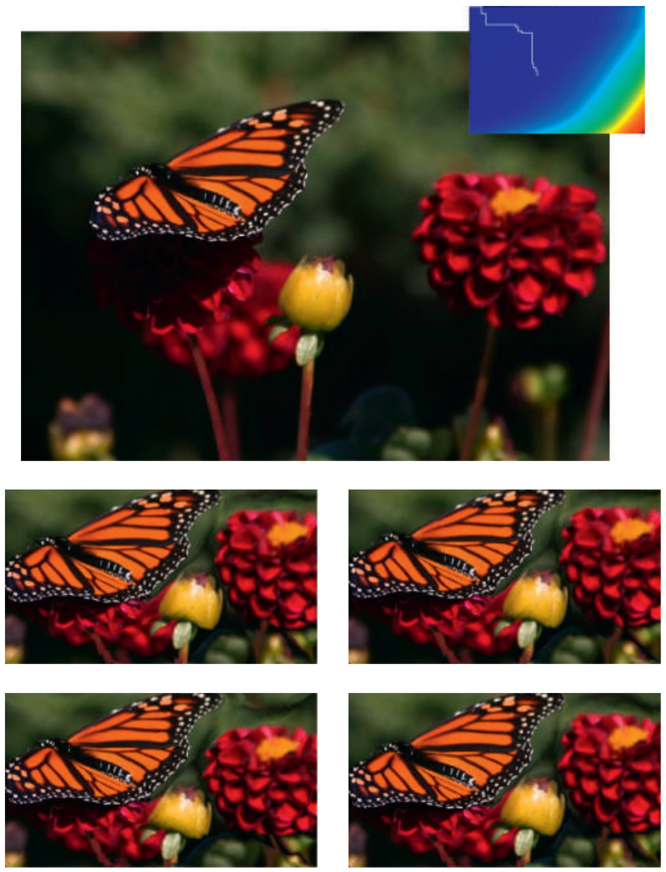
where $\mathbf{I}_{\mathbf{n-r\times m-c}}$ denotes an image of size $n - r \times m - c$, $E(\mathbf{s}^x(\mathbf{I}))$ and $E(\mathbf{s}^y(\mathbf{I}))$ are the cost of the respective seam removal operation. We store a simple $n \times m$ 1-bit map which indicates which of the two options was chosen in each step of the dynamic programming. Choosing a left neighbor corresponds to a vertical seam removal while choosing the top neighbor corresponds to a horizontal seam removal. Given a target size $n' \times m'$ where $n' = n - r$ and $m' = m - c$, we backtrack from $\mathbf{T}(r, c)$ to $\mathbf{T}(0, 0)$ and apply the corresponding removal operations. Figure 4 shows an example of different retargeting strategies on an image.

3.3. Forward energy

To evaluate the effectiveness of the different strategies for content-aware resizing, we can examine the average energy of all the pixels in an image $\frac{1}{|\mathbf{I}|}\sum_{p\in I} e(p)$ during resizing. Randomly removing pixels should keep the average unchanged, but content-aware resizing should raise the average as it removes low energy pixels and keeps the high energy ones (Figure 3(f)).

Choosing to remove the seam with the least amount of energy from the image (Equation 5) may work for many

Figure 4: Optimal order retargeting: on the top is the original image and its transport map T. Given a target size, we follow the optimal path (white path on T) to obtain the retargeted image (top row, right). For comparison we show retargeting results by alternating between horizontal and vertical seam removal (top row, left), removing vertical seams first (bottom row, left), and removing horizontal seams first (bottom row, right).



images but ignores energy that is *inserted* into the target image. The inserted energy is due to new intensity edges created by previously nonadjacent pixels that become neighbors once the seam is removed (see, e.g., the steps artifacts in Figure 5). Assume we resize an image $I = I_{t=1}$ using k seam removals ($t = 1, \ldots, k$). 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 $E(S_i)$). The energy difference after the ith seam carving operation is

$$\Delta E_{t=i+1} = E(I_{t=i+1}) - [E(I_{t=i}) - E(S_i)] \tag{8}$$

As can be seen in Figure 5 (right), ΔE_t can actually increase as well as decrease for different seam removals using Equation 5. The figure also shows a specific example of a seam that inserts more energy to the image than it removes.

