The Interdisciplinary Center, Herzlia

Efi Arazi School of Computer Science

DISCRETE APPROACHES TO CONTENT-AWARE IMAGE AND VIDEO RETARGETING

M.Sc Dissertation

Submitted by Michael Rubinstein

Under the supervision of Dr. Ariel Shamir

May 21, 2009

Overview

Adapting media to different display devices is an important problem in image analysis. In this process, commonly termed *retargeting*, the input image or video is adapted to accommodate arbitrary displays, varying across resolutions and aspect ratios. The media might be shrunk to create smaller summarizations, or expanded to generate larger syntheses. This problem gained attention in the graphics and vision research communities during recent years due to the increase in variety of commonly used display devices, and the prevalent use of mobile devices as an available mean for media intake. With the explosion of image and video content on the web, we would like to be able to present a feature film on a small iPod, cell-phone, or pocket computer, and show our photographs on projected presentation systems. Indeed, in many of these conversions the media must undergo extensive deformation, often resulting in an unappealing display. Current industrial video scalers are quite primitive in nature, and distorted or "stretched" objects are commonly witnessed in many wide-screen TV sets.

Simple methods for fitting media to display apply standard uniform (per direction) scaling to the image or video. However, this can cause main objects to shrink or expand and gives the scene an unnatural appearance, especially when the media's original aspect-ratio is altered. A commonly used method for retaining the aspect-ratio, known as letterboxing, pads the content by constant-color margins after adapting one of its dimensions to the required size. This might also create distortions in cases of large difference between the source and target sizes, and moreover, does not utilize the entire display space. Simple non-uniform scaling techniques that are used by some TV manufacturers apply an inverse Gaussian scaling kernel over the media, such that its borders are scaled more extensively than the middle, under the assumption that main objects tend to appear in the center of frames. The common property to all these methods is that they are oblivious to the actual media *content*.

An efficient retargeting operator, however, is one that not only considers the geometric constraints of the output display, but also utilizes the structural and semantical information in the input media. Several such *content-aware* operators have been proposed by the academia over the recent years. Their intent is to create a retargeted version of an image or video that captures its most important features on the account of less important ones, such that the required changes in size are gracefully propagated to areas which are less noticeable. This approach hopes to retain, as much as possible, the viewer's experience of the original media. Such operators are in the main focus of this thesis.

Content-aware retargeting is an ill-posed problem, as it relies closely on the definition of content, and within content, on the decision of which is more important than another. Clearly, these objectives might be perceived differently by different viewers. Thus, existing operators all rely on some models describing the importance, or *saliency*, throughout the media. Such methods range from gradient magnitudes and color entropy, through higher-level cues such as motion and object detection, to more sophisticated psychologically-based measures that are based on human perception and eye-gazing studies. Essentially, content-aware operators work in a two-phase approach where first the media is analyzed to understand its important regions using some saliency model, and then apply some operator based on this measure to achieve the retargeted result.

On top of these challenges, the retargeting problem is much more difficult for videos than for images. The reason for this is twofold. First, video presents another dimension, time, to the problem. The retargeting operator cannot simply be applied independently to each video frame, as temporal coherency must be maintained. This is because the human perception was shown to be highly sensitive to motion irregularities. Second, a video sequence is composed of frames, typically 15-30 frames per second ranging from surveillance captures to standard recordings. As such, even relatively short videos already involve massive amounts of data that have to be processed.

Earlier methods for performing content aware retargeting concentrate on adapting the media to smaller displays, and are based on finding an optimal cropping window. This rectangular window is constructed such that the most important information is captured, under the constraints of the target display dimensions. For videos, window trajectories were devised, essentially inserting virtual camera motions like panning and zooming to the result. These methods aim to imitate and automate the *pan-and-scan* process which is still being done manually for adapting wide screen feature films to smaller TV displays. In many cases, however, the media essence cannot be efficiently captured using a rigid fixed size window, as considerable amount of the scene information might be lost. Further, if several important objects appear on the frame peripheries, the result is clearly suboptimal.

More recent methods propose to *deform* the media in a content aware manner. These methods can be divided into two categories: continuous and discrete. In the continuous *warping* formulation, the image is represented using a grid mesh, which is deformed into the required target size by resolving the tension between different requirements of the result. A discrete approach recently proposed for content aware image deformation is the "Seam-Carving" operator, which is based on finding low-energy connected paths of pixels in the image, removing them for image size reduction, and duplicating them for expansion. It was shown to produce very nice results on images.

In this work we have chosen to take the discrete approach to media retargeting. There are certainly advantages to warping methods, as their solution tends to be more global and smooth. However, the energy functionals that are used in the formulations are usually non-quadratic and as such have to be solved iteratively to achieve some local minima. The discrete approach on the other hand, is more local and greedy in nature, but has several advantages: 1. its simplicity and elegance, 2. its speed, and 3. once the problem is formulated in a discrete manner, an *optimal* closed form solution can be obtained. For these reasons we believe that investigating such approaches is interesting and important.

A discrete approach for video retargeting was not yet devised. Therefore, we first work to extend the Seam-Carving operator to video. This extension is not straightforward as the original algorithm relies on dynamic programming, which is not naturally extended to the three dimensions of videos. In order to extend the operator to 3D, we present an exact reduction of the seam carving problem to a minimal cut problem on graphs, that later enables a relatively simple extension to videos. Instead of removing 1D seam paths from an image, we remove 2D *seam manifolds* from a 3D volume representation of a video, generated by stacking its frames. The 3D seams we find are the *optimal* manifolds under the seam constraints. However, we show that this extension alone is not sufficient for producing good quality retargeting due to issues inherent in the operator. Thus, we suggest a novel energy criterion, which we term *Forward Energy* that is integrated into the original operator, can be encoded both in the dynamic programming and graph-cut formulations, and achieves significant improvement in the results. Although we use quite simple saliency measures and temporal models, we are able to show visually appealing results for video size reduction and expansion, as well as other editing operations.

Unfortunately however, each content-aware operator has its strengths and weaknesses,

and will not work in all cases and for all target sizes. Our second observation is that retargeting media using a combination of multiple operators might achieve better results than using any single one alone. We model retargeting operations in a multi-dimensional *resizing space*, and investigate the geometric properties of this space. Under some assumptions, we design an algorithm that is able to combine multiple operators in an *optimal* manner. Here too we take a discrete approach in the sense that the search through different paths of operators is conducted by advancing in the resizing lattice in discrete steps. We also devise a novel measure for assessing the quality of a retargeted result that is independent of the operator used to achieve it. We term our similarity measure *Bidirectional Warping*. We base our models on a user study, showing that users tend to prefer a combination of multiple operators for retargeting, and compare our automatic results with the collected ground truth. We later extend this approach to video and present several interfaces that enable users to efficiently explore tradeoffs between resizing operators, and achieve different retargeting results.

Related Publications

Michael Rubinstein, Ariel Shamir, and Shai Avidan. Improved seam carving for video retargeting. *ACM Trans. Graph.*, 27(3), 2008.

Michael Rubinstein, Ariel Shamir, and Shai Avidan. Multi-operator media retargeting. *ACM Trans. Graph. (to appear)*, 28(3), 2009.

Projects Websites

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

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

Contents

List of Figures

List of Tables

Chapter 1

Video Retargeting using Seam-Carving

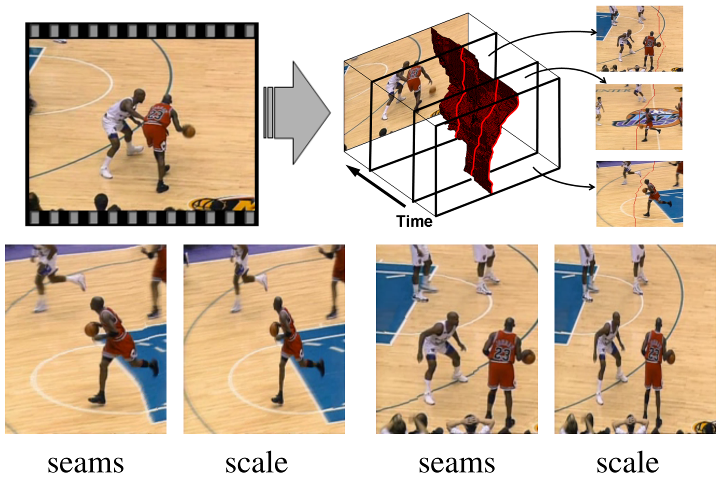


Figure 1.1: Improved seam carving for video sequences combines the frames of the video to form a 3D cube and finds 2D monotonic and connected manifold seams using graph cuts. The intersection of the manifolds with each frame defines the seams on the frame. The manifolds are found using a new forward-energy criterion that reduces both spatial and temporal artifacts considerably.

Abstract

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

1.1 Introduction

Seam carving is an effective technique for content aware image retargeting. In a similar manner, video should support retargeting capabilities as it is displayed on TVs, computers, cellular phones and numerous other devices. A naive extension of seam carving to video is to treat each video frame as an image and resize it independently. This creates jittery artifacts due to the lack of temporal coherency, and a global approach is required. The approach we take is to treat video as a 3D cube and extend seam carving from 1D paths on 2D images, to 2D manifolds in a 3D volume (Figure 1.1). Nevertheless, because we need to build a 2D connected manifold through space-time volume, the dynamic programming approach used for image resizing is no longer applicable. In this paper we define a new formulation of seam carving using graph cuts. However, a simple cut cannot define a valid seam. A seam must be *monotonic*, including one and only one pixel in each row (or column), and *connected*. We show how to define a graph whose cut creates a monotonic and connected seam, which is equivalent to the one created by dynamic programming on images. Using this formulation, we extend seam carving to video and define a monotonic and connected 2D manifold seam inside the video cube. We also discuss a multiresolution approach to speed up the computation time of seams for video.

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

The difficulties imposed by video resizing using seam carving can therefore be characterized as algorithmic, dimensional and cardinal. The algorithmic difficulty follows from the fact that we cannot extend the original dynamic programming method to a 3D video cube. Dimensional difficulties originate from the additional, temporal, dimension of a video, which enhances spatial artifacts and introduces new ones involving motion. Cardinal difficulties stem from the fact that a video is a sequence of frames, and hence any processing of a video sequence involves larger amounts of data. This paper addresses these difficulties and presents results for video resizing applications such as size reduction and expansion, multi-size videos for interactive size manipulation and object removal.

1.2 Background

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

Attention models, based on human spatiotemporal perception have been used to detect Regions Of Interest (ROIs) in image and video. The ROIs are then used to define ”display paths” ([45]) to be used on devices in which the display size is smaller than the video (or image) size. The least important content of the video is cropped, leaving the important features in larger scale, essentially creating a zoom-in-like effect ([9]). Virtual camera motions or pseudo zoom-in/out effects are used to present the content in a visually pleasing manner.

A similar system was proposed by [18], where both cropping and scaling are used together with virtual camera motion to mimic the process of adapting wide screen feature films and DVDs to standard TV resolution. Their system minimizes information loss based on image saliency, object saliency and detected objects (e.g. faces). Cropping, however, discards considerable amounts of information and might be problematic, for instance, if important features are located at distant parts of the image or frame, which is common in wide or over-the-shoulder shots in videos.

An alternative approach is to segment the image into background and foreground layers, scale each one of them independently and then recombine them to produce the retargeted image. This was first proposed by [31] for non-photorealistic retargeting of images and later extended to video by [37]. While this is an appealing approach, it relies crucially on the quality of segmentation - a difficult and complicated task in itself. For video, [22] propose an “object-based” approach to webcam synopsis, where they segment the input video into objects and activities, rather than frames. Then they compose a short video synopsis, in response to user query. Their work only deals with retiming the video, not changing its spatial extent.

Recently, [48] presented a system to retarget video that uses non-uniform global warping. They concentrate on defining an effective saliency map for videos that comprises of spatial edges, face detection and motion detection. Results are shown mainly for reducing video size. Our work differs since we take a discrete approach and we also show results for video expansion, object removal, and introduce multisize videos, which are not supported by their system. We mostly use image edge energies but also show results using their saliency map.

We build on and extend the work of [3]. They proposed seam carving for image retargeting and used dynamic programming to find the optimal seam iteratively. We propose a graph based approach to seam carving, allowing us to handle video retargeting. This extension defines 2D surfaces to be removed from the 3D video cube. An alternative approach is to map these 2D manifolds to frames in a new video sequence [23]. This approach, termed Evolving Time Fronts, gives users the ability to manipulate time in dynamic video scenes.

Note that both [18] and [48] incorporate previously devised methods for identifying shot or scene boundaries, within which the camera motion is continuous. Each shot is then processed by their systems independently. In this work, we assume the input video is comprised of a single shot, and refer to existing scene detection solutions (see [18] and [48], and in addition [49]) in case a multi-shot video is given.

Graph partitioning and graph-based energy minimization techniques are widely used in image and video processing applications such as image restoration, image segmentation, object recognition and shape reconstruction. A graph representing an image, together with some constraints, is partitioned into disjoint subsets by connecting pixels or voxels based on their similarity. Traditionally, similarity is defined by some variation of intensity change or gradients. For videos, it is often convenient to consider the sequence of frames as a 3D space-time volume [15, 30, 44, 43]. In such cases, the extension of energy minimization from 2D images to 3D space-time video is usually straightforward. We are influenced by [15], that use graph cuts to seamlessly patch two 2D or 3D textures. However, there are differences in the way we construct the graph, and the terminal nodes in our method are placed differently than in theirs. The challenge we face is in designing a graph that produces only admissible cuts, that is, cuts that are monotonic so that only one pixel is removed from every row, and are connected. As we will show, standard graph cut based constructions do not satisfy these constraints and new ones must be defined.

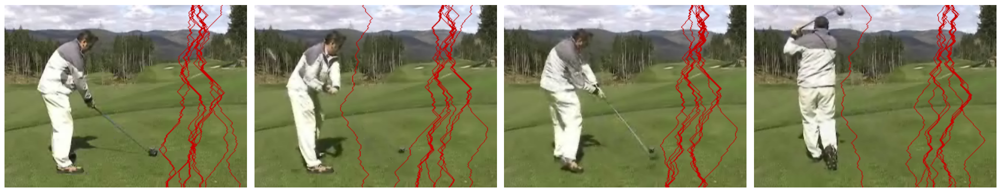


Figure 1.2: Seam carving on each video frame independently creates locally optimal seams that can be totally different over time. This creates a jittery resized video. In this example we show the first ten seams removed. A similar illustration is shown in the supplemental video and project website.

1.3 Preliminaries

A seam is a monotonic and connected path of pixels going from the top of the image to the bottom, or from left to right. By removing one seam from an image, the image size is reduced by one either in the horizontal or the vertical dimension. Seam carving uses an energy function defined on the pixels and successively removes minimum energy paths from the image. In video, we search for a resizing operator in the granularity of shots (i.e. a sequence of frames where the camera shoots continuously). Simply applying the seam carving operator separately to each frame of the video introduces serious artifacts (Figure 1.2).

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

$$\begin{aligned} E_{\text{spatial}}(i,j) &= \max_{t=1}^{N}\{|\frac{\partial}{\partial x}I_t(i,j)| + |\frac{\partial}{\partial y}I_t(i,j)|\} \\ E_{\text{temporal}}(i,j) &= \max_{t=1}^{N}\{|\frac{\partial}{\partial t}I_t(i,j)|\} \\ E_{\text{global}}(i,j) &= \alpha \cdot E_{spatial} + (1-\alpha)E_{temporal} \end{aligned} \quad (1.3.1)$$

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

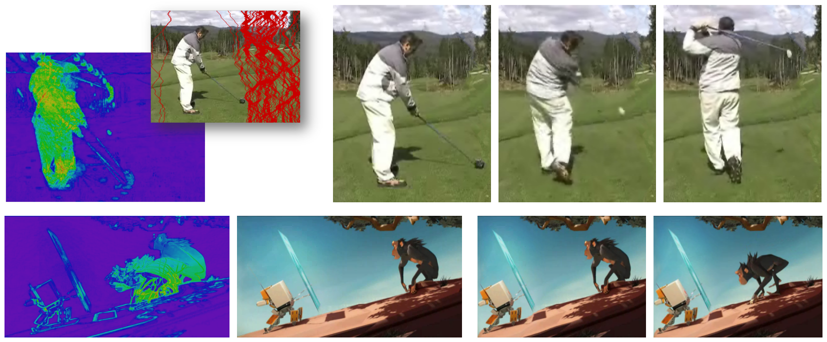


Figure 1.3: Static seams for the golf video and ape animation. The global energy function is shown using color mapping from violet (low) to red (high). The actual static seams are shown for the golf sequence at the top. Some representative resized frames are also shown for both videos (example results can be seen in the project website).

conservative in the cost calculation. Figure 1.3 shows examples for the global energy map and static seams removal from videos.

The main appeal of such a static method is its simplicity and speed. It gives good results when the video is created by a stationary camera, and the foreground and background are separated (Figure 1.3). However, in more complex video scenes where the camera is moving or when multiple motions are present, seams must be allowed to adapt over time.

Towards this end, we define a video seam as a connected 2D manifold “surface” in space-time that cuts through the video 3D cube. The intersection of the surface with each frame defines one seam in this frame. Hence, removing this manifold removes, in effect, one seam from each video frame. On the one hand, because the surface is flexible, the seams can change adaptively over time in each frame (Figure 1.1). On the other hand, because the surface is connected, the seams preserve temporal coherency. Unfortunately, there is no simple extension of the dynamic programming algorithm of 2D images to a 3D space-time volume, and we must employ another algorithm, namely graph cut.

1.4 Seam Carving using Graph Cuts

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

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

Converting dynamic programming to graph cuts was already done in the past for the purpose of texture synthesis [15]. However, there is a crucial difference between our work and theirs. The reason is that a general cut does not define a valid seam for seam-carving, as it must satisfy two constraints:

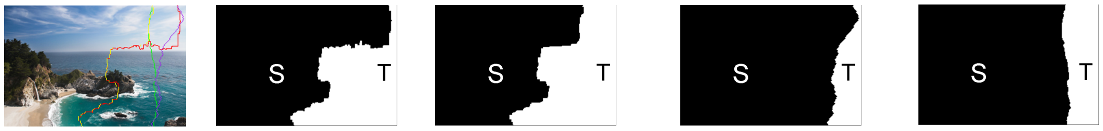
Monotonicity the seam must include one and only one pixel in each row (or column for horizontal seams).

Connectivity the pixels of the seams must be connected.

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

1.4.1 Graph Cuts for Images

In a standard grid graph construction, every internal node $p_{i,j}$ is connected to its four neighbors $Nbr(p_{i,j}) = \{p_{i-1,j}, p_{i+1,j}, p_{i,j-1}, p_{i,j+1}\}$. Following the L_1-norm gradient magnitude E_1 energy that was used in [3], we define the weight of arcs as the forward difference between the corresponding pixels in the image either in the horizontal direction: $\partial x(i,j) = |I(i,j+1) - I(i,j)|$ or in the vertical: $\partial y(i,j) = |I(i+1,j) - I(i,j)|$. Under this formu-



(a) Non-monotonic (b) Unconnected (c) Original (backward) (d) Forward

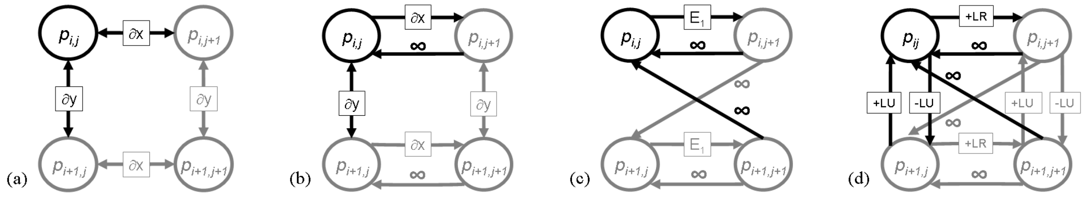


Figure 1.4: Minimum cut on the waterfall image (top left) for various graph constructions. The seam is composed of the pixels to the left of the cut. The different graph constructions are illustrated by four nodes representing four pixels in the image. The actual image graph is created by tiling these sub-graphs across the image (see text for details). Graph (a) creates a general path and not a valid seam, while (b) creates a monotonic but piecewise-connected seam. The construction at (c) is equivalent to the original seam carving algorithm (with E_1). The construction at (d) represents the new forward energy we present in Section 1.5.

lation, Figure 1.4(a) shows an optimal partition of the waterfall image into source and target parts. This cut does not satisfy the seam carving constraints.

To impose the monotonicity constraint on a cut, we use different weights for the different directions of the horizontal arcs. For forward arcs (in the direction from S to T), we use the weight as defined above, but for backward arcs we use infinite weight. Appendix 1.A gives the proof why the monotonicity constraint is maintained under this construction (Figure 1.4(b)).

The main difference between this graph cut construction and the original dynamic programming approach is that there is no explicit constraint on the cut to create a *connected path*. The cut can pass through several consecutive vertical arcs, in effect creating a piecewise-connected seam. Although this behavior is penalized as more vertical arcs are cut, it does happen in practice. Our empirical results show that connected seams are important to preserve both spatial and temporal continuity and to minimize visual artifacts. To constrain cuts to be connected, we use infinite weight diagonal arcs going "backwards". Using similar arguments, Appendix 1.A shows why this construction imposes the connectivity constraint.

