Visual Media Retargeting

Ariel Shamir
The Interdisciplinary Center, Herzliya

Olga Sorkine
New York University

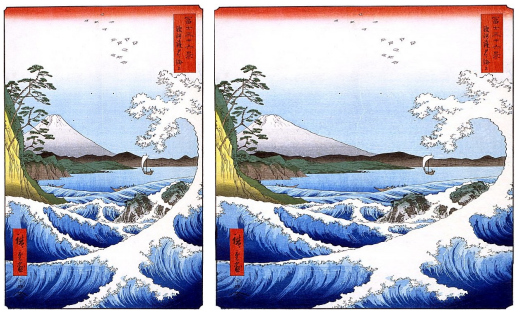


Figure 1: *Retargeting of a print by the Japanese master Utagawa Hiroshige*

Keywords: media retargeting, seam carving, image warping, visual saliency, temporal coherence

1 Introduction

In recent years, many content aware methods for media manipulations have gained popularity. Images and videos are analyzed, segmented, and semantic information is extracted to assist many manipulation algorithms. One of the problems that drew much attention is media retargeting. Due to the increase in the variety of commonly used display devices, and the prevalent use of mobile devices as available means for media intake, media needs to be adapted to different resolutions and aspect ratios. This problem further increases with the explosion of image and video content on the web. One would like to be able to present a feature film on a small iPod, cellular phone or pocket computer, or show photographs on projected presentation systems.

Simple scaling and cropping do not provide satisfactory results since they mostly consider the geometric constraints of the output display alone. Recently, several works have presented *content aware retargeting* methods. These works can be classified into two basic approaches: discrete approaches such as seam carving [Avidan and Shamir 2007; Rubinstein et al. 2008; Rubinstein et al. 2009] remove pixels judiciously to preserve media content, while continuous solutions [Gal et al. 2006; Wolf et al. 2007; Wang et al. 2008; Zhang et al. 2008] optimize a mapping (warping) from the source media size to some target size using several types of constraints to protect media content. In effect, these methods utilize the structural and semantic information in the input media to achieve better resizing results. In fact, retargeting methods also impact many other media applications such as object removal, composition and layout control, compaction, temporal manipulations, summarization and abstraction of media.

The goal of this course is to present the basic problem of media retargeting and detail the different methods devised recently to solve it. We will start with a short overview of image and video representation and concentrate on the different view points of media as a discrete entity (pixels, graphs) or as a sampling of a continuous entity (signal). We will then present the common pipeline for resizing, used in both discrete and continuous methods. This includes first extracting some importance or saliency maps from the media, and then using this information while applying the different retargeting operators. We will present several ways to define importance maps that use spatial information in images and also temporal information in video.

The basic seam carving approach on images uses dynamic programming to find the best (least important or least noticeable) seam to be removed. There are several ways to define the best seam using just simple image gradients to reduce artifacts, or combining high level information such as face detectors or user constraints. Moving from 2D images to 3D video cubes, the basic dynamic programming approach can no longer be used. Dynamic programming is thus replaced by a graph cut algorithm. In this course we will cover both image and video retargeting and show both types of methods based on dynamic programming and graph-cut, show how to combine reduction and enlarging, in both dimensions and show advanced issues such as seam carving in gradient domain, using seam carving for object removal and more.

The warping approach to visual media retargeting views the image/video as a continuous domain and computes a warping of that domain. Each point in the domain is assigned a local transformation; important regions are constrained to undergo a shape-preserving transformation, as much as possible (i.e., a rigid or similarity transformation), whereas less salient areas are allowed to deform more. The computation of the local transformations is done by a global optimization process that minimizes the total deformation energy. The problem can be discretized at different resolutions, trading accuracy for speed. Warping can be utilized for both image and video retargeting; in the case of video, the temporal coherence constraints need to be designed differently from spatial smoothness constraints. Note that, different from seam-carving, continuous warping does not remove image content but rather redistributes its density.

We will discuss the pros and cons of both approaches in the course and devote the last part to more advanced issues such as combining several techniques together.

2 Representations

A digital image $\mathbf{I}$ is defined by a grid of pixels with n rows and m columns whose values contain numbers that encode color information (either gray level or red, green and blue (RGB) values – see Figure 2). A video can be seen as an ordered sequence of digital images and often it is represented by a 3D grid or "video cube". If the size of the frame is $n \times m$ and there are t frames, this cube is created by stacking the video frames one after the other to an $n \times m \times t$ cube.

Although the inherent representation is discrete, digital images are often viewed as a sampling of some continuous signal. This view is also based on the acquisition process of digital images today that utilizes sampling devices for the light coming through the lens of a camera. This view is often used for filtering, noise reduction and other image manipulations. In this course the two basic approaches for retargeting can be distinguished by their view of an image either as a discrete entity or as a sampling of a continuous signal. This in turn affects the particular algorithms used: discrete algorithms such as dynamic programming and graph-cut, or continuous algorithms such as warping.

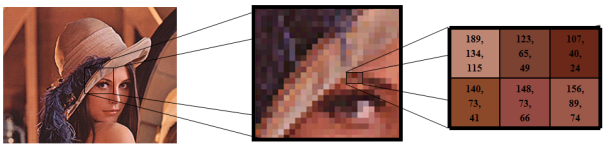


Figure 2: *A digital image as a 2D discrete grid of pixels. In this case the cells contain 3 values of RGB color.*

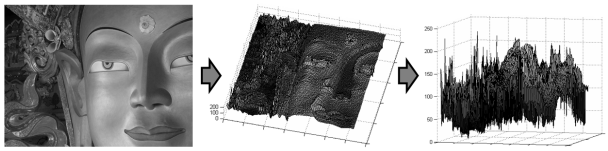


Figure 3: *A digital image as a sampling of a continuous function.*

3 Problem Statement

For simplicity we will begin our discussion with images. The retargeting problem can be stated as follows. Given an image $\mathbf{I}$ of size $n \times m$, and a new size $n^* \times m^*$ we would like to produce a new image $\mathbf{I}^*$ of size $n^* \times m^*$ which will be a good representative of the original image $\mathbf{I}$. However, to date, there is no clear definition or measure as to the quality of $\mathbf{I}^*$ being a good representative of $\mathbf{I}$. In loose terms there are three main objectives for retargeting:

1. The important *content* of $\mathbf{I}$ should be preserved in $\mathbf{I}^*$.
2. The important *structure* of $\mathbf{I}$ should be preserved in $\mathbf{I}^*$.
3. $\mathbf{I}^*$ should be free of visual *artifacts*.

It is clear that the definition of *important* can be subjective and can depend on the application in mind. Indeed, different works define different importance functions on images that contain both low level visual cues such as image edges and high level cues such as people's faces. However, the common approach taken by all media retargeting algorithm is composed of two steps:

- The first step is the definition of an importance map and other constraints on the original media being retargeted.
- The second step applies some operator to the media to change the size while taking into consideration the importance map and its constraints.

In the following we will first discuss different saliency measures and then describe various retargeting operators.

4 Saliency Measures

For the purpose of image retargeting, a visual saliency measure should identify important image regions that should remain intact while the image's aspect ratio is altered. We can define an image importance map as a map $S : \mathbf{I} \rightarrow [0, 1]$. This map ranks every pixel in an image according to its visual importance (1 being most salient). There are different visual cues that affect our perception of visual content, some of them are low-level features such as edges (detected by, e.g., high local intensity gradients), and others are high-level, for example faces, structures and symmetries.