Following these observations, we formulate the *forward* looking criterion. 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 cost, but will leave less artifacts in the target image, after removal. 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 "intensity-edges" created in the image. The cost of these intensity edges is measured as the differences between the values of the pixels that become new neighbors, after the seam is removed. Depending on the connectivity of the seam, three such cases are possible (see Figure 6). 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) $C_R(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j+1)|$

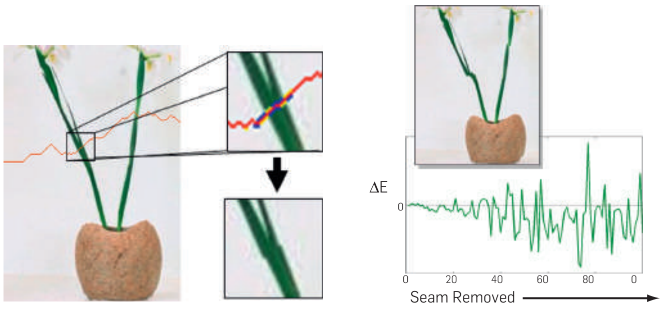


Figure 5: In some cases, choosing to remove the seam with the least amount of energy can create artifacts due to energy inserted into the image. Left: an example of the change in energy after a specific seam is removed. In some pixels energy is reduced (blue), while in others increased (yellow). Right: the actual change in energy ∅Δ for a sequence of seam removals.

We use these costs in a new forward-cumulative cost matrix *MF* to calculate the seams using dynamic programming. For vertical seams, each cost *MF*(i,j) is updated using the following rule:

$$MF(i,j) = P(i,j) + \min \begin{cases} MF(i-1,j-1) + C_L(i,j) \\ MF(i-1,j) + C_U(i,j) \\ MF(i-1,j+1) + C_R(i,j) \end{cases} \tag{9}$$

where $P(i, j)$ is an additional pixel based energy measure, such as the result of high level tasks (e.g., face detector) or a user supplied weight, that can be used in addition to the forward energy cost. Figure 7 shows a comparison of the results using the two formulations.

3.4. Seam carving using graph cut

The dynamic programming formulation works well for images, as would a shortest path approach. However, neither one scales to video and hence we switch to a graph-cut formalism that can be used either for images or video. 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 is created by connecting pixels based on their similarity together with some constraints. The graph is partitioned into disjoint subsets by removing, or cutting, some of its edges (arcs). For videos, it is often convenient to consider the sequence of frames as a 3D space–time cube,[11, 19, 24, 26] and use voxels connecting temporally instead of just pixels.

Our graph construction is a little different than most previous work both for images and for video. The challenge we face is to design a graph that produces only admissible cuts, for example, cuts whose intersection with each video frame (or image) will produce a valid seam, that is, it must satisfy two constraints:

Figure 6: 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.

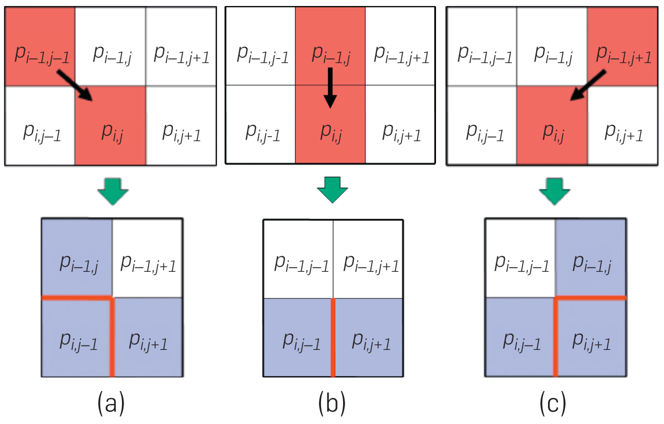
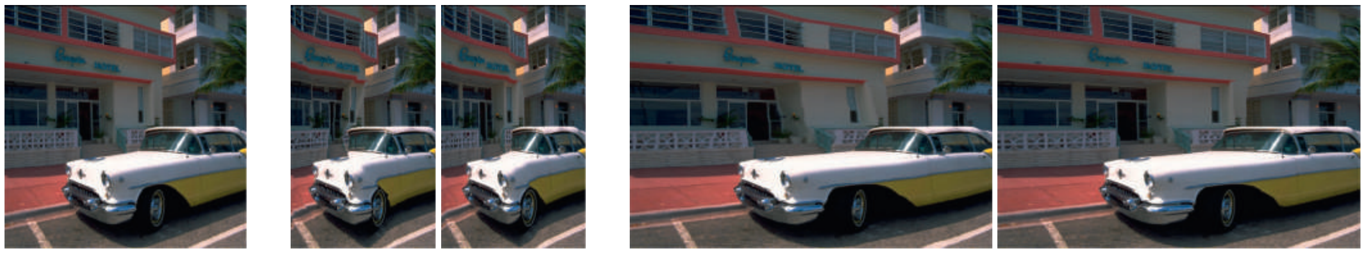