In fact, by combining the weights of the vertical and horizontal arcs together, we can create a graph whose cut will define a seam that is *equivalent* to the one found by the original dynamic programming algorithm. For example, we assign the weight $E_1(i,j) = \partial x(i,j) +$

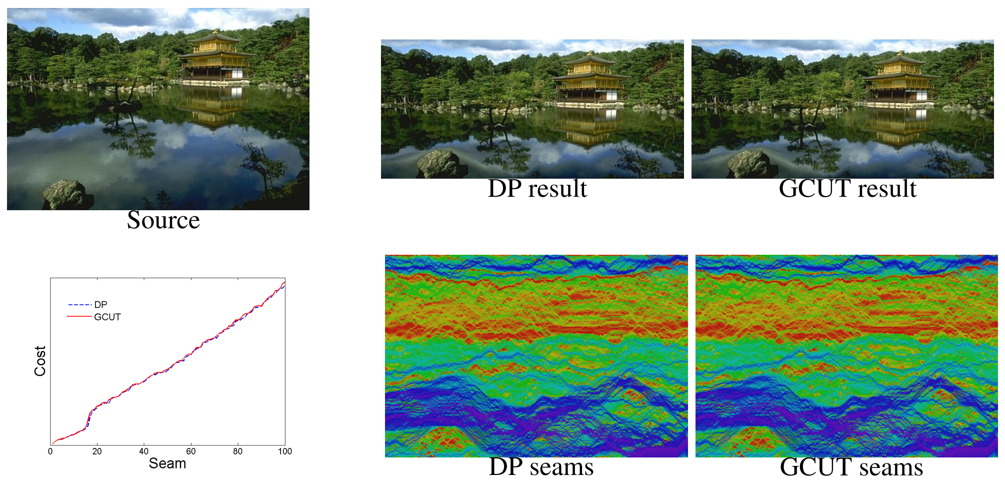


Figure 1.5: Comparison between Dynamic programming (DP) and graph-cut (GCUT) results. The results are equivalent theoretically, and nearly identical practically. On the bottom left, the plot shows the seams costs using DP (dashed blue) and GCUT (red) for the first 100 seams that were applied to generate the results. On the bottom right, the complete seam maps are shown for the two methods, colored by the seam costs from low (blue) to high (red).

$\partial y(i,j)$ to the horizontal forward arc and remove the vertical arcs altogether (Figure 1.4(c)). A cut in this graph is monotonic and connected. It consists of only horizontal forward arcs (the rest are infinite weight arcs that pose the constraints and cannot be cut), hence its cost is the sum of $E_1(i,j)$ for all seam pixels, which is exactly the cost of the seam in the original seam carving operator. Because both algorithms guarantee optimality, they must have the same cost, and (assuming all seams have different costs) the seams must be the same.

This suggests we can use any energy function defined on the pixels as the weight of the forward horizontal arcs and achieve the same results as the original dynamic programming based seam carving. Moreover, high level functions such as a face detector [41], 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.

We use the implementation by [5] to solve for the minimal cut. In practice, when comparing between the dynamic programming and graph-cut methods, we noticed that small differences exist in the results. we have traced these differences both to numerical precisions, but mostly to the non-uniqueness of the solution. Due to the natural smoothness of images, there are many cases in which several seams of minimal cost exist. Since the decision taken by the two solvers in such cases is not synchronized, different seams of minimal cost might be removed. Due to the greedy nature of the seam-carving algorithm, this might also affect seams at later iterations. However, the effect of these anomalies on the resulting seams, the seams costs and the final results are minimal (Figure 1.5).

The running time of the graph cut algorithm is $O(mn^2)$ for m arcs and n vertices, but in practice was observed to have linear running time on average [5]. Given an N-pixel image, our construction requires $m \simeq 4N$ arcs (two horizontal arcs and two backward diagonal arcs per vertex). It is not hard to further prove that (using 4 or less pixel neighborhood) the diagonal arcs are sufficient to impose both constraints, resulting in a small factor improvement in running time ($m \simeq 3N$). However, we continue to use them in an additive manner for clarity. Note moreover that piecewise connectivity can be obtained to some extent in this setting by spreading backward diagonal arcs "blocks" (Figure 1.19(b-c)) in larger intervals in the graph.

1.4.2 Graph Cuts for Video

The extension to video is straightforward. Assuming we are searching for a vertical seam, we consider the $X \times T$ planes in the video cube and use the same graph construction as in $X \times Y$ including backward diagonal infinity arcs for connectivity. We connect the source and sink nodes to all left and right (top/bottom in the horizontal case) columns of all frames respectively. A partitioning of the 3D video volume to source and sink using graph cut will define a manifold inside the 3D domain (Figure 1.6). 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.

For the full video volume, the computation time of our algorithm depends on the number of nodes times the number of arcs in the graph, which is quadratic in the number of voxels.

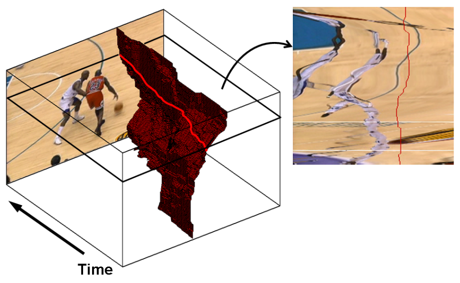
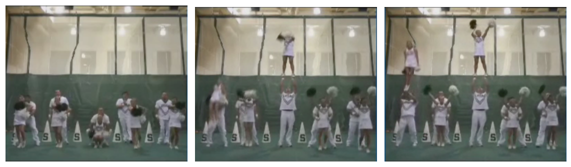


Figure 1.6: The intersection of every $X \times T$ plane with the seam surface defines a spatiotemporal seam.



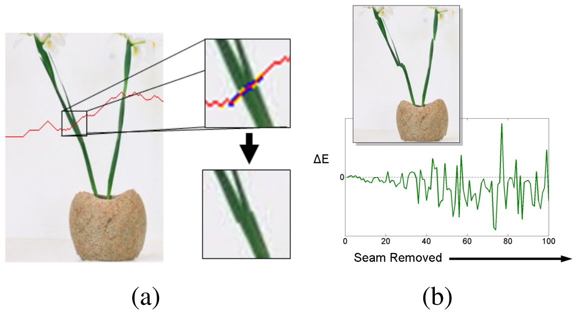


Figure 1.7: The artifacts seen in video retargeting (top) can also be seen on a static vase image (bottom). We show an example of the change in energy after a specific seam is removed (a). In some pixels (blue) energy is reduced and in others (yellow) increased. This seam inserts more energy to the image than removes, creating a step artifact in the stem of the flower. The actual change in energy ΔE after each seam removal is shown in (b).

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 [20]. 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.

The graph cut approach to seam carving allows us to extend the benefits of content-aware resizing to video. Still, the method is not perfect and no single energy function was shown to perform properly in all cases [3]. Therefore, we introduce a new energy function that better protects media content, and improves image and video results.

1.5 Forward Energy

The artifacts created in video frames can actually be seen on static images as well (Figure 1.7). They are created because the original algorithm chooses to remove the seam with

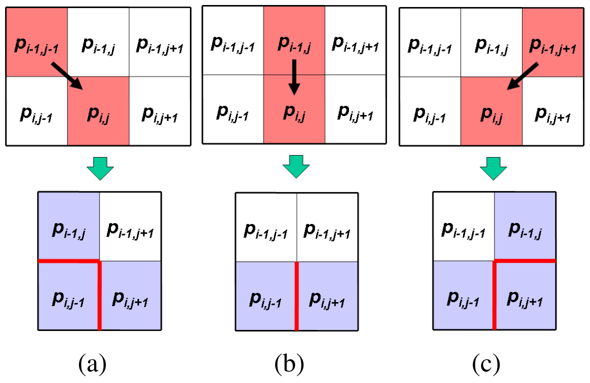


Figure 1.8: Calculating the three possible vertical seam step costs for pixel $p_{i,j}$ using forward energy. After removing the seam, new neighbors (in gray) and new pixel edges (in red) are created. In each case the cost is defined by the forward difference in the newly created pixel edges. Note that the new edges created in row $i-1$ were accounted for in the cost of the previous row pixel.

the least amount of energy from the image, ignoring energy that is *inserted* into the retargeted result. The inserted energy is due to new edges created by previously non adjacent pixels that become neighbors once the seam is removed (see e.g. the steps artifacts in Figure 1.7(a)). Assume we resize an image $I = I_{t=1}$ using k seam removals ($t = 1 \dots k$). To measure the real change in energy after a removal of a seam, we measure the difference in the energy of the image after the removal ($I_{t=i+1}$) and the energy of only those parts that were not removed in the previous image $I_{t=i}$ (i.e. the image energy $E(I_{t=i})$ minus the seam energy). In our new graph cut formulation, the energy of the image is no longer an attribute of the pixels, but rather an attribute of the arcs in the graph. Hence, the energy of an image $E(I)$ is given by the sum of all finite arcs of its induced graph, and the energy of a seam $E(C)$ is simply the cost of the cut C. The energy difference after the i^{th} seam carving operation is:

$$\Delta E_{t=i+1} = E(I_{t=i+1}) - [E(I_{t=i}) - E(C_i)] \tag{1.5.1}$$

As can be seen in Figure 1.7(b), ΔE_t can actually increase as well as decrease for different seam removals using the original seam carving approach (the energy measured in this case is E_1). The figure also shows a specific example of a seam that inserts more energy to the image than it removes.

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

As the removal of a connected seam affects the image, and its energy, only at a local neighborhood, it suffices to examine a small local region near the removed pixel. We consider the energy introduced by removing a certain pixel to be the new "pixel-edges" created in the image. The cost of these pixel edges is measured as the forward differences between the pixels that become new neighbors, after the seam is removed. Depending on the direction of the seam, three such cases are possible (see Figure 1.8).

1.5.1 Forward Energy in Dynamic Programming

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

$$\begin{aligned} &(a)\; C_L(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j-1)| \\ &(b)\; C_U(i,j) = |I(i,j+1) - I(i,j-1)| \\ &(c)\; C_R(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j+1)| \end{aligned} \tag{1.5.2}$$

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

$$M(i,j) = P(i,j) + \min \begin{cases} M(i-1,j-1) + C_L(i,j) \\ M(i-1,j) + C_U(i,j) \\ M(i-1,j+1) + C_R(i,j) \end{cases} \tag{1.5.3}$$

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

The first row of M is initialized using the vertical seam case (Figure 1.8(b)), taking previous weights to be zero. A nuance to note is the cost to remove the upper leftmost and rightmost pixels. Theoretically, the cost for removing these pixels is zero according to the above scheme, as no energy is introduced by removing them (no new edges are created). However, following this logic, the zero cost will be diffused throughout the leftmost and rightmost columns which will in turn cause the algorithm to always choose to remove a boundary column, essentially creating a cropping effect. To counter this, we set C_U for border pixels to the forward difference with their respective neighbor.

1.5.2 Forward Energy in Graph Cut

Note that in order to define the forward energy cost in graph cut, we cannot use the same construction as for the backward energy (Section 1.4.1), since the energy is not defined per pixel. Instead, we need to create a graph whose arc weights will reflect the cost of the pixel removal according to the three possible seam directions, such that the cost of inserted edges will be properly integrated by the cut. Figure 1.4(d) illustrates this construction. A new horizontal pixel-edge $p_{i,j-1}p_{i,j+1}$ is created in all three cases because $p_{i,j}$ is removed. Hence, we assign the difference between the **L**eft and **R**ight neighbors $+LR = |I(i, j+1) - I(i, j-1)|$ to the graph arc between the nodes representing $p_{i,j}$ and $p_{i,j+1}$. To maintain the seam monotonicity constraint as before, we connect $p_{i,j+1}$ and $p_{i,j}$ with a (backward) infinite weight arc. We also add diagonal backward infinite arcs to preserve connectivity.

Next, we need to account for the energy inserted by the new vertical pixel-edges. In the case of a vertical seam step (Figure 1.8(b)), there are no new vertical edges so no energy is inserted. From the corollary in appendix 1.A we have that all nodes to the left of the cut must be labeled S and all nodes to the right of the cut must be labeled T. By definition, the cost of a cut will only consider arcs directed from nodes labeled S to nodes labeled T. It therefore follows that only upward vertical arcs will be counted in right-oriented cuts (Figure 1.8(a)), and only downward vertical arcs will be counted in left-oriented cuts (Figure 1.8(c)). Hence, we assign the difference between the **L**eft and **U**p neighbors $+LU = |I(i-1, j) - I(i, j-1)|$ to the upward vertical arc between $p_{i,j}$ and $p_{i-1,j}$, and the weight $-LU = |I(i+1, j) - I(i, j-1)|$ to the downward vertical arc between $p_{i,j}$ and $p_{i+1,j}$ ($-LU$ means the difference between the **L**eft and **U**p neighbors with respect to the end point of the arrow).

Figure 1.9 illustrates the difference between removing seams using the original algorithm with E_1, and removing seams using the new forward energy we propose. In the original cost map the cost is increased with every crossing of a bar in the bench, as it defines an edge in the image. This drives the seams to the image sides while creating disturbing artifacts. In the improved criterion, vertical seams can intersect the bars without inserting energy to the im-

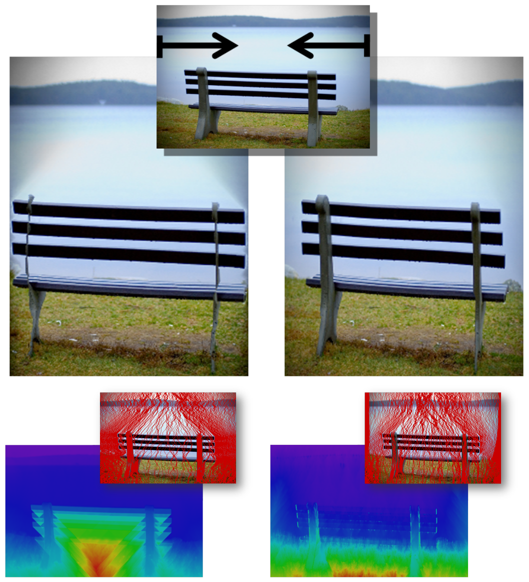


Figure 1.9: Comparison between the original seam carving backward energy (left) and the new forward energy (right) for resizing an image (original shown in small at the top). At the bottom are the respective cost maps M of both techniques and the seams removed from the image. The new results suffer much less from the artifacts generated using backward energy such as the difference in water color and the distortions of the bench bars and skeleton.

age, resulting in almost no increase in the cost map in these areas and a more plausible result. More examples are given in Figure 1.11 and in the supplemental material. Figures 1.12 and 1.13 show some frames from video sequences retargeted with graph cuts using the improved forward energy.

For video, we examine slices in the 3D video-cube depending on the seam direction. For vertical seams (Y-direction), the intersection of every slice on the $(X \times T)$ dimension with the seam manifold creates a seam on that plane (Figure 1.6). We use the same formulation in $(X \times T)$ as we did in $(X \times Y)$. Hence, we define the cost of every pixel removal as the new *temporal* pixel-edges created between frames in the temporal direction, that are introduced to the video when this pixel is removed. We then create arcs between nodes in the graph between time-steps with the appropriate costs exactly as in the spatial $X \times Y$ domain.

1.5.3 Minimizing Energy Change

Taking the forward energy approach one step further, we might wish to minimize the *change* in energy as opposed to just the inserted one. That is, we can search for seams such that their removal will not only discourage the insertion of new edges to the image, but also prevent the removal of existing edges from it. Continuing to use the E_1 energy, this can be formulated as follows. We define the change in energy as the change in edges (gradients) between pixels as a result of removing a certain pixel, and we consider again the different cases of inserted edges (Figure 1.8). Note that in either case, and for both the horizontal and vertical directions, we have to calculate the change in energy between two edges that exist before the relevant pixel is removed, and a single edge that remains between the two new neighbors after its removal. That is, we can write the change in energy as a function of the new edge, e_{new}, and the two old edges, e^1_{old} and e^2_{old}. One possibility is:

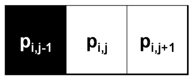
$$\Delta e(e_{new}, e^1_{old}, e^2_{old}) = e_{new} - (e^1_{old} + e^2_{old}) \tag{1.5.4}$$

For convenience, denote $e^{i,j}_{k,l} = |I(i,j) - I(k,l)|$. Note that this cost is always non-negative, while Δe can be either positive or negative (or zero), as energy can be inserted or removed from the image. For example, if the new edge is stronger than the previous edges then we have $\Delta e > 0$ by Equation 1.5.4, which indicates that energy was inserted to the image.

Given a delta function Δe, we can then rewrite Equations 1.5.2 in terms of the change in energy:

$$\begin{aligned}
&(a)\ C_L(i,j) = |\Delta e(e^{i,j-1}_{i,j+1}, e^{i,j-1}_{i,j}, e^{i,j}_{i,j+1})| + |\Delta e(e^{i,j-1}_{i-1,j}, e^{i,j-1}_{i-1,j-1}, e^{i,j}_{i-1,j})| \\
&(b)\ C_U(i,j) = |\Delta e(e^{i,j-1}_{i,j+1}, e^{i,j-1}_{i,j}, e^{i,j}_{i,j+1})| \\
&(c)\ C_R(i,j) = |\Delta e(e^{i,j-1}_{i,j+1}, e^{i,j-1}_{i,j}, e^{i,j}_{i,j+1})| + |\Delta e(e^{i,j+1}_{i-1,j}, e^{i,j+1}_{i-1,j+1}, e^{i,j}_{i-1,j})|
\end{aligned} \tag{1.5.5}$$

and use these equations in the calculation of the energy map M as before (Equation 1.5.3). Our treatment of boundary pixels is also similar, but here it also correctly models the change in energy, as removing a boundary pixel removes from the image the edge between that pixel and its neighbor. Moreover, consider the case of an edge occurring between $p_{i,j-1}$ and $p_{i,j}$ at the point of examining pixel $p_{i,j}$. Using the formulation from Section 1.5.1, the horizontal cost of removing pixel $p_{i,j}$ would be high, due to the edge created between $p_{i,j-1}$ and $p_{i,j+1}$. However, this edge is clearly not the result of removing that pixel. Examining the change in energy instead, will result in zero change in this case. In fact, this formulation generalizes the previous, as the forward minimally inserted energy can be achieved by taking $\Delta e(e_{new}, e^1_{old}, e^2_{old}) = e_{new}$.



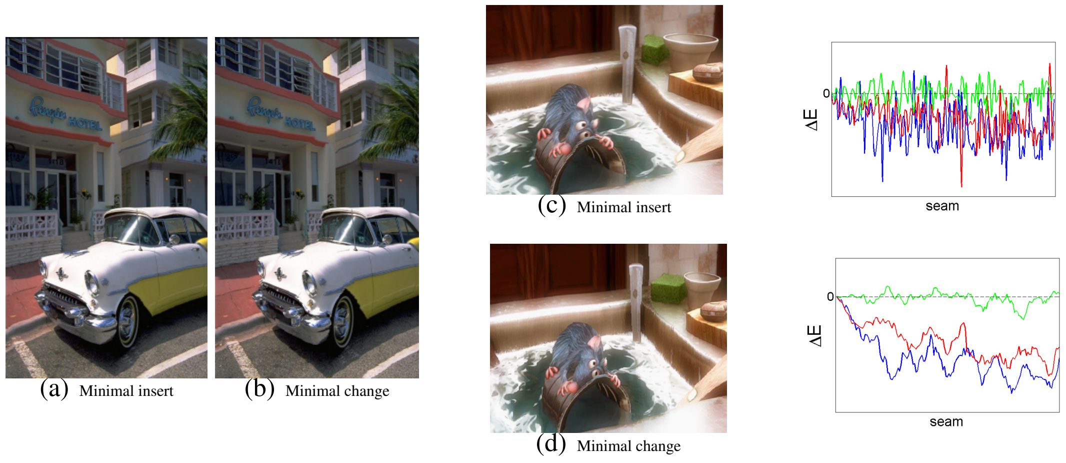


Figure 1.10: Comparison between minimal inserted energy and minimal change in energy on the car and Ratatouille figures. It can be seen that minimizing against the change in energy attempts to maintain information (e.g. the white fence in the car image) and protect edges (e.g. the sink rim in the Ratatouille image), but on the account of occasionally inserting artifacts. On the right, the top graph shows the change in energy (Equation 1.5.1) in the car image throughout the seam removal process for minimal energy seams (original algorithm, blue), MIE seams (red) and MCE seams (green). At the bottom, a trend graph is shown using 5-point moving average.

Our results were inconclusive. There are cases in which minimizing inserted energy (MIE) produces better results, and there are other cases in which minimizing change in energy (MCE) produces more satisfactory solution. Figure 1.10 shows some comparisons between the two methods. By tracking the change in energy (Equation 1.5.1) during seam calculation, we note that indeed when using MCE, the change in energy curve oscillates around zero. Thus, less energy is removed from the image in comparison to MIE, though at the cost of inserting energy to the result. The results generated by MIE usually appear smoother.

We have experimented with several permutations of delta functions and ways to incorporate them into the energy map. However, due to the discrete and local nature of the operator, all formulations are very similar, and achieve more or less the same results.

