*Science in China Series F: Information Sciences*


www.scichina.com
info.scichina.com
www.springerlink.com

# Real-time content-aware image resizing

HUANG Hua[1†], FU TianNan[1], ROSIN Paul L.[2] & QI Chun[1]

[1] School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China;

[2] School of Computer Science, Cardiff University, Cardiff CF24 3AA, UK;

**Content-aware image resizing is a kind of new and effective approach for image resizing, which preserves image content well and does not cause obvious distortion when changing the aspect ratio of images. Recently, a seam based approach for content-aware image resizing was proposed by Avidan and Shamir. Their results are impressive, but because the method uses dynamic programming many times, it is slow. In this paper, we present a more efficient algorithm for seam based content-aware image resizing, which searches seams through establishing the matching relation between adjacent rows or columns. We give a linear algorithm to find the optimal matches within a weighted bipartite graph composed of the pixels in adjacent rows or columns. Therefore, our method is fast (e.g. our method needs only about 100 ms to reduce a 768×1024 image's width to 1/3 while Avidan and Shamir's method needs 12 s). This supports immediate image resizing whereas Avidan and Shamir's method requires a more costly pre-processing step to enable subsequent real-time processing. A fast method such as the one proposed will be also needed for future real-time video resizing applications.**

content aware, image resizing, video resizing, real time, matching

## 1 Introduction

With the development of network and computer hardware technology, many mobile media devices such as cell phones, PDAs or MP4s are developing quickly, and have a large impact in people's lives. There is now a large consumer market in which people use these devices to browse images and play videos. However, a problem that needs to be overcome is that different devices often have different resolutions, and so it is necessary to resize images or videos efficiently and effectively in order to adapt well to these devices.

Although uniform scaling can be used to change the size of images and videos easily, the results are often unsatisfactory. Since all parts of the image are treated equally, it is impossible to preserve certain important areas, which may therefore become unacceptably degraded at the lower resolution. Moreover, changing the aspect ratio will distort the image content, which is generally undesirable.

An approach taken by many authors is to perform cropping, which involves finding the best rectangular sub-window in the image. Generally, this is determined using a salience map such as that proposed by Itti and Koch[1] which measures local con-

Received July 14, 2008; accepted October 23, 2008
doi: 10.1007/s11432-009-0041-9
†Corresponding author (email: huanghua_xjtu@hotmail.com)
Supported by National Natural Science Foundation of China (Grant Nos. 60575002 and 60641002)

trast in intensity, color, orientation, etc. at several scales and then combines them into a single measure. Some authors use alternative or additional saliency measures provided by face, skin and text detection. Having computed salience, cropping can be carried out either by selecting the smallest rectangle that contains a specified proportion of the image's salience, or else given a fixed size rectangle, its position is found to maximize the contained salience. Examples of this approach are given by Sue et al.[2] and Chen et al.[3]. Sue[2] noted that if the saliency threshold approach is used, then the proportion of the image saliency contained by the cropping rectangle should be adapted to the image content, and described an algorithm for selecting the proportion at a value such that adding further small amounts of additional saliency requires a large increase in the rectangle size.

Ciocca et al.[4] proposed applying different cropping methods to different types of images. They trained a classifier to identify images as being one of the three categories: landscapes, close-ups, or other. Landscapes were resized rather than cropped; close-ups were cropped using Itti and Koch's salience map; remaining images were cropped using face and skin salience maps. Instead of relying on bottom-up computation of salience maps, Santella et al.[5] used eye tracking to identify regions of interest and find the optimal cropping window with respect to these regions. However, it is not convenient to rely on eye tracking for most applications.

Although resizing images using cropping as described above will not cause distortion, it is difficult to guarantee that all important areas are included in the cropping window. Thus some important areas may be discarded completely, which leads to an unsatisfactory result. This led Liu and Gleicher[6] to describe a different approach that they called focus+context. They attempted to retain a high level of detail at areas of interest, while still retaining all of the remainder of the image to provide context, but with less fidelity. First, following standard image cropping practice, the smallest rectangle was found to enclose any identified objects and also to contain a fixed proportion of the image saliency. The image was then warped to give the effect of a fish-eye lens, centred on the rectangle. Whilst this distorted the image, Liu and Gleicher argued that it is better to alert the viewer that part of the image has been modified. In comparison, with cropping, it may not be obvious to the viewer that parts of the image have been removed.

Setlur et al.[7] took an ambitious approach to retargetting images that attempted to include all important regions without distorting the image. This required extracting regions of interest (ROI), and then pasting them back onto their corresponding positions in the downsized background. Regions are found by performing image segmentation (using the mean shift algorithm followed by region merging), which requires several parameters. ROIs are selected from the segments using saliency maps. The remaining regions make up the background, and gaps are filled using inpainting. A problem with this approach is that even state of the art segmentation algorithms are unreliable, which compromises the robustness of the proposed system.

Avidan and Shamir[8] recently provided a new approach to image resizing. Rather than cropping, they adjusted the image size by adding or removing seams. Seams were defined as 8-connected paths of pixels running along the image from top to bottom or left to right, containing exactly one pixel in each row or column respectively. They used dynamic programming to find seams which pass through unimportant areas in order to preserve the important areas to the greatest extent. An advantage of this approach is that it will not cause obvious distortion when changing the aspect ratio. The results are impressive, but because the method uses dynamic programming, it is slow. Processing a single image often needs several seconds (for example, it needs about 5 s to reduce a $600\times800$ image's width to half). Although they provide a solution to facilitate fast image resizing, it still requires a slow pre-processing step. This time may be unacceptable, especially when many images need to be resized.

