Automatic Image Cropping : A Computational Complexity Study

Jiansheng Chen Gaocheng Bai Shaoheng Liang Zhengqin Li
Department of Electronic Engineering, Tsinghua University, Beijing, China
jschenthu@mail.tsinghua.edu.cn, {bgc13, liangsh13, li-zq12}@mails.tsinghua.edu.cn

Abstract

Attention based automatic image cropping aims at preserving the most visually important region in an image. A common task in this kind of method is to search for the smallest rectangle inside which the summed attention is maximized. We demonstrate that under appropriate formulations, this task can be achieved using efficient algorithms with low computational complexity. In a practically useful scenario where the aspect ratio of the cropping rectangle is given, the problem can be solved with a computational complexity linear to the number of image pixels. We also study the possibility of multiple rectangle cropping and a new model facilitating fully automated image cropping.

1. Introduction

With the rapid development of imaging and storage technologies, pixel resolution of digital images captured by modern imaging devices such as the digital camera, webcam or mobile phone has increased dramatically. Images containing over millions of pixels has become more and more common even in mobile devices. This has brought difficulties to the transmission and sharing of the images especially through the mobile Internet. Considering both the time and the cost, images are usually excessively down sampled or compressed before transmission, leading to serious degradation of image quality on the receiver's side.

Visual importances vary a log across different regions in real life images. Examples are shown in Figure 1. Actually, with the continuous increasing of pixel resolution of capturing devices, people tends to include a unnecessary amount of less important background or unrelated scenery in the image when taking pictures. In addition to causing difficulties to image transmission, these relatively less important image pixels may also harm the visual effectiveness of important image parts, especially after image retargeting for mobile devices equipped with small sized displays. To solve this problem, previous researchers have proposed a number of context aware image cropping/resizing methods which can be generally divided into two categories, attention based methods and aesthetics oriented methods [24].

1.1. Previous works

The fundamental idea of attention based methods is trying to preserve visually important area of an image after cropping or resizing. Pixel importances for visual attention are usually estimated using their saliency scores [5] [20] [18] [16], objectness [27] [7], or empirically defined energy functions [15] [1]. Chen et al. were among the first to study the image cropping problem in order to facilitate viewing large images on small sized displays [5]. Pixel saliency values calculated using Itti's model [9] was combined with face and text detection results for generating the attention map. Suh et al. extended this work by using summed saliency values within cropping rectangles for determining the best cropping position [20]. Santella et al. acquired image saliency values by means of human-computer interaction [18]. User fixation data were captured and utilized together with image segmentation results to identify important image contents and compute the best crop. By assuming that images sharing similar global visual appearances are likely to share similar salience, Marchesotti et al.

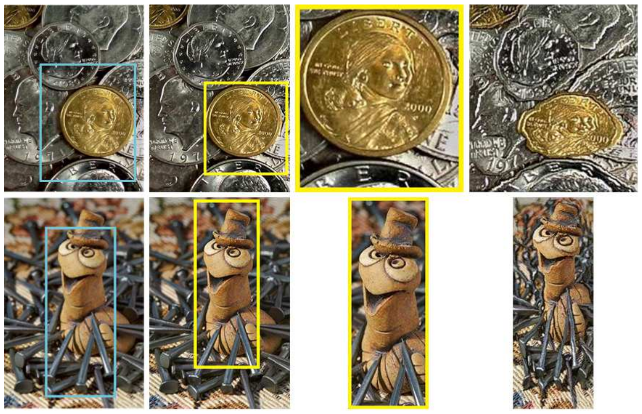


Figure 1. Examples of automatic image cropping. Images are selected from the MSRA Salient Object Database [13]. 1st column: cropping results using the graph cut based method [16]. 2nd column: cropping results $\ddot{R}(\tau^*)$ using the proposed method. 3rd column: cropped images using the proposed method. 4th column: retargeted images using the shortest path base method [1].

trained a simple classifier on an annotated image database for generating attention maps based on which image thumbnailing were achieved [16]. Zhang et al. focused on faces in the image by selecting regions of interest according to face detection results. Images were then cropped by aligning faces according to predefined image composition templates [27]. Ciocca et al. combined visual saliency information with face and skin color detection results for placing bounding box in image cropping [7]. Ma et al. proposed an comprehensive energy function based on image entropy, area size and position for measuring the visual importance before cropping [15]. Avidan et al. simply used the amplitude of image gradient as the energy function for describing pixel importance [1]. Instead of preserving pixels with high importances, Avidan et al. removed 8-connected paths of minimum summed energy values consecutively.

The aesthetics oriented method aims at maximizing the visual attractiveness of the cropped images. Although the visual aesthetics obeys certain general principles, it is also known to be influenced by subjective factors such as the culture, personal experiences, education level, or even the psychological state [4]. Therefore, most existing aesthetics oriented image cropping approaches are based on photo quality assessment studies [11] [3] [22] using certain objective aspects of images, such as low level image features and empirical photographic composition rules. Nishiyama et al. statistically built a image quality classifier using low level image features such as color histogram and Fourier coefficients. The image cropping candidate with the highest quality score was then selected [17]. Cheng et al. studied the spatial correlation distributions of two arbitrary patches in an image for generating an omni-context prior which was combined with visual words to form a posterior probability model for measuring image quality [6]. Zhang et al. introduced small connected subgrpahs, or graphlets, extracted from the region adjacency graph, for representing image aesthetic features. A probabilistic model based on the graphlet was then used for transfer aesthetic features from the training images onto the cropped images [26]. In a more recent work, Yan et al. proposed features for modeling what is changed after image cropping. The influence of these features on cropping learned from manually marked image pairs was then used for generating effective crops [24].

