Resizing by Symmetry-Summarization

Huisi Wu[1,3] Yu-Shuen Wang[2] Kun-Chuan Feng[2] Tien-Tsin Wong[1] Tong-Yee Lee[2] Pheng-Ann Heng[1,3]
[1]The Chinese University of Hong Kong [2]National Cheng Kung University [3]SIAT, China

(a) Original (b) Multi-Operator [Rubinstein et al. 2009] (c) Shift-Map [Pritch et al. 2009] (d) Warping [Wang et al. 2008] (e) Ours

Figure 1: *By summarizing the symmetry structure in (a) the input image, our method (e) can preserve the original curved lattice structure without breaking, over-squeezing or cropping the lattice. Input resolution is 999×820. Target resolution is 516×820.*

Abstract

Image resizing can be achieved more effectively if we have a better understanding of the image semantics. In this paper, we analyze the *translational symmetry*, which exists in many real-world images. By detecting the symmetric lattice in an image, we can *summarize*, instead of only distorting or cropping, the image content. This opens a new space for image resizing that allows us to manipulate, not only image pixels, but also the semantic *cells* in the lattice. As a general image contains both symmetry & non-symmetry regions and their natures are different, we propose to resize symmetry regions by summarization and non-symmetry region by warping. The difference in resizing strategy induces discontinuity at their shared boundary. We demonstrate how to reduce the artifact. To achieve practical resizing applications for general images, we developed a fast symmetry detection method that can detect multiple disjoint symmetry regions, even when the lattices are curved and perspectively viewed. Comparisons to state-of-the-art resizing techniques and a user study were conducted to validate the proposed method. Convincing visual results are shown to demonstrate its effectiveness.

Keywords: image resizing, summarization

1 Introduction

Image resizing techniques fit an input image to the target resolution by reducing or replicating the image content. Image seamlessness is the key. Existing methods [Wang et al. 2008; Rubinstein et al. 2009; Dong et al. 2009] rely on the importance or saliency map to prevent the modification from being over-aggressive. However, the importance or saliency may not confirm to the true semantics, since mainly local and low-level features (such as gradient and/or entropy) are considered. Without higher-level understanding of the image content, the image seamlessness is hard to maintain.

Although computational understanding of general image content is infeasible in the near future, analysis of certain high-level semantics is feasible. One of them is *symmetry*. It exists everywhere, from windows on the buildings to soldiers in marching. The symmetry structure and the repetitive elements (or cells) reinforce the visual importance. Previous resizing techniques attempt to modify without preserving the symmetry structure, and may easily result in apparent visual artifacts, such as the obvious seam (Figure 1(c)) or over-squeezing (1(b) and 1(d)). By considering the knowledge of symmetry, resizing can then be achieved via *summarization*, which removes or replicates the cells with respect to the semantics (Figure 1(e)). Note that we extend the original meaning of summarization to include both reduction and repetition of cells. In other words, symmetry opens an additional space for resizing. Instead of squeezing or stretching the image *pixel-wisely*, we can now remove or replicate it *cell-wisely*.

In this paper, we propose a novel summarization operator for image resizing that handles one common type of symmetry, the *translational symmetry* [Liu et al. 2004]. Existing symmetry detection methods are usually too slow for practical resizing applications. Instead, we propose a real-time and automatic method to detect symmetry over arbitrary surfaces (planar or non-planar) with arbitrary viewing perspective, and to extract the corresponding lattice in image space without reconstructing the underlying 3D geometry. By trimming and extending the lattice, we can resize the content with more respect to the semantics. By smoothing the transformation and intensity of cells across the lattice, we can maintain the seamlessness in both geometry and illumination respectively. In contrast, most existing resizing techniques do not consider the seamlessness of illumination.

However, a general image normally contains both symmetry region (S-region) and non-symmetry region (NS-region), leading to complication. While the S-region can be resized with our summarization operator, the NS-region can be resized with carving or warping operators. But most importantly, there is no guarantee that both resultant regions can be seamlessly combined, due to the different natures of resizing strategies. In this paper, we propose a frame-

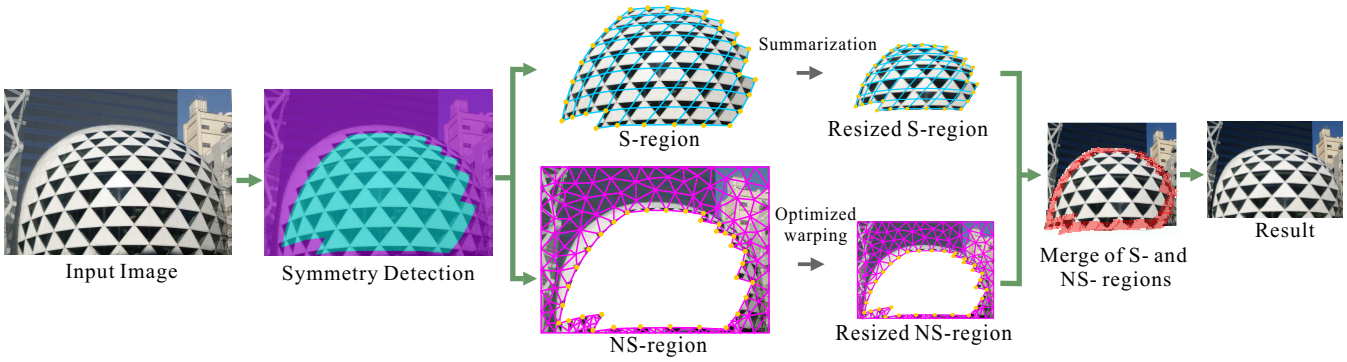


Figure 2: *Framework of our system*

work to minimize the inconsistency of the sharing boundary between S- and NS- regions. A mesh-based representation is used to offer constrained warping for the NS-region and ensure its boundary confirms to that of S-region. A graph-cut is then performed to minimize the intensity discontinuity at overlapped area near the boundary. Convincing results are obtained and a user study is performed to validate our method.

Our major contributions can be summarized as follow:

- An automatic and real-time summarization operator for image resizing. By smoothing the transformation and intensity of cells over the lattice, we can maintain the seamlessness of not only the geometry, but also the illumination.
- A framework that seamlessly combines S- and NS-regions, where they are resized differently due to their difference in nature.

2 Related Work

Image Resizing Most content aware methods attempt to take advantages from the detection of pixel prominence. They either discard or distort the homogeneous regions in order to absorb the resulting distortion when changing the resolution of an image. The cropping methods [Chen et al. 2003; Liu et al. 2003; Suh et al. 2003; Santella et al. 2006] remove the less important objects from the image periphery. The seam carving methods [Avidan and Shamir 2007; Rubinstein et al. 2008] judiciously carves the interior seams without touching the salient features. The warping methods [Gal et al. 2006; Wolf et al. 2007; Zhang et al. 2008; Wang et al. 2008] squeeze or stretch the regions with less importance values to preserve the sizes or aspect ratios of prominent objects. Recently, the multi-operators techniques [Rubinstein et al. 2009; Dong et al. 2009] crop and carve seams on only the inconspicuous materials. Thus, all the above methods potentially suffer from visual distortions when the image is dramatically retargeted or the homogeneous regions run out.