A common low-level saliency measure which we will call E_1 is in fact an edge map, computed simply from the magnitude of intensity

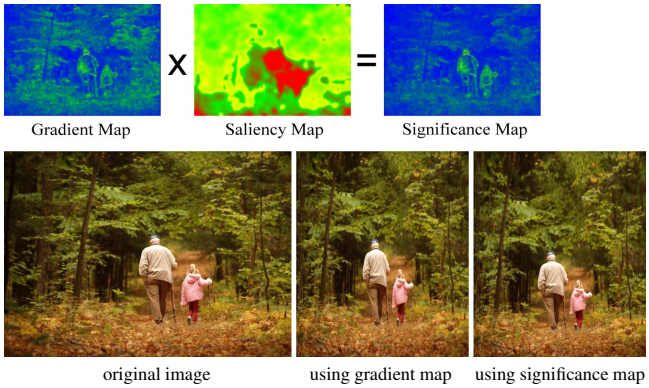


Figure 4: *Various importance maps. The top row shows the magnitude of gradients map and Itty's saliency map [1998]; these two are combined by multiplication to obtain a visual importance in [Wang et al. 2008]. The bottom row displays retargeting results with the gradient map along, and using the combined map; adding the multiresolution saliency measure helps filter out spurious gradients in the foliage area and leads to a better result.*

gradients (and then normalized to $[0, 1]$):

$$E_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|. \quad (1)$$

The rationale behind using edges is to preserve strong contours in the image, as those usually delineate prominent objects. Furthermore, human vision is more sensitive to edges and this importance map gives high importance value to edges and low importance value to smooth areas. Instead of using L_1 metric some works have used L_2:

$$E_2(\mathbf{I}) = \sqrt{\left(\frac{\partial}{\partial x}\mathbf{I}\right)^2 + \left(\frac{\partial}{\partial y}\mathbf{I}\right)^2}. \quad (2)$$

Other edge detection mechanisms such as Canny edge-detector could also be used. However, strong edges might sometimes appear also in noisy regions which are not necessarily salient. A more elaborate image saliency measure was developed, for instance, by Itty et al. [1998]. They build a multiresolution pyramid of the image and look for significant intensity and color changes on all levels, combining those into a single, high-resolution map. Other possible low-level measures are Harris-corners measure [Harris and Stephens 1988], Histograms of gradients (HoG) [Dalal and Triggs 2005] and entropy, which all measure some properties of a window around each pixel. Not surprisingly, it was shown by Wang et al. [2008] that combining the gradient map and the saliency map together (by multiplication) has benefits over using just a single measure, since Itty's saliency filters out noisy gradients (see example in Figure 4).

High-level important features can sometimes also be detected automatically, for example, human faces are often detected using [Viola and Jones 2004]. In video high-level features also include temporal phenomena, such as scene cuts [Zabih et al. 1995], and camera and object motions [Wolf et al. 2007; Wang et al. 2009].

Some of these elements can be computed automatically, yet for each particular input the user may have a specific idea about which content is important, and this may not necessarily correspond to the automatic measures. A possible solution is to measure the eye gaze of real viewers [DeCarlo and Santella 2002] and infer the more important (salient) parts in the media [Santella et al. 2006]. Many recent works also allow some direct degree of user intervention

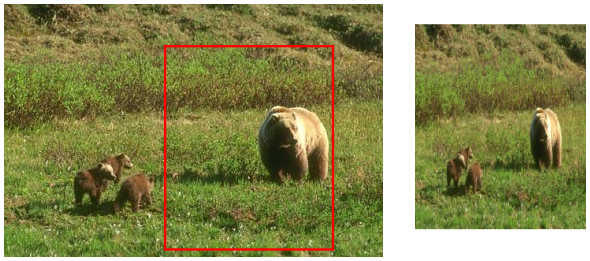


Figure 5: *Cropping to a given size may discard important information in the image. Homogeneous scaling can reduce the objects size and may also create distortions if it applied in a non-uniform manner to account for aspect ratio changes.*

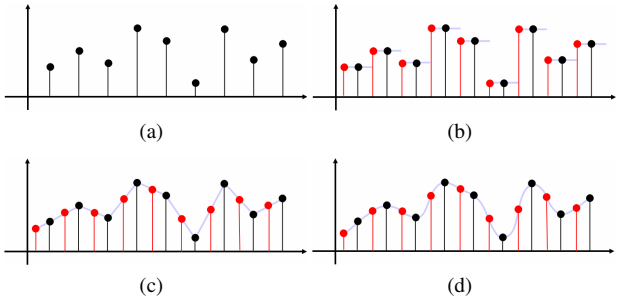


Figure 6: *Reconstruction and super-sampling of a 1D signal using piecewise-constant (b), linear (c) and cubic (d) interpolations respectively. On images the interpolation kernels are 2D.*

either by designating specific structural constraints that should be preserved [Krähenbühl et al. 2009] or even directly drawing on the importance map to assign importance values to pixels [Avidan and Shamir 2007].

Preserving all the important features at once may be a difficult, if not infeasible task, as the different constraints might conflict with each other. The proper weighting of all the different measures against each other is a challenge and highly depends on the semantics of the image content itself. Therefore, as mentioned before, requiring some amount of user intervention may be unavoidable at times for acceptable results.

5 Cropping

The cropping operator can be considered the simplest resizing operator since it simply extracts a window of the desired size $n^* \times m^*$ from the original image. Obviously, this can be done only if $m^* \leq m$ and $n^* \leq n$. Many times cropping is done manually, i.e. the user chooses a window of the desired size inside the original image. However, given some importance map one can also search automatically for the window of size $n^* \times m^*$ that will contain the most important parts [Fan et al. 2003; Santella et al. 2006]. This cannot always be done as seen, for instance, in Figure 5.

A possible extension of cropping to enlarge the image size is to pad the frame with "black" pixels. This is sometimes called *letterboxing* and is often used to adapt old TV frottage to new screens and vice verse. In terms of the three objectives for retargeting, cropping mostly preserves the structure of the image and does not produce artifacts apart from cutting the image at the fringes. The main downside of using cropping is that content is always lost from the image and the composition can be damaged.

6 Scaling

Scaling is define by a homogeneous map between the pixels of the original image $\mathbf{I}$ and the pixels of the target image $\mathbf{I}^*$. By *homogeneous* we mean that all pixels undergo the same mapping function. If the mapping function is the same for the horizontal and vertical direction we call the scaling *uniform*. One may try to use forward-mapping to transform the pixels of the original image to their new positions as follows:

$$\mathbf{I}(x,y) \rightarrow \mathbf{I}^*(\lfloor x \cdot \frac{m^*}{m} \rfloor, \lfloor y \cdot \frac{n^*}{n} \rfloor)$$

However, forward mapping leads to many-to-one mapping as well as to empty pixels in the target image. A solution to this is to use backward mapping as follows:

$$\mathbf{I}^*(x,y) = \mathbf{I}(\lfloor x \cdot \frac{m}{m^*} \rfloor, \lfloor y \cdot \frac{n}{n^*} \rfloor)$$

However, simple backward mapping may lead to loosing some of the original pixels or to pixels being duplicated. A better solution is based on the continuous view of images. First, a model of the original signal is defined and reconstructed using some interpolation function of the pixels of $\mathbf{I}$, the original image. Consequently, this model function is sampled at the desired target resolution and size to create the target image $\mathbf{I}^*$ (see example for up-sampling a 1D signal in Figure 6). The most common approach for scaling uses bi-cubic interpolation of the original image pixels to model the signal.

