中華民國國立成功大學

資 訊 工 程 學 系 博 士 班

博 士 論 文

基於內容為主之圖片與影像縮放最佳化研究

Content Aware Image and Video Resizing

研 究 生 ： 王昱舜

指 導 教 授 ： 李同益博士

中 華 民 國 九 十 九 年 一 月

國立成功大學

博士論文

基於內容為主之圖片與影像縮放最佳化研究

Content Aware Image and Video Resizing

研究生：王昱舜

本論文業經審查及口試合格特此證明

論文考試委員：

李同益

楊熙年 王宗銘

李蔡彥 [illegible]

[illegible] [illegible]

歐陽明 莊永裕

指導教授：李同益

系(所)主管：[illegible]

中　華　民　國　99　年　1　月　28　日

國立成功大學

Content Aware Image and Video Resizing

By

Yu-Shuen Wang

A thesis submitted to the graduate division in partial
Fulfillment of the requirements for the degree of PhD of
Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan, R.O.C.
January 2010

Approved by

[illegible] [illegible]

[illegible] [illegible]

[illegible] [illegible]

Chung-Ming Wang [illegible]

[illegible]

Advisor: [illegible]

Chairman: [illegible]

基於內容為主之圖片與影像縮放最佳化研究

作者： 王昱舜　　指導教授： 李同益博士

中華民國國立成功大學資訊工程學系博士班

由於不同螢幕的解析度和長寬比例上的差異，如傳統電視、數位電視和手持電話，近年來圖片與影片縮放的研究變得愈來愈熱門。傳統的做法是內插圖片中的內容來改變解析度。然而這種將全部內容做單一縮放的方法，很容易改變圖片中的人或是一些具有結構性物體的比例，產生縮放上的失真。於是，有一些學者利用裁切的方法，移除圖片週圍一些比較不重要的區域來改變圖片的大小。他們利用一些自動的方法偵測每一個區域的重要性，然後尋找一個可以包含最多資訊的裁切視窗，將內容的損失降至最低。

然而這種裁切的方法是有其限制的，如果重要的物體都緊鄰著圖片的週圍，那不論怎麼裁切都會將這些重要的物體移除。因此，基於內容為主的圖片縮放技術接著被提出來了。這些方法利用移除或擠壓圖片中比較不重要的內容，來隱藏不等比例縮放所產生的瑕疵。例如壓扁天空中的雲就不會產生視覺上的不適。基於這項論點，我們的目的是將圖片中的每個區域作不等比例的變形，達到減少瑕疵的目的。與前人的方法相比，我們對每一個區域的大小與長寬比例作最佳化，使得縮放後的圖片更加自然。

此外，我們還將這樣的技術應用到影片上。除了維持住重要物體的比例外，我們還必需維持相同的物體在不同的時間點的縮放是一致的。否則物體會忽大忽小，產生另一種時間上的瑕疵。為了達成這個目的，我們偵測物的運動軌跡，找出每個物體在不同時間點的關連性，然後限制這些套用在相關物體的變形是一致的。最後，我們利用最佳化技術，在時間和空間上尋找一個平衡點，將視覺上的失真降至最低。

關鍵字：不等比例縮放、裁切、最佳化技術

Content Aware Image and Video Resizing

Student: Yu-Shuen Wang Advisor: Dr. Tong-Yee Lee

Department of Computer Science and Information Engineering

National Cheng Kung University, Tainan, Taiwan, R.O.C.

Abstract

Research on automatic resizing of images and videos is becoming ever more important with the proliferation of display units, such as television, notebooks, PDAs and cell phones, which all come in different aspect ratios and resolutions. To achieve full presentation of images and videos, we introduce a content aware technique which considers the interior pixels while resizing the images and videos. Specifically, we represent an image/frame with a grid mesh and then warp the mesh based on the saliency measure. Unlike the previous methods, which strove to preserve the prominent objects untouched, our method allows them to be scaled uniformly, enabling the distortion propagation in multiple directions. In addition to the resizing of static images, we extend our resizing technique to videos. The most important issue on this extension is the temporal coherence since the interior contents keep changing when the video is played. Due to the camera and object motions, simply preserving consistent resizing of temporally adjacent pixels cannot achieve temporal coherence and thus, resulting in flickering or waving artifacts. To solve this problem, we detect the camera motion based on the SIFT features and then decompose the scene into foreground and background regions. Obviously, the background motions depend on the camera while the foreground motions are arbitrary. We introduce different constraints to preserve their temporal coherences due to their different natures. All the criteria are formulated into energy terms and we solve for the resized images and videos by minimizing the objective function.

Keyword: Content Aware, Image and Video Resizing, Warping, Temporal Coherence, SIFT features, Optimization

Acknowledgements

My sincere thanks to my advisor Tong-Yee Lee. His advice and guidance have been invaluable, his enthusiasm and scientific knowledge both motivating and inspiring. I am also grateful to Prof. Yung-Nien Sun, his leadership and wisdom guides the visual system laboratory keep improving. Many thanks also to Prof. Chiew-Lan Tai, Prof. Kwan-Liu Ma and Prof. Olga Sorkine. It has been a great experience to work with them and share their knowledges in many interesting discussions during the internship in Hong Kong University of Science and Technology, University of California, Davis and New York University.

I have learned much from my other collaborators: Oscar Kin-Chung Au, Hongbo Fu, Carlos Correa, Chaoli Wang and Hung-Kuo Chu, each of whom I hope to work with again someday. And thanks to all the members in the visual system laboratory, National Cheng Kung University, for their companionship and encouragement.

Finally, I am grateful to my family for the love and endless support that made it all possible. I dedicate this dissertation to them.

Table of Contents

List of Figures

Chapter 1 Introduction

Research on automatic resizing of images and videos is becoming ever more important with the proliferation of display units, such as television, notebooks, PDAs and cell phones, which all come in different aspect ratios and resolutions. Cropping images or videos to fit the display medium inevitably discards information, and resizing the contents to arbitrary aspect ratios produces distortion. Any homogeneous algorithm that applies the same operator to every local region (e.g., bicubic interpolation) distributes distortion equally, the prominent objects to appear squashed or squeezed. Our goal is to design a resizing scheme that minimizes noticeable distortion of prominent features and structural objects, such as people, vehicles or buildings.

Start from image resizing, seam carving [1,2] and image warping [3,4] have been proposed to resize images non-homogeneously. Seam carving greedily removes or inserts 1D seams that pass through the less important regions in the image. Warping methods place a grid mesh onto the image and then compute a new geometry for this mesh, such that the boundaries fit the new desired image dimensions, and the quad faces covering important image regions remain intact at the expense of larger distortion to the other quads. Since humans are less sensitive to distortion of homogeneous information, such as clouds or sea, both classes of methods attempt to keep the prominent objects untouched and distort only the homogeneous regions. Unfortunately, keeping the prominent objects unchanged is certain to fail if their widths are larger than the target image width. In other words, the absence of homogeneous regions along the resizing direction would cause obvious distortion.

It follows that both the carving and the warping approaches proposed so far do not utilize well the homogenous information along the direction orthogonal to the resizing direction. As shown by the *Car* example in Figure 1.1, squeezing or stretching the sky and clouds clearly causes only little distortion, however reducing the image width by compressing the horizontal pixels not only warps the sky but also distorts the car. In contrast, reducing the image width indirectly by increasing its height, followed by a downsampling, can utilize the sky better to absorb distortion. Unfortunately, whether

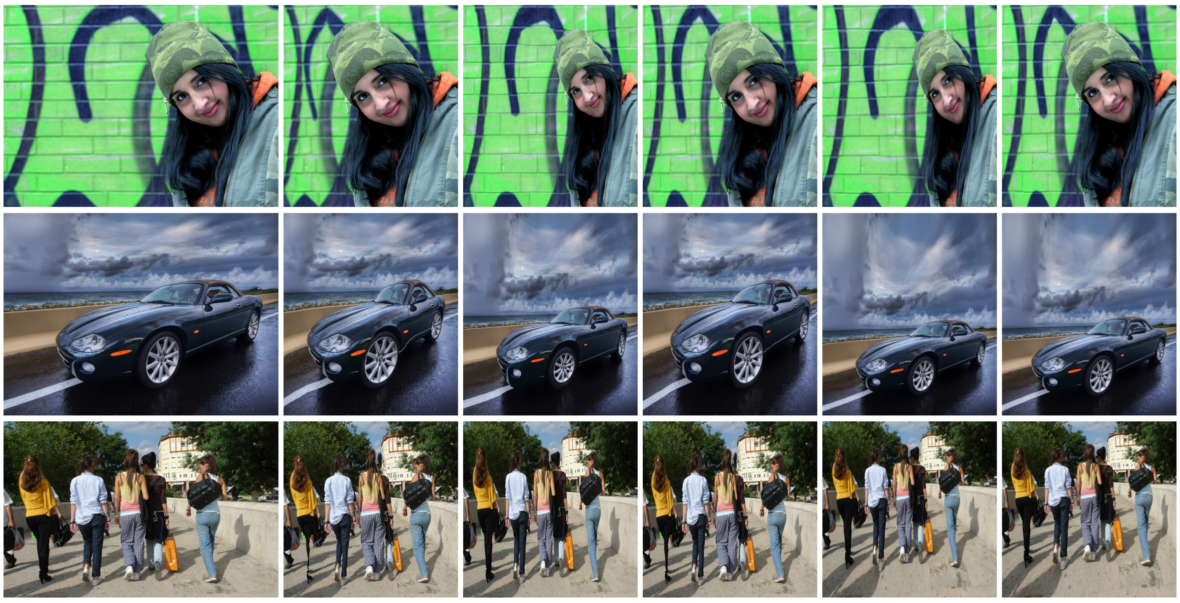


Figure 1.1: The middle columns present direct and indirect narrowing results by the forward seam carving method [2] (denoted as *forward SC*) and the warping method of [4]. Indirect narrowing achieves the desired aspect ratio by first increasing the height of the image and then downsampling. Note that structural objects, such as the car and jumping people, suffer obvious discontinuity with seam carving. Direct resizing by seam carving preserves the girl's face but not the car, and indirect resizing preserves the car but not the girl's face. This is because the homogeneous information is distributed horizontally in the *Girl* image and vertically in the *Car* image. For the more complex *Walking people* image, both previous methods cannot preserve well the aspect ratios of the people with either direct or indirect resizing strategy due to their one-directional propagation of distortion. Our algorithm achieves better results for all three images.

the direct or indirect resizing will achieve better results depends on the image content: the two strategies produce opposite effects for the *Car* and the *Girl* images. In contrast to the previous approaches, our algorithm finds optimal uniform scaling for prominent objects by means of a global optimization, thus implicitly allowing distortion to be distributed in all directions to fully utilize the available homogeneous areas.

Specifically, we represent the input image with a grid mesh and deform the mesh to achieve image resizing. The importance value of each pixel is determined by the combination of gradient magnitude and the saliency measure. We then set the quad importance as the average of its interior pixel importance values. To absorb distortions using all the homogeneous regions, we determine a scaling factor for each quad and wish the deformed quad as similar as possible to its original version. Due to the various importance values, the quads covering prominent objects are uniformly resized while quads with homogeneous materials inside are squeezed and stretched to satisfy the target resolution. The overall resizing scheme is formulated into energy terms and we minimize the objective function to achieve image resizing. Since both the scaling and the deformed mesh are unknown, we solve this non linear objective function by alternatively updating the deformed vertex positions and the quad scaling factors.

The extension from image resizing to video resizing requires the consistent resizing of corresponding objects between neighboring frames. Naïvely resizing individual frames in a content-aware manner easily leads to temporal incoherence, causing flickering or waving artifacts. To address the problem, previous works consider videos as spatiotemporal cubes and constrains temporally adjacent pixels to transform coherently (by “temporally adjacent pixels”, we mean pixels in consecutive frames that have the same spatial location, up to 1-ring neighborhood). However, this approach often fails to guarantee temporal coherence, since it is *motion-oblivious*: it assumes that features remain in the same spatial location or 1-ring neighborhood between consecutive frames, and this assumption breaks down when large camera or object motion is presented. For example, camera zooming makes object features occupy regions of different sizes even between consecutive frames, possibly causing seam carving to remove features inconsistently across frames due to its strategy of one seam removal per frame. Camera or object sliding also easily leads to deviation from correspondence between temporally adjacent pixels (Figure 1.2). Similar temporal incoherence problems happen with the methods based on non-uniform warping [4,5] (see examples in the accompanying videos).

We introduce *motion-aware* constraints for temporally coherent resizing of videos, which, to the best of our knowledge, have not been studied before. We observe that temporal coherence can be achieved by preserving the motion information of the input

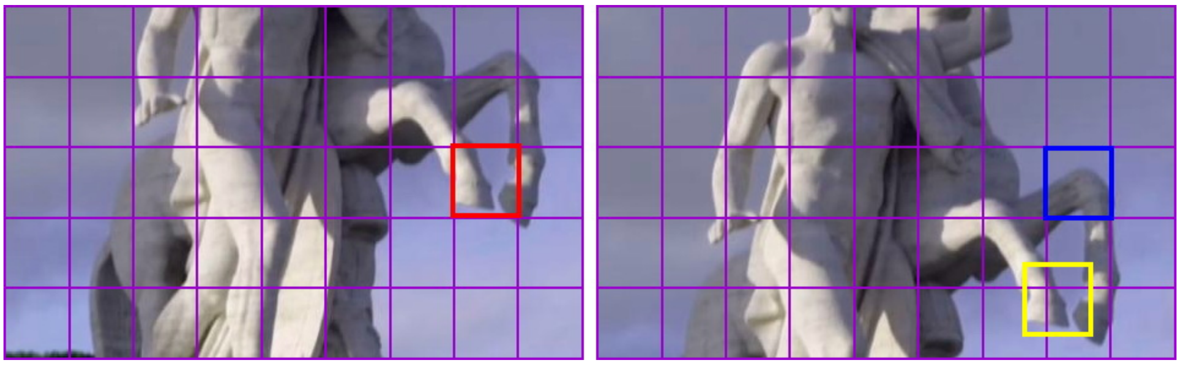


Figure 1.2: Object or camera motion diverts feature correspondence from temporally adjacent pixels. In this example, due to camera movement, features within the quad in red should be constrained to those within the yellow quad instead of the temporally adjacent quad in blue.

video, usually consisting of camera and object motion, and we thus design separate constraints to preserve camera motion and object motion. Specifically, we align every pair of consecutive frames (i.e., grid meshes) using their interframe camera motion and constrain their relative positions to retain camera motion. We preserve object motion by detecting distinct moving areas of objects across multiple aligned frames and constraining each of them to be resized consistently. To combine with the spatial preservation of prominent objects, our final resizing optimization is naturally formulated over all video frames aligned in a common camera coordinate system, where the resizing effect of individual frames is driven by content-aware deformation of per-frame uniform grids. We strike a balance between spatial content preservation and temporal coherence to minimize the visual artifacts. Finally, to improve performance and scalability, we break long video sequences into short overlapping clips and resize the individual clips in a streaming manner while constraining their in-between temporal coherence over the overlapping frames.

In addition to retargeting images and videos, our resizing technique can also benefit the Focus+Context visualization. The need to examine and manipulate large surface models is commonly found in many science, engineering, and medical applications. On a desktop monitor, however, seeing the whole model in detail is not possible. Therefore,

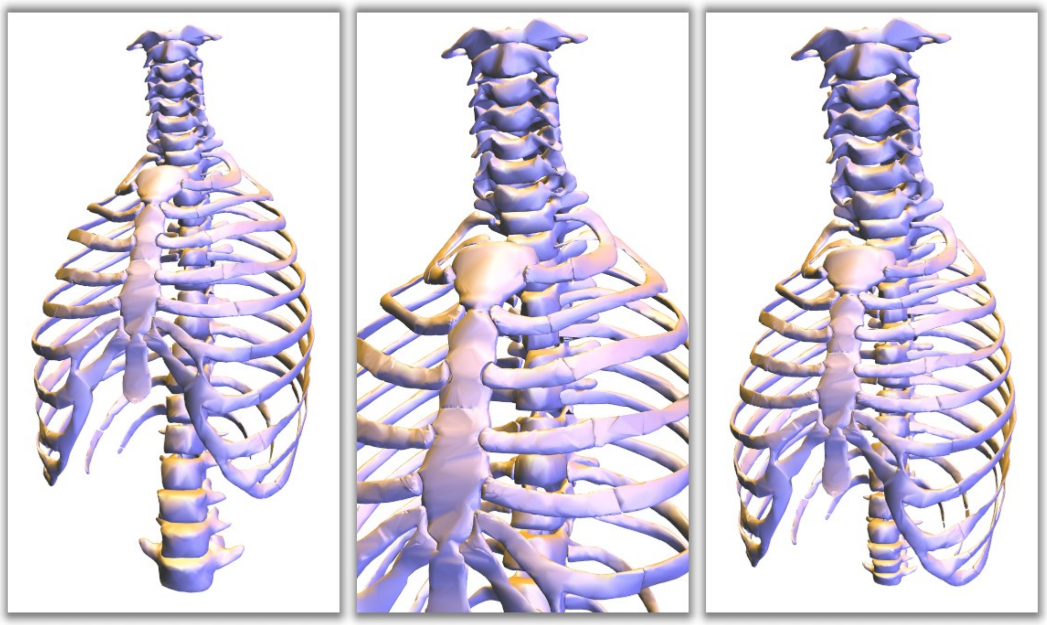


Figure 1.3: (left) Original view of the thorax model. For a detailed observation of the cervical vertebra, an intuitive approach is to shorten the distance between the model and the camera. However, the other regions, such as the lower part of the spine, will be clipped off due to the limited screen space (middle). In contrast, our method magnifies the focal region while keeping the entire model displayed on the same screen (right).

previous methods allow the user moving the mouse cursor to specify a region that he/she wants to observe in more detail, and the system displays an enlarged version of that region in another part of the screen. Unfortunately, such a straightforward sub-window technique requires the user to interpret translation relation between the sub-window and the main window. Another simple approach is to zoom in on the region of interest while cropping off parts of object that are farther away. Such local magnification visualization, however, cannot maintain a full view of the model (see Figure 1.3).

To display complex models on the screen which has limited resolution, researchers have proposed Focus+Context frameworks, which magnify the area of interest without clipping off the other parts [6–12]. These methods expand the region of interest through the theory of optical lens or other distortion methods to achieve this aim. However, none of them has addressed the issue of stretching and distorting the remaining parts of the model while magnifying a specific region. While expanding the focal region, these methods simply let the distortion occur in the surrounding area and ignore the artifact. In contrast, our method lets the free space absorb the resulting distortion rather than letting the distortion uniformly spread throughout the nearby spaces.

We present a new, interactive Focus+Context method for visualizing large surface models. Our method, based on an energy optimization model, allows the user to magnify an area of interest to see it in detail while deforming the rest of the area without perceivable distortion. The rest of the surface area is essentially shrunk to use as little of the screen space as possible in order to keep the entire model displayed on screen. This method allows the user to clearly observe the model's detail in the region of interest while not losing the overall view of the model's shape and topology. The visualization conveys a complete visual message to the user and reminds the user of the overall perception of the model at all time, while the user's attention is focused on a local region. We demonstrate the efficacy and robustness of our method with a variety of models.

Chapter 2 Related Work

Image Retargeting

Many algorithms have appeared in the literature for retargeting images to displays of different resolutions and aspect ratios. Traditional methods perform homogeneous resizing without considering the image content, equally propagating the distortion over the entire image and noticeably squeezing prominent objects. To achieve resizing without distortion, many approaches attempt to remove the unimportant information from the image periphery [13–16]. Based on a face detection technique [17] and a saliency measure [18,19], the image is cropped to fit the target aspect ratio and then uniformly resized by traditional interpolation. However, cropping methods may potentially remove prominent objects close to the image boundary.

Recently proposed retargeting methods try to retain prominent objects while reducing or removing other image content. Seam carving methods [1,2] reduce the image size in a certain direction by removing monotonic 1D seams of pixels that run roughly in the orthogonal direction (image expansion is achieved by duplicating such seams instead). To reduce artifacts, they search for minimal-cost seams that pass through homogeneous regions by computing their forward [2] or backward energy [1]. These methods produce very impressive results, however their discrete nature may cause noticeable jags in structural objects.

Image warping [3,4] offers a continuous solution to image resizing. The warping functions are generally obtained by a global optimization that squeezes or stretches homogeneous regions to minimize the resulting distortion. Gal et al. [3] warp an image into various shapes, enforcing the user-specified features to undergo similarity transformations. Wolf et al. [4] automatically determine the importance of each pixel and merge the pixels of lesser importance in the reduction direction. Gal et al. [3] employ a simple heuristic to determine the scaling of the marked features, so that when the image is squeezed in one direction, the features are uniformly scaled by the squeezing ratio, and when stretched, the features do not scale at all.

Redistribution of pixels under patch-based coherence and completeness constraints is studied in [20,21]. These methods afford more flexibility for image editing operations, including image resizing, though at much higher computational cost. The concepts from image retargeting have also been transferred to content-aware shape resizing [22] and focus+context visualization of 3D models [23].

Video Retargeting

Almost all the image retargeting methods can be adapted to resize videos by addressing two problems: augmenting image importance models with motion information and resizing individual frames in a temporally coherent manner. We show that the influence of camera and object motion should be considered in both problems. However, to the best of our knowledge, none of the existing importance models except those used in the cropping-based retargeting methods [24,25] take camera motion into account. Liu and Gleicher [24] compute motion contrast, i.e., the motion at each pixel subtracted from the background motion, to define motion saliency, which is then incorporated into the importance model together with image saliency and object saliency. Tao et al. [25] explicitly extract moving foreground objects to solely define important parts.