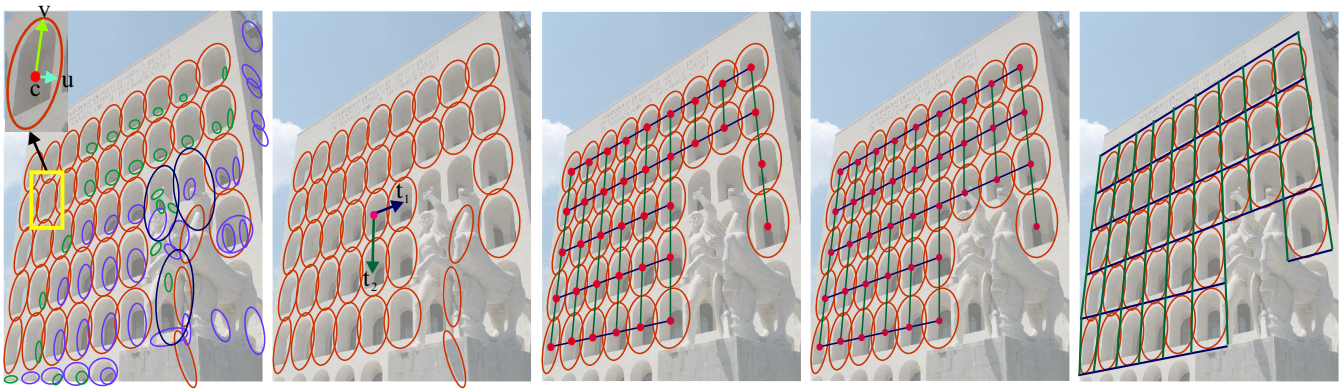
Image Summarization Alternatively, the patch redistribution methods have been presented to achieve image editing and retargeting applications [Simakov et al. 2008; Cho et al. 2008; Barnes et al. 2009]. These techniques measure the patch similarity and then preserve the content coherence between the source and target images. As a result, the repetitive patterns are discarded when the target image is getting small. Pritch et al. [2009] presented a Shift-Map technique that allows removing a band region at a time, instead of a pixel-wise seam used in the seam carving methods, enabling the removal of entire objects. However, all these methods cannot handle the patterns with different sizes or perspective distortion due to the lack of understanding of high-level symmetry semantic. In contrast, our method removes or replicates the repetitive elements/regions in a more semantic fashion, and hence can better preserve the symmetry structure.

Symmetric patterns detection Leung and Malik [1996] proposed a greedy algorithm to extract repetitive elements that are added in a connected set without a global topological structure. Later, Liu et al. [2004; 2006; 2009] developed a more complete theory of computational symmetry. They first proposed a computational model to extract periodic repeated elements and then classify symmetric patterns based on theory of crystallographic groups [Liu et al. 2004]. Hays et al. [2006] formulated discovering symmetric regularity as a higher-order correspondence problem and solved it using a spectral method. Lin and Liu [2007] tracked dynamic near-regular patterns based on a lattice-based Markov-random-field (MRF) model within a 3D spatiotemporal space. Ahuja and Todorovic [2007] represented the image with a segmentation tree and proposed a learning algorithm which combines tree matching, belief propagation and expectation-maximization to extract texels from nearly planar textures whose viewing direction is nearly along surface normal. Park et al. [2009] used a mean-shift belief propagation (MSBP) method to extract deformed lattice from repeated patterns. The automatic symmetry detection of our work is based on their work [Hays et al. 2006; Park et al. 2009] but with a significant improvement on speed at the expense of detection accuracy. Recently, Cheng et al. [2010] employed a user assisted segmentation to scribble a template object and extract approximately repeated elements by finding similar boundary band map.

3 Overview

Figure 2 overviews our framework. Given an input image, our system automatically detects one or more regions containing symmetric patterns, and divides the image into one NS-region and one or more S-regions (Section 4.1). As each S-region contains repetitive cells organized in a grid structure, it is resized via the proposed summarization operator, in order to maintain the symmetry structure, overall shape, as well as the illumination distribution (Section 4.2). On the other hand, the NS-region contains general content. It is resized via warping, so that sub-regions with high salient objects are scaled uniformly while homogeneous sub-regions are squeezed to fit the target resolution (Section 4.3).

The last step is to merge the resized S- and NS- regions. However, due to the different resizing strategies in S- and NS- regions, the boundaries of resized S- and NS- regions may not be consistent to each other. Figure 7(a) shows one such example. To reduce the inconsistency, we represent both S- and NS- regions using meshes (Figure 2). For the S-region, it is natural to have the region being tessellated according to the detected lattice. For the NS-region, the tessellation can be achieved according to the interior contents, such as straight lines. Their shared boundaries must be contained by the edges in their meshes in order to maintain the continuity. The mesh topology of both S- and NS- regions does not change throughout the resizing. The only change is the "texture content" (image) mounted on the mesh. Section 4.4 describes the details on minimizing discontinuity artifact at the boundary.



(a) (b) (c) (d) (e)

Figure 3: *Symmetry detection. (a) MSER detection and clustering. Each cluster is color-coded with a distinct color. (b) Determination of translation vectors* $\mathbf{t_1}$ *&* $\mathbf{t_2}$*. (c) Lattice formation. (d) Picking missed cells by interpolation and extrapolation. (e) Dual lattice.*

4 Algorithm

4.1 Symmetry Detection

Identifying Cells The first step for symmetry detection is the choice of the basic element or *cell*. One can utilize Canny edges [Canny 1986], Harris corners [Harris and Stephens 1988], KLT corners [Shi and Tomasi 1994] or SIFT points [Lowe 2004] to identify the feature points. However, a cell in a symmetry structure can seldom be identified by a single feature point, but rather as a region containing a group of features. The clustering and identification of such group based on the above isolated feature points could be inefficient and time-consuming. Hence, we need an efficient method to quickly locate regions, instead of points, with similar content. In particular, we employ the maximally stable extremal region (MSER) [Matas et al. 2002] for identifying a feature region.

MSERs are affine invariant and less influenced by the illumination. This means cells with the same pattern, but viewed from different perspectives and illuminated by different lighting conditions can still be equally identified. The detailed mathematical definition of MSER and its implementation can be found in [Matas et al. 2002]. Suppose $\mathbf{R} = \{r_i\}$ is the set of MSERs detected in an image (Figure 3(a)). Each MSER r_i is graphically a 2D ellipse represented by a triplet $r_i = \{\mathbf{c_i}, \mathbf{u_i}, \mathbf{v_i}\}$ where $\mathbf{c_i}$ is its center, $\mathbf{u_i}$ and $\mathbf{v_i}$ are its major and minor axis vectors.

Clustering and Indexing A symmetry structure normally contains multiple cells and similar cells should have similar MSER feature. This suggests that we can perform clustering on the MSERs to identify the major clusters, and simultaneously filter away the outliers. Figure 3(a) shows the major clusters by color-coding each cluster with a distinct color. Each major cluster corresponds to a potential symmetry structure (S-region). The number of S-regions in a single image is usually not large. Hence, we only identify the major κ_c clusters which have the largest total area of coverage. In our current implementation, $\kappa_c = 5$. We use adaptive mean-shift clustering [Comaniciu and Meer 2002] to group the detected MSERs. For each MSER r_i, we form a 2D vector by concatenating its $|\mathbf{u_i}|$ and $|\mathbf{v_i}|$. These vectors are fed for clustering in order to group MSERs with similar shapes and sizes.

