2018 IEEE Confs on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics

An Improved Automatic Thumbnail Generation Algorithm Based on Interpolation Technique

Guifan Weng[1], Chun Yang[1*], Jinyi Long[2] ,Shanying Chen[1]
*1.College of Mathematics and Informatics,South China Agricultural University,*Guangzhou 510642, China

2. *College of Information Science and Technology, Jinan University,*
Guangzhou 510632, China.

***Abstract*—In servers for mobile devices, the generating speed for thumbnails should be fast and efficient. Previous research focused on the visual quality of thumbnails. In this paper, an interpolation-based thumbnail generation method is proposed to accelerate the generation processes. The proposed algorithm was based on area average sampling interpolation, which an improved sampling method to reduce the amount of sample data. The algorithm also compared with the gaussian blur transiting interpolation-based algorithm and box blur transiting interpolation based algorithm. They were also compared with open source projects called ImageMagick and OpenCV. The result shows that the area average interpolation-based algorithm is faster than the other two algorithms without affecting picture quality.**

Keywords—thumbnails, Gaussian Blur, Box Blur, ImageMagick, Area Average Interpolation

1. Introduction

With the development of Internet and Multimedia, most of the servers have to process large amounts of digital images of big sizes and high qualities and transformed them into very small-scale images for users to browse. Thumbnails are the small size images representing those normal size images being scaled down on websites or in computers. Thumbnails are generated by interpolation algorithms from the original image.

There are three traditional thumbnail generation algorithms, the linear interpolation-based, bilinear interpolation-based and bicubic interpolation-based[1]. However, they would lead to the intensification of aliasing due to the amplifying of gradients of the edges by ignoring the transiting pixels on the original images[2-4]. Since the range of pixels are comprehensive enough to represent integrity of the pixel feature of the area for sampling to fill the objective pixel, these traditional methods are just suitable for upsampling, which is to enlarge the original image.

Though many following research better extract "salient" information from the orginal image, they do this at the expense of efficiency. Jingwei Huang[5] proposed a method for thumbnail genaration with the process of recognizing the salient objects in the foreground. Before the process of detection, it has to spend time to train the model and use the model to recognize targets. [6] spends time to detect salient object in the foreground by finding the edges and drawing contours. In the preprocessing module for the importance analysis of [7], the calculation of attention map and detection of face for the importance map negatively affect the efficiency of generating thumbnails. In [8], the procedure of dividing the original image into the attention object and background is also very time-consuming. In purpose of detecting the importance of an image, [9] needs to find the similar objects and compute for the sematic information, which needs a certain amount of time. Samadani and his team [10] preserves the blur and noise from original pictures for the thumnail in order to maintain its blurry feature from the original image. It needs extra memmory and time. Maiko M. I. Lie with his team[11] used the data from saliency map to detect salient objects in the original image and adapt content in the objective image. Both [12] and [13] need GPU to reduce their high-cost computation since the former use convolutional neural network to do training and detection and the latter use bilinear interpolation algorithm.

These programs and algortihms are not practical for industrial applications, in which most companies just want to generate thumbnails with some basic ideas of the orginal image for users to browse and identify. They want to generate thumbnails in a very short time, less than 50ms in average on the Linux severs just with CPU. Moreover, they do not want to use extra hard drive and memory to store those data for training models for recognition and detection. Moreover, in most cases, showing the whole content of the original image is more important than just the "salient" part of the image. Overemphasizing "slient" content is an extra and high expense without much benefit to user experience and company profits.

Open source projects for Linux such as ImageMagick，GraphicMagick and OpenCV have deficiencies in their speeds. Moreover, ImageMagick has memory leakage when it is generating a large number of thumbnails. All of them can not guarantee the average speed is not slower than 50ms per image.

This paper has proposed a algorithm to generate thumbnail efficiently and without much negative effects on visual qualities by experimenting and improving the operations of sampling and interpolation, comparing the speeds and visual effects with the thumbnails generated by ImageMagick and OpenCV. Instead of focus on the salient objects, the algorithm remains the whole content from the original image.