The cropping-based retargeting methods achieve temporally coherent results by searching for a smooth cropping sequence. The retargeting methods involving local redistribution of pixels demand pixel-level temporal coherence, which is apparently more difficult. The existing methods enforce coherence between temporally adjacent pixels in a spatiotemporal video cube. For example, Wolf et al. [4] propose to penalize position changes of temporally adjacent pixels in a linear least-squares optimization formulation. Similar temporal coherence is achieved using a 3D random walk model in [5], which instead focuses on improving the efficiency of [4]. Rubinstein et al. [2] obtain time-smoothing seams by solving for monotonic 2D connected manifold seams using graph cuts. However, we observed that simply enforcing constraints between temporally adjacent pixels is often insufficient or even invalid, especially when large object movement or large camera motion is involved, causing flickering or waving artifacts.

Note that the above retargeting methods have their own advantages and disadvantages [2]. For example, compared to cropping-based methods, which completely discard less important regions, nonlinear retargeting, such as seam carving and nonuniform warping, has better ability to preserve scene context at the cost of allowing some degree of distortion, especially to less important regions. Our resizing framework, as another nonlinear retargeting method, is proposed not to replace any existing resizing tool, but to provide users more options for their specific needs. As recently shown, several types of retargeting methods may need to be used together to produce visually pleasing resizing of a general image or video [26].

Focus+Context Visualization

Many related algorithms have been developed to visualize complicated information of 3D volumetric models. The outer opaque layer always overlays the internal information and results in visualization problems. Hence, Viola et al. [27] automatically compute the importance of each voxel to avoid hiding the important regions by the outer trivial voxels. Zhou et al. [28] advocate a feature-based method to enhance the volumetric features and render the parts of the model inside and outside the focal region in different styles. McGuffin et al. [29] applied deformation techniques to browse volumetric data. Their approach opens up, spreads apart, or peels away the outer layers to reveal the hidden structures.

To visualize tiny information, such as the bump surface of a human colon, in a clear and detailed view, researchers have proposed many methods to magnify the focal area and either distort or overlay the neighboring regions to highlight the region of interest. Keahey et al. [6–8] deformed the texts or 2D images by a transformation grid, determined by nonlinear magnification fields. Bier et al. [30] presented an intuitive interface for the user to specify the focal region and render the information inside the focal region with a different style to enhance the feature of interest. Carpendale et al. [9,10] proposed several distortion patterns, such as stretch orthogonal and nonlinear radial, to demand more space for the focal region to achieve 3D distortion that is independent of the viewpoint. LaMar et al. [11] applied hardware acceleration to deform the rendered 2D images or 3D volumes. They accomplished the goal by dynamically computing the texture coordinates for the grid vertices of the applied mask and rendering the texture with the coordinates that are projected onto the homogeneous space to make the results desirable. Unlike the above-mentioned methods, Wang et al. [12] proposed an interactive technique to render volumetric models according to optical-lens theory. Their method simulates the ray direction that is determined by the position of the focal point and displays the expanded image within the magnifying lens. Both [11] and [12] provide different shapes of bounded lens for the user to magnify the regions of interest.

Chapter 3

Optimized Scale-and-Stretch for Image Resizing

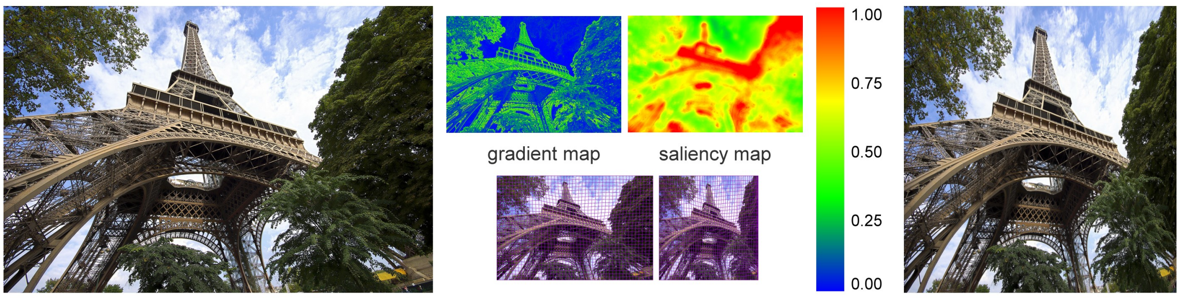


Figure 3.1: We partition the original image (left) into a grid mesh and deform it to fit the new desired dimensions (right), such that the quad faces covering important image regions are optimized to scale uniformly while regions with homogeneous content are allowed to be distorted. The scaling and stretching of the image content is guided by a significance map which combines the gradient and the saliency maps.

3.1 Overview

We present a "scale-and-stretch" warping method that allows resizing images into arbitrary aspect ratios while preserving visually prominent features. To minimize the resulting distortion, we determine an optimal scaling factor for each local region (see Figure 3.1) instead of enforcing the size of salient image regions to remain unchanged. The scaling factors are iteratively optimized, and the amount of deformation to each region is guided by a significance map that characterizes the visual attractiveness of each pixel; this significance map is computed automatically using a combination of gradient- and salience-based measures. We call our strategy "optimized scale-and-stretch" since it allows regions with high importance to scale uniformly and regions with homogeneous content to be distorted. We warp the grid mesh that represents the image such that it fits the new image dimensions, and each quad's deformation matches the local scaling factor. The scaling transformations and the positions of the grid vertices are both variables in the global optimization process. We design a very efficient alternating optimization procedure that allows resizing fairly large images in real time and is easy to implement. The efficiency stems from the specially-tailored objective function formulation that reduces the nonlinear problem to a series of linear problems with a fixed system matrix. The matrix can be pre-factorized, and each iterative step only requires a back-substitution.

The key aspect of our method is that the distortion due to image resizing is *optimally* distributed over the image, irrespective of the direction of the resizing operation (horizontal, vertical or both). This gives our technique the full freedom to utilize homogeneous image regions to hide the distortion. Moreover, our method enjoys the advantage of respecting structures within the image (such as straight lines or arches) thanks to the continuity of the warping function.

3.2 Arbitrary image resizing

We represent an image as a mesh $\mathbf{M} = (\mathbf{V}, \mathbf{E}, \mathbf{F})$ with vertices $\mathbf{V}$, edges $\mathbf{E}$ and quad faces $\mathbf{F}$, where $\mathbf{V} = [\mathbf{v}_0^T, \mathbf{v}_1^T, ...\mathbf{v}_{end}^T]$ and $\mathbf{v}_i \in \Re^2$ denote the initial vertex positions. The vertices and edges form horizontal and vertical grid lines partitioning the image into quads. To resize an image of $m \times n$ pixels into an arbitrary size of $m' \times n'$ pixels, we fix the position of the top-left vertex $\mathbf{v}_0$ and let the user specify a new position for the bottom-right vertex $\mathbf{v}_{end}$. The rest of the boundary vertices are constrained to slide along their respective boundaries in order to keep the image rectangular. We solve the problem of finding a deformed mesh geometry $\mathbf{V}' = [\mathbf{v}_0'^T, \mathbf{v}_1'^T, ...\mathbf{v}_{end}'^T]$, where ideally each quad undergoes a transformation that consists of uniform scaling only. Clearly, it is impossible to meet this goal for an arbitrary new image size, so some quads should inevitably deform; we spread the distortion according to the significance of each quad and obtain the new mesh geometry by a global optimization.

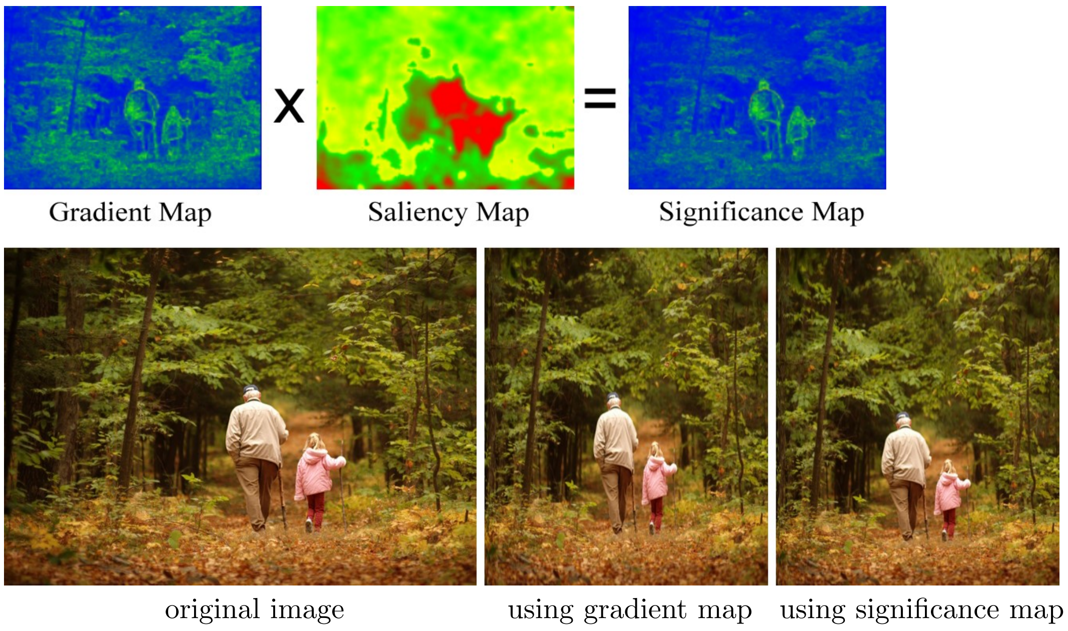


Figure 3.2: We define the significance map as the product of the gradient magnitude and the saliency measure. Compared with the gradient map, the significance map is less sensitive to the disturbance of trees and leaves, focusing on the old man and the little girl. We compare the results of narrowing the original image using the gradient map and our significance map. The shapes of the old man and little girl are preserved better with our significance map.

3.2.1 Quad significance

Previous image retargeting methods have used various measures to determine the significance value of a pixel automatically. Both Avidan and Shamir [1] and Wolf et al. citeWolf2007 consider pixels with large gradient magnitudes as important. Rubinstein et al. [2] determine the pixel significance by accumulating the discontinuity of its neighbors if the pixel is removed. We propose a new measure of quad significance that can better detect prominent image objects. We observe that only quads that are both attractive to the human eye and contain structured objects should be protected against distortion and thus should have high significance values. Hence, we combine two measures to determine the pixel significance: image gradient and saliency map [18]. To compute the saliency, Itti et al. [18] apply various filters to extract color, intensity and orientation properties, and then look for regions that have different properties than their surroundings; this analysis is performed on multiple scales. The gradient indicates the presence of structures, while the saliency map determines the attractiveness of a region. In particular, the gradient magnitude can be misled by trivial and repeated structures, while the saliency measure also considers regions that are attractive but homogeneous as salient. By combining these two measurements, a region is considered significant only if it is both structural and attractive.

Let I denote the input image. We define the pixel significance map as $W = W_\alpha \times W_\beta$, where $W_\alpha = ((\frac{\partial}{\partial x}I)^2 + (\frac{\partial}{\partial y}I)^2)^{1/2}$ is the 2-norm of the gradient and W_β is the saliency map. The significance w_f of quad f is defined as the average pixel significance within the quad. We normalize the w_f values to obtain weighting factors within 0 and 1. Smaller values mean less importance. Figure 3.2 shows an example of a noisy gradient map and the significance map with the trees filtered out by the saliency measure.

3.2.2 Mesh-based image resizing

Given the new image size, represented by the new position for the lower right corner $\mathbf{v}_{end}$, we compute an optimally deformed mesh, such that quads of higher significance enjoy uniform scaling and quads of lower significance are allowed to be distorted more (i.e., non-uniformly squeezed or stretched). In addition, we would like to minimize the bending of the grid lines, because prominent objects usually span a set of connected quads. Specifically, the deformed mesh is solved for by optimizing the quad deformation and grid line bending energy terms subject to boundary constraints.

Quad deformation

We formulate the shape distortion energy for each quad by measuring how far the deformed quad is from a uniformly scaled version of the original quad. Ideally, for each $f \in \mathbf{F}$ there would be a scale factor s_f such that for each vertex $\mathbf{v}$ of the quad, $\mathbf{v}' = s_f\mathbf{v} + \mathbf{t}$ (where $\mathbf{t}$ is a constant translation vector). Let us denote the set of (directed) edges of f by $\mathbf{E}(f)$; the distortion energy of f is defined as

$$D_u(f) = \sum_{\{i,j\}\in\mathbf{E}(f)} \|(\mathbf{v}'_i - \mathbf{v}'_j) - s_f(\mathbf{v}_i - \mathbf{v}_j)\|^2. \tag{3.1}$$

The translation vector $\mathbf{t}$ is eliminated by considering the edge vectors. For equal aspect ratio distortion, larger quads are penalized more, since the distortion is more visible on an enlarged area. Note that the optimal scaling factor s_f is completely defined by $\mathbf{v}_i$ and $\mathbf{v}'_i$: indeed, if the vertices of the original and deformed quads are fixed, we obtain s_f that minimizes $D_u(f)$ by differentiating Eq. (3.1) and equating to zero:

$$\frac{\partial D_u(f)}{\partial s_f} = \sum_{\{i,j\}\in\mathbf{E}(f)} 2\left(\|\mathbf{v}_i - \mathbf{v}_j\|^2 s_f - (\mathbf{v}_i - \mathbf{v}_j)^T(\mathbf{v}'_i - \mathbf{v}'_j)\right)$$

$$\frac{\partial D_u(f)}{\partial s_f} = 0 \;\Rightarrow\; s_f = \frac{\sum_{\{i,j\}\in\mathbf{E}(f)}(\mathbf{v}_i - \mathbf{v}_j)^T(\mathbf{v}'_i - \mathbf{v}'_j)}{\sum_{\{i,j\}\in\mathbf{E}(f)}\|\mathbf{v}_i - \mathbf{v}_j\|^2}. \tag{3.2}$$

Therefore, s_f is defined in terms of the deformed mesh vertices, and $D_u(f)$ is only dependent on those. We define the total mesh uniformity energy by summing up the individual quad energy terms and adding the significance weights, such that more distortion would be allowed in areas of lesser significance:

$$D_u = \sum_{f\in\mathbf{F}} w_f D_u(f)\,. \tag{3.3}$$

Grid line bending

Since prominent objects often occupy multiple connected quads, to prevent their distortion we wish to minimize the bending of the grid lines. Specifically, the optimization system scales the edge lengths but retains the edge orientations during deformation. We define the length ratio of the edges before and after deformation as $l_{ij} = \|\mathbf{v}'_i - \mathbf{v}'_j\|/\|\mathbf{v}_i - \mathbf{v}_j\|$, and introduce the following energy term:

$$D_l = \sum_{\{i,j\}\in\mathbf{E}} \|(\mathbf{v}'_i - \mathbf{v}'_j) - l_{ij}(\mathbf{v}_i - \mathbf{v}_j)\|^2. \tag{3.4}$$

We illustrate the effect of this energy term by comparing with the result of [3] which does not regard grid line bending (see Figure 3.3). We use the same significance map in

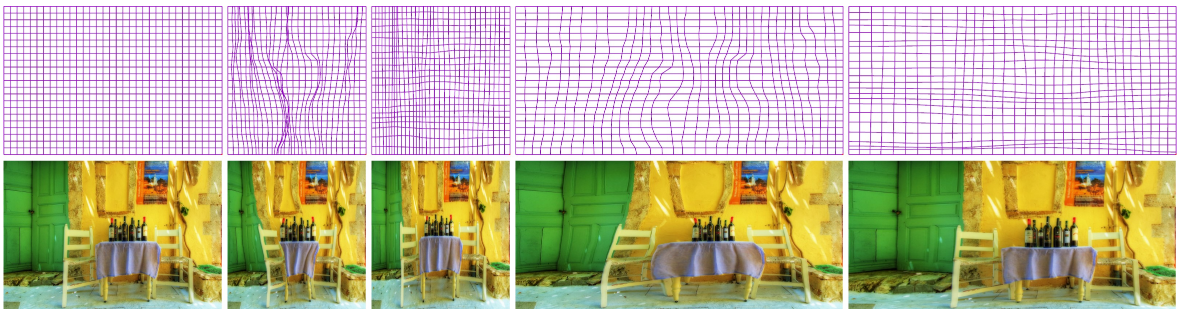


Figure 3.3: We compare our method with [3]. The lower row shows the reduction and expansion results (guided by our significance map in all cases), and the upper row displays the respective deformed meshes. Our grid line bending energy ensures that the grid lines are smoothly bent to reduce distortion, while [3] produces C^0-continuous grid lines, distorting structural objects.

both cases. It can be observed that by preventing the grid lines from serious bending, our algorithm avoids distorting the door and chairs. In addition, this energy term also alleviates the edge flipping problem which could occur in image warping methods [3,4]. This is because the edge ratio l_{ij} is always positive in our definition, encouraging each edge towards its original direction. Figure 3.4 demonstrates the effectiveness of this measure, comparing with the result of [4]. Note that both results were obtained without using constrained systems (see details in the Appendix).

Total energy and boundary conditions

In summary, we wish to minimize the sum of the quad deformation and the line bending energies:

$$D = D_u + D_l\,, \tag{3.5}$$

subject to some boundary constraints. The boundary constraints are the locations of the top left and bottom right vertices

$$\mathbf{v}'_0 = (0,0)^T, \quad \mathbf{v}'_{end} = (n', m')^T, \tag{3.6}$$

and in addition, the y coordinates (or x, respectively) of horizontal (vertical) boundary vertices are constrained to remain constant to make sure we get a rectangular image

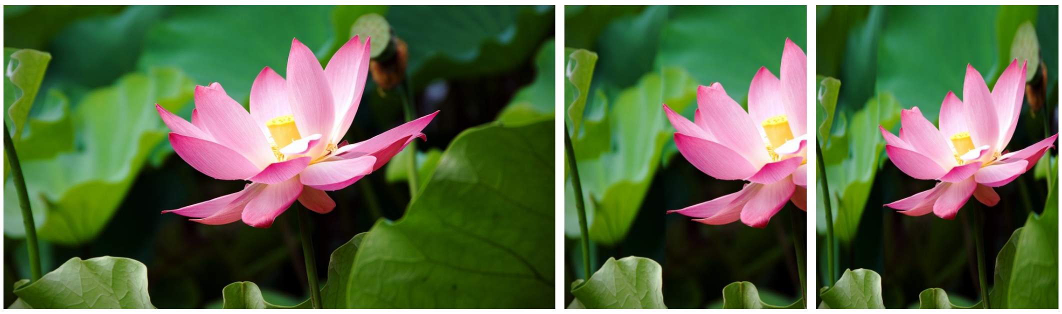


Figure 3.4: The original image is narrowed by [4] and our algorithm. Since [4] only constrains the vertically connected pixels to have similar displacements, self-intersection may occur. In this example, the right part of the image gets flipped, concealing some petals.

shape:

$$
\mathbf{v}'_{i,y} = \begin{cases} 0 & \mathbf{v}_i \text{ is on the top boundary} \\ m' & \mathbf{v}_i \text{ is on the bottom boundary,} \end{cases}
$$
$$
\mathbf{v}'_{i,x} = \begin{cases} 0 & \mathbf{v}_i \text{ is on the left boundary} \\ n' & \mathbf{v}_i \text{ is on the right boundary.} \end{cases} \tag{3.7}
$$

These constraints are simply substituted into the linear system during the optimization.

We solve for the deformed mesh using an iterative solver; note that both the scaling transformations s_f and the deformed edge lengths l_{ij} are unknown, and the latter depend nonlinearly on the vertex positions. The iterative solver starts from an initial guess for $\mathbf{V}'$ (see details in the next section) and determines the quad transformations s_f and the edge ratio l_{ij} for each edge. The new vertex set $\mathbf{V}'$ is then solved for by minimizing the total energy (3.5) subject to constraints (3.6) and (3.7). Note that with s_f and l_{ij} fixed, the energy is a quadratic function of $\mathbf{V}'$, thus the minimization problem is linear. Furthermore, when keeping s_f fixed, the x and y coordinates of the grid vertices are not coupled and can be solved separately with the same system matrix (and different right-hand sides), reusing the matrix factorization. The alternating steps continue until all the vertex movements are smaller than 0.5.

In our experiments we have observed that the scaling transformations s_f of neighboring prominent quads should be similar because the prominent objects usually span

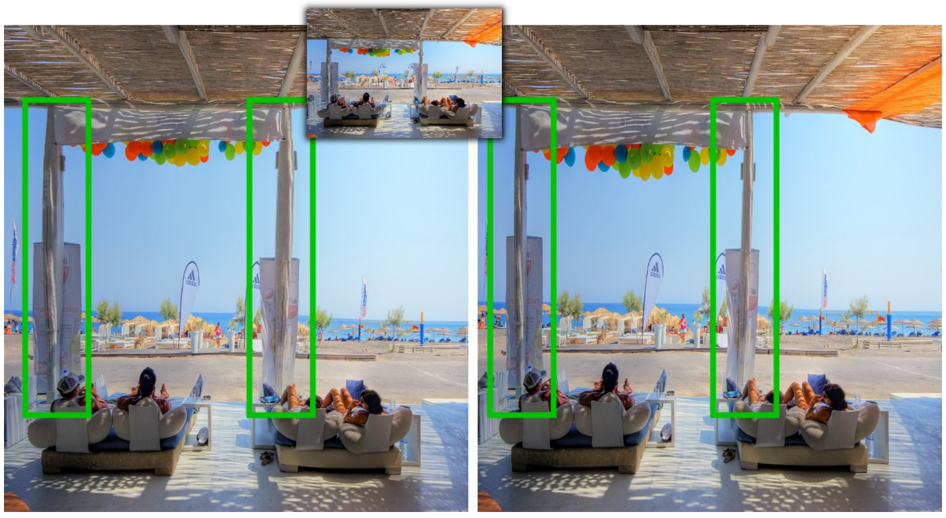


Figure 3.5: The original image (inset) is expanded in height. The sky region is stretched to reduce distortion, causing the quads covering various regions of the pillars to be scaled to different sizes (left). We solve an optimization problem to make the sizes of connected prominent quads similar; this keeps the uniform thickness of the pillars (right).

several quads. We therefore propose to reduce the difference of adjacent scaling factors after they are determined by (3.2), i.e., smooth the s_f's in each iteration. Denote by $\mathbf{N}(f)$ the adjacent quads of f and by w_g the average of all the quad significance values; we obtain the smooth scaling factors s'_f by minimizing the following energy each time:

$$\sum_{f\in\mathbf{F}} \sum_{q\in\mathbf{N}(f)} \frac{1}{2}(w_f + w_q)(s'_f - s'_q)^2 + \sum_{f\in\mathbf{F}} w_g(s'_f - s_f)^2. \tag{3.8}$$

Figure 3.5 shows a resizing example demonstrating the effect of this smoothing process.