A fast content-aware resizing algorithm is given in this paper. We find all seams in parallel by performing matching between the rows (or columns)

in the image, which was proposed by Avidan and Shamir[8]. The main contribution of this paper is that we provide a novel method to establish the matching relations that is very fast. We can obtain results which are similar to standard seam carving's (see Figure 1), but only about 100 milliseconds are needed to process a 768× 1024 image.

Such a fast method is useful for several applications. For example, some web servers may need a content-aware image resizing application to adjust the size of images embedded in web pages when the pages need to be displayed at different resolutions and/or aspect ratios. If the servers are providing personalized page content that is dynamically generated, then suitable resizing of images is even more important. Since the visitor traffic of web pages can be huge, an efficient algorithm is necessary.

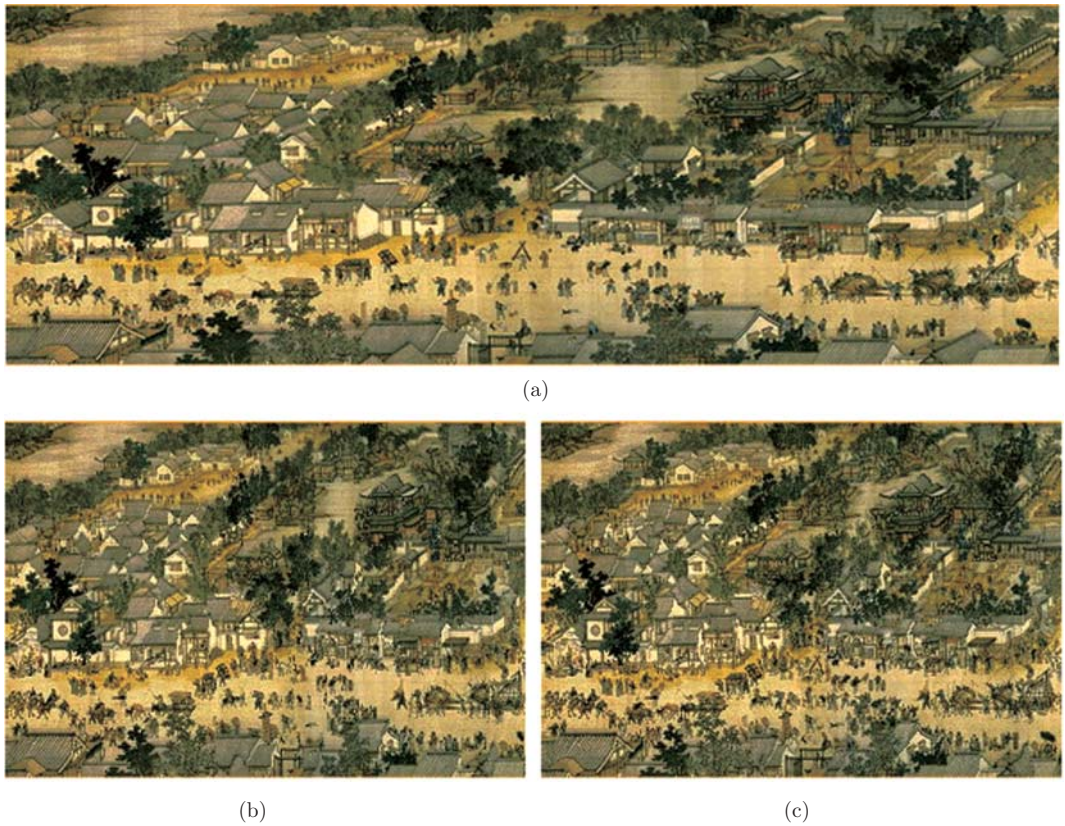
We introduce how to find seams through matching in section 3 and give our algorithm for solving the matching problem in section 3.1. In section 3.2, we discuss the definition of the edge's weights between pixels in adjacent rows (or columns). Finally, we give several experimental results in section 4.

## 2 Seam carving

Standard seam carving[8] computes the energy of each pixel in the image first before performing resizing. The energy indicates the importance of pixels. Avidan and Shamir suggest several energy functions, the simplest being image gradient magnitude, which is defined as follows:

$$e(I) = \left|\frac{\partial}{\partial x}I\right| + \left|\frac{\partial}{\partial y}I\right|. \quad (1)$$

Not only can the gradient magnitude be calculated more easily and quickly than the other energy functions, but the quality of the results are comparable.



(a)

(b) (c)

**Figure 1** A part of image "Qingming shanghe tu (Ascending the River at Qingming Festival)". We use standard seam carving and our approach to reduce the image width to half. Note that results of these two methods are similar. (a) Original image; (b) our approach; (c) standard seam carving.