Figure 7: A comparison of results for reduction and expansion of the car image (leftmost) using least cost seam of Equation 5 (left in each pair) and least inserted cost seams of Equation 9 (right in each pair). We discuss image enlarging in Section 4.3.



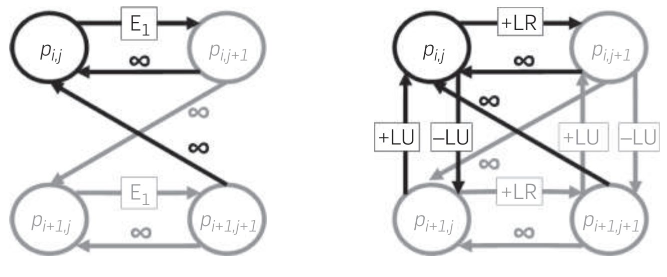
Monotonicity: the seam must include one, and only one pixel, in each row (or column for horizontal seams).
Connectivity: the pixels of the seams must be connected.

The monotonicity constraint ensures it is a function, while the connectivity constraint enforces continuity. Hence, the challenge is to construct a graph that guarantees the resulting cut will be a continuous function over the relevant domain. However, standard graph-cut-based constructions do not satisfy these constraints.

In our construction every node represents a pixel, and connects to its neighboring pixels in a grid-like structure. Virtual terminal nodes, S (source) and T (sink) are created and connected with infinite weight arcs only to the pixels of the leftmost and rightmost columns of the image, respectively (and only to the sides of the cube for video). Moreover, the key difference is that we use backward infinity arcs between the nodes. These arcs constrain the resulting cut. 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. Figure 8 illustrates the construction of such graphs on images for the two approaches of Equations 5 and 9. These graphs constructions guarantee that the seam defined by the graph-cut algorithm would be the optimal, monotonic, and connected.

In the case of video we repeat the grid-like construction of the graph both in space (within a frame) and in time (between frames in one direction). In addition, we connect the source and sink nodes to the leftmost and rightmost pixels in all the frames of the video (for the case of vertical seams). The resulting cut of this 3D graph is a 2D manifold whose intersection with every video frame produces a valid seam (Figure 2, right). Hence, a vertical seam can be thought of as a discretely continuous function $S: Y \times T \rightarrow X$ from (row, time) to column. Note that similar constructions can be used to change the height as well as the *length* of the video in time. More details can be found in Rubinsteing.[17]

Figure 8: Two graph constructions that are used by the graph-cut algorithm illustrated by four neighboring nodes. The actual image graph is created by tiling these subgraphs across the image. The left graph is equivalent to the dynamic-programming seam carving approach of Equation 5 and right graph represents the new forward energy of Equation 9.



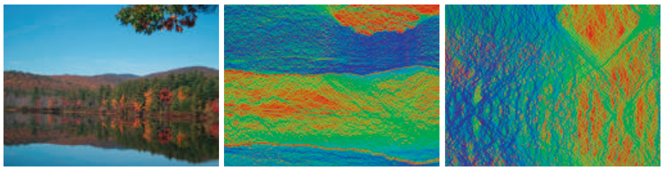
3.5. Multisize media

So far we have assumed that the user knows the target size ahead of time, but this might not be possible in all cases. Consider, for example, an image embedded in a web page. The web designer does not know, ahead of time, at what resolution the page will be displayed and therefore cannot generate a *single* target image. In a different scenario, the user might want to try different target sizes and choose the one most suitable for his or her needs. Seam carving is linear in the number of pixels and resizing is therefore linear in the number of seams to be removed, or inserted. Nevertheless, when video is concerned carving one seam surface could take up to a few seconds. Therefore, computing seams in real time is a challenging task.