To make sure the minimization of the non-negative objective function D eventually converges, we determine the value of D after each new vertex set $\mathbf{V}'$ is obtained and verify that $D^{t+1} \leq D^t$, where t is the iteration number. Specifically, we repeatedly update the vertex set $\mathbf{V}^{t+1} = 0.7\,\mathbf{V}^{t+1} + 0.3\,\mathbf{V}^t$ until the new obtained D^{t+1} is smaller than D^t or all the vertex movements are smaller than 0.5. Overall, the alternating optimization approach is quite efficient, since the system matrix (the gradient matrix of D), as well as the smoothing matrix in (3.8), is sparse and remains fixed. Therefore, we can precompute a sparse factorization, and then each iteration requires only a back-substitution.

3.2.3 Initial guess

The convergence speed of nonlinear optimization depends on its starting position. One possibility is to use the original mesh as the initial guess, however this leads to slow convergence if the final deformed mesh is very different from the original mesh. It is desirable to choose a good starting point that is close to the optimal solution. We use two types of initial guesses in our experiments:

Previous frame. If a user resizes an image by continuously manipulating the handle vertex $\mathbf{v}_{end}$, we take the previous frame as an initial guess. Since the deformation is continuous and the sizes of two consecutive frames are similar, their optimal solutions are expected to be close to each other. We used this type of initial guess for all the examples in this paper except Figure 3.6, and found that the number of iterations is typically less than 10.

Homogenous resizing. If an image is resized directly by specifying the new dimensions, a better initial guess may be obtained by simple homogeneous resizing $(x, y) \to (ax, by)$. Figure 3.6 illustrates such an initial guess and the iterative refinement process.

3.2.4 Significance-aware initial mesh

To achieve interactive resizing, we represent the image by a coarser quad mesh, and create the resized image by linearly interpolating the interior content after the mesh is nonlinearly deformed. The quad mesh can be considered as a subspace of the given image. To reduce the linearization artifacts, it makes sense to place the mesh vertices more densely in the salient regions, so as to approximate the nonlinear deformation better there. Using an adaptive mesh might is one option; however this complicates the mesh structure and makes the implementation more difficult as well. To enjoy the regular structure, we propose to adapt the initial uniform grid mesh $\mathbf{M}$ to the image significance map by slightly deforming the shapes and sizes of the quads while keeping

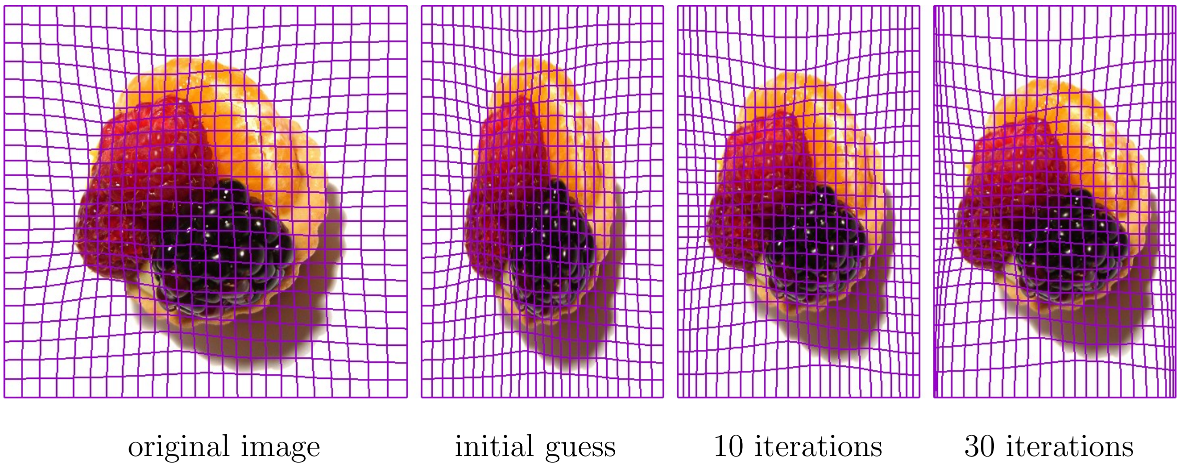


Figure 3.6: The original image is resized starting from an initial guess obtained by homogeneous resizing, and then iteratively refined by our optimization algorithm such that the distortion concentrates in the homogeneous areas.

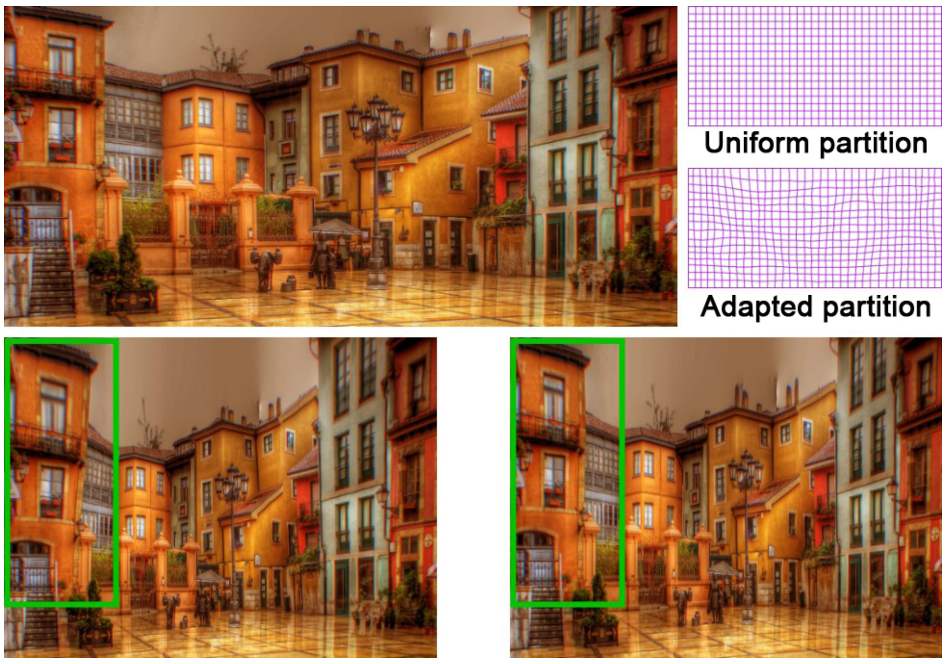


Figure 3.7: Results of resizing an original image (top) using a uniformly partitioned mesh (bottom left) and a significance-aware mesh (bottom right). Notice that the building structures are preserved better with the use of the significance-aware initial mesh.

the mesh connectivity intact (see Figure 3.7).

The basic idea is to attract more vertices to the important regions. Specifically, we shorten an edge $\{i, j\}$ if its nearby quads cover some prominent objects. To deform the mesh, we minimize the following objective functional:

$$\Omega = \sum_{\{i,j\}\in\mathbf{E}} (1 + w_{ij})\|\mathbf{v}'_i - \mathbf{v}'_j\|^2, \tag{3.9}$$

where w_{ij} is a weighting factor determined by averaging the significance of the quads sharing the edge $\{i, j\}$. The constant value 1 is added to reduce the proportion of this weighting factor. The optimization is constrained by the boundary vertices in the same fashion as in Section 3.2.2. Since the entire mesh is constrained to remain rectangular, the edge contraction process actually redistributes edge lengths according to their weighting factors. The edges of higher significance will get shorter, while those of smaller significance will get longer. We solve for the mesh geometry iteratively, updating the vertex positions by minimizing (3.9) in each step and recomputing the weights, as the quad significance changes during the deformation. We stop when all the vertex movements are smaller than 0.5 pixel.

3.3 Results and discussion

We have implemented our image resizing system on a PC with Duo CPU 2.33GHz. The technique is very efficient even though the solver iteratively updates the vertex positions until the process converges. This is because the matrix of our least-squares system is fixed and its factorization can be pre-computed. Therefore, each step only needs a back substitution to determine the vertex positions. Obviously, the computational cost depends on the size of the mesh. A finer mesh leads to better results but slower interactive speed. In all our experiments, we found that an initial quad of 20×20 pixels produces sufficiently good results. Using such a coarse mesh is reasonable since the deformations applied to neighboring regions should be similar. Factoring a matrix of 1976 vertices for a 1024×754 image typically takes 0.034 seconds, and a back substitution to update the vertex positions takes 0.002 seconds.

We demonstrate the effectiveness of our resizing technique with several examples (see Figures 1.1, 3.3, 3.4, 3.8, 3.9, 4.8). By adding user interface to place and manipulate additional positional constraints, our system also allows the user to perform freeform deformation on general images without significantly distorting the prominent objects. We show the interactive system in the accompanying video.

Edge flipping, which leads to local self-intersections, may occur in warping algorithms. This causes discontinuity and may conceal prominent objects. Solving for the deformed mesh using a constrained system (see the Appendix for details) can completely eliminate this problem, with the tradeoff of increased computational cost. Fortunately, our grid line bending energy is effective in alleviating edge flipping; it rarely occurs even when the constrained system is turned off. This energy component also reduces the frequency of re-factorization when the constrained system is turned on.

Comparison

In Figure 3.11, we compare some results of [2,4] and our method. It can be observed that Wolf et al.'s and our methods, which are both warp-based, produce smooth results, while seam carving produces noticeable discontinuity, especially in images containing structural objects. For instance, the coral, people, San Francisco Heart and house roof are jagged since the pixels are directly removed. Comparing with the results of [4], our method can preserve the aspect ratios of prominent objects better and avoid self-intersections (see supplemental material for more comparison results). In Figure 3.9, we demonstrate that seam carving guided by forward energy [2] produces less discontinuities than seam carving using our significance map. This is because the forward energy considers the discontinuity of merged pixels after seam removal, while our significance map does not. Nevertheless, our significance map coupled with our optimized warping produces smoother results and preserves structural objects better than seam carving with forward energy. We believe the combination of our significance map and the forward energy can also benefit the seam carving method.

Seam carving is excellent in dealing with highly textured areas. For example, removing pixels from a sandy region would be better than compressing it. In addition, direct editing of pixels in seam carving methods leads to higher flexibility in terms of

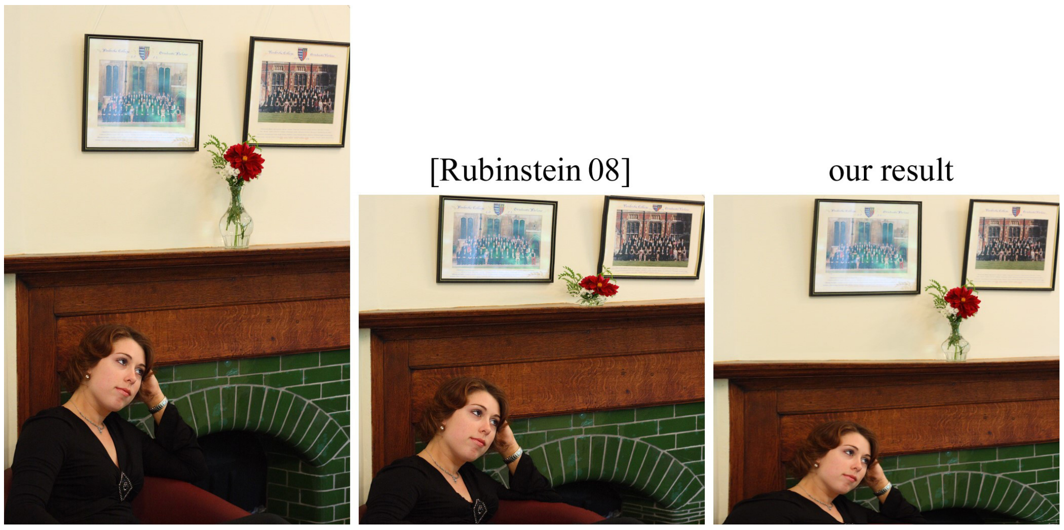


Figure 3.8: Greedy resizing strategies may produce artifacts. When the image on the left is shortened by [2], seams passing through the flower vase have low accumulated distortion; however, their removal destroys the vase structure. Our algorithm solves a global optimization and preserves the vase.

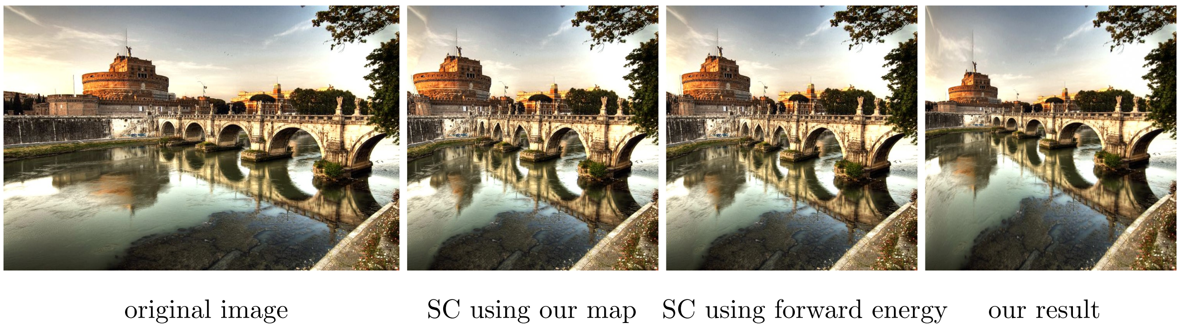


Figure 3.9: Comparison of our result with seam carving methods (denoted by SC) using our significance map and the forward energy. SC with forward energy does a better job of preserving the structures of the castle and bridge than SC using our map. In contrast, our warp-based method has no discontinuity problems, and preserves the aspect ratios of the castle and bridge better than SC with forward energy.

image content manipulation, enabling applications like object removal. However, the discrete nature of seam carving may damage structures because the information on the removed seam is lost. In contrast, warping methods have better ability to preserve structural objects since the applied deformations are continuous and the image components are linearly interpolated. Moreover, warping methods solve a global optimization on the entire image at once, rather than proceeding sequentially on 1D components, thus they do not suffer from greediness effects. Figure 3.8 demonstrates how the greedy strategy could misdirect the resizing process. Finally, the computational cost of warping methods is independent of the resizing dimensions while that of seam carving is proportional to the number of seams.

Limitations

Our algorithm may contract a quad into a line or even a point, removing the quad content. Such direct removal may introduce discontinuities (although the removed information is the least important). It could be argued that when the significance map allows it, our algorithm performs cropping, which is actually the most reasonable in some cases.

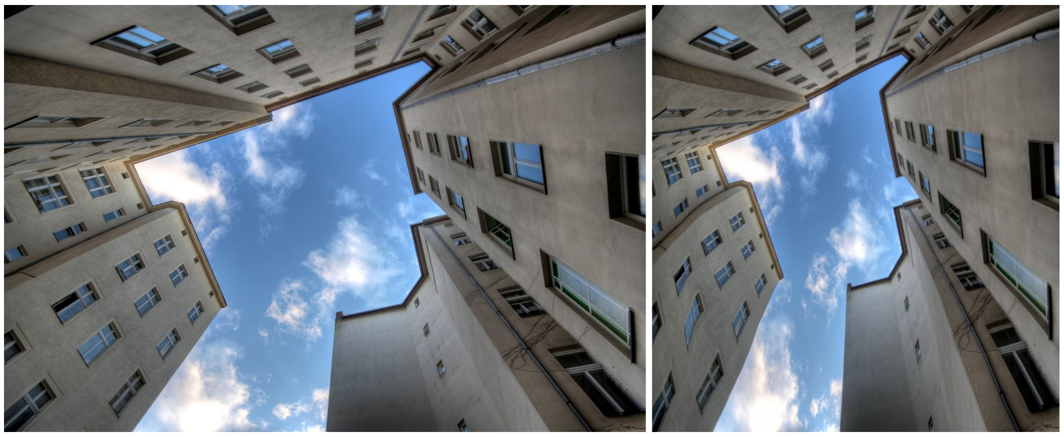


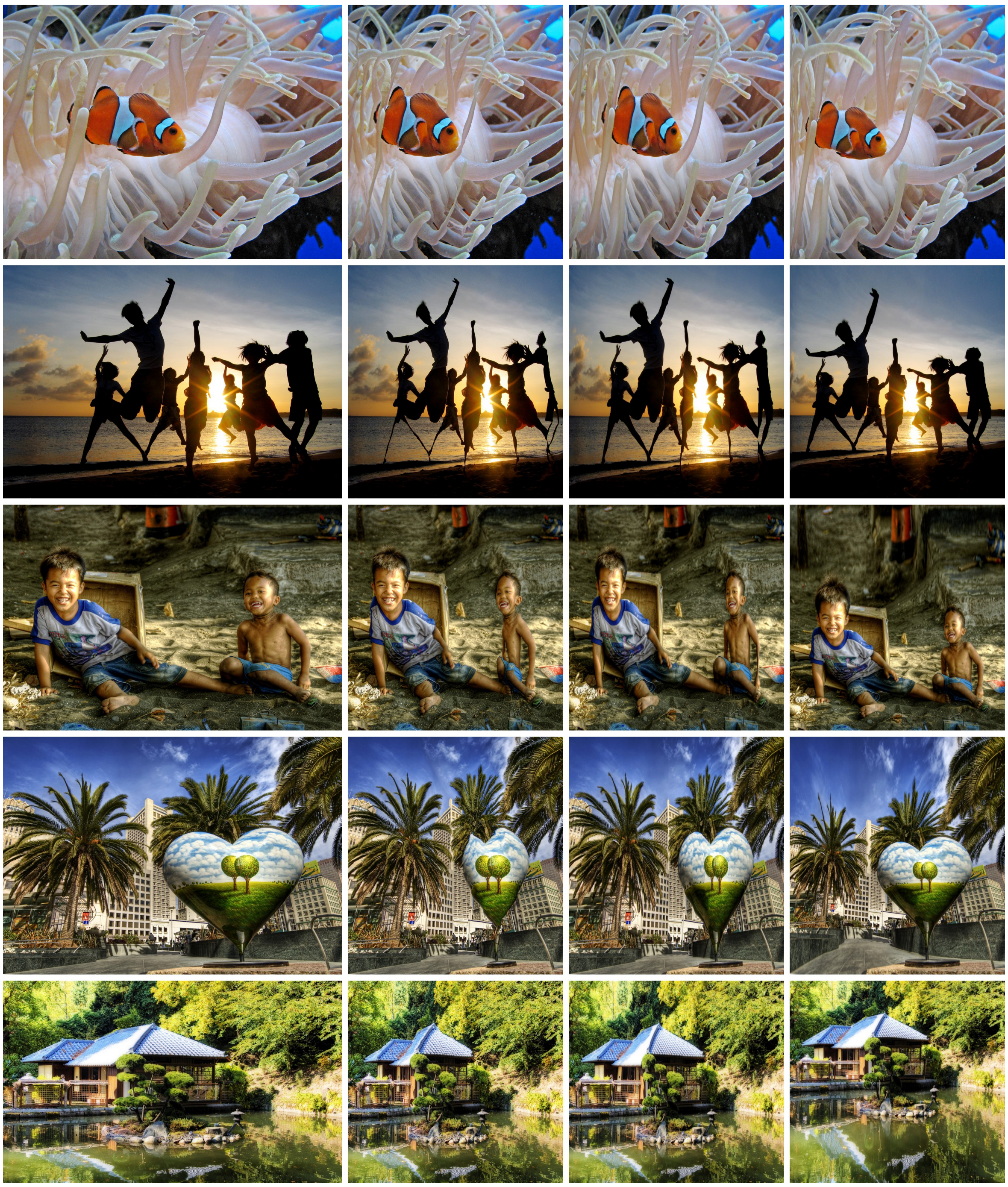
Figure 3.10: (left) Original image. (right) Prominent lines not parallel to the main axes may be distorted due to different degrees of squeezing of adjacent quads.

Our algorithm may significantly stretch homogeneous quads to preserve the aspect ratios of prominent objects, causing the linear interpolation of the quad interior to show artifacts. Filling the interior using texture synthesis methods can solve this problem [31]. For images without any homogeneous regions, our method has no region to which to concentrate the distortion and thus produces results similar to conventional (homogeneous) resizing.

Like all previous methods, our method may fail to preserve the shapes of prominent image lines of arbitrary orientations (as opposed to parallel to the main axes) due to the quads not being aligned with the feature. The adapted mesh we create is an attempt to compensate for that somewhat, but obviously, if the feature is diagonally oriented, it is very hard to adapt the mesh by our mechanism. Although our grid line bending energy term can reduce the bending distortion, when adjacent quads covering the prominent line are squeezed to different degrees, the slope of each prominent segment may change inconsistently (see Figure 3.10). User-defined marking of the quads covering prominent lines and increasing their significance can successfully keep prominent lines straight. However, this strategy may lead to an over-constrained system causing other regions to distort more.