After defining the energy of each pixel, they can remove or duplicate the lower energy pixels in order to decrease or increase the image size. Of course, when taking into consideration the continuity of image content, they do not arbitrarily choose the pixels to be removed or duplicated, but perform removal or duplication in terms of seams. In an $n \times m$ image $I$, a vertical seam is defined to be

$$s^x = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n,$$
$$\text{s.t. } \forall i,\ |x(i) - x(i-1)| \leqslant 1, \quad (2)$$

where $x$ is a mapping from $[1, \ldots, n]$ to $[1, \ldots, m]$. Similarly, a horizontal seam is defined as follows:

$$s^y = \{s_j^y\}_{j=1}^m = \{(j, y(j))\}_{j=1}^m,$$
$$\text{s.t. } \forall j,\ |y(j) - y(j-1)| \leqslant 1, \quad (3)$$

where $y$ is a mapping from $[1, \ldots, m]$ to $[1, \ldots, n]$.

The energy of a seam is defined as the sum of all pixels' energy in this seam. Avidan and Shamir[8] used dynamic programming to look for the seam with minimal energy. Taking the vertical seam as an example, they establish the cumulative energy matrix $M$ first:

$$M(i,j) = e(i,j) + \min\{M(i-1, j-1), M(i, j-1), M(i+1, j-1)\}, \quad (4)$$

and then find the pixel in the matrix $M$'s last row whose cumulative energy is minimal. After that, they backtrack from this pixel to obtain an optimal vertical seam. Finally, the image width can be adjusted by removing or duplicating this optimal vertical seam.

Although using dynamic programming to find the seam is effective, it is slow, requiring $O(nm^2)$ or $O(n^2m)$ time for all vertical or horizontal seams. In order to accelerate the speed, Avidan and Shamir[8] proposed creating a multi-sized image by a pre-processing step, which stores the order of all seams in the image and can be used to retarget images continuously in real time. However, horizontal and vertical seams can collide in more than one place. For such inconsistent seams, their removal may destroy the seams in the other direction. Therefore this method only supports one dimensional seam removal, while in the other direction only degenerate seams (i.e. rows or columns) are supported. Furthermore, the pre-processing also needs a lot of computing time (several seconds for a single image) and memory space (hundreds of kbytes for a single image).

In their appendix, Avidan and Shamir[8] give an alternative method to overcome the problem of inconsistent seams, which more properly supports two dimensional resizing. First, they establish matching relations between all neighboring pairs of rows in the image. Next, by imposing some constraints, they find consistent seams (such that every horizontal seam intersects all vertical seams, every vertical seam intersects all horizontal seams and each intersection includes exactly one pixel), which enables seam removal in both dimensions. The drawback of their approach is that they optimize the matches between rows using the Hungarian algorithm[9], which is slow. Thus they are still unable to provide proper two-dimensional resizing in real time without substantial pre-processing.

## 3 Algorithm

Our algorithm is now presented. Without loss of generality, only reducing or enlarging the width of an image is described. Adjusting the height of an image is similar.

We treat resizing of images as a three-step process. First of all, the energy of each pixel in the image is computed using the energy function (1). Secondly, the matching relations are established between all neighboring pairs of rows in order to find all seams in parallel. Finally, we compute the energy of each seam and remove or duplicate them in ascending order.

Establishing the matching relations between neighboring pairs of rows is the key to the entire process. This matching consists of finding a one-to-one correspondence between pixels in neighboring pairs of rows. That is, every pixel has exactly one matching pixel in the row above and also in the row below. After establishing matching relations between all neighboring rows, starting from any pixel in the first row (or last row), we can obtain a pixel sequence $S$ through the image by following the matches across the rows. In an $n \times m$ image $I$, assume $I(m(i,j), j+1)$ to be the matching pixel of $I(i,j)$ in row $j+1$. Then if the following constraint

holds:

$$\forall i \in [1, m] \wedge j \in [1, n),\ |m(i, j) - i| \leqslant 1, \quad (5)$$

we can see that $S$ is just a vertical seam according to the definition (2). Because a pixel sequence $S$ can always be found starting at any pixel in the first row (or last row), we can find all vertical seams after establishing the matching relations between all neighboring rows.

Now the problem is how to establish the matching relations between neighboring pairs of rows. Obviously, the matching relations between neighboring pairs of rows are not unique, although some constraints do exist, namely (5). In order to identify preferred matching relations, we introduce a weight for each edge that connects two arbitrary pixels in adjacent pairs of rows (defined in section 3.2). After this, we also define a criterion to evaluate various matching relations (in this paper, this criterion is defined to be the sum of the weights of matching edges, and the bigger the better). Now the problem becomes finding the optimal matches within a weighted bipartite graph, for which the standard method is the Hungarian algorithm. However, the time complexity of this algorithm is $O(m^3)$, which is too expensive to allow real-time image resizing. But because of constraint (5), the weighted bipartite graph is a special one, and so there is no need to solve it using a general method. We can greatly reduce the computational complexity by taking constraint (5) into account. Our method for matching is described in section 3.1.

After establishing the matching relations between all neighboring pairs of rows, we find the sequence from each pixel in the first row, also calculating the summed energy of pixels in each sequence. Finally, we sort the vertical seams in ascending order according to their energy, and remove or duplicate them in order to adjust the width of an image.

### 3.1 Optimal matching of a special bipartite graph

Without loss of generality, for an $n \times m$ image, we solve the optimal matching of the bipartite graph which is composed of pixels in row $k$ and row $k + 1$. Suppose that $w(i, j)$ represents the weight of the edge which connects pixel $I(i, k)$ with pixel $I(j, k + 1)$. Let $F(x)$ represent the sum of the weights of matching edges in the optimal matching of the sub-graph which is composed of the first $x$ pairs of pixels.