To address these issues we define a representation of multisize media that encodes, for an image or video of size $(m \times n)$, an entire range of retargeting sizes from 1×1 to $m \times n$ and even further for expansion to $N' \times M'$, when $N' > n$, $M' > m$ (see Section 4.3). Using a preprocessing stage to compute and encode the seams, multisize images and video allow real time retargeting to any target size even on very low end processing units.

Looking at it from a different perspective, multisize media can be seen as storing an *explicit* representation of the time-evolution implicit process of seam removals and insertions. Consider, for example, the case of changing the width of the media. We define an index map $\mathbf{V}$ of size $n \times m$ ($\times t$ if it is a video), that encodes, for each pixel, the index of the seam that removed it, i.e., $\mathbf{V}(i, j) = t$ means that pixel (i, j) was removed by the tth seam removal. To get an image (or frame in the video) of width m', we only need to gather, in each row, all pixels with seam index greater than or equal to $m - m'$. A similar map $\mathbf{H}$ can be defined to change the height of the media (Figure 9).

Figure 9: An image with its vertical and horizontal seam index maps V and H, colored by their index from blue (first seams) to red (last seams).



4. APPLICATIONS

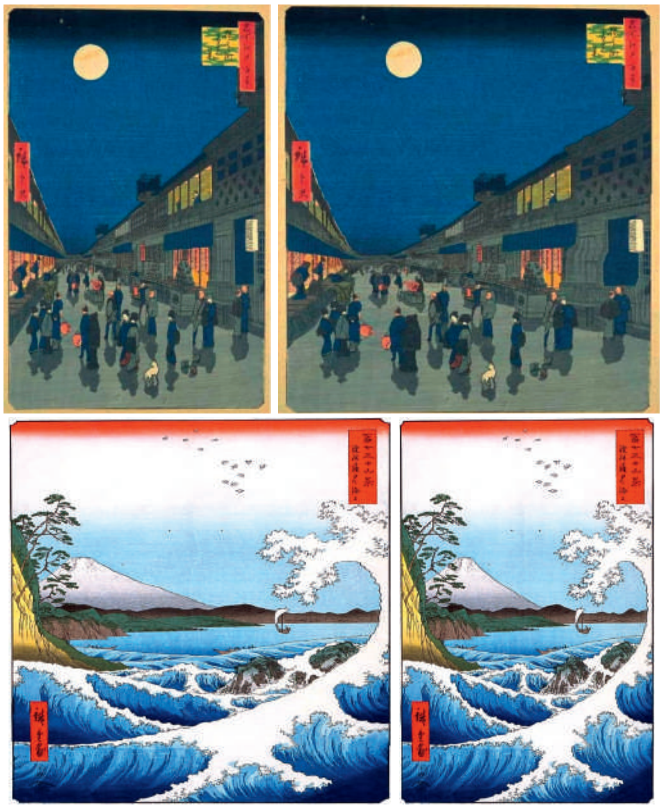
In this section we demonstrate a number of applications based on seam carving.

4.1. Aspect ratio change

Assume we want to change the aspect ratio of a given image **I** from $n \times m$ to $n \times m'$ where $m - m' = c$. This can be achieved simply by successively removing c vertical seams from **I**. Contrary to simple scaling, this operation will not alter important parts of the image (as defined by the energy function), and in effect creates a nonuniform, content-aware resizing of the image (Figure 7).

The same aspect ratio correction, from $n \times m$ to $n \times m'$, can also be achieved by increasing the number of rows by a factor of m/m' (Figure 10). The added value of such an approach is that it does not remove any information from the image. We discuss our strategy for *increasing* an image size in details in Section 4.3.

Figure 10: Aspect ratio change of pictures of the Japanese master Utagawa Hiroshige. In both examples the original image is enlarged by seam insertion.



4.2. Image retargeting

In case the user is interested in changing both the width and the height of the image, this can be achieved by finding the optimal sequence of horizontal and vertical seams. User input can be provided in the form of positive pixel weights to guide the retargeting process. Figure 4 demonstrates automatic image retargeting, while Figure 11 shows a user guided example.

