TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 08/11 pp452-462
Volume 17, Number 4, August 2012

Feature Preserving Milli-Scaling of Large Format Visualizations[*]

Yunwei Zhang, Aidong Lu**, Jian Huang[†]

Department of Computer Science, University of North Carolina, Charlotte, NC 28223, USA;
[†] Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996, USA

Abstract: Ultra-scale data analysis has created many new challenges for visualization. For example, in climate research with two-dimensional time-varying data, scientists find it crucial to study the hidden temporal relationships from a set of large scale images, whose resolutions are much higher than that of general computer monitors. When scientists can only visualize a small portion ($< 1/1000$) of a time step at one time, it is extremely challenging to analyze the temporal features from multiple time steps. As this problem cannot be simply solved with interaction or display technologies, this paper presents a milli-scaling approach by designing downscaling algorithms with significant ratios. Our approach can produce readable-sized images of multiple ultra-scale visualizations, while preserving important data features and temporal relationships. Using the climate visualization as the testing application, we demonstrate that our approach provides a new tool for users to effectively make sense of multiple, large-format visualizations.

Key words: visualization scaling; feature preserving; large scale visualization

Introduction

The difficulty we encounter in the current influx of large complex data has prompted many novel initiatives in visualization research. In this paper, we address the challenge where limited display resolutions do not even allow one pixel per datum, as demonstrated by Fig. 1. Unlike many other technological constraints, the gap between limited display resolutions versus increasing data sizes will likely persist in the foreseeable future or continue to widen.

For example, satellite observation data crucial for validating today's climate modeling research, such as NASA's MODerate resolution Imaging

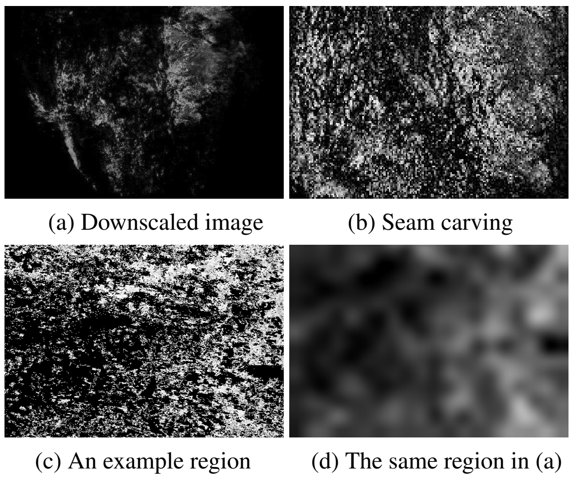
(a) Downscaled image (b) Seam carving
(c) An example region (d) The same region in (a)

Fig. 1 An example to show that downscaling and seam carving cannot preserve object shapes and textures, which are important aspects of perceptual image features.

Received: 2012-05-01; revised: 2012-06-10

[*] Co-authors Zhang and Lu were supported by DHS Center of Excellence - Natural Disasters, Coastal Infrastructure and Emergency Management (DIEM) and DOE (No. DE-FG02-06ER25733). Work by co-author Huang was in part funded through the Institute of Ultra-Scale Visualization (http://www.ultravis.org) under the auspices of the SciDAC program within the U.S. Department of Energy (No. DE-FC02-06ER25778).

** To whom correspondence should be addressed.
E-mail: aidong.lu@uncc.edu; Tel: 1-704-687-8570

Spectroradiometer dataset (MODIS[1]), is publicly available. In the 500-meter resolution MODIS data set, the entire globe is covered by a spatial grid of 86 000 by 43 000. In the meantime, datasets at ultra high resolution are also appearing in communities such as digital pathology, where two-dimensional (2-D) datasets are being produced at 40 000 by 40 000 resolution. It is very difficult to visualize such ultra high resolution data at full scale using today's display technology.

In this work, we develop a visualization method, which we call "milli-scaling", to downscale an image by roughly a factor of 30×30 (both horizontal and vertical dimensions). Using milli-scaling methods, we can visualize MODIS caliber datasets at resolutions which are commonly available today. In addition, we design our milli-scaling approach to handle multiple, instead of just one, large-format visualizations. The large-format visualizations are considered together as a collection, which can represent geo-spatial data captured by satellites at different instances of time or pathological data recorded from sequential cutting planes.

Specifically, our approach downscales a small collection of ultra-scale images by converting them consistently for effective data comparison and analysis. Our method leverages both image-space features as well as collection-wide relationships. We first divide the image space into regions with a gap searching algorithm that measures imagery saliency and data similarities. A space reorganization method is then applied to emphasize important regions and maintain meaningful background. The important regions are further downscaled through a multi-scale texture mapping method under the constraint of a feature-aligned mesh to preserve perceptually important image features. We demonstrate our results at several commonly used resolutions. Both example results and discussions are provided to show that our milli-scaling approach can achieve arbitrary downscaling suitable for data visualization purposes.

The main contribution of this paper is our milli-scaling approach for visualizing ultra-scale images from a 2-D data collection at readable resolutions. Our approach allows arbitrary downscaling and preserves perceptually important image features with a mesh constrained texture mapping approach. Different from image editing techniques, our approach considers image features as well as collection-wide data relationships. Our automatic downscaling results can be used to visualize, compare, and document 2-D data from different fields or time steps directly.

