Seam Carving for Content-Aware Image Resizing

Shai Avidan
Mitsubishi Electric Research Labs

Ariel Shamir
The Interdisciplinary Center & MERL

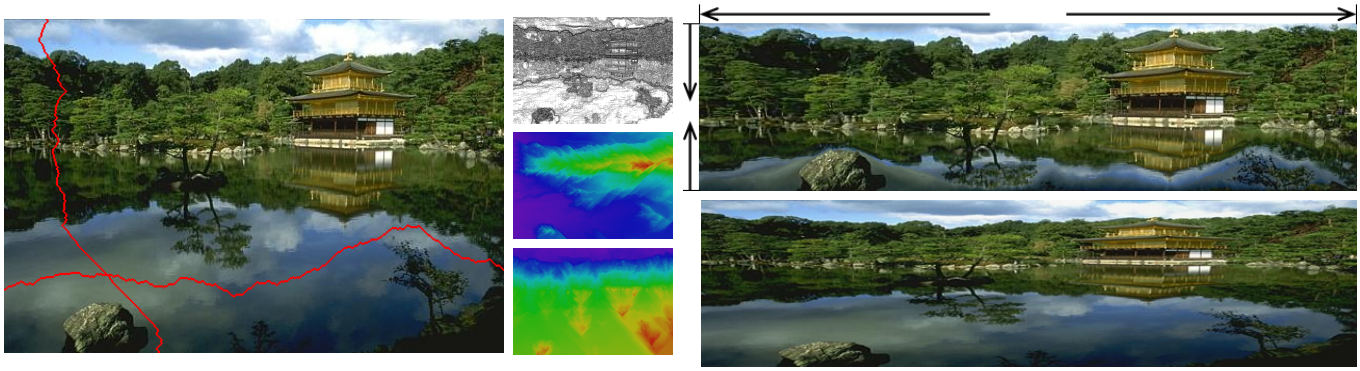


Figure 1: A seam is a connected path of low energy pixels in an image. On the left is the original image with one horizontal and one vertical seam. In the middle the energy function used in this example is shown (the magnitude of the gradient), along with the vertical and horizontal path maps used to calculate the seams. By automatically carving out seams to reduce image size, and inserting seams to extend it, we achieve *content-aware resizing*. The example on the top right shows our result of extending in one dimension and reducing in the other, compared to standard scaling on the bottom right.

Abstract

Effective resizing of images should not only use geometric constraints, but consider the image content as well. We present a simple image operator called *seam carving* that supports content-aware image resizing for both reduction and expansion. A seam is an optimal 8-connected path of pixels on a *single* image from top to bottom, or left to right, where optimality is defined by an image energy function. By repeatedly carving out or inserting seams in one direction we can change the aspect ratio of an image. By applying these operators in both directions we can retarget the image to a new size. The selection and order of seams protect the content of the image, as defined by the energy function. Seam carving can also be used for image content enhancement and object removal. We support various visual saliency measures for defining the energy of an image, and can also include user input to guide the process. By storing the order of seams in an image we create *multi-size* images, that are able to continuously change in real time to fit a given size.

CR Categories: I.3.0 [Computing Methodologies]: Computer Graphics—General; I.4.10 [Computing Methodologies]: Image Processing And Computer Vision —Image Representation

Keywords: Image resizing, Image retargeting, Image seams, Content-aware image manipulation, Display devices

ACM Reference Format
Avidan, S., Shamir, A. 2007. Seam Carving for Content-Aware Image Resizing. *ACM Trans. Graph. 26*, 3, Article 10 (July 2007), 9 pages. DOI = 10.1145/1239451.1239461 http://doi.acm.org/10.1145/1239451.1239461.

1 Introduction

The diversity and versatility of display devices today imposes new demands on digital media. For instance, designers must create different alternatives for web-content and design different layouts for different devices. Moreover, HTML, as well as other standards, can support dynamic changes of page layout and text. Nevertheless, up to date, *images*, although being one of the key elements in digital media, typically remain rigid in size and cannot deform to fit different layouts automatically. Other cases in which the size, or aspect ratio of an image must change, are to fit into different displays such as cell phones or PDAs, or to print on a given paper size or resolution.

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. More effective resizing can only be achieved by considering the image *content* and not only geometric constraints.

We propose a simple image operator, we term *seam-carving*, that can change the size of an image by gracefully carving-out or inserting pixels in different parts of the image. Seam carving uses an energy function defining the importance of pixels. A seam is a connected path of low energy pixels crossing the image from top to bottom, or from left to right. By successively removing or inserting seams we can reduce, as well as enlarge, the size of an image in both directions (see Figure 1). For image reduction, seam selection ensures that while preserving the image structure, we remove more of the low energy pixels and fewer of the high energy ones. For image enlarging, the order of seam insertion ensures a balance between the original image content and the artificially inserted pixels. These operators produce, in effect, a *content-aware* resizing of images.