1.2. Motivation

In this work, we focus on a specific aspect which has to some extend been overlooked in previous studies. In most existing attention based image cropping approaches [20] [7] [14] [19] [15], a common task after generating the attention map is to search for an optimum cropping rectangle. Usually, this optimum rectangle search process aims at achieving a tradeoff between minimizing the cropping area and maximizing the total pixel attention values inside it. Considering the huge number of possible candidate rectangles, brute force search could be prohibitively slow. To solve this problem, Luo et al. adopted the integral image [23] to speed up the global search [14]; Suh et al. used a greedy algorithm to incrementally including salient peak points outside the current rectangle [20]; Stentiford et al. reduced the search space by setting a series of fixed sizes for the rectangles [19]; Ciocca et al. binarized the attention map and considered only the connectivity of pixels [7]; Ma et al. introduced human interaction to facilitate the searching process [15]. These methods are either heuristic or of nearly the same complexity as the brute force search. In this work, we propose several practical formulations of the optimum rectangle search problem and design algorithms with essentially low computational complexity to solve them.

The rest of this paper is organized as follows. Section 2 presents our problem formulations. Section 3 elaborates the proposed algorithms and corresponding complexity analysis. Experimental results are demonstrated in Section 4. The last section concludes our work.

2. Problem formulations

As we have stated above, the target of optimum cropping rectangle search on a given attention map is twofold. Firstly, the area of the rectangle should be minimized so as to crop out as much visually unimportant image regions as possible. Secondly, the sum of attention value inside the rectangle should be maximized so as to preserve as much visually important image regions as possible. These two objectives are dual and the problem can be defined either way.

Suppose G is a non-negative valued attention map extracted from an image I. Larger attention values in G indicate higher visual importance of corresponding pixels in I. Without loosing generality, we formulate the optimum cropping rectangle search problem as *Problem 1*, in which τ is the minimum percentage of total attention to be preserved and $\ddot{R}$ is the smallest rectangle satisfying this requirement.

Problem 1 ***[Minimum Rectangle Search]*** : *Given a percentile ratio τ, find a rectangle $\ddot{R}(\tau)$ of the minimum possible area size, inside G, to satisfy (1).*

$$\sum_{p\in \ddot{R}(\tau)} G(p) \geq \tau \sum_{p} G(p), \quad \tau \in [0,1] \qquad (1)$$

The attention value of an image pixel can be considered as the measurement of its visual importance. It is reasonable to think that every pixel may contain certain amount of visual information. Therefore in this work we simply assume that attention values are non-negative. This is also consistent with most existing works for calculating the attention map [9] [10] [25] [8]. In case of no ambiguity, we denote $\sum_{p\in\ddot{R}} G(p)$ as $\sum_{\ddot{R}} G$, and $\sum_p G(p)$ as $\sum G$. Also, a rectangle is called *valid* if it satisfies (1).

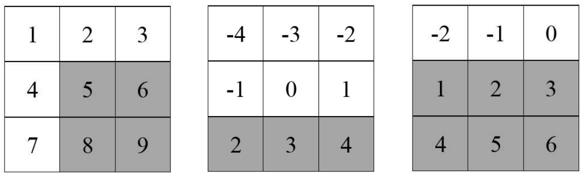
Let $\|\ddot{R}(\tau)\|$ be the rectangular area of $\ddot{R}(\tau)$. For any given G, $\|\ddot{R}(\tau)\|$ is an increasing function of τ as is expressed in (2). This can be deduced using abductive reasoning. Suppose that $\|\ddot{R}(\tau_1)\| < \|\ddot{R}(\tau_2)\|$, combining (1) and (2), we have $\sum_{\ddot{R}(\tau_1)} G \geq \tau_1 \sum G \geq \tau_2 \sum G$. This indicates that $\ddot{R}(\tau_1)$ is also a rectangle with summed attention value greater than or equal to $\tau_2 \sum G$. However, according to *Problem 1* definition, among all such rectangles, $\ddot{R}(\tau_2)$ should be the smallest, leading to obvious contradiction. In case of no ambiguity, we set the rectangular area of the attention map to be 1, so that the value of $\|\ddot{R}\|$ stands for the percentage of area occupied by $\ddot{R}$ in G.

$$\forall \tau_1 \geq \tau_2, \quad \|\ddot{R}(\tau_1)\| \geq \|\ddot{R}(\tau_2)\| \tag{2}$$

It should be emphasized that for a given τ, $\ddot{R}(\tau)$ may not be unique. In our algorithms, we always choose $\ddot{R}(\tau)$ with the largest summed attention value. Even so, the uniqueness of $\ddot{R}(\tau)$ still cannot be ensured. This is acceptable in practice considering that it only lead to different cropping results but with equal area sizes as well as equal visual importance. It should also be noted that $\|\ddot{R}(\tau)\|$ is not a monotonic increasing function of τ. It is possible that $\|\ddot{R}(\tau_1)\| = \|\ddot{R}(\tau_2)\|$ when $\tau_1 \neq \tau_2$. This usually happens when the difference between τ_1 and τ_2 is very small.

At the first glance, *Problem 1* is similar to the famous *Maximum Submatrix* problem which is to find for a matrix its submatrix $\ddot{S}$ of which the sum of elements is maximized [2] [21]. Despite of their seeming resemblance, these two problems are intrinsically different. First of all, finding the maximum submatrix of a non-negative valued matrix, such as G, is trivial since the solution is usually the matrix itself. Also, converting these two problems to each other leads to meaningless results. A easy to come up with, yet incorrect solution to *Problem 1* is to subtract average attention $\bar{G}$ from G and then get the maximum submatrix of $G - \bar{G}$. A slight more reasonable solution is to take τ into account and get the maximum submatrix for $G - \tau\bar{G}$. However, Figure 2 demonstrates that the three problems may lead to absolutely different answers shown by shaded rectangles.

Another practical consideration for image cropping is related to the application of image retargeting. Nowadays, aspect ratio various a lot across different display devices such as the desktop PC, mobile phone, or wearable device. To achieve the optimum display efficiency, a promising choice



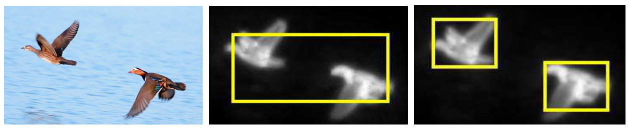
1	2	3
4	5	6
7	8	9

-4	-3	-2
-1	0	1
2	3	4

-2	-1	0
1	2	3
4	5	6

(a) $\ddot{R}(0.6)$ of G (b) $\ddot{S}$ of $G - \bar{G}$ (c) $\ddot{S}$ of $G - 0.6\bar{G}$

Figure 2. *Problem 1* v.s. *Maximum Submatrix* problem.