1 Related Work

1.1 Image resizing

Image resizing tools uniformly resize an image to a target size and are available in many image processing applications. To obtain image features, simply applying existing techniques like cropping or down-sampling does not often work. Especially when they are applied to images with complex objects, important image features may be lost or deformed, thereby changing their original meanings.

Content-aware resizing has gained popularity lately. These approaches generally use important measures like image gradients, saliency and entropy, as well as high level cues such as face or motion detection. For example, Chen et al.[2] presented a visual attention model for resizing an image. Setlur et al.[3] presented an image retargeting method that resized an image by constructing a topologically constrained epitome of an image. Avidan and Shamir[4] presented a seam carving approach that supported content-aware image resizing for both reduction and expansion by repeatedly carving out or inserting seams to retarget the image to a new size. Later, Rubinstein et al.[5] showed that seam carving did not preserve object shapes or appearances and presented a multi-operator approach that combined seam carving with cropping and scaling to find an optimal resizing solution. Simakov et al.[6] presented an approach based on optimization of a bi-directional similarity measurement. Dong et al.[7] designed a method based on optimization of a well-defined image distance function to preserve both important regions and the global visual effect. Wang et al.[8] presented a scale-and-stretch warping method that preserved visually prominent features by diverting the distortion due to resizing to image regions with homogeneous content. This approach has also been extended to volumetric datasets recently[9].

Resizing techniques have also been explored for videos, such as video carving[10], video retargeting[11], improved seam carving[12], integrated content-aware[13], and motion-aware constraints[14].

1.2 Milli-scaling related work

Due to the degree of downscaling, the problem of milli-scaling cannot be solved by general image resizing approaches effectively. There are very few works that address similar problems. For example, Liu et al.[15] presented an automatic image browsing strategy with an image attention model to display large pictures on mobile devices. Liu and Gleicher[16] presented a fisheye-view warping method to use image salience and object recognition to preserve information content while displaying an image on a small display. Setlur et al.[3] pointed out that a comprehensible and size varying approach was suitable for display-critical applications. Similar to this approach, we also segment an image into regions, identify important regions, and treat the background differently. Some of the recently published approaches, such as the scale-and-stretch[8] and the bidirectional similarity methods[7], could also resize an image to arbitrary sizes.

However, none of the previous approaches has really downscaled an image to the extent of milli-scale addressed in this paper. Also, since many of them iteratively remove a line each time, they are either impossible or inefficient for this application. The measurement of image features is also different, as milli-scale downscaling has to remove significant image portions. Generally these image editing approaches handle images from 1024×754. Our data resolution is $20\,705 \times 12\,000$.

2 Approach and Application Overview

Milli-scaling is different from general image and video scaling problems from two aspects. First, due to the extreme degree of milli-scaling, it is inevitable that a significant portion of image details may be lost or deformed during the downscaling process. Second, milli-scaling is for a group of images. A good downscaling algorithm in such a scenario should preserve key information in the input images, such as important details from regions-of-interest. It should also maintain meaningful background for understanding the global picture. This requires the identification of important regions that are related to significant data features and different downscaling methods for treating important and background regions. Our milli-scalling approach consists of the following stages.

Selection of important regions We segment the image space considering data features and relationships among multiple images with a gap searching method. An interactive function is also provided for users to specify interesting regions manually.

Reshaping of image space We reorganize the image space as foreground and background through selecting important regions and calculating their ideal target resolutions.

Image scaling Important and background regions are handled separately. We design a feature preserving method to downscale each important region to its target resolution. The background is generated with an image warping and morphing method to provide meaningful context.

Our test cases are collections of images resulting from the NASA MODIS data with a flexible query method[17], which allows the exploration of various temporal relationships by testing queries abstracted from different events. Our inputs can be viewed as small groups of grey scale images in the resolution of $20\,705 \times 12\,000$. According to an input query, a score is calculated for each pixel by measuring the correspondence degree of local data to the represented event. Generally, regions with high values draw more attention from users. Therefore, the pixel values can indicate important locations on the image plane. One special feature of our input images is that most of them are composed of discrete points, since features are usually sparse by nature. The image groups we use in our experiment include no more than eight images from different time steps. Preserving individual data features as well as the temporal relationships are both important to this application.

3 Milli-Scaling Approach

This section describes the details of our milli-scaling approach. Let's denote the input image collection as $I = \{M_1, M_2, \cdots, M_n\}$, where n is the number of images.

3.1 Region division with gap searching

The first step of milli-scaling is to find a suitable division of the image space, so that we can preserve data details from one or several important regions and generate the background as context. Since our objective is to explore temporal events or relationships among the input image collection, the region division should be operated on the input image set, instead of an individual image. Also, an important region is defined as a sub-image space that contains crucial and continuous data patterns or significant temporal events.

We have developed two methods for identifying important regions. First, an interactive method is provided for a user to specify important regions by manually drawing a map over their interested regions. Second, we design an automatic method to divide the image space according to data features and relationships. Both methods enable easy selection of important regions with various shapes and sizes. The following describes the details of our gap searching algorithm for automatic region division.