1.6 Results

In the supplemental video and project website, we present results for aspect ratio changes of videos by removing, as well as inserting seams (see also Figure 1.12 and Figure 1.13). We also support multisize videos for interactive resizing (Figure 1.15, top, and the supplemental

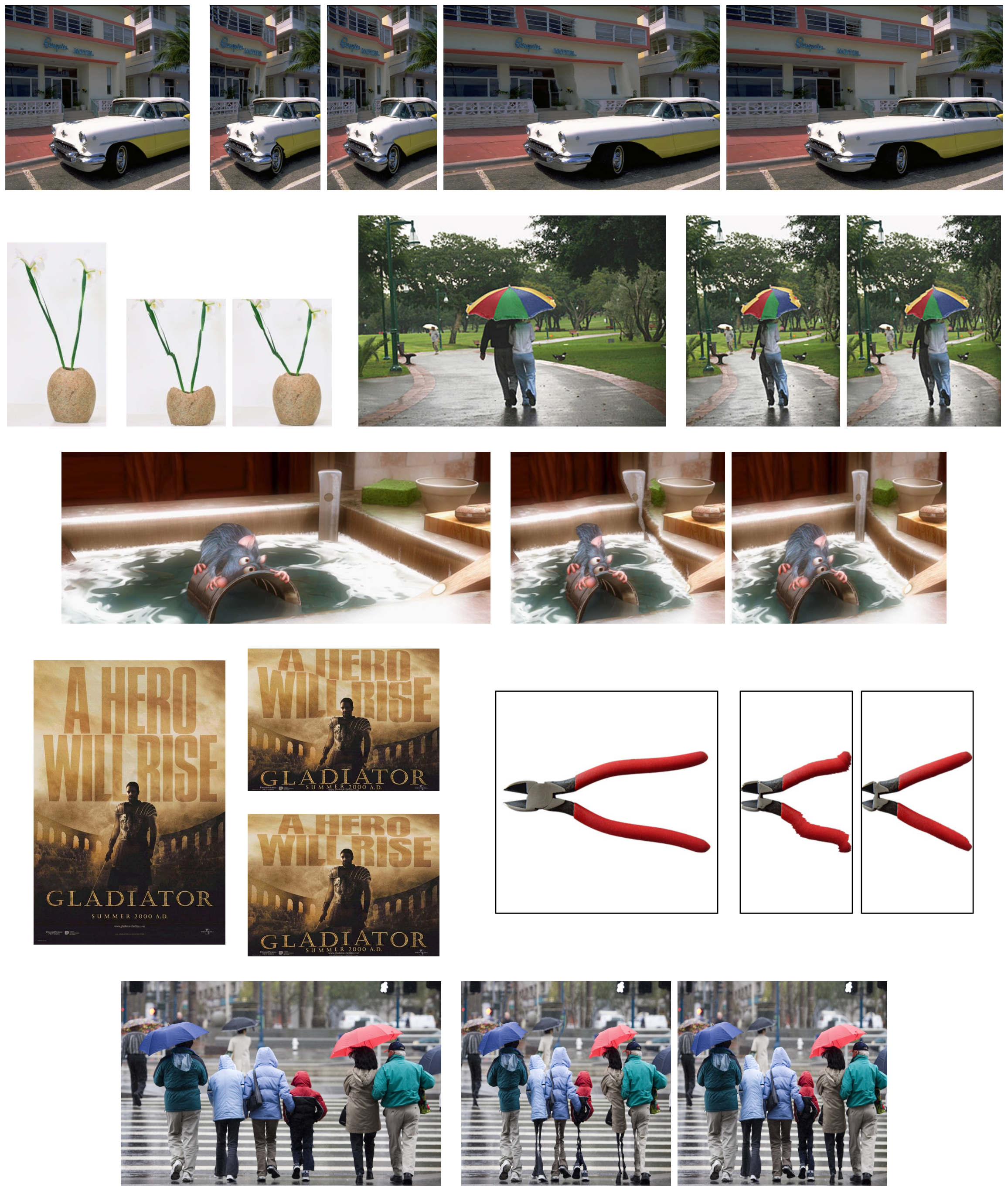


Figure 1.11: Several comparisons between the original seam carving algorithm (left/top image of pairs) and forward energy (right/bottom image of pairs). The reader is encouraged to zoom-in for better view. At the top, the car image (first on the left) was first condensed and then extended. In the middle, one frame from the Ratatouille video is given for comparison. The complete sequence in shown in the supplemental video. It can be seen that our forward energy approach yields more visually-plausible results than backward energy, while artifacts generated by seam removal are greatly reduced. Further results can be found on the project website.

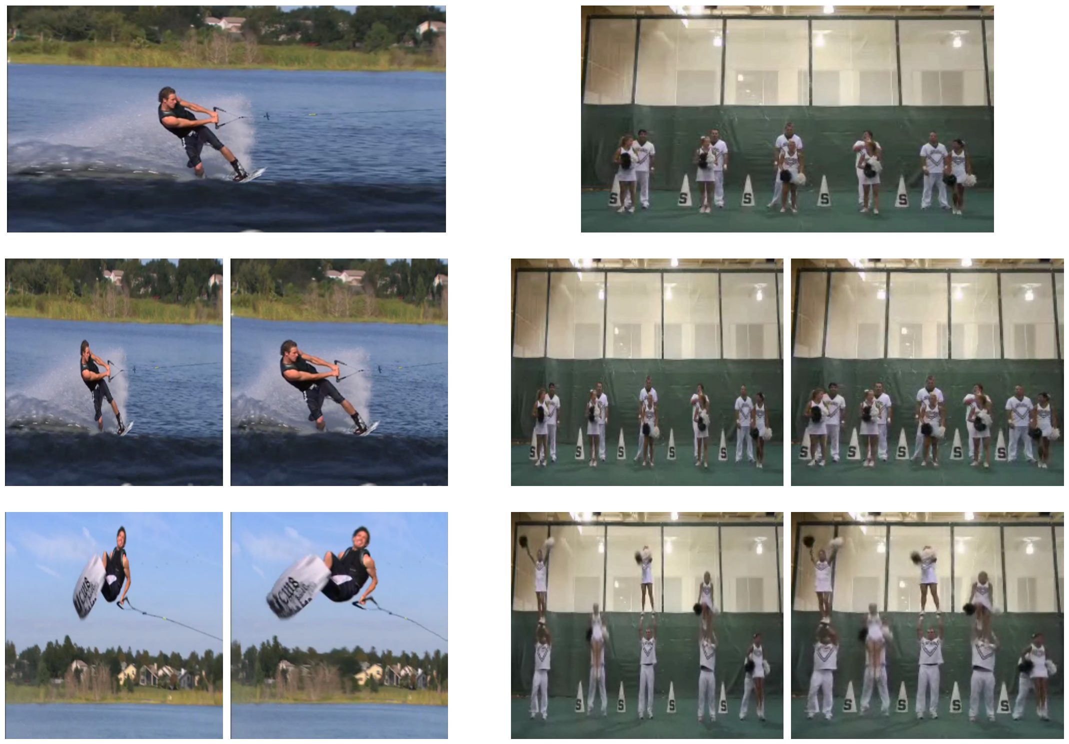


Figure 1.12: Examples of video retargeting. Top row, an original frame. In the following rows we show a rescaled frame on the left and a retargeted one on the right.

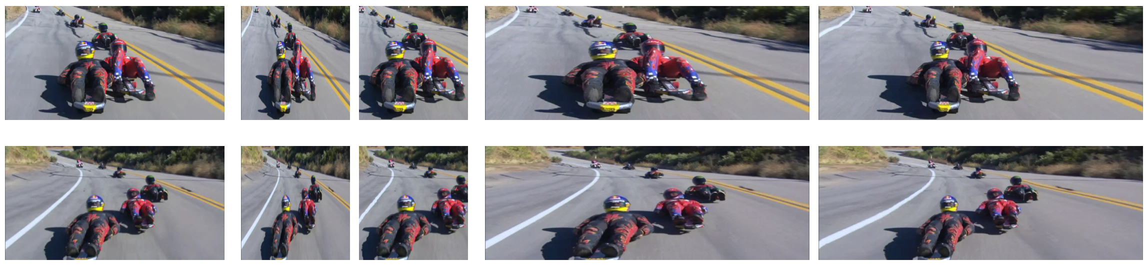


Figure 1.13: Each row shows a different frame from a 100 frames long video sequence. From left to right, the original image, a scaled down image, a targeted down image, a scaled up image and targeted up image.

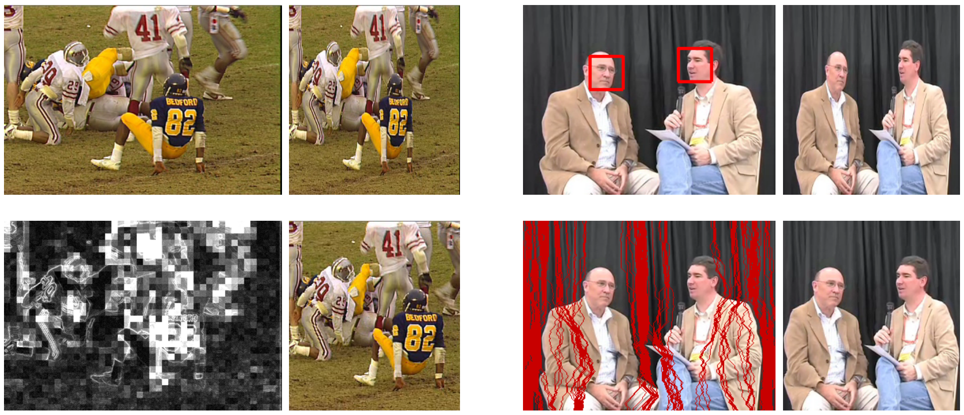


Figure 1.14: Retargeting using external energy (saliency) functions. On the left, from left to right and top to bottom: a frame from the football video; saliency map used by [48]; the rescaled frame; the retargeted frame. On the right, in the same order: a frame from the interview video, with the detected face regions; the calculated seams for this frame, accounting for the faces constraints; the rescaled frame; the retargeted result.

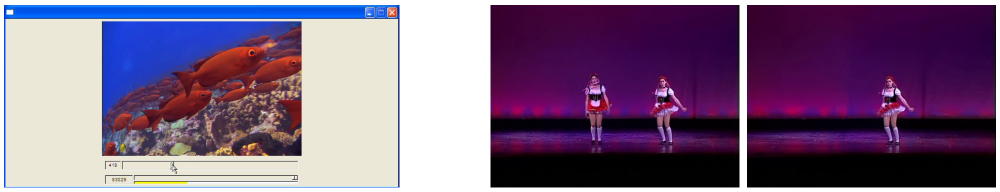


Figure 1.15: A snapshot of the multisize video interface is shown on the left. After pre-computation, the user is able to resize the video interactively while it plays. On the right, a frame from the dancers video is shown (left), with its corresponding frame from the video in which the left dancer was removed using user markings. Actual results can be viewed in the supplemental video.

video). We extend the method suggested by [3] of precomputing seam index maps for images, to each frame in the video. As we cannot hold the entire index structure in memory, these maps are stored on disk, and are loaded on demand before the frame is displayed.

As discussed, we also support other energy functions for retargeting. For example, Figure 1.14 shows the results of our method on the football video using the saliency map of [48]. Our system also supports other energy functions such as object detectors and manually inserted weights. As our approach is global, the algorithm is relatively robust to cases in which the energy function is not given for every frame, and to occasional false positive or false negative detections. An example using face detector is shown in Figure 1.14. Both sequences are also shown in the supplemental video.

By marking pixels with positive weights, the user can protect certain parts of a video during the retargeting process. For similar reasoning as above, the user need not mark every frame, but only once every k frames (in practice we used $k \simeq 10$). Since we achieve a temporally smooth solution, the markings' coherency is maintained. By supplying negative weights, the user can also attract seams to desired parts of the video, for example, for object removal (Figure 1.15).

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

Our technique does allow however to achieve speed gains for processing high definition images. For instance, the three mega-pixel lake image (Figure 1.16) was resized to 60% its width (using forward energy) in approximately 12 minutes using dynamic programming, and less than 5 minutes using multiresolution graph cut on our computer configuration. As discussed in Section 1.4.1, the graph cut result is equivalent to the dynamic programming result due to our reduction. Although the multiresolution solution only approximates the optimal cut, it can be seen that the result is visually in par with the optimal.

1.7 Limitations

The forward energy criteria we propose is designed to protect the structure of media. However, maintaining the structure can sometime come at the expense of content. For example, important objects that can be resized without noticeable artifacts (i.e. inserted energy) may be jeopardized during resizing (Figure 1.17). In such cases, a combination of the forward criteria with E_1 energy can help to achieve better results. This is because E_1 can better protect content. There are other situations on video and images where forward energy fails to achieve plausible results. Some are illustrated in Figure 1.18. In general, due to motion and camera movement, the problem of video resizing is more challenging than image resizing. To solve some of those challenges, it may be better to revert to other methods of resizing such as

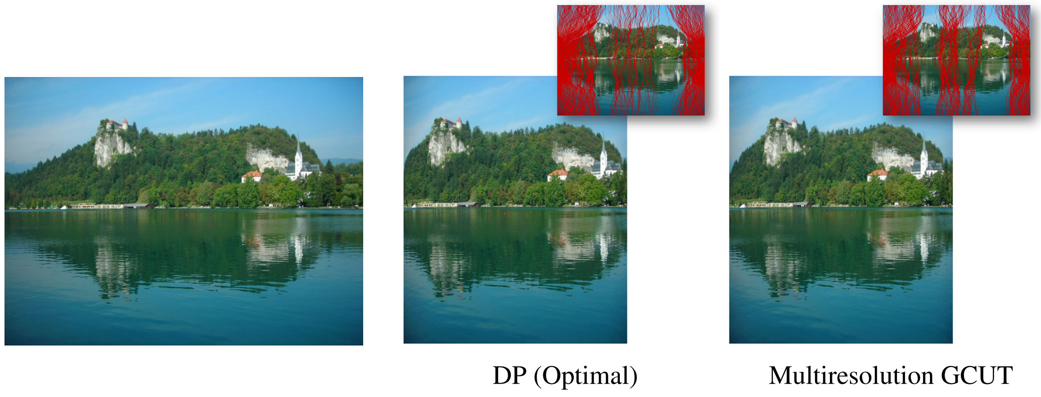


Figure 1.16: Example result of the multiresolution graph-cut on a 3-gigapixel image (left). The multiresolution result (right), though only an approximation to the optimal result (middle), is equally appealing and much more efficient to compute.

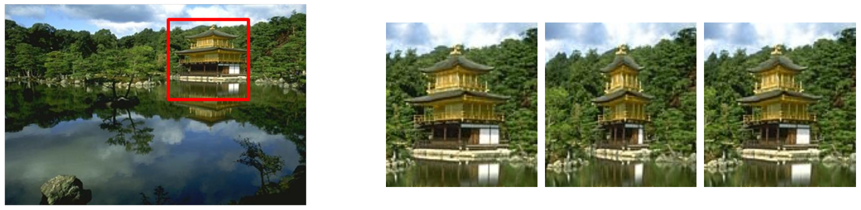


Figure 1.17: Structure vs. content. From left to right: a zoom-in on the Kinkaku temple (Kyoto, Japan) using backward energy, forward energy and combination of the forward and E_1 energies respectively. There were no significant differences between the results in other image regions.

scaling or cropping or combine them together with seam carving. Lastly, our current method runs on the video in batch mode. In contrast, online techniques could also support resizing while streaming the video.

1.8 Conclusions and Future Work

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

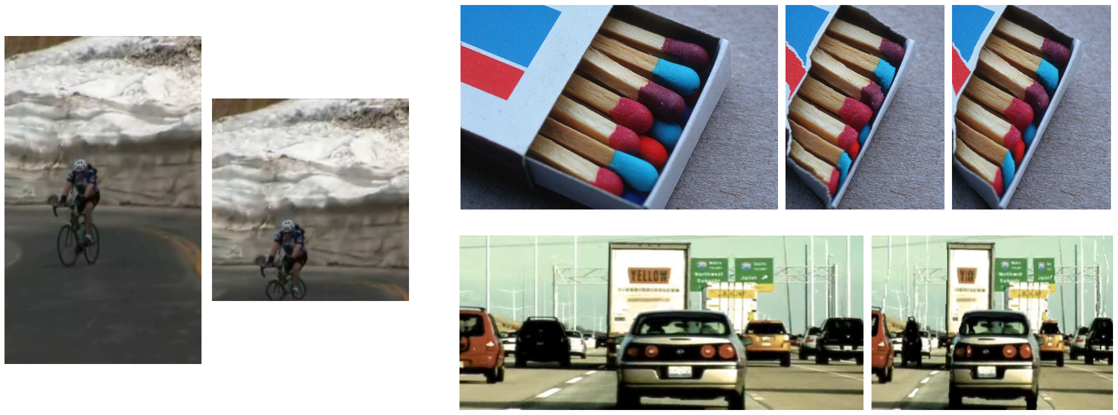


Figure 1.18: Cases where forward energy can fail. On the left, a frame from a bicycle video sequence (left) is shown with its retargeted result. The bicycles are shrunk as the algorithm abstains from cutting the textured rocks. On the upper right, although the forward energy result (right) shows some improvement over the backward energy result (middle), it still suffers from side-effects due to its local nature. A grainy background texture is considered as important content, while the matchbox is distorted by both methods. On the bottom row, a frame from the highway video (left) is shown with its corresponding frame from the retargeted video. Forward energy fails to achieve plausible result in this case due to rapid camera and object motions.

the effectiveness of our contributions on several images and video sequences.

We have outlined some future extensions in the Limitation section. Also, by switching to graph cut based representation we could rely on some advances to speed up computations. For example, [14] proposed a method for computing minimum cuts on an updated graph, which can hopefully yield speed gains of up to two orders of magnitude. Moreover, recent methods allow for efficient graph cut calculation on the GPU ([12, 40]), with similar reported running time improvements over CPU implementations. Using such methods would significantly decrease the computational time of the multiresolution algorithm, and might also enable calculation at the pixel level, thus resulting in an optimal solution.

Our methods can also be adapted to resize videos *temporally*. By rotating the video cube to $Y \times T$ view, we can find seam manifolds that cut through the temporal domain. Each manifold, when removed, will decrease the length of the video by one, thus resulting in a shorter video. A similar method was recently proposed also by [7]. They too use graph cuts for finding low gradient sheets to remove. A basic difference between their method and ours is that they remove an approximation to the minimal energy surface, while our method guarantees optimality under the seam constraints (Section 1.4.2). Their graph construction is similar to the one described in Figure 1.4(a), which yields non-monotonic and unconnected cuts. Moreover, they counter the cardinality problem by splitting the input video into smaller pieces and removing one frame at a time from each piece. By using a multiresolution scheme, we target a more global solution. Enforcing seam temporal connectivity using our approach might also help avoiding cases of event reordering that they refer to in their work, as seams will not be able to stretch too far in time.

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

Appendix 1.A Seam Constraints Proof

We show that the graph construction introduced in section 1.4 using horizontal backward infinite arcs induces a minimal cut which necessarily maintains monotonicity.

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

The optimal cut passes each row only once: W.l.g. assume that there exists a row j in the grid in which the cut passes twice (in fact it must then cut the row an odd number of times). Let us examine two consecutive cuts in row j. Let node $p_{i,j}$ be labeled S, the nodes $p_{i+1,j}$ to $p_{k-1,j}$ will be labeled T and the nodes $p_{k,j}$ will be labeled S again. However, this also means that the arc $p_{k,j} \rightarrow p_{k-1,j}$, which is an infinite weight arc, must be included in the cut (figure 1.19(a)). This makes it an infinite cost cut, which contradicts optimality since it is always possible to cut only horizontal arcs at some column of the grid and achieve a finite cost cut.

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

If we want the cut to be connected as well (as shown in Figure 1.4(c-d)), we use backward-going diagonal arcs. The same argument as above can prove connectivity as illustrated in Figure 1.19(b-c).

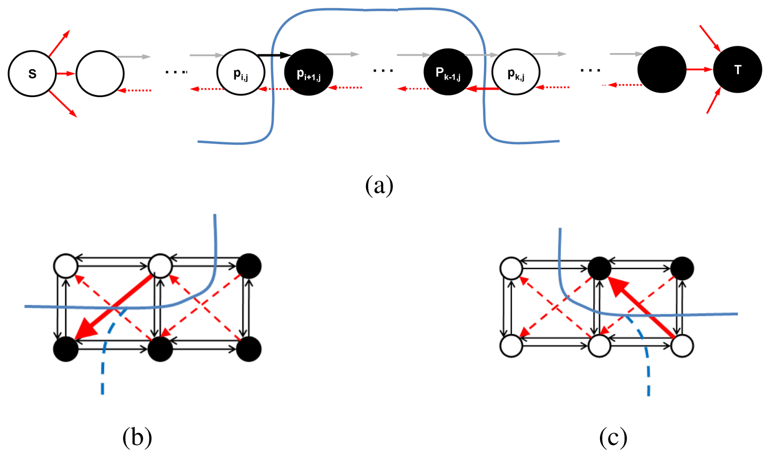


Figure 1.19: Using infinity weight edges (red) in the graph construction maintains the seam constraints. Horizontal infinity arcs maintain monotonicity (a) - see details in text. Diagonal infinity arcs maintain connectivity. If the cut skips more than one pixel to the left (b) or right (c) - a diagonal infinity arc from a source node (white) to a target node (black) must be cut.

Chapter 2

Multi-operator Media Retargeting

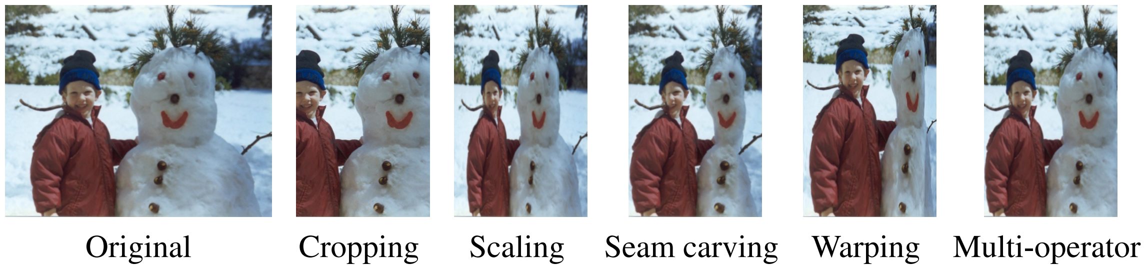


Figure 2.1: The multi operator algorithm uses dynamic programming to find the optimal combination of retargeting operators. Here we show a comparison of several methods. The original image (left) is retargeted using: simple cropping, uniform scaling, seam carving [Rubinstein *et al.* 2008], non-uniform warping [Wang *et al.* 2008] and our multi-operator algorithm. In this example, the multi-operator algorithm combines cropping, scaling and seam carving to optimize our new image-to-image similarity measure, termed Bidirectional Warping (BDW). The algorithm can use other retargeting operators and similarity measures.

Abstract

Content aware resizing gained popularity lately and users can now choose from a battery of methods to retarget their media. However, no single retargeting operator performs well on all images and all target sizes. In a user study we conducted, we found that users prefer to combine seam carving with cropping and scaling to produce results they are satisfied with. This inspires us to propose an algorithm that combines different operators in an optimal manner. We define a *resizing space* as a conceptual multi-dimensional space combining several resizing operators, and show how a path in this space defines a sequence of operations to retarget media. We define a new image similarity measure, which we term Bi-Directional Warping (BDW), and use it with a dynamic programming algorithm to find an optimal path in the resizing space. In addition, we show a simple and intuitive user interface allowing users to explore the resizing space of various image sizes interactively. Using key-frames and interpolation we also extend our technique to retarget video, providing the flexibility to use the best combination of operators at different times in the sequence.