To facilitate the following lattice formation, we further index the centers $\mathbf{c_i}$ of MSERs in each cluster with a KD-tree. For fast retrieval, we implemented the KD-tree on GPU to support the parallel KNN-search.

Similarity Metric We can then form a lattice for each cluster by starting from a seed MSER and connecting the neighboring MSERs. However, the resulting clusters may potentially contain some wrong MSERs that are not really cells, as the clustering is only based on a crude measurement of ellipse similarity without considering the region content. Hence, we need a more sophisticated metric to validate and select neighboring cells during the lattice formation. We define a metric to quantify the similarity of two neighboring MSERs in terms of the shape and appearance similarities. Even when the symmetry structure is not planar, two neighboring cells remain close in shape. Hence, the shape dissimilarity D_s of two neighboring MSERs r_i and r_j is a function of their major and minor axes ($\mathbf{u}$ and $\mathbf{v}$) and defined as

$$D_s(r_i, r_j) = \frac{|\mathbf{u_i} - \mathbf{u_j}|}{\max\{|\mathbf{u_i}|, |\mathbf{u_j}|\}} + \frac{|\mathbf{v_i} - \mathbf{v_j}|}{\max\{|\mathbf{v_i}|, |\mathbf{v_j}|\}} \tag{1}$$

The appearance dissimilarity D_a measures their difference in terms of pixel intensity, after registration and normalization. Suppose the $\mathbf{P_i}$ and $\mathbf{P_j}$ are the image regions falling inside the ellipses of two considering MSERs r_i and r_j, respectively. A simple registration or transformation $\mathbf{A}$ from $\mathbf{P_j}$ to $\mathbf{P_i}$ can be obtained by translating from the center $\mathbf{c_j}$ to $\mathbf{c_i}$, rotating the major axis vector from $\mathbf{u_j}$ to $\mathbf{u_i}$, and scaling by $|\mathbf{u_i}|/|\mathbf{u_j}|$ and $|\mathbf{v_i}|/|\mathbf{v_j}|$ along the major and minor axes. Then the appearance dissimilarity is computed as,

$$D_a(r_i, r_j) = \frac{1}{n}\sqrt{\sum^{n}\left((p_i - \overline{p}_i) - (p_j - \overline{p}_j)\right)^2} \tag{2}$$

where p_i is the pixel value in $\mathbf{P_i}$; p_j is the corresponding pixel value in the transformed $\mathbf{P_j}$; $\overline{p}_i$ and $\overline{p}_j$ are the mean pixel values of $\mathbf{P_i}$ and the transformed $\mathbf{P_j}$, respectively; n is the size of $\mathbf{P_i}$. The zero-mean design in (2) allows us to minimize the influence of illumination and focus only on the pattern. The overall dissimilarity D is defined as a product of D_s and D_a,

$$D(r_i, r_j) = D_s(r_i, r_j) \cdot D_a(r_i, r_j) \tag{3}$$

Lattice Formation According to the group theory of wallpaper patterns [Grünbaum and Shephard 1986], all translational symmetric patterns can be represented by a 2D lattice generated by a pair of shortest translation vectors $\mathbf{t_1}$ and $\mathbf{t_2}$. Hence, the key is to identify the translation vectors during the lattice formation. There exists methods that automatically extract the optimal lattice, but they are usually too slow for our resizing application. The method by [Park et al. 2009] generally takes about 5-10 minutes to handle a medium size image. As our goal is not to recover the real geometry, but to identify the symmetry for resizing, we can trade the lattice accuracy for speed.

The symmetry in an image is usually perspectively viewed and the underlying geometry can be non-planar (e.g. windows on cylindrical buildings or irregularly curved surface of modern museums). A pair of constant translation vectors is not sufficient to represent the lattice. Instead of recovering the 3D lattice, we are only interested

Algorithm 1 Lattice Formation

// A function computing deviation of two vectors

Define $\texttt{diff}(\mathbf{v_1},\mathbf{v_2})=\begin{cases} +\infty, & \frac{|\mathbf{v_1}-\mathbf{v_2}|}{\max(|\mathbf{v_1}|,|\mathbf{v_2}|)} > \tau_v \\ \frac{|\mathbf{v_1}-\mathbf{v_2}|}{\max(|\mathbf{v_1}|,|\mathbf{v_2}|)}, & \text{otherwise} \end{cases}$

// Initialization Phase
Given a seed cell r
$\mathbf{C} = \emptyset$; $\mathbf{K} = \emptyset$ // $\mathbf{C}$: cells, $\mathbf{K}$: translation vectors
$\mathbf{G} = \{\mathbf{C}, \mathbf{K}\}$
Search two closest neighbors r_a&r_b, if exist, by minimizing
$D(r, r_a)D(r, r_b)|\mathbf{t_1}(r)||\mathbf{t_2}(r)|e^{\tau_s(\theta-\frac{5\pi}{12})}$, where $\mathbf{t_1}(r) = \mathbf{c_a} - \mathbf{c}$;
$\mathbf{t_2}(r) = \mathbf{c_b} - \mathbf{c}$; and $\theta = \arccos(\mathbf{t_1}(r) \cdot \mathbf{t_2}(r))$ // $\mathbf{c_i}$ means center of cell r_i
Search opposite neighbors r'_a&r'_b, if exist, by
$r'_a = \text{argmin}_i\ D(r, r'_a) \cdot \texttt{diff}(\mathbf{c} - \mathbf{c}_i,\ \mathbf{t_1}(r))$ // left
$r'_b = \text{argmin}_i\ D(r, r'_b) \cdot \texttt{diff}(\mathbf{c} - \mathbf{c}_i,\ \mathbf{t_2}(r))$ // top
Put r into $\mathbf{C}$ // processed
Put $\mathbf{t_1}(r)$ & $\mathbf{t_2}(r)$ into $\mathbf{K}$;
for each $r_j \in \{r_a, r_b, r'_a, r'_b\}$
  $\mathbf{t_1}(r_j) = \mathbf{t_1}(r)$; $\mathbf{t_2}(r_j) = \mathbf{t_2}(r)$
  put r_j into $\mathbf{U}$ // i.e. not yet processed
  put $\mathbf{t_1}(r_j)$ & $\mathbf{t_2}(r_j)$ into $\mathbf{K}$

// Propagation Phase
while $\mathbf{U}$ is not empty
  Pick a cell $r_k \in \mathbf{U}$ and its $\mathbf{t_1}(r_k)$, $\mathbf{t_2}(r_k) \in \mathbf{K}$
  Search r_k's neighbor set $\mathbf{N_k} = \{r_c, r_d, r'_c, r'_d\}$, if exist, by
    $r_c = \text{argmin}_i\ D(r_k, r_i) \cdot \texttt{diff}(\mathbf{c}_i - \mathbf{c_k},\ \mathbf{t_1}(r_k))$ // right
    $r_d = \text{argmin}_i\ D(r_k, r_i) \cdot \texttt{diff}(\mathbf{c}_i - \mathbf{c_k},\ \mathbf{t_2}(r_k))$ // bottom
    $r'_c = \text{argmin}_i\ D(r_k, r_i) \cdot \texttt{diff}(\mathbf{c_k} - \mathbf{c}_i,\ \mathbf{t_1}(r_k))$ // left
    $r'_d = \text{argmin}_i\ D(r_k, r_i) \cdot \texttt{diff}(\mathbf{c_k} - \mathbf{c}_i,\ \mathbf{t_2}(r_k))$ // top
  Put r_k into $\mathbf{C}$ and remove r_k from $\mathbf{U}$ // processed
  $\mathbf{t_1}(r_k) = \mathbf{c_c} - \mathbf{c_k}$; $\mathbf{t_2}(r_k) = \mathbf{c_d} - \mathbf{c_k}$
  for each $\mathbf{r_j} \in \mathbf{N_k}$ and $\mathbf{r_j} \notin \mathbf{C}$
    $\mathbf{t_1}(r_j) = \mathbf{t_1}(r_k)$; $\mathbf{t_2}(r_j) = \mathbf{t_2}(r_k)$
    Put r_j into $\mathbf{U}$ // not yet processed
    Put $\mathbf{t_1}(r_j)$&$\mathbf{t_2}(r_j)$ into $\mathbf{K}$
Output: $\mathbf{G} = \{\mathbf{C}, \mathbf{K}\}$