Since many datasets used in scientific visualization are composed of discrete points, it is difficult to distinguish local seams and global gaps. Instead of identifying important regions directly, we design a succinct algorithm that searches for obvious gaps between regions with continuous local and temporal patterns. This can produce a near global "optimized" division result for various inputs. Our approach consists of three steps: we first generate a significance map through measuring image and temporal features; then a gap search algorithm is performed based on the significance map; finally a list of important regions is derived according to the region gaps.

3.1.1 Measuring pixel significance

To measure pixel significance, we calculate a significance map by combing two data features: gradients for representing the degree of data changes and score values for indicating the correspondence degrees to a querying event. Since the original data is often composed of discrete points, the gradient map is calculated from a downscaled image that averages pixel values, which measure the data changes better. From our experiments, different scales do not matter much to the final results. Therefore, we use $1/8 \times 1/8$ of the original resolution (closest to the typical resolution of computer monitors) for the significance map. This can significantly improve the performance of our next step, search of obvious gaps. Also, a Gaussian smoothing operator is applied prior to the gradient calculation to reduce noise in data. Correspondingly, we generate a score map at the same scale. Since each pixel represents a region in the original data, we choose the maximum score value as the result for the score map. The significance map is measured by multiplying the values from the gradient map and score map. Figure 2 shows examples of significance maps where regions with high score values and variations are more obvious. To calculate pixel significance for the input set, we normalize significance maps for individual image before summing them so that they have equal effects on the final region division. The final significance map for every pixel p is calculated as: $\mathrm{sig}(p) = \sum_{i=1}^{n} (\mathrm{gradient}(p) \times \text{maximum score}(p))$.

To incorporate data relationships among the input set, we measure pixel similarities for comparing the temporal trend at each pixel location. Specifically, a $1 \times n$ vector $\mathbf{tr}$ is generated for each pixel by concatenating the score values at the location from all the images according to their temporal order. During the division, the pixel similarities can be used in the same way as the significance map, meaning that pixels with similar temporal trends should be considered as the same region.

3.1.2 Searching for gaps

The region division is achieved through utilizing the significance map and pixel similarities to search for obvious gaps. We define gaps as the horizontal or

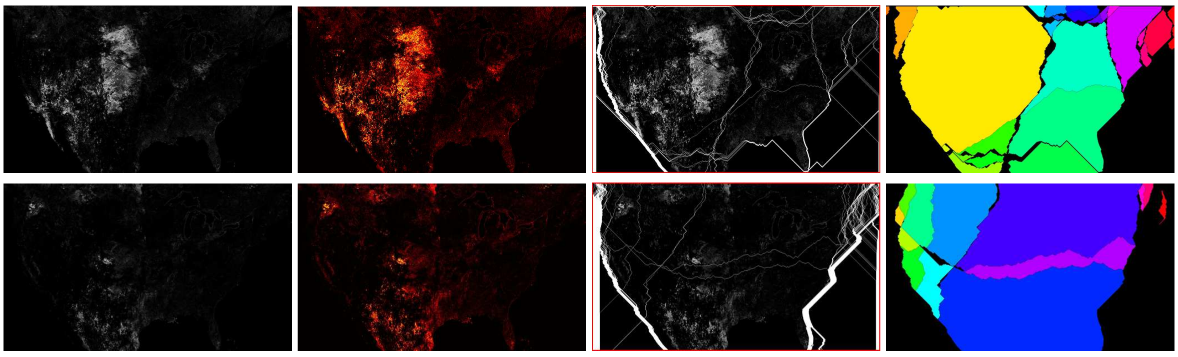

Fig. 2 Examples of region division. Our gap searching method divides the image space by identifying obvious gaps between local and temporal continuous regions. We add a red frame to the results of segmenting seams to clarify the locations of some white seams around the image boundaries.

vertical passes with the minimum energy sums, similar to the definition used in the seam carving approach[4]. This design is from the observation that adjacent regions are often separated by a gap with different data properties, which can be measured with the differences of values from the maps of final significance and temporal trends. The combination of horizontal and vertical seams allows complex gaps that are not available in the original seam carving approach.

Specifically, we search for obvious gaps by modifying the seam carving approach from two aspects. First, we combine differences of adjacent significance values sig and temporal trends **tr** as follows, which extends the energy definition of the original seam carving approach.

$$s^* = \min E(s) = \min \sum (\text{sig}_1 - \text{sig}_2) \times (|\mathbf{tr}_1 - \mathbf{tr}_2|) \tag{1}$$

This modification incorporates the requirements of milli-scaling and achieves simultaneous region division among the input image collection.

Since the gaps between regions can face any direction in an image, our second modification is to search for optimal seams from all four edges of the image space. All the seams are combined in one map with a binary mask, as shown in the third column of Fig. 2. Original seam carving uses an iterative process to remove a horizontal or vertical seam each time. The seam is located by finding a pass with the minimum accumulative energy value. Our improvement avoids the inertia of the algorithm that tends to be bounded by the seam direction and starting edge, which can still be seen by the white margins on the left of the image space. This modification also allows complex gaps that consist of both horizontal and vertical seam segments. To prevent an image from being overly divided, we terminate the algorithm when the accumulated energy value of a seam is beyond a threshold of 1/10 of the corresponding dimension.