Appendix

To completely eliminate the mesh self-intersection problem, we can solve for the deformed mesh while checking the inequality constraints $\mathbf{e}_{ij}^T \mathbf{e}'_{ij} \geq 0$, where $\{i, j\} \in \mathbf{E}$, $\mathbf{e}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ and similarly for $\mathbf{e}'_{ij}$. We modify the objective function to be $L = D + \sum_{ij} \lambda_{ij} \|\mathbf{e}'_{ij}\|^2$, where λ_{ij} is a weighting factor determined according to the inspection of the mesh after each iteration. Initially we set $\lambda_{ij} = 0$ since all the inequality constraints are satisfied. During the search for the optimal solution, we detect conflicts with the inequality constraints after each iteration. If this happens, we increase the corresponding factor λ_{ij} to a large number (we used 10000) to enforce the flipping edge to have zero length. This changes the objective functional L, so we recompute the system matrix $\partial L / \partial \mathbf{V}'$ and its factorization.


original image
[Rubinstein08]
[Wolf07]
our results

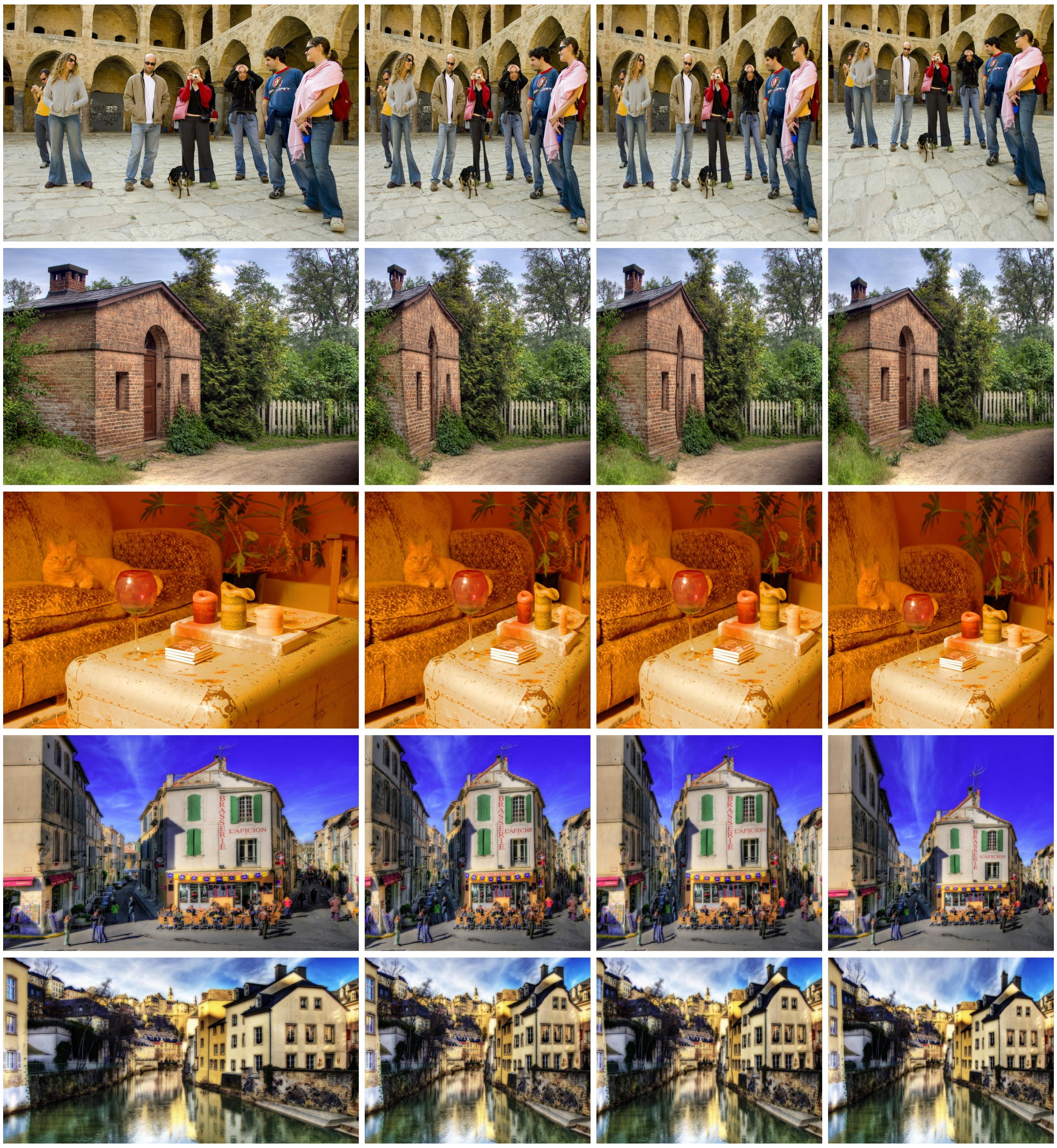
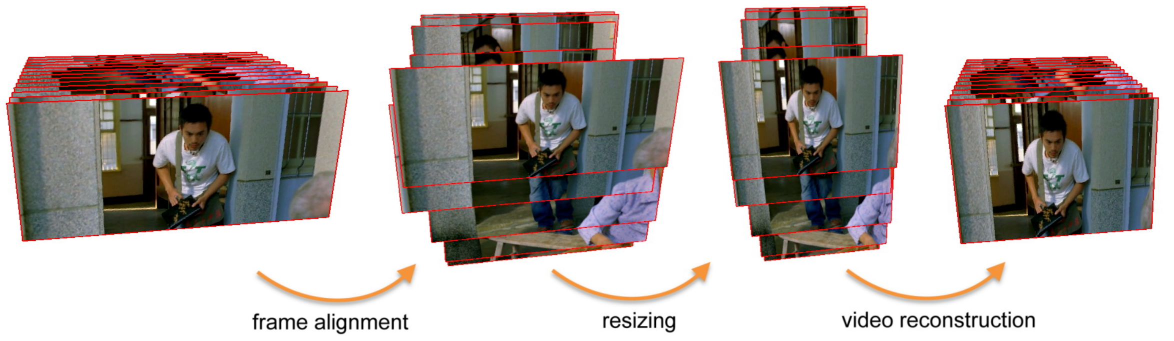


Figure 3.11: Comparison of our results with those of improved seam carving [2] and the warping method of [4]. The results of [4] and our method tend to be smoother than those of seam carving. Notice the discontinuities in the corals, people, San Francisco Heart and the house roof, which are due to the pixels being directly removed. Compared with [4], our method can preserve the aspect ratios of prominent features better.

Chapter 4

Motion-Aware Temporal Coherence for Video Resizing



©ARS Film Production

Figure 4.1: Overview of our automatic content-aware video resizing framework. We align the original frames of a video clip to a common coordinate system by estimating inter-frame camera motion, so that corresponding components have roughly the same spatial coordinates. We achieve spatially and temporally coherent resizing of the aligned frames by preserving the relative positions of corresponding components within a grid-based optimization framework. The final resized video is reconstructed by transforming every video frame back to the original coordinate system.

4.1 Overview

An ideal solution to temporally coherent video resizing is to first recognize compatible objects across different video frames and then resize them within individual frames in a consistent manner. However, this involves object recognition and tracking, which are challenging tasks on their own. Observing that achieving temporal coherence largely means avoiding motion artifacts, such as flickering and waving, we aim to preserve the motion information in an input video, usually consisting of camera motion and object motion.

Camera motion and object motions have very different natures, demanding separate strategies to preserve them. Camera motion is of low degree of freedom and brings a global visual effect to whole scene, usually containing both static and dynamic objects. Assuming that input videos always contain static objects (e.g., static background) whose visual movement is completely due to camera movement, we use a feature-based method to estimate the camera motion between every pair of consecutive frames (Section 4.2.1). By "object motion", we refer to the intrinsic motion of dynamic objects, independent of camera movement. Object motion can often be of high degree of freedom and simultaneously caused by multiple objects at different locations. Precise estimation of object motion is a challenging task. Fortunately, by the smooth warping nature of the core technique, it is sufficient to use the remaining motion subtracted from the camera motion to roughly estimate object motion, avoiding the necessity for precise alpha-masks of the dynamic objects.

We build a new video resizing framework by designing motion-aware temporal coherence constraints (Section 4.3.2) and applying the importance maps to guide content-aware resizing of individual frames (Section 3.2.2, also briefly denoted in Section 4.3.1). We embed each frame into a uniform grid mesh. Our system simultaneously deforms all the meshes with spatial and temporal constraints. We preserve camera motion by constraining relative positions of every two consecutive meshes, aligned using the estimated interframe camera motion. We achieve temporally coherent resizing of dynamic objects by detecting their moving areas and deforming each distinct moving area in a consistent manner. As our temporal constraints and importance maps are all dependent on the frame alignment, we found it more intuitive to formulate the optimization in a common camera coordinate system. Once we obtain the deformed meshes, we transform them back to the original coordinate system of each frame and warp the corresponding images to produce the final resized video. Figure 4.1 gives an overview of our resizing framework. Note that we show every twentieth frame for better visualization.

4.2 Video Importance Map

In this section we first introduce our adopted method for frame alignment and then present a method to compute an importance map for each video frame, such that the consecutive maps change smoothly. Instead of defining the importance map of each frame individually, we measure the importance of a region by considering the contents of neighboring frames that are aligned at that region.

4.2.1 Frame Alignment

We align video frames by estimating camera motion between every two consecutive frames. Camera motion estimation has been studied extensively (see [32] for an insightful survey). Our preliminary experimentation with a 2D affine transformation as the camera model easily gave unreliable results when the area occupied by dynamic objects was significantly increasing. To trade precision for robustness, we express inter-frame motion using a restricted model which consists of scaling and translation. While losing the ability of modeling camera roll operations, which are seldom used in video production, our model is able to robustly estimate the other camera motion effects, such as sliding, zoom, yaw, and pitch. Although we are aware that a 2D projective transformation might be a more precise camera model for this task, we found that this restricted model is more robust and works well for most regular videos. More importantly, it allows us to solve for the x and y components in the optimization separately, thus significantly reducing computational cost and memory requirements (Section 4.3).

We employ a feature-based method to estimate our camera model, similar to those used in the literature of video stabilization [33,34]. We first detect the feature points of each frame by SIFT [35], which is reported to perform best among many local feature descriptors. We then use RANSAC [36] to robustly extract the feature correspondence between the frames and estimate the restricted model (i.e., solving for the scaling and translation parameters). We need to handle a degenerate problem when there are too few pairs of feature correspondence found and will discuss its solution in Section 4.4.

Once we obtain the transformations between every pair of consecutive frames, we are able to accumulate them to align the video frames to a common camera coordinate system. Figures 4.1 and 4.7 show some examples of frame alignment with respect to a camera coordinate system defined at the first frame of a video clip. Note that we do not compute alignment between every frame back to a fixed reference frame, since temporal incoherence is often noticeable only for neighboring frames. Even more importantly, there generally exists no single reference frame that shares sufficient backgrounds with every other frame to allow for robust alignment. We denote by $\mathbf{T}_{t\mapsto\ell}$ the accumulated transformation from frame t to frame ℓ, which transforms pixels at time t to the coordinate system defined at time ℓ. We use homogeneous coordinates to represent positions and vectors and thus express $\mathbf{T}_{t\mapsto\ell}$ as 3×3 matrices. Note that we have $\mathbf{T}_{\ell\mapsto t} = (\mathbf{T}_{t\mapsto\ell})^{-1}$. The accumulated transformations will be used for both our importance map computation and the motion-aware temporal coherence formulation. We discuss the quality of the accumulated transformations further in Section 4.4.

4.2.2 Aligned Importance Map Blending

Since the importance map of each frame largely determines how each image is non-uniformly deformed during resizing, we require importance maps that change smoothly across adjacent frames. This motivated us to define the importance map of each frame by blending the importance maps of neighboring frames at aligned positions. Each importance map takes into account salient information in both spatial and temporal context, but excludes motion purely caused by camera movement, since it is almost homogeneous within individual frames and thus of little importance. Specifically, we define the blended importance value at pixel $\mathbf{p}$ of frame t as

$$\bar{I}^t(\mathbf{p}) = \max_{\ell=t}^{t+k}\{I^t(\mathbf{p}),\ \delta\ I^t(\mathbf{p}) + (1-\delta)\ I^\ell(\mathbf{T}_{t\mapsto\ell}\ \mathbf{p})\}, \tag{4.1}$$

where I^t denotes a traditional (single-image) importance map at frame t and k denotes the bounded number of neighboring frames. We mitigate the contribution of neighboring frames away from frame t by setting blending factor as $\delta = (\ell - t)/k$. Defining the importance map of an image, I^t, is challenging on its own, requiring scene understand-

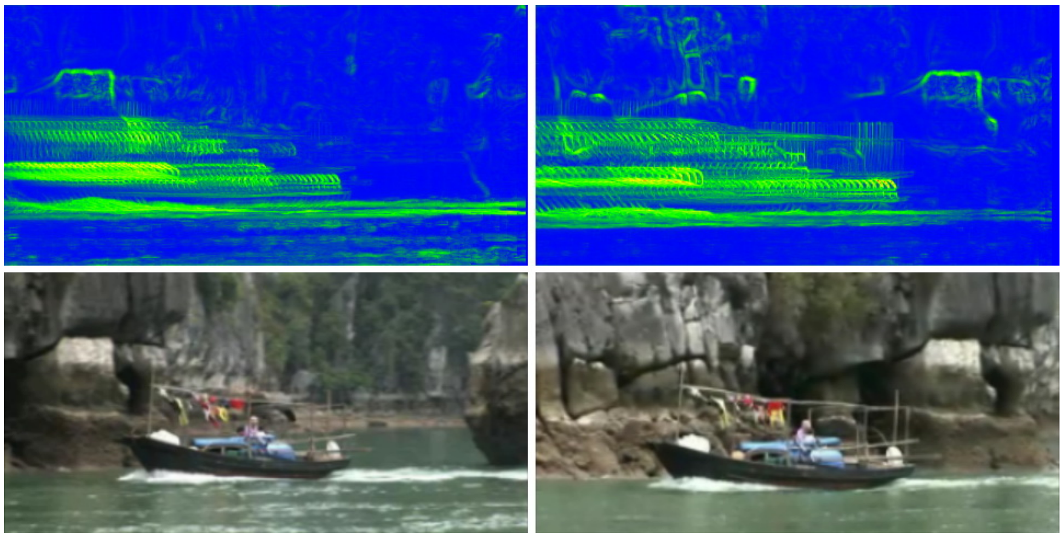

©MAMMOTH HD

Figure 4.2: We blend the information of a bounded number of aligned neighboring frames to define an importance map at each frame. Pixels with high and low importance are visualized in green and blue, respectively. Our model produces time-smoothing maps that capture salient information in both spatial and temporal context.

ing. We adopted the method in Section 3.2.1 to compute I^t as the multiplication of gradient magnitude and image saliency [18], though other information (e.g., from face detection) can be easily incorporated as well.

In Equation 4.1, we chose to take the (weighted) maximum importance among the aligned frames at a given pixel, which guarantees that object motion, usually reflected as moving object boundaries, can be implicitly captured by our importance model. We do not incorporate an explicitly defined motion saliency map here to avoid the problem of weighing and fusing irrelevant saliency cues [37]. Unlike the previous importance models [4,24], which consider motion only between two consecutive frames, our model also captures motion information only observable over a longer time period. For example, our importance maps record objects' motion paths by observing their movement within multiple frames. Figure 4.2 shows two blended importance maps of a video containing simultaneous camera and object motion. Note that the blending process marks some background pixels as important, since their corresponding pixels in the neighboring aligned frames are important due to the motion of the moving boat.

This is a desirable effect, as the blended maps give higher importance values to moving objects and capture salient information in both spatial and temporal context, thus better preserving the aspect ratio of foreground objects.

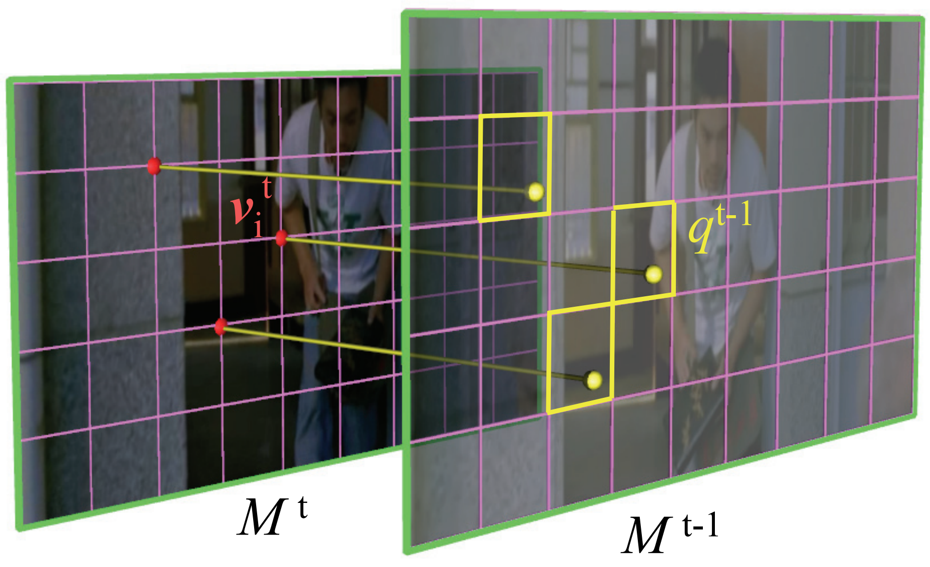
By increasing the value of k, we obtain more time-smoothing importance maps and also capture the motion context better. On the other hand, larger k means that a larger amount of important regions from different frames are combined into a single map, which may lead to a more homogenous map in some scenarios. An extreme example is when each pixel is marked as equally important when multiple objects move around the entire scene within the involved frames, reducing the scheme to homogenous resizing. We have experimented with different values for the blending parameter k. Please see the supplemental video *Kcomparison.mp4* for comparisons. Although an adaptive time window that considers video contents might be more appreciated from a theoretical point of view, we found that setting $k = 60$ works well for all of our experimental examples.

Rubinstein et al. [2] discussed the possibility of using all video frames to compute a single importance map for carving the video with static seams. However, their model works only for videos produced by stationary cameras, since their formulation does not exclude camera motion. As our model processes frames aligned by interframe camera motion, it can successfully handle videos created by dynamic cameras.

4.3 Grid-based Resizing Optimization

We now describe our video resizing framework, which uses the blended importance maps to guide the spatial content preservation of individual frames and motion-aware temporal coherence constraints to preserve both camera and object motion. Since we are always operating on aligned frames, it is more intuitive to formulate the optimization over all frames of a video clip aligned in a common camera coordinate system[1], determined by the first frame of the video clip in our case (Figure 4.1). We drive the

[1]Note that we can equivalently formulate the optimization in the original spatiotemporal coordinate system (i.e., before frame alignment) through coordinate transformation.



©ARS Film Production

Figure 4.3: We preserve camera motion by retaining relative positions of consecutive uniform grids associated with video frames aligned at a common camera coordinate system.

deformation of each aligned frame at time t using an associated uniform grid mesh $M^t = \{\mathbf{V}^t, \mathbf{E}^t, \mathbf{Q}^t\}$ with vertex positions $\mathbf{V}^t$, edges $\mathbf{E}^t$ and quads $\mathbf{Q}^t$. All grid meshes are independent of video content and have the same connectivity but they are usually of different sizes and locations due to frame alignment, leading to a non-cubic shape in the spatiotemporal space (Figure 4.3).

4.3.1 Spatial Content Preservation Energies

We adopt the previously introduced method (Section 3.2.2) to resize individual frames by redistributing the vertices of the associated grid meshes. To deform the grid meshes while respecting the video content, we need to compute an importance value for each quad $q^t \in \mathbf{Q}^t$ of M^t based on $\bar{I}^t(\mathbf{p})$. We define it as the average importance over the pixels in q^t; the importance values are normalized into the range $[0, 1]$.

The energy for preserving the quad aspect ratios according to the normalized quad importance ω_q^t is formulated as

$$D_u = \sum_t \sum_{q^t \in \mathbf{Q}^t} \omega_q^t \sum_{\{i,j\} \in \mathbf{E}(q^t)} \left\|(\tilde{\mathbf{v}}_i^t - \tilde{\mathbf{v}}_j^t) - s_q^t(\mathbf{v}_i^t - \mathbf{v}_j^t)\right\|^2, \tag{4.2}$$

where $\tilde{\mathbf{v}}_*^t$ is the (unknown) deformed vertex position of $\mathbf{v}_*^t \in \mathbf{V}^t$ after resizing, $\mathbf{E}(q^t)$ denotes the edge set of q^t, and s_q^t is the unknown uniform scaling factor of quad q^t, depending on both $\tilde{\mathbf{v}}_*^t$ and $\mathbf{v}_*^t$. We also adopt the energy in Section 3.2.2 which penalizes bending of grid lines and thus alleviates the edge flipping problem:

$$D_e = \sum_t \sum_{\{i,j\} \in \mathbf{E}^t} \left\|(\tilde{\mathbf{v}}_i^t - \tilde{\mathbf{v}}_j^t) - l_{ij}^t(\mathbf{v}_i^t - \mathbf{v}_j^t)\right\|^2 \tag{4.3}$$

with $l_{ij}^t = \|\tilde{\mathbf{v}}_i^t - \tilde{\mathbf{v}}_j^t\| / \|\mathbf{v}_i^t - \mathbf{v}_j^t\|$. Please refer to Section 3.2.2 for more details about these two energies.

4.3.2 Temporal Coherence Energies

Due to the different natures of camera motion and object motion, we design separate constraints (or more precisely, energy terms) to minimize temporally inconsistent distortion. Both types of constraints are equally important to achieve temporally coherent resizing and they are not interchangeable.

Camera Motion Preservation

Interframe transformations naturally reflect camera motion and should be preserved in order to retain it. This can be achieved by preserving the relative positions of consecutive aligned frames, i.e., by asking the positions of corresponding pixels in adjacent frames (aligned in a common camera coordinate system) to be the same after resizing. Since interframe transformations are of low degree of freedom, this can be equivalently achieved by preserving the relative positions of consecutive grid meshes. The above discussion leads us to preserving the relative coordinate of grid vertex $\mathbf{v}_i^t$ (in red) with respect to the corresponding quad q^{t-1} (in yellow) of M^{t-1} that contains the spatial location of $\mathbf{v}_i^t$ (Figure 4.3). Specifically, we use the following energy term:

$$D_\alpha(\mathbf{v}_i^t) = \left\|\tilde{\mathbf{v}}_i^t - \textstyle\sum_{\tilde{\mathbf{v}}_j^{t-1} \in q^{t-1}} \mathrm{a}_j^{t-1} \tilde{\mathbf{v}}_j^{t-1}\right\|^2, \tag{4.4}$$

where a_j^{t-1} denotes the relative coordinate of $\mathbf{v}_i^t$ with respect to q^{t-1} before resizing (we use barycentric coordinates).