We consider the pixel $I(m, k)$ first (the red pixel in Figure 2). Because of the constraint (5), there are only two possible cases: pixel $I(m, k)$ either matches with pixel $I(m, k + 1)$ or else with pixel $I(m - 1, k + 1)$. So the optimal matching of the bipartite graph also has two possible cases. We consider these two cases respectively.

Firstly, when pixel $I(m, k)$ matches with pixel $I(m, k + 1)$ (see Figure 2(a)), the edge which connects $I(m, k)$ with $I(m - 1, k + 1)$ and the edge which connects $I(m-1, k)$ with $I(m, k+1)$ cannot be matching edges any longer (because each pixel can only connect to one matching edge). Consequently the matching of the sub-graph composed of the first $m - 1$ pairs of pixels must be a subset of the optimal matching of the full graph. In this case, the sum of the weights of matching edges is $F_1 = F(m - 1) + w(m, m)$.

Secondly, when pixel $I(m, k)$ matches with pixel $I(m - 1, k+1)$ (see Figure 2(b)), pixel $I(m - 1, k)$ must match with pixel $I(m, k+1)$, since there is no other pixel which satisfies the constraint (5) that can match with pixel $I(m, k + 1)$. Similar to the first case, the four edges shown by black dashed lines in Figure 2(b) cannot be matching edges and the matching of the first $m-2$ pairs of pixels must be optimal. In this case, the sum of the weights of matching edges is $F_2 = F(m - 2) + w(m, m - 1) + w(m - 1, m)$.

Because the optimal matches are those which make the sum of the weights of matching edges maximal, we can determine which case above is the optimum by comparing $F_1$ and $F_2$. More specifically, if $F_1 > F_2$, the first case holds. So we have $F(m) = F_1 = F(m - 1) + w(m, m)$, and pixel $I(m, k)$ matches with pixel $I(m, k+1)$. In contrast, if $F_1 < F_2$, then the second case holds. So we have $F(m) = F_2 = F(m-2)+w(m, m-1)+w(m-1, m)$, pixel $I(m, k)$ matches with pixel $I(m-1, k+1)$ and pixel $I(m - 1, k)$ matches with pixel $I(m, k + 1)$.

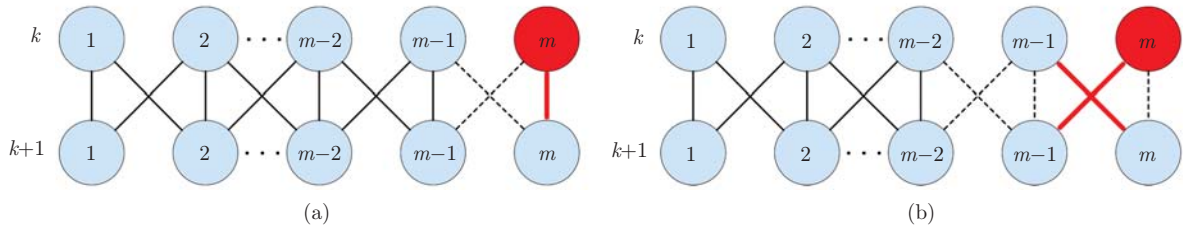



**Figure 2** The two possible configurations of matches. The circles represent the pixels. The red thick lines represent the matching edges. The black solid lines represent the possible matching edges. The black dashed lines represent the impossible matching edges. We just draw the edges that satisfy the constraint (5).

We can summarize these two cases as the following equation:

$$F(m) = \max\{F(m-1) + w(m,m), \\ F(m-2) + w(m,m-1) \\ + w(m-1,m)\}. \quad (6)$$

Testing the value of $F(m)$ determines which of the above two cases applies. Extending this to a more general situation, for the first i pairs of pixels, we have

$$F(i) = \max\{F(i-1) + w(i,i), \\ F(i-2) + w(i,i-1) \\ + w(i-1,i)\}. \quad (7)$$

Similarly, we can use $F(i)$ to determine for the subgraph which is composed of the first $i$ pairs of pixels which case holds.

We define $F(-1) = F(0) = 0$ and $w(0,1) = w(1,0) = -\infty$, and use eq. (7) to solve $F(i), 1 \leqslant i \leqslant m$. Now, to solve the optimal matching of a bipartite graph which is composed of m pairs of pixels, we first test $F(m)$ in order to determine which case holds. If it is the first case, then pixel $I(m,k)$ has been matched, and now $F(m-1)$ is tested on the remaining pixels. Otherwise pixels $I(m,k)$ and $I(m-1,k)$ have been matched and $F(m-2)$ is tested. Iterating this process reduces $x$, the number of pixels still to be matched. Eventually, when $x \leqslant 0$ the process terminates and the matching pixel of each pixel in row $k$ has been found. That is, we have established the matching relation between row $k$ and row $k$+1 and this matching relation is optimal.

### 3.2 Definition of edge weight

The discussion about solving the optimal matching of a weighted bipartite graph above is based on the assumption that all edges' weights have already been defined. Now we discuss how to define the weight function.