4.3. Image enlarging

The process of removing vertical and horizontal seams can be seen as a time-evolution process. We denote $\mathbf{I}^{(t)}$ as the smaller image created after t seam have been removed from **I**. To enlarge an image we approximate an "inversion" of this time evolution and insert new "artificial" seams to the image. Hence, to enlarge the size of an image **I** by one we compute the optimal vertical (horizontal) seam **s** on **I** and duplicate the pixels of **s** by averaging them with their left and right neighbors (top and bottom in the horizontal case).

Using the time-evolution notation, we denote the resulting image as $\mathbf{I}^{(-1)}$. Unfortunately, repeating this process will most likely create a stretching artifact by choosing the same seam (Figure 12(b)). To achieve effective enlarging, it is important to balance between the original image content and the artificially inserted parts. Therefore, to enlarge an image by k, we find the first k seams for *removal*, and duplicate them to arrive at $\mathbf{I}^{(-k)}$ (Figure 12(c)). This can be viewed as the process of traversing back in time to recover pixels from a larger image that would have been removed by seam removals (although it is *not* guaranteed to be the case).

Figure 11: Retargeting the Buddha. At the top is the original image, a cropped version where the ornaments are gone, and a scaled version where the content is elongated. Using simple bottom up feature detection for automatic retargeting cannot protect the structure of the face of the Buddha (bottom, left) and this is a challenging image for face detectors as well. By adding simple user constraints to protect the face (bottom, middle) or the face and flower (bottom, right), better results are achieved.

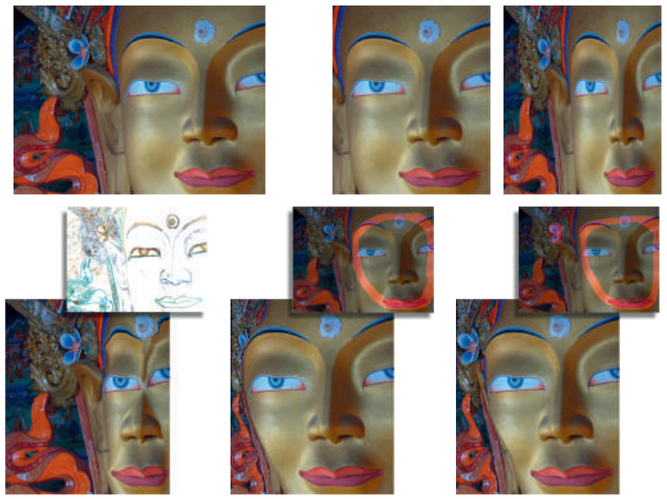
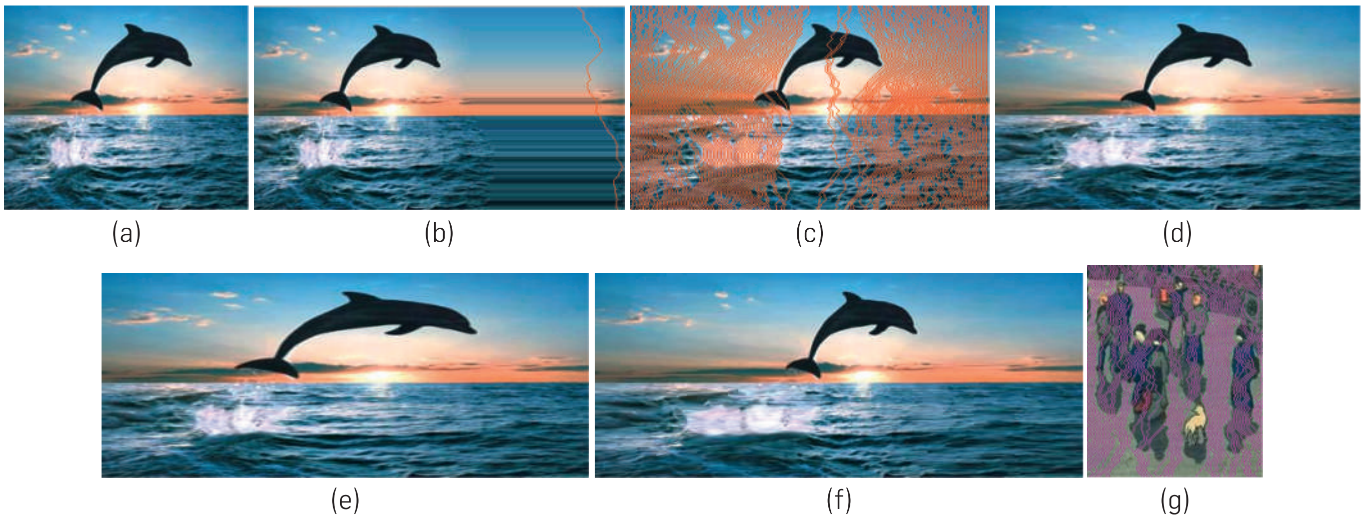