Equation 4.4 works only for grid vertices that have correspondence in the previous frame. However, due to frame alignment, there are usually some vertices (near to the grid mesh borders) that fail to find the corresponding positions in the previous frame, denoted as $\mathbf{V}_\beta^t$. Since temporal coherence is required on every local region of the video frames, we need special treatment for the vertices without correspondence. For every such vertex $\mathbf{v}_i^t \in \mathbf{V}_\beta^t$, a naïve solution might be to simply constrain the positions of the pixels that are temporally adjacent *before alignment* to be same after resizing, i.e., by minimizing $\|\tilde{\mathbf{v}}_i^t - \mathbf{T}_{t\mapsto t-1}\ \tilde{\mathbf{v}}_i^{t-1}\|^2$, where $\mathbf{T}_{t\mapsto t-1} = (\mathbf{T}_{t-1\mapsto t})^{-1}$ is needed to offset the influence of frame alignment already encoded in the coordinates of $\mathbf{v}_i^{t-1}$ and $\mathbf{v}_i^t$. However, this naïve solution is undesirable: although the sets of vertices with and without aligned features in the previous frame (i.e., $\mathbf{V}^t \setminus \mathbf{V}_\beta^t$ and $\mathbf{V}_\beta^t$) are disjoint, they are governed by the same set of interframe camera motions. As discussed before, constraining temporally adjacent pixels before alignment always attempts to retain a motion-oblivious interframe transformation (i.e., an identity transformation), which conflicts with the preservation of motion-aware interframe transformations in Equation 4.4.

Instead, we enforce the *deformations* around the vertices that are temporally adjacent before alignment to be as similar as possible. To achieve this, we use the Laplacian coordinates [38], denoted as $L(\mathbf{v}_i^t) = \sum_{\{i,j\}\in\mathbf{E}^t}(\mathbf{v}_i^t - \mathbf{v}_j^t)$, to represent local features, and use $\delta(\tilde{\mathbf{v}}_i^t) = L(\tilde{\mathbf{v}}_i^t) - L(\mathbf{v}_i^t)$ to measure the deformation caused by resizing. Note that the original Laplacian coordinates are always the same at corresponding vertices up to interframe transformations, that is, $L(\mathbf{v}_i^t) \equiv \mathbf{T}_{t\mapsto t-1}\ L(\mathbf{v}_i^{t-1})$. Thus we measure the deformation difference at corresponding positions by comparing the corresponding new Laplacian coordinates

$$\begin{aligned} D_\beta(\mathbf{v}_i^t) &= \left\|\delta(\tilde{\mathbf{v}}_i^t) - \mathbf{T}_{t\mapsto t-1}\ \delta(\tilde{\mathbf{v}}_i^{t-1})\right\|^2 \\ &= \left\|L(\tilde{\mathbf{v}}_i^t) - \mathbf{T}_{t\mapsto t-1}\ L(\tilde{\mathbf{v}}_i^{t-1})\right\|^2. \end{aligned} \tag{4.5}$$

By combining the above criteria, our final energy for preserving camera motion can be

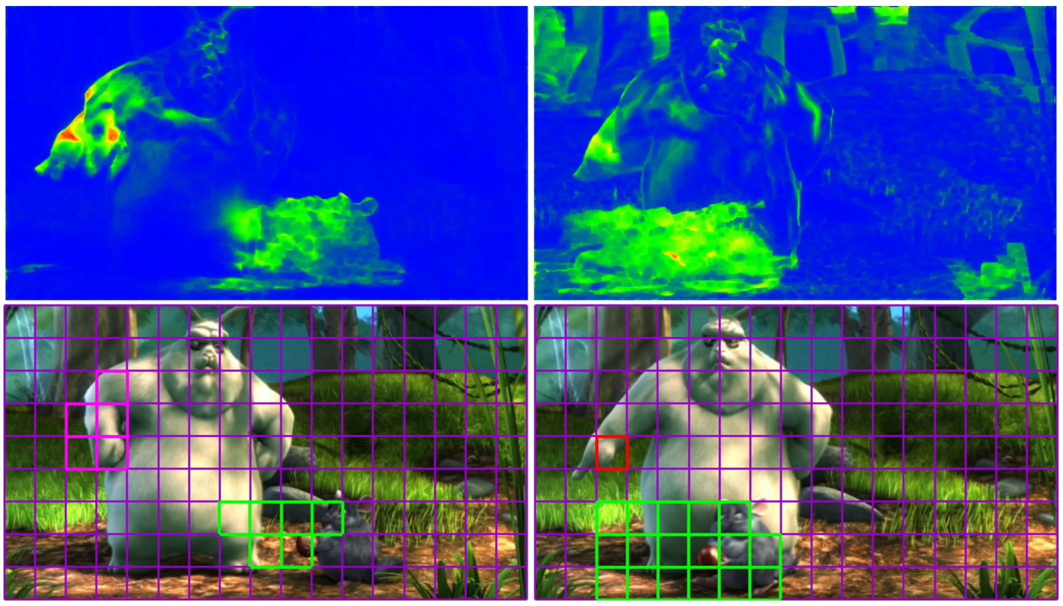


Figure 4.4: We use the motion information in a bounded number of aligned neighboring frames to define a motion saliency map at each frame. We preserve object motion by detecting distinct moving volumes of foreground objects, covered by quads in different colors, and resizing each of them consistently.

formulated as

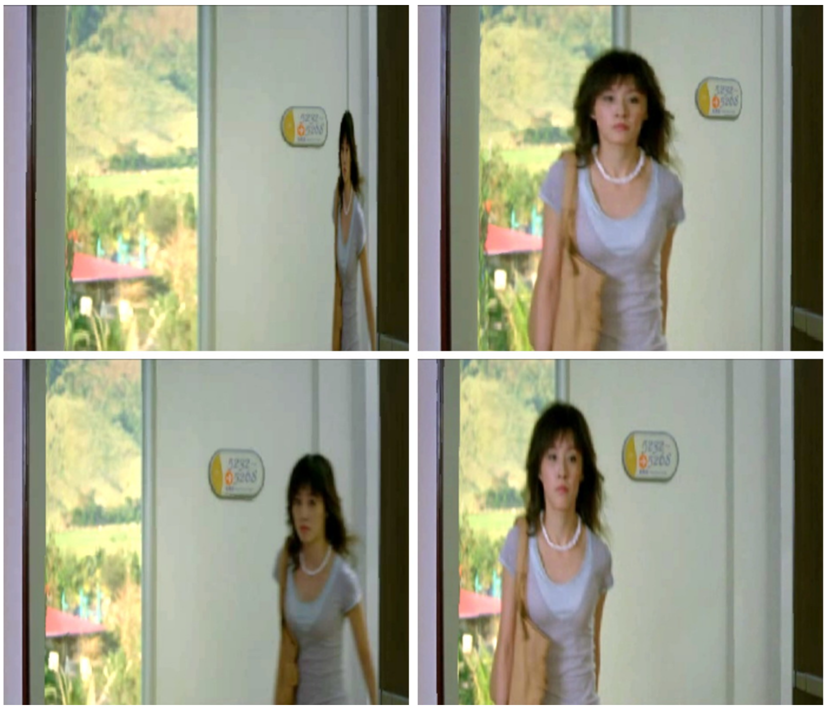
$$D_c = \sum_t \sum_{\mathbf{v}_i^t \in \mathbf{V}^t \setminus \mathbf{V}_\beta^t} D_\alpha(\mathbf{v}_i^t) + \sum_t \sum_{\mathbf{v}_i^t \in \mathbf{V}_\beta^t} D_\beta(\mathbf{v}_i^t). \tag{4.6}$$

Object Motion Preservation

The camera motion constraints above are essentially based on the assumption that the corresponding features across frames are already aligned at the same position, which works well for (static) backgrounds. However, the corresponding features from (dynamic) foreground objects usually have different locations even in the aligned frames (e.g., the moving lady in Figure 4.5), as foreground objects have their own motion which is independent of camera motion. Therefore we need additional constraints to preserve the motion of foreground objects. We observed that relative sizes of dynamic objects are roughly retained during resizing thanks to the smooth warping nature of the regular grid meshes. Thus we only need to consider how to preserve the dynamic motion of an individual object. As dynamic objects usually attract most attention, they should be preserved entirely during resizing. These observations motivated us to detect moving areas of a dynamic object in individual frames and resize all the moving areas associated to the same object in a consistent manner. In other words, our aim is to consistently resize the object's entire moving volume in the spatiotemporal space.

Since the basis of our resizing method is a smooth mesh warp, we do not require a precise segmentation of the object's moving areas. We use a simple technique to estimate the moving volume, though we can always resort to more robust but complex methods such as the one proposed by Kang et al. [39] for video montage. To begin with, we first build an image mosaic [32] as the background scene image of frame t by averaging the aligned pixel colors from frame t to $t + k$ ($k = 60$ in all of our experimental results). We then define a motion saliency map O^t as the L^2-norm of the RGB color difference between the aligned frame t and the background image. Note that we exclude the influence of camera motion from O^t. Since we rely on color variations to detect object motion, our method can handle all kinds of foreground objects as long as they exhibit detectable color variations. To avoid possible interference by pixels with small motions, which might be due to frame misalignment, we only keep pixels $\{\mathbf{p}|O^t(\mathbf{p}) \geq \gamma \max(O^t)\}$, where $\gamma = 0.5$, and detect spatiotemporally connected components of these pixels as the distinct moving volumes (Figure 4.4).

To preserve the consistency of each moving volume, we require all its covering quads to be resized consistently. Let B^u be the set of quads covering a moving volume u. See an example of B^u in Figure 4.4 where for instance all the quads covering the moving volume associated with the mouse are shown in green across frames. Since our sole concern is the final resizing effect (i.e., frames transformed back to the original coordinate system), similar to the design of Equation 4.5, we need to offset the influence of interframe transformations when constraining quads from different frames. Let $q_{u,h}$ be a quad with index h in B^u and $t_{u,h}$ the time coordinate of $q_{u,h}$. Rather than constraining all the quad pairs, which would lead to a much denser system matrix, we found it sufficient to resize all the quads equally to some randomly chosen quad $q_{u,h_0} \in B^u$ up to interframe transformations, where h_0 is a random number. To allow



©ARS Film Production

Figure 4.5: Top: Camera motion constraints alone cannot guarantee consistent resizing of foreground objects. Bottom: Adding object motion constraints leads to temporally more coherent results.

possibly large distortion for moving volumes detected as less important, we constrain the vertical and horizontal edges of quads separately. Specifically, we formulate the energy term for object motion as

$$\begin{aligned} D_o &= \sum_u \sum_{h \neq h_0} D_{o,x}(q_{u,h}) + D_{o,y}(q_{u,h}), \text{ with} \\ D_{o,d}(q_{u,h}) &= \sum_{\{i,j\} \in \mathbf{E}_d(q_{u,h})} \left\| \tilde{\mathbf{e}}_{i,j}^{t_{u,h_0}} - \mathbf{T}_{t_{u,h_0} \mapsto t_{u,h}} \tilde{\mathbf{e}}_{i,j}^{t_{u,h}} \right\|^2, \end{aligned} \tag{4.7}$$

where $\tilde{\mathbf{e}}_{i,j}^t = \tilde{\mathbf{v}}_i^t - \tilde{\mathbf{v}}_j^t$, $d \in \{x, y\}$ and $\mathbf{E}_x(q_{u,h})$ and $\mathbf{E}_y(q_{u,h})$ denote the horizontal and vertical edges of $q_{u,h}$, respectively. Intuitively, minimizing the above energy means resizing the corresponding edges of $q_{u,h}$ and q_{u,h_0} in the same manner (up to their interframe transformation). We are allowed to simply compare the edge vectors because the corresponding edge vectors of all the quads before resizing are the same up to interframe transformations. Figure 4.5 compares the resizing results with and without the object motion energies.

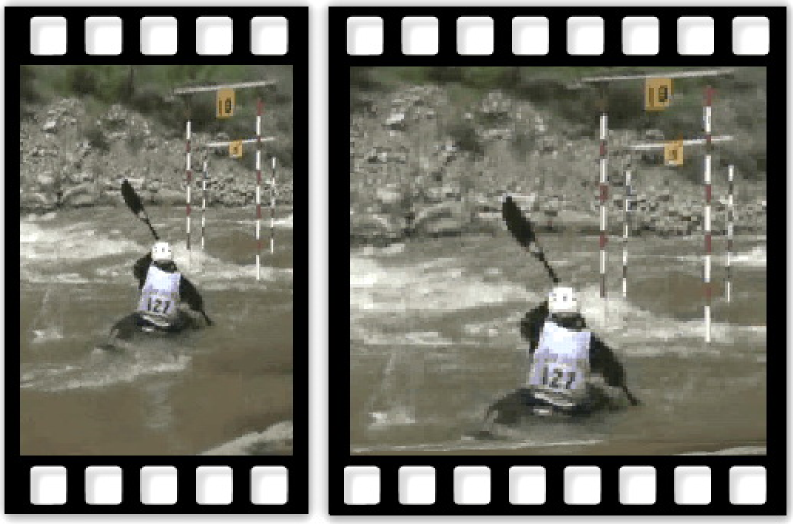


Figure 4.6: An example of video expansion achieved with our method. Left: the original frame. Right: the expanded frame.

4.3.3 Minimization of Energy Functions

By combing spatial and temporal energies, our final optimization is formulated as

$$\underset{\tilde{\mathbf{v}}_i^t}{\operatorname{argmin}} \ \big(D_u + D_e + \lambda\,(D_c + D_o)\big), \tag{4.8}$$

subject to positional, boundary and size constraints. We use the weighting factor λ to balance the spatial and temporal contribution. Since motion artifacts are more noticeable, we use a large value of λ ($\lambda = 10$ in all our experiments). Each energy term is dependent on the sizes of individual frames/meshes, which are often different due to frame alignment. To remove this dependence and revert to the same importance of individual frames before alignment, we divide per-frame formulation in each energy term by the corresponding scaling factor (i.e., the scaling component of $\mathbf{T}_{t\mapsto 0}$). Similar to Section 3.2, we fix the position of the top-left vertex of the first frame and constrain all the boundary vertices of each frame to slide along their respective boundary lines. We incorporate the user-specified resizing factor (S_x, S_y) into the size constraints

$$\tilde{\mathbf{v}}_{n,d}^t - \tilde{\mathbf{v}}_{0,d}^t = S_d(\mathbf{v}_{n,d}^t - \mathbf{v}_{0,d}^t), \ \ \forall t, \ d \in \{x, y\}, \tag{4.9}$$

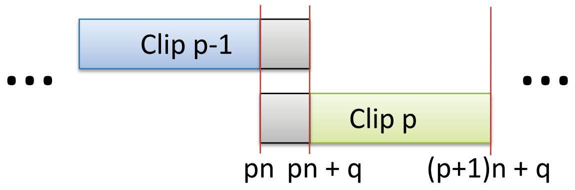
where $\mathbf{v}_0^t$ and $\mathbf{v}_n^t$ are the top left and the bottom right vertices of frame t, respectively.

Our optimization is essentially a nonlinear least-squares problem, with the nonlinearity stemming from D_e. We consider the uniform scaling factor s_q^t and length factors $l_{i,j}^t$ as additional unknowns, and solve for $\{\tilde{\mathbf{v}}_i^t\}$ and $\{s_q^t,\ l_{i,j}^t\}$ iteratively using an alternating method. Please refer to Section 3.2.2 for more technical details. Each alternating iteration involves solving a large sparse linear system, whose system matrix can be pre-factorized. Therefore we only need to perform fast back substitutions for each iteration. Note that the x and y coordinates of the vertices are independent in the objective function, allowing us to solve for them separately.

4.3.4 Scalability

Preserving the temporal coherence at scene/shot boundaries of videos is not necessary since there are no corresponding objects between subsequent frames. Therefore we manually segment input videos into individual scenes and leave the implementation of an automatic scene boundary detection algorithm (e.g., [40]) for future work.

The performance of our optimization (Equation 4.8) depends on both the resolution of the video and the number of frames involved. Adopting multigrid algorithms, as done in [5], would alleviate the performance problem to some extent. Since human beings are often not sensitive to small changes between temporally distinct frames, this motivated us to break a long video (of a single scene) into shorter clips and solve the resizing problem on individual clips sequentially. That is, we constrain the temporal coherence in the overlapping areas in order to achieve coherent resizing of the entire sequence. Since we trade speed for coherence quality, our current implementation can achieve interactive performance (around 5 fps).



To achieve a smooth resizing effect between consecutive clips, we slightly overlap consecutive clips and apply additional temporal coherence constraints to the overlapping frames. Specifically, we divide an input video into multiple clips, with each clip containing $n + q$ frames, where q is the number of overlapping frames.

For example, the p-th clip contains frames from pn to $(p+1)n+q$. We set $n = 100$ and $q = 30$ in our experiments. We resize the first clip ($p = 0$) using the optimization in Equation 4.8. For each resized clip $p-1$ ($p \geq 1$), we directly output its first n frames as the final resized frames and leave its last q frames as constraints to achieve smooth resizing transition to its next clip p.

We achieve temporal coherence between clips $p-1$ and p by minimizing the differences of the corresponding vertex positions between the last q frames of clip $p-1$ and the first q frames of clip p:

$$D_s = \varphi_t \sum_{t=pn}^{pn+q} \sum_{\tilde{\mathbf{v}}_i^t \in \tilde{\mathbf{V}}^t} \left\| \tilde{\mathbf{v}}_i^{t,p} - \tilde{\mathbf{v}}_i^{t,p-1} \right\|^2, \tag{4.10}$$

where $\tilde{\mathbf{v}}_i^{t,p}$ denotes the unknown position of $\tilde{\mathbf{v}}_i^t$ in clip p and $\tilde{\mathbf{v}}_i^{t,p-1}$ the already-solved position of $\tilde{\mathbf{v}}_i^t$ in clip $p-1$. We use φ_t to control the transition speed. We found that a simple linear function $\varphi_t = (pn+q-t)/q$ already works well in our experiments. To resize the whole clip p, we add λD_s as an extra temporal energy term into the objective function in Equation 4.8 and solve the resulting optimization for all the frames of clip p aligned at its first frame $t = pn$. Note that the positional constraint is unnecessary in this scenario since D_s provides an alternative positional specification.

4.4 Results and Discussion

We tested our method on aspect ratio changes of a variety of videos. The chosen videos range from indoor scenes to outdoor scenes, from scenes containing one object movement to those involving multiple moving objects, and from intentional camera motion to unconscious camera shaking. Many of them involve simultaneous camera and object motion, making the task of content-aware resizing rather challenging. Figures 4.1, 4.6 and 4.7 show some of the tested examples under different types of camera and object motions. Although our camera model for frame alignment only contains the translation and scaling parameters, it can handle almost all types of camera motions except camera roll, which seldom happens in video production. As demonstrated in the accompanying videos, our method successfully produces spatiotemporally coher-

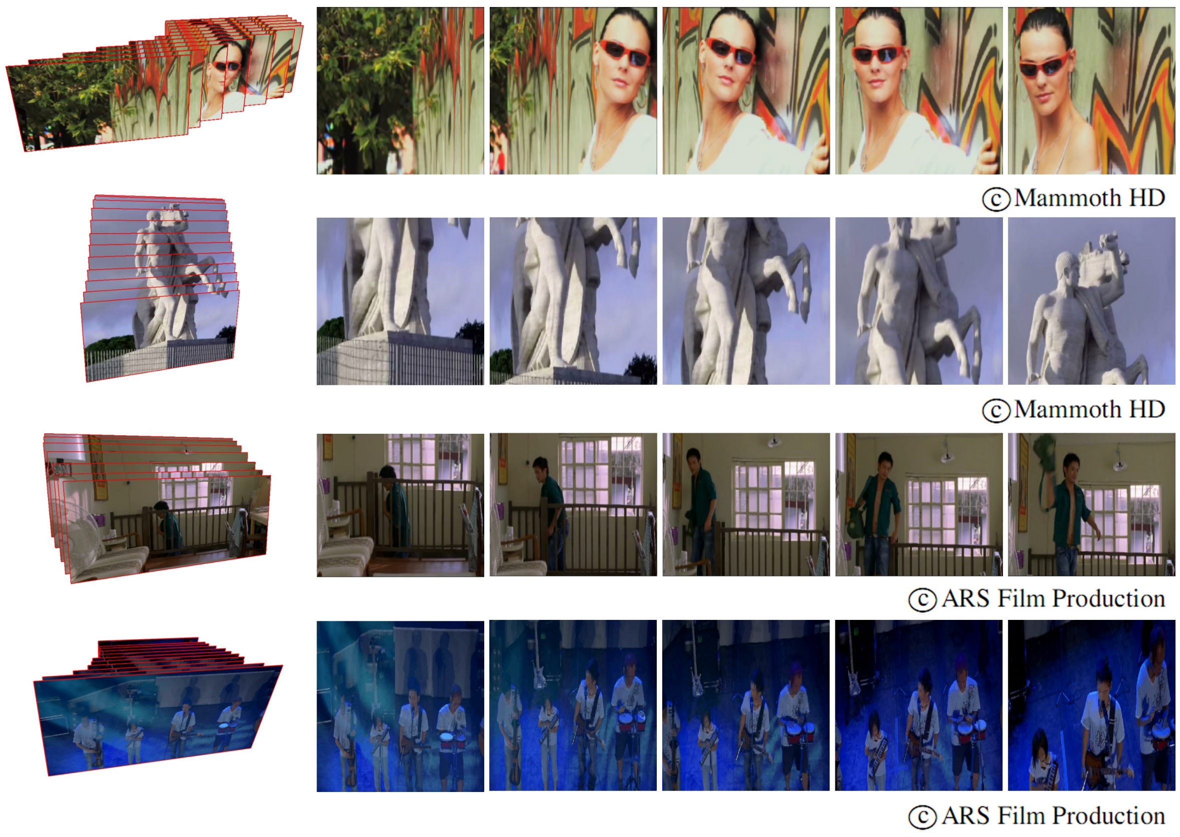


Figure 4.7: Left: frame alignment examples under different types of camera motions, consisting of sliding, yaw, pitch, and zoom motions. Right: the resized key frames. Note that the visually prominent features (e.g., human shapes and window shapes) are well preserved both spatially and temporally.