We illustrate the application of seam carving and insertion for aspect ratio change, image retargeting, image content enhancement, and object removal. Furthermore, by storing the order of seam removal and insertion operations, and carefully interleaving seams in

both vertical and horizontal directions we define *multi-size images*. Such images can continuously change their size in a content-aware manner. A designer can author a multi-size image once, and the client application, depending on the size needed, can resize the image in real time to fit the exact layout or the display.

Seam carving can support several types of energy functions such as gradient magnitude, entropy, visual saliency, eye-gaze movement, and more. The removal or insertion processes are parameter free; however, to allow interactive control, we also provide a scribble-based user interface for adding weights to the energy of an image and guide the desired results. This tool can also be used for authoring multi-size images. To summarize, our main contributions are as follows:

- Define seam carving and present its properties.
- Present algorithm for image enlarging using seam insertions.
- Use seams for content-aware image size manipulations.
- Define multi-size images for continuous image retargeting.

2 Background

Image resizing is a standard tool in many image processing applications. It works by uniformly resizing the image to a target size. Recently, there is a growing interest in image retargeting that seeks to change the size of the image while maintaining the important features intact, where these features can be either detected top-down or bottom-up. Top down methods use tools such as face detectors [Viola and Jones 2001] to detect important regions in the image, whereas bottom-up methods rely on visual saliency methods [Itti et al. 1999] to construct a visual saliency map of the image. Once the saliency map is constructed, cropping can be used to display the most important region of the image. Suh *et al.* [2003] proposed automatic thumbnail creation, based on either a saliency map or the output of a face detector. The large image is then cropped to capture the most salient region in the image. Similarly, Chen *et al.* [2003] considered the problem of adapting images to mobile devices. In their approach the most important region in the image is automatically detected and transmitted to the mobile device. Liu *et al.* [2003] also addressed image retargeting to mobile devices, suggesting to trade time for space. Given a collection of regions of interest, they construct an optimal path through these regions and display them serially, one after the other, to the user. Santella *et al.* [2006] use eye tracking, in addition to composition rules to crop images intelligently. All these methods achieve impressive results, but rely on traditional image resizing and cropping operations to actually change the size of the image.

Jacobs *et al.* [2003] consider an adaptive grid-based document layout system that maintains a clear separation between content and template. The page designer constructs several possible templates and when the content is displayed the most suitable template is used. The templates can use different discrete alternatives of an image if they are provided, but no specific reference to image resizing is made.

A compromise between image resizing and image cropping is to introduce a non-linear, data dependent scaling. Such a method was proposed by Liu and Gleicher [2005; 2006] for image and video retargeting. For image retargeting they find the Region-Of-Interest (ROI) and construct a novel Fisheye-View warp that essentially applies a piecewise linear scaling function in each dimension to the image. This way the ROI is maintained while the rest of the image is warped. The retargeting can be done in interactive rates, once the ROI is found, so the user can control the desired size of the image by moving a slider. In their video retargeting work they use a combination of image and saliency maps to find the ROI. Then they use a combination of cropping, virtual pan and shot cuts to retarget the video frames.

Setlur *et al.* [2005] proposed an automatic, non-photorealistic algorithm for retargetting large images to small size displays. This is done by decomposing the image into a background layer and foreground objects. The retargeting algorithm segments an image into regions, identifies important regions, removes them, fills the resulting gaps, resize the remaining image, and re-insert the important region.

The first solution to the general problem of warping an image into an arbitrary shape while preserving user-specified features was recently proposed by Gal *et al.* [2006]. The feature-aware warping is achieved by a particular formulation of the Laplacian editing technique, suited to accommodate similarity constraints on parts of the domain. Since local constraints are propagated by the global optimization process, not all the constraints can always be satisfied at once. Our algorithm is discrete, so carving a single seam has no affect on the rest of the image.

The use of seams for image editing is prevalent. Agarwala *et al.* [2004] describe an interactive Digital Photomontage system that finds perfect seams to combine parts of a set of photographs into a single composite picture, using minimal user assistance. Jia *et al.* [2006] proposed Drag-and-Drop Pasting that extends the Poisson Image Editing technique [Perez et al. 2003] to compute an optimal boundary (*i.e.* seam) between the source and target images. Rother *et al.* [2006] developed AutoCollage, a program that automatically creates a collage image from a collection of images. This process requires, among other things, finding optimal boundaries, or seams, between many image fragments. None of the above methods discuss the problem of image retargeting. A notable exception is the work of Wang and Cohen [2006] that proposes to simultaneously solve matting and compositing. They allow the user to scale the size of the foreground object and paste it back on the original background. Zomet *et al.* [2005] evaluated several cost functions for seamless image stitching and concluded that minimizing an l_1 error norm between the gradients of the stitched image and the gradients of the input images performed well in general. Computing the seam can be done in a variety of ways, including Dijkstra's shortest path algorithm [1998], dynamic programming [2001] or graph cuts [2001].