This work was supported by the funding from the National Natural Science Foundation of China (Grant No. 61773179 and 61403147), and Guangdong Provincial Natural Science Foundation of China (Program No.2014A030313233). Asterisk indicates corresponding author. Correspondence e-mail: yangch@scau.edu.cn.

978-1-5386-7975-3/18/$31.00 ©2018 IEEE
DOI 10.1109/Cybermatics_2018.2018.00141

2. Algorithm

2.1 Anti-Aliasing

Figure 1. The thumbnail on the left generated by traditional interpolation algorithm and the thumbnail on the right generated by the new algorithm from this paper.

According to the results shown in Fig.1, the aliasing effects of the thumbnail generated by the traditional interpolation on the left is severer than the one on the right, which is generated by the algorithm from this paper. The visual influence of the pixels of object's edges in an interpolated image is very significant because of the particular sensitivity of human eyes focusing on the edges of images[14]. Thus, the process of reducing the blocky or stair-wise appearance of pixels on computer image is inevitable, such process is so called the Anti-aliasing[15].

2.2 Gaussian Blur Transiting Interpolation

The main idea of Gaussian Blur Transiting Interpolation(GBTI) is processing gaussian filter before linear interpolation.

$gauss_{x,y}$ is the weight of *(x,y)* in the Gaussian distribution matrix, $d\_blur_{i,j}$ with the coordinate of *(i,j)* is the new pixel value by Gaussian filtering. $d_{x,y}$ with the coordinate *(x,y)* is the pixel value on the original image. And the equation for Gaussian Blur is Equ.1.

$$d\_blur_{i,j} = \frac{\sum_{x=j-r}^{x\le j+r}\sum_{y=i-r}^{y\le i+r}(d_{x,y}\times gauss_{x-j+r,y-i+r})}{\sum_{x=0}^{x\le 2r}\sum_{y=0}^{y\le 2r} gauss_{x,y}} \quad (1)$$

r is the radius of filter window for sampling from the original image. Then, use linear interpolation to generate thumbnails after Gaussian Blur.

The main purpose of Gaussian blur is to reduce the distortion to the extent that human eyes cannot perceive the distorted edges. Thus, it is not necessary to maintain the originality and accuracy of Gaussian blur at the expense of speed. It can be improved by implementing Gaussian blur of one dimension.

In Equ.2, the gauss matrix only has one row. We only consider the pixels on the same column or on the same row of the coordinate of *(i,j)* on the original image.

$$d\_blur_{i,j} = \frac{\sum_{x=i-r}^{x\le i+r}(d_{x,j}\times gauss_{x-i+r}) + \sum_{y=j-r}^{y\le j+r}(d_{i,y}\times gauss_{y-j+r})}{2\sum_{x=0}^{x\le 2r} gauss_x} \quad (2)$$

In the original algorithm, the number of iterations of the loop for generating one objective pixel is $2(2\times r+1)$. The height of the original image is *H*, width is *W*, so the number of iterations is $(H\times W\times(2\times r+1)\times 2)$. After the improvement, the number of iterations is $(H\times W\times 2(2\times r+1))$. Therefore, the updated result is $2/(2\times r+1)$ of the original count. However, it is still very time-consuming for such operation.

Thus, it is necessary to improve further. According to the fact that aliasing occurs only the size of the objective image is relatively small enough comparing to the original size. The threshold size for the aliasing to occur by linear interpolation is *ThreSize*, the objective size is *ObjSize*, the whole process can be separated into 2 circumstances:

1) while *ObjSize*>=*ThreSize*, use linear interpolation to generate the objective images.

2) while *ObjSize*<*ThreSize*, use linear interpolation to generate the intermediate matrix. Its size is smaller than the size of the original matrix, and bigger than the size of the objective matrix. It is a intermediate matrix for reducing the time complexity without much affection to visual quality. Implement Gaussian blur and linear interpolation to generate the thumbnail from the intermediate matrix.