Still, even homogeneous bi-cubic scaling can create artifacts such as blockiness and aliasing. Moreover important objects could be scaled beyond the point of recognition. More serious artifacts and distortions appear when non-uniform scaling is applied to accommodate changes in the aspect ratio of the image (Figure 5).

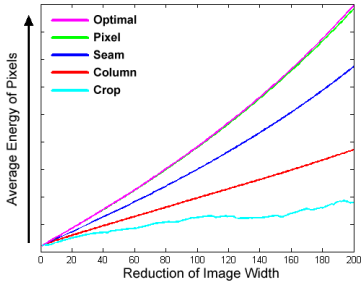
7 Seam-Carving Images

7.1 Backward Energy

Assume we need to reduce the width of an image. Later, we will show how to use seam carving for changes of size in both directions and also for extension of the image size. The seam carving approach to content-aware resizing is to remove some pixels from the image in a judicious manner [Shamir and Avidan 2009]. Therefore, the key question is how to choose the pixels to be removed? Intuitively, the goal is to remove unimportant pixels using some importance map on I. For example, unnoticeable pixels that blend with their surroundings could be good candidates for removal. One can use for instance E_1 edge map from Equation 1.

Given this, or similar, energy function one can think of several strategies to achieve reduction in width. For instance, an optimal strategy to preserve energy (i.e., keep as many pixels with high energy value) would be to remove the pixels with lowest energy in ascending order. However, this destroys the rectangular shape of the image, because we may remove a different number of pixels from each row (see Figure 7(e)). If we want to prevent the image shape 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 7(d)). To preserve both the shape and the visual coherence of the image we can use some automatic cropping mechanism. We can look for a sub-window of the given target-width where the sum

(a) Original Image and its Energy



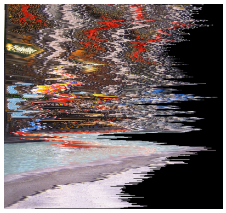
(b) Average Energy

(c) Crop

(d) Column

(e) Seam

(f) Pixel



(g) Optimal

Figure 7: *Results of 5 different strategies for reducing the width of an image by removing pixels. (a) the original image and a color mapping of the 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 curve of each strategy.*

of its pixels energy is the highest from all possible sub-windows (Figure 7(a)). Another possible strategy somewhat between removing pixels and cropping is to remove whole columns with the lowest energy. Still, artifacts may appear in the resulting image (Figure 7(b)). Therefore, we seek 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 the use of seams (Figure 7(c)).

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

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

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 8). Similarly, if y is a discretely continuous function $y : [1, \ldots, m] \rightarrow [1, \ldots, n]$, then an image *horizontal seam* is:

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

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. The visual impact is noticeable only along the path of the seam, leaving the rest of the image intact.

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

$$s^* = \arg\min_{\mathbf{s}} E(\mathbf{s}) = \arg\min_{\mathbf{s}} \sum_{i=1}^{n} E(\mathbf{I}(s_i)) \qquad (3)$$

Although there seems to be an exponential number of possible seams, because each path going through a pixel can come only from one of three neighboring pixels, the optimal seam can in fact be found using dynamic programming in linear complexity. The algorithm to find the optimal seam is composed of two steps. The first step creates the cumulative energy map M and the second step backtracks on the map from the last row (column) to the first to find the seam path. For example, to create the vertical M we traverse the image from the second row to the last row and compute the cumulative minimum energy M for the three possible connected seams at each entry $M(i, j)$:

$$M(i,j) = E(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1)) \qquad (4)$$

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 (Figure 8). 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 it and repeat the process for n times. It is worth noting that this approach resembles the dynamic shortest paths problem [Roditty and Zwick 2004], 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.

7.2 Forward Energy

Choosing to remove the seam with the least amount of energy from the image (Equation 4), works in many cases, but ignores energy that is *inserted* into the resized image *after* the seams are removed. This inserted energy is due to new intensity edges created by previously non-adjacent pixels that become neighbors once a seam is removed. Following this observation, we formulate the *forward*

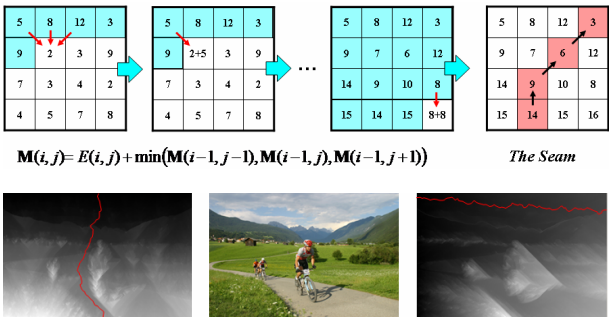


Figure 8: *Top: example of creating the vertical accumulated energy map M and backtracking to find the seam. Bottom: visual representation of the vertical and horizontal maps of an image (middle) and the optimal seams found.*

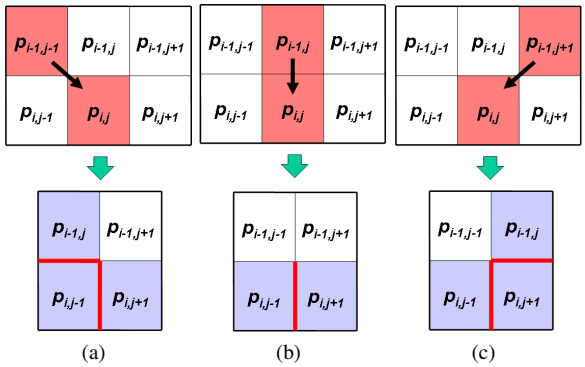


Figure 9: *Calculating the three possible vertical seam step costs for pixel $p_{i,j}$ using forward energy. After removing the seam, new neighbors (in gray) and new pixel edges (in red) are created. In each case the cost is defined by the forward difference in the newly created pixel edges. Note that the new edges created in row $i-1$ were accounted for in the cost of the previous row pixel.*

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

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

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:

$$MF(i,j) = P(i,j) + \min \begin{cases} MF(i-1,j-1) + C_L(i,j) \\ MF(i-1,j) + C_U(i,j), \\ MF(i-1,j+1) + C_R(i,j) \end{cases} \quad (5)$$

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 10 shows a comparison of the results using the the two formulations.

7.3 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 $\mathbf{I}$. 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 11(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 11(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 11 (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 11 (f)). Another example of enlarging an image using seam insertion is found in Figure 1.

7.4 Retargeting in Both Dimensions

Image retargeting can change the image size in both directions, such that an image $\mathbf{I}$ of size $n \times m$ can be retargeted to size $n^* \times m^*$, where both $m^* \neq m$ and $n^* \neq n$. Using seam carving, this raises the question of what is the correct order of seam removal (or insertion)? Vertical seams first? Horizontal seams first? Alternate between the two? Or more generally, is there an optimal order of seam removal (or insertion)?

We define an objective function using the seam costs and search for the optimal order of seam removal (insertion) by minimizing this function:

$$\min_{\mathbf{s^x},\mathbf{s^y},\delta} \sum_{i=1}^{k} E(\delta_i \mathbf{s_i^x} + (1-\delta_i)\mathbf{s_i^y}) \quad \text{s.t.} \quad (6)$$
$$k = r + c, \quad r = (m - m^*), \quad c = (n - n^*)$$
$$\delta_i \in \{0,1\}, \quad \textstyle\sum_{i=1}^{k} \delta_i = r, \quad \sum_{i=1}^{k} (1-\delta_i) = c$$

δ_i is used as a parameter that determines if at step i we remove (insert) a horizontal or vertical seam.