Figure 12: Seam insertion: finding and inserting the optimum seam on an enlarged image will most likely insert the same seam again and again as in (b). Inserting the seams in order of removal (c) achieves the desired 50% enlargement (d). Using two steps of seam insertions of 50% in (f) achieves better results than scaling (e). In (g), a close view of the seams inserted to expand Figure 10 is shown.



Duplicating all the seams in an image is equivalent to standard scaling (see Figure 12(e)). To continue in content-aware fashion for excessive image enlarging (for instance, greater than 50%), we break the process into several steps. Each step does not enlarge the size of the image in more than a fraction of its size from the previous step, essentially guarding the important content from being stretched. Nevertheless, extreme enlarging of an image would most probably produce noticeable artifacts (Figure 12(f)).

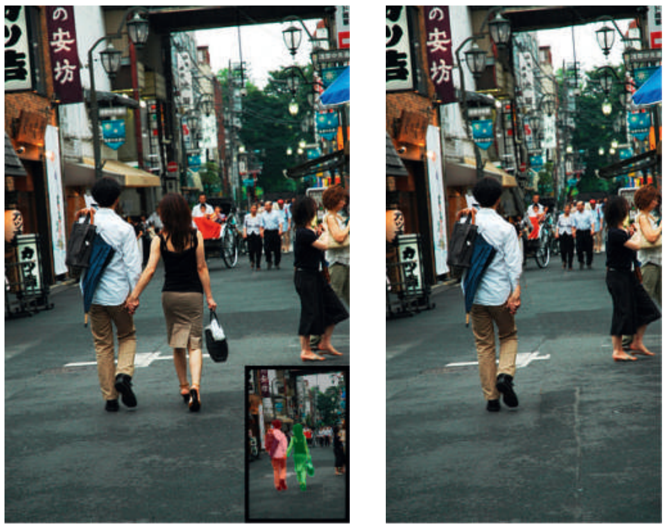
4.4. Object removal

We use a simple user interface for object removal. The user marks the object to be removed using negative weights, in effect drawing the seams to pass through these pixels. Consequently, seams are removed from the image until all marked pixels are gone. The system can automatically calculate the smaller of the vertical or horizontal diameters (in pixels) of the target removal region and perform vertical or horizontal removals accordingly (Figure 13). Moreover, to retain the original size of the image, seam insertion is employed on the resulting (smaller) image (see Figure 14). Note that this scheme alters the whole image (either its size or its content if it is resized back). This is because both the removed and inserted seams may pass anywhere in the image.

The reader is referred to more examples of retargeting and resizing of video and images at

http://www.faculty.idc.ac.il/Arik/SCWeb

Figure 13: Simple object removal: the user marks a region for removal (green), and possibly a region to protect (red), on the original image (see inset in left image). On the right image, consecutive vertical seam were removed until no "green" pixels were left.



5. LIMITATIONS

Most examples shown in this paper were computed automatically using the e_1 error function. However, it is clear that this scheme does not work well on all images. Other types of importance functions either manual or automatic could be used in combination with higher level cues such as face detectors to achieve better results (Figure 11).

Still, there are times when not even high level information can solve the problem. We can characterize two major factors that limit the seam carving approach. The first is the amount of content in an image. If the image is too condensed, in the sense that it does not contain "less important" areas, then any type of content-aware resizing strategy will not succeed. The second type of limitation is the layout of the image content. In certain types of images, albeit not being condensed, the content is laid out in a manner that prevents the seams from bypassing important parts (Figure 15).

Figure 14: Object removal: find the missing shoe (original image is top left). In this example, in addition to removing the object (one shoe), the image was enlarged back to its original size. Note that this example would be difficult to accomplish using in-painting or texture synthesis.