The visual quality and speed are all improved after adding the condition of whether *ObjSize* is bigger or smaller than *ThreSize*.

The size of the objective image is $h\times w$, the size of the intermediate matrix is $th\times tw$, the size of original matrix is $H\times W$, the radius of the filter window of Gaussian blur is *r*, the Gaussian matrix array is *gauss*. The the value of the pixel on the coordinate *(i,j)* is $d_{i,j}$ in the objective image.

The corresponding coordinate *(ti,tj)* in terms of $h\times w$ *and* $th\times tw$ is represented by Equ.3.

$$ti = \frac{i}{h}\times th, tj = \frac{j}{w}\times tw \quad (3)$$

The corresponding coordinate *(si,sj)* in terms of *(ti,tj)* is represented by Equ.4.

$$si = \frac{ti}{th}\times H, sj = \frac{tj}{tw}\times W \quad (4)$$

The simultaneous equations are Equ.5. *gauss* is the Gaussian Array of one dimension.

$$\begin{cases} d\_blur_{ti,tj} = \dfrac{\sum_{y=ti-r}^{y\le ti+r}(d_{y,tj}\times gauss_y) + \sum_{x=tj-r}^{x\le tj+r}(d_{ti,x}\times gauss_x)}{2\sum_{t=0}^{t\le 2r+1} gauss_t} \\ d_{y,tj} = d_{(\frac{y}{th}\times sh, \frac{tj}{tw}\times sw)} \\ d_{ti,x} = d_{(\frac{ti}{th}\times sh, \frac{x}{tw}\times sw)} \end{cases} \quad (5)$$

We want to calculate the value of $d_{i,j}$ for every pixel on the objective image. Every $d_{i,j}$ has a corresponding area with the center of $d_{ti,tj}$ from Equ.3. The result of *Gauss blur* in such area is $d\_blur_{ti,tj}$ represented in the first formula in Equ.5. Thus, each $d_{i,j}$ corresponds to a $d\_blur_{ti,tj}$. Second and third formulas are the way of sampling pixels from the original image to fill the intermediate image. For each $d_{ti,tj}$ there must be $d_{si,sj}$ corresponding to the value of $d_{ti,tj}$.

2.3 Box Blur Transiting Interpolation

The main idea of Box Blur Transiting Interpolation(BBTI) is to execute box blur filtering[16] before linear interpolation. Weights for pixels are distributed evenly in box blur.

The pixel value $d\_blur_{i,j}$ with the coordinate *(i,j)* in the blurred image is represented by Equ.6. $d_{x,y}$ is the pixel in the original image.

$$d_{i,j} = \frac{\sum_{x=j-r}^{x\le j+r}\sum_{y=i-r}^{y\le i+r} d_{x,y}}{(2r+1)^2} \tag{6}$$

After box blur, use linear interpolation for generating thumbnails.

The introduction of integral image[17] is a efficient way to enhance the speed of Algorithm BBTI based on the even distribution of weights in the filter window.

1) *(si,sj)* is the coordinate on the original matrix. Its pixel value is $d_{si,sj}$. $S_{si,sj}$ represents the sum of the whole pixels on the region denoted by four coordinates: *(0,0), (0,sj), (si,0), (si,sj)*. Then the equation is Equ.7.

$$S_{si,sj} = \sum_{y=0}^{y<=si}\sum_{x=0}^{x<=sj} d_{y,x} \tag{7}$$

Step1: all the $S_{si,\,sj}$ are calculated by one traverse.

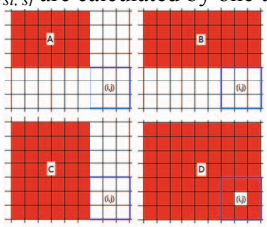


Figure 2. each rectangle cell corresponds to one pixel. (i,j) is the coordinate. A, B, C and D indicates 4 different red areas.