3.1.3 Generating important regions

After finding important gaps, we generate a list of regions with the Connected Component Labeling (CCL) approach[18]. Specifically, we first mark all the gaps with 1 s and the others with 0 s. The CCL approach is then applied to identify each region with continuous 0 s. We calculate the importance of each region by summarizing all the scores inside the region and sort them according to their importance values. Further, we reduce fragmented areas by treating all the gaps as empty regions and discard regions with small sizes or importance values. To assist the exploration of the input set, we also provide interactions for users to select important regions and adjust their degrees of importance.

It is worth mentioning that our problem is different from object recognition. What we are looking for is a quick space segmentation approach that can divide ultra-scale images into smaller pieces. Due to the discrete property of our datasets, it is difficult to use approaches like region growing or texture analysis for good segmentation results. Also, we need to consider the similarity of temporal trends during the segmentation so that our results can keep important data relationships. The presented region division approach fits our query data quite well. As shown in Figs. 2 and 3, this method can locate several significant gaps between regions and segment the image space succinctly.

3.2 Reorganization of image space

With the list of segmented regions, we select important regions and determine their level-of-detail given a target resolution. The image space is then reorganized as foreground and background regions, which are treated differently to emphasize important data features. For arbitrary downscaling, we automatically select all the parameters, including the number of important regions and their final resolutions.

3.2.1 Determining parameters for arbitrary downscaling

We summarize the ideal reorganization of the target image space with the following criteria. First, important regions are scaled equally in both dimensions to best

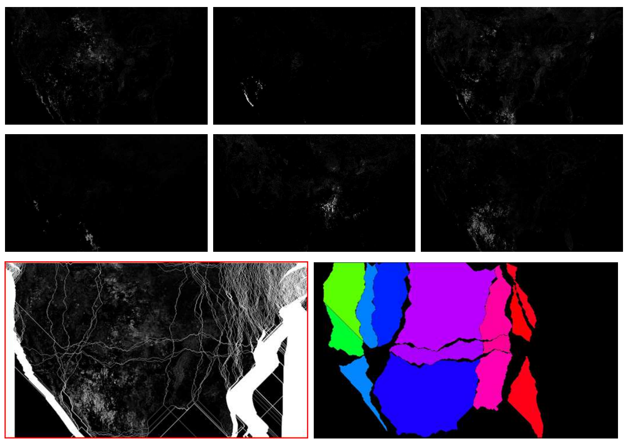

Fig. 3 Segmentation example from multiple images. The top rows are a set of 6 images. The bottom row shows the segmenting seams and resulting region map.

preserve their perceptual features. Second, important regions are relocated on the target image without much overlapping. Third, it is ideal to keep some gaps between important regions to provide background context. Fourth, important regions should be scaled as large as possible while satisfying the previous criteria.

We first sort all the regions according to their degree of importance, calculated as $\text{IM(region)} = \sum_{p \in \text{region}} \text{score}(p)$. The degree of importance is also used to calculate the relative scales between important regions. For example, the final resolutions of two regions i and j will be proportional to $\text{IM}(i) : \text{IM}(j)$.

We design our approach to adapt its parameter selection automatically according to the target resolution by changing the number of important regions. The results concerning working memory from cognitive science[19] have clearly indicated that the human mind can only process the information from a very limited number of objects at the same time. Referring to the number of objects used in these studies and considering that our regions are often complex, we choose only one region when the downscaling ratio is smaller than $1/500$ (e.g., for handheld devices) and four regions at most when it is larger than $1/50$ (e.g., for computer monitors). A linear function is used with these two key points to calculate the number of allowable regions for arbitrary target resolution. This selects 1-4 regions for typical computer monitors, which is consistent with our design.

We then calculate the final resolution of selected important regions. Since the proportions of important regions have been determined, we just need to find a final scale Fc that can satisfy the criteria of our ideal choices. A simple iterative process can be used to calculate this scale quickly. Starting from a large number, such as the largest scale to fit a region on the target image, represented as $\text{Fc} = \min\{\text{DimX}_{\text{space}}/\text{DimX}_{\text{region}}, \text{DimY}_{\text{space}}/\text{DimY}_{\text{region}}\}$, we perform a halfway search process until $\sum \text{DimX}_{\text{region}} < \text{DimX}_{\text{space}}$ and $\sum \text{DimY}_{\text{region}} < \text{DimY}_{\text{space}}$. For simplicity, we use the bounding box of each region as its boundary. The final resolution of a selected important region is then settled as $\text{Fc} \times \text{IM(region)}$.

3.2.2 Generating background

After selecting important regions, we reorganize the target image space to emphasize important regions. We first locate important regions on the target image and use the feature preserving approach from the next section to downscale them. To preserve the relative locations of important regions, we use their center locations as directly downscaled. Since we keep the same proportion of X and Y dimensions for an important region, with the center and final resolution, we can locate all the pixels belonging to this region. We also allow users to adjust the locations and final resolutions of important regions to achieve their desired effects. In case important regions overlap, which mostly happen around region corners, we apply the image quilting algorithm[20] to synthesize overlapped regions. This algorithm finds a pass with minimum accumulated energy through overlapped regions, which is very similar to the basic idea of gap searching. Therefore, we can use the same energy function defined in Eq. (1).