(a) Original image (b) $\|\ddot{R}\| = 0.44$ (c) $\|\ddot{R}_1 \cup \ddot{R}_2\| = 0.22$

Figure 3. Multiple rectangle cropping.

is to let the cropped image to have the same aspect ratio as the target display, leading to the definition of *Problem 2*. For a rectangle, we define the aspect ratio to be its width divided by its height. Later we will see that by constraining the aspect ratio of the cropping rectangle, *Problem 2* is intrinsically simpler than *Problem 1*. Hence, its low computational complexity as well as its appropriateness for image retargeting make it practically useful. Nevertheless, it should be noticed that unlike $\ddot{R}(\tau)$, $\ddot{R}(\tau, r)$ may not exist for certain τ and r values in a given attention map due to the hard constraint of the aspect ratio. What is more, because of the spatial discretization of pixels, sometimes the aspect ratio constraint can only be approximately satisfied.

Problem 2 ***[Fixed Aspect Ratio Rectangle Search]*** *: Given a percentile ratio τ, find a rectangle $\ddot{R}(\tau, r)$ of the minimum possible area size, with a fixed aspect ratio $r > 0$, inside G, to satisfy (1).*

An issue that has seldom been addressed in previous studies is that sometimes it may not be appropriate to select only one cropping rectangle from an image containing multiple visually important regions that are spatially scattered. A typical example is shown in Figure 3, in which τ is set to 0.75. By selecting two instead of only one cropping rectangle, the total cropping area size is halved and the cropping result is visually much more reasonable. Based on this understanding, we define *Problem 3* which can be regarded as a generalization of *Problem 2*.

Problem 3 ***[Multiple Rectangle Search]*** *: Given a percentile ratio τ, find no more than N non-intersected rectangles $\ddot{R}_1, \ddot{R}_2, ..., \ddot{R}_N$ inside G, all with fixed aspect ratio $r > 0$. Denote the union of these rectangles to be $\ddot{R}(\tau, r, N) = \ddot{R}_1 \cup \ddot{R}_2 \cup ... \cup \ddot{R}_N$. Minimizing the total area size $\|\ddot{R}\|$ while satisfying (1).*

By allowing more cropping rectangles, *Problem 2* increases the degree of freedom of the search process, leading to higher effectiveness of image cropping by decreasing the total area size to be preserved. Nevertheless, it is obvious that such a generalization will increase the problem complexity combinatorially. Therefore, in this paper we will only discuss the case of $N = 2$. It should be noted that for certain images, using more than one cropping rectangles may not be advantageous. In other words, some of the rectangles may be found empty while solving *Problem 3*.

3. Algorithms and analysis

In this section, we will present algorithms for solving the three problems defined above. We will focus on elucidating their correctness and analyzing their computational complexity. We assume that the attention map G is of m rows and n columns, and $m \leq n$. To solve the spatial discretization problem mentioned above, we use the following approximation for aspect ratio calculation. Given a aspect ratio r, suppose the height of a candidate rectangle is h, then its width is decided by (3).

$$w = \lceil h \times r \rceil \tag{3}$$

For a given attention map G, we adopt matrix like notations : $G(i,j)$ stands for the attention value at the $i^{th}(i \in [1,m])$ row and $j^{th}(j \in [1,n])$ column; $G(i,:)$ (or $G(:,j)$) stands for the one dimensional array of the i^{th} row (or j^{th} column) of G. In case of no ambiguity, we let $G(i,j) = 0$ whenever $i \leq 0$ or $j \leq 0$. We define the integral map G^+ of G, so that $G^+(i,j) = \sum_{k=1}^{i}\sum_{l=1}^{j} G(k,l)$. To facilitate descriptive conciseness, we also define a column based integral map G_c^+ which stores the column-wise accumulative sum of G, so that $G_c^+(i,j) = \sum_{k=1}^{i} G(k,j)$. Figure 4 shows samples of the two integral maps. G_c^+ and G^+ can be calculated simutaneously using accumulative summation with an overall computational complexity of $\mathcal{O}(mn)$ [23].

3.1. Problem 1

A brute force algorithm for solving *Problem 1* is to exhaustively examine every possible rectangle inside G so as to find the smallest rectangle satisfying (1). More specifically, as is shown in Figure 5(a), for each point (i,j) in G, the algorithm examines all the rectangles R with (i,j) as their upper left corner. The summed attention value inside R can be efficiently calculated using the integral map as is expressed by (4). For each upper left corner point, there are $\mathcal{O}(mn)$ rectangles to be examined. Looping through all possible upper left conner points leads to an overall all computational complexity of $\mathcal{O}(m^2n^2)$.

$$\begin{aligned}\sum\nolimits_R G = G^+(i_2,j_2) - G^+(i_2,j_1-1)\\ -G^+(i_1-1,j_2) + G^+(i_1-1,j_1-1)\end{aligned} \tag{4}$$

Careful observation of Figure 5(a) reveals that many unnecessary calculations have been performed in the brute force algorithm. For example, if we have already found that

1	2	3
4	5	6
7	8	9

(a) G

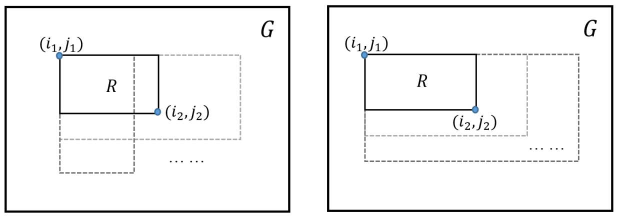
1	3	6
5	12	21
12	27	45

(b) G^+

1	2	3
5	7	9
12	15	18

(c) G_c^+

Figure 4. Integral map and column-wise integral map.



(a) *Problem 1* (b) *Problem 2*

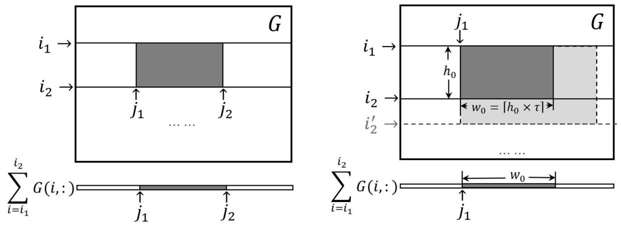
Figure 5. Illustration of brute force algorithms.