Changing the size of the image has been extensively studied in the field of texture synthesis, where the goal is to generate a large texture image from a small one. Efros *et al.* [2001] find seams that minimize the error surface defined by two overlapping texture patches. This way, the original small texture image is quilted to form a much larger texture image. This was later extended to handle both image and video texture synthesis by Kwatra *et al.* [2003] that showed how to increase the space and time dimensions of the original texture video.

As for object removal, Bertalmio *et al.* [2000] proposed an image inpainting method that smoothly propagates information from the boundaries inwards, simulating techniques used by professional restorators. Patch based approaches [Drori et al. 2003; Criminisi et al. 2003; Bertalmio et al. 2003] use automatic guidance to determine synthesis ordering, which considerably improves the quality of the results. And recently, Sun *et al.* [2005] proposed an interactive method to handle inpainting in case of missing strong visual structure, by propagating structure along used-specified curves.

(a) Original (b) Crop (c) Column (d) Seam (e) Pixel (f) Optimal

Figure 2: Results of 5 different strategies for reducing the width of an image. (a) the original image and its e_1 energy function, (b) best cropping, (c) removing columns with minimal energy, (d) seam removal, (e) removal of the pixel with the least amount of energy in each row, and finally, (f) global removal of pixels with the lowest energy, regardless of their position. Figure 3 shows the energy preservation of each strategy.

3 The Operator

Our approach to content-aware resizing is to remove pixels in a judicious manner. Therefor, the question is how to chose the pixels to be removed? Intuitively, our goal is to remove unnoticeable pixels that blend with their surroundings. This leads to the following simple *energy function* that was used in many figures in this paper such as Figures 1, 6, 5, 8, 11, 12, 13 (we explore other energy functions in subsection 3.2):

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

Given an energy function, assume we need to reduce the image width. 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 2(f)). 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 2(e)). To preserve both the shape and the visual coherence of the image we can use Auto-cropping. That is, look for a sub-window, the size of the target image, that contains the highest energy (Figure 2(b)). 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 2(c)). 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 2(d)) and the definition of internal seams.

Formally, let $\mathbf{I}$ be an $n \times m$ image and define a *vertical seam* to be:

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

where x is a mapping $x : [1,\ldots,n] \rightarrow [1,\ldots,m]$. That is, a vertical seam is an 8-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 1). Similarly, if y is a mapping $y : [1,\ldots,m] \rightarrow [1,\ldots,n]$, then a *horizontal seam* is:

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

The pixels of the path of seam $\mathbf{s}$ (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.

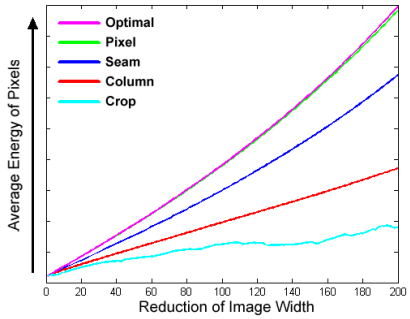


Figure 3: Image energy preservation. A comparison of the preservation of content measured by the average pixel energy using five different strategies of resizing. The actual images can be seen in Figure 2.

The visual impact is noticeable only along the path of the seam, leaving the rest of the image intact. Note also that one can replace the constraint $|x(i) - x(i-1)| \leq 1$ with $|x(i) - x(i-1)| \leq k$, and get either a simple column (or row) for $k = 0$, a piecewise connected or even completely disconnected set of pixels for any value $1 \leq k \leq m$.

Given an energy function e, we can define the cost of a seam as $E(\mathbf{s}) = E(\mathbf{I_s}) = \sum_{i=1}^n e(\mathbf{I}(s_i))$. We look for the optimal seam s^* that minimizes this seam cost :

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

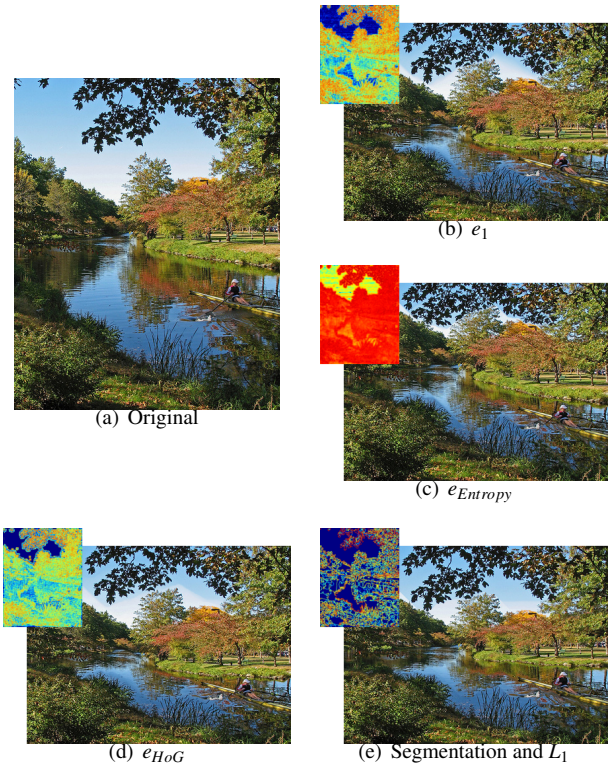
The optimal seam can be found using dynamic programming. The first step is to traverse the image from the second row to the last row and compute the cumulative minimum energy M for all possible connected seams for each entry (i,j):

$$M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

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 (see Figure 1). The definition of M for horizontal seams is similar.