We again take pixels in row $k$ and row $k$+1 as an example. Suppose that $w(i,j)$ represents the weight of the edge which connects pixel $I(i,k)$ with $I(j,k+1)$ and $e(i,j)$ represents the energy of pixel $I(i,j)$. The energy of a matching edge is defined as the sum of the energy of two pixels attached to it. That is, if we use $E_i$ to represent the energy of the matching edge which connects to pixel $I(i,k)$, we have $E_i = e(i,k) + e(m(i,k),k+1)$.

When we reduce the image size via removing seams, we always choose the seam that has the lowest energy in order to preserve the energy of the image as much as possible. If we only consider the pixels in a given adjacent pair of rows, when removing two pixels attached to a matching edge we attempt to preserve the energy of these two rows, that is, to make the total energy of remaining matching edges maximal. Notice that no matter how they are matched, the total energy of all matching edges between two rows is a constant equal to $\sum_{i=1}^{m} e(i,k) + e(i,k+1)$. One strategy is to maximize the variance of the energy of matching edges in order to make the removed matching edges have a lower energy and also make the remaining matching edges have a higher total energy. Assume that the average energy of matching edges is $\overline{E}$. The variance of the energy of matching edges is defined as follows:

$$\sigma^2 = \frac{1}{m} \cdot \sum_{i=1}^{m} (\bar{E} - E_i)^2 \\ = \frac{1}{m} \cdot \sum_{i=1}^{m} \bar{E}^2 - \frac{2}{m} \cdot \bar{E} \cdot \sum_{i=1}^{m} E_i$$

$$
\begin{aligned}
&+\frac{1}{m}\cdot\sum_{i=1}^{m}E_i^2\\
&=C_1-C_2+\frac{1}{m}\cdot\sum_{i=1}^{m}[e(i,k)\\
&\quad+e(m(i,k),k+1)]^2\\
&=C_1-C_2+\frac{1}{m}\cdot\sum_{i=1}^{m}[e^2(i,k)\\
&\quad+e^2(m(i,k),k+1)]\\
&\quad+\frac{2}{m}\cdot\sum_{i=1}^{m}e(i,k)\cdot e(m(i,k),k+1)\\
&=C_1-C_2+C_3+\frac{2}{m}\\
&\quad\cdot\sum_{i=1}^{m}e(i,k)\cdot e(m(i,k),k+1).
\end{aligned}
$$

Since $C_1, C_2, C_3$ are all constants, we should maximize $\sum_{i=1}^{m} e(i,k)\cdot e(m(i,k),k+1)$ in order to maximize $\sigma^2$. Thus, we define the weight as follows:

$$
w_1(i,j)=\begin{cases}e(i,k)\cdot e(j,k+1), & |i-j|\leqslant 1,\\ -\infty, & \text{otherwise.}\end{cases}\tag{8}
$$

Examining the results, we find that the vertical seam found under the weight definition $w_1$ is nearly straight (see Figure 3(d)). For certain images, using this definition may cause bad results (see Figure 3(b)). This is mainly because we only consider