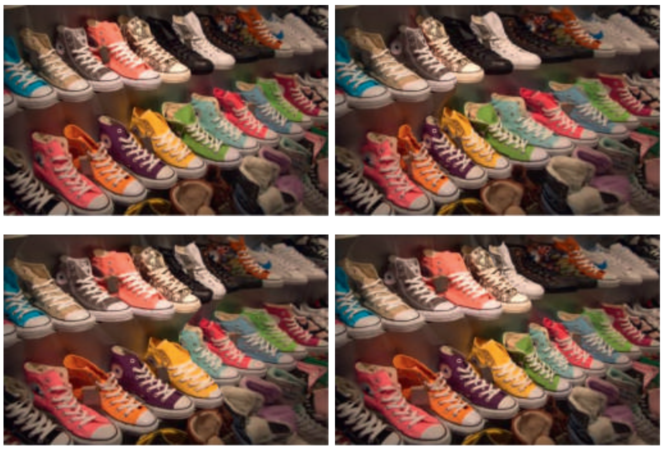
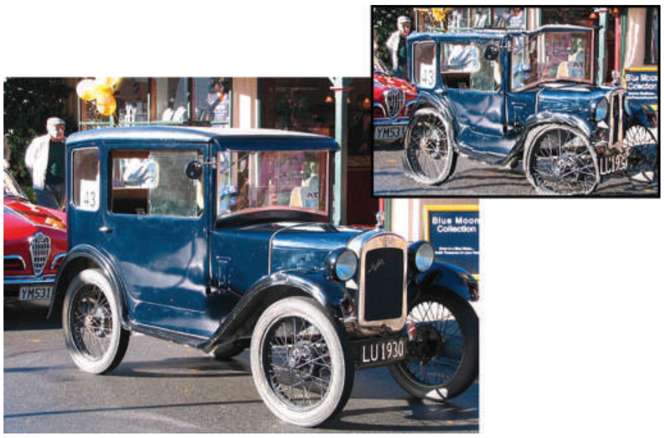


Figure 15: An example where seam carving fails because the image is too condensed. In such cases the best strategy might be to use scaling.



6. CONCLUSIONS AND FUTURE WORK

We presented an operator for content-aware resizing of images and video using seam carving. Seams are computed as the optimal paths on an image or video and are either removed or inserted. This operator can be used for a variety of image and video manipulations including aspect ratio change, image and video retargeting and object removal. The operator can be easily integrated with various saliency measures, as well as user input, to guide the resizing process. In addition, we define a novel media representation called multisize images and video that support continuous resizing ability in real time.

There are numerous possible extensions to this work. One would be to investigate better importance function that produces visually more pleasing results. Another direction is to rely on the seam map created by our method as a saliency map that can guide other processes such as content-aware compression. Finally, one could think of ways to unify the seam carving operator with other types such as scaling and cropping.

Acknowledgments

We thank Micheal Rubinstein for his work on video retargeting, Eric Chan for the use of the waterfall image and numerous flickr (*http://www.flickr.com/*) members for making their images available through creative common rights (*http://creativecommons.org/*): crazyegg95 (Buddha), Gusthy (Couple and Surfers), mykaul (Car), 02ma (Vase). C