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.

Figure 10: *A comparison of results for reduction and expansion of the car image (leftmost) using least cost seam of Equation 4 (left in each pair) and least inserted cost seams of Equation 5 (right in each pair).*

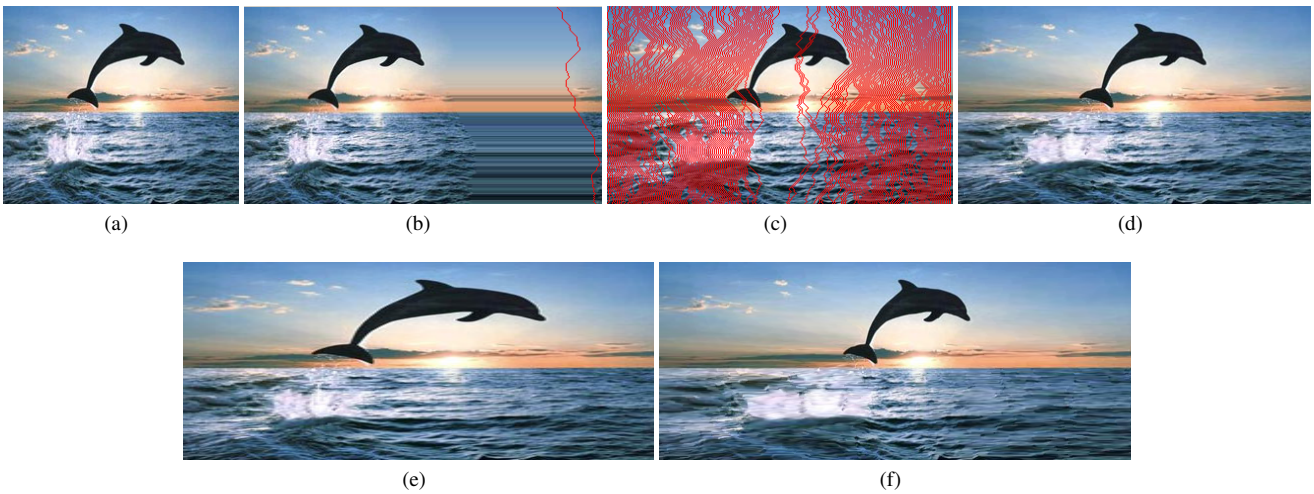


Figure 11: *Seam insertion: finding and inserting the optimum seam on an enlarged image will most likely insert the same seam again and again as in (b). Inserting the seams in order of removal (c) achieves the desired 50% enlargement (d). Using two steps of seam insertions of 50% in (f) achieves better results than scaling (e).*

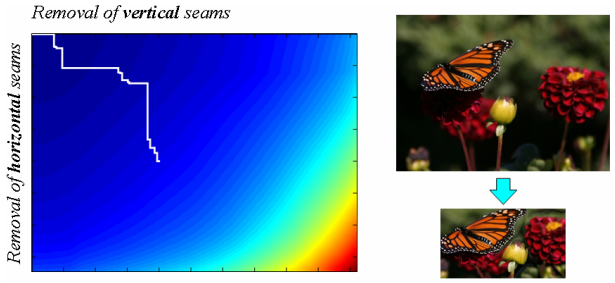


Figure 12: *Optimal order retargeting: On the left is the transport map* $\mathbf{T}$ *of the retargeted image on the right. Given a target size, we follow the optimal path from the target size to* $(0,0)$ *(on the upper left corner) and apply either a vertical or a horizontal seam removal to obtain the retargeted image (see example of white path on* $\mathbf{T}$*).*

That is, entry $T(r,c)$ holds the minimal cost needed to obtain an image of size $n-r \times m-c$. Again, there are an exponential number of possible ordering of vertical and horizonal seams. However, we compute $\mathbf{T}$ greedily 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 $n-r+1 \times m-c$:

$$\mathbf{T}(r,c) = min(\mathbf{T}(r-1,c) + E(\mathbf{s^x}(\mathbf{I_{n-r-1 \times m-c}})), \mathbf{T}(r,c-1) + E(\mathbf{s^y}(\mathbf{I_{n-r \times m-c-1}}))) \quad (7)$$

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 12).

8 Seam-Carving Video

8.1 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

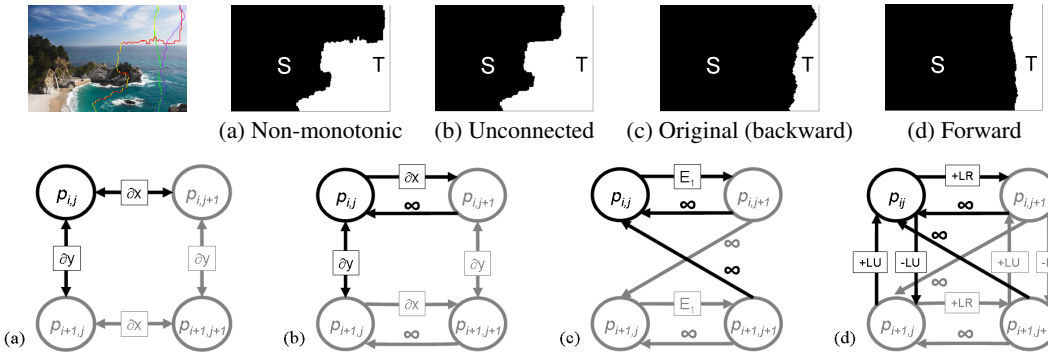


Figure 13: *Minimum cut on the waterfall image (top left) for various graph constructions. The seam is composed of the pixels to the left of the cut. The different graph constructions are illustrated by four nodes representing four pixels in the image. The actual image graph is created by tiling these 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 7.2.*

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 [Kwatra et al. 2003; Schödl et al. 2000; Wang et al. 2004; Wang et al. 2005], 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, e.g. cuts whose intersection with each video frame (or image) will produce a valid seam, that is, 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.

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.

8.2 Graph Cut on Images

For simplicity we will formulate the seam carving operator as a minimum cost graph cut problem assuming we search for vertical seams in an image. Later we will extend the discussion for video. 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. 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).

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. Note that 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. The optimal seam is defined by the *minimum cut* which is the cut that has the minimum cost among all valid cuts.

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: $\partial x(i,j) = |\mathbf{I}(i,j+1) - \mathbf{I}(i,j)|$ or in the vertical: $\partial y(i,j) = |\mathbf{I}(i+1,j) - \mathbf{I}(i,j)|$. Under this formulation, Figure 13(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. These infinity arcs impose the monotonicity constraint as follows:

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 14(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.

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 as above, we can show that this construction also imposes the connectivity constraint (see Figure 14(b-c)).