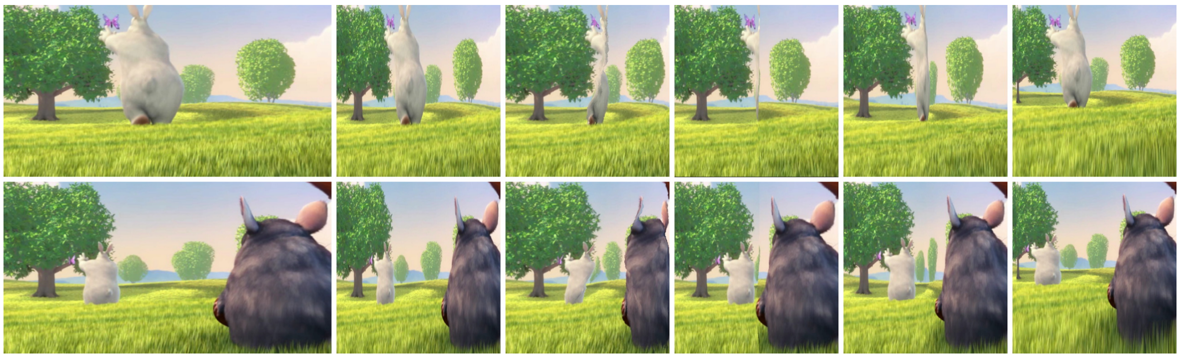
ent resizing results and faithfully preserves visually prominent regions and motions of objects and cameras in most cases.

It is well known that interframe motion estimation incurs some approximation errors, even if 2D projective transformations are used as a more precise camera model [32]. To avoid the increasing accumulation error in long videos, we use only a bounded number of frames to define the importance maps, two consecutive frames to define camera motion constraints, and only the frames involved in individual moving volumes to define object motion constraints. More importantly, we break a long video into short clips, for which the accumulation errors are generally small. As demonstrated in our supplemental video, this strategy preserves the objects' aspect ratios better since the side effect of the alignment error is reduced. Note that our streaming implementation usually does not introduce any noticeable resizing artifacts between consecutive clips, thanks to the blending strategy of importance maps and the smooth transition constraints applied at the clip overlapping areas. We use our streaming method to generate all the resizing examples except those used for comparisons with and without the streaming implementation.

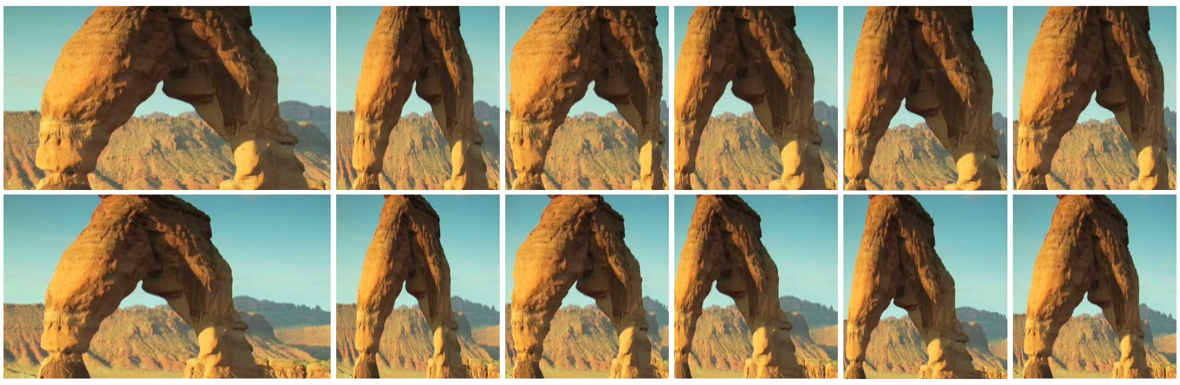
Comparisons

We have compared our resizing results with those produced by homogeneous resizing, one-directional warping (ODW for short) [4] and seam carving (SC) [2]. We have also compared to a naïve extension of Wang et al.'s [41] omnidirectional warping method for video resizing (NDW), in which temporal coherence is enforced by simply constraining temporally adjacent vertices between consecutive frames, similar to Wolf et al.'s constraints. We use our *blended* importance model for both ODW and NDW and keep the original forward energy of SC, since the forward energy considers energy changes caused by seam removal and thus preserves structures better than a backward energy [41]. We have also experimented with an importance model determined from individual frames without blending when comparing to ODW and NDW, but found that it usually produced similar or even worse results (see a comparison example in the supplemental videos). Since we want to compare the effectiveness of these resizing methods for general types of videos, we do not use saliency measures designed for certain special types of objects, e.g., faces.

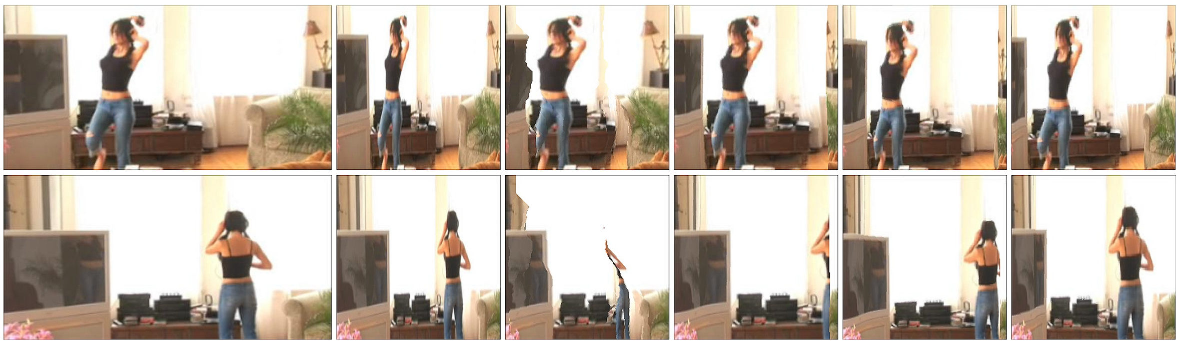
In Figure 4.8 we chose to show three representative comparison scenarios involving (multiple) object motion only, camera (zoom) motion only, and simultaneous object and camera motion, respectively. Please refer to the accompanying videos for more comparison examples. Obviously, homogeneous resizing always achieves the best temporal coherence but at the cost of introducing the most serious distortion into important content. The previous content-aware methods preserve important content better but exhibit different kinds of artifacts due to their motion-oblivious nature. By the discrete nature of SC, it causes high-frequency artifacts in both the spatial and temporal domains, exhibiting "jaggies" and flickering. ODW and NDW lead to smooth waving



©Blender Foundation



©Mammoth HD



©Mammoth HD

Figure 4.8: From left to right columns: the original frame images, resizing results with homogeneous resizing, [2], [4], the naïve extension of [41], and our method. Clearly, only our method can well preserve the visually prominent features while successfully retaining temporal coherence. Due to the motion-oblivious temporal coherence constraints, the previous content-aware resizing methods often cause inconsistent alteration of corresponding features across frames, e.g., the white bunny in the first example, the arch in the second example and the woman's body in the third example.

artifacts spatially and temporally, since they distribute resizing distortion across the whole image of each frame in a least-squares manner. We observed that low-frequency artifacts caused by ODW or NDW are generally less noticeable than high-frequency artifacts by SC. We also found that due to its edge flipping constraints, NDW often produces less fold-over artifacts than ODW, noticeable in the areas of human body of the sixth row. However, the waving artifacts by NDW occurring in structural or high-contrast regions are still visually noticeable. Although the previous methods do not exhibit very serious spatial artifacts in the above examples, they cause a much more serious problem of temporal incoherence, as shown in the accompanying videos. On the other hand, our method consistently achieves spatiotemporally coherent resizing of these videos. For some complex examples such as the third one, achieving both perfect spatial content preservation and perfect temporal coherence is extremely hard. For those scenarios, our method still achieves better spatial content preservation than homogeneous resizing and better temporal coherence than the previous content-aware methods. Thus we believe that our method strikes a good balance even in complex situations. In short, compared to previous work, our method achieves comparable results for trivial cases and visually better results for challenging examples that involve large camera and/or object movements.

We show the effects of individual components of our algorithm by comparing the resizing results with and without certain components. For example, we demonstrate the pure impact of blending the aligned importance maps by comparing the results with and without using our importance model for ODW (see the accompanying video *MapComparison.mp4*). The comparisons in Figure 4.5 show the significance of object motion constraints. We demonstrate the pure effect of the criteria for preserving both camera and object motions by comparing our method with NDW, since the same importance maps are used.

Performance

We use uniform grid meshes to drive the deformation of individual frames. Clearly, denser meshes allow more effective distribution of resizing distortion and thus produce better results, at the cost of longer computation and larger memory consumption. Fortunately, we found that rather coarse meshes are often sufficient to achieve satisfactory results. In our experiments, we always use similar mesh resolutions (each quad roughly covers 20×20 pixels), though we could use even coarser meshes without sacrificing resizing quality for some of the tested videos. We associate a grid mesh with each frame rather than introducing a mesh for several frames, since otherwise motion artifacts are more noticeable.

Our resizing method solves the nonlinear optimization problem efficiently by prefactorizing the system matrices and performing fast back substitution at each iteration. The streaming implementation makes our method scalable to long video sequences. Please refer to the accompanying video for some resizing results of long videos. In our experiments, it usually takes less than 200 iterations for convergence in the first clip and less than 100 iterations for the remaining clips, since the resized overlapping frames from a previous clip already provide a good initial guess for resizing the following clip. For the first example in Figure 4.7, whose resolution is 480×240, our unoptimized implementation takes 20 seconds to resize the first 100 frames (around 5 fps on average), with the memory usage of 180Mb, measured on a PC with Duo CPU 2.33GHz. Of that time, 5 seconds are spent on the factorization of the system matrices of both x and y coordinates and 15 seconds on the 105 alternating iterations. We believe that introducing a GPU based multi-grid solver would further improve the performance of our system, possibly allowing real-time resizing.

Limitations

We model camera motion using a 2D camera model, which assumes that the world is a single plane, or the camera rotates around its optical center [32]. Since our model ignores the parallax effect, where the image displacements of scene points should be dependent on their distances from the camera, it might cause misalignment for scene points of varying-depth backgrounds (see an example in the supplemental video *SIFT-Features.mp4*). Fortunately, the smooth warping nature of our system tolerates some degrees of error from misalignment. We observed that the alignment deviation usually causes only local waving artifacts, i.e., makes some static scene points move locally. Compared to global waving artifacts by ODW and NDW or flickering artifacts by SC, local waving artifacts are much less noticeable, as demonstrated in the supplemental comparison video *LocalMisalignment.mp4*. In the future it would be interesting to see if a more sophisticated camera or object motion detection technique could improve the resizing results further. Resizing of videos containing depth information, possibly from stereo cameras, is another interesting topic to explore.

Our feature-based frame alignment method may become unreliable when either too few features are detected (e.g., due to homogeneous backgrounds) or the detected feature correspondences disagree on the implied camera transformation (e.g., due to dynamic backgrounds or large foregrounds occupying the scene). In these scenarios, we replace the unreliable interframe transformation with a linear blending of neighboring reliable transformations (see the supplemental video *AlignmentError.mp4*). In the extreme case where there are too many unreliable interframe transformations, we apply an identity transformation to every frame of the video and lose the temporal preservation of background contents.

Like the other video resizing methods [2,4,5], our method would degenerate to homogeneous resizing if the importance map is nearly homogeneous. This may be due to failure of the saliency measure, too large areas of static objects detected as important, or too many moving parts spreading over the scene (see the supplemental video: *LinearScaling.mp4*). In addition, our blended importance map further relies on the precision of frame alignment since the homogeneous pixels might be erroneously marked as important from neighboring frames due to frame misalignment. This problem usually occurs at scenes with highly varying depths or very long videos. In this case, an intuitive user interface would be useful for users to guide the object correspondence and saliency measure, achieving better resizing results [42].

Chapter 5
Application: Focus+Context Visualization with Distortion Minimization

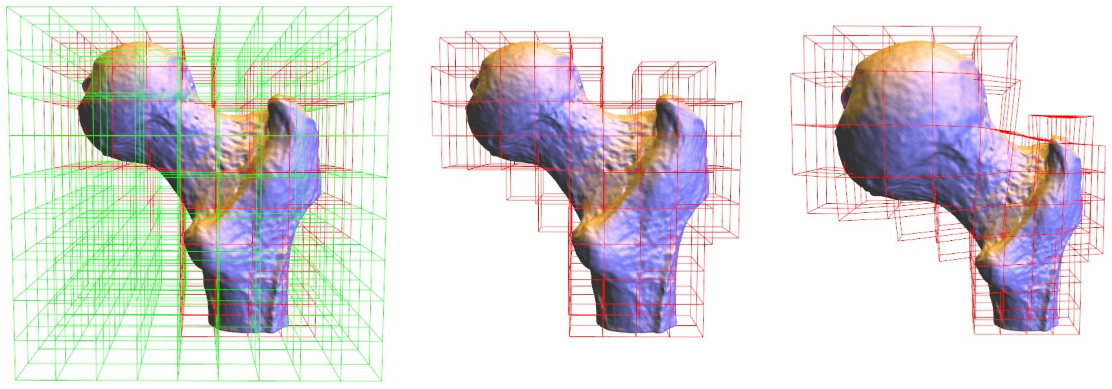


Figure 5.1: (left) The original model partitioned with a uniform grid space. Since solving for those deformed cubes that do not contain any model information (in green) provides zero contribution to the magnification result, we omit their energy terms from our optimization system (middle). That is, we solve only for the deformed grid vertices of the cubes occupied by the model (in red) and reconstruct the embedded model by space interpolation to achieve magnification visualization (right).

5.1 Overview

We propose a framework to interactively deform a polyhedral model to achieve Focus+Context visualization. Our goal is to non-homogeneously rescale different regions while preventing the global bounding space of the model from being expanded (see Figure 5.1) and thus keeping the entire model displayed on the screen. We construct a uniform grid space for a given model such that the model vertices are embedded in the grid cubes. While enlarging the cubes covering the user-specified focal region, our system automatically reduces the other cubes to keep the entire model within the global bounding space. The deformed model is reconstructed by computing each model

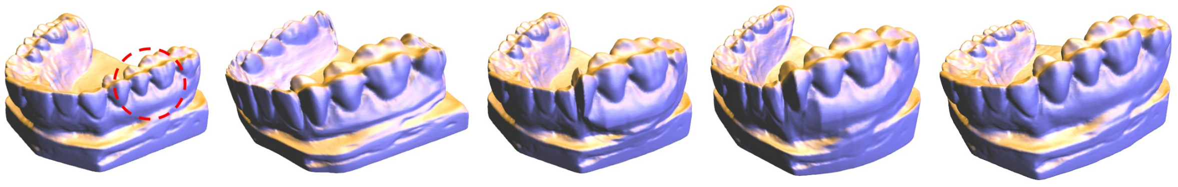


Figure 5.2: (left) Original model. The focal region (inside the red dotted circle) is magnified to observe in detail the teeth model using the stretch orthogonal method [9,10] (middle left), radial Gaussian distortion [9,10] with faster (middle) and slower (middle right) fall-off and our method (right). Note that, with the stretch orthogonal method and the radial Gaussian, the surroundings of the focal region are seriously stretched due to the distortion being uniformly distributed on the model; wider Gaussian fall-off only distributes the distortion to a larger region. In contrast, our method minimizes distortion and preserves the shape of the local feature such that each tooth remains similar to its original shape.

vertex as a linear combination of its respective set of cube vertices in the deformed grid space. Specifically, we wish to scale each local region of the model but keep it similar to its original appearance. The aim is to preserve the aspect ratio of each local region, avoiding squeezing or squashing which would result in distortion. We design an optimization procedure that allows cubes covering the model to undergo uniform scales, while letting empty cubes to be stretched to absorb the resulting distortion. Since the transformations of connected cubes are not identical, the deformation inevitably causes distortion. To minimize this resulting distortion, we propose a set of energy terms to form an optimization system and solve for the grid vertex positions of the deformed space in a least-squares sense.

We resize each local region of the given model by an approximate uniform scale such that the locally magnified model resembles the original shape, except that their local sizes are different. Furthermore, our method accomplishes Focus+Context visualization with the regions close to the focal region automatically expanded to reduce the distortion. This is a desirable property since these surrounding areas are likely to be the next focal region during interactive visualization and thus are important as well. Overall, our algorithm can visualize Focus+Context information on the screen while keeping the distortion under control and making it unnoticeable (see Figure 5.2).

5.2 Focus+Context Visualization

Given an input model, we first rescale the model to be of unit size. To ensure that the entire model remains displayed on the screen, the given model should be placed at a proper position such that the eight vertices of its $1 \times 1 \times 1$ bounding box are all projected within the screen. Then, we partition the model using a n^3 uniform grid space, $\mathbf{G} = \{\mathbf{V}, \mathbf{E}, \mathbf{C}\}$, with vertices $\mathbf{V}$, edges $\mathbf{E}$, and cubes $\mathbf{C}$, where $\mathbf{V} = [\mathbf{v}_0^T, \mathbf{v}_1^T, ...\mathbf{v}_{m-1}^T]^T$, $m = (n+1)^3$, and $\mathbf{v}_i \in \Re^3$ denotes the position of vertex i. In other words, there are n^3 identical cubes inside the bounding space.

Using our 2D graphical interface, the user moves a magnifying lens represented by a red dotted circle to specify a *focal region* to magnify. The user also specifies a parameter λ as the magnification factor of the focal region. We refer to the cubes covering the focal region (i.e., those whose centroids are projected within the red dotted circle) as the *focal cubes*. The focal cubes are expanded according to the parameter λ, and the remaining cubes are automatically expanded or reduced to obtain a deformed grid space $\mathbf{G}'$ while maintaining the global bounding size. Our system solves for the grid vertex positions by minimizing a set of energy functions retaining the aspect ratios of the cubes covering the given model as well as constraining the grid vertices to be within the bounding space. After deforming the embedding space, the model is reconstructed by computing each vertex position according to its mean value coordinates within the grid space [43]. Since the global bounding space remains unchanged during the magnification of the focal region (see Figure 5.3), and the distance between the model and the camera also remains the same, our system accomplishes the aim of Focus+Context visualization.

5.2.1 Space Deformation

We compute the deformed grid space by determining the scaling transformation of each cube. Clearly, since the cubes are connected and are not scaled by the same factor, it is impossible to magnify a specific region without causing distortion to other regions. Therefore we strive to minimize the deviation of each local transformation from a uniform scale to keep the resulting distortion under control and unnoticeable. To satisfy the above requirements, we propose several energy functions and formulate an optimization system to compute the deformed grid space.

Individual Cube Rescaling

Given a cube $\mathbf{c}_k$, we compute its deformed version $\mathbf{c}'_k = \mathbf{s}_k\mathbf{c}_k$, where $\mathbf{s}_k$ is a 3×3 uniform scaling matrix. To solve for the entire grid space in a least-squares sense, we integrate the equations into a linear system

$$\|\mathbf{C}' - \mathbf{S}\mathbf{C}\|^2 = 0, \quad where$$
$$\mathbf{S}_{uv} = \begin{cases} \mathbf{s}_k & if \quad u = v \\ 0 & otherwise \end{cases}, \quad and\, \mathbf{C} = [c_0, c_1, ..., c_{n^3-1}]^T. \tag{5.1}$$

Rather than uniformly spreading the distortion over the grid space, we float distortion to the cubes that are not occupied by the model because the distortion of those cubes does not influence the magnification result. Thus, we classify the cubes into two groups: the *principle cubes* which cover the input model $\mathbf{C}_P \subset \mathbf{C}$ and the *trivial cubes* $\mathbf{C}_T \subset \mathbf{C}$ which do not. Our goal is to prevent the principle cubes from being squeezed. While solving for the deformed grid space by minimizing the objective function, our algorithm gives higher penalties to the principle cubes $\mathbf{C}_P$ if their transformations deviate from uniform scales so as to better preserve their aspect ratios. The trivial cubes $\mathbf{C}_T$ have zero penalties, sacrificing the uniformity of scaling transformation to absorb the distortion. To implement this idea, we rewrite Equation 1 into the form:

$$\|\mathbf{C}'_P - \mathbf{S}_P\mathbf{C}_P\|^2 + \gamma\|\mathbf{C}'_T - \mathbf{S}_T\mathbf{C}_T\|^2 = 0. \tag{5.2}$$

By setting $\gamma = 0$ since the distortion on the trivial cubes C_T does not influence the model's shape, our system computes only the principle cubes C_P by solving the following simplified equation:

$$\|\mathbf{C}'_P - \mathbf{S}_P\mathbf{C}_P\|^2 = 0. \tag{5.3}$$

Note that a cube is formed by a set of vertices and edges. That is, if a cube is uniformly resized, then its 12 edges would be uniformly expanded or contracted. While deforming

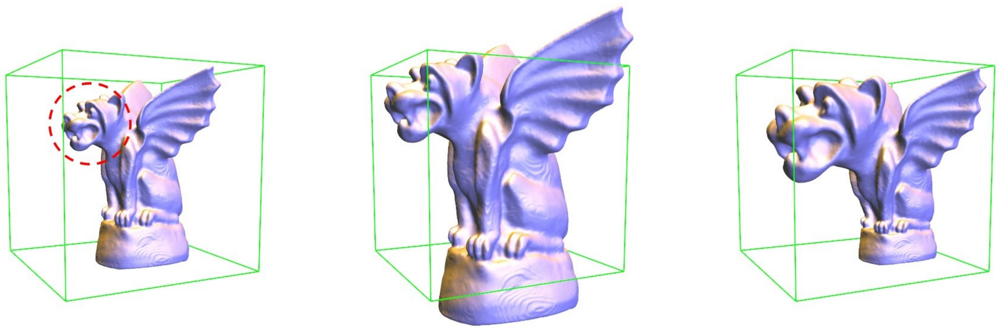


Figure 5.3: (left) Original model and its global bounding space. While magnifying the gargoyle's head without imposing inequality constraints, the entire model is uniformly expanded and some regions may go out of the bounding space (middle). We constrain the outer grid vertices to be located on the bounding surface (right), and thus ensure that the deformed model is always displayed on screen in its entirety.

the grid space, our algorithm retains the connectivity of the original grid structure and only moves the grid vertices to scale individual cubes to different sizes. Let $\mathbf{e}_p = \{i, j\}$ be the p^{th} edge of the cube $\mathbf{c}_k$ and $\epsilon_p = \mathbf{v}_i - \mathbf{v}_j$, we can represent the cube as $\mathbf{c}_k^T = \mathbf{q}_k \mathbf{V}$, where

$$\mathbf{q}_{k,uv} = \begin{cases} 1 & if \quad u = p,\ v = i \\ -1 & if \quad u = p,\ v = j \\ 0 & otherwise \end{cases} , \qquad \mathbf{c}_k = [\epsilon_0^T, \epsilon_1^T, ..., \epsilon_{11}^T].$$

Putting the equations together, we can denote $\mathbf{Q} = [q_0^T, q_1^T, ..., q_{n^3-1}^T]^T$ and replace the matrix $\mathbf{C}$ by $\mathbf{QV}$. Thus, Equation 3 can be reformulated as

$$\|\mathbf{Q}_P \mathbf{V}'_P - \mathbf{S}_P \mathbf{Q}_P \mathbf{V}_P\|^2 = 0. \tag{5.4}$$

where $\mathbf{V}_P$ denotes the vertices of the principle cubes.