Step2: As the illustration shown in Fig.2, *(i,j)* is the center of the filter window of radius *r*, which is 1 in Fig.2, *P* is the sum of pixels in the window, which can be represented by Equ.8. S_A, S_B, S_C and S_D are the sum of pixel values in corresponding area within the blue rectangle in Fig.2.

$$P = S_D - S_B - S_c + S_A \tag{8}$$

In Equ.9, the blurred pixel value d_blur is in terms of *P* and *r*. The size of the area is $r \times r$, the sum of the pixel values in the area is *P*.

$$d\_blur = \frac{P}{r \times r} \tag{9}$$

Step3: The objective image is generated from this integral image by linear interpolation.

2) It is very time-consuming for the process to be executed on the original image. Thus, this algorithm also introduces the intermediate matrix. The intermediate matrix is an integral image generated from the original image by linear interpolation. The size of the intermediate matrix is between the sizes of the original image and objective image.

Then, the objective pixel values can be calculated by Step2 and generated by linear interpolation from the intermediate matrix in 1 traverse.

The size of the original matrix is $H \times W$. They are *th* and *tw* for intermediate matrix, and *h* and *w* for objective matrix. Thus, the number of iterations is $H \times W + h \times w$ in the original method while it is $th \times tw + h \times w$ after improvement.

2.4 Area Average Interpolation

Algorithm AAI extracts the average value of pixels around the sampling centers on the original images to fill in the objective image.

In the normal interpolations relating to area, objective pixel values are calculated by considering the adjacent pixels in a filter window with a radius of 1 or 2. However, the distort is severe due to the small range of pixels for sampling.

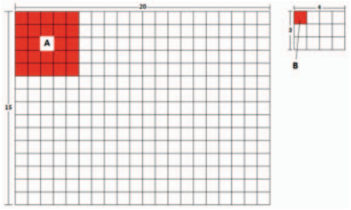


Figure 3. Pixel B corresponds to the original pixels of the red area A

Thus, the range of pixels must be bigger. The height of the original image is *H*, the width is *W*. The height of the objective image is *h*, width is *w*. Obviously, the most ideal size of the area corresponds to a particular pixel in a small size thumbnail is *(H/h)×(W/w)*.

As the illustration shown in Fig.3, the single pixel of B corresponds to the whole pixels in area A. *(i,j)* is a coordinate in the objective image. *(x,y)* are the coordinates of pixels in the area corresponding to *(i,j)*. The pixel value of the original image is $d_{x,y}$. The objective pixel value is $d_{i,j}$. The formula of $d_{i,j}$ in terms of $d_{x,y}$ would be Equ.10.

$$d_{i,j} = \frac{\sum_{x=i\times\frac{H}{h}}^{x\le(i+r)\frac{H}{h}}\sum_{y=j\times\frac{W}{w}}^{y\le(j+r)\frac{W}{w}} d_{x,y}}{\frac{H}{h}\times\frac{W}{w}} \tag{10}$$

However, it is not necessary to maintain the integrity of the whole corresponding area on the original image. Thus, the size of the area for sampling can be reduced. Sampling less pixels in the area with smaller size around the corresponding center(Fig.4).

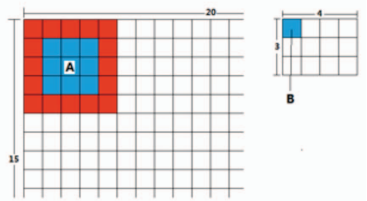


Figure 4. Pixel B corresponds to the original pixels of the blue area A

The new simultaneous equations are Equ.11.

$$\begin{cases} h\_rate = \dfrac{H}{h}, w\_rate = \dfrac{W}{w} \\ h\_rate' \le h\_rate, w\_rate' \le w\_rate \\ d_{i,j} = \dfrac{\sum_{y=(i\times h\_rate)-\frac{h\_rate'}{2}}^{y\le(i\times h\_rate)+\frac{h\_rate'}{2}}\left(\sum_{x=(j\times w\_rate)-\frac{w\_rate'}{2}}^{x\le(j\times w\_rate)+\frac{w\_rate'}{2}} d_{x,y}\right)}{h\_rate'\times w\_rate'} \end{cases} \tag{11}$$

In Equ.11, the new size of the area for sampling is *h_rate'*×*w_rate'*, which is smaller than the original size of *h_rate* ×*w_rate*. *h_rate* is the ratio of *H* to *h* and *w_rate* are is *W* to *w*.