R is valid, or in other words, it satisfies (1). Then any rectangle with area size larger than R should not be considered any more. A typical example is the larger dash lined rectangle containing R shown in Figure 5(a). Even more aggressively, all the rectangles with upper left corner (i_1,j_1) and lower right corner $(i_2, j > j_2)$ can be safely ignored.

$$\sum\nolimits_{i=i_1}^{i_2} G(i,:) = G_c^+(i_2,:) - G_c^+(i_1-1,:) \tag{5}$$

Suppose we are examining all the candidate rectangles with their upper border at row i_1 and lower border at row i_2, as is shown in Figure 6(a). We are actually looking for two column index j_1 and j_2 as close to each other as possible, while the shaded rectangle is valid. By accumulating row i_1 to row i_2 column-wisely, this two dimensional problem can be converted to a one dimension problem illustrated at the bottom of Figure 6(a). Given a non-negative input array, we are to find the *shortest subarray* of which the sum of elements is larger than or equal to a given threshold. Specifically in our case, the threshold is $\tau \sum G$, and the input array is $\sum_{i=i_1}^{i_2} G(i,:)$ which can be calculated using the column-wise integral map with $\mathcal{O}(n)$ complexity using (5).

This *Shortest Subarray* problem can be efficiently solved using **Algorithm 1**, in which st and ed are two moving pointers pointing to the starting and ending positions of the current subarray. Whenever the sum of the current subarray is smaller than the threshold T, it is prolonged by moving ed one step forward (line 7). Otherwise, the current subarray is valid and will be used to update the *shortest subarray* when necessary (line 16), after which it will be shortened by moving st one step forward (line 18). The above two steps are repeated until the pointers reach the end of the input array. The key idea of the algorithm is that whenever a valid subarray candidate is found, it will be shortened in the next step



(a) *Problem 1* (b) *Problem 2*

Figure 6. Illustration of proposed algorithms.

so that any longer subarray starting from st are automatically ignored to avoid redundant calculation. Specifically, $j_1 = j_2 = 0$ if the subarray is not found or $T \leq 0$. Inside each loop, either st or ed will be increased by 1. Noticing that both st and ed will not exceed n on exit, the loop body will be executed for at most $2n$ times. Therefore, the overall computation complexity is $\mathcal{O}(n)$.

Combining **Algorithm 1** and the idea illustrated Figure 6(a), **Algorithm 2** is proposed for solving *Problem 1*. The basic idea it to loop for all possible (i_1, i_2) while finding the corresponding *shortest subarray*. The most time consuming operations in this algorithm are at lines 5 and line 6. As we have already explained, both of these two operations are of $\mathcal{O}(n)$ complexity. They will be executed $m^2/2$ times by looping for all possible (i_1, i_2), leading to an overall computation complexity of $\mathcal{O}(m^2 n)$. Or more accurately, $\mathcal{O}(m^2 n + mn)$ considering the extra calculation of the two integral maps.

3.2. Problem 2

Due to the restriction of the aspect ratio of the cropping rectangle, *Problem 2* is intrinsically simpler than *Problem 1*. This can be observed from the illustration of the brute force algorithm shown in Figure 5(b). As stated before, given the height h of a rectangle with fixed aspect ratio r, its width will be uniquely decided by (3). Therefore in Figure 5(b), the number of possible rectangles with upper left corner (i_1, j_1) becomes much smaller. It is actually decided by the number of different height values, which is basically $\mathcal{O}(m)$. By looping for all possible (i_1, i_2), the overal computational complexity of the brute force algorithm is $\mathcal{O}(m^2 n)$.

The idea for improving the algorithm is illustrated in Figure 6(b). With a fixed aspect ratio, all the candidate rectangles bounded by i_1 and i_2 are of the same size $w_0 \times h_0$, in

Algorithm 1 $\boldsymbol{shortestSubarray(\hat{a}, T)}$

Input: $\hat{a}$ is a non-negative valued array with length n; T is threshold of the subarray sum.

Output: Starting and ending index, j_1 and j_2, of the shortest contiguous subarray of $\hat{a}$ which has a sum larger than T.

```
1:  $j_1 \leftarrow 0, j_2 \leftarrow 0$
2:  $L_{min} \leftarrow \infty, S_{min} \leftarrow -1$
3:  $st \leftarrow 1, ed \leftarrow 1, S_0 \leftarrow \hat{a}(1)$
4:  if $T > 0$ then
5:      repeat
6:          if $S_0 < T$ then
7:              $ed \leftarrow ed + 1$
8:              if $ed > n$ then
9:                  Break
10:             else
11:                 $S_0 \leftarrow S_0 + \hat{a}(ed)$
12:             end if
13:         else
14:             $L \leftarrow ed - st + 1$
15:             if $L < L_{min} \vee (L = L_{min} \wedge S_0 > S_{min})$ then
16:                 $j_1 \leftarrow st, j_2 \leftarrow ed, L_{min} \leftarrow L, S_{min} \leftarrow S_0$.
17:             end if
18:             $S_0 \leftarrow S_0 - \hat{a}(st)$, $st \leftarrow st + 1$
19:         end if
20:     until $st > n$
21: end if
22: return $j_1, j_2, S_{min}$
```

Algorithm 2 $\boldsymbol{Mininum\_Rectangle(G, \tau)}$

Input: G is a non-negative attention map with size $m \times n$; τ is the percentage of total attention to be preserved; suppose integral maps G^+ and G_c^+ are calculated.