Positional Constraints

To solve Equation 4, we need at least one absolute position to locate the deformed grid space since the equation only expresses relations among vertex positions. However, constraining only one vertex at a specific position may move the model when the focal region is magnified. This is because the optimization may transform individual cubes

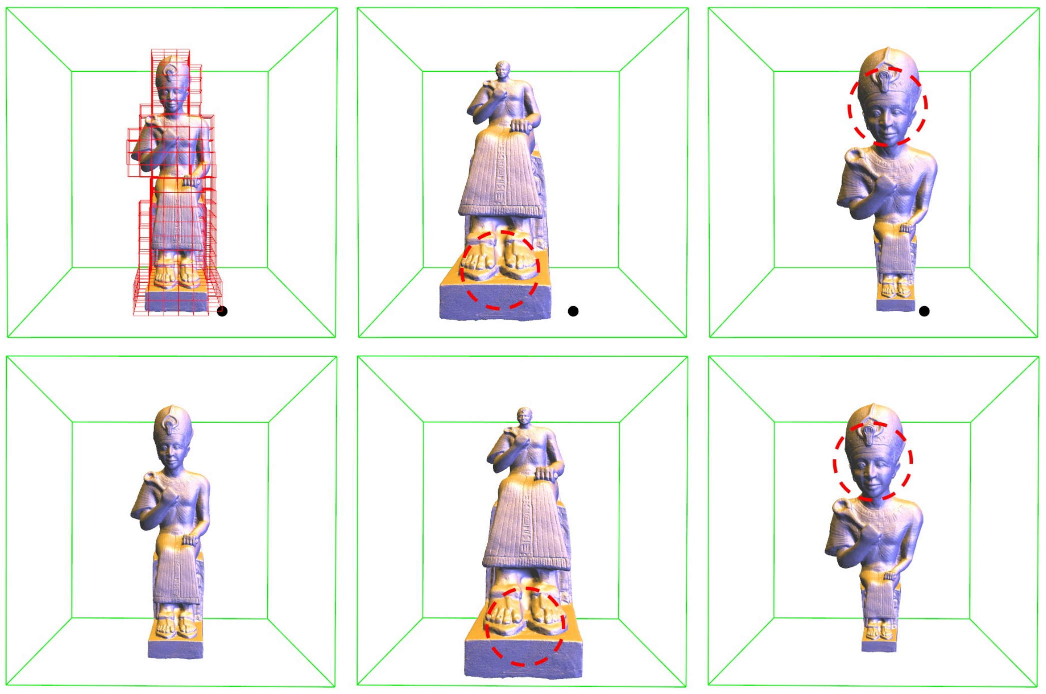


Figure 5.4: (top row) Only one position constraint is applied at the bottom right of the Ramesses model (black point). While magnifying his feet (top middle) and head (top right), his body would move to the right or left as our algorithm tries to minimize distortion. (bottom row) All the grid vertices are slightly constrained to their original positions. The model's position is now more stable after local magnification.

to different positions to reduce their distortion. Figure 5.4 demonstrates this effect. To enhance the stability of the visualization, we want all the cubes to be close to their original positions. Specifically, we constrain each grid vertex to be at its original position with a small weighting factor and introduce the following energy term:

$$\omega \|\mathbf{I}\mathbf{V}'_P - \mathbf{V}_P\|^2 = 0. \tag{5.5}$$

We set $\omega = 0.001$ in our experiments to retain the overall model's position. Larger ω will lead to immovable grid vertices and failure to deform the grid space.

Non-Linear Constrained Optimization

We solve for the deformed vertex positions by minimizing the integration of the above energy terms

$$arg \min_{\mathbf{V}'_P, \mathbf{S}_P} \quad \|\mathbf{Q}_P \mathbf{V}'_P - \mathbf{S}_P \mathbf{Q}_P \mathbf{V}_P\|^2 + \omega \|\mathbf{I}\mathbf{V}'_P - \mathbf{V}_P\|^2, \tag{5.6}$$

which can be formulated as an over-determined linear system $\mathbf{A}\mathbf{V}'_P = \mathbf{b}(\mathbf{V}_P)$, where $\mathbf{A} = [\mathbf{Q}_P^T, \omega I^T]^T$ and $\mathbf{b}(\mathbf{V}_P) = [\mathbf{S}_P^T \mathbf{Q}_P^T \mathbf{V}_P^T, \mathbf{V}_P^T]^T$. To prevent the global bounding space from expanding, we add inequality constraints to the objective function such that the grid vertices move only within the bounding space. That is, we minimize

$$\begin{gathered} arg \min_{\mathbf{V}'_P} \quad \|\mathbf{A}\mathbf{V}'_P - \mathbf{b}(\mathbf{V}_P)\|^2, \quad \text{subject to} \\ x_l \leq \mathbf{v}_x \leq x_u, \quad y_l \leq \mathbf{v}_y \leq y_u, \quad and \quad z_l \leq \mathbf{v}_z \leq z_u, \end{gathered} \tag{5.7}$$

where x_l, x_u, y_l, y_u, z_l, and z_u are the lower and upper bounds of x, y, and z coordinate, respectively. Since $\mathbf{A}$ is not a positive definite matrix, we multiply the equation by $\mathbf{A}^T$ and solve the system $(\mathbf{A}^T\mathbf{A})\mathbf{V}'_P = \mathbf{A}^T\mathbf{b}(\mathbf{V}_P)$ to obtain the unknown variables $\mathbf{V}'_P$. Note that the uniform scaling transformations of the cubes are still unknown when we determine the deformed grid space. Therefore, we can only solve this non-linear optimization problem by iteratively updating the vertex positions [44].

The scaling factor for the focal cubes is obtained from the input parameter λ. For the remaining (non-focal principle) cubes, we compute their scaling transformations according to their deformed (M'_k) and original (M_k) volumes. Specifically, $\mathbf{s}_k = (M'_k/M_k)^{1/3}\mathbf{I}$. In the beginning, we set $\mathbf{V}_P^0 = \mathbf{V}_P$ to start the iteration. We then compute the scaling transformations for the non-focal cubes from $\mathbf{V}_P^t$, and use the obtained $\mathbf{b}(\mathbf{V}_P^t)$ to solve for the new vertex positions $\mathbf{V}_P^{t+1}$. Although the scaling transformation obtained from $\mathbf{V}_P^0$ is merely an identity matrix, the least-squares solver will still reduce the volumes of some cubes so as to expand the focal region. However, the set of vertex positions computed in the first iteration is not the optimum solution of our proposed objective function because the applied scales are only determined by the initial guess. Therefore, we repeat the process to update the vertex positions until the solution converges.

While iteratively updating the grid vertex positions, we detect whether any vertex violates the inequality constraints. Since only a few vertices might be transformed outside the bounding space, not all the inequality constraints have influence on the optimization problem. Therefore, we consider an inequality constraint $f_x(\mathbf{v})$ as

$$\begin{array}{lll} \textit{active} & if & \mathbf{v}_x > x_u \quad or \quad \mathbf{v}_x < x_l \\ \textit{inactive} & if & x_l \leq \mathbf{v}_x \leq x_u \end{array} .$$

Similarly, $f_y(\mathbf{v})$ and $f_z(\mathbf{v})$ are inactive if $\mathbf{v}_y$ and $\mathbf{v}_z$ satisfy their respective constraints. In the beginning, we consider all the inequality constraints as inactive since all the grid vertices are located within the bounding space. After computing the new vertex positions in each iteration, our algorithm detects if any inequality constraint has become active, and if so, restricts that vertex to be sliding on the space boundary in the next iteration. To combine the inequality constraints with our proposed objective function, these constraints are transformed into energy terms [45] (for example, $(\mathbf{v}_x - x_l)^2 = 0$ if $\mathbf{v}_x < x_l$) and are added into the linear system if they are active. Note that the active constraints of x, y, and z coordinates are different; therefore, the three coordinates of the grid vertices should be solved separately. Let $\mathbf{F}_x = \{f_{x,0}, f_{x,1}, f_{x,2}, ...\}$ be the index set of active inequality constraints of the x coordinate. We can solve the linear system $\mathbf{A}_x \mathbf{V}'_{P,x} = \mathbf{b}_x(\mathbf{V}_P)$ to obtain the x coordinates of the grid vertices, where $\mathbf{A}_x = [\mathbf{Q}_P^T, \omega \mathbf{I}^T, \mathbf{R}_x^T]^T$, $\mathbf{b}_x(\mathbf{V}_P) = [\mathbf{S}_P^T \mathbf{Q}_P^T \mathbf{V}_{P,x}^T, \mathbf{V}_{P,x}^T, \mathbf{H}_x]^T$,

$$\mathbf{R}_{x,uv} = \begin{cases} \delta & if \ v = f_{x,u} \in \mathbf{F}_x \\ 0 & otherwise \end{cases}, \quad \mathbf{H}_x = \begin{cases} \delta x_l & if \quad \mathbf{v}_x < x_l \\ \delta x_u & if \quad \mathbf{v}_x > x_u \end{cases},$$

and δ is a large number to enforce the soft constraint (we set $\delta = 100$ in our experiments). The other two coordinates of the vertices can be determined in the same manner. Since the number of equations changes whenever any inequality constraint becomes active, causing the size and the structure of the linear system to change as well, our system needs to re-factorize the matrix $\mathbf{A}_x$ (or $\mathbf{A}_y$, $\mathbf{A}_z$) to compute the new vertex set. Fortunately, the factorization only takes place when an inequality constraint is activated or deactivated, which occurs very infrequently because only the boundary vertices of the principle cubes might violate the inequality constraints. Although our algorithm deforms the grid space in an iterative manner, the computation is still efficient because the factorization of the linear system is not necessary in every step. In

Model	Mesh vertex number	Grid vertex number	Factorization	Back substitution	Recons- truction
bonefoot	12612	605	0.032	0.001	0.002
ball joint	137062	343	0.021	0.001	0.023
Goddess	523578	565	0.106	0.001	0.087
thorax	99920	973	0.126	0.001	0.016
skull	63264	2714	0.145	0.001	0.011
Ramesses	826266	920	0.151	0.001	0.144
colon	44500	1351	0.101	0.001	0.008
hip	132538	1470	0.176	0.001	0.024
gargoyle	100002	1644	0.134	0.001	0.018
teeth	116604	1851	0.175	0.001	0.020
dancing children	100000	1697	0.131	0.001	0.018
Chinese dragon	437645	1480	0.177	0.001	0.076

Table 5.1: The second and third columns show the model information and the last three columns show the timing statistic (in seconds). The computation cost of reconstructing the model depends on the model's vertex number while the computation cost of grid space deformation (i.e., factorization and back-substitution) depends on the grid's vertex number.

most steps, we need only to apply back substitutions to compute the new vertex set and thus the grid space can be deformed in real time.

For grid vertices whose inequality constraints are active, we examine their positions to decide when to deactivate them. Since our system solves for the deformed grid space using soft constraints, the vertices with active inequality constraints are not exactly located on the grid space boundary, rather they may be slightly inside or outside the boundary. Specifically, the activated inequality constraint of an outside vertex pulls the vertex close to the space boundary, but the vertex remains outside. Then, subsequent movement of the focal region may cause the vertex to move in. When this happens, its inequality constraint can be deactivated.

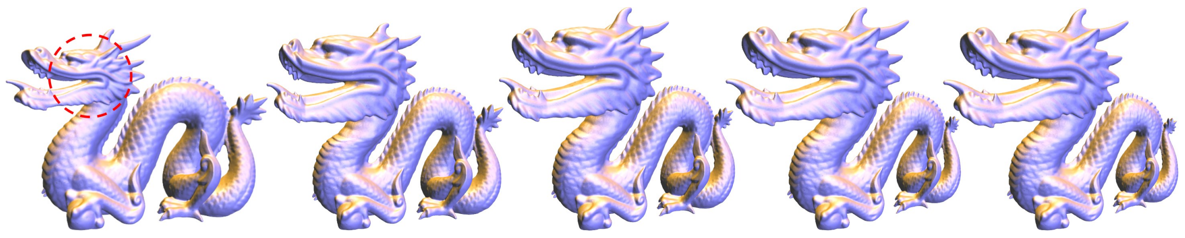


Figure 5.5: We magnify the dragon's head (in the red dotted circle) to achieve Focus+Context visualization. From left to right are the original model and the locally magnified models computed by our algorithm after 1, 5, 10, 30 iterations. Notice that the surrounding region of the dragon's head is more distorted in the beginning but the distortion is rapidly reduced in subsequent iterations. The results obtained in 10 and 30 iterations are very similar because the iterative solver has converged.

Initial Guess

Solving a nonlinear optimization problem always needs a starting position, which is commonly called the initial guess. Obviously, an initial guess that is close to the optimum solution would lead to faster convergence. Choosing a good starting point is therefore an important issue. Although it is always possible to start deforming the grid space from its uniform shape, the extra iterations needed will increase the computation cost and thus slow down the interactive rate. Figure 5.5 shows an example to demonstrate the initial guess and the subsequent iterations. Here, we assume that the cursor movement of the user while specifying the focal region is continuous and the algorithm starts the iteration from the previous frame because the magnification results should be similar if their magnified regions are close to each other. In the beginning, we set the original grid space as the initial guess since there is no previous frame. Our system repeats the updating process until the movement of each grid vertex is less than 0.001, which takes 8 iterations on average to achieve the minimum solution.

5.2.2 Model Reconstruction

We reconstruct the given model by space interpolation after the grid space is deformed. Since each model vertex is embedded in a local cube, we can determine its new position by a combination of their respective 8 cube vertices. Let $\mathbf{u}$ be a vertex of the given model, and $\mathbf{v}_0 \sim \mathbf{v}_7$ be its surrounding cube vertices. The model vertex $\mathbf{u}$ can be represented as $\sum_{i=0}^{7} w_i \mathbf{v}_i$, where w_i is the weight determined by the mean value coordinates [43]. We compute the weighting factor w_i in the pre-computation step and use these values to reconstruct the model each time the bounding space is deformed since the w_i's remain the same.

5.3 Results and Discussion

We have implemented our algorithm on an Intel Core2 2.33 GHz PC with 2 Gb RAM. The linear system is solved with Cholesky Factorization provided by the Taucs library [46]. We partition each input model by a 20^3 grid space, except the ball joint model (Figure 5.1) by 10^3. Clearly, partitioning the model with a finer grid space (especially for complex models) will lead to better results but increase the computation cost. To balance between the quality and interactive rate, we found that a 20^3 grid space is a good compromise. Table 1 shows the timing statistic and the model information. The slowest part of our algorithm is the matrix factorization, which mainly depends on the number of grid cubes. The reconstruction of the given model is efficient because computing each vertex only requires a linear combination of the surrounding cube vertices. For example, the Ramesses model in Figure 5.4 contains 826,266 vertices, which requires only 0.144 seconds to determine the positions of all the model vertices.

Our system can provide Focus+Context information to the user, allowing the user to view detailed content of the focal region while maintaining the overall perception of the model. The results shown in Figure 5.8 demonstrate the effectiveness of our technique. For each input model, we expand different focal regions specified with a dotted red circle and the model is deformed smoothly to keep the deformed shape within the global bounding box. In addition, due to the distortion minimization in our formulation, our system not only retains the aspect ratios of non-focal regions but also results in the smooth magnification of the surroundings of the focal region. This means the user needs not carefully specifies the shape and size of the magnifying lens

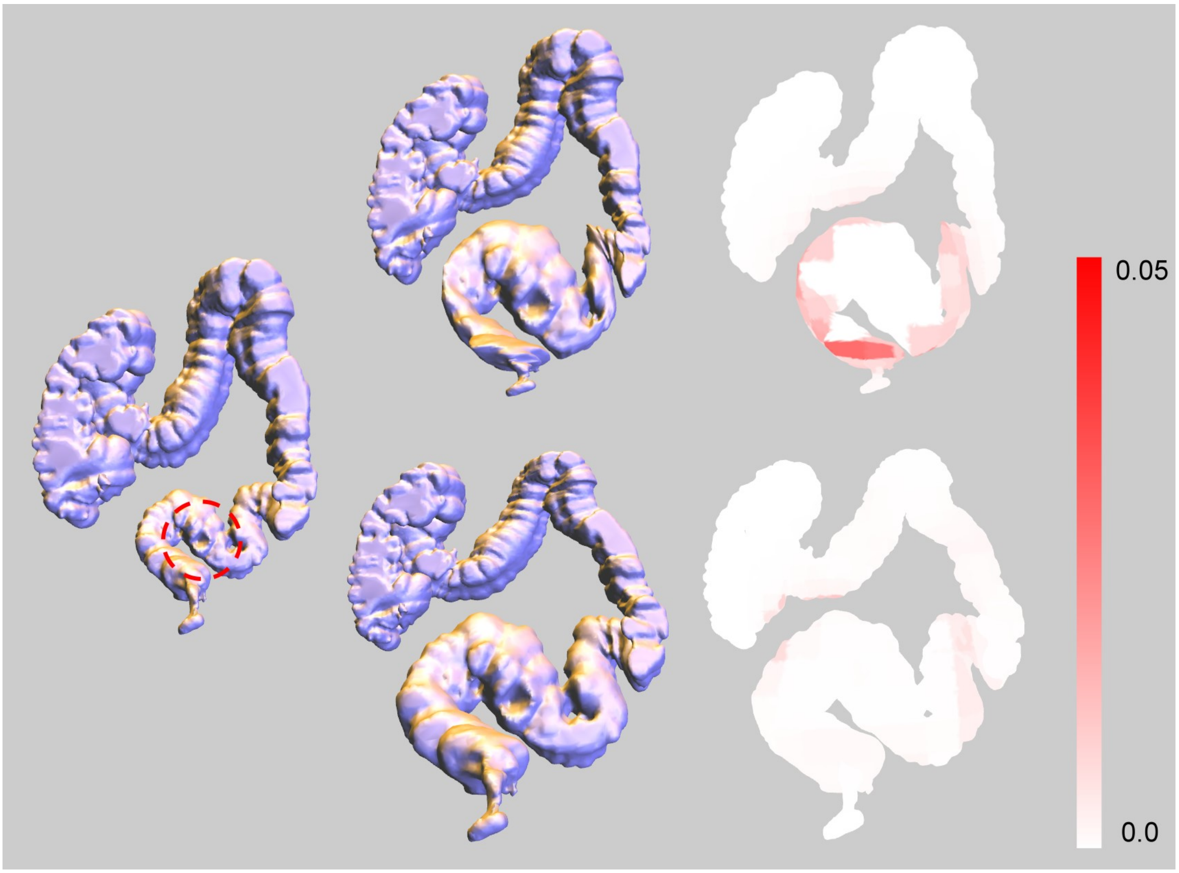


Figure 5.6: (left) Original model. (middle) Deformed models by radial Gaussian distortion (top) [9,10] and by our method (bottom). We use different shades of red to represent the distortion distribution (right). Notice that the distortion is much smaller with our method and does not gather around the focal region.

to fit the feature of interest, leading to an easy and intuitive interface.

5.3.1 Distortion Measurement

We determine the degree of similarity between the original and deformed models by considering the distortion of each local region. The distortion can be measured in terms of the amount of stretching of individual cubes because the embedded input model is deformed by space interpolation. Obviously, a deformed cube that is of the same shape, but different size, as the original cube suffers from zero distortion. Therefore, to determine the amount of distortion, we first scale each pair of original and deformed cubes to the same size and then accumulate the difference of their 12 edge vectors. Specifically, we compute the equation

$$\sum_{\{i,j\}\in E_k} |\delta(\mathbf{v}_i - \mathbf{v}_j) - (\mathbf{v}'_i - \mathbf{v}'_j)|^2, \tag{5.8}$$

where E_k denotes the 12 edges of the local cube k, and $\delta = (M'_k/M_k)^{1/3}$, with M'_k and M_k denoting the volume of its original and deformed cubes, respectively. The region with lower distortion value means that it is similar to its original shape and the region with higher distortion value is naturally stretched. For example, in Figure 5.6, we expand a portion of the colon and measure the resulting distortion due to the magnification. It can be clearly observed that, without distortion minimization, the regions surrounding the focal region are seriously distorted. In comparison, our system produces results with much smaller and unnoticeable distortion. In the case when all the cubes are principle cubes, our system produces results similar to those with radial Gaussian distortion [9,10] since there is no space to absorb distortion. Under this situation, the distortion is uniformly propagated outward from the focal region to other regions.