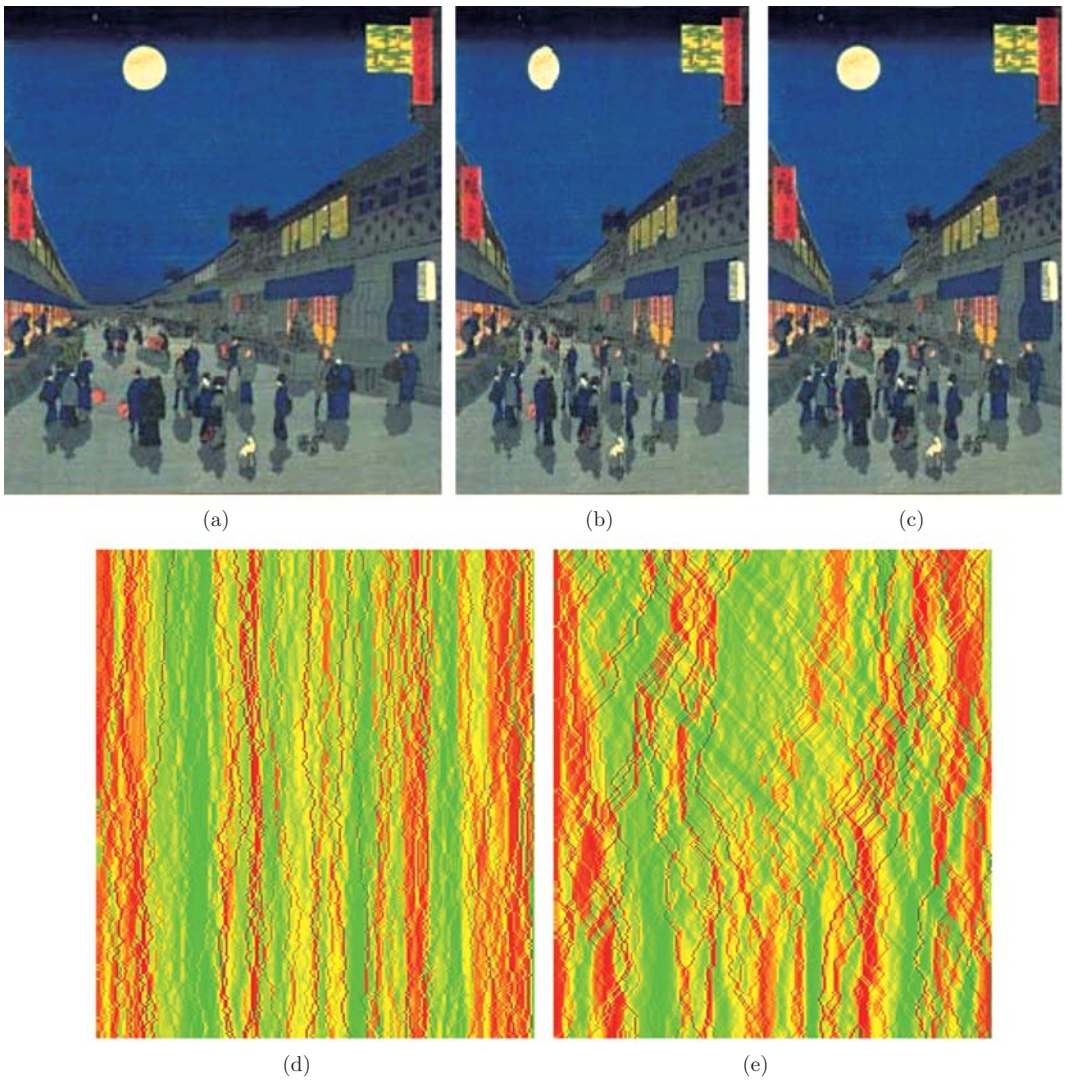


**Figure 3** Two kinds of weight definition. (a) Original image; (b) image width reduced by 1/3 using weight $w_1$; (c) image width reduced by 1/3 using weight $w_2$; (d) the vertical seams when we define weight as $w_1$. The color goes from green (first seams) to red (last seams); (e) the vertical seams when we define weight as $w_2$.

two rows when defining the weight. These two rows are isolated from the whole image. Therefore this definition only guarantees the variance of matching edges between certain two rows is the biggest but does not guarantee the largest variance between vertical seams in the whole image. That is, it only achieves a local optimum but not the global optimum.

To solve this problem we define matrices $A$ and $M$. $A(i,j)$ represents the cumulative energy along the seam which passes through pixel $I(i,j)$ from row 1 to $j$. This matrix is computed incrementally during the matching process. After establishing the matching relation between row $k-1$ and row $k$, we can compute $A(x,k), 1 \leqslant x \leqslant m$.

$M(i,j)$ represents the cumulative energy of an optimal seam which starts from pixel $I(i,j)$ and ends with a pixel in the last row. Matrix $M$ is established via using dynamic programming from bottom to top (we only use dynamic programming once for a single image and its complexity is $O(\mathrm{nm})$, which will not involve a high overhead):

$$M(i,j) = e(i,j) + \min\{M(i-1,j+1), M(i,j+1), M(i+1,j+1)\}. \quad (9)$$

We use $A(i,k)$ and $M(m(i,k),k+1)$ to replace $e(i,k)$ and $e(m(i,k),k+1)$ in the above discussion respectively. Then we have $E_i = A(i,k) + M(m(i,k),k+1)$, where $A(i,k)$ represents the cumulative energy between the first row and row $k$ and $M(m(i,k),k+1)$ represents the cumulative energy between row $k+1$ and the last row. That is $E_i$ is the energy of the entire vertical seam. Maximizing the variance of $E_i$ is also maximizing the variance of vertical seams in the image. So we define the weight as follows:

$$w_2(i,j) = \begin{cases} A(i,k) \cdot M(j,k+1), & |i-j| \leqslant 1, \\ -\infty, & \text{otherwise.} \end{cases} \quad (10)$$

The vertical seams found under this definition (see Figure 3(e)) can preserve the energy of the image better and avoid adverse effects brought by local optima, yielding better resizing (see Figure 3(c)).

## 4 Results

In order to compare the effect and time efficiency of different algorithms, we also implemented Avidan and Shamir's seam carving, their alternative version of seam carving which is also based on matching but uses the Hungarian algorithm, and the optimal cropping, all of which were applied to $e(I)$ to perform rescaling in one dimension. Table 1 gives the time needed by various seam-based algorithms to reduce the image width to 1/3. Obviously, our method runs substantially faster than the other seam carving methods.

Table 2 gives the average frame rate when using our method to process different sized videos. After establishing matching relations between all neighboring pairs of rows, we find all vertical seams in parallel. In other words, we get a multi-sized video, and so we do not need to give the target size of the output video in Table 2. All processing was carried out on an Intel 2.4 GHz Pentium IV Desktop with 512M memory.

Figures 4 and 5 visually compare reducing image width using standard seam carving, our approach and the optimal cropping respectively. From the results, we notice that despite our approach being much faster than standard seam carving it still produces similar results. Another example, this time of image enlargement, given in Figure 6, confirms its effectiveness.

Evaluating the relative effectiveness of different image resizing techniques is difficult due to the subjective nature of determining the quality of results.