Output: The smallest valid cropping rectangle $\ddot{R}$; four values to define $\ddot{R}$: i and j are the upper left corner coordinates, w and h are the width and height.

```
1:  $i \leftarrow 0, j \leftarrow 0, w \leftarrow \infty, h \leftarrow \infty$
2:  $S_{min} \leftarrow -1, T \leftarrow \tau G^+(m, n)$
3:  for $i_1 = 1$ to $m$ do
4:      for $i_2 = i_1$ to $m$ do
5:          $\hat{a} \leftarrow G_c^+(i_2, :) - G_c^+(i_1 - 1, :)$
6:          $j_1, j_2, S_0 \leftarrow \boldsymbol{shortestSubarray(\hat{a}, T)}$
7:          if $j_1 > 0 \wedge j_2 > 0$ then
8:              $w_0 \leftarrow j_2 - j_1 + 1, h_0 \leftarrow i_2 - i_1 + 1$
9:              if $w_0 h_0 < wh \vee (w_0 h_0 = wh \wedge S_0 > S_{min})$
                then
10:                 $i \leftarrow i_1, j \leftarrow j_1, w \leftarrow w_0, h \leftarrow h_0$
11:                 $S_{min} \leftarrow S_0$
12:             end if
13:         end if
14:     end for
15: end for
16: return $i, j, w, h$
```

Algorithm 3 $\boldsymbol{maxSubarrayFL(\hat{a}, w, T)}$

Input: $\hat{a}$ is a non-negative valued array with length n; w is length of the subarray; T is the threshold of the subarray sum.

Output: Starting index j_1 of the contiguous subarray of fixed length w having the maximum sum $\geq T$; suppose the accumulative sum array $\hat{a}^+$ is calculated.

```
1:  $j_1 \leftarrow 0, S_{max} \leftarrow -1$
2:  if $T > 0 \wedge w > 0$ then
3:      for $st = 1$ to $n - w + 1$ do
4:          $S_0 \leftarrow \hat{a}^+(st + w - 1) - \hat{a}^+(st - 1)$
5:          if $S_0 \geq T \wedge S_0 > S_{max}$ then
6:              $j_1 \leftarrow st, S_{max} \leftarrow S_0$
7:          end if
8:      end for
9:  end if
10: return $j_1, S_{max}$
```

which $w_0 = i_2 - i_1 + 1$. Obviously, there are $\mathcal{O}(n)$ different such rectangles. We only need to find among these rectangles the one with the maximum summed attention value. Similar to the idea used in the last section, this problem can be converted to a one dimensional search problem which is to find a fixed length subarray with maximum sum. Specifically in our case, this maximum sum should be greater than or equal to the given threshold $\tau \sum G$. This is a naive problem which can be easily solved with $\mathcal{O}(n)$ complexity as is shown in **Algorithm 3**.

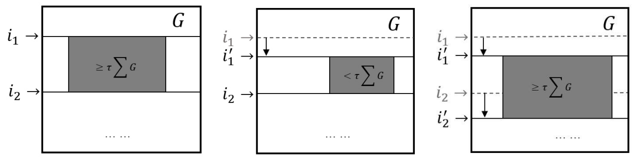
Problem 2 can be solved by simply looping for all possible (i_1, i_2) while invoking **Algorithm 3**. Unfortunately, such an approach is meaningless since it will lead to a computational complexity of $\mathcal{O}(m^2 n)$, which is identical to the brute force search. However, as stated above, in Figure 6(b), area size of the candidate rectangle is fully decided by the distance between i_1 and i_2. As such, for a given i_1 value, if a valid rectangle has been found for a certain i_2, then any position below i_2, for example $i_2^{'}$ in Figure 6(b), will no longer need to be considered because of the definitely increased rectangular area size.

Algorithm 4 $\boldsymbol{Fixed\_AspRatio\_Rectangle(G, \tau, r)}$

Input: G is a non-negative attention map with size $m \times n$; τ is the percentage of total attention to be preserved; r is the aspect ratio of cropping rectangle; suppose integral maps G^+ and G_c^+ are calculated.

Output: The smallest rectangle $\ddot{R}$ with aspect ratio r that satisfies (1); four values to define $\ddot{R}$: i and j are the upper left corner coordinates, w and h are the width and height.

```
1:  $i \leftarrow 0, j \leftarrow 0, w \leftarrow \infty, h \leftarrow \infty$
2:  $i_1 \leftarrow 1, i_2 \leftarrow 1, T \leftarrow \tau G^+(m,n), S_{min} \leftarrow -1$
3:  repeat
4:      $h_0 \leftarrow i_2 - i_1 + 1, w_0 \leftarrow \lceil h_0 \times r \rceil$
5:      if $w_0 > n$ then
6:          $i_1 \leftarrow i_1 + 1$
7:      else
8:          $\hat{a} = G_c^+(i_2, :) - G_c^+(i_1 - 1, :)$
9:          $j_1, S_0 \leftarrow \boldsymbol{maxSubarrayFL(\hat{a}, w_0, T)}$
10:         if $j_1 > 0$ then
11:             if $w_0 h_0 < wh \vee (w_0 h_0 = wh \wedge S_0 > S_{min})$ then
12:                 $i \leftarrow i_1, j \leftarrow j_1, w \leftarrow w_0, h \leftarrow h_0$
13:                 $S_{min} \leftarrow S_0$
14:             end if
15:             $i_1 \leftarrow i_1 + 1$
16:         else
17:             $i_2 \leftarrow i_2 + 1$.
18:         end if
19:     end if
20: until $i_2 > m \wedge i_1 \geq m$
21: return $i, j, w, h$
```



(a) $v(i_1, i_2) = 1$ (b) $v(i_1^{'}, i_2) = 0$ (c) $v(i_1^{'}, i_2^{'}) = 1$

Figure 7. Illustration of **Algorithm 4**.

Algorithm 4 is proposed based on this understanding. Two pointers i_1 and i_2 are pointing to the upper and lower boundaries of the candidate rectangle. Pointer i_2 is moved forward to enlarge the search area until the first valid rectangle is found (line 17). Then pointer i_1 is moved forward to reduce the search area until no valid rectangle can be found (line 15). The above two steps are performed alternatively until the two pointers reach the bottom of G. During the process, $\ddot{R}$ keeps being updated accordingly .