The number of iterations of the loop is *h*×*h_rate'*×*w*×*w_rate'*. The ratio of the running time of the new version to the old version is *(h*×*h_rate'*×*w*×*w_rate')/(H*×*W)*. Nevertheless, the visual quality is negatively affected when the size of *h_rate'*×*w_rate'* is not big enough to represent the comprehension of pixel feature of the area.

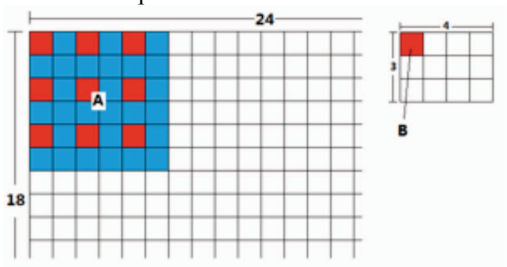


Figure 5. Pixel B corresponds to the original pixels of the blue area A

Thus, another way is to sample discontinuous pixels in a bigger area instead of all pixels in a small area (Fig.5). The practical way is to sample pixels every row and column at the identical interval around the corresponding center to calculate the average value.

In Equ.12, the step of increment of *x* and *y* is 2. *x* starts from *col*, *y* starts from *row*. Since the number of pixels sampled is *1/2*×*1/2=1/4*. of the original number, the ratio of time complexity of the new method to the previous method can be reduced to *1/4*.

$$\begin{cases} h\_rate = \dfrac{H}{h}, w\_rate = \dfrac{W}{w} \\ h\_rate' \le h\_rate, w\_rate' \le w\_rate \\ col = (j \times w\_rate) - \dfrac{w'}{2} \\ row = (i \times h\_rate) - \dfrac{h'}{2} \\ d_{i,j} = \dfrac{\sum_{y=(i \times h\_rate)-\frac{h\_rate'}{2}}^{y \le (i \times h\_rate)+\frac{h\_rate'}{2}} \left(\sum_{x=(j \times w\_rate)-\frac{w\_rate'}{2}}^{x \le (j \times w\_rate)+\frac{w\_rate'}{2}} d_{x,y} \right)}{\frac{h\_rate'}{2} \times \frac{w\_rate'}{2}} \\ x = col, col+2, col+4, \ldots \\ y = row, row+2, row+4, \ldots \end{cases} \quad (12)$$

Typically for a small size thumbnail (*size<=100*×*100*), the integrity of the whole pixels of the sampling area is not that significant. In a particular area for sampling, the value of adjacent pixels are so close that their difference can be ignored. Thus, reducing the number of pixels for sampling can improve the efficiency of the algorithm without much affecting the visual quality.

3 Experimental and Results

3.1 Test Environment

All the algorithms are implemented by C language in Red Hat Enterprise Linux Server release 6.0(x64), 4 cores. The version of ImageMagick is 6.5.4.7-7. el6_5.x86_64, using Shell scripts to call its method thumbnail() to generate thumbnails. Use OpenCV-devel 2.0.0. to generate thumbnail by its method cvResize(). Use libjpeg-turbo to decompress jpeg file and compress bit map. There are two datasets being tested in the experiment, the Paris[18] and INRIA Holidays dataset[19]. Analysis and statistics for data are implemented by Excel and OpenCV.