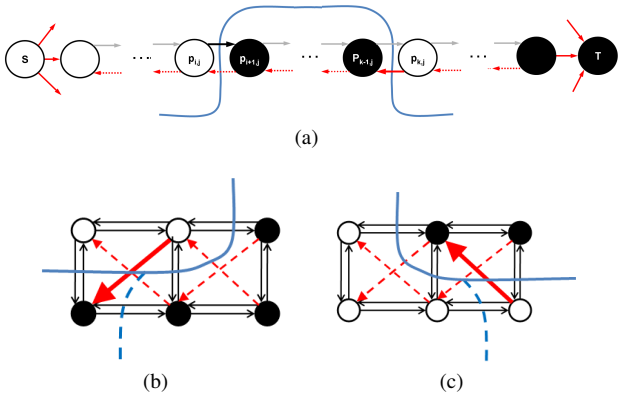


Figure 14: *Using infinity edges (red) in the graph construction maintains the seam constraints. Horizontal infinity arcs maintain monotonicity (a) - see details in text. Diagonal infinity arcs maintain connectivity. If the cut skips more than one pixel to the left (b) or right (c) - a diagonal infinity arc from a source node (white) to a target node (black) must be cut.*

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 13(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, or a weight mask scribbled by the user, can be used in any of the graph constructions we present. We simply add the pixel's energy to the horizontal arc going out of the pixel.

To define the forward energy cost (Section 7.2) in a graph, 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 13(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 $+LR = |\mathbf{I}(i,j-1) - \mathbf{I}(i,j+1)|$ 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 9(b)), there are no new vertical edges so no energy is inserted. Remember that all nodes to the left of the cut must be labeled S and all nodes on 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 a right-oriented cuts (Figure 9(a)), and only downward vertical arcs will be counted in a left-oriented cuts (Figure 9(c)). Hence, we assign the difference between the **L**eft and **U**p neighbors $+LU = |\mathbf{I}(i-1,j) - \mathbf{I}(i,j+1)|$ to the upward vertical arc between $p_{i,j}$ and $p_{i-1,j}$, and the weight $-LU = |\mathbf{I}(i-1,j) - \mathbf{I}(i,j-1)|$ to the downward vertical arc between $p_{i-1,j}$ and $p_{i,j}$ ($-LU$ means the difference between the **L**eft and **U**p neighbors with respect to the end point of the arrow).

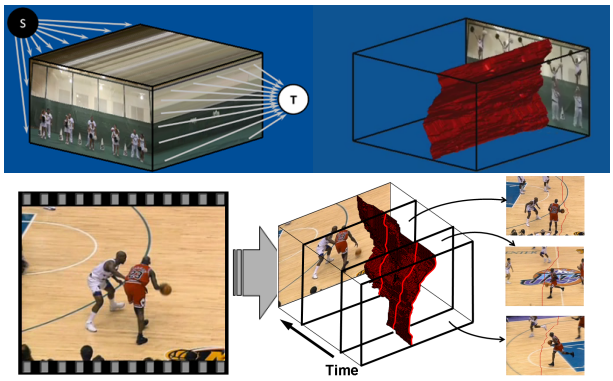


Figure 15: *Top: the construction of a graph on a video cube guarantees the graph-cut will find a connected and monotonic manifold in the volume. Bottom: The intersection of the manifold with each video frame defines the seams on the frame.*

8.3 Graph Cut on 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 15). 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 Jolly 2001]. 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 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.

9 Warping

Image warping views the image as a continuous domain and performs a continuous geometric deformation to fit the image into the new desired shape (an $n^* \times m^*$ rectangle). The image content (the signal) conceptually undergoes a continuous transformation and is eventually discretized by sampling the continuous model (typically the model uses bi-cubic or bi-linear interpolation). Homogeneous scaling discussed before is a simple example of image warping for retargeting purposes: it is called homogeneous because every point within the image domain undergoes the same transformation (scaling by some factor along the x and y axes). This kind of warp does not regard the image content at all, however, and all the objects in the image are thus equally distorted. Recent image warping approaches suggest designing non-homogeneous warps that are adapted to the image content. More specifically, non-homogeneous warps take the image importance map as input and attempt to preserve the shape of the salient parts while allowing more distortion in regions of low importance. To obtain such a warping function, all recent methods employ variational formulations, namely: an objective functional is designed, which measures, roughly speaking, how well the warp preserves the image content, and then the warping function that minimizes the objective functional is found by a numerical optimization process.

We adopt the notation from [Krähenbühl et al. 2009] and [Wang et al. 2008] and define the concrete setup of the variational optimization, and then discuss the details of the various warping approaches.

Setup and notation. Let us denote the image importance map by $S : \mathbf{I} \to [0, 1]$ (see Section 4). We are looking for a warping function $F : \mathbb{R}^2 \to \mathbb{R}^2$ that transforms our input image $\mathbf{I}$ into the desired output image $\mathbf{I}^*$ with the desired dimensions $n^* \times m^*$. We denote the horizontal component of F by F_x and the vertical one by F_y such that $F(x, y) = (F_x(x, y), F_y(x, y))$. The warp should transform the boundary rectangle of the input image into the new dimensions, so assuming that we attach the coordinate $(0, 0)$ to the bottom left corner of the image and the coordinate (m, n) to the top right corner, F should satisfy

$$F_x(0, \cdot) = 0, \;\; F_x(m, \cdot) = m^*, \;\; F_y(\cdot, 0) = 0, \;\; F_y(\cdot, n) = n^*.$$

The above equations are the so-called boundary constraints; the optimization will search for the best possible F that satisfies those constraints. The "best" behavior is typically defined in terms of the local behavior of F, i.e., its Jacobian

$$J_F(x, y) = \begin{pmatrix} \frac{\partial F}{\partial x} \frac{\partial F}{\partial y} \end{pmatrix} = \begin{pmatrix} \partial F_x/\partial x & \partial F_x/\partial y \\ \partial F_y/\partial x & \partial F_y/\partial y \end{pmatrix}.$$

The Jacobian $J_F(x, y)$ is essentially a linear transformation (scaling, shearing, etc.) that best approximates F in a small neighborhood around the point (x, y). Ideally, the Jacobian would equal the identity matrix everywhere, which would mean that F does not distort the image at all. However, this is of course impossible if the size of the image is to be changed, so a more flexible objective functional needs to be defined. For example, one would like the regions of high importance to be distorted the least while sacrificing other regions. This means we could ask the Jacobian of each point (x, y) to be as close as possible to the identity matrix I in the least squares sense, weighted by the importance $S(x, y)$. This results in the following objective functional:

$$E(F) = \int_{x=0}^{m} \int_{y=0}^{n} S(x, y) \left\| J_F(x, y) - I \right\|^2 dxdy\,, \qquad (8)$$

where $\| \cdot \|$ is the Frobenius matrix norm. Written out more explicitly:

$$\|J_F(x, y) - I\|^2 = \left(\frac{\partial F_x}{\partial x}(x, y) - 1 \right)^2 + \left(\frac{\partial F_y}{\partial x}(x, y)) \right)^2 + \left(\frac{\partial F_x}{\partial y}(x, y) \right)^2 + \left(\frac{\partial F_y}{\partial y}(x, y) - 1 \right)^2.$$

There exist many variations on the particular terms used in the objective functional, as we will see below, but they all typically involve the partial derivatives of F. The goal is to find the optimal F:

$$F = \underset{F}{\operatorname{argmin}}\, E(F) \;\text{ subject to the boundary constraints.}$$

Discretization. So far we have formulated everything in a continuous manner, but in order to compute the optimal warp F it is usually necessary to discretize the problem so that standard numerical optimization methods can be applied. This is fairly easy thanks to the regular structure of the image domain. Typically a grid (quad) mesh is superimposed against the image and the discrete objective functional is defined using the mesh vertices. The mesh can have arbitrary resolution: it can coincide with the pixel grid (or even be finer), but often for the sake of efficiency a coarser grid is used. Let us denote the grid mesh by $\mathbf{M} = (\mathbf{V}, \mathbf{E}, \mathbf{Q})$ with vertices $\mathbf{V}$, edges $\mathbf{E}$ and quad faces $\mathbf{Q}$, where $\mathbf{V} = [\mathbf{v}_0^T, \mathbf{v}_1^T, ...\mathbf{v}_{end}^T]$ and $\mathbf{v}_i \in \mathbb{R}^2$ denote the initial vertex positions. The vertices and edges form horizontal and vertical grid lines partitioning the image into quads. The problem is then to find the deformed mesh geometry $\mathbf{V}' = [\mathbf{v}_0'^T, \mathbf{v}_1'^T, ...\mathbf{v}_{end}'^T]$, i.e., $F(\mathbf{v}_i) = \mathbf{v}_i'$ for each i. Once the discrete F is computed for the vertices, the image content within each face of the mesh can be reconstructed by interpolation.