5.3.2 Soft Constraint

An interesting feature of our system is the use of soft constraints to solve for the deformed grid space in a least-squares sense. This leads to the actual size of the deformed focal region being slightly smaller than the user-specified magnifying factor for the focal region. We take the gargoyle model in Figure 5.3 as an example to illustrate this situation. Although we apply a $2 \times \mathbf{I}$ transformation on the focal region to determine the deformed grid space, the average volume of the deformed focal region is 1.46×10^{-4} while their original volume is 0.26×10^{-4}. The expanding ratio, i.e., 6.35, is less than the expected 2^3 because the uniformity and the magnification requirements conflict with each other (i.e., within the limited space, it is impossible for the focal region to expand with no distortion), and the least-squares solver finds the optimum solution to satisfy the overall requirements.

Solving for the deformed grid space using soft constraints inevitable causes a little

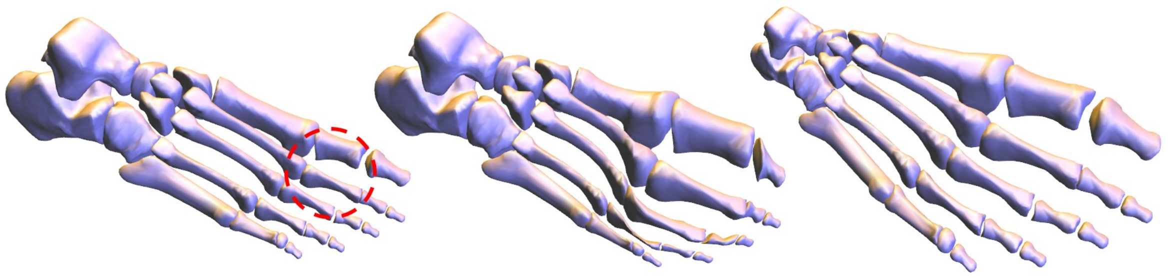


Figure 5.7: Original model (left). Observe that the Gaussian distortion method (middle) better preserves the global shape of the foot bone than our method (right). However, the user usually pays more attention to the regions inside and surrounding the magnifying lens and our method produces very little artifact in these regions.

distortion to the focal region because the system compromises the uniformity of the focal cubes to reduce the distortion of the overall model. Some applications may require the focal region to be expanded without distortion. Minimizing the objective function with hard constraints can achieve this aim and ensure that the size of the deformed focal cube is equal to the user's specification. Unfortunately, this approach would rapidly increase the cost if the focal region changes constantly since the modification of the hard constraints (i.e., focal region) changes the structure of the linear system and thus requires matrix refactorization. For efficiency, we solve the objective function with only soft constraints while the user is changing the focal region interactively and change the soft constraints into hard constraints to exactly retain the size and uniformity of the focal cubes after the user's specification remains fixed for some time.

5.3.3 Pros and Cons

Our system deforms the entire model to achieve Focus+Context visualization. While magnifying the focal region to display it with higher resolution, some regions are expanded while some are reduced to minimize distortion of every local region. Unlike the radial Gaussian distortion method [9,10], which deforms only the cubes inside or close to the focal region, our system rescales almost all the cubes covering the model. Although the deformation is everywhere, the user can easily observe the continuous variation of the model's shape because the model is deformed interactively. Since every local region is rescaled, our method can not retain the model's global shape while minimizing the local distortion. For example, in Figure 5.7, the global shape of the foot bone model is changed with our approach but is better preserved by the radial Gaussian distortion. We argue that the change of the global shape is not necessarily a bad feature. According to the theory of human central vision, only a small area in the center of retina contains a rich collection of cone cells, which is sensitive to light, fine detail and color. Therefore, users usually concentrate on a region of interest and its surroundings when using a magnifying lens. This means that the model's overall shape is only a concept and the variation of the global shape is not disturbing to the user. In addition, since our system achieves real time performance, the user can easily move the magnifying lens to any free space not occupied by the model any time to see the model's undeformed global shape.

Our system offers another advantage for Focus+Context visualization. The magnifying lens is of limited size (i.e., smaller than the screen size), and during interactive visualization, the user is often also interested in the non-focal regions close to the lens border. Thanks to the distortion minimization, these surrounding regions, which are likely to be the next focal regions, are also enforced to expand because the cubes are connected.

Overall, we introduce a novel and interactive technique to achieve Focus+Context visualization of 3D models. The main contribution of our method is to minimize the resulting distortion such that each local region appears similar to its original counterpart. Our algorithm retains the original global size of the model by constraining the grid vertices to move within the global bounding space and prevents the local regions from squeezing through approximating the applied transformations to be close to uniform scales. Finally, our algorithm has the potential to handle point models and volumetric data since the embedded model is deformed by space interpolation. While the cubes are transformed to satisfy the local magnification requirement, the positions of the point clouds or the voxels are redistributed as the combination of its surrounding cube vertices.

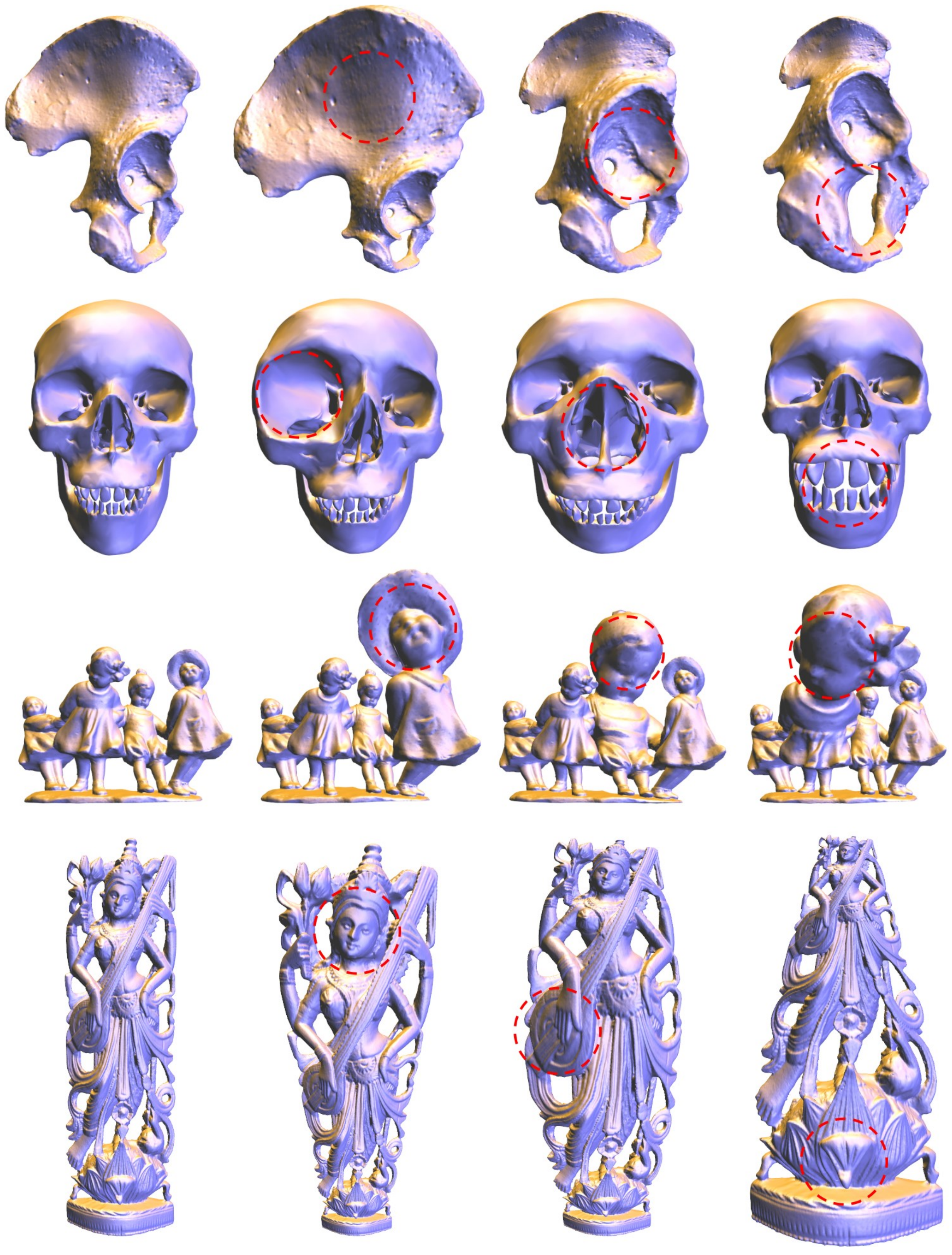


Figure 5.8: Original models (leftmost column). Our algorithm expands the focal region and reconstructs the embedded model from the deformed grid space to achieve Focus+Context visualization (the remaining columns).

Chapter 6 Conclusion

We introduced spatial content preservations to retain the aspect ratios of prominent objects when resizing images and videos. The squeezing and stretching of each local region is determined based on the obtained importance values. Since we allowed prominent objects to be uniformly resized, our method can fully utilize all the available homogeneous regions to float the resulting artifacts. The preservation of grid lines can avoid the distortions of objects that occupy connected quads. Besides, this constraint can also prevent the folded over problem, which is commonly seen in the image warping methods.

To extent the image resizing to video resizing, we also presented a practical framework which can handle videos of complex dynamic scenes. We observed that camera and object motion cause feature correspondences to deviate from temporally adjacent pixels, easily causing flickering or waving artifacts. We found that minimizing motion artifacts during resizing can be achieved by preserving both camera and object motion and introduced motion-aware temporal coherence constraints to preserve them. Our streaming implementation leads to a scalable video resizing system that consistently produces spatiotemporally coherent resizing results. We believe that by introducing the concept of motion-aware constraints, we have taken a significant step towards a more practical video resizing system.

Different content-aware image/video resizing methods each have their own strengths. It would be interesting to combine our ideas of motion-aware constraints with the other image resizing methods or with the existing video retargeting methods to achieve better temporal coherence. In addition, the preservation of global structures, such as straight lines and different types of symmetries, is as well important, requiring explicit structure detection and extra structural constraints.

Reference

[1] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM Trans. Graph.*, vol. 26, no. 3, p. 10, 2007.

[2] M. Rubinstein, A. Shamir, and S. Avidan, "Improved seam carving for video retargeting," *ACM Trans. Graph.*, vol. 27, no. 3, 2008.

[3] R. Gal, O. Sorkine, and D. Cohen-Or, "Feature-aware texturing," in *Proceedings of Eurographics Symposium on Rendering*, pp. 297–303, 2006.

[4] L. Wolf, M. Guttmann, and D. Cohen-Or, "Non-homogeneous content-driven video-retargeting," in *Proceedings of IEEE ICCV*, pp. 1–6, 2007.

[5] Y.-F. Zhang, S.-M. Hu, and R. R. Martin, "Shrinkability maps for content-aware video resizing," in *PG '08*, 2008.

[6] T. A. Keahey and E. L. Robertson, "Nonlinear magnification fields," in *INFOVIS '97: Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*, p. 51, IEEE Computer Society, 1997.

[7] T. A. Keahey and E. L. Robertson, "Techniques for non-linear magnification transformations," in *INFOVIS '96: Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96)*, p. 38, IEEE Computer Society, 1996.

[8] T. A. Keahey, "The generalized detail-in-context problem," in *INFOVIS '98: Proceedings of the 1998 IEEE Symposium on Information Visualization*, pp. 44–51, IEEE Computer Society, 1998.

[9] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia, "Distortion viewing techniques for 3-dimensional data," in *INFOVIS '96: Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96)*, p. 46, IEEE Computer Society, 1996.

[10] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia, "Extending distortion viewing from 2d to 3d," *IEEE Comput. Graph. Appl.*, vol. 17, no. 4, pp. 42–51, 1997.

[11] E. LaMar, B. Hamann, and K. I. Joy, "A magnification lens for interactive volume visualization," in *PG '01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, p. 223, IEEE Computer Society, 2001.

[12] L. Wang, Y. Zhao, K. Mueller, and A. E. Kaufman, "The magic volume lens: An interactive focus+context technique for volume rendering," in *IEEE Visualization*, p. 47, 2005.

[13] 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," *ACM Multimedia Systems Journal*, vol. 9, no. 4, pp. 353–364, 2003.

[14] H. Liu, X. Xie, W.-Y. Ma, and H.-J. Zhang, "Automatic browsing of large pictures on mobile devices," in *Proceedings of ACM International Conference on Multimedia*, pp. 148–155, 2003.

[15] A. Santella, M. Agrawala, D. DeCarlo, D. Salesin, and M. Cohen, "Gaze-based interaction for semi-automatic photo cropping," in *Proceedings of CHI*, pp. 771–780, 2006.

[16] B. Suh, H. Ling, B. B. Bederson, and D. W. Jacobs, "Automatic thumbnail cropping and its effectiveness," in *Proceedings of UIST*, pp. 95–104, ACM, 2003.

[17] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[18] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 11, pp. 1254–1259, 1998.

[19] D. DeCarlo and A. Santella, "Stylization and abstraction of photographs," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 769–776, 2002.

[20] T. S. Cho, M. Butman, S. Avidan, and W. T. Freeman, "The patch transform and its applications to image editing," in *CVPR '08*, 2008.

[21] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani, "Summarizing visual data using bidirectional similarity," in *CVPR '08*, 2008.

[22] V. Kraevoy, A. Sheffer, D. Cohen-Or, and A. Shamir, "Non-homogeneous resizing of complex models," *ACM Trans. Graph.*, vol. 27, no. 5, p. 111, 2008.

[23] Y.-S. Wang, T.-Y. Lee, and C.-L. Tai, "Focus+context visualization with distortion minimization," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, no. 6, 2008.

[24] F. Liu and M. Gleicher, "Video retargeting: automating pan and scan," in *Multimedia '06*, pp. 241–250, 2006.

[25] C. Tao, J. Jia, and H. Sun, "Active window oriented dynamic video retargeting," in *Workshop on Dynamical Vision, ICCV '07*, 2007.

[26] M. Rubinstein, A. Shamir, and S. Avidan, "Multi-operator media retargeting," *ACM Trans. Graph.*, vol. 28, no. 3, p. 23, 2009.

[27] I. Viola, A. Kanitsar, and M. E. Groller, "Importance-driven volume rendering," in *VIS '04: Proceedings of the conference on Visualization '04*, pp. 139–146, IEEE Computer Society, 2004.

[28] J. Zhou, M. Hinz, and K. D. Tonnies, "Focal region-guided feature-based volume rendering.," in *3DPVT*, pp. 87–90, IEEE Computer Society, 2002.

[29] M. J. McGuffin, L. Tancau, and R. Balakrishnan, "Using deformations for browsing volumetric data," in *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, p. 53, IEEE Computer Society, 2003.

[30] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose, "Toolglass and magic lenses: the see-through interface," in *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 73–80, ACM, 1993.

[31] H. Fang and J. C. Hart, "Detail preserving shape deformation in image editing," *ACM Trans. Graph.*, vol. 26, no. 3, p. 12, 2007.

[32] R. Szeliski, "Image alignment and stitching: a tutorial," *Foundations and Trends in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2006.

[33] B.-Y. Chen, K.-Y. Lee, W.-T. Huang, and J.-S. Lin, "Capturing intention-based full-frame video stabilization," *Computer Graphics Forum*, vol. 27, no. 7, pp. 1805–1814, 2008.

[34] M. L. Gleicher and F. Liu, "Re-cinematography: Improving the camerawork of casual video," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 5, no. 1, pp. 1–28, 2008.

[35] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[36] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[37] T. Deselaers, P. Dreuw, and H. Ney, "Pan, zoom, scan - time-coherent, trained automatic video cropping," in *CVPR '08*, 2008.

[38] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *SGP '04*, pp. 179–188, 2004.

[39] H.-W. Kang, Y. Matsushita, X. Tang, and X.-Q. Chen, "Space-time video montage," in *CVPR '06*, 2006.

[40] Z. Rasheed and M. Shah, "Scene detection in hollywood movies and tv shows," in *CVPR '03*, vol. 2, pp. II–343–8, 2003.

[41] Y.-S. Wang, C.-L. Tai, O. Sorkine, and T.-Y. Lee, "Optimized scale-and-stretch for image resizing," *ACM Trans. Graph.*, vol. 27, no. 5, p. 118, 2008.

[42] P. Krähenbühl, M. Lang, A. Hornung, and M. Gross, "A system for retargeting of streaming video," *ACM Trans. Graph.*, vol. 28, no. 5, 2009.

[43] T. Ju, S. Schaefer, and J. Warren, "Mean value coordinates for closed triangular meshes," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 561–566, 2005.

[44] K. Madsen, H. Nielsen, and O. Tingleff, "Methods for nonlinear least squares problems," *Tech. rep., Informatics and Mathematical Modelling*, 2004.

[45] K. Madsen, H. Nielsen, and O. Tingleff, "Optimization with constraints," *Tech. rep., Informatics and Mathematical Modelling*, 2004.

[46] S. Toledo, D. Chen, and V. Rotkin, "Taucs: A library of sparse linear solvers, version 2.2.," *Tel-Aviv University, Available online at http://www.tau.ac.il/ stoledo/taucs/.*

Vita

Yu-Shuen Wang received his PhD degree from National Cheng Kung University, Tainan, Taiwan, Republic of China, in 2010. His research interests include computer graphics, Image/Video Resizing, Focus+Context Visualization, mesh segmentation, skeletonization, and computer animation.

Publications

Journals

1. **Yu-Shuen Wang**, Hongbo Fu, Olga Sorkine, Tong-Yee Lee, and Hans-Peter Seidel, "Motion-Aware Temporal Coherence for Video Resizing." *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia 2009)*, Vol. 28, No.5, Dec. 2009 (SCI/EI)

2. **Yu-Shuen Wang**, Chaoli Wang, Tong-Yee Lee, Kwan-Liu Ma, "Feature Preserving Data Reduction and Focus+Context Visualization." *IEEE Transactions on Visualization and Computer Graphics*, To appear (SCI/EI)

3. **Yu-Shuen Wang**, Chiew-Lan Tai, Olga Sorkine, Tong-Yee Lee, "Optimized Scale-and-Stretch for Image Resizing." *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia 2008)* Vol. 27, No.5, Dec. 2008 (SCI/EI)

4. **Yu-Shuen Wang**, Tong-Yee Lee, Chiew-Lan Tai, "Focus+Context Visualization with Distortion Minimization." *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE Visualization 2008)*, Volume 14, Number 6, November, 2008 (SCI/EI)

5. **Yu-Shuen Wang**, Tong-Yee Lee, "Curve Skeleton Extraction Using Non-Linear Least Squares Optimization." *IEEE Transactions on Visualization and Computer*

Graphics, July/Aug, Vol. 14, No. 4. 2008, pp. 926-936 (SCI/EI)

6. Tong-Yee Lee, Chao-Hung Lin, **Yu-Shuen Wang**, Tai-Guang Chen,” Animation Key-frame Extraction and Simplification Using Deformation Analysis.” *IEEE Transactions on Circuits and Systems for Video Technology*, April, Vol. 18, No. 4, 2008, pp. 478-486 (SCI/EI)
7. **Yu-Shuen Wang**, Tong-Yee Lee, ”Example-driven Animation Synthesis.” *The Journal of Visual Computer (Proceedings of Graphics International 2008)* (SCI/EI)
8. **Yu-Shuen Wang**, Chao-Hung Lin, Tong-Yee Lee, ”Interactive Model Decomposition Using Protrusive Graph.” *International Journal of Innovative Computing, Information and Control (IJICIC)*, Vol.4, No.8, August 2008, pp. 1887-1896. (SCI/EI)
9. Tong-Yee Lee, Ping-Hsien Lin, Shao-Wei Yen, Ming-Te Chi, **Yu-Shuen Wang,** Chih-Yuan Yao1 and Jin-Lung Lin, ”Exaggeration Cloning From Example Sequence”, *(invited) IJCSES International Journal of Computer Sciences and Engineering Systems*, Vol. 3, No. 4, Oct. 2009.
10. Tong-Yee Lee, Chao-Hung Lin, Hung-Kuo Chu, **Yu-Shuen Wang**, Shao-Wei Yen, Chang-Rung Tsai, ”Mesh Pose-Editing Using Examples.” *Computer Animation and Virtual Worlds Journal*, Volume 18, Issue 4-5 (Sep. - Dec. 2007), pp. 235-245. (SCI/EI)
11. Tong-Yee Lee, **Yu-Shuen Wang** , Tai-Guang Chen, ”Segmenting a Deforming Mesh into Near-Rigid Components.” *The Journal of Visual Computer (Proceedings of Pacific Graphics 2006)* Vol. 22, No. 9-11, Sept. 2006, pp. 729-739 (SCI/EI)

Conferences

12. **Yu-Shuen Wang**, Tong-Yee Lee, ”WYSIWYG: Mesh Decomposition for Static Models.” *Proceedings of Intelligent Information Hiding and Multimedia Signal Processing (Special Session on Computer Graphics)*, Nov. 26-28, 2007. (EI)

13. **Yu-Shuen Wang**, Tong-Yee Lee, Chao-Hung Lin, "Interactive Model Decomposition." *Proceeding of Computer-Aided Design and Computer Graphics (CAD/Graphics 2007 sponsored by IEEE and ACM SIGGRAPH)*, Oct. 2007. (EI)

Professional activities

Reviewers

Journals:	ACM Transactions on Graphics, IEEE Transactions on Visualization and Computer Graphics, Computer Graphics Forum.
Conferences:	ACM SIGGRAPH, ACM SIGGRAPH Asia, Eurographics, Pacific Graphics, IEEE Visualization, Graphics Interface.

Research visits and collaborations

08.2007	Hong Kong University of Science and Technology
02.2009	University of California, Davis
10.2009	New York University

Invited talks, conference presentations

12.2009	ACM Siggraph Asia Conference, Yokohama, Japan
10.2009	New York University, New York, USA (invited talk)
12.2008	ACM Siggraph Asia Conference, Singapore
10.2008	IEEE Visualization Conference, Columbus, USA
06.2008	Computer Graphics International Conference, Istanbul, Turkey
10.2006	Pacific Graphics Conference, Taipei, Taiwan