We use Figure 7 to help proving the correctness of **Algorithm 4**. Let's define a bool function $v(i_1, i_2)$ to denote whether a valid rectangle bounded by row i_1 and row i_2 exists. Obviously, this function fulfills (6) due to the non-negativity of G. Supposed at a certain stage of execution, we have $v(i_1, i_2) = 1$ as is shown in Figure 7(a). According to the algorithm, i_1 is then moved to the very first position to let $v(i_1^{'}, i_2) = 0$ as is shown in Figure 7(b). Notice that this also implies $v(i_1^{'} - 1, i_2) = 1$. Finally, i_2 is moved to the very first position to let $v(i_1^{'}, i_2^{'}) = 1$, implying $v(i_1^{'}, i_2^{'} - 1) = 0$.

$$v(i,j) = 0 \quad \Rightarrow \quad \forall i^*, j^* \in [i,j], v(i^*, j^*) = 0 \qquad (6)$$

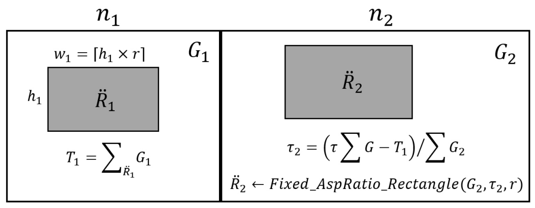
To ensure completeness, we ought to inspect all the cases when $i_2^* \in [i_2, i_2^{'})$ and $i_1^* \in [1, i_2^*]$. If $i_1^* \in [1, i_1^{'} - 2]$, any rectangle bounded by row i_1^* and row i_2^* can be safely ignored since it is definitely larger than the already considered valid rectangle bound by row $i_1^{'} - 1$ and row i_2 considering $i_2^* - i_1^* \geq i_2 - (i_1^{'} - 2) > i_2 - (i_1^{'} - 1)$. When $i_1^* = i_1^{'} - 1$, the case of $i_2^* = i_2$ has already been considered and all the other cases where $i_2^* > i_2$ can be similarly ignored due to the definitely increased rectangular area size. If $i_1^* \in [i_1^{'}, i_2^*]$, since $i_1^*, i_2^* \in [i_1^{'}, i_2^{'} - 1]$ and $v(i_1^{'}, i_2^{'} - 1) = 0$, we have $v(i_1^*, i_2^*) = 0$ according to (6). The above reasoning is valid through out the execution of the whole algorithm.

Algorithm 4 is of very low computation complexity. Inside each loop, either i_1 or i_2 will be increased by 1. Since both i_1 and i_2 will not exceed m on exit, the loop body will be executed for at most $2m$ times. Therefore the overall computational complexity is $\mathcal{O}(mn)$, which is actually the naive lower bound of the problem considering that there are altogether $m \times n$ elements in G and each element has to be considered for at least once.

3.3. Problem 3

For *Problem 3*, we only consider the case when $N = 2$. Namely, to find at most two disjoint rectangles. Even though, the complexity is substantially increased considering the explosion of the solution space due to the combination of the positions and relative sizes of two rectangles. An even tougher problem is the distribution of total attention values among the two rectangles. In this work, we merely explore the feasibility of this problem by proposing a preliminary solution which is illustrated in Figure 8.

Two non-intersected rectangles can be spatially separated either vertical or horizontal. Without losing generality, suppose the two rectangles are vertically separable. Divided the attention map vertically to get G_1 and G_2 with n_1 columns and n_2 columns respectively, and $n_1 + n_2 = n$. Suppose h_1 is the height of the left sided rectangle $\ddot{R}_1$ inside G_1. Then the width of $\ddot{R}_1$ equals $\lceil h_1 \times r \rceil$. Let T_1 be the sum of attention values inside $\ddot{R}_1$, maximize the value of T_1 by exhaustive search in G_1. This is straightforward since the shape of $\ddot{R}_1$ is fixed, and the complexity is obviously $\mathcal{O}(mn_1)$. Find in G_2 the smallest rectangle $\ddot{R}_2$ with summed attention value no less than $\tau \sum G - T_1$ using **Algorithm 4** with computational complexity $\mathcal{O}(mn_2)$. Loop for all possible values of h_1 and n_1 to minimize $\|\ddot{R}_1\| + \|\ddot{R}_2\|$. It is not difficult to see that the overall computational complexity is $\mathcal{O}(m^2 n^2)$. Similar method can be used for $N > 2$ with even higher complexity.



n_1
n_2
$w_1 = \lceil h_1 \times r \rceil$
G_1
h_1
$\ddot{R}_1$
$T_1 = \sum_{\ddot{R}_1} G_1$
$\ddot{R}_2$
G_2
$\tau_2 = (\tau \sum G - T_1)/\sum G_2$
$\ddot{R}_2 \leftarrow Fixed\_AspRatio\_Rectangle(G_2, \tau_2, r)$

Figure 8. A solution to *Problem 3*.

3.4. Automatic selection of τ

According to our definition, τ is the percentage of attention to be preserved. Generally speaking, the value of τ can be selected empirically or according to user requirements. Due to the low complexity of the proposed algorithms, we may even allow users to change τ in real time. Nevertheless, it may still be interesting to investigate the possibility of selecting τ automatically. We only study this issue for *Problem 1* considering that $\ddot{R}(\tau)$ always exist for $\forall \tau \in [0, 1]$ in *Problem 1*, leading to conciseness in analysis.

Considering the nature of image cropping, selection of τ relies heavily on the mathematical property of function $\|\ddot{R}(\tau)\|$. Obviously, $\|\ddot{R}(\tau)\|$ is a complicate function which varies a lot for different attention maps. Ideally, for a uniform attention map in which all pixels are equally important, we have $\|\ddot{R}(\tau)\| = \tau$. Also for a positive valued attention map, $\|\ddot{R}(0)\| = 0$ and $\|\ddot{R}(1)\| = 1$. In real life images, pixels with high attention values are often spatially concentrated, leading to the phenomenon that $\|\ddot{R}(\tau)\|$ usually increases slowly for small τ and fast for large τ values as is shown in the third row of Figure 9. The function curve may vary significantly for different images. However, by plotting them in a logarithmic coordinate shown in the fourth row of Figure 9, strong linear correlation between $log(\|\ddot{R}(\tau)\|)$ and $log(\tau)$ can be observed. To further validate this observation, we calculate the Pearson's correlation coefficients between $log(\|\ddot{R}(\tau)\|)$ and $log(\tau)$ for 1000 randomly selected Microsoft COCO images [12]. The mean and standard deviation of the correlation coefficients are 0.995 and 0.004 respectively, indicating a strong statistical validity of this assumption of logarithmic linearity.