2.1 Introduction

Media retargeting has become an important problem due to the diversity of display devices and versatility of media sources for both images and video. Recently, content aware methods such as seam carving and non-uniform warping were proposed to supplement content oblivious methods such as scaling or cropping. A content aware retargeting operator relies on an importance map to preserve the important parts of the media at the expense of the less-important ones. Importance measures include image gradients, saliency and entropy, as well as high level cues such as face detectors, motion detectors and more.

However, content aware methods do not succeed in all cases and for all sizes. For example, in case the important object occupies large portions of the image or video frame, content aware resizing might distort it. Often, the best resizing method depends on the image itself: one method might work best on one image, while another on a different image. In such cases using a combination of several methods (operators) might achieve better results than any specific one alone (Figure 2.1). In this paper we propose to combine several operators together, instead of searching for the best operator that will work on all images. Our approach is supported by a user study we conducted that clearly shows that users prefer to use more than one operator to achieve better results.

We first define the *resizing space* as a conceptual multi-dimensional space combining several retargeting operators. Each axis in this space corresponds to a particular type of operator, and a point in this space corresponds to a particular target image size. A path in this space defines a sequence of operations that retargets an image to a particular size (Figure 2.5). Many paths arrive at the same point, meaning that there are many ways to retarget an image to a particular size. But not all paths are created equal because resizing operators are *not* commutative (e.g. scaling followed by cropping is different from cropping followed by scaling).

To combine several operators there is a need to compare and evaluate different retargeting results. Hence, we need some global similarity measure between the source and target images. And given the similarity measure, we need an algorithm that maximizes this measure by finding the best path (i.e. sequence of operators) to the respective point in resizing space.

In this paper we propose a novel similarity measure between images that we term Bi-Directional Warping (BDW). This measure is based on a non-symmetric variant of Dynamic Time Warping (DTW) [27]. DTW takes two $1D$ signals (e.g. rows or columns of pixels) and finds the best non-uniform alignment between them, subject to order constraints. To measure the similarity between two images, BDW measures the similarity between every row (or column) and then takes the *maximum* alignment error as the distance. We also extend the measure to work on a row (or column) of patches instead of pixels, as patches can better capture spatial information.

There are infinitely many paths that can be used to retarget an image. Unless mentioned otherwise, we focus on *monotonic* paths, i.e. paths where all operators either increase the size of the image, or decrease it, but not both. Of all the monotonic paths, we consider two types of paths that we term *regular* and *mixed*. A *regular* path is composed of consecutive single operator sequences, one per operator (e.g. first apply seam carving, then cropping and finally scaling). In this case, the only question left is how many times to apply each operator in the retargeting process? The search space is polynomial in the image size and can be enumerated to find the optimal *regular* path. However, in a *mixed* path, the order of the operations, as well as the number of times each operator is used is not fixed. Hence, the search space is exponential in the image size. However, using a simple assumption we show a polynomial algorithm that automatically determines the optimal *mixed* multi-operator path. In both cases the search space is exponential in the number of retargeting operators. Nevertheless, the number of operators is typically very small (say four operators), making the solutions tractable.

It is worth noting that the multi-operator algorithm can work with various image similarity measures as well as different retargeting operators. *Regular* paths can also be controlled by the user and we show a simple user interface for image retargeting. Finally, we extend the *regular* path approach to video retargeting by interpolating paths between key-frames. This approach provides the flexibility to use the best combination of operators at different times in the video. We demonstrate our approach for high quality reduction and expansion of images and videos.

Our main contributions are as follows, 1. We show that using several operators can potentially give better results for retargeting than using a single operator, 2. We present a new global measure, Bi-Directional Warping, to assess the retargeting results, 3. We give an algorithm for finding an optimal multi-operator retargeting sequence under some assumptions, 4. We describe an intuitive user interface that helps users combine multiple operators interactively, and 5. We show how our method is extended to support multi-operator video retargeting.

2.2 Background

Content-aware retargeting has drawn a lot of attention in recent years. Most methods proposed use a two-step approach where first some saliency or importance map is created from the media and then a resizing operator is applied based on this map. As our work concentrates on combining multiple operators and not on saliency, we focus on the different types of operators for resizing media.

Cropping was used by Suh *et al.* [35] for automatic thumbnail creation, based on either a saliency map or the output of a face detector. Similarly, Chen *et al.* [8] considered the problem of adapting images to mobile devices, by automatically detecting the most important connected region in the image and transmitting it to the mobile device. Liu *et al.* [19] also addressed image retargeting to mobile devices, suggesting to trade time for space. Given a collection of Regions Of Interest (ROI), they construct an optimal path through these regions and display them in a consecutive manner. Santella *et al.* [29] use eye tracking, in addition to composition rules to crop images intelligently. Setlur *et al.* [31] use segmentation and re-composition for non-photorealistic retargeting.

Several different methods could be characterized as non-homogeneous scaling. Liu and Gleicher [17, 18] find the 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. In their video retargeting work they use a combination of cropping, virtual pan and shot cuts to retarget the video frames. Gal *et al.* [10] solve the general problem of warping an image into an arbitrary shape while preserving user-specified features. 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. Wolf *et al.* [48] extend this approach to video using non-homogenous mapping of the source video frames to the target resized frames. They use a combination of motion detectors and face detectors to define the saliency map. A different approach presented by [46] partitions the image into a grid mesh and deforms it to fit the new desired dimensions. Important image regions are optimized to scale uniformly while regions with homogeneous content are allowed to distort.

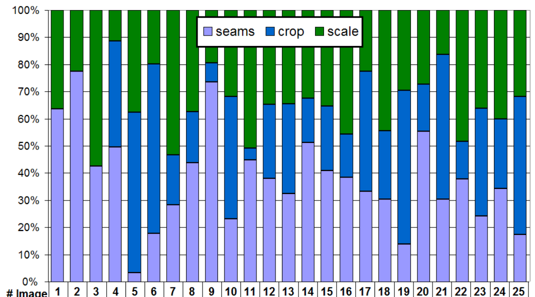
Recently, several works used the seam carving operator originally proposed by Avidan and Shamir [3] to resize images in a content aware fashion. They use dynamic programming to find the optimal seam in an image according to some image energy map (usually based on the gradient field of the image). The seams can be removed for shrinking images, or duplicated for expanding them. Later, this work was extended by Rubinstein *et al.* [25] for video retargeting. The dynamic programming was replaced by a graph cut approach and a new image energy was proposed that creates less artifacts in the resulting media. Graph cuts were also used by Chen and Sen [7] for temporally resizing video.

In cases where one of these operators does not perform well, it might be better to use another or revert to simpler resizing methods such as cropping and scaling. On the one hand the latter methods are not content aware, but on the other, they can be considered less harmful as they do not distort the media. The key question is how to decide when one operator fails, and which operator to use instead?

Some measures were suggested, for example, by Avidan and Shamir [3] to indicate the order of seam carving by their cost. However, we have not found this cost to be indicative for measuring retargeting quality. Moreover, similar measures are not easy to find for other operators. We follow more global measures such as the bi-directional similarity [33] and inverse texture synthesis [47] that define image similarity. Two images S and T are considered visually similar if all patches of S (at multiple scales) are contained in T, and vice versa. Although this approach is effective on several applications for summarization and synthesis, it does not preserve the order of elements inside the image. Trying to match two images while preserving full order is a difficult problem [13]. Still, using some constraints we present a variant of Dynamic Time Warping [27, 39] that can be utilized to measure retargeting quality.

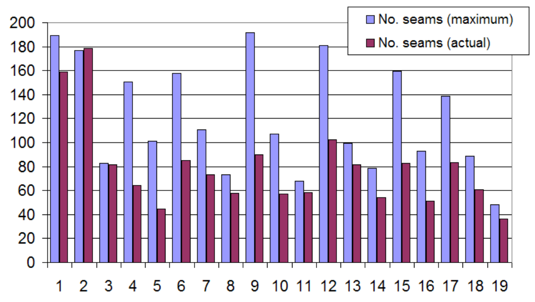
2.3 The User Study

Our basic hypothesis in this work is that using multiple operators for resizing images is often better than using a single one. To assess this hypothesis we conducted an experiment where users are given the option to use a combination of three operators: seam carving, cropping and scaling, including also the option of using just a single one. We present users with an original image in one window, and in another window, a resized image that is either reduced or enlarged in one dimension to a fixed size (the change in size was between 30% to 60% of the original size). We specifically chose images that contain either structure or content that presents difficulties for existing methods. The resized image is retargeted using a combination of the above three operators using *regular* sequences. For example, to reduce the width of an image by n pixels, first n_1 seams are removed, then n_2 columns are cropped from the left and



# Image	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Ratings	7.7	7.3	6.9	7.9	7.6	7.6	6.9	6.9	8.6	8	8.3	8.5	7.8	8.1	6.9	7.5	7.9	8.1	7	7.3	7.8	8	7.7	8	8

(a) The mean of the ratio of operators used for retargeting each image using multiple operators. Although the ratio depends on the image, in all cases, when given the option, users combine several operators to achieve better results. Users were asked to rate the results between $1-10$. As can be seen most users were satisfied with the retargeting results (average 7.7). For image enlargement (the first three results) only seam carving and scaling were allowed. The images themselves are shown below.



(b) Comparing the maximum number of adequate seam removals (or insertions) to the actual number when using multiple operator retargeting. The mean values of the overlapping participants and images in both experiments are shown. In most cases, users prefer switching to other operators for retargeting even though the seam carving results were rated as adequate. Note that when images are enlarged, i.e. seams are inserted (first three results), the numbers are much closer, and in one case the number is even slightly larger.

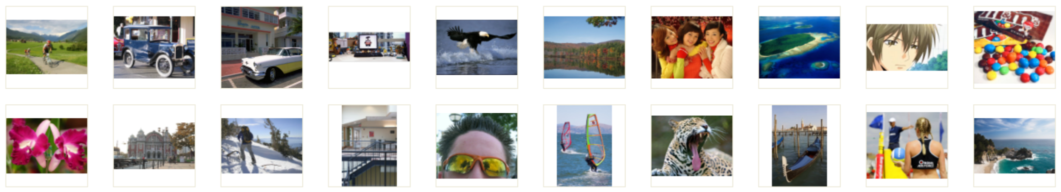


Figure 2.2: A user study of 50 participants clearly indicates that combining multiple operators can be beneficial for retargeting.

right sides of the image and lastly the image is scaled by n_3 pixels, where $n_1 + n_2 + n_3 = n$. Users were asked to change the ratio between n_1, n_2, and n_3 interactively using a scroll bar while examining the resulting image, until they reach the best results for the given fixed size. Note that for image enlargement only seam carving and scaling were used while the ratio between them could be changed. Figure 2.3 shows a snapshot of the user study application. The user interface itself is described later in Section 2.7.

Figure 2.2(a) summarizes the results of this experiment for 50 participants. Most participants were computer-science students or graphic designers. They were asked to rate their graphical background level from novice to expert: 28 rated themselves as novices, 15 as intermediates, and 7 as graphic experts. 22 images of different nature were used in the experiment (Figure 2.2 bottom). As can be seen from the combined mean results (no significant

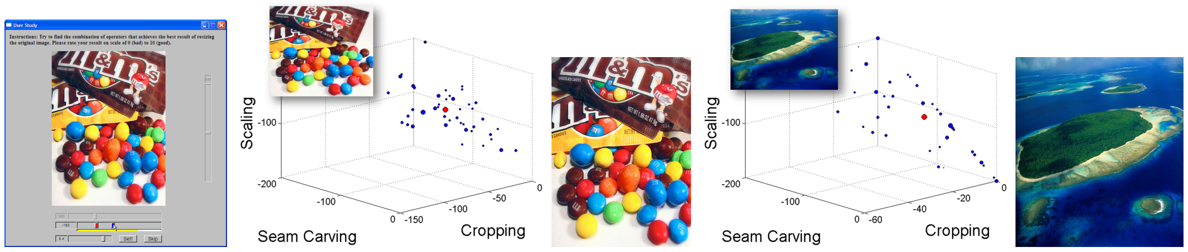


Figure 2.3: Data gathered in our user study. On the left is a snapshot of the user study application. For each image and target size, the users were asked to find the best-looking balance of three retargeting operators: seam-carving, cropping and scaling, and rate their result on a scale of 0 (bad) to 10 (good). In the middle and right, users results are shown for two images that were used in the study. The data points are plotted in the three-dimensional operator space, all reside on a plane induced by the resizing interface (see Section 2.7). The size of the data points indicate their relative weights in the statistics. The (weighted) center-of-mass appear as red points in both plots, and their corresponding retargeted results are shown. The mean user results were $\langle -58 \cdot \text{SC}, -60 \cdot \text{CR}, -62 \cdot \text{SL} \rangle$, and $\langle -90 \cdot \text{SC}, -9 \cdot \text{CR}, -102 \cdot \text{SL} \rangle$ for the mnm and islands images respectively.

differences were found between the groups) we have $n_i > 0$ for all $i = 1, 2, 3$. Moreover, for almost all images and all participants we had $n_i > 0$ for all i. In general, this suggests that better results are achieved using a combination of more than one operator.

To generate the aforementioned statistics, the data points were weighted as function of both a normalized measure of the user rating, and the user expertise in graphics, such that larger influence was given to expert users who were satisfied with their result. Additionally, we employed basic outlier removal by percentile thresholding. Figure 2.3 shows the resulting user data in the induced three-dimensional resizing space (Section 2.4). As expected, we noticed that more abstract images exhibit larger variance in the user results in comparison to more structured images, for which the results were more concentrated. However, in general we found that the variance was relatively large, indicating that different users find different results attractive. As approximately 45 data points remained for each dataset, we did not identify specific clustering of the data, and took the (weighted) center-of-mass as the representative result.

In a different experiment, users were given the option to change the size of an image using seam-carving alone and were asked to minimize (or maximize) the width (or height) of the image as long as the resulting image appears visually adequate. Figure 2.2(b) compares the mean of the number of seams removed (or inserted) in this experiment to the number of seams actually removed (or inserted) while using multiple operator resizing on the same images in the first experiment. Results clearly show that although users were satisfied with the quality of removing (or inserting) more seams from an image, they still preferred using other operators while retargeting the image to achieve better visual results.

2.4 The Resizing Space

2.4.1 Multi-Operator Sequences

We define a *retargeting operator* O as a procedure that reduces or enlarges an image either in its width or its height, while preserving its rectangular shape. We concentrate on retargeting operations that are *discrete* and *separable* (in dimension). This means that the atomic operation in our setting is adding or removing one pixel to the width or the height of the image. Two dimensional resizing can be treated as a sequence of width and height resizing, which means that different operators can be used for different dimensions (e.g. use scaling for height change and seam carving for width change). In this paper we use bi-cubic scaling (SL), cropping (CR) and seam carving (SC). This particular combination seems promising, as these operators take somewhat complementary approaches to resize media. Using other operator combinations is left for future work.

Not all retargeting operators can actually support enlarging. For instance, cropping is usually used only for reducing image size. However, for the sake of completeness we define crop-enlarging as adding a black frame to an image (letter-boxing). Similarly, to make cropping separable, we remove rows or columns from the image borders independently. We also choose the sides separately, for instance, either the left or the right column is removed depending on which has the lower cost according to $E_1(I) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|$. Scaling can support separable and discrete resizing, but scaling an image by one pixel k-times is inferior to scaling by-k at once. Hence, whenever applicable we perform a scale by-k instead of applying k 1-pixel scalings.

Combining several operators together in an ordered sequence is a *multi-operator sequence*. Note that a certain type of operator can appear multiple times in different places in the sequence; in some it can be used to enlarge the image and in others to reduce it, and also in different directions (width and height). Figure 2.4 shows examples of different multi-operator sequences that create different valid variations for retargeting an image.

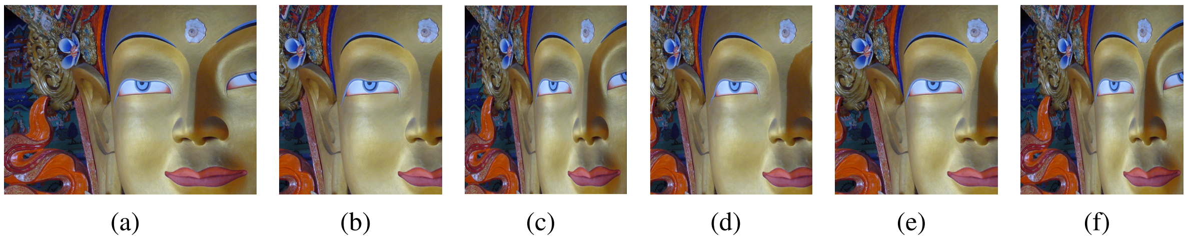


Figure 2.4: Different multi-operator sequences create a variety of results for retargeting the width of an image: (a) original image, (b-f) results using the following sequences respectively: $\langle 0 \cdot \mathrm{SC}, -178 \cdot \mathrm{CR}, 0 \cdot \mathrm{SL}\rangle$, $\langle 0 \cdot \mathrm{SC}, 0 \cdot \mathrm{CR}, -178 \cdot \mathrm{SL}\rangle$, $\langle +48 \cdot \mathrm{SC}, -149 \cdot \mathrm{CR}, -77 \cdot \mathrm{SL}\rangle$, $\langle -30 \cdot \mathrm{SC}, -148 \cdot \mathrm{CR}, 0 \cdot \mathrm{SL}\rangle$, $\langle -32 \cdot \mathrm{SC}, 0 \cdot \mathrm{CR}, -146 \cdot \mathrm{SL}\rangle$.

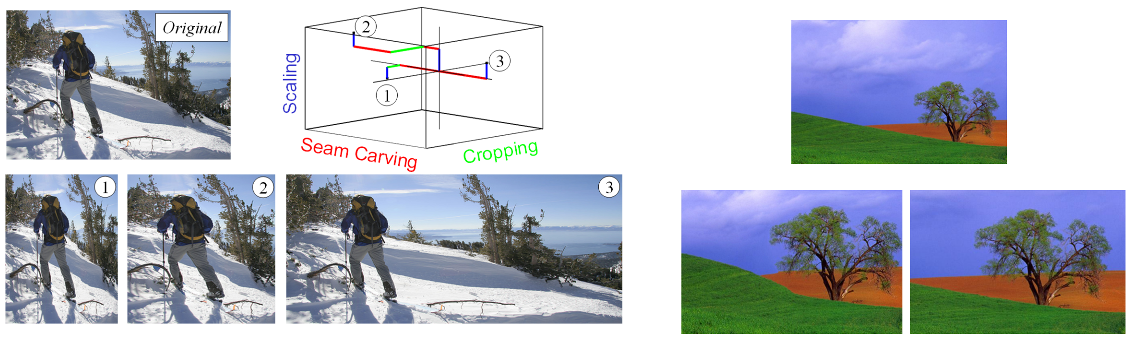


Figure 2.5: On the left, an example of a resizing space of an image using only changes in width by scaling, cropping and seam carving. Different retargeting results can be achieved using different multi-operator sequences represented by paths in the space. On the right, content enhancement of the upper image can be achieved using multi-operator paths in both width and height. Different results can be achieved using different paths: either scaling up and removing seams (bottom left) or scaling up and cropping (bottom right).

2.4.2 The Resizing Space

For a given image I of size (w, h) we define the *resizing space* Φ as the space spanned by any subset of n types of retargeting operators, each one in two directions - width and height. Hence, the dimension of this space is at most $2n$. A multi-operator sequence defines a directed path in this space beginning at the origin and following the path's operator sequence using integer steps. One step in the operator sequence is equivalent to a step either in the positive or negative direction of the respective operator axis, which can change either the width or the height of the image. Since only integer steps are used, we treat the resizing space as a lattice rather than a continuous space (Figure 2.5).

For a k-dimensional resizing space, $k \leq 2n$, a point on this lattice $p \in \Phi, p = (p_1, \ldots, p_k)$, $p_i \in \mathbb{Z}$, represents the set of images $\{I'\}$ whose dimensions are (w', h') where $w' = w + \sum_i p_i^w$ and $h' = h + \sum_j p_j^h$, and p_i^w, p_j^h are the coordinates in $(p_1, \ldots, p_k)$ representing operators that change the width or height respectively. These coordinates can be positive as well as negative to signify enlargement or reduction of size. There is an infinite number of such images for each point p in the resizing space since there is an infinite number of paths starting at the origin and ending at p. All such paths define multi-operator sequences where the change of width or height by each specific operator i is fixed and amounts to the coordinate p_i. However, the order of applying the operators can be different. To complicate things further, there is an infinite number of points $q \in \Phi$ that represent images of dimensions (w', h'). For r horizontal operators and t vertical operators ($r + t = k$), these are points for which $\sum_{i=1}^{r} p_i^w + w - w' = 0$ and $\sum_{j=1}^{t} p_j^h + h - h' = 0$. Each of these equations represent a $r + t - 1 = k - 1$ dimension hyperplane in the k-dimensional resizing space, the intersection of which is a hyperplane of dimension $k - 2$ (for $r, t > 0$). All points in this subspace represent images of the required size, where the difference between them is in the amount of applying specific operators (i.e. the ratio between them). Our main goal therefore, is to find the best path from the origin to one of these points, subject to some global image similarity measure.

2.5 Bi-Directional Warping

2.5.1 Motivation

As a motivating example, consider the task of combining seam carving and scaling. One way to combine the two is to start with seam carving and then switch to scaling when the cost of a seam goes above a certain threshold, as this might indicate that seam carving starts introducing visual artifacts. In fact, Avidan and Shamir [3] used the seam cost to find the optimal "multi-operator" sequence using just seam-carving to change both the width and the height of an image. Unfortunately, in our experiments we found that the seam cost is not very indicative of the quality of retargeting. It is a monotonically increasing function (with some local fluctuations), and usually does not contain steps that indicate when "bad" seams are removed. Figure 2.6 illustrates this with specific examples.