in extracting the 2D image-based lattice for our resizing application. So, we represent the lattice as a set of spatially varying 2D translation vectors $\{\mathbf{t_1}(r), \mathbf{t_2}(r)\}$, in which we index the vectors by the corresponding cell r.

For each MSER r in a major cluster, we try to construct the lattice $\mathbf{G}$ by propagation (Figure 3(b)). Since the constructed lattices from different seeds may be different, due to the noise, we exhaustively construct the lattices from all r to obtain the lattice(s) with as much coverage as possible. We implement this method on GPU to parallelize the construction. Note that MSERs previously grouped in the same cluster only mean they are similar in shapes and appearances. They are not necessarily cell-by-cell connected. Hence, disjoint lattices have to be disconnected and resized separately.

Algorithm 1 presents the pseudo-code for the lattice formation starting from an arbitrary seed cell r and outputs the lattice $\mathbf{G}$, composed by the set of cells $\mathbf{C}$ and the set of spatially varying translation vectors $\mathbf{K}$. In the first part of the pseudo-code, we locate the four neighbors of the seed r, via the previously constructed KD-tree. We first find an optimal pair of two neighbors, r_a and r_b, such that the associated translation vectors $\mathbf{t_1}(r) = \mathbf{c_a} - \mathbf{c}$ and $\mathbf{t_2}(r) = \mathbf{c_b} - \mathbf{c}$ form a subtended angle within $[\frac{\pi}{3}, \frac{\pi}{2}]$ [Schattschneider 1978], and give the minimal score of an objective function, $D(r, r_a)D(r, r_b)|\mathbf{t_1}(r)||\mathbf{t_2}(r)|e^{\tau_s(\theta-\frac{5\pi}{12})}$. Here, parameter τ_s weights the importance of the subtended angle and equals to 0.5 in all our experiments. Once r_a and r_b are located, the other two r'_a and r'_b can be trivially located by computing the opposite translation vectors. Then the current cell r can be marked as processed and put into $\mathbf{C}$, with the obtained translation vectors being put into $\mathbf{K}$. Cells r_a, r_b, r'_a and r'_b are put inside a set of unprocessed cells $\mathbf{U}$. In the second part of the pseudo-code, this neighbor searching process continues for each of the unprocessed neighboring cells in a recursive manner, until all cells are processed.

MSER may fail to identify cells due to image noise or partial occlusion. With the formed lattice, we can extend the lattice outward to check whether there is any missing cell. This can be done by extrapolating (or interpolating) the cell position from the external (or internal) boundary of the current lattice (Figures 3(c)&(d)). Note that there can be missing cells (holes) within the lattice. We can either interpolate the cell position from the whole lattice or infer the cell position using the translation vectors of the nearest neighbors. The matching criteria for missing cells can be more relaxed, by considering two cells as matched if their appearance dissimilarity $D_a < \tau_m$. In all our experiments, we set $\tau_m = 0.2$.

So far, we only connect the centers of MSERs to form lattice. Optionally, we can also form the dual lattice with the MSER centers coinciding with the lattice cell centers (Figure 3(e)). Such choice is left to users.

4.2 Summarizing S-region

Once all S- and NS- regions are identified, we can construct meshes that respect the shared boundaries (Figure 2). For the S-region, it is natural to construct a mesh according to the detected lattice (Figure 4(a)). In other words, the S-region is represented as a mesh dressed with a texture, which is the S-region image content.

Summarization To resize the S-region, we *summarize* it by simply removing or adding rows/columns of cells. However, directly modifying the mesh elements (quads) will change the mesh topology and introduce complication. The shapes of consecutive quads may change abruptly when a large number of quads are removed (Figure 4(b)). The boundary of the S-region may no longer confirm to that of the NS-region (Figures 4(b)&(c)).

Instead of modifying the mesh, we retain the mesh topology and only modify the texture content during the summarization (Figure 5). The summarization is performed in a rectified domain. By regarding each observed quad in the image (Figure 5(e)) as a perspective projection of a square, we can compute a transformation matrix $\mathbf{T}$, based on the vertex positions, to map each quad together with the texture content to a square (Figure 5(f)). The rectified cell content can be obtained by resampling the input image using $\mathbf{T}^{-1}$ (Figure 5(b)). In this rectified domain, the texture can be seamlessly resized by inserting and removing rows/columns of cells using graphcut [Kwatra et al. 2003] (Figure 5(c)). We choose to insert and remove rectified cells at the middle of rectified S-region, as content of cells near the boundary normally changes more vigorously. Hence this strategy can reduce the chance of visual artifact appearing. The number of columns (ΔC_n) and rows (ΔR_n) to remove or insert is automatically computed based on the target resizing ratio. We project the changes of image width and height onto the major axes of the lattice and obtain the two projected changes, Δw_p and Δh_p, one for each major axis. Then $\Delta C_n = \lfloor \Delta w_p / \overline{w_c} \rfloor$ and $\Delta R_n = \lfloor \Delta h_p / \overline{h_c} \rfloor$, where $\overline{w_c} \times \overline{h_c}$ is the average size of cells.

On the other hand, the mesh is topologically unchanged and only linearly scaled in the rectified domain according to the change of number of rows/columns (Figure 5(g)). As the texture content is changed, the texture coordinates of the mesh vertices have to be recomputed by linear interpolation. To map the rectified cells back to the image domain, we cannot simply map the scaled mesh quads in Figure 5(g) to the original quad in Figure 5(e) because the S-region in the image domain should also be scaled. For simplicity, the S-region is linearly scaled in the image domain according to the target resizing ratio (Figure 5(h)). The pair of scaled corresponding quads in image (Figure 5(h)) and rectified (Figure 5(g)) domains gives another transformation matrix $\mathbf{T}'$ to transform the cell content back to the image domain.