Background regions are used to provide the context of important regions. It is desirable to keep the correct correlation of important regions to their surroundings, so that the background can still provide useful information to identify locations and global data changes. We use an image warping and morphing method to achieve this. With the boundaries of important regions on the source image space, we first build a triangle mesh to divide the background region. As shown in Fig. 4, every vertex of this source triangle mesh is located on either the boundaries of important regions or the edges of the image space. We can easily map the source triangle mesh to the target image space according to the reorganization result. Then, given the coordinates of vertices for both a target triangle and a source triangle, we can get the affine transformation matrix $\boldsymbol{H}$[21]. To maintain a smooth pattern, we apply the inverse affine transformation $\boldsymbol{H}^{-1}$ to each point in the target triangle to locate its projection on the source image. Then, as described in Refs. [22,23], we calculate its intensity by interpolating the nearest four pixels on the source image bilinearly.

The usage of a transformation matrix can guarantee the smooth transition of downscaling ratios between important regions as well as important regions to the space boundary. As shown in Fig. 4, the examples of our background generation show that the background region in each triangle is correctly warped and preserves the continuous texture appearance.

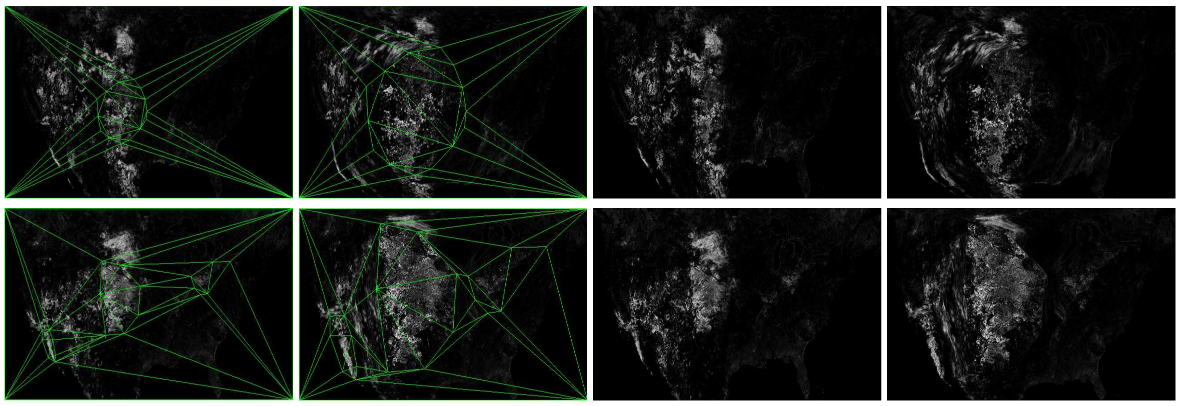

Fig. 4 Example results of background generation to provide context for important regions. The first two images compare the background meshes before and after deformation. The second two images show the directly downscaled images and our results.

3.3 Perceptually feature preserving downscaling

Since milli-scaling is designed to assist the visual analysis of data distribution and temporal relationships, it is essential to preserve perceptually important image features, such as object boundaries, layouts, and textures. As Fig. 1 shows, simple downscaling can lose much detailed information, while image editing techniques like seam carving cannot always preserve object shapes. Our approach is to build a feature-aligned triangle mesh to constrain image deformation during the downscaling process and a multi-resolution texture mapping method to preserve similar object appearance.

3.3.1 Building a feature-aligned triangle mesh

We first build a feature-aligned triangle mesh so that we can constrain texture deformation under the downscaling process. Our mesh is generated from two sets of feature points which are selected through detecting object boundaries and local data statistics. Specially, we use either the Sobel filter[24] or canny edge detector[21] on the binary mask image to get boundary points, which controls the overall shape of the region. The points shown on the second row of Fig. 5 are the response to the filters — the locations of local maxima as the feature points. The local texture feature points are detected using variance gradients and histograms. Feature points are also located by finding the local maxima. Due to the problem of milli-scaling, a multi-level detection procedure is used to search for feature points on different scales, including $1/4 \times 1/4$, $1/8 \times 1/8$, and $1/16 \times 1/16$. The locations where score values change significantly on different scales are also included, since they indicate pixels that are mostly affected by the downscaling process. All the feature points from different scales are summarized on the target resolution. Starting from the smallest resolution, we record the locations of feature points. Extra points, that are adjacent to already identified feature points, are removed since they may produce poor triangles.

A feature-aligned mesh is then built with all the feature points using the delaunay triangulation algorithm[25], which tends to avoid skinny triangles by maximizing the minimum angle of all the triangles in the mesh. Since this mesh captures the relative locations and distances of feature points, controlling its deformation can preserve a portion of important image features, including object boundaries and layout. What's more, when a suitable number of feature points is selected, this mesh divides a whole important region into small triangle pieces with simple point distributions. It allows us to divide a milli-scaling problem for a large region into independent small scale tasks. For visualizing important regions, since we keep the proportion of X and Y dimensions, we restrict the location of this feature-aligned mesh to best preserve object appearances.