3.1 Energy Preservation Measure

To evaluate the effectiveness of the different strategies for content-aware resizing, we can examine the average energy of all of pixels



(a) Original (b) e_1 (c) $e_{Entropy}$ (d) e_{HoG} (e) Segmentation and L_1

Figure 4: Comparing different energy functions for content aware resizing.

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 shows a plot of the change in average energy while changing the image width of Figure 2 using the five different strategies outlined above. As expected, removing the low energy pixels in ascending order gives the optimal result. This is closely followed by pixel removal. But both methods destroy the visual coherence of the image. Cropping shows the worst energy preservation. Column removal does a better job at preserving energy, but still introduce visual artifacts. Seam carving strikes the best balance between the demands for energy preservation and visual coherency. This graph results are characteristic to many images in general.

3.2 Image Energy Functions

We have examined several possible image importance measures found in literature as the energy function to guide seam carving. We have tested both L_1 and L_2-norm of the gradient, saliency measure [Itti et al. 1999], and Harris-corners measure [Harris and Stephens 1988]. We also used eye gaze measurement [DeCarlo and Santella 2002], and the output of face detectors.

Figure 4 compares the results of the e_1 error, entropy, segmentation, and Histogram of Gradients (HoG). The entropy energy computes the entropy over a 9×9 window and adds it to e_1. The segmentation method first segments the image [Christoudias et al. 2002] and then applies the e_1 error norm on the results, effectively leaving only the edges between segments. Finally, e_{HoG} is defined as follows:

$$e_{HoG}(\mathbf{I}) = \frac{|\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|}{\max\left(HoG(\mathbf{I}(x,y))\right)},$$

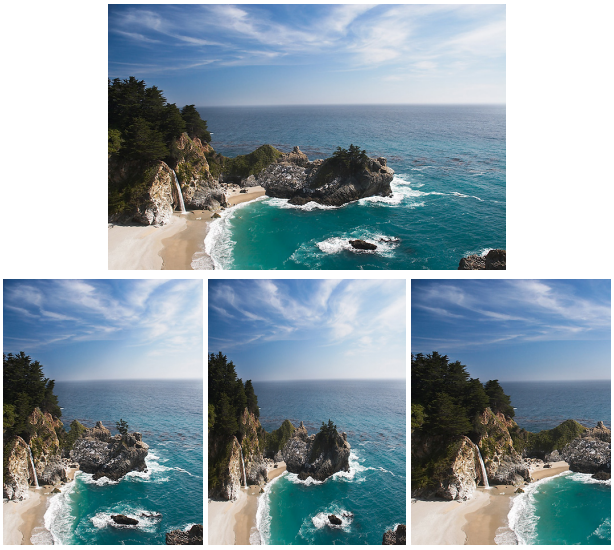


Figure 5: Comparing aspect ratio change. From left to right in the bottom: the image resized using seam removals, scaling and cropping.

where $HoG(\mathbf{I}(x,y))$ is taken to be a histogram of oriented gradients at every pixel [Dalal and Triggs 2005]. We use an 8-bin histogram computed over a 11×11 window around the pixel. Thus, taking the maximum of the HoG at the denominator attracts the seams to edges in the image, while the numerator makes sure that the seam will run parallel to the edge and will not cross it. e_{HoG} was also used in Figures 9 and 10.

As expected, no single energy function performs well across all images but in general they all accommodate a similar range for resizing. They vary in the rate at which they introduce visual artifacts and the parts of the image they affect. We found either e_1 or e_{HoG} to work quite well.

4 Discrete Image Resizing

4.1 Aspect Ratio Change

Assume we want to change the aspect ratio of a given image $\mathbf{I}$ from $n\times m$ to $n\times m'$ where $m-m'=c$. This can be achieved simply by successively removing c vertical seams from $\mathbf{I}$. 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 non-uniform, content-aware resizing of the image (Figure 5).

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 6). 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 sub-section 4.3.

4.2 Retargeting with Optimal Seams-Order

Image retargeting generalizes aspect ratio change from one dimension to two dimensions such that an image $\mathbf{I}$ of size $n\times m$ will be retargeted to size $n'\times m'$ and, for the time being, we assume that $m'<m$ and $n'<n$. This begs the question of what is the correct order of seam carving? Remove vertical seams first? Horizontal seams first? Or alternate between the two? We define the search for the optimal order as an optimization of the following objective function:

Figure 6: Aspect ratio change of pictures of the Japanese master Utagawa Hiroshige. In both examples the original image is widened by seam insertion.

$$\min_{\mathbf{s^x},\mathbf{s^y},\alpha} \sum_{i=1}^{k} E(\alpha_i \mathbf{s_i^x} + (1-\alpha_i)\mathbf{s_i^y}) \tag{5}$$

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

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 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 $n - r + 1 \times m - c$:

$$\begin{aligned} \mathbf{T}(r,c) \quad = \quad & min(\mathbf{T}(r-1,c) + E(\mathbf{s^x}(\mathbf{I_{n-r-1\times m-c}})), \\ & \mathbf{T}(r,c-1) + E(\mathbf{s^y}(\mathbf{I_{n-r\times m-c-1}}))) \end{aligned} \tag{6}$$

where $\mathbf{I_{n-r\times m-c}}$ 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 7 shows an example of different retargeting strategies on an image.