It is generally assumed that all the faces of the mesh are square with fixed unit edge length; therefore the partial derivatives of F can be easily discretized using finite differences:

$$\frac{\partial F}{\partial x}(\mathbf{v}_i) = \mathbf{v}_j' - \mathbf{v}_i'\,,$$

where vertex j is the right-hand horizontal neighbor of vertex i, and similarly for the vertical direction:

$$\frac{\partial F}{\partial y}(\mathbf{v}_i) = \mathbf{v}_k' - \mathbf{v}_i'\,,$$

where vertex k is the top vertical neighbor of vertex i.

The importance map S is also discretized and lumped to the mesh vertices. Since usually S is defined on the discrete digital image domain to begin with, if the grid mesh does not coincide with the pixel grid (e.g., the mesh is coarser than the image resolution), then one can take the average of the pixels' importance values in some fixed-size neighborhood of each mesh vertex $\mathbf{v}_i$ to define $S(\mathbf{v}_i)$.

Using such discretization, and replacing integrals with sums, the example objective functional we saw earlier in Eq. (8) would become

$$E(F) = \sum_{\{i,j\} \in \mathbf{E}} S(\mathbf{v}_i) \|(\mathbf{v}_j' - \mathbf{v}_i') - (\mathbf{v}_j - \mathbf{v}_i)\|^2.$$

Explanation: the objective functional would like to keep all the horizontal derivatives equal to $(1, 0)^T$ and all the vertical derivatives equal to $(0, 1)^T$; of course all the original horizontal neighbor vertices $\mathbf{v}_i, \mathbf{v}_j$ satisfy $\mathbf{v}_j - \mathbf{v}_i = (1, 0)^T$, and the vertical neighbors

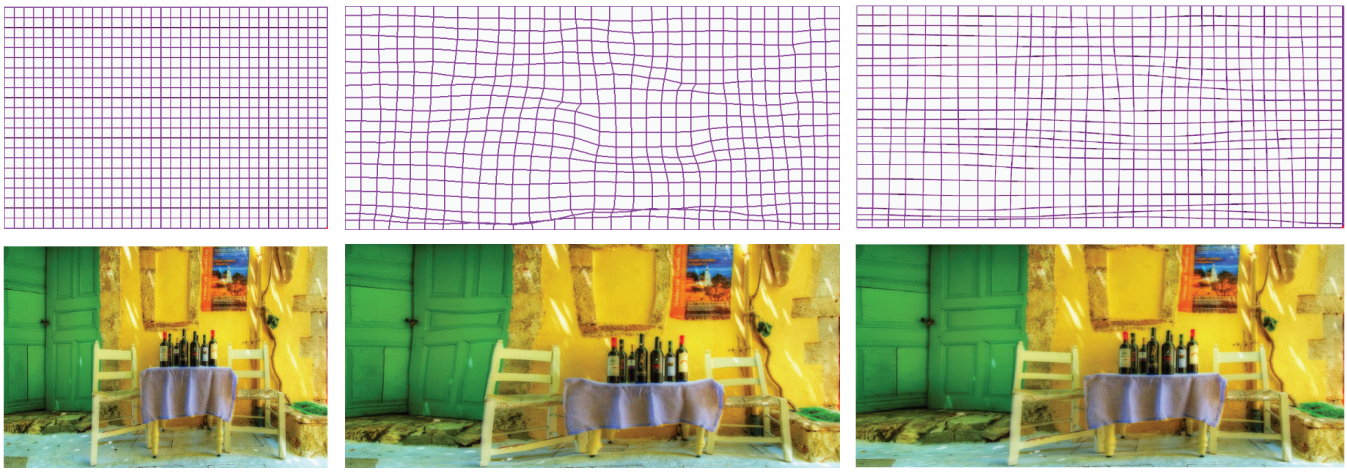


Figure 16: *Demonstration of the effect of the line bending energy term in the scale-and-stretch warping technique of [Wang et al. 2008]. Note how adding the energy term prevents strong grid line bending and produces a less distorted retargeting result.*

have $\mathbf{v}_j - \mathbf{v}_i = (0,1)^T$, so it is convenient to compactly write the energy as above, without distinguishing vertical and horizontal cases, simply as $(\mathbf{v}'_j - \mathbf{v}'_i) - (\mathbf{v}_j - \mathbf{v}_i)$.

In the functional above, the unknowns are the warped vertex locations $\mathbf{v}'_i$; note that the functional in this case is quadratic in the unknowns, and thus we can find the warp by solving a system of sparse linear equations (for more details, see [Gal et al. 2006]). This can be done robustly and quite efficiently with modern numerical solvers; all warp-based retargeting methods try to avoid more involved, nonlinear functionals when possible to keep the computational costs low and the implementation simple.

9.1 Warp-based Image Retargeting

Given the basics described above, let us now detail the various warp-based retargeting methods. Most techniques fall into the general optimization framework described above and differ by the particular objective functionals they formulate.

Gal et al. [2006] were among the first to propose non-homogeneous warps for images. They used a simple binary, hand-drawn importance map S, and the objective functional for the warping function was accordingly divided into two parts: the Jacobian of "important" points should be as close as possible to scaled identity, and the Jacobian of "unimportant" points is allowed to be the standard homogeneous (non-uniform) scaling A that fits the new dimensions of the image. The objective functional can be written as

$$\iint S(x,y)\|J_F(x,y) - sI\|^2 + (1 - S(x,y))\|J_F(x,y) - A\|^2.$$

Gal et al. [2006] noted that uniform scaling of important regions may be beneficial, as it preserves the shape, and thus the image content, yet allows more flexibility in the warp. They simply defined the uniform scaling factor s to be the minimum between the horizontal and vertical scaling induced by the new dimensions of the image (so for instance, if the image is stretched or shrunk just along one dimension, the uniform scaling factor would be 1).

The idea of taking advantage of local scaling was taken further by Wang et al. [2008]. They suggested to use varying uniform scaling factors for each point in the image and find those by optimization (hence naming the technique "optimized scale-and-stretch"). In discrete form, the initial objective functional then has the form

$$\sum_i S(\mathbf{v}_i) \sum_{j \text{ s.t. } \{i,j\}\in\mathbf{E}} \|(\mathbf{v}'_j - \mathbf{v}'_i) - s_i(\mathbf{v}_j - \mathbf{v}_i)\|^2.$$

Both the new mesh vertices $\mathbf{v}'_i$ and the uniform scaling factors s_i are unknowns; they are computed by alternating optimization. Fixing s_i allows to find the optimal warped vertex positions $\mathbf{v}'_i$ by solving a sparse linear system; given the current $\mathbf{v}'_i$ the scaling factors s_i can be then updated, and the iterations repeat until convergence (see the paper [Wang et al. 2008] for details).