Careful readers may notice the texture content at the boundary changes as we retarget the mesh and texture content separately

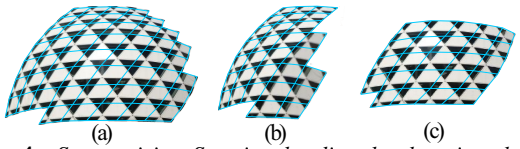


Figure 4: *Summarizing S-region by directly changing the mesh topology. (a) Original S-region. (b) Removing 3 columns in the middle. (c) Removing 1 column from the right, 2 columns from the left, and 3 rows from the top.*

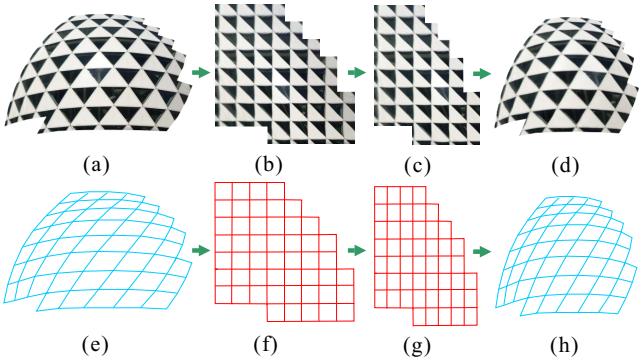


Figure 5: *S-region summarization. Bottom row: the mesh topology remains unchanged during the summarization. Top row: only the texture content is modified by removing columns/rows of cells in order to respect the symmetric structure.*

in the rectified domain (top right boundaries in Figures 5(b)&(c)). This implies the disagreement of image content between resized S- and NS- regions. We describe how to reduce such visual artifact in Section 4.4.

Illumination Adjustment Inserting or removing rows/columns of cells may introduce abrupt change of intensity among consecutive cells (Figure 6(a)). Such artifact is especially apparent when the underlying surface is curved, and at the boundary when the resized S-region is recombined with the resized NS-region. To solve the problem, we *relight* each cell in order to maintain the original lighting distribution over the S-region.

For each rectified cell content before summarization (Figure 5(b)), we compute its intensity mean μ_i and standard deviation σ_i. As all cell contents are statistically the same, the change of μ_i and σ_i over the lattice is mainly due to the illumination distribution. Our goal is to maintain such illumination distribution even after summarization. With the lattice of μ_i and σ_i, we resample the $\mu_i^{'}$ and $\sigma_i^{'}$ at the new cell centers (Figure 5(c)) using the thin-plate spline interpolation. Each cell can then be relit by shifting the intensity mean to $\mu_i^{'}$ and mapping the intensity variance to $\sigma_i^{'}$. To avoid blocking artifact at the cell boundary, we relight, not just the cell, but a region slightly larger than the cell, and perform a simple feathering to blend neighboring relit regions. Figure 6 compares the results with and without illumination adjustment.

4.3 Resizing NS-region

We resize the NS-regions using a warping technique. To implement this idea, we generate a triangular mesh for the given image using the Delaunay triangulation, where some vertices are randomly distributed, some are from the boundary of S-regions and some are uniformly sampled on straight lines. Straight lines are automatically detected by a Hough transform. To protect prominent objects, we require the triangles covering salient objects to undergo similarity transformations. To keep the straight lines from bending, we enforce edges on the same line to have similar slopes. The above mentioned constraints are formulated into energy terms and we minimize the objective function to determine the deformed mesh. Finally, the image is reconstructed by the linear interpolation of contents within triangles.

(a) Without relighting (b) With relighting

Figure 6: *Illumination adjustment on S-region.*

Let us denote by $\mathbf{M} = \{\mathbf{P}, \mathbf{E}\}$ a triangular mesh, where $\mathbf{P} = [\mathbf{p}_0^T, \mathbf{p}_1^T, ...\mathbf{p}_n^T]$, $\mathbf{p} = (x, y)$ is the vertex position and $\mathbf{E}$ denotes the set of edges on $\mathbf{M}$. In addition, we define $\mathbf{M}'$ and $\mathbf{P}'$ to be the deformed version of $\mathbf{M}$ and $\mathbf{P}$, respectively.

Shape Similarity We apply the conformal energy to maintain the similarities of the original and the deformed triangles. Under this constraint, the corresponding triangles are enforced to have the same shape while the sizes and orientations are allowed to be different. Formally, we transform the vertex $\mathbf{p}$ using

$$\begin{bmatrix} s & -r \\ r & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}, \tag{4}$$

where s and r denote the scaling and rotation factors, respectively, and $[u, v]$ is the translation vector. By putting vertices of the triangle f together and define

$$\mathbf{A}_f = \begin{bmatrix} x_{i_1} & -y_{i_1} & 1 & 0 \\ y_{i_1} & x_{i_1} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{i_3} & -y_{i_3} & 1 & 0 \\ y_{i_3} & x_{i_3} & 0 & 1 \end{bmatrix}, \quad \mathbf{b}'_f = \begin{bmatrix} x'_{i_1} \\ y'_{i_1} \\ \vdots \\ x'_{i_3} \\ y'_{i_3} \end{bmatrix}, \tag{5}$$

we can obtain the equation $\mathbf{A}_f[s, r, u, v]^T = \mathbf{b}'_f$ and this equation can be transformed into $\Omega_s = (\mathbf{A}_f(\mathbf{A}_f^T\mathbf{A}^T)^{-1}\mathbf{A}_f^T - \mathbf{I})\mathbf{b}'_f = 0$. Please refer to [Zhang et al. 2009; Wang et al. 2010] for more details.

Line Preservation We require edges that lie on the same straight line to have similar slopes. Specifically, we pick up two vertices $\mathbf{p}_{e0}$ and $\mathbf{p}_{e1}$ from the line which are closest to the line center and we wish the slopes of other edges on the line to be similar to the slope of $\mathbf{p}_{e0} - \mathbf{p}_{e1}$. The formal expression is given as:

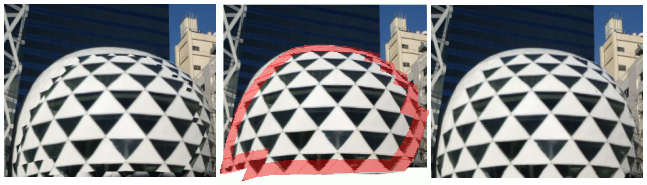
$$\Omega_\ell = \sum_{\{i,j\} \in \mathbf{E}_k} \left| (\mathbf{p}'_i - \mathbf{p}'_j) - \ell_{ij}(\mathbf{p}'_{e0} - \mathbf{p}'_{e1}) \right|^2, \tag{6}$$

where $\ell_{ij} = |\mathbf{p}'_i - \mathbf{p}'_j| / |\mathbf{p}'_{e0} - \mathbf{p}'_{e1}|$ and $\mathbf{E}_k$ are edges on the line k.

We solve for the deformed mesh by minimizing $\Omega_s + \Omega_\ell$ subject to the boundary and positional constraints. The positional constraints define the target resolution. The boundary constraints require boundary vertices moving along the respective lines such that the resized image can be retained as a rectangle. We compute this non-linear objective function in an iterative manner because the unknown variables in Equation 6 are correlated. Please refer to [Wang et al. 2008; Wang et al. 2009] for the implementation details.

4.4 Merge of S-Regions and NS-Region

Recall that we represent both S- and NS- regions using meshes. The correspondences between vertices from S- and NS- regions along the shared boundaries can be set up before the resizing. The pairs of boundary vertices are used to minimize the discontinuity artifacts during the warping of the NS-region. This is done by minimizing the distance between the corresponding vertices, and hence gives the objective, $\Omega_m = \sum_i^n |\mathbf{p}_{S,i} - \mathbf{p}_{NS,i}|^2$, where $\mathbf{p}_S$ and $\mathbf{p}_{NS}$ are the pair of corresponding vertices in S- and NS- regions, respectively, and $n = |\mathbf{p}_S| = |\mathbf{p}_{NS}|$ is the total number of pairs.