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

Figure 7: Optimal order retargeting: On the top is the original image and its transport map $\mathbf{T}$. Given a target size, we follow the optimal path (white path on $\mathbf{T}$) 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)

created after t seam have been removed from $\mathbf{I}$. 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 $\mathbf{I}$ by one we compute the optimal vertical (horizontal) seam $\mathbf{s}$ on $\mathbf{I}$ and duplicate the pixels of $\mathbf{s}$ 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 8(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 in order to arrive at $\mathbf{I^{(-k)}}$ (Figure 8(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).

Duplicating all the seams in an image is equivalent to standard scaling (see Figure 8 (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 8 (f)).

4.4 Content Amplification

Instead of enlarging the size of the image, seam carving can be used to amplify the content of the image while preserving its size. This can be achieved by combining seam carving and scaling. To preserve the image content as much as possible, we first use standard

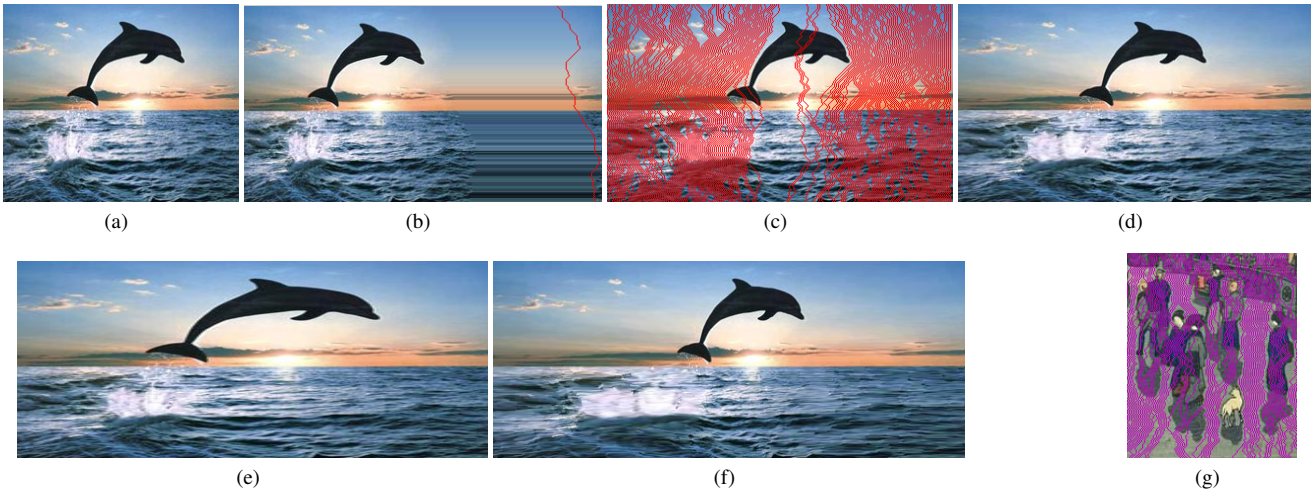


Figure 8: Seam insertion: finding and inserting the optimum seam on an enlarged image will most likely insert the same seam again and again as in (b). Inserting the seams in order of removal (c) achieves the desired 50% enlargement (d). Using two steps of seam insertions of 50% in (f) achieves better results than scaling (e). In (g), a close view of the seams inserted to expand figure 6 is shown.

Figure 9: Content amplification. On the right: a combination of seam carving and scaling amplifies the content of the original image (left).

scaling to enlarge the image and only then apply seam carving on the larger image to carve the image back to its original size (see Figure 9). Note that the pixels removed are in effect sub-pixels of the original image.

4.5 Seam Carving in the gradient domain

There are times when removing multiple seams from an image still creates noticeable visual artifacts in the resized image. To overcome this we can combine seam carving with Poisson reconstruction ([Perez et al. 2003]). Specifically, we compute the energy function image as before, but instead of removing the seams from the original image we work in the gradient domain and remove the seams from the x and y derivatives of the original image. At the end of this process we use a poisson solver to reconstruct back the image. Figure 10 shows an example of this technique.