**Table 1** Speed comparison

| Image size | 240×320 | 480×640 | 600×800 | 768×1024 |
|---|---|---|---|---|
| Standard seam carving | 0.5 s | 3.3 s | 6.3 s | 12.2 s |
| Hungarian algorithm | 8.0 s | 1.8 min | 4.0 min | 9.5 min |
| Our approach | 8 ms | 39 ms | 63 ms | 137 ms |

**Table 2** Frames per second on average (including decoding time)

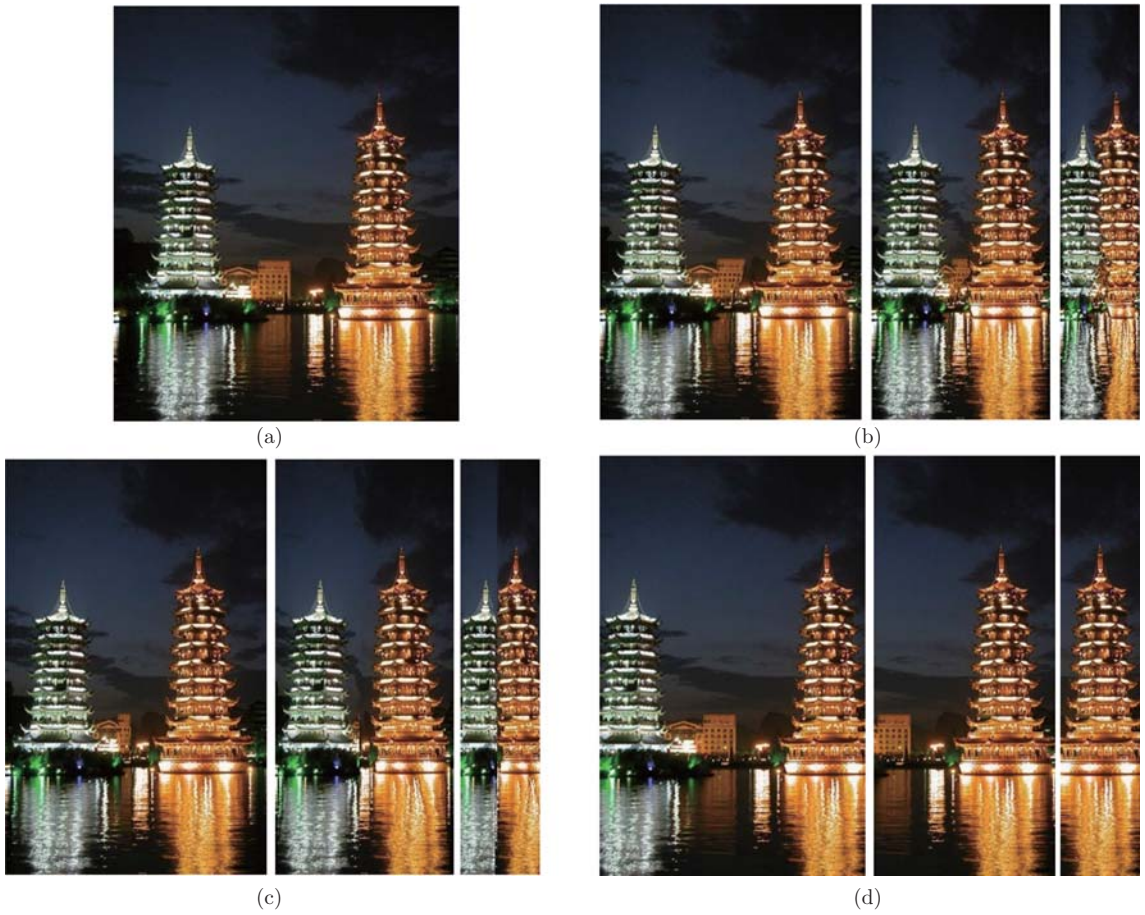
| Video size | 240×320 | 256×608 | 480×640 |
|---|---|---|---|
| FPS | 32.6 | 25.1 | 15.8 |

For the more extreme cases of resizing using the fish-eye lens warping[6] or cropping down to thumbnail size[2], substantial differences between alternative methods was evident, and so in both these cases the authors carried out user testing. In ref. [6] a two-alternative forced choice methodology was used: users were asked which of two resized images they preferred. In ref. [2] users were asked to identify the contents of thumbnails, or to match a thumbnail against a set of 100 images. Since we do not claim that our results are perceptually better than that from standard seam carving (just that they are faster), it is not necessary to carry out such user testing.

Instead, we follow Avidan and Shamir[8] who provided a simple quantitative indication of the effectiveness of the seam carving approach by plotting the energy remaining in the image after resizing. The energy should be maximized while preserving the coherence of the image. We note that in general there are many images that are resized well by all the tested methods. This is exemplified by the image in Figure 4 whose energy is plotted in Figure 7(a). All the energy curves are close, although it can be observed that seam carving does the best job at preserving energy while the optimal cropping does the worst. The proposed method is

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

**Figure 4** Comparison of using several methods to reduce the image width by 1/3. (a) Original image; (b) standard seam carving; (c) our approach (using weight definition $w_2$); (d) optimal cropping.



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

**Figure 5** Comparison of using several methods to reduce the image width to three different sizes. (a) Original image; (b) standard seam carving; (c) our approach (using weight definition $w_2$); (d) optimal cropping – note the missing pagoda in the middle column.