(a) (b) (c)

Figure 7: *Merge of S- and NS-regions. A ring of cells is extended to provide overlapping region for determining the seamless cut-path.*

However, the above optimization only binds the geometric boundaries of S- and NS- regions together. There may exist discontinuity of image contents between the S- and NS- regions, since these two types of regions are retargeted differently according to their own natures. As demonstrated in Figure 5(c), the image content on the boundary of the S-region may change after summarization. To reduce the discontinuity artifact, we grow the S-region outward by extrapolating one ring of cells (Figure 7(b)). We do so in order to generate an overlapping area with the NS-region, and determine a seamless cut path using graphcut [Kwatra et al. 2003] to hide the discontinuity artifact.

Both the cell contents and quads are extended to create the overlapping area. The cell content can be extrapolated by replicating the boundary cell content with its illumination adjusted according to the extrapolated μ_i' and σ_i'. The vertex positions of imaginary quads in the image domain are computed by extrapolation (Figure 7(b)). These imaginary quads are mainly used for computing the transformation matrices $\mathbf{T}'$s. They are not physically added to the mesh of a S-region. Figures 7(a)&(c) compare the results without and with hiding the discontinuity, respectively.

5 Results and Discussions

Figures 1, 8-14 shows our resized images. To validate our method, we tested it on images with symmetry structure on both planar (Figures 8, 9, 11-14) and curve surfaces (Figures 1 & 10), without (Figures 8-9, 13) or with perspective projection (Figures 11-12, 14). In all results, our summarization operator preserves the overall symmetry structure in terms of shape, perspective, and clarity. Planar (curved) structure remains planar (curved). The sizes of cells are reasonably preserved, without over-squeezing/over-stretching or uneven squeezing/stretching of cells over the lattice. Figure 12 shows the natural extensions of a house, both horizontally and vertically, via our summarization operator. We have implemented our method on a PC with 2 Xeon(TM) CPUs 3.20GHz, 12 GB RAM, and nVidia Geforce GTX 280 GPU with 1GB video memory. The timing statistics of resizing examples shown in this paper can be found in Table 1. Even for the "BMW" example (Figure 1) that contains three disjoint lattices (one for each cylindrical tower), the time for symmetry detection and lattice formation is very small and achieves a real-time rate.

Multiple S-regions Our method can simultaneously handle multiple disjoint S-regions when they are far away from each other. They are simply summarized independently and merged to the NS-region in a *single pass* (e.g. Figure 1). However, if two or more S-regions are too close to each other (their extended rings of cells overlap), they have to be summarized in *multiple passes*. In first pass, a S-region is selected while all other S-regions are ignored and regarded as part of the NS-region. The selected S-region is summarized and merged with the resized NS-region to produce an intermediate result $\mathbf{R_1}$. In the next pass, an unresized S-region is selected, summarized, and merged to $\mathbf{R_1}$ to produce an incremental result $\mathbf{R_2}$. This multi-pass operation continues until all S-regions are resized. In practice, we prefer to first resize larger S-regions than those smaller ones. Figures 13 and 14 show the examples of such multi-pass resizing.

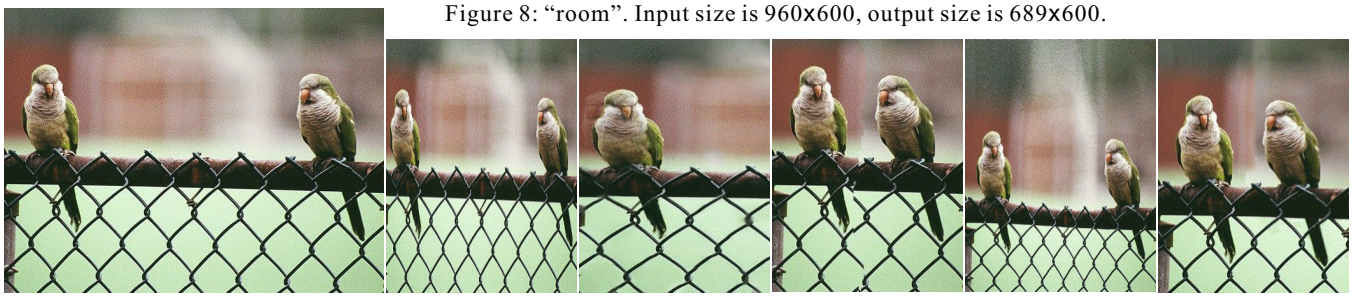
Visual Comparisons We compared our results to five state-of-the-art methods, including warping [Wang et al. 2008], multi-operator [Rubinstein et al. 2009], [Dong et al. 2009], Shift-Map [Pritch et al. 2009] and Patch-Match [Barnes et al. 2009]. The resized images produced by Shift-Map method are obtained via the online program http://www.cs.huji.ac.il/projects/shiftMapFlash/. Other compared results are obtained from the original authors. Figures 1, 8-11 and 14 show the comparisons. Since multi-operator and warping approaches utilize only pixels with low-level saliencies when resizing images, they can do nothing on the preservation of symmetry structures. Multi-operator may sometimes unevenly scale the symmetry structure (Figure 8(b)) and crop the surrounding contents (Figures 1(b) and 11(b)). Warping over-squeezes or stretches the homogeneous contents (Figures 1(d), 8(d) & 14(d)). In contrast, our method preserves the symmetry structure via summarization. Although Shift-Map and Patch-Match allow region-wise deletion or insertion, they have no knowledge of symmetry. Shift-Map may introduce obvious seams in some cases (Figures 1(c) and 9(d)); Patch-Match removes one of the birds and produces the ghost shadow in Figure 9(c). With the knowledge of symmetry, our method can reasonably summarize the lattice (curved and/or perspectively viewed) without uneven scaling of cells or obvious seams inside the lattice. Figure 14 shows an example being progressively reduced in size. Note that only our method produces reasonable results, in particular, at the drastic resizing ratio.

User study To further evaluate our method, we perform a user study to score the results from different methods. 70 subjects from different ages and backgrounds are invited to score 9 sets of the resized images. In the experiment, we showed the original image, our result and the image of a competitor, and ask which of the two resized images the participant prefers. Three methods, multi-operator, warping, and Shift-Map, are compared. Table 2 shows the statistics. Each row shows the competition between our method and the one indicating on that row. The "Mean of wins" indicates the number of times (or percentage) our method wins during the competitions. From the statistics, our method generally outperforms than all competitors, though participants may prefer the multi-operator and warping methods when the resizing ratio is moderate. This is because the homogeneous regions are not run out or the discontinuity artifacts are less accumulated. Since all methods can produce reasonable results in this scenario and participants may have different tastes, our method becomes less outstanding. On the other hand, our method is more preferred if the resizing ratio is extreme. This can be demonstrated by the comparison in Figure 14, in which [Dong et al. 2009] and [Wang et al. 2008] were compared. In their results, the windows are blurred and over-squeezed due to the drastic scaling.