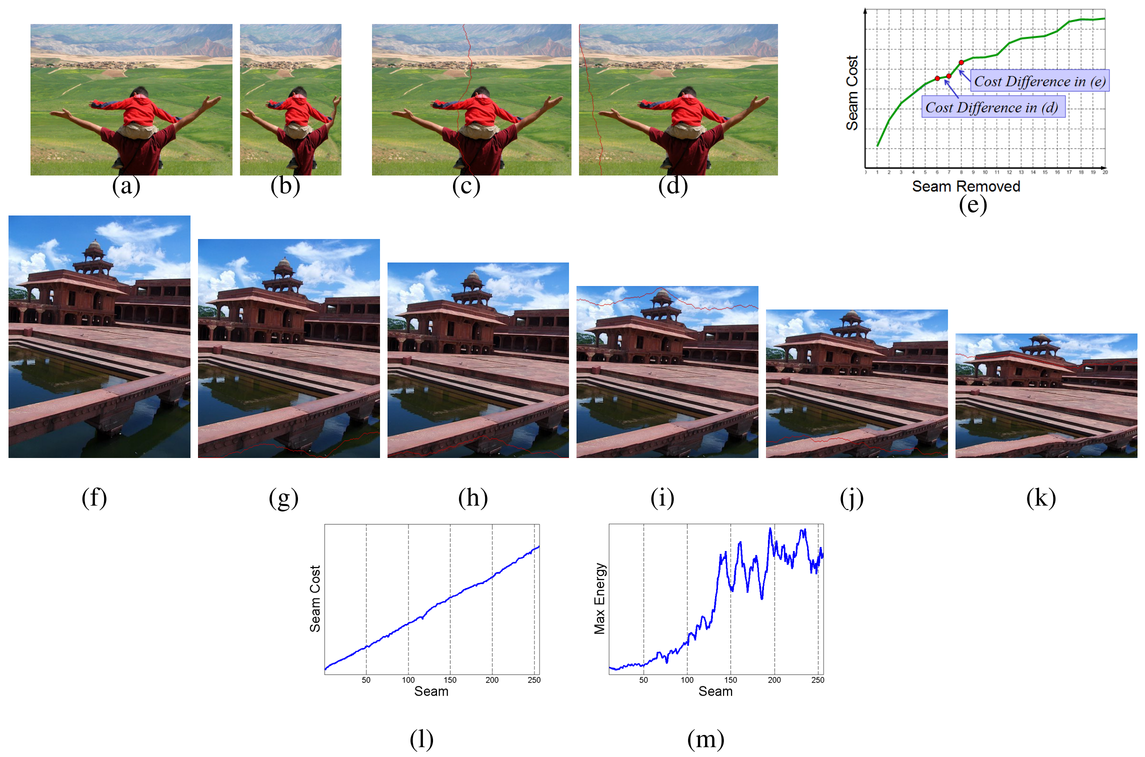


Figure 2.6: Retargeting using seam carving can destroy image content (a-b,f-k). Examining the seam cost function (e) as in [Avidan and Shamir 2007] one cannot anticipate when foreground objects are distorted. If we track not only the actual cost, but also the difference in the cost of removing seams, we can find seams that pass through the main object (c) but have smaller cost and smaller difference in cost than seams that pass only through the background (d). In the middle row, the seam results (g-k) are shown for every 50th seam-carving iteration while vertically resizing an image (f). The cost function (l) in this case is nearly linear. Examining the maximal energy for each seam instead of the average (m) reveals more useful information, but exhibits fuzzy transition between low energy and high energy, and is very oscillatory in nature. Hence, seam cost cannot be used as a measure for retargeting quality.

Generally, the seam energy functions retain these shapes due to 1. the inherent locality of the seams, and 2. since the energy averages over all seam pixels. Considering the maximal pixel energy at each seam instead, shows unstable results and also exhibits general monotonic trend with no specific points of singularities (Figure 2.6(m)). Hence, to combine seam carving with scaling the seam cost should not be used. In the case of other operators, such as cropping and scaling that operate on an image as a whole, even the definition of an effective cost for the operator is not immediately clear. So instead of dealing with each potential operator independently, we need a global objective function that will allow us to combine operators in a principled manner.

We propose Bi-Directional Warping (BDW) as the image similarity measure. BDW uses a variant of Dynamic Time Warping (DTW) that is geared specifically to the problem of media retargeting.

2.5.2 Image Similarity

We define the cost of applying an operator as the difference between the retargeted image and the original image. Although the definition of a distance function between a pair of images is an ongoing research problem for many years, in our setting there are several simplifying factors. First, we do not search for a general distance measure between two images, as we know that the target image is a resized version of the source image and is aimed, by definition, to preserve its content as much as possible. Second, each application of an operator changes the image size only in one direction. We take a similar approach to the recently proposed bidirectional similarity measure [33], in which a bidirectional mapping between patches in a source and resized image testify for the coherency and completeness of the result. And we show that enforcing ordering of such mapping is important for assessing retargeting results.

Stated more formally, in a given image representation S and T, we define a mapping $M_{S\to T}(i,j) \mapsto (i',j')$ that maps every element of S (either a pixel or a patch) to an element of T. Also let M^i, M^j denote the i-component (row) and j-component (column) of the map respectively over the image. We define the cost of this mapping as

$$\gamma(M) = \sum_{i,j} d(S_{i,j}, T_{M(i,j)}) \tag{2.5.1}$$

where $d()$ is a function for measuring the distance between the elements (typically based on intensity differences). We can then define our objective distance measure as

$$D(S,T) = \frac{1}{N_S} \min_{M_{S\to T}} \gamma(M_{S\to T}) + \frac{1}{N_T} \min_{M_{T\to S}} \gamma(M_{T\to S}) \tag{2.5.2}$$

Subject to, for all i, j:

$$\begin{aligned} M^i_{S\to T}(i,j) &= M^i_{T\to S}(i,j) = i \\ M^j_{S\to T}(i,j) &\le M^j_{S\to T}(i,j+1) \\ M^j_{T\to S}(i,j) &\le M^j_{T\to S}(i,j+1) \end{aligned} \tag{2.5.3}$$

where the first equation in 2.5.3 poses the dimensionality constraint, and the second and third inequalities force the mappings to be monotonic. N_S and N_T are the total number of elements in S and T respectively. Note that the above constraints correspond to change in width, and can be adapted to change in height by appropriately substituting M^i and M^j.

2.5.3 Dynamic Time Warp

Dynamic Time Warping (DTW) [27], is an algorithm for measuring similarity between two one-dimensional signals or time-series. It has been previously applied to various applications in video and images, and is extensively used with audio signals for speech recognition. The DTW algorithm finds the optimal matching between two 1D sequences t and s by non-linearly warping the one to the other, under several constraints: (1) boundary constraints: the first and last elements of t must be matched to the first and last elements of s, respectively, (2) all elements of t and s must be used in the warp path, and (3) the warp must be *monotonic*, meaning that matching cannot go backward, thus preserving the sequence order. It is easy to see that the warp is symmetric, that is, DTW(s,t)=DTW(t,s), and can contain both one-to-many and many-to-one matchings. This algorithm can be solved efficiently using dynamic programming, in $O(|s||t|)$ time and space.

2.5.4 Bi-Directional Warping

We relax the first two constraints of DTW. First, we allow the algorithm to insert gaps in the warp, which also removes the boundary constraints. Second, for each element in the source we want a *single* match that minimizes the warping cost under the ordering constraint. Therefore, one-to-many matchings from the source to target image are disallowed. We do allow many-to-one matchings from the source to target image as it assists better matches and does not violate the ordering constraint. This creates an Asymmetric-DTW measure (A-DTW) detailed in Algorithm 1. The signals s and t can be either $1D$ arrays of pixels, or $1D$ arrays of patches, and the distance $d(s[i], t[j])$ between an element of s and element of t is taken to be the sum-of-square-differences of pixel values[1].

Given images S and T of height h, let S_i, T_i denote row i in images S and T, respectively.

[1]We have experimented with numerous measures, such as the L_1 and L_2 norms of the intensity differences in both grayscale, RGB and L*a*b colorspaces. For other applications, the normalized cross correlation between the patches can be used in order to account for lighting changes.

The BDW distance is given by:

$$\text{BDW}(S,T) = \frac{1}{N_S}\sum_{i=1}^{h}\text{A-DTW}(S_i,T_i) + \frac{1}{N_T}\sum_{i=1}^{h}\text{A-DTW}(T_i,S_i) \tag{2.5.4}$$

and it is easy to verify that BDW solves for the optimal mappings as defined by our objective distance function (Equation 2.5.2).

Algorithm 1 Asymmetric-DTW($s[1..|s|], t[1..|t|]$)

```
1: allocate $M[|s|+1][|t|+1]$
2: $M[0,0] := 0$
3: for $i = 1$ to $|s|$ do
4:     $M[i,0] := \infty$
5: end for
6: for $j := 1$ to $|t|$ do
7:     $M[0,j] := 0$
8: end for
9: for $i := 1$ to $|s|$ do
10:     for $j := 1$ to $|t|$ do
11:         $M[i,j] := \min(M[i-1,j-1] + d(s[i],t[j]),$
                           $M[i,j-1],$
                           $M[i-1,j] + d(s[i],t[j]))$
12:     end for
13: end for
14: return $M[|s|,|t|]$
```

We found that using the $\max$ operator works better than the mean in equation 2.5.4, because in retargeting most elements are usually well aligned, yet a small number of deformed elements are enough to cause a visual artifact. To find the maximum distance between elements from S and T, we need to recover the elements' alignment created by the asymmetric-DTW. To do this, we keep track of our path while filling the table M, and backtrack from $M[|s|,|t|]$ to $M[1,1]$ according to the optimal decisions made along the path. Figure 2.8 illustrates the results of aligning an image with its retargeted version (by seam carving) using A-DTW and several patch sizes. In practice, to calculate the BDW we combine the scores of four scales of patch size (Figure 2.9). For images S of size $h \times w$ and T of size $h \times w'$, $w' < w$, BDW is $O(hw^2)$ in time, and $O(w^2)$ in space.

Practically, the BDW between two 640×480 images and using 8×8 patches takes about 3

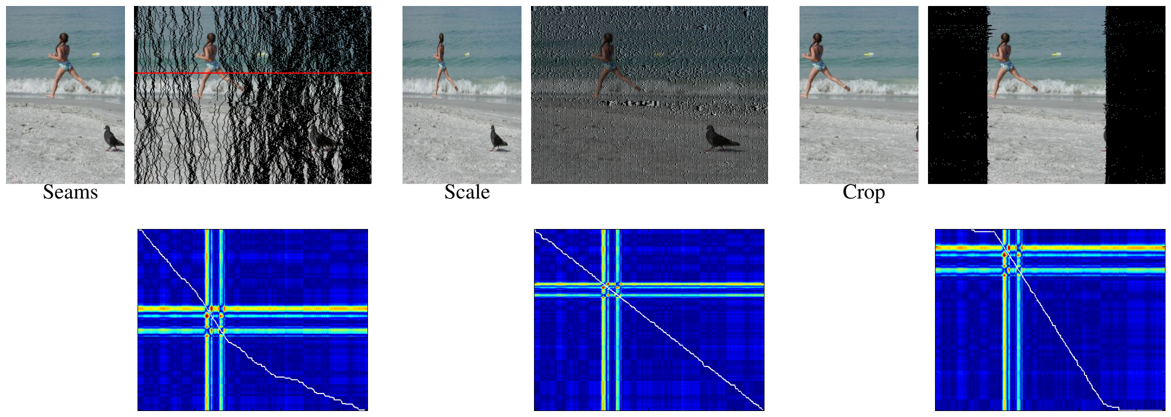
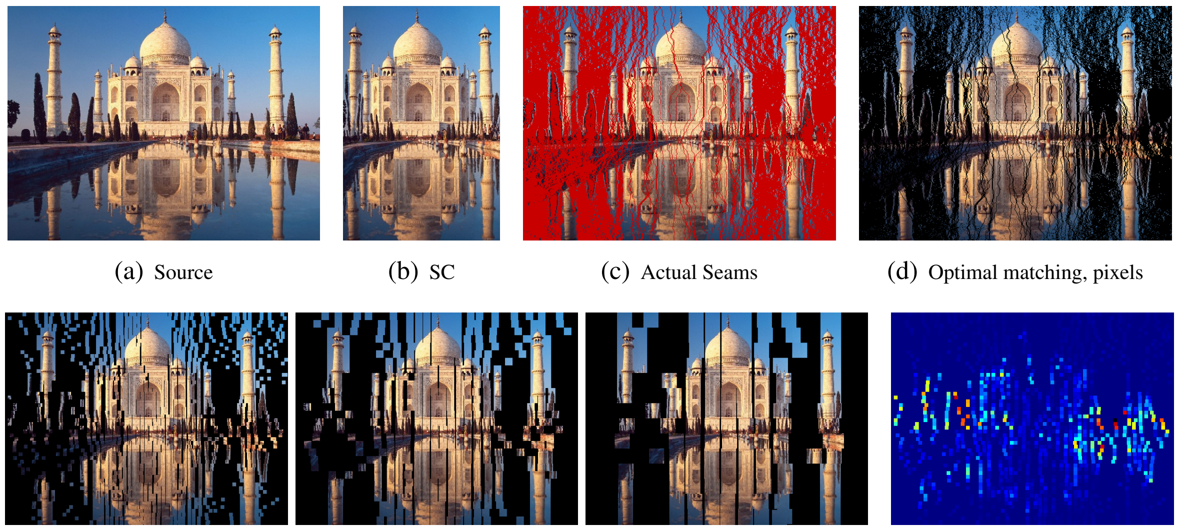


Figure 2.7: A-DTW results for retargeting an image. For each operator, the result is shown on the left and the optimal alignment with the original image on the right. In this case, the alignment was calculated at pixel level, using the L_1-norm of RGB differences. Black pixels represent gaps in the alignment. On the bottom is a visualization of the resulting warp for a specific scanline (shown in red on the seams alignment) as a white path superimposed on the pairwise pixel distance maps. Right orientation of the warp introduces gaps in the alignment; Down orientation represents many-to-one matching; Diagonal orientation indicates matched pixels.

seconds to compute. For further speed-up, we can limit the search space to a narrow window, or *band*, within the dynamic programming table [27]. As the retargeted image and its source should be relatively similar, it is unlikely that a good warp path will wander far from the diagonal (Figure 2.7). In fact, apart for the speed-up, this can also show useful in preventing pathological warps. For example, we might want to prevent cases where many patches of one image are mapped to the same patch in the other, or cases where the warp contains too many gaps. Other optimization techniques exist in the literature. In particular, [28] use an approximate multilevel approach and report linear time and space complexity.

Both BDW and bidirectional similarity (BDS) explain patches in one signal using patches from the other. However, there are two main differences between these measures. First, BDW searches for matches along a single direction (column or row) as opposed to the entire image in BDS. This is sufficient in our settings for assessing results created by operators which are separable in dimension. It is also more efficient to compute as we replace the computationally intensive nearest-neighbor search in BDS with an efficient matching algorithm. Second, BDW achieves optimal alignment that is order preserving. Order is important when assessing retargeting results, because we prefer as few as possible structural modifications of the media. The many-to-one alignment also supports repetitive content, albeit in an order-preserving



(a) Source (b) SC (c) Actual Seams (d) Optimal matching, pixels

(e) 8×8 matching patches (f) 16×16 matching patches (g) 32×32 matching patches (h) Distance map of 8×8 patches

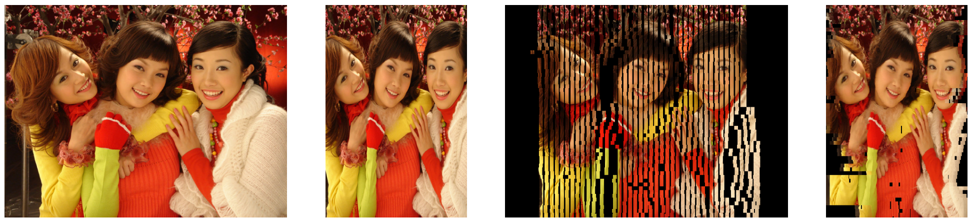
Figure 2.8: Finding the optimal match using Asymmetric-DTW of image (b) to image (a) using different patch sizes (d)-(g). Note that black pixels represent gaps in the matching. The distance itself is defined as the average or maximum of the cost of matching each patch (h).

manner. Figure 2.10 highlights the difference between BDW and BDS in the context of image retargeting.

2.6 The Optimization

Suppose we want to reduce the width w of input image S by m pixels, using a collection of n operators $\{O_1, \ldots, O_n\}$, and given some similarity measure $D()$ (e.g. BDW), indicating the distance between two images, where larger distance score means lower similarity and vice versa. This means that we seek a target image T of width $w' = w - m$ that minimizes $D(S, T)$. Even if we use monotonic sequences (e.g. do not reduce, then extend, then reduce back again), there are still $O(n^m)$ different multi-operator sequences that retarget S to width w'. This means the search space is exponential in the size change m.

To solve this problem we need to limit our search space and we consider two types of paths: *mixed* and *regular*. We will focus on *mixed* paths here and defer discussion on *regular* paths to section 2.7. Recall that we define a *mixed* path to be a path where we don't know, ahead of time, the order of the operators, nor the number of times each operator is to be used.



(a) S (original) (b) T (retarget) (c) T→S (d) S→T

Figure 2.9: BDW uses two alignments of the source image to the retargeted image and vice versa. The distance would be the maximal matching cost of elements in the two. Note how the alignment reveals that the retargeting used a combination of cropping (gaps on the sides of the alignment), scaling (uniform spacing between patches in the middle) and seam carving (large gaps in the middle).

We could use a full greedy approach, and build the multi-operator sequence incrementally by choosing the best of n operators in each step locally. This would work if we assume that any subsequence in an optimal sequence is also optimal. This is obviously not the case. We could use known approximation methods such as beam search to explore a specific sub-space for a local optimal solution. However, by using a specific assumption we can formulate the search as to achieve a closed-form solution.

The basic assumption we use is that the ratio of operators in a sequence (i.e. the total amount each one is used) is more important than their order in the sequence. This leads to a dynamic programming formulation of the problem. In our search we always keep just one representative for each sequence with a given ratio of operators. We represent it by the point $(p_1, \ldots, p_n)$ in resizing space where p_i, the coordinate for operator i, denotes the total number of times of applying operator i. In a dynamic programming table we store the optimal cost and optimal sequence $\sigma(p_1, \ldots, p_n)$ including the order of applying all the operators for this representative point.

We begin with an empty sequence denoted by the point $(0, \ldots, 0)$ and cost 0. Next, we apply each operator once and create n sequences denoted by the points $(1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)$ with costs $\{D(S, \langle O_i \rangle (S))\}_{i=1}^{n}$ of applying operators $O_i,\ 0 \leq i \leq n$ respectively on the original image S. Next, we store the cost of sequences $\sigma(2, 0, \ldots, 0), \ldots, \sigma(0, \ldots, 0, 2)$, but for each sequence containing the application of two distinct operators $O_i, O_j, i \neq j$ we have 2 possible sequences: $\sigma = \langle O_i, O_j \rangle$ or $\sigma = \langle O_j, O_i \rangle$. We check the two options, and keep only the one whose cost is smaller in the table at position $(\ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots)$, where the 1s appear in positions i and j.

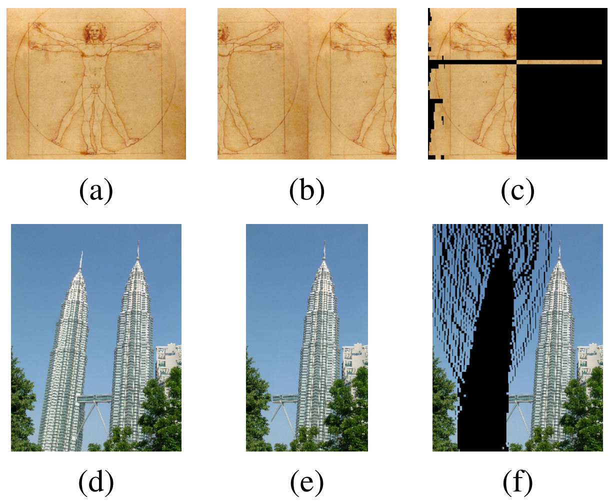


Figure 2.10: Comparing Bidirectional Warping (BDW) and Bidirectional Similarity (BDS) of [Simakov *et al.* 2008]. In image (b) we switched the left and right parts of image (a), and image (e) is missing some repetitive structure (a tower) found in image (d). In both cases the BDS is one order of magnitude smaller than BDW as measured in equation 2.5.4 (it is two order of magnitudes smaller if we use $\max$ instead of mean). This is because every patch in one image will have, with high probability, a similar patch in the other image, and vice versa. On the other hand, the BDW measure is order-preserving, thus recovering the best alignment (as shown in images (c) and (f)) resulting in larger gap errors.