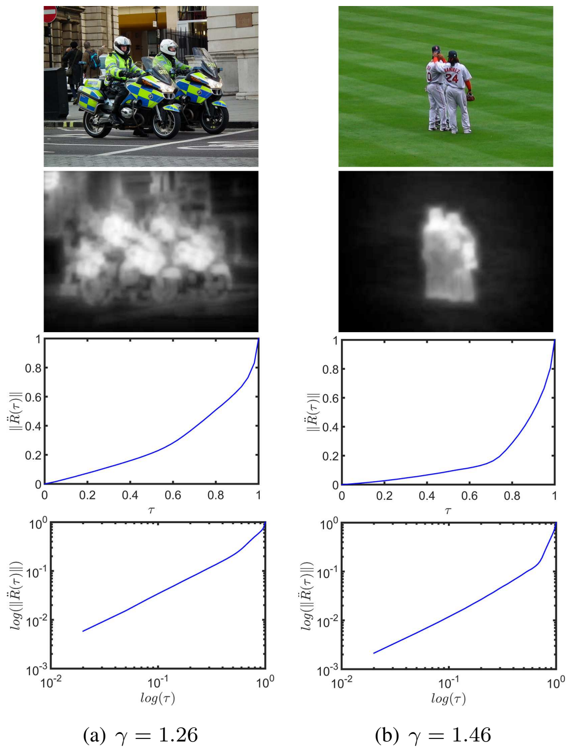


Figure 9. Relationship between $\|\ddot{R}(\tau)\|$ and τ.

$$\tau^* = \underset{\tau}{\operatorname{argmax}}\, \tau(1 - \|\ddot{R}(\tau)\|) = \underset{\tau}{\operatorname{argmax}}\, (\tau - \tau^{1+\gamma}) \quad (7)$$

We thus propose a simple power function model as $\|\ddot{R}(\tau)\| = \tau^\gamma, (\gamma \geq 1)$, in which γ can actually be used to measure the degree of concentration of the intention map. Larger value of γ usually indicates higher degree of visual attention concentration as is shown in Figure 9(b). For a given image, γ can be estimated by linearly fitting $log(\|\ddot{R}(\tau)\|)$ to $log(\tau)$. In practice, we choose 10 sampling points of τ for fitting. Based on this model, different objectives can be easily defined for selecting the optimum τ. As an example, we propose a simple objective function in (7) of which the rationale is to achieve a equilibrium between attention preserving and region cropping. Analytically solving (7) leads to $\tau^* = (1 + \gamma)^{-1/\gamma}$.

4. Experiments

Although we focus on improving the computational efficiency of cropping rectangle search, it is of no doubt that the key to cropping effectiveness is still the reliability of the attention map. In all the visual results shown below, we use attention maps generated using two different methods. The yellow and red rectangles are calculated based on attention maps generated using [8] and [10] respectively. All the images used in our experiments are selected from the Microsoft COCO database [12]. Figure 10 and Figure 11 show results of minimum area cropping defined in *Problem 1*. Figure 10 illustrates the influence of τ value. It can be observed that both the size and position of the cropping rectangle change with τ. Figure 11 demonstrates the effectiveness of automatic τ selection. Figure 12 presents the visual results of *Problem 2* when the aspect ratio changes. Quite surprisingly, cropping rectangles with different aspect ratio all seem to be visually reasonable. Figure 13 are the multiple rectangle cropping results. The attention model proposed in [10] intentionally emphasizes the visual importance near the image center, leading unsatisfactory results shown by red rectangles in Figure 13.

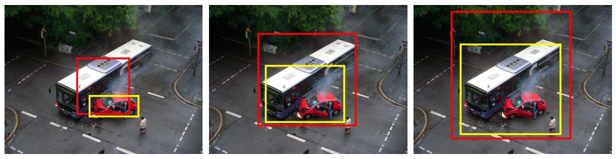

(a) $\ddot{R}(0.2)$ (b) $\ddot{R}(0.5)$ (c) $\ddot{R}(0.7)$

Figure 10. *Problem 1* : cropping results for different τ.

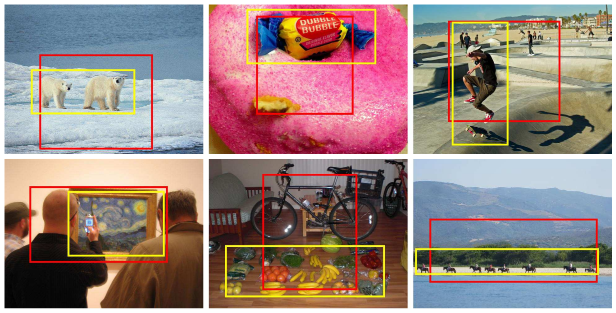


Figure 11. *Problem 1* : automatic τ selection $\ddot{R}(\tau^*)$

We also compare the average running time of **Algorithm 2** and **Algorithm 4** to the corresponding brute force algorithms on 1000 randomly selected images. The experiment is performed on a desktop PC equipped with a 3.6GHz CPU and 16GB memory using Matlab implementations. The acceleration ratios are plot against τ in Figure 14. All the attention maps are generated using [8] and $m = 188, n = 250$. The average running time for all τ values is 137.8ms for **Algorithm 2** and 4.2ms for **Algorithm 4**.

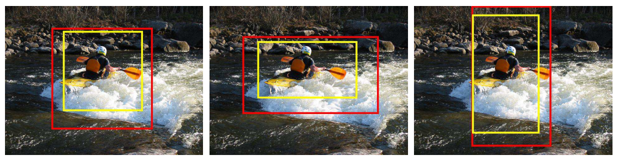

(a) $\ddot{R}(0.5, 1)$ (b) $\ddot{R}(0.5, 16/9)$ (c) $\ddot{R}(0.5, 9/16)$

Figure 12. *Problem 2* : cropping results for different aspect ratio.