3.2 Visual Quality Comparison

The visual qualities of objective images are evaluated and compared in the aspects of the subjective evaluation by human eyes and objective evaluations by four numeric standards, including Mean Square Error(MSE), Peak Signal to Noise Ratio(PSNR)[20], Normalized Correlation (NC), Structural Similarity(SSIM)[20]. The sizes of objective images are classified into 2 level of different heights: 200, 424, as the standard data for evaluations.

Considering ImageMagick as a famous and standard open source project in Linux System, the images generated by it are the standard thumbnail for comparison.

TABLE I. THE COMPARISON OF VISUAL QUALITY OF THE GENERATED THUMBNAILS WITH THE HEIGHT OF 424

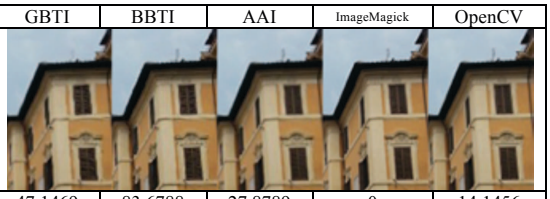
	GBTI	BBTI	AAI	ImageMagick	OpenCV
image with height 424					
MSE	47.1469	83.6788	27.8789	0	14.1456
PSNR	32.9007	30.3880	35.2911	MAX	38.4752
NC	0.9981	0.9968	0.9989	1	0.9992
SSIM	0.9904	0.9836	0.9944	1	0.9967

TABLE II. THE COMPARISON OF VISUAL QUALITY OF THE GENERATED THUMBNAILS WITH THE HEIGHT OF 200

	GBTI	BBTI	AAI	ImageMagick	OpenCV
image with height 200					
MSE	55.1952	49.4907	26.4155	0	85.9494
PSNR	31.8976	32.4123	34.9814	MAX	30.5349
NC	0.9979	0.9981	0.9989	1	0.9972
SSIM	0.9852	0.9894	0.9942	1	0.9837

Thumbnails generated by OpenCV is also included in the table for the comparisons (TABLE I and TABLE II).

According to the subjective evaluation, all the images in TABLE I are similar except the first one on the left, which is generated by algorithm GBTI. Though algorithm GBTI produces an advanced visual quality, its efficiency is lower than others. So, the program run in high speed in compensation for the visual qualities.

The 2nd image in TABLE II is more distorted than others. However, most people cannot perceive it. Generally, the subjective evaluations of all of them are still similar.

According to the objective evaluation, the evaluations in the column of OpenCV perform better than others in the first table. But the differences are not big between Algorithm AAT and OpenCV.

However, in TABLE II, OpenCV does not remain the advanced performance as it does in the previous table. It is the worst in this table. Obviously, Algorithm AAI has the best performance than others.

3.3 Speed Comparison

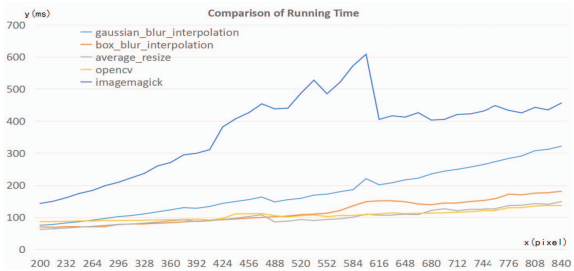


Figure 6. Comparison of running time of generating thumbnails of range of height [200,840] from the two datasets mentioned before. The x-coordinates are the heights of objective images, y-coordinates are the program running time.

The comparison of speeds is based on the average time of each algorithm processing images in the datasets (Fig.6). Average time is calculated from the time of generating the thumbnails with the same specific objective size. The range of the sizes of images represented by the range of heights of objective images in [200,840] with the increment of 16 units. The width is proportional to the heights based on the original image. Because what the paper focus on is thumbnail, the ratio of sizes of the objective image to the original image is less than 0.5 in the proportion of one dimension. When the ratio is more than 0.5, the operation can be done by traditional algorithm effectively, such as Linear interpolation.