4.6 Object Removal

We use a simple user interface for object removal. The user marks the target object to be removed and then 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 11). Moreover, to regain the original size of the image, seam insertion could be employed on the resulting (smaller) image (see Figure 12). Note that, contrary

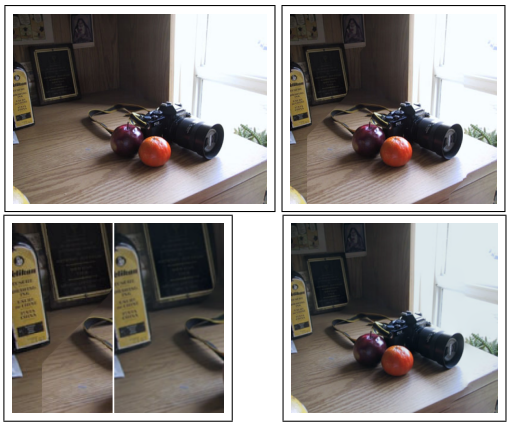


Figure 10: Seam Carving in the gradient domain. The original image (top left) is retargeted using standard technique (top right) and in the gradient domain (bottom right). Zoom in comparison is shown on bottom left.

Figure 11: 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.

Figure 12: 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.

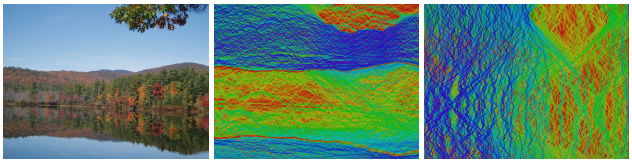


Figure 13: An image with its vertical and horizontal seam index maps **V** and **H**, colored by their index from blue (first seams) to red (last seams).

to previous object removal techniques [Drori et al. 2003; Criminisi et al. 2003; Bertalmio et al. 2003], this scheme alters the whole image (either its size or its content if it is resized). This is because both the removed and inserted seams may pass anywhere in the image.

5 Multi-size Images

So far we have assumed that the user knows the target size ahead of time, but this might not be possible in some 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. On average, we retarget an image of size 400×500 to 100×100 in about 2.2 seconds. However, computing tens or hundreds of seams in real time is a challenging task. To address this issue we present a representation of multi-size images that encodes, for an image of size $(m \times n)$, an entire range of retargeting sizes from 1×1 to $m \times n$ and even further to $N' \times M'$, when $N' > n, M' > m$. This information has a very low memory footprint, can be computed in a couple of seconds as a pre-processing step, and allows the user to retaget an image continuously in real time.

From a different perspective, this can be seen as storing an *explicit* representation of the time-evolution implicit process of seam removals and insertions. Consider, first, the case of changing the width of the image. We define an index map **V** of size $n \times m$ 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 t-th seam removal (Figure 13). To get an image of width m', we only need to gather, in each row, all pixels with seam index greater than or equal to $m - m'$.

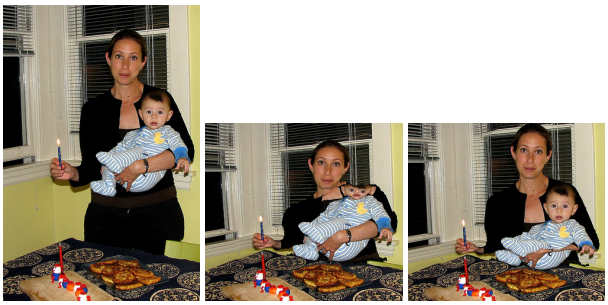


Figure 14: Retargeting the left image with e_1 alone (middle), and with a face detector (right).

Figure 15: 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.

This representation supports image enlarging as well as reduction. For instance, if we want to support enlarging of the image up to size $M' > m$, we enlarge the image using seam insertion procedure to a size $n \times M'$ similar to Section 4.3. However, instead of averaging the k-th seam with it's two neighbors, we do not modify the original image pixels in the seam, but insert new pixels to the image as the average of the k-th seam and its left (or right) pixel neighbors. The inserted seams are given a negative index starting at -1. Consequently, to enlarge the original image by $k, (m < k \leq M')$, we use exactly the same procedure of gathering (from the enlarged image) all pixels whose seam index is greater than $(m-(m+k)) = -k$, and get an image of size $m-(-k) = m+k$.

Computing a horizontal index map **H** for image height enlarging and reduction is achieved in a similar manner (see Figure 13). However, supporting both dimension resizing while computing **H** and **V** independently will not work. This is because horizontal and vertical seams can collide in more than one place, and removing a seam in one direction may destroy the index map in the other direction. More details can be found in the appendix. However, a simple way to avoid this is to allow seam removal in one direction, and use de-

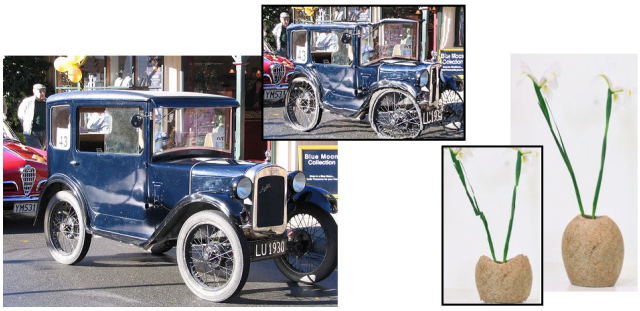


Figure 16: Examples when resizing using seams fails: images that are too condensed (left) or where the content layout prevents seams to bypass important parts (right). In such cases the best strategy would be to use scaling.

generate seams, i.e. rows or columns, in the other direction. Note that although retargeting a multi-size image to any size is instantaneous, due to the additional constraints, the resulting image would be different than the one created in the optimal order induced by the implicit time-evolving process of subsection 4.2. The reader is referred to the attached video for example of continuously resizing multi-size images to various sizes in real-time.