In general, to fill the entry $(p_1, \ldots, p_n)$ we examine all its predecessor sequences where the application of one of the operators was less by one. These correspond to points where one of the coordinates is less by one, which were already calculated and stored in the table. Denote them for abbreviation by $\sigma_i = \sigma(p_1, \ldots, p_i - 1, \ldots, p_n),\ 1 \leq i \leq n$. We append the operator O_i to sequence σ_i to get the new operator sequence denoted by $\langle \sigma_i \cup O_i \rangle$, apply this new sequence to the original image and choose the best one:

$$i^* = \arg \min_{1 \leq i \leq n} D(S, \langle \sigma_i \cup O_i \rangle (S)) \qquad (2.6.1)$$

The table structure is an n-dimensional simplex that is constructed in m stages. For example in Figure 2.11 the table is an equilateral triangle, which is a 2-simplex. In practice, we sample the search space in lower rates than 1 pixel (usually 5 or 10 pixels), meaning we apply each operator more than once between stages. At the last stage, all points $(p_1, \ldots, p_n)$ where $\sum_{i=1}^{n} p_i = -m$ represent target images of size $w' = w - m$. We choose the one that stores the smallest cost. To obtain the optimal sequence of retargeting operators $\langle O_{i_1}, \ldots, O_{i_m} \rangle$

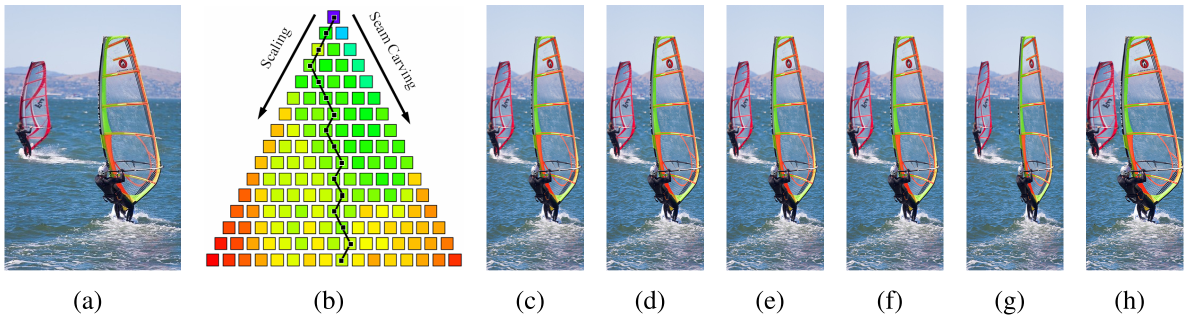


Figure 2.11: An illustration of the dynamic programming table used to optimize the search for the best *mixed* path using two operators only - seam carving (SC) and scaling (SL). The colors in table (b) indicate the BDW distance of the best image in each step. The original image is shown in (a) and the retargeted result is shown in (c) - this is the best result using a *mixed* path (i.e. the algorithm automatically determines the order of operators and how much each should contribute). The optimal operator sequence found is $\langle -30SL, -30SC, -10SL, -20SC, -10SL, -10SC, -10SL, -20SC, -10SL \rangle$. For comparison, we show the results of using two *regular* paths (d) $\langle -70SL, -80SC \rangle$ and (e) $\langle -80SC, -70SL \rangle$, and the optimal *regular* path (f) $\langle -90SC, -60SL \rangle$. (g) uses scaling and (h) seam carving.

we backtrack to the first entry and in each step recover the operator that had been chosen. Note that this approach is independent of the choice of resizing space, and so can be used to combine width and height operations in two directions.

The time and space complexities of the algorithm are $O(m^n)$ (ignoring the similarity measure complexity, which is independent of m and n) which is polynomial in the amount of size change, but exponential in the number of operators to be used. In fact, a tighter bound on the complexity can be derived by examining the volume of an n-simplex [34], given by:

$$\mathcal{V}(n) = \frac{m^n}{n!}\sqrt{\frac{n+1}{2^n}} \tag{2.6.2}$$

where m is the required size change as before (in case different change is required for the width and height then we take the larger of the two). Although this equation does not accurately describe the required DP table size due to the discretization, it gives an idea of the size of the search space with respect to its bounding hypervolume m^n. For example, for a 2-operator space, the actual search space is smaller by a factor of 2.3; for 3-operator space it is smaller by a factor of 8.48; for 4-operator space - by 42.91, and so on. Finally, the running time of the algorithm with the BDW measure is $O(m^n hw^2)$, dependent on both the required size change, the number of operators and the image size.

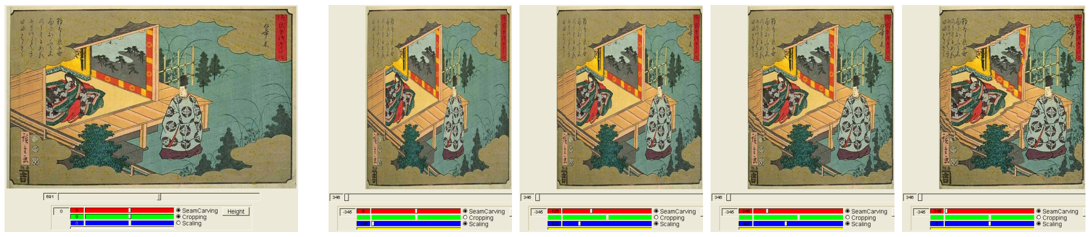


Figure 2.12: The multi-operator interface allows the user to change image size and operators combination to reach different points in the resizing space that correspond to *regular* paths. Using coupling of operators, the user is able to visually examine the continuous tradeoff between two operators (in this example, between scaling and seam carving) for all sizes.

Notice that the search for best seam order for reducing the width and height of an image, as described in [3], can be achieved in this framework by constructing a 2-dimensional resizing space using seam carving for both width and height (hence two axes in this space), and taking $D()$ to be the cost of the removed seam at each step (added to the previous cost). Note that the possible results in this case will reside within a sub-space of dimension $k - 2 = 0$, which corresponds to a single position in the table, namely (w', h'). Note moreover that their result is also optimal in a greedy manner, as not all possible permutations are explored.

Subject to the assumptions outlined above, our discrete optimization is guaranteed to find the optimum multi-operator sequence. However, these assumptions mean we only search in a sub-space of the resizing space and do not reproduce all images of the desired target size. There might exist a retargeted image that is more similar to the source image. Furthermore, the definition of best results may change depending both on the user and on the goal for retargeting the image. In our study (Section 2.3) we found that all users tend to use a combination of operators (Figure 2.2). However, the variance between users choices was very large ($\sigma \approx \frac{1}{2}\mu$). Therefore, in addition to the automatic optimization method we present an interactive technique that allows users to explore a sub-space of retargeting possibilities in a simple manner.

2.7 Interactive Multi-Operator Retargeting

Recall that *regular* paths fix the order of operators ahead of time and can conveniently be written as: $\langle k_1 \times O_{i_1}, \ldots, k_n \times O_{i_n} \rangle$, where $\sum_{j=1}^{n} k_j = m$ (see e.g. Figure 2.11(d)-(f)). So the only question is how much does each operator contribute to the image retargeting process? This creates a one-to-one mapping between a point in the resizing space and a *regular* path (Figure 2.13). Fixing m is equivalent to choosing a hyper-plane in the resizing space, and choosing how much each operator will contribute corresponds to choosing a point on this hyper-plane. This means that we can find the optimal solution for this problem using exhaustive search in $O(m^{n-1})$, which is polynomial in the size change m but exponential in the number of operators n. Since n is usually small (say, three or four operators) and we can sample m in discrete steps, this search is feasible.

Regular paths also lend themselves to a simple interface that assists users search in this sub-space for desired results. First, the order of operators in the sequence is chosen ahead of time (e.g. $\langle k_1 \times SC, k_2 \times CR, k_3 \times SL\rangle$ or $\langle k_1 \times SL, k_2 \times SC, k_3 \times CR\rangle$). Next, there is a slider governing the image size change m, and a slider for each operator separately. Since the contribution of all operators must sum to the total size change m, users must choose a coupling of a pair of sliders to change their value. Moving one in a positive direction will drive the other to move in the negative direction, and vice versa (Figure 2.12). We take advantage of the ability to precalculate seam index maps to allow fluent interactive retargeting for enlarging and reduction of size. The cropping and scaling operations can also be efficiently

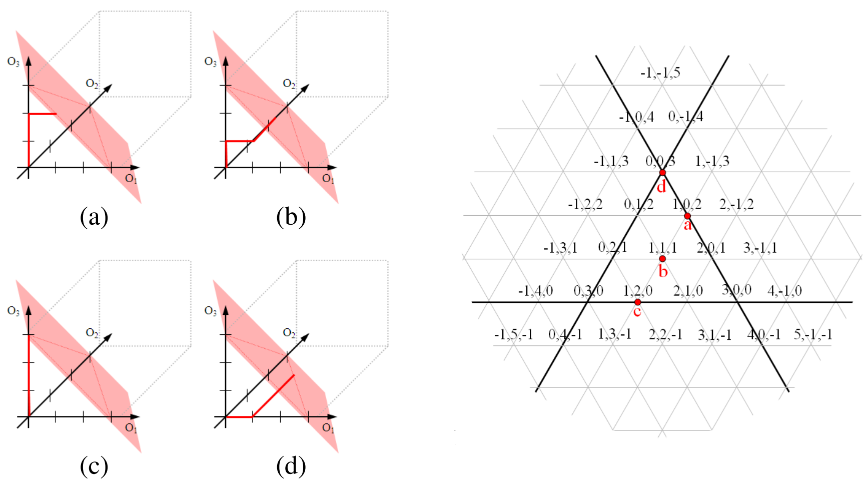


Figure 2.13: A depiction of a specific hyper-plane $k_1 + k_2 + k_3 = m$ in 3-dimensional operator space, for $m = 3$. By using a specific ordering of the operators in regular paths, there is a one-to-one mapping between each point on the plane and a specific multi-operator path. In this example the operator ordering is O_3, O_1, O_2, hence the four points shown in red on the plane represent the paths: (a)$\langle 2 \times O_3, 1 \times O_1, 0 \times O_2\rangle$, (b)$\langle 1 \times O_3, 1 \times O_1, 1 \times O_2\rangle$, (c)$\langle 0 \times O_3, 1 \times O_1, 2 \times O_2\rangle$, (d)$\langle 3 \times O_3, 0 \times O_1, 0 \times O_2\rangle$.

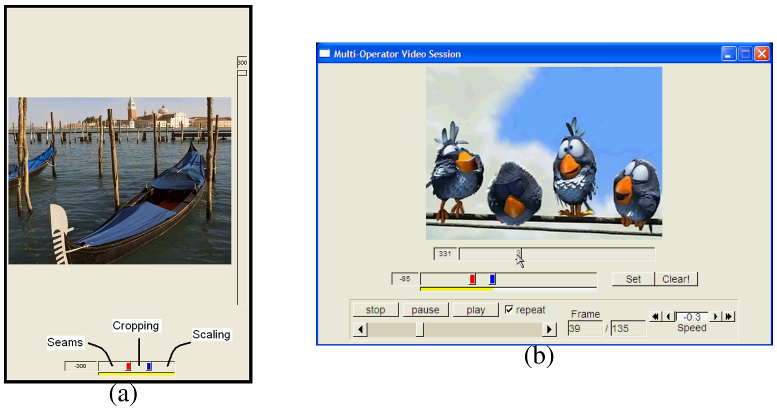


Figure 2.14: The regular-monotonic interface for images (left) and videos (right). The user uses one slider to control the target size, and another slider with two nobs for choosing the desired balance between the operators. For videos, the user is able to specify different operator sequences at different frames.

computed at interactive rates.

Note that such interface enables using both positive and negative amounts of specific operators. For example, we can bound the contribution of each operator O_{i_j} to $m \geq k_j \geq -m$ such that $\sum_{j=1}^{n} k_j = m$. In our user study of Section 2.3, we constrain $sign(k_j) = sign(m)$ and fix the order of operators to be $\langle k_1 \times SC, k_2 \times CR, k_3 \times SL \rangle$ to define a simpler interface using just one slider for all operators (see accompanying video). This constrains the retargeting to *regular-monotonic* sequences, and further confines the search space to the intersection of the hyperplane with the axes (Figure 2.13). Although somewhat limiting, users found this method to be intuitive and productive. Note that in the general case, the intersection of the $(n-1)$-dimension hyperplane with the axes is an $(n-1)$-simplex in Z^n.

2.8 Retargeting Video

Video provides a more challenging medium for retargeting. On the one hand, a single multi-operator path cannot be applied to the entire video. This is because objects and camera movements inside the video most often mean that an optimal solution for one frame will not be optimal for other frames. On the other hand, retargeting each frame individually may create jittery artifacts since temporal coherency must be maintained between frames. For retargeting videos using multiple operators, we additionally require the operators to be temporally coherent (which is the case when using SC;CR;SL), and extend multi-operator image retargeting to video using key frames.

Assume we are given a video sequence $\{I_t\}_{t=1}^N$, and retargeting operators $\{O_1, \ldots, O_n\}$. Also assume we would like to change the width of the video by m pixels as previously done for images. For each key-frame $I_{t_i} \in \{I_{t_1}, I_{t_2}, \ldots, I_{t_r}\}$, where $1 \leq t_1 < t_2 < \cdots < t_r \leq N$, we find the best *regular* path $\langle k_1^{t_i} \times O_{i_1}, k_2^{t_i} \times O_{i_2}, \ldots, k_n^{t_i} \times O_{i_n} \rangle$ according to some pre-determined operator ordering. This defines a set of multi-operator sequence constraints for r time-steps that should be interpolated in-between. However, each path is actually defined by the amounts of each operator: $(k_1^{t_1}, \ldots, k_n^{t_1}), (k_1^{t_2}, \ldots, k_n^{t_2}), \ldots, (k_1^{t_r}, \ldots, k_n^{t_r})$. These sequences can be interpolated linearly on in-between frames, for each operator separately. This is because if $k_1^{t_i} + k_2^{t_i} + \ldots + k_n^{t_i} = m$, and $k_1^{t_{i+1}} + k_2^{t_{i+1}} + \ldots + k_n^{t_{i+1}} = m$ then for every $0 \leq \alpha \leq 1$ we have:

$$\begin{aligned} (\alpha \cdot k_1^{t_i} &+ (1-\alpha)k_1^{t_{i+1}}) + \\ (\alpha \cdot k_2^{t_i} &+ (1-\alpha)k_2^{t_{i+1}}) + \\ &\vdots \\ (\alpha \cdot k_n^{t_i} &+ (1-\alpha)k_n^{t_{i+1}}) = m \end{aligned}$$

Hence, once we define a method to linearly interpolate different amounts of applications for each operator, we can create a temporal coherent interpolation between multi-operator paths. For scaling, this interpolation is trivial since we just change the scale factor linearly. Seam carving also supports linear interpolation since we can remove $(\alpha \cdot k_1^{t_i} + (1-\alpha)k_1^{t_{i+1}})$ least cost seams. For cropping, we separate the amount of cropping in each key-frame to the left and right cropping (or top and bottom) and interpolate linearly between each one separately.

Such sequence interpolations can sometime insert virtual camera motion into the resulting video. For example, interpolating between cropping k columns on the left to cropping k columns on the right introduces a *panning* effect, while interpolating between different scaling levels may introduce a *zoom in/out* effect. Some example results can be found in Figure 2.20 and the supplemental video.

The key-frames that guide this process can either be sampled uniformly (say every 10 frames) or chosen using more sophisticated methods which consider the video content (see e.g. [38] for a survey on key-frame selection techniques). In addition, we supply an interface that allows the user to choose key-frames and examine *monotonic-regular* multi-operator paths for videos (Figure 2.14(b)).

2.9 Results

To validate our optimization and BDW similarity measure we compared the mean results of the user study to results obtained by our optimization on *regular* paths (Figure 2.15). The average difference between the automatic and mean user choice is 20%, which is well within the standard deviation of about 50% in the user choices. Figure 2.15 also illustrates that the visual results are comparable (all image results can be found in the supplemental material). Moreover, the mean user study result usually does not differ much in terms of the BDW score from the best score (i.e. our result). This demonstrates the effectiveness of the BDW measure itself.

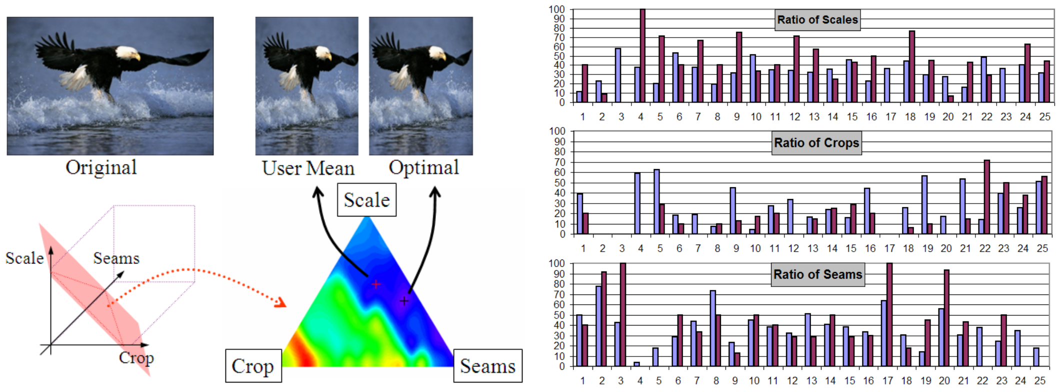


Figure 2.15: Using *regular* paths we are limited to searching on a plane in resizing space (left). We find the optimal multi-operator resizing sequence having the minimum BDW cost (0.241 in this case). The distance throughout the search space is colored from blue (small distance) to red (large distance) and was interpolated for visualization purposes. We compare our results to the results and score of the mean of the user study (where BDW = 0.355). On the right we show a summary of the comparison of the ratios of all results. Blue is the mean of the user study and Red is our results using optimization with BDW. The average difference is around 20%, well within the standard deviation of the user study which is 50%. More image results can be found in Appendix 2.B.

Our multi-operator framework supports various scenarios for finding the optimal combination for retargeting. We illustrate this by showing results of a number of cases where we change the set of operators and also the image similarity measure used. Figure 2.16 illus-

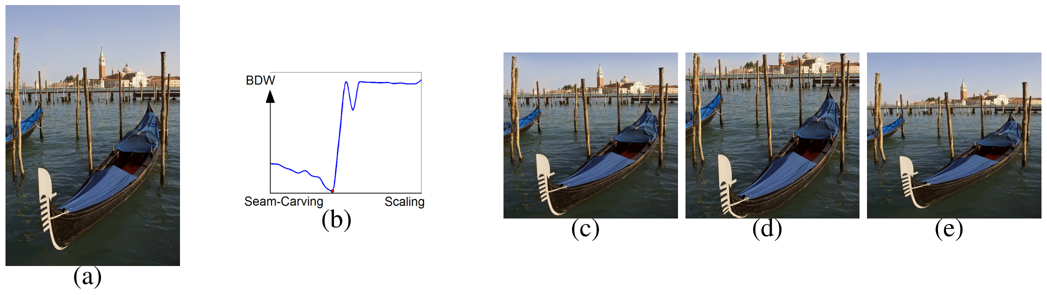


Figure 2.16: We find the optimal *regular* path by finding the minimal BDW score (the red dot in (b)) for a combination of two operators (seam carving and scaling in this case) to retarget an image (a). Compare the resulting image (c) to using just seam carving (d) or just scaling (e).

trates an example of optimally combining seam carving and scaling in a *regular* sequence. We find the best transition point between applying seam carving and scaling by measuring the BDW of the results. In Figure 2.17 we show results of computing the optimal *mixed* sequences using two operators (seam carving and scaling) by dynamic programming using the BDW score. The teaser figure (Figure 2.1) shows the result of finding the optimal *mixed* sequence for changing the width of an image consisting of three operators (scaling, cropping and seam carving), subject to the BDW image similarity measure. In Figure 2.19 we find the best *mixed* path using the bidirectional similarity measure [33] and four image retargeting operators (horizontal and vertical scaling and seam carving). As can be seen, the horizontal dimension is retargeted mainly with seam carving while the vertical dimension is mainly scaled. Figure 2.18 shows a comparison between several retargeting methods and our multi-operator results.

In some cases, the optimal multi-op result might reduce to a single operator (e.g. scaling). However, we should note that this reduction is achieved automatically by the algorithm in an *informed* manner. This is exactly the purpose of the suggested system. A good result is not necessarily one that utilizes all available operators, but rather one that achieves higher similarity of the required size. In particular, by attempting to use cropping, scaling and seam-carving, the algorithm chose the scaling approach for more structured media, as seam carving tends to insert artifacts in such cases, and cropping might remove too much important information. In fact, we deliberately used images that are difficult cases for non-uniform operators (such as seam carving) to test if our method can recognize this automatically.

In Figure 2.20 we show an example of a key-frame from a video that demonstrates why multi-operator retargeting provides the flexibility to achieve better results than a single opera-

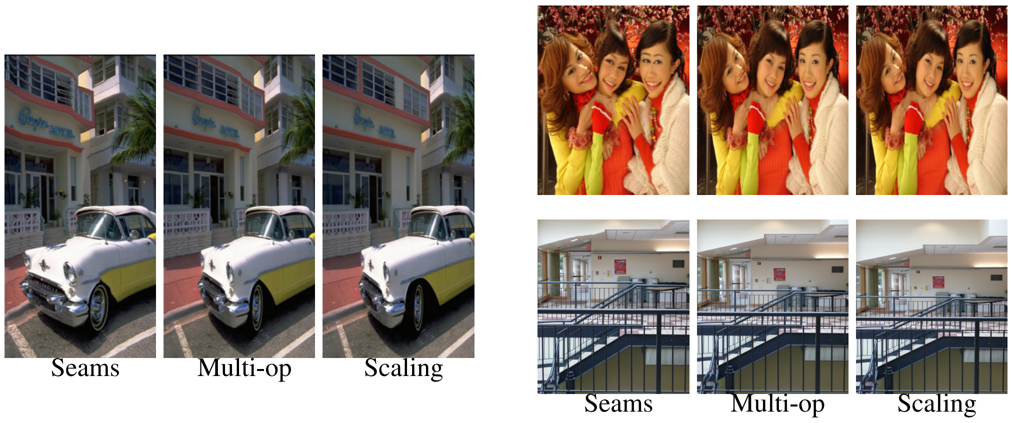


Figure 2.17: A comparison between seam carving (left), Multi-operator (center) and scaling (right). The multi-operator algorithm uses the BDW image similarity measure and finds the best *mixed* path using two image retargeting operators (seam carving and scaling).

tor in video. Lastly, our framework also enables utilizing a simple user interface (Figure 2.12) for combining *regular* multi-operator sequences in an intuitive manner. In the accompanying video we also show several interactive sessions for image retargeting, and more retargeting results for images and video. Taken together, these results show that our multi-operator algorithm can combine multiple operators together using various similarity measures (e.g. BDW or BDS), various paths (either *regular* or *mixed*) and various operators (horizontal and vertical, seam carving, scaling and cropping).