References

1. Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., and Cohen, M. Interactive digital photomontage. *ACM Trans. Graph. 23*, 3 (2004), 294–302.
2. Avidan, S. and Shamir, A. Seam carving for content-aware image resizing. *ACM Trans. Graph. 26*, 3 (2007), 10.
3. Boykov, Y. and Jolly, M.-P. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *International Conference on Computer Vision (ICCV)*, volume I, 2001, 105–112.
4. Chen, L., Xie, X., Fan, X., Ma, W., Zhang, H., and Zhou, H. A visual attention model for adapting images on small displays. *Multimedia Systems 9*, 4 (2003), 353–364.
5. Davis, J. Mosaics of scenes with moving objects. In *Proceedings of CVPR*, 1998.
6. Efros, A.A. and Freeman, W.T. Image quilting for texture synthesis and transfer. In *SIGGRAPH 2001, Computer Graphics Proceedings*. E. Fiume, ed. ACM/ACM SIGGRAPH, 2001, 341–346.
7. Fan, X., Xie, X., Zhou, H.-Q., and Ma, W.-Y. Looking into video frames n small displays. In *MULTIMEDIA '03: Proceedings of the 11th ACM International Conference on Multimedia*. ACM, 2003, 247–250.
8. Gal, R., Sorkine, O., and Cohen-Or, D. Feature-aware texturing. In *Eurographics Symposium on Rendering*, 2006.
9. Itti, L., Koch, C., and Neibur, E. A model of saliency-based visual attention for rapid scene analysis. *PAMI 20*, 11 (1999), 1254–1259.
10. Jia, J., Sun, J., Tang, C.-K., and Shum, H.-Y. Drag-and-drop pasting. In *Proceedings of SIGGRAPH*, 2006.
11. Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3 (2003), 277–286.
12. Liu, F. and Gleicher, M. Automatic image retargeting with fisheye-view warping. In *ACM UIST*, 2005, 153–162.
13. Liu, F. and Gleicher, M. Video retargeting: automating pan and scan. In *MULTIMEDIA'06: Proceedings of the 14th annual ACM International Conference on Multimedia*. ACM, 2006, 241–250.
14. Liu, H., Xie, X., Ma, W., and Zhang, H. Automatic browsing of large pictures on mobile devices. *Proceedings of the 11th ACM International Conference on Multimedia*, 2003, 148–155.
15. Roditty, L. and Zwick, U. On dynamic shortest paths problems. In *Proceedings of the 12th Annual European Symposium on Algorithms (ESA)*, 2004, 580–591.
16. Rother, C., Bordeaux, L., Hamadi, Y., and Blake, A. Autocollage. In *Proceedings of SIGGRAPH 2006*, 2006.
17. Rubinsteing, M., Shamir, A., and Avidan, S. Improved seam carving for video retargeting. *ACM Trans. Graph. 27*, 3 (2008), 10.
18. Santella, A., Agrawala, M., DeCarlo, D., Salesin, D., and Cohen, M. Gaze-based interaction for semi-automatic photo cropping. In *ACM Human Factors in Computing Systems (CHI)*, 2006, 771–780.
19. Schödl, A., Szeliski, R., Salesin, D. H., and Essa, I. Video textures. In *SIGGRAPH'00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques.* ACM/Addison-Wesley, 2000, 489–498.
20. Setlur, V., Takagi, S., Raskar, R., Gleicher, M., and Gooch, B. Automatic image retargeting. In *The Mobile and Ubiquitous Multimedia (MUM)*. ACM, 2005.
21. Simakov, D., Caspi, Y., Shechtman, E., and Irani, M. Summarizing visual data using bidirectional similarity. In *Proceedings of CVPR*, 2008.
22. Suh, B., Ling, H., Bederson, B.B., and Jacobs, D.W. Automatic thumbnail cropping and its effectiveness. In *UIST'03: Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, 95–104.
23. Viola, P. and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.
24. Wang, J., Bhat, P., Colburn, R. A., Agrawala, M., and Cohen, M. F. Interactive video cutout. *ACM Trans. Graph. 24*, 3 (2005), 585–594.
25. Wang, J., Reinders, M., Lagendijk, R., Lindenberg, J., and Kankanhalli, M. Video content presentation on tiny devices. In *IEEE International Conference on Multimedia and Expo (ICME)*, volume 3, 2004, 1711–1714.
26. Wang, J., Xu, Y., Shum, H.-Y., and Cohen, M. F. Video tooning. *ACM Trans. Graph. 23*, 3 (2004), 574–583.
27. Wang, Y.-S., Tai, C.-L., Sorkine, O., and Lee, T.-Y.. Optimized scale-and-stretch for image resizing. *ACM Trans. Graph. Proceedings of ACM SIGGRAPH ASIA 27*, 5 (2008).
28. Wei, L.-Y., Han, J., Zhou, K., Bao, H., Guo, B., Shum, H.-Y.. Inverse texture synthesis. *ACM Trans. Graph. 27*, 3 (2008), 1–9.
29. Wolf, L., Guttmann, M., and Cohen-Or, D. Non-homogeneous content-driven video-retargeting. In *Proceedings of the 11th IEEE International Conference on Computer Vision (ICCV'07)*, 2007, 1–6.

Ariel Shamir (arik@idc.ac.il), Efi Arazi School of Computer Science, The Interdisciplinary Center, Herzliya, Israel.

Shai Avidan (avidan@adobe.com), Adobe Systems, Inc., Newton, MA.

© 2009 ACM 0001-0782/09/0001 $5.00