In addition to this energy functional, Wang et al. [2008] also suggest adding a grid line bending energy: only having the scaling functional above may cause the grid lines to arbitrary bend, so to limit the "wild" behavior it is necessary to encourage the grid lines to keep their original orientation while allowing the length to change (see Figure 16). The bending energy is

$$\sum_{\{i,j\}\in\mathbf{E}} \|(\mathbf{v}'_j - \mathbf{v}'_i) - l_{ij}(\mathbf{v}_j - \mathbf{v}_i)\|^2. \tag{9}$$

Here, again, the length factors l_{ij} are unknowns and are iteratively found during the alternating optimization. In fact, the lengths l_{ij} can be written as nonlinear expressions in $\mathbf{v}'_i$:

$$l_{ij} = \|\mathbf{v}'_i - \mathbf{v}'_j\| / \|\mathbf{v}_i - \mathbf{v}_j\|.$$

By employing the alternating optimization strategy, i.e., fixing the length factors, solving for the vertex positions and then updating the length factors by simply applying the formula above with the current vertex positions $\mathbf{v}'_i$, it is possible to avoid a complex nonlinear optimization, and robustly reach a reasonable solution within just a few iterations.

In the work of Wolf et al. [2007], scaling is performed along one dimension only (i.e., the image points are encouraged to move in the direction of the stretch but not orthogonal to it); this is achieved by formulating an objective functional that asks w.l.o.g. the horizontal derivatives of the warp to equal 1 while smoothing the vertical derivatives. Their objective functional thus has the form

$$\iint S(x,y)\left(\frac{\partial F_x}{\partial x}(x,y) - 1\right)^2 + w\left(\frac{\partial F_y}{\partial x}(x,y)\right)^2 dxdy.$$

original image [Rubinstein et al. 2008] [Wolf et al. 2007] [Wang et al. 2008]

Figure 17: *Comparison of warp-based and discrete retargeting techniques.*

Wang et al. [2008] note, however, that this formulation may often lead to self-intersections and is sometimes sub-optimal in terms of effective use of the image domain, because it does not allow features to scale down in both directions.

In a recent work, Krähenbühl and colleagues [2009] note that allowing different parts of the image to scale differently can often lead to significant changes of proportions, so they opt to replace the local scaling factors s_i by one global scaling factor s, which is also found by alternating optimization. They also propose two additional energy terms that improve the appearance of edges:

$$\iint S_e(x,y)\left(\left(\frac{\partial F_x}{\partial y}\right)^2+\left(\frac{\partial F_y}{\partial x}\right)^2\right)dxdy,$$

$$\iint S_e(x,y)\left(\left(\frac{\partial F_x}{\partial x}-1\right)^2+\left(\frac{\partial F_y}{\partial y}-1\right)^2\right)dxdy.$$

Here, $S_e(x,y)$ is an edge saliency map (computed by running, e.g., a Sobel operator); the first term prevents bending of features (similar to [Wolf et al. 2007]) and the second term prevents edge blurring by enforcing similar image gradients on feature edges.

In addition to the automatic warping, the work by Krähenbühl et al. [2009] explores various interactive control mechanisms for the retargeting process, allowing users to interactively mark objects in the image (or video) and influencing their location, as well as manual markup of lines and edges, which are then constrained to remain straight. Note that automatic edge bending energies (like in Eq. (9) or above) can only prevent local bending of features, but do not prevent global distortion of feature lines. A manual line constraint is created by drawing a line represented as $l : sin(\alpha)x + cos(\alpha)y + b = 0$. Each image point is then associated with a value $c(x,y)$ that indicates the coverage percentage by the line; line preservation energy term can be simply formulated as

$$\iint c(x,y)\left(\sin(\alpha)F_x(x,y)+\cos(\alpha)F_y(x,y)+b\right)^2 dxdy.$$

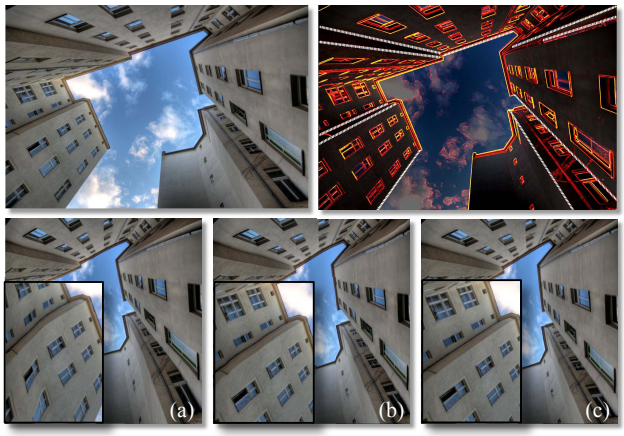


Figure 18: *Demonstration of feature edges retargeting. The top row shows the original frame (left) and the edge saliency map S_e (right) used in [Krähenbühl et al. 2009]. Manually added global line constraints are marked in white. The bottom row shows different retargeting results: (a) Wang et al. [2008], (b) Krähenbühl et al. [2009] with automatic line bending energy terms only, (c) Krähenbühl et al. [2009] using the additional manual line constraints. Images taken from [Krähenbühl et al. 2009].*

Note that initially the line parameters α and b are set as in the original image, but they are updated after each iteration in an alternative optimization manner, just like the scaling parameter. See Figure 18 that shows the effect of the automatic and manual line constraints.

Some warp-based image retargeting results are shown in Figure 17. It is interesting to compare the results of the different warping techniques, and the discrete methods as well: generally, warp-based retargeting does not suffer from discontinuity artifacts typical for seam carving, but the trade-off may be more significant distortion of

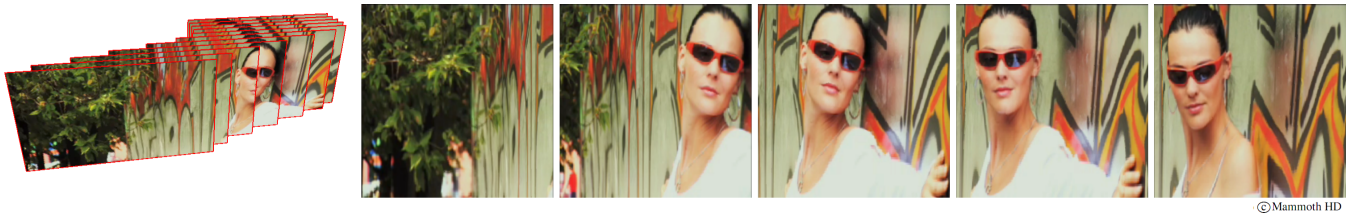


Figure 19: *A video retargeting example from [Wang et al. 2009]. First, the video frames are aligned into one common coordinate system, eliminating camera motion. The video cube is then warped while respecting object motion.*

image objects, smearing artifacts and, when spatially-varying scaling is involved, also change in relative proportions.