Limitations and Discussions The ability of our symmetry detection strongly relies on the ability of MSER. If MSER fails to identify a potential cell, our method will not be able to form any lattice. Figure 15(a) shows one such example whose cells are obvious to human, but not to MSER. Another major limitation is on the lattice formation. We assume there is no self-occlusion inside the lattice. If self-occlusion exists, it breaks the lattice structure and terminates our lattice formation at the discontinuity (Figure 15(b)). Our method may not be able to handle lattice structure that is not very regular and nearly stochastic. We believe more sophisticated symmetry detection [Park et al. 2009] can be done, but at the expense of speed performance. We cannot handle the overlapped lattices. An image region can only belong to one lattice. Currently, we resize S- and NS- regions using the same ratio without considering their relative prominence. It may be possible to define a metric to account for the importance of symmetry as well as other image saliency, so that we can scale each S-region more effectively. Besides, cells in symmetry regions are mainly added or removed when resizing. The content within the cell is not modified. Although a certain degree of

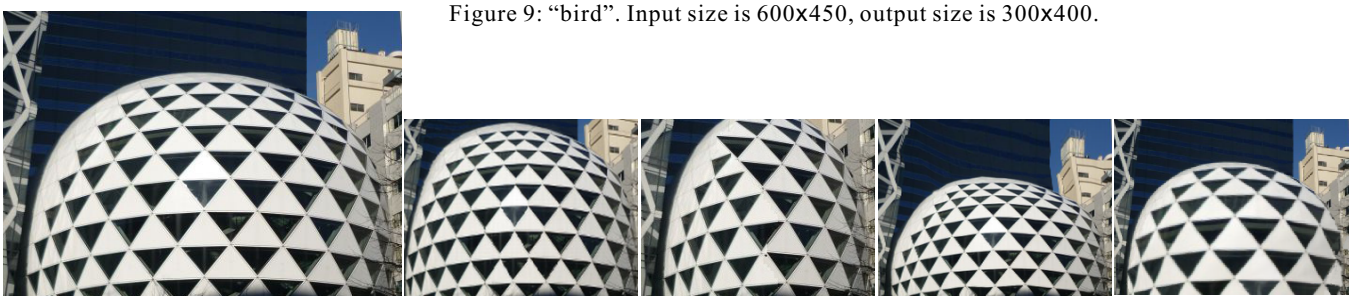
(a) Original (b) Multi-operator (c) Shift-Map (d) Warping (e) Ours

Figure 8: “room”. Input size is 960x600, output size is 689x600.



(a) Original. © IEEE 2009 (b) Multi-operator (c) Patch-Match (d) Shift-Map (e) Warping (f) Ours

Figure 9: “bird”. Input size is 600x450, output size is 300x400.



(a) Original (b) Multi-operator (c) Shift-Map (d) Warping (e) Ours

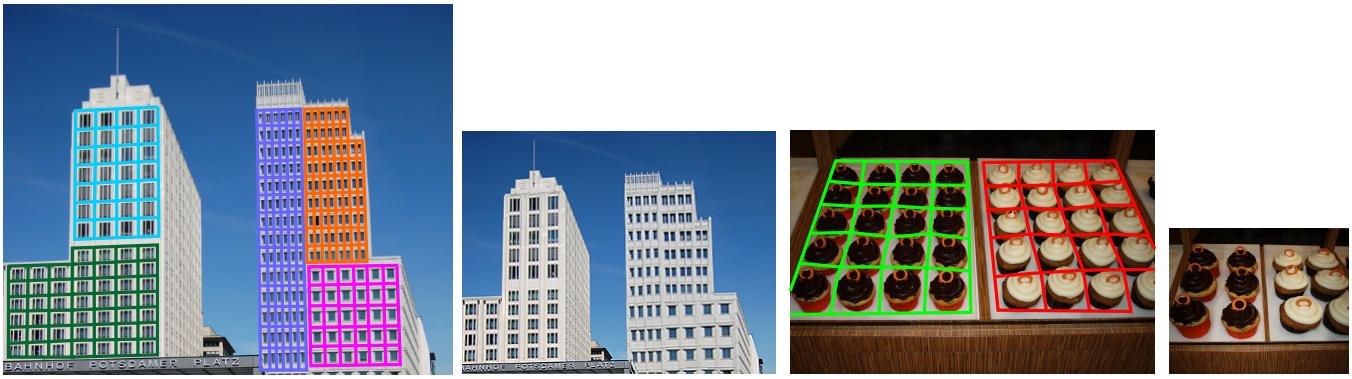
Figure 10: “ball”. Input size is 423x302, output size is 263x198.

(a) Original (b) Multi-operator (c) Shift-Map (d) Warping (e) Ours

Figure 11: “colosseo”. Input size is 768x1024, output size is 525x1024.

(a) Original, 800x533 (b) Extend 2 columns, 1095x533 (c) Extend 2 columns & 1 row, 1095x680

Figure 12: Extended “house”.



(a) Multiple adjacent lattices, 452x373 (b) Summarized, 316x244 (c) Two adjacent lattices, 1024x683 (d) Summarized, 501x410

Figure 13: *Multiple adjacent different kinds of lattices are summarized in a multi-pass manner.*

	Input size	Output size	No of disjoint lattices	Time of symmetry detection	Total time of resizing
Fig. 1	999x820	516x820	3	158 ms	331 ms
Fig. 8	960x600	689x600	1	62 ms	130 ms
Fig. 9	600x450	300x400	1	34 ms	80 ms
Fig. 10	423x302	263x198	1	27 ms	75 ms
Fig. 11	768x1024	525x1024	1	52 ms	115 ms
Fig. 12(b)	800x533	1095x533	1	46 ms	103 ms
Fig. 12(c)	800x533	1095x680	1	46 ms	107 ms
Fig. 13(b)	452x373	316x244	5	208 ms	563 ms
Fig. 13(d)	1024x683	501x410	2	175 ms	264 ms
Fig. 14(b) left	1024x625	768x625	6	233 ms	601 ms
Fig. 14(b) center	1024x625	512x625	6	233 ms	606 ms
Fig. 14(b) right	1024x625	256x625	6	233 ms	629 ms

Table 1: *Timing statistics.*

Methods	Mean(%)of wins	Std.dev.	95% confidence interval	
			Lower Bound	Upper Bound
Multi-operator	6.03 (67.00%)	1.91	5.58	6.48
Shift-Map	7.07 (78.56%)	1.42	6.74	7.40
Warping	5.44 (60.44%)	2.05	4.96	5.92

Table 2: *User study.*

warping is caused by the forward and backward transformation between the original image and the rectified domain, the cell content is generally unchanged. In the future, more sophisticated resizing treatment to the cell content can be considered, especially when the cells are enlarged.

6 Conclusion

We demonstrate that by understanding one more piece of semantics, we can open one extra space for image resizing. Although computational understanding of general image semantics is hard, analysis and acquisition of symmetry is feasible and practical for resizing applications. The new space for resizing also leads to a new issue, the seamlessness of image content at the boundary, as general images contain both symmetry regions and non-symmetry region, and these two types of regions have to be resized differently. We proposed a real-time symmetry detection system and methodology to minimize the discontinuity artifact. Currently, we only tackle the translational symmetry, our next step is obviously its extension to other types of symmetry, e.g. rotational symmetry.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. We are also grateful to all participants who took part in the user study. We thank the flickr contributors for sharing their images. This project is supported in part by the Hong Kong Research Grants Council of under General Research Funds (Project No. CUHK417107), the CUHK Group Research Scheme 2009, the Landmark Program of NCKU Top University Project (contract B0008), and the National Science Council (contract NSC-99-2221-E-006-066-MY3), Taiwan.

References

Ahuja, N., and Todorovic, S. 2007. Extracting texels in 2.1d natural textures. In *ICCV*, 1–8.