3.3.2 Preserving textures for perception

Another important step to preserve perceptually important image features is through texture mapping. Since our input images are often composed of discrete points, the image textures can indicate important

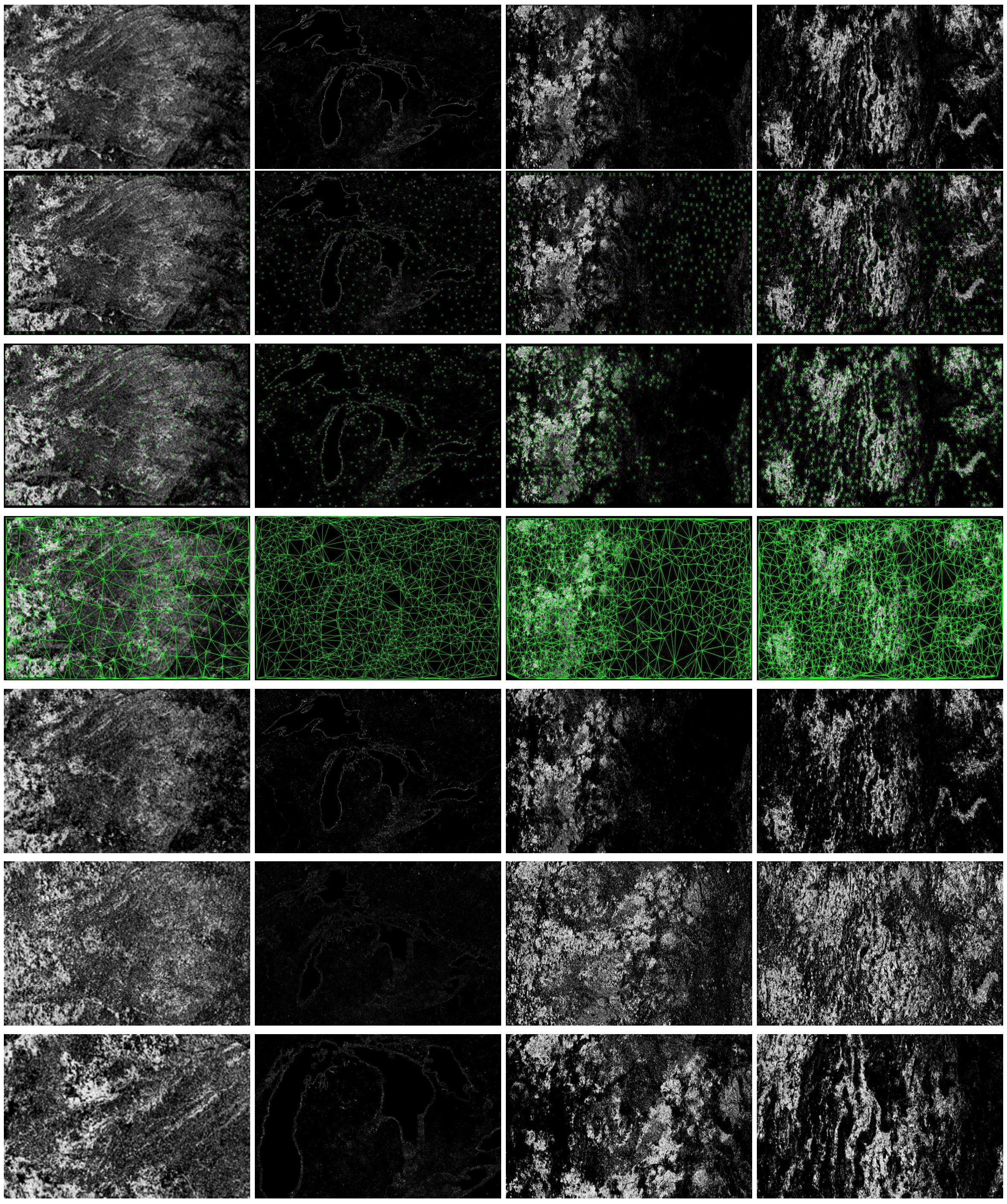

Fig. 5 Example results of our feature preserving downscaling method. Each column shows the directly downscaled image, feature points from boundary detection, feature points from local data statistics, feature-aligned triangle mesh, our downscaling result, seam carving result, and cropping result. These four example sets demonstrate that our approach can handle various regions, ranging from sharp edges to smooth homogeneous regions.

features of data distribution. Therefore, it is crucial for milli-scaling results to retain similar textures as the original images. We can use texture mapping to achieve this.