9.2 Warp-based Video Retargeting

As mentioned before, video retargeting can be viewed as a generalization of image retargeting, where all the video frames are stacked in a space-time cube. Warp-based video retargeting methods will thus warp the cube to obtain the video of the desired dimensions. We will denote the warping function by $F(x, y, t)$, adding the third (time) dimension. As with images, video warping should satisfy some boundary constraints: spatial constraints on the boundary of each frame ensure the proper output size, and in addition, the time should not be warped at all, so each frame retains its time stamp $F_t(x, y, t) = t$ (or in other words, the warp does not affect the temporal dimension).

As we have seen, most recent image warping techniques employ global optimization to obtain the warping function. Direct generalization of this to video is impractical for several reasons: first, global optimization on the space-time cube is very expensive, since it requires solving linear or nonlinear systems for $n \times m \times t$ variables, where $n \times m$ is the resolution of each video frame (or the superimposed grid mesh) and t is the number of frames. Secondly, a global optimization that involves all the video frames at once might not even make sense, since temporally distant frames have little to no effect on each other and thus should not be tied in one common objective functional. Therefore, at least scene-cut detection is performed to break the video into individual scenes. This can be done either manually or using an automatic method such as [Zabih et al. 1995].

To retarget an individual video sequence in a meaningful way, one must consider temporal coherence in addition to the spatial behavior of each individual frame. Initially, temporal coherence was dealt with by simply asking temporally-adjacent image points to be warped (or transformed) similarly, i.e., $F(x, y, t)$ should be close to $F(x, y, t \pm 1)$. Wolf et al. [2007] formulate a temporal energy term for the global optimization simply as

$$\iiint \left\| \frac{\partial F}{\partial t}(x, y, t) \right\|^2 dxdydt.$$

However, for videos with object and/or camera motion, this type of objective functional does not work well, since the same spatial location in adjacent frames is not necessarily occupied by the same object, so that constraining temporally adjacent pixels to undergo similar transformations might actually distort the video. Krähenbühl et al. [2009], aiming at a real-time restargeting system that works in a streaming setting, propose to deal with the problem by temporally filtering the importance maps using a small look-ahead window (of approximately five frames). Temporally "smearing" the importance map helps predicting the future appearance of salient objects, and indirectly temporally smoothes the retargeting result. Yet, in this approach each video frame is still resized individually; if more computationally-intensive video processing is allowed (offline or in a preprocess), temporal coherence can be more substantially handled by looking at the entire scene.

Such global processing was recently suggested by Wang and colleagues [2009]: they first align all the video frames (in a given scene) in a common coordinate system by extracting and matching SIFT features, which effectively removes camera motion (see Figure 19, left). Then, individual moving objects are roughly detected and a motion saliency map is built. The warp is applied to the aligned video frames using the scale-and-stretch framework of [Wang et al. 2008], but with additional temporal coherence energy terms: the scaling factors of moving objects are constrained to be similar across frames, and generally the warp of the space-time cube is governed by a combination of the motion saliency map and spatial importance map, blended across several neighboring frames. Wang et al. [2009] show that it is possible to solve for the warping function using a sliding window approach, such that the optimization is relatively efficient (about 5 frames per second for a 480×240 video). As a final step, each video frame is transformed back to its original coordinate frame.

10 Summary

In this course, we attempt to summarize the recent advances in visual media retargeting, while classifying the existing body of work into two rough categories: discrete and continuous approaches. While the concepts and the algorithms used in both categories are quite different, clear common grounds and parallels exist: both types of methods try to achieve the best possible retargeting result by optimizing an appropriate energy functional, and they do this by removing (or shrinking) unimportant visual content in order to leave room for well-preserved salient visual information. Both types of approaches have their advantages and disadvantages: roughly speaking, discrete methods generalize cropping and thus handle removal of unnecessary content well, which is especially evident for high-frequency, textured image content (such as foliage, sand, water, etc.). Continuous approaches tend to avoid discontinuity artifacts and typically preserve the overall shapes of image objects more coherently. Interestingly, some continuous methods do not heavily penalize extreme shrinking of unimportant image regions, in which case these regions may shrink to nearly vanishing width, effectively resulting in complete content removal, just like in the discrete methods.

Future research in visual media retargeting will certainly continue to focus on video content, as this area has extremely high impact in terms of applications and still remains a challenge. Additionally, perceptual studies of media retargeting with human subjects should play an important role in further progress. A few recent works attempted some limited experiments of this kind [Rubinstein et al. 2009; Krähenbühl et al. 2009], and there remains a considerable amount of questions to be asked. These include: what do people

like and dislike in transformed images and videos? which artifacts affect this judgement, and which factors contribute to the overall perception of the retargeting results? An important avenue for future research would be to come up with a perceptually significant and validated visual metric, suitable for the evaluation of retargeting methods.

Acknowledgments

We would like to thank all our co-authors for the years of inspiring and fruitful collaborations on the subject of media retargeting. Ariel Shamir's research is funded by the Israel Science Foundation grant 315/07 and the Israel Ministry of Science grant 3-3421. Olga Sorkine's research is supported in part by an NYU URCF grant and an NSF award IIS-0905502.

References

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

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.

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.

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

FAN, X., XIE, X., ZHOU, H.-Q., AND MA, W.-Y. 2003. Looking into video frames on small displays. In *Multimedia '03*, ACM, New York, NY, USA, 247–250.

GAL, R., SORKINE, O., AND COHEN-OR, D. 2006. Feature-aware texturing. In *Eurographics Symposium on Rendering*, 297–303.

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 NIEBUR, E. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell. 20*, 11, 1254–1259.

KRÄHENBÜHL, P., LANG, M., HORNUNG, A., AND GROSS, M. 2009. A system for retargeting of streaming video. *ACM Trans. Graph. 28*, 5.

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.

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)*, vol. 1, 259–265.

RODITTY, L., AND ZWICK, U. 2004. On dynamic shortest paths problems. In *In Proceedings of 12th Annual European Symposium on Algorithms (ESA)*, 580–591.

RUBINSTEIN, M., SHAMIR, A., AND AVIDAN, S. 2008. Improved seam carving for video retargeting. *ACM Trans. Graph. 27*, 3.

RUBINSTEIN, M., SHAMIR, A., AND AVIDAN, S. 2009. Multi-operator media retargeting. *ACM Trans. Graph. 28*, 3.

SANTELLA, A., AGRAWALA, M., DECARLO, D., SALESIN, D., AND COHEN, M. 2006. Gaze-based interaction for semi-automatic photo cropping. In *Proceedings of CHI*, 771–780.

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.

SHAMIR, A., AND AVIDAN, S. 2009. Seam carving for media retargeting. *Communications of the ACM 52*, 1, 77–85.

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

WANG, Y.-S., TAI, C.-L., SORKINE, O., AND LEE, T.-Y. 2008. Optimized scale-and-stretch for image resizing. *ACM Trans. Graph. 27*, 5.

WANG, Y.-S., FU, H., SORKINE, O., LEE, T.-Y., AND SEIDEL, H.-P. 2009. Motion-aware temporal coherence for video resizing. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH ASIA) 28*, 5.

WOLF, L., GUTTMANN, M., AND COHEN-OR, D. 2007. Non-homogeneous content-driven video-retargeting. In *IEEE International Conference on Computer Vision (ICCV)*.

ZABIH, R., MILLER, J., AND MAI, K. 1995. A feature-based algorithm for detecting and classifying scene breaks. In *ACM Multimedia*, 189–200.

ZHANG, Y.-F., HU, S.-M., AND MARTIN, R. R. 2008. Shrinkability maps for content-aware video resizing. *Computer Graphics Forum 27*, 7, 1797–1804.