6 Limitations

All the examples shown so far in this paper were computed automatically, but our method does not work automatically on all images. This can be corrected by adding higher level cues, either manual or automatic. For example, in Figure 14 the e_1 error function fails, but combined with a face detector we get much better results. Figure 15 shows an example of adding user constraints.

Other times, not even high level information can solve the problem. We can characterize two major factors that limit our 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 16).

7 Conclusions and Future Work

We presented an operator for content-aware resizing of images using *seam carving*. Seams are computed as the optimal paths on a single image and are either removed or inserted from an image. This operator can be used for a variety of image manipulations including: aspect ratio change, image retargeting, content amplification 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 data structure for multi-size images that support continuous resizing ability in real time.

There are numerous possible extensions to this work. We would like to extend our approach to other domains, the first of which would be resizing of video. Since there are cases when scaling can achieve better results for resizing, we would like to investigate the possibility to combine the two approaches, specifically to define more robust multi-size images. We would also like to find a better way to combine horizontal and vertical seams in multi-size images.

Acknowledgments

We would like to thank Frédo Durand and the graphics group at MIT for reviewing an early version of this work. We thank Stark Draper for narrating the video. We thank 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), Gustty (Couple and Surfers), JeffKubina (Capitol), mykaul (Hannuka and Car), o2ma (Vase), sigs66 (Long beach and Two people near sea). We also thank the anonymous reviewers and referees for their comments.

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. 2004. Interactive digital photomontage. *ACM Trans. Graph. 23*, 3, 294–302.

BERTALMIO, M., SAPIRO, G., CASELLES, V., AND BALLESTER, C. 2000. Image inpainting. In *Proceedings of ACM SIGGRAPH*, 417–424.

BERTALMIO, M., VESE, L., SAPIRO, G., AND OSHER, S. 2003. Simultaneous structure and texture image inpainting. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 707–714.

BOYKOV, Y., AND JOLLY, M.-P. 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *International Conference on Computer Vision, (ICCV)*, vol. I, 105–112.

CHEN, L., XIE, X., FAN, X., MA, W., ZHANG, H., AND ZHOU, H. 2003. A visual attention model for adapting images on small displays. *Multimedia Systems 9*, 4, 353–364.

CHRISTOUDIAS, C., GEORGESCU, B., AND MEER, P. 2002. Synergism in low-level vision. In *16th International Conference on Pattern Recognition*, vol. IV, 150–155.

CRIMINISI, A., PEREZ, P., AND TOYAMA, K. 2003. Object removal by exemplar-based inpainting. In *In IEEE Conference on Computer Vision and Pattern Recognition*, 417–424.

DALAL, N., AND TRIGGS, B. 2005. Histograms of oriented gradients for human detection. In *International Conference on Computer Vision & Pattern Recognition*, vol. 2, 886–893.

DAVIS, J. 1998. Mosaics of scenes with moving objects. In *Proceedings of CVPR*.

DECARLO, D., AND SANTELLA, A. 2002. Stylization and abstraction of photographs. In *Proceedings of SIGGRAPH*, 769–776.

DRORI, I., COHEN-OR, D., AND YESHURUN, Y. 2003. Fragment-based image completion. In *Proceedings of ACM SIGGRAPH*, 303–312.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *SIGGRAPH 2001, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., 341–346.

GAL, R., SORKINE, O., AND COHEN-OR, D. 2006. Feature-aware texturing. In *Eurographics Symposium on Rendering*.

HARRIS, C., AND STEPHENS, M. 1988. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, 147–151.

ITTI, L., KOCH, C., AND NEIBUR, E. 1999. A model of saliency-based visual attention for rapid scene analysis. *PAMI 20*, 11, 1254–1259.

JACOBS, C., LI, W., SCHRIER, E., BARGERON, D., AND SALESIN, D. 2003. Adaptive grid-based document layout. In *Proceedings of ACM SIGGRAPH*, 838–847.

JIA, J., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2006. Drag-and-drop pasting. In *Proceedings of SIGGRAPH*.

KUHN, H. W. 1955. The hungarian method for the assignment problem. In *Naval Research Logistic Quarterly*, 2:83–97.

LIU, F., AND GLEICHER, M. 2005. Automatic Image Retargeting with Fisheye-View Warping. In *ACM UIST*, 153–162.