Specifically, every triangle T_{target} in the downscaled image has a counter-part in the original resolution T_{original}. In order to collect the texture for T_{target} from T_{original}, we allow a triangle T_{search} with the same size of T_{target} to sweep across T_{original}. During the sweeping process, a similarity measurement of the triangles is recorded into a score value map $\text{ScoreMap}_{\text{target}}$. The similarity measurement takes two aspects into consideration. The first aspect measures the constituent similarity of score values. Since the measurement is taken under triangles with different resolutions, we use Bhattacharyya distance. Given two histograms of score values h_{original} from T_{original} and h_{search} from T_{search}, let B be the number of bins in the histogram. The Bhattacharyya coefficient is

$$\text{BhaC}(T_{\text{original}}, T_{\text{search}}) = \sum_{j=1}^{B} \sqrt{\sum h_{\text{original}_j} \sum h_{\text{search}_j}} \tag{2}$$

The Bhattacharyya distance between T_{original} and T_{search} is

$$\text{distB}(T_{\text{original}}, T_{\text{search}}) = -\ln(\text{BhaC}(T_{\text{original}}, T_{\text{search}})) \tag{3}$$

The second aspect measures the distribution similarity of two triangle textures, which is the Euclidean distance of the two triangles. Let P be the number of pixels in T_{target}.

$$\text{distE}(T_{\text{original}}, T_{\text{search}}) = \sqrt{\sum_{j=1}^{P} (T_{\text{original}}(j) - T_{\text{search}}(j))^2} \tag{4}$$

All the Euclidean distances are normalized to the range $[0, 1]$ and the score map $\text{ScoreMap}_{\text{target}}$ is generated by multiplying the values from the two aspects of similarities as follows:

$$\text{ScoreMap}_{\text{target}}(i) = \text{distE}(T_{\text{original}}, T_{\text{search}}) \cdot \text{distB}(T_{\text{original}}, T_{\text{search}}) \tag{5}$$

Therefore the final mapping is achieved by searching for the minimum distance score in the ScoreMap:

$$t^* = \text{argmin}(\text{ScoreMap}_{\text{target}}) \tag{6}$$

The sampling frequency can be used to adjust the performance by modifying the number of T_{search}.

We further design a multi-resolution texture mapping method to achieve arbitrary downscaling. The original image is downscaled at several resolutions, $1/2 \times 1/2$, $1/4 \times 1/4$, $1/8 \times 1/8$, until it reaches the target image size. Correspondingly, a feature-aligned triangle mesh is scaled for every resolution. We perform the above texture mapping for all adjacent resolutions, starting from the largest $1/2 \times 1/2$. Due to the original image size, this step is time-consuming and can be pre-processed. Given a target resolution, we can use the closest large scale to perform the texture mapping method. This can reduce the number of possible choices and accelerate the texture mapping process. Also, with two close scales, we can find good matchings to both our score map and the histogram of score values.

Figure 5 provides several results of our approach. We compare them to the results of direct downscaling, seam carving, and cropping. Our results show that direct downscaling removes data details, seam carving changes object shapes, and cropping misses important regions. Overall, our approach produces the best results by preserving both object shapes and textures.

4 Discussion

4.1 Performance evaluation

We ran our algorithm on a computer with an Intel Xeon 5560 2.80 GHz processor and 3 GB RAM. All of our images are 20 705 × 12 000 and each set contains 2-8 images. The memory requirement is directly linked to the size of the images we are working with. Since we take a multilevel approach in extracting textures from large sized images and patch them to a target image space, the memory size should at least match that of our texture source.

The performances of several components of our approach, including region segmentation, space reorganization, and background warping, can be finished within several minutes with the setup described above. Table 1 provides example performance of our

Table 1 Gap searching performance

Resolution	SeamsCountX	SeamsCountY	TimeSpent (s)
750 × 1295	45	5	40.3812
750 × 1295	90	10	79.1216
1500 × 2589	45	5	358.7152
1500 × 2589	90	10	569.5020

Tabel 2 Background mapping performance

Resolution	TimeSpent (s)
334 × 576	1.0621
467 × 810	1.2231
750 × 1295	1.8573
1500 × 2589	3.2774

gap searching algorithm for region division and Table 2 provides performance for the background generation method.

The performance of feature-preserving scaling is highly affected by the choice of parameters, like the radius for finding local maxima of feature points and the number of feature points. Table 3 provides example results. The step frequency is a parameter to control the number of T_{search} and can be used to adjust the running time according to performance requirements. Generally smaller step size results in better texture mapping effects.

4.2 Discussions

The judgement of interactivity should be relevant to the data size. For ultra-scale datasets, only the I/O performance will be below interactive speeds. The running time of our approach is neglectable compared to the time to generate the datasets.

It is worth mentioning that milli-scaling is different from previous image editing techniques. First, no image editing approach has ever downscaled images to such a level, taking special care to preserve data features. Second, image editing techniques also purposely distort homogeneous regions, while we try to preserve important features of object appearances for visualization purposes. Third, to preserve temporal relationships, our milli-scaling approach considers information from multiple images, while image editing techniques just handle one image.

Our approach can be used for arbitrary downscaling. The pipeline of our approach, region division, independent region scaling, and space reorganization is scalable to the size of input images. We also automate parameter selections according to the target image resolution. Our approach can also be applied to 2-D time-varying or multi-field datasets directly by using an image saliency map to replace the significance map.

As general image editing techniques, failure cases may happen. For example, it is possible that our region division stage results in only one important region, whose size is close to the original image. This differs from our ideal space reorganization scenarios which enlarge important regions significantly compared to their surroundings. Also, during our feature-preserving scaling process, we may fail to find suitable texture pieces, which can be improved by adding additional feature points.