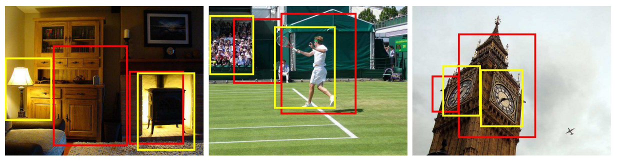
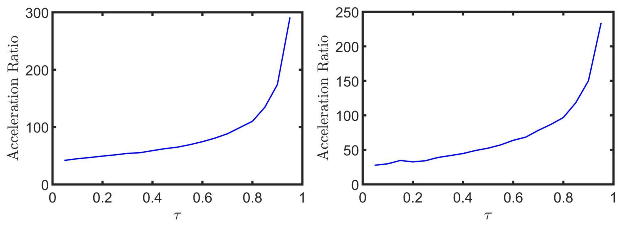


Figure 13. *Problem 3* : multiple rectangle cropping $\ddot{R}(0.5, 3/4, 2)$.

The high acceleration ratio for large τ values are caused by the extremely low practical complexity of the proposed algorithms for large τ values. For example, when $\tau = 0.9$, the average running time is 5.7ms for **Algorithm 2** and 0.9ms for **Algorithm 4**.



(a) **Algorithm 2** (b) **Algorithm 4**

Figure 14. Acceleration ratio of proposed algorithms.

5. Conclusions

We study the computational complexity of the optimum rectangle search in the attention based automatic image cropping. According to different application requirements as well as image properties, we propose three problem formulations, for which algorithms with low computational complexity are designed. We also propose a fully automated image cropping approach based on a new model describing the relationship between attention preserving and region cropping. Experimental results have demonstrated the effectiveness and efficiency of our proposals. There are still problems left to be studied in the future. For example, the relationship between visual satisfactory and the selection of τ value; and the possibility of fusing different attention maps. It is also interesting to extend this research to aesthetics oriented image cropping methods.

This work was supported by the Tsinghua University Initiative Scientific Research Program (20131089382), the Beijing Higher Education Young Elite Teacher Project (YETP0104), and the National Natural Science Foundation of China (61101152).

References

[1] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. *ACM Trans. on Graphics*, 26(3):10–1–10–9, 2007. 1, 2

[2] J. Bentley. *Programming Pearls*. Addison Wesley, 2000. 3

[3] S. Bhattacharya, R. Sukthankar, and M. Shah. A framework for photo-quality assessment and enhancement based on visual aesthetics. In *ACM Multimedia*, pages 271–280, 2010. 2

[4] P. Bourdieu. *Distinction, A Social Critique of the Judgement of Taste*. Harvard University Press, 1987. 2

[5] L. Chen, X. Xie, X. Fan, W. Ma, H. Zhang, and H. Zhou. A visual attention model for adapting images on small displays. *Multimedia Systems*, 9:353–346, 2003. 1

[6] B. Cheng, B. Ni, S. Yan, and Q. Tian. Learning to photograph. In *ACM Multimedia*, pages 291–300, 2010. 2

[7] G. Ciocca, C. Cusano, F. Gasparini, and R. Schettini. Self adaptive image cropping for small displays. *IEEE Trans. Consumer Electronics*, 54(4):1622–1627, 2007. 1, 2

[8] S. Goferman, L. Zelinik-Manor, and A. Tal. Context-aware saliency detection. *IEEE Trans. PAMI*, 34(10):1915–1926, 2012. 2, 8

[9] L. itti and C. Koch. A model of saliency based visual attension of rapid scene analysis. *IEEE Trans. PAMI*, 20(9):1254–1259, 1998. 1, 2

[10] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *ICCV*, 2009. 2, 8

[11] Y. Ke, X. Tang, and F. Jing. The design of high-level features for photo quality assessment. In *CVPR*, pages 419–426, 2006. 2

[12] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L.Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 2106–2113, 2014. 7, 8

[13] T. Liu, J. Sun, N. Zheng, and X. Tang. Learning to detect a salient object. In *CVPR*, 2007. 1

[14] J. Luo. Subject content-based intelligent cropping of digital photos. In *ICME*, pages 2218–2221, 2007. 2

[15] M. Ma and J. K. Guo. Automatic image cropping for mobile devices with built-in camera. In *Consumer Communication and Networking*, pages 710–711, 2004. 1, 2

[16] L. Marchesotti, C. Cifarelli, and G. Csurka. A framework for visual salicency detection with applications to image thumbnailing. In *ICCV*, pages 2232–2239, 2009. 1, 2

[17] M. Mishiyama, T. Okabe, Y. Sato, and I. Sato. Sensation based photo cropping. In *ACM Multimedia*, pages 669–672, 2009. 2

[18] A. Santella, M. Agrawala, D. DeCarlo, D. Salesin, and M. Cohen. Gaze-based interaction for semi-automatic photo cropping. In *ACM SIGCHI*, pages 771–780, 2006. 1

[19] F. Stentiford. Attention based auto image cropping. In *ICVS*, 2007. 2

[20] B. Suh, H. Lin, H. Bederson, and D. Jacobs. Automatic thumbnail cropping and its effectiveness. In *ACM Symp. UIST*, pages 95–104, 2003. 1, 2

[21] T. Takaoka. Efficient algorithms for the maximum subarray problem by distance matrix multiplication. *Electronic Notes in Theoretical Computer Science*, 61(4):191–200, 2002. 3

[22] X. Tang, W. Luo, and X. Wang. Content-based photo quality assessment. *IEEE Trans. on Multimedia*, 15(8):1930–1943, 2013. 2

[23] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004. 2, 4

[24] J. Yan, S. Lin, S. B. Kang, and X. Tang. Learning the change for automatic image cropping. In *CVPR*, June 2013. 1, 2

[25] J. Yang and M. H. Yang. Top-down visual saliency via joint crf and dictionary learning. In *CVPR*, 2012. 2

[26] L. Zhang, M. Song, Q. Zhao, X. Liu, J. Bu, and C. Chen. Probabilistic graphlet transfer for photo cropping. *IEEE Trans. on Image Processing*, 22(2):802–815, 2013. 2

[27] M. Zhang, L. Zhang, Y. Sun, L. Feng, and W. Ma. Auto cropping for digital photographs. In *ICME*, 2005. 1, 2