LIU, F., AND GLEICHER, M. 2006. Video Retargeting: Automating Pan and Scan. In *ACM international conference on Multimedia*, 241–250.

LIU, H., XIE, X., MA, W., AND ZHANG, H. 2003. Automatic browsing of large pictures on mobile devices. *Proceedings of the eleventh ACM international conference on Multimedia*, 148–155.

PEREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22*, 3, 313–318.

ROTHER, C., BORDEAUX, L., HAMADI, Y., AND BLAKE, A. 2006. Autocollage. In *Proceedings of SIGGRAPH 2006*.

SANTELLA, A., AGRAWALA, M., DECARLO, D., SALESIN, D., AND COHEN, M. 2006. Gaze-based interaction for semi-automatic photo cropping. In *ACM Human Factors in Computing Systems (CHI)*, 771–780.

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.

SUH, B., LING, H., BEDERSON, B. B., AND JACOBS, D. W. 2003. Automatic thumbnail cropping and its effectiveness. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, ACM Press, New York, NY, USA, 95–104.

SUN, J., YUAN, L., JIA, J., AND SHUM, H. 2005. Image completion with structure propagation. In *Proceedings of ACM SIGGRAPH*.

V. KWATRA, A. SCHDL, I. E. G. T., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. In *Proceedings of SIGGRAPH*.

VIOLA, P., AND JONES, M. 2001. Rapid object detection using a boosted cascade of simple features. In *Coference on Computer Vision and Pattern Recognition (CVPR)*.

WANG, J., AND COHEN, M. 2006. Simultaneous Matting and Compositing. *Microsoft Research Technical Report, MSR-TR-2006-63* (May).

ZOMET, A., LEVIN, A., PELEG, S., AND WEISS, Y. 2005. Seamless image stitching by minimizing false edges. *IEEE Transactions on Image Processing 15*, 4, 969–977.

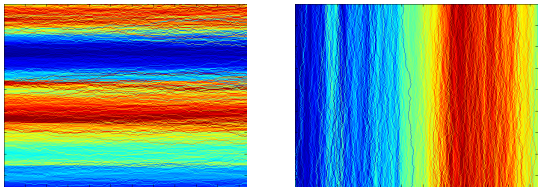


Figure 17: The horizontal and vertical *consistent* index maps of the image on figure 13. Color index goes from blue (first seams) to red (last seams).

A Constructing Consistent Index Maps

Computing an horizontal index map $\mathbf{H}$ and vertical index map $\mathbf{V}$ independently for multi-size image will not work. We say that $\mathbf{H}$ and $\mathbf{V}$ are *consistent* if every horizontal seam intersects (or touches) all the *vertical* seam indexes and every vertical seam intersects all *horizontal* seam indexes. Consistency assures that removing a seam in any dimension will remove exactly one pixel from all seams in the other dimension, retaining the index map structure. If consistency is not maintained, then after removing one horizontal seam we might be left with vertical seams with different number of pixels and the rectangular structure of the image will be destroyed.

Aside from limiting seams to be rows or columns in one, or both dimensions, we present here another approach. We use only temporally 0-connected seams, i.e. seams that are spatially connected on the original size image $\mathbf{I}^{(0)}$. For such seams, the only possible violation of consistency between the $\mathbf{H}$ and $\mathbf{V}$ maps can occur in diagonal seam steps. Our method first computes temporally 0-connected seams in one direction, and then imposes the constraints on the diagonal when computing the seams in the other direction.

To understand why the only violation of consistency occurs in diagonals, assume without loss of generality, that some vertical seam $j \in \{1 \dots m\}$ violates the consistency constraint. This means it must touch some horizontal seam $i \in \{1 \dots n\}$ more than once. Denote those pixels where seam j meets seam i as p and q. Since p and q are part of a vertical seam j, they cannot be in the same row. However, they are also part of the horizontal seam i, and cannot be in the same column. Let us examine the rectangle defined by p and q in its corners. seams i and j must be connected inside this rectangle and they both touch it's corners. However, one is a vertical seam and the other a horizontal seam. The only possibility for this to happen is that the rectangle is in fact a square, and both seams pass through its diagonal.

Note that the above claim relies on the fact that all seams are connected in the original image, which is not true if we use non 0-connected seams. However, because we are using 0-connected seams, we can simultaneously compute *all* of them in one direction. Without loss of generality, for 0-connected vertical seams we examine all pairs of rows of the original image independently. For each such pair we find the optimal set of 1-edge paths linking all pixels of one row to all pixels of the next row. The global multiple seam paths from the top of the image to its bottom would simply be the concatenation of those 1-edge paths.

Finding the best 1-edge paths between a pair of rows is similar to a weighted assignment problem where each pixel in one row is connected to its three neighboring pixels in the other row. We use the Hungarian algorithm [Kuhn 1955] to solve this weighted assignment problem. Once we find the seams in one direction, we repeat the process in the other direction, but we mask out every diagonal edge that was already used by any of the first direction seams. This guarantees that the seams in the second direction will be consistent with the first direction (Figure 17).