Table 3 Texture mapping performance

StepFreq	TriangleNum	Area	TimeSpent (s)	Time/pixel (s)
25	16 350	59 697	1939.8672	0.0324
25	9734	32 581	1039.0925	0.0319
50	16 350	59 697	4021.6841	0.0674
50	9734	32 581	2106.6024	0.0619

5 Conclusions and Future Work

This paper presents the problem of milli-scaling for ultra-scale 2-D data visualization. We have designed a new approach to downscale a set of images to arbitrary target resolutions. Our approach preserves various data features, including important regions, perceptual image features, data distribution, and region relationships among multiple images, which are not available from image editing techniques. Our results can be used by users to browse, compare ultra-scale 2-D datasets, and document their findings directly.

In addition to the arbitrary milli-scaling of 2-D data, our approach has two potential usages in volume visualization and multi-field data visualization. First, since many large scale datasets are in 3-D, we are very interested in extending our approach for time-varying 3-D datasets. It is likely that we can achieve a 3-D arbitrary down-scaling approach with a similar pipeline which produces volumes in sizes that can be visualized interactively with available volume visualization methods. The 3-D relationships need to be considered as well. Second, we are interested in extending our approach for multi-field data visualization, in which more thought will be given to allow users to study the relationships among different data fields effectively.

References

[1] MODIS. The moderate resolution imaging spectroradiometer. http://modis.gsfc.nasa.gov, 2010.

[2] Chen L, Xie X, Fan X, et al. A visual attention model for adapting images on small displays. *Multimedia Systems*, 2003, **9**(4): 353-364.

[3] Setlur V, Takagi S, Raskar R, et al. Automatic image retargeting. In: Proceedings of the 4th International Conference on Mobile and Ubiquitous Multimedia. Christchurch, New Zealand, 2005: 59-68.

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

[5] Rubinstein M, Shamir A, Avidan S. Multi-operator media retargeting. *ACM Trans. Graph.*, 2009, **28**(3): 1-11.

[6] Simakov D, Caspi Y, Shechtman E, et al. Summarizing visual data using bidirectional similarity. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. Anchorage, Alaska, USA, 2008: 1-8.

[7] Dong W, Zhou N, Paul J C, et al. Optimized image resizing using seam carving and scaling. *ACM Trans. Graph.*, 2009, **28**(5): 1-10.

[8] Wang Y S, Tai C L, Sorkine O, et al. Optimized scale-and-stretch for image resizing. *ACM Trans. Graph.*, 2008, **27**(5): 1-8.

[9] Wang Y S, Wang C, Lee T Y, et al. Feature-preserving volume data reduction and focus+context visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2011, **17**(2): 171-181.

[10] Chen B, Sen P. Video carving. In: Proceedings of Eurographics. Crete, Greece, 2008.

[11] Liu F, Gleicher M. Video retargeting: Automating pan and scan. In: Proceedings of the 14th Annual ACM International Conference on Multimedia. Santa Barbara, CA, USA, 2006: 241-250.

[12] Rubinstein M, Shamir A, Avidan S. Improved seam carving for video retargeting. *ACM Trans. Graph.*, 2008, **27**(3): 1-9.

[13] Krahenbuhl P, Lang M, Hornung A, et al. A system for retargeting of streaming video. *ACM Trans. Graph.*, 2009, **28**(5): 1-10.

[14] Wang Y S, Fu H, Sorkine O, et al. Motion-aware temporal coherence for video resizing. *ACM Trans. Graph.*, 2009, **28**(5): 1-10.

[15] Liu H, Xie X, Ma W Y, et al. Automatic browsing of large pictures on mobile devices. In: Proceedings of the Eleventh ACM International Conference on Multimedia'03. Berkeley, CA, USA, 2003: 148-155.

[16] Liu F, Gleicher M. Automatic image retargeting with fisheye-view warping. In: Proceedings of ACM UIST. Seattle, WA, USA, 2005: 153-162.

[17] Glatter M, Huang J, Ahern S, et al. Visualizing temporal patterns in large multivariate data using textual pattern matching. *IEEE Transactions on Visualization and Computer Graphics*, 2008, **4**(6): 1467-1474.

[18] Samet H, Tamminen M. Efficient component labeling of images of arbitrary dimension represented by linear bintrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1988, **10**(4): 579-586.

[19] Bly B M, Rumelhart D E. Cognitive Science (Handbook of Perception and Cognition). Academic Press, 1999.

[20] Efros A A, Freeman W T. Image quilting for texture synthesis and transfer. In: Proceedings of SIGGRAPH 2001. New York, NY, USA, 2001: 341-346.

[21] Trucco E, Verri A. Introductory Techniques for 3-D Computer Vision. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[22] McMillan Jr L, Pizer R S. An image-based approach to three-dimensional computer graphics. Technical Report, 1997.

[23] Wolberg G. 360 image morphing: A survey. *The Visual Computer*, 1998, **14**: 360-372.

[24] Canny J. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1986, **8**(6): 679-698.

[25] Lee D T, Schachter B J. Two algorithms for constructing a delaunay triangulation. *Int. J. Computer Information Sci.*, 1980, **9**: 219-242.