All results were created either on a 1.8 GHz dual core laptop with 2GB memory or on a 2.2 GHz dual core desktop with 4GB memory. Several processing time statistics for computing BDW and the optimizations are detailed in Table 2.1 based on our unoptimized implementation. As for the interactive interfaces, in most cases (as seen in the video) the interaction is performed in real time. There are waiting periods, for instance, when there is a switch in the direction of size change between height and width. This is because the change triggers a recalculation of the seams which may take a few seconds. For video, we store just the interpolation values of the operators for all frames. Once seam carving has been pre-computed on the video, the video playback is instant. Moreover, by constraining the ratio between the amounts of applying each operator to remain constant during resize, we can interactively change the video size as well.

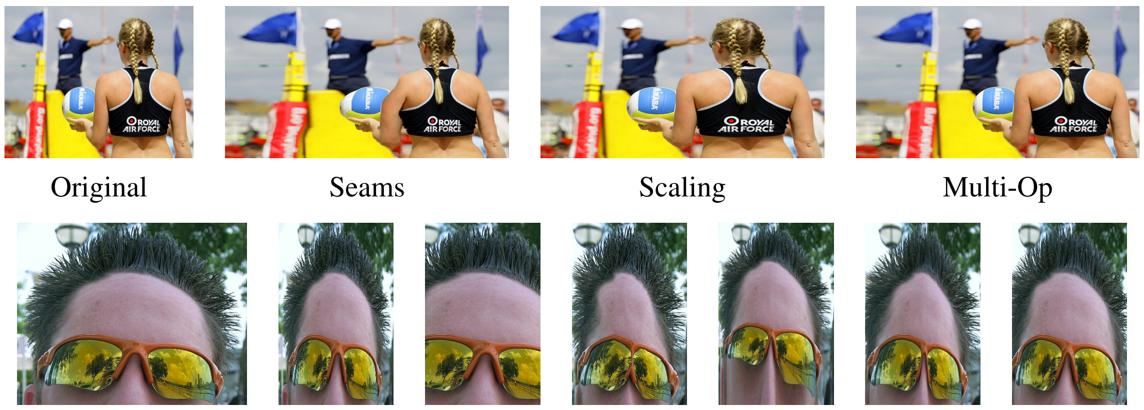


Figure 2.18: Comparison of retargeting results (expansion and reduction) using various image retargeting methods. These examples illustrate cases where using optimized multi-operator retargeting combining seam caving, cropping and scaling achieves better results. Seams is using seam carving from [Rubinstein *et al.* 2008], Warp(1) and Warp(2) are non-homogeneous scaling from [Wang *et al.* 2008] and [Wolf *et al.* 2007] respectively.

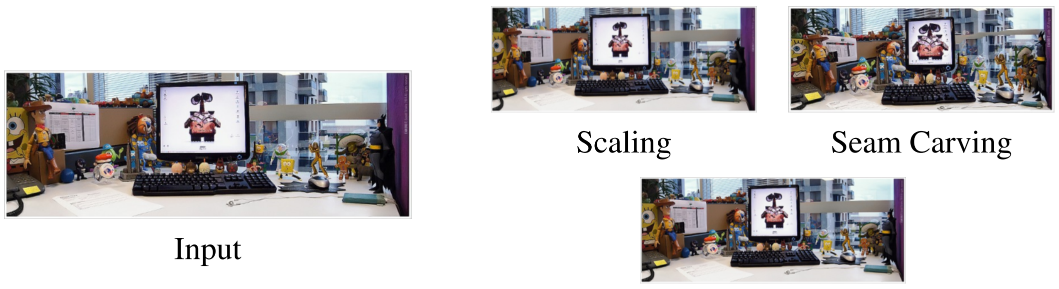


Figure 2.19: Result of 2D retargeting. In this case we find the optimal *mixed* path using the bidirectional similarity measure and a combination of four retargeting operators (horizontal and vertical seam carving and scaling). The multi-operator result finds the best result by mainly applying seam carving in the horizontal dimension (compare the size of the monitor in the different methods) and scaling in the vertical dimension (look at the bottom of the desk and the face of Woody on the left). For comparison, we show the result of applying a uniform 2D scaling, or seam carving (running horizontal seam carving first, followed by vertical seam carving).

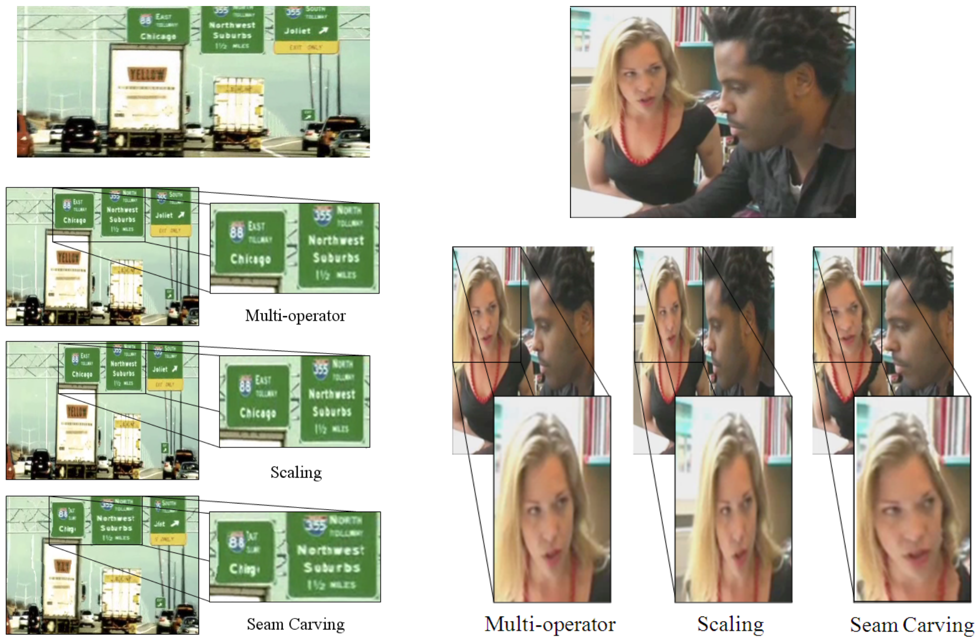


Figure 2.20: An example of a video key-frames where multi-operator retargeting achieves better results than scaling or seam carving.

2.10 Limitations

Figure 2.15 and the supplemental material (Section 2.B) show that our automatic results do not always agree with the users' preference. Still, one must remember that the users also did not agree on the "correct" result and we have used the barycenter of all users choices as our ground truth.

As mentioned in Section 2.6, the optimization is still exponential in the number of operators. Devising a more efficient algorithm is an interesting challenge, and we have yet been able to improve on this bound. The overall complexity of the algorithm is measured by the number of entries in the dynamic programming table that we need to fill. In each iteration we need to apply one operator and compute the bidirectional warping, which is the major bottleneck as can be seen in Table 2.1. It currently takes about 3 seconds to compute one bidirectional warp, and we have discussed relevant optimization methods in Section 2.5.4. To the best of our knowledge, only one other metric, namely bidirectional similarity [33], was proposed in the context of media retargeting. Compared to their method, our measure is asymptotically faster. In context of the optimization, additional means might be considered.

Resolution	Running time (sec)	
	Large overlap	No overlap
Pixels	-	7.5
4×4 Patches	5	3.1
8×8 Patches	3.2	1.5
16×16 Patches	2.4	1.1
32×32 Patches	0.9	0.4

Optimization	Running time (min)
2-regular	2
2-mixed	10
3-regular	10
3-mixed	15
4-mixed	20

Table 2.1: Average processing times for BDW (left, in seconds) and several optimization schemes (right, in minutes) for the examples used in this paper. For BDW, times are shown for patches taken in 1-pixel steps (large overlap) and with no overlap. Larger overlap can result in better alignment (smaller distance).

For example, we can employ early termination of the BDW evaluation in case the images are too different. Moreover, our experiments show that the resizing space can be sampled at coarser resolution (e.g 20-pixel steps) while still producing good results, which significantly reduces running times as well. The process can then be repeated with a finer resolution grid around the coarse solution in a standard multiresolution fashion.

Lastly, it is clear that the suggested algorithm is only as good as the operators used. Limitations imposed by the specific methods (cropping, scaling, seam-carving) will also be carried over to our solution. Towards this end we found that the combination of several operators could alleviate some limitations of specific ones.

2.11 Conclusions and Future Work

We proposed an algorithm for combining multiple retargeting operators. We first defined the *resizing space* as a conceptual multi-dimensional space combining several resizing operators, and showed how a path in this space defines a sequence of operations to retarget media. Then, we presented the multi-operator algorithm that relies on dynamic programming to find the optimal path in resizing space, given a global objective function that measures the similarity between the source and target images.

The resizing space model was introduced both for modeling the problem, but- not less importantly- to understand its complexity. Using this model we were able to show that the problem of combining multiple operators for retargeting is exponential in the amount of size change, and introduced an assumption (and understood its geometric implication in this space) that reduces the problem to a polynomial one, given fixed number of operators. Thus, the running time is largely dependent on the complexity of the underlying similarity measure.

For the global objective function we proposed Bi-Directional Warping (BDW) which is based on Dynamic Time Warping (DTW). Remarkably, all levels of our algorithm benefit from dynamic programming. It is used to compute Seam Carving, used to compute a-symmetric alignment between two signals that forms the basis to our BDW image similarity measure and finally, it is the basis of our algorithm to find the optimal multi-operator path.

We tested our approach on a large number of images and videos. Note that many of those images were difficult cases for previous single retargeting operators. We also validated our results by comparing them with ground truth data, collected in the user study. In addition, we described a simple and intuitive user interface to interactively explore the resizing space and achieve high quality results.

The BDW measure we presented is best suited to changes applied to the image in one direction. However, BDW can be extended to match 2D modifications in some cases by recursively applying asymmetric-DTW on the rows (or columns) of the image (Appendix 2.A). In the future we plan to investigate other applications for BDW, as well as possible extensions to 2D and 3D. For given source and retargeted video volumes, the BDW will result in a mapping which is order-preserving both spatially and temporally, thus giving higher similarity scores to results which preserve the original video structure and time-line. We also intend to combine other types of operators in our multi-operator framework.

Finally, there are further ways to utilize the user data. For example, it should be interesting to verify more carefully whether the BDW metric agrees with the users' preference, or get direct user feedback on the automatically generated results. Another interesting direction is to search for correlation between the preferred resizing paths and the media content, which will enable a "retargeting-by-example" framework.

Appendix 2.A Two-Dimensional Asymmetric-DTW

For retargeting operators which work along a single dimension such as cropping, scaling, Seam-Carving [3] and non-homogeneous warping [48], applying our Asymmetric-DTW on the rows (or columns) of the image independently is sufficient to correctly estimate the underlying transformation. However, we should note that the framework presented above supports operators that resize images in one direction (either horizontally or vertically), yet, the operators still have the freedom to distribute changes along the two dimensions. An example of such operator is the recently proposed Scale-and-Stretch warping [46]. This method extends its predecessor [48] by diverting the distortions to both spatial dimensions, regardless of the resizing direction.

The two-dimensional version of DTW is commonly known as *Dynamic Planar Warping* (DPW), and its definition is similar to its one-dimensional counterpart. Unfortunately, this problem was shown to be NP-complete [13], and several approximation methods have been proposed [39].

A heuristic extension of our one-dimensional solution is to recursively apply Asymmetric-DTW on the rows of the image, taking the cost of matching row i of S to row j of T as the

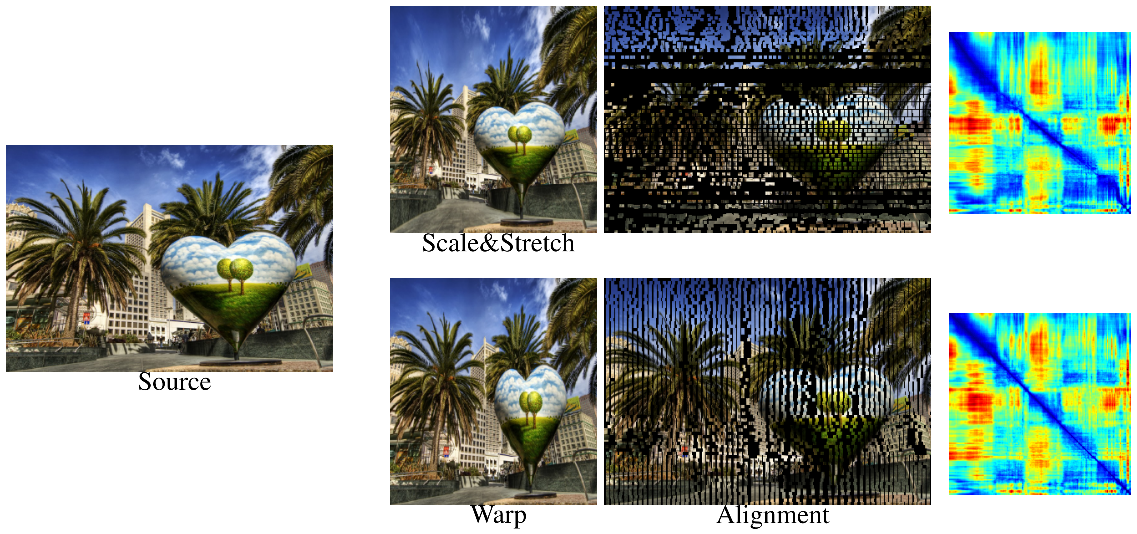


Figure 2.21: Examples of 2D A-DTW. Alignments of the Scale-and-Stretch result (top) and non-homogeneous warping result (bottom) to the original image (left) using 4×4 patches. On the right, the pairwise 1D A-DTW row distances are shown for each result, colored from blue (small distance) to red (large distance).

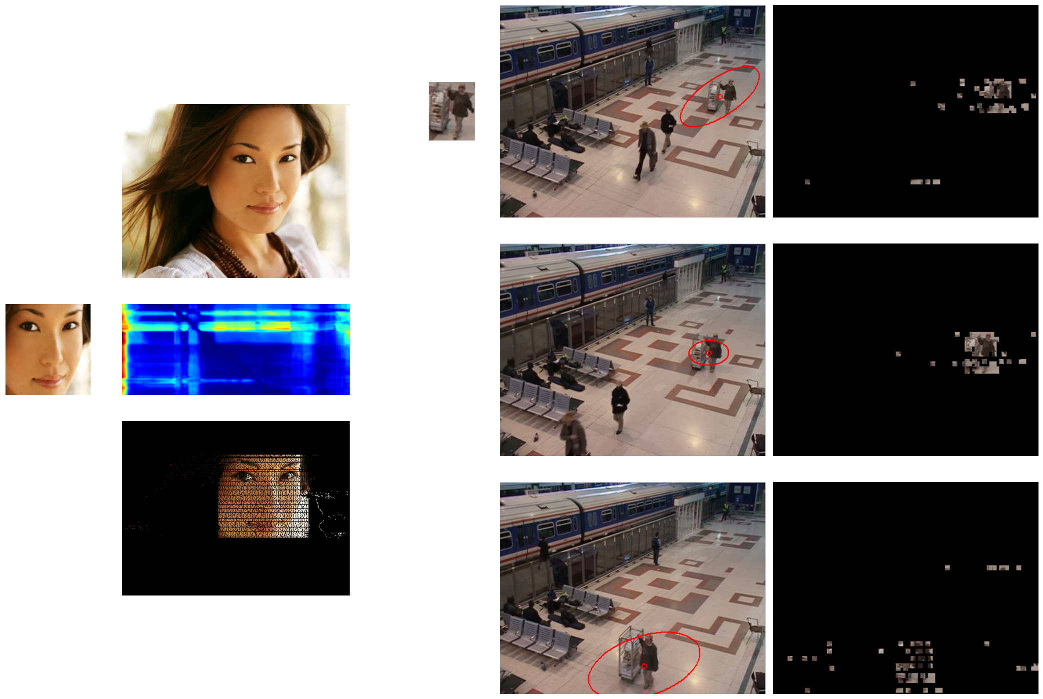


Figure 2.22: 2D A-DTW can be used to detect reference objects in images, even when they undergo some deformations. On the left, an excerpt from the woman face image (top), was scaled-down horizontally and vertically at different rates (left), and then aligned to the original image. The A-DTW distance between the rows of the reference and source, and the optimal alignment result are shown in the middle and bottom respectively. On the right, a reference object (upper left) was extracted from a frame of a surveillance video taken from the PETS'2006 database [1], and aligned to each frame of the sequence. By considering the mean and covariance (middle column, plotted in red on the respective frames) of the aligned patches (right column), we are able to detect the reference object in the video, although it exhibits different deformations and scales.

A-DTW distance between them, thus dividing the two-dimensional problem to a collection of one-dimensional ones. This results in a two-dimensional order-preserving mapping between the two images that is optimal under this rigid row-to-row alignment. Although this method will not estimate correctly all possible transformations, we found it to produce good approximation for assessing image similarity. Some results of the two-dimensional A-DTW are shown in Figure 2.21. Notice that as the Scale-and-Stretch operator deforms the image in both directions, the optimal alignment inserts gaps in some rows, while for non-homogeneous warping, which works along the resized dimension, every row in the retargeted image is matched to its corresponding row in the source (still solving for the best 2D alignment). As can be seen in the row distance maps, the A-DTW distance between rows in the sky region is relatively small as they share similar structure. This is shown as a blurred low-distance area in the upper-left region of the maps.

The running time of this algorithm is $O(h^2w^2)$ using naive implementation, but can be further optimized using the techniques mentioned in Section 2.5.4. Specifically, there is no reason to compare each row to all other rows. Instead, we can compare each row to its neighboring rows in a bounded distance, which significantly reduces the search space. Other applications for this method might include object detection and tracking, as shown in Figure 2.22. We are currently investigating these approaches.

Appendix 2.B Supplemental Results

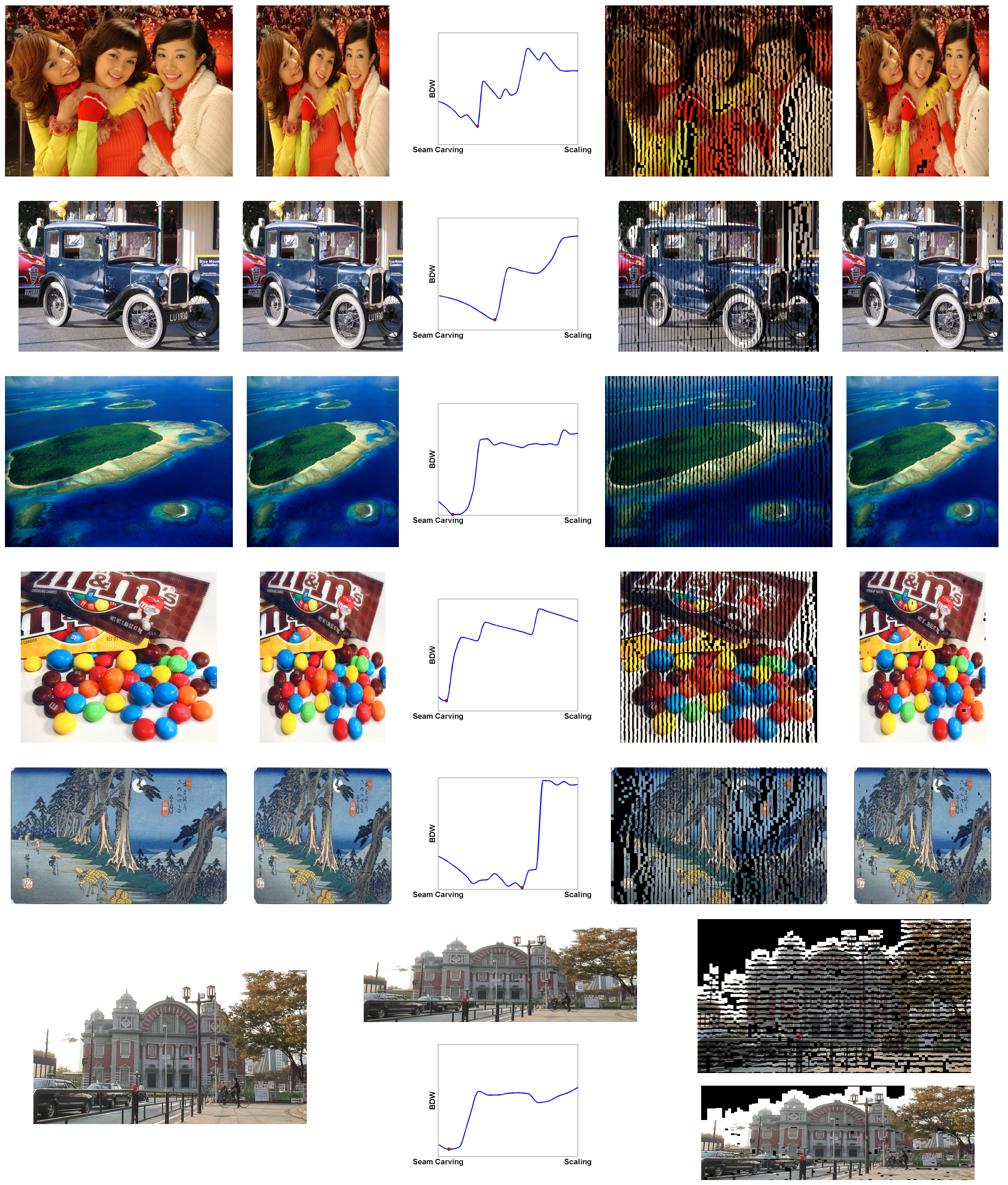


Figure 2.23: Optimal 2-operator regular paths using seam-carving and scaling. From left to right: the source image; the retargeted image; a plot of the BDW distance versus different transition points between seam-carving and scaling; the BDW alignment of the retargeted image to the source image; the BDW alignment of the source image to the retargeted image. For the latter visualizations, in case several pixels of one image are mapped to the same pixel in the other - their average value is taken.