(a) (b)

**Figure 6** Image enlargement using our method. (a) Original image; (b) enlarged image (using weight definition $w_2$).

situated between these two methods, and as expected using weight $w_2$ outperforms $w_1$. However, there are other images where, as discussed previously, cropping is unable to capture all the desired content, an example of which is shown in Figure 5. Small amounts of cropping are satisfactory, but when the cropping window cannot entirely include both pagodas then one is retained at the expense of the other that is excluded, causing a large drop in energy, which is clearly visible in Figure 7(b). When more extreme resizing is applied such that the output image is small enough to tightly enclose just one pagoda, then the results of cropping are satisfactory again and its energy is comparable to seam carving. Note however, that the seam carving methods have succeeded in squeezing both pagodas into the image, although at the cost of eliminating some of the peripheral details of the buildings.

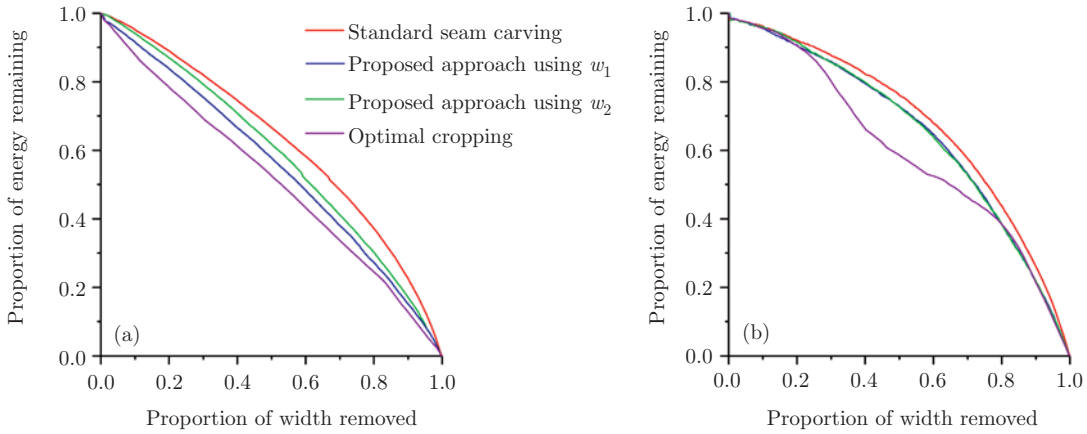



**Figure 7** Comparison of energy preservation using different methods of resizing applied to (a) the image in Figure 4—all methods produce similar energies; (b) the image in Figure 5—at intermediate levels of resizing cropping becomes ineffective.

## 5 Conclusions and future work

This paper has described an improvement of the seam carving method of content-aware image or video resizing. We propose a novel method to quickly determine the matching relations between neighboring pairs of rows. Thus the processing speed is greatly improved; no slow pre-processing step is required, making it possible to perform real-time image or video resizing. Furthermore, we also propose a good definition for the weights that connect adjacent pixels, which gives satisfactory results as long as the energy function is appropriate to the image.

There are many possible extensions to this work. First, although the weight definition $w_2$ worked well for the majority of images, there are still noticeable artifacts in the results for some images. Future research will consider more effective weights to reduce or overcome such artifacts. Second, if seam carving is applied independently to individual frames, then the results will often contain unacceptable jitter. Thus requires seam carving to be extended to incorporate continuity constraints. In recent follow up work by Avidan and Shamir[10] they do just this, and demonstrate good results. However, their method which is based on graph cuts is extremely time consuming, even after speeding up the process using multi-resolution techniques. Another work by Zhang[11] examined the shrinkability of each pixel and achieved good results for video resizing, however, the construction of shrinkability map is not so efficient for real time applications. Our goal is to achieve continuous video resizing at real time rates.


1 Itti L, Koch C, Niebur E. A model of saliency-based visual attention for rapid scene analysis. IEEE Trans Patt Anal Mach Intell, 1998, 20(11): 1254–1259

2 Sue B, Ling H, Bederson B, et al. Automatic thumbnail cropping and its effectiveness. In: Proceedings of User Interface Software and Technology, 2003. 95–104

3 Chen L, Xie X, Fan X, et al. A visual attention model for adapting images on small displays. Multimedia Syst, 2003, 9(4): 353–364

4 Ciocca G, Cusano C, Gasparini F, et al. Self-adaptive image cropping for small displays. IEEE Trans Consumer Electr, 2007, 53(4): 1622–1627

5 Santella A, Agrawala M, DeCarlo D, et al. Gaze-based interaction for semi-automatic photo cropping. In: Proceedings of Human Factors in Computing Systems, 2006. 771–780

6 Liu F, Gleicher M. Automatic image retargeting with fisheye-view warping. In: Proceedings of User Interface Software and Technology, 2005. 153–162

7 Setlur V, Lechner T, Nienhaus M, et al. Retargeting images and video for preserving information saliency. IEEE Comput Graph Appl, 2007, 27(5): 80–88

8 Avidan S, Shamir A. Seam carving for content-aware image resizing, ACM Trans Graph (SIGGRAPH), 2007, 26(3): 10–18

9 Kuhn H. The Hungarian method for the assignment problem. Naval Research Logist Quarterly, 1955, 2: 83–97

10 Rubinstein M, Shamir A, Avidan S. Improved seam carving for video retargeting. ACM Trans Graph (SIGGRAPH), 2008, 27(3): 1–9

11 Zhang Y F, Hu S M, Martin R R. Shrinkability maps for content-aware video resizing. Comput Graph Forum, 2008, 27(7): 1797–1804