Avidan, S., and Shamir, A. 2007. Seam carving for content-aware image resizing. *ACM Trans. Graph. 26*, 3, 10.

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph 28*, 3.

Canny, F. J. 1986. A Computational Approach to Edge Detection. *IEEE Trans. PAMI 8*, 6, 679–698.

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

Cheng, M.-M., Zhang, F.-L., Mitra, N. J., Huang, X., and Hu, S.-M. 2010. Repfinder: Finding approximately repeated scene elements for image editing. *ACM Trans. Graph 29*, 3.

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

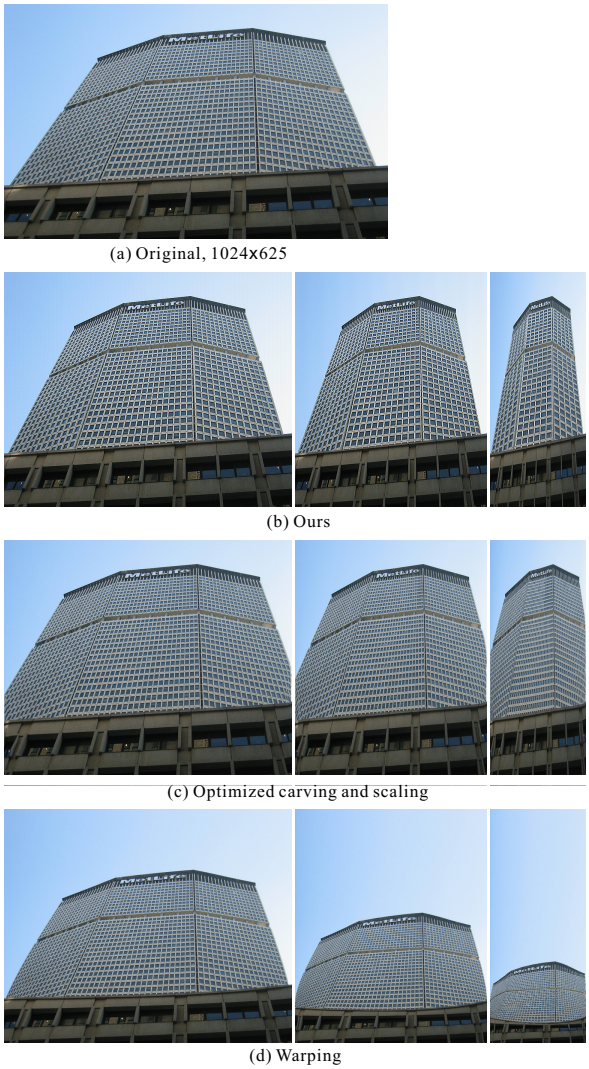
Comaniciu, D., and Meer, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 5, 603–619.

Dong, W., Zhou, N., Paul, J.-C., and Zhang, X. 2009. Optimized image resizing using seam carving and scaling. *ACM Trans. Graph. 28*, 5, 1–10.

Gal, R., Sorkine, O., and Cohen-Or, D. 2006. Feature-aware texturing. In *EGSR '06*, 297–303.

Grünbaum, B., and Shephard, G. C. 1986. *Tilings and patterns*. W. H. Freeman & Co., New York, NY, USA.

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



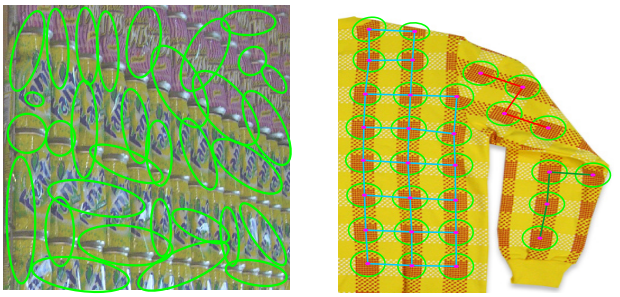
(a) Original, 1024x625

(b) Ours

(c) Optimized carving and scaling

(d) Warping

Figure 14: *Progressively reduced "metlife." From left to right: 768×625, 512×625, 256×625.*



(a) Fail to identify cells. (b) Occlusion breaks the lattice.

Figure 15: *Failure cases.*

110.

Hays, J., Leordeanu, M., Efros, A. A., and Liu, Y. 2006. Discovering texture regularity as a higher-order correspondence problem. In *ECCV (2)*, 522–535.

Kwatra, V., Schödl, A., Essa, I. A., Turk, G., and Bobick, A. F. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003 22*, 3 (July), 277–286.

Leung, T. K., and Malik, J. 1996. Detecting, localizing and grouping repeated scene elements from an image. In *ECCV (1)*, 546–555.

Lin, W.-C., and Liu, Y. 2007. A lattice-based mrf model for dynamic near-regular texture tracking. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 5, 777–792.

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

Liu, Y., Collins, R. T., and Tsin, Y. 2004. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 3, 354–371.

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2, 91–

Matas, J., Chum, O., Urban, M., and Pajdla, T. 2002. Robust wide baseline stereo from maximally stable extremal regions. In *In British Machine Vision Conference*, 384–393.

Park, M., Brocklehurst, K., Collins, R. T., and Liu, Y. 2009. Deformed lattice detection in real-world images using mean-shift belief propagation. *IEEE Trans. Pattern Anal. Mach. Intell. 31*, 10, 1804–1816.

Pritch, Y., Kav-Venaki, E., and Peleg, S. 2009. Shift-map image editing. In *ICCV'09*.

Rubinstein, M., Shamir, A., and Avidan, S. 2008. Improved seam carving for video retargeting. *ACM Trans. Graph. 27*, 3, 16.

Rubinstein, M., Shamir, A., and Avidan, S. 2009. Multi-operator media retargeting. *ACM Trans. Graph. 28*, 3, 23.

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

Schattschneider, D. 1978. The plane symmetry groups: Their recognition and notation. *The American Mathematical Monthly 85*, 6, 439–450.

Shi, J., and Tomasi, C. 1994. Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition*, 593–600.

Simakov, D., Caspi, Y., Shechtman, E., and Irani, M. 2008. Summarizing visual data using bidirectional similarity. In *CVPR '08*.

Suh, B., Ling, H., Bederson, B. B., and Jacobs, D. W. 2003. Automatic thumbnail cropping and its effectiveness. In *Proceedings of UIST*, 95–104.

Wang, Y.-S., Tai, C.-L., Sorkine, O., and Lee, T.-Y. 2008. Optimized scale-and-stretch for image resizing. *ACM Trans. Graph. 27*, 5, 118.

Wang, Y.-S., Fu, H., Sorkine, O., Lee, T.-Y., and Seidel, H.-P. 2009. Motion-aware temporal coherence for video resizing. *ACM Trans. Graph. 28*, 5.

Wang, Y.-S., Lin, H.-C., Sorkine, O., and Lee, T.-Y. 2010. Motion-based video retargeting with optimized crop-and-warp. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH) 29*, 3.

Wolf, L., Guttmann, M., and Cohen-Or, D. 2007. Non-homogeneous content-driven video-retargeting. In *ICCV '07*.

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

Zhang, G.-X., Cheng, M.-M., Hu, S.-M., and Martin, R. R. 2009. A shape-preserving approach to image resizing. *Computer Graphics Forum 28*, 7, 1897–1906.