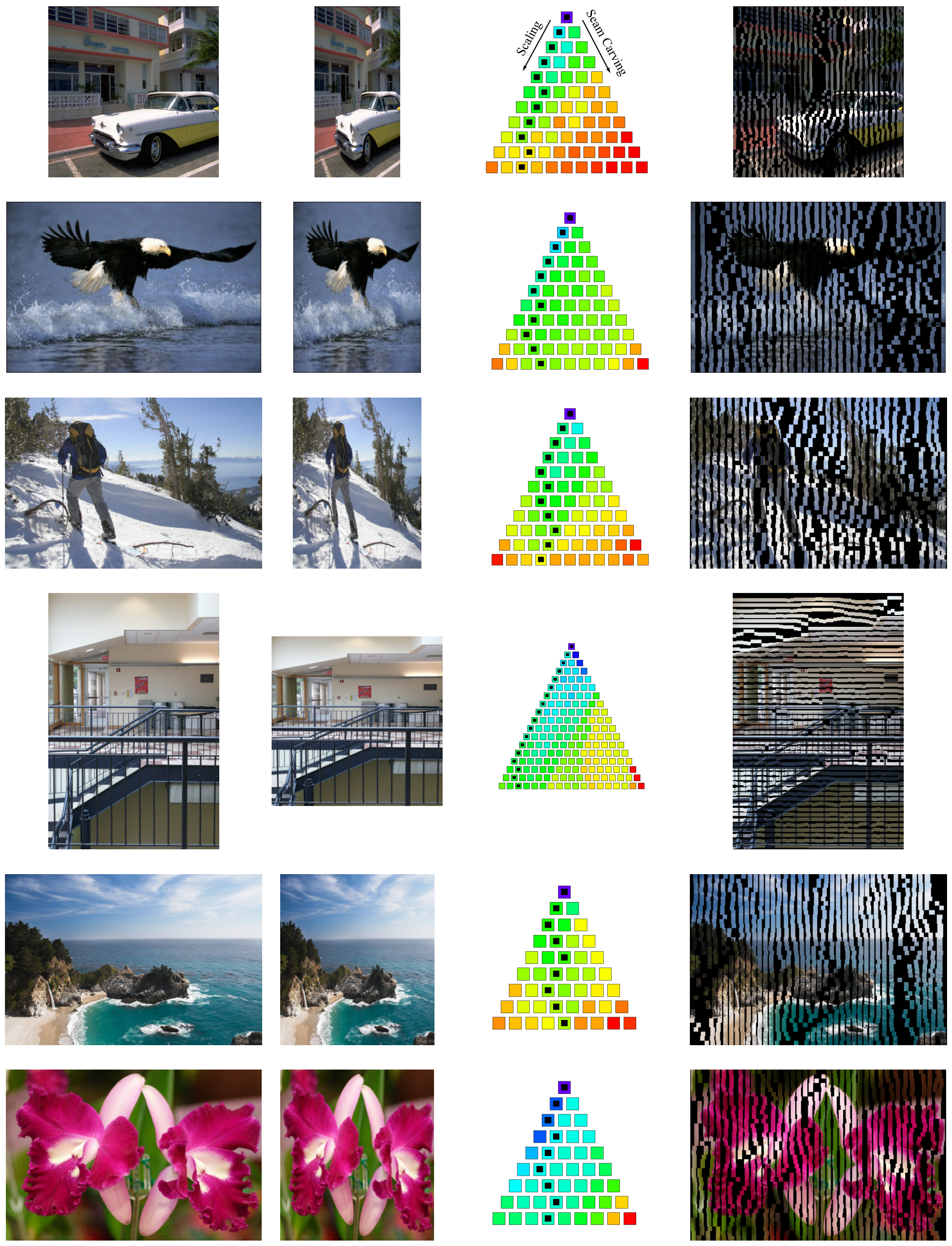


Figure 2.24: Optimal 2-operator mixed paths using the seam-carving and scaling operators, and BDW similarity. From left to right: the source image; the retargeted image; the dynamic programming table colored by the BDW distance (relative per image); the BDW alignment of the retargeted image to the source image. The table size corresponds to the amount of change and the rate at which the search space is sampled.

Comparison to the User Study:

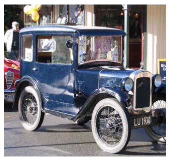
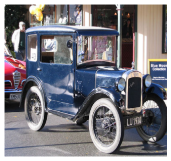
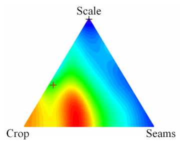
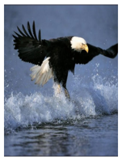
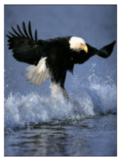
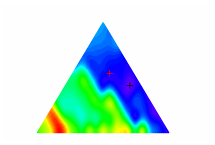
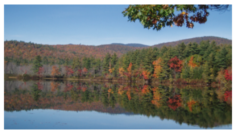
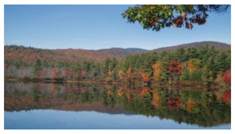
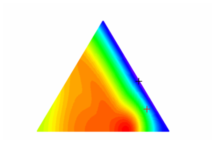
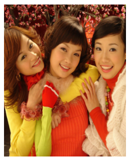
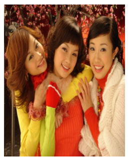
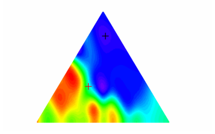
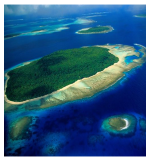
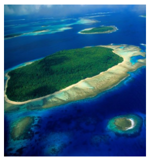
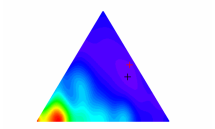
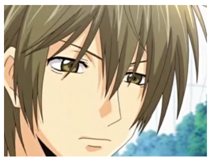
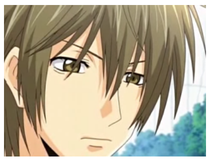
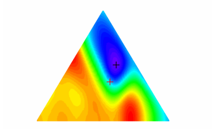
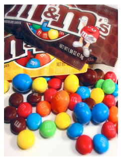
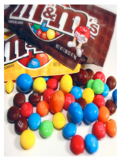
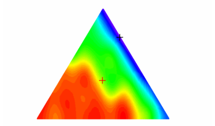
Image	Mean User Study	Optimal 3-operator regular path	Optimization search space
car			
eagle			
foliage			
girls			
islands			
manga			
mnm			

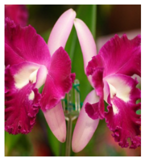
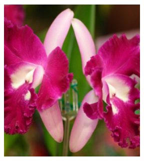
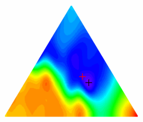
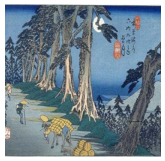
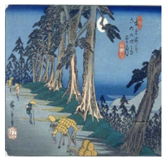
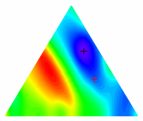
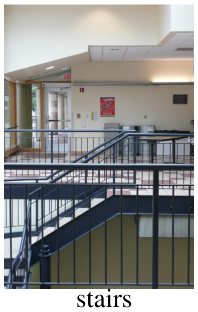
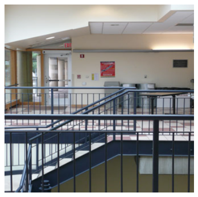
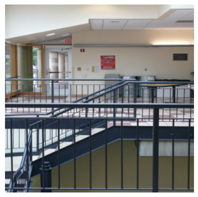
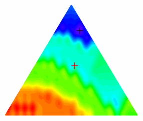
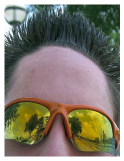
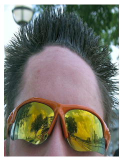
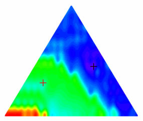
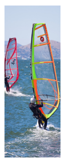
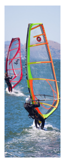
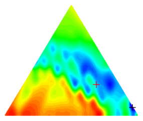
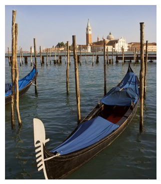
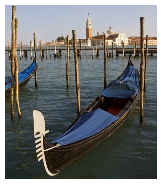
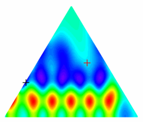
Image	Mean User Study	Optimal 3-operator regular path	Optimization search space
orchid			
mochizuki			
 stairs			
sunglasses			
surfers			
venice			

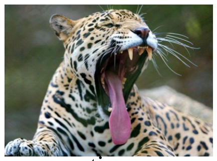
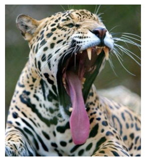
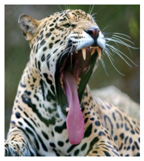
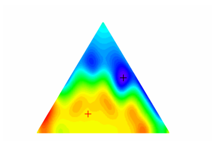
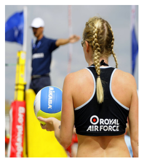
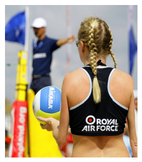
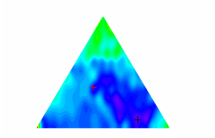
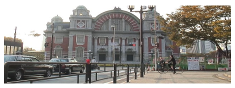
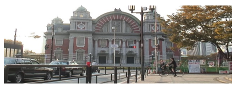
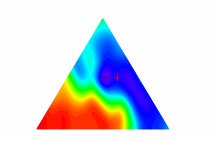
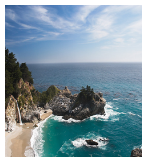
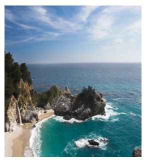
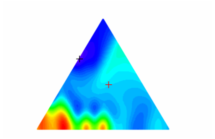
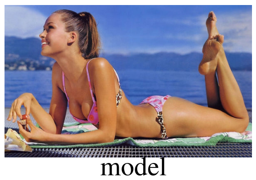
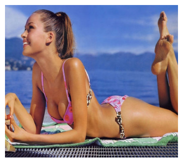
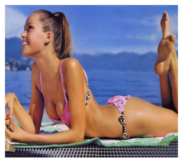
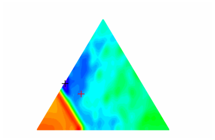
Image	Mean User Study	Optimal 3-operator regular path	Optimization search space
 tiger			
volleyball			
osaka			
waterfall			
 model			

Figure 2.25: Comparison between automatic (3-operator regular paths) and user study results. From left to right: the image used in the study; the mean user result, taken as the weighted center of mass of the user data; the optimal retargeted image; a visualization of the search space as discussed in the paper and shown in the first (car) figure. The automatic and user results are marked on the search space with black and red markings respectively.

Chapter 3

Conclusion

This thesis investigates discrete solutions to the media resizing problem. Our experiments are based on the Seam-Carving operator [3] recently proposed for content-aware image retargeting. We first adapted this operator to the video domain using a reduction to a minimal-cut graph problem, and extended this technique to videos by carving low-energy surfaces from video cubes. We also suggested a novel energy criteria that significantly improves the results for both images and videos. We showed results for various video editing operations that are considered state-of-the-art in this field, or successfully compete with (and sometimes outperform) existing content-aware video retargeting techniques.

Additionally, we suggested the first *meta-algorithm* for media retargeting. It is guaranteed to find the optimal combination of existing, as well as future, media retargeting operators in polynomial time under some assumptions. We also defined a new objective function: the bidirectional warping. We demonstrated the effectiveness of this measure and approach for both quantifying the quality of retargeted results, as well as combining multiple resizing operators. We are also the first, in the recent crop of media retargeting works, to conduct a user study to investigate the implication of media resizing on user perception, and compare automatically generated results against the collected ground truth.

Our experiments show that there is no clear superiority of discrete operators over continuous ones, or vice versa. We also showed that there is no single method that is strictly better than any other for all images and target sizes. Instead, each resizing operator has its strengths and weaknesses, and we have studied and enumerated some of them throughout this thesis. This emphasizes the need for a quality measure that identifies when one method performs better than another, and supports the suggested meta approach.

Our work can be extended in numerous ways and we have touched upon specific directions in the respective conclusion sections in each chapter. Content-aware media retargeting is a relatively nascent field of research and there is much room for further study and improvement. Recent non-linear deformation methods such as the Scale-and-Stretch warping and Seam-Carving took over previous rigid methods that were bound to choosing an optimal cropping window. These new methods show very nice results for both images and video, but are based, by definition, on distorting the media. Specifically, they were shown to create serious artifacts and awry results when homogenous regions are scarce, and there are cases where the result might not agree with what the users perceive as satisfactory resizing.

The problem with these methods is that they are content-aware but not *user-aware*. In particular, all content-aware operators are driven by well-studied perceptual models that have been investigated for many years and aim to describe what the human vision discerns as important, or salient, in a media. However, to our knowledge, no studies were conducted to investigate what is the preferred way to deform the media in case some of its content has to be misplaced. What bothers the users more- distortion, or loss of information? loss of proportions, or broken structures? is there some correlation between the type of content and the preferred operator? Such studies are needed to better understand the objectives of media retargeting, which will in turn allow to devise better suited resizing algorithms. Understanding these tradeoffs will probably stand as major goal in forthcoming research in this field.

References

[1] Pets'06: Performance evaluation of tracking and surveillance. *The Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2006.

[2] Aseem Agarwala, Ke Colin Zheng, Chris Pal, Maneesh Agrawala, Michael Cohen, Brian Curless, David Salesin, and Richard Szeliski. Panoramic video textures. *ACM Trans. Graph.*, 24(3):821–827, 2005.

[3] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. *ACM Trans. Graph.*, 26(3):10, 2007.

[4] Oren Boiman and Michal Irani. Detecting irregularities in images and in video. *International Journal of Computer Vision*, 74(1):17–31, 2007.

[5] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[6] Yuri Boykov and Olga Veksler. Graph cuts in vision and graphics: Theories and applications. In *Handbook of Mathematical Models in Computer Vision*, pages 79–96. Springer.

[7] Billy Chen and Pradeep Sen. Video carving. In *Short Papers Proceedings of Eurographics*, 2008.

[8] L.Q. Chen, X. Xie, X. Fan, W.Y. Ma, H.J. Zhang, and H.Q. Zhou. A visual attention model for adapting images on small displays. *Multimedia Systems*, 9(4):353–364, 2003.

[9] Xin Fan, Xing Xie, He-Qin Zhou, and Wei-Ying Ma. Looking into video frames on small displays. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 247–250. ACM, 2003.

[10] R. Gal, O. Sorkine, and D. Cohen-Or. Feature-aware texturing. In *Eurographics Symposium on Rendering*, pages 297–303, 2006.

[11] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203.

[12] Mohamed Hussein, Amitabh Varshney, and Larry Davis. On implementing graph cuts on cuda. 2007.

[13] Daniel Keysers and Walter Unger. Elastic image matching is np-complete. *Pattern Recogn. Lett.*, 24(1-3):445–453, 2003.

[14] Pushmeet Kohli and Philip H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 29(12):2079–2088, 2007.

[15] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22(3):277–286, 2003.

[16] Yin Li, Jian Sun, and Heung-Yeung Shum. Video object cut and paste. *ACM Trans. Graph.*, 24(3):595–600, 2005.

[17] F. Liu and M. Gleicher. Automatic Image Retargeting with Fisheye-View Warping. In *ACM UIST*, pages 153–162, 2005.

[18] Feng Liu and Michael Gleicher. Video retargeting: automating pan and scan. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 241–250. ACM, 2006.

[19] H. Liu, X. Xie, W.Y. Ma, and H.J. Zhang. Automatic browsing of large pictures on mobile devices. *Proceedings of the 11tn ACM international conf. on Multimedia*, pages 148–155, 2003.

[20] Herve Lombaert, Yiyong Sun, Leo Grady, and Chenyang Xu. A multilevel banded graph cuts method for fast image segmentation. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)*, volume 1, pages 259–265, 2005.

[21] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pages 674–679, 1981.

[22] Yael Pritch, Alex Rav-Acha, and Shmuel Peleg. Non-chronological video synopsis and indexing. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, (to appear), 2008.

[23] Alex Rav-Acha, Yael Pritch, Dani Lischinski, and Shmuel Peleg. Dynamosaicing: Mosaicing of dynamic scenes. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 29(10):1789–1801, 2007.

[24] Alex Rav-Acha, Yael Pritch, and Shmuel Peleg. Making a long video short: Dynamic video synopsis. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pages 435–441. IEEE Computer Society, 2006.

[25] Michael Rubinstein, Ariel Shamir, and Shai Avidan. Improved seam carving for video retargeting. *ACM Trans. Graph.*, 27(3), 2008.

[26] Michael Rubinstein, Ariel Shamir, and Shai Avidan. Multi-operator media retargeting. *ACM Trans. Graph. (to appear)*, 28(3), 2009.

[27] Hiroaki Sakoe. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[28] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5):561–580, 2007.

[29] A. Santella, M. Agrawala, D. DeCarlo, D. Salesin, and M. Cohen. Gaze-based interaction for semi-automatic photo cropping. In *ACM Human Factors in Computing Systems (CHI)*, pages 771–780, 2006.

[30] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498. ACM Press/Addison-Wesley Publishing Co., 2000.

[31] V. Setlur, S. Takagi, Ramesh. Raskar, M. Gleicher, and B. Gooch. Automatic image retargeting. In *In the Mobile and Ubiquitous Multimedia (MUM)*. ACM Press, 2005.

[32] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '94)*, pages 593–600, 1994.

[33] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[34] Duncan M'Laren Young Sommerville. *An Introduction to the Geometry of n Dimensions*. Dover, New York, 1958.

[35] Bongwon Suh, Haibin Ling, Benjamin B. Bederson, and David W. Jacobs. Automatic thumbnail cropping and its effectiveness. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 95–104. ACM Press, 2003.

[36] Martin Szummer and Rosalind W. Picard. Temporal texture modeling. In *IEEE Intl. Conf. Image Processing*, volume 3, pages 823–826, 1996.

[37] Chenjun Tao, Jiaya Jia, and Hanqiu Sun. Active window oriented dynamic video retargeting. In *Proceedings of the Workshop on Dynamical Vision, ICCV 2007*, 2007.

[38] Ba Tu Truong and Svetha Venkatesh. Video abstraction: A systematic review and classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 3(1):3, 2007.

[39] Seiichi Uchida and Hiroaki Sakoe. A monotonic and continuous two-dimensional warping based on dynamic programming. In *ICPR '98: Proceedings of the 14th International Conference on Pattern Recognition-Volume 1*, page 521, 1998.

[40] Vibhav Vineeth and P J Narayanan. Cuda cuts: Fast graph cuts on the gpu. 2008.

[41] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, 2004.

[42] Hongcheng Wang, Ramesh Raskar, and Narendra Ahuja. Seamless video editing. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04)*, volume 3, pages 858–861. IEEE Computer Society, 2004.

[43] Jue Wang, Pravin Bhat, R. Alex Colburn, Maneesh Agrawala, and Michael F. Cohen. Interactive video cutout. *ACM Trans. Graph.*, 24(3):585–594, 2005.

[44] Jue Wang, Yingqing Xu, Heung-Yeung Shum, and Michael F. Cohen. Video tooning. *ACM Trans. Graph.*, 23(3):574–583, 2004.

[45] Jun Wang, Marcel Reinders, Reginald Lagendijk, Jasper Lindenberg, and Mohan Kankanhalli. Video content presentation on tiny devices. In *IEEE International Conference on Multimedia and Expo (ICME)*, volume 3, pages 1711–1714, 2004.

[46] Yu-Shuen Wang, Chiew-Lan Tai, Olga Sorkine, and Tong-Yee Lee. Optimized scale-and-stretch for image resizing. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH ASIA*, 27(5), 2008.

[47] Li-Yi Wei, Jianwei Han, Kun Zhou, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Inverse texture synthesis. *ACM Trans. Graph.*, 27(3):1–9, 2008.

[48] Lior Wolf, Moshe Guttmann, and Daniel Cohen-Or. Non-homogeneous content-driven video-retargeting. In *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV '07)*, pages 1–6, 2007.

[49] Ramin Zabih, Justin Miller, and Kevin Mai. A feature-based algorithm for detecting and classifying scene breaks. In *MULTIMEDIA '95: Proceedings of the third ACM international conference on Multimedia*, pages 189–200, 1995.

Media Acknowledgments

We wish to thank the proprietaries who have contributed to this work by granting their permission, either directly or indirectly, to use their media for research purposes.

Video Retargeting using Seam-Carving

We thank Angelo Garcia for narrating our video. We thank Mammoth HD library (mammothhd.com) for allowing us to use their royalty free demo reel (road ski, water ski, kayak, fish, bicycle, nature). We thank Wolf et al. for letting us use their video samples (basketball, football) and saliency maps (football). We thank the members of the following communities for publicly sharing their media: youtube (www.youtube.com): Nmbr5 (golf). blip.tv (http://blip.tv): Detroit Free Press (cheerleaders), aaron (cheerleader shaky camera), mindcaster (ape animation), Mike Krumlauf (highway), cuecast (interview). stage6 (stage6.com): dancers, Osaka hall image. flickr (www.flickr.com): Ben McLeod (bench), Thomas Hawk (rain). Other images were borrowed from Avidan and Shamir (waterfall, car, vase, umbrella, matches, snow). The footage from RATATOUILLE is courtesy of Disney/Pixar. The SIGGRAPH evolve sample was taken from the ACM SIGGRAPH 2008 demo video.

Multi-operator Media Retargeting

We thank the anonymous SIGGRAPH reviewers for their comments. We thank Maya Yaniv for narrating our video. We thank the flickr members who have kindly made their media available for research purposes via the creative commons license: Ben Harris-Roxas (fishing), danorbit (desk), david.bunting (volleyball), g_magnan (italy), Greg Gladman (church, wheels), i am indisposed (snow), iboy daniel (mnm), Pandiyan (pond), romainguy (surfers), thomas23 (glasses), van swearingen (orchid), etrusia_uk (Bodiam castle). We also thank the users of publicdomainpictures.net and morguefile.com who have shared their images

through public domain (tiger, eagle, stairs, islands). The Taj Mahal image is courtesy of ictopon2009.uwo.ca. The San Francisco heart image and results were borrowed from [46]. The bicycle, Buddha, car, malibu, foliage, face, mochizuki, venice and waterfall images are borrowed from [3]. The osaka image and highway video are taken from [25]. The birds video sequence is a snipped from “for the birds”, courtesy of Disney/Pixar.