As the chart shown in Fig.6, it is obvious that the speeds of ImageMagick and Algorithm GBTI are slower than all three of the others. Three of the others, including Algorithm BBTI, Algorithm AAI and OpenCV run faster. The three lines in the chart are so close that they have to be amplified and analyzed separately.

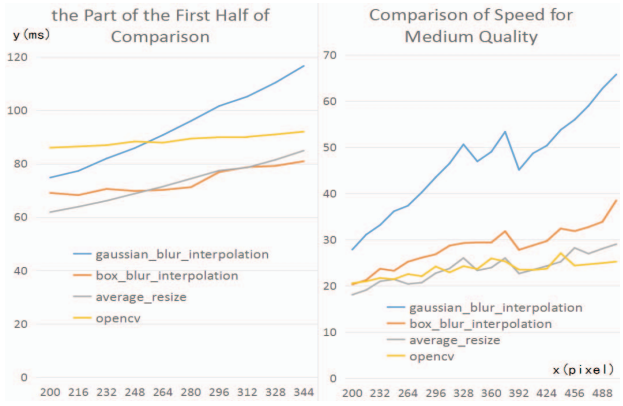


Figure 7 The first line chart on the left shows the amplification of the part of the first half of Fig.6. The second one shows the relationship of running time and size of thumbnails generated from images only with medium qualities.

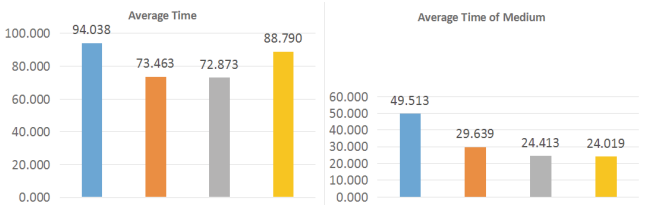


Figure 8 The first histogram on the left shows the average speeds of 4 algorithms corresponding to the part of the first half of Fig.6. The second one shows the average speeds of generating thumbnails from images only with medium qualities.

Generally, in Fig.7 and Fig.8, it shows that the performance of Algorithm AAI is superior to other algorithms in these two charts, but it is not better than others through the whole range of different sizes. Though the difference of lines between OpenCV and Algorithm AAI is bigger than the difference between BBTI and AAI in the left chart in Fig.7 and the difference of heights of their bins is bigger than latter in Fig.8, the lines and bins of them are very close in the right charts in Fig.7 and Fig.8. On the other hand, the difference between Algorithm BBTI and Algorithm AAI is small in the left charts in both of Fig.7 and Fig.8, but it becomes bigger in the right charts. According to the best performance in the range of relatively small sizes of objective images, Algorithm AAI should be chosen for generating small thumbnails. Because small thumbnails have a lower requirement for visual quality and details of images and high requirement for program speed.

4 Conclusion and Discussion

Generally, the objective evaluations of visual qualities indicate that Algorithm AAI has the best performance. However, the distinctions of visual qualities among images generated by different algorithms are so subtle, especially according to the subjective evaluations. They have to be magnified for human eyes to be able to perceive.

Thus, the trivial differences can be ignored according to the specific requirement, in which a large number of small size images need to be generated automatically in the mobile servers. Most users only care about the main ideas of the images without focusing on details. This is also the main function of thumbnails.

Therefore, generating thumbnails with high speed are more significant than generating them with specific details, which is not necessary for browsing. Algorithm AAI, scaling down based on area sampling, can fulfil the requirement properly. Thumbnails generated by this algorithm have the highest visual qualities according to the comparisons and they are clear enough for basic identification. The speed of Algorithm AAI, especially when the difference of sizes between objective image and original image are big, is more advanced than others. Therefore, it is the best algorithm for generating thumbnails in mobile servers in this paper.

This algorithm has been applied to e-mail server of the NetEase enterprise. The thumbnail service is based on this algorithm. For example, the attached files may contain images. When an e-mail is showing its content, the attached images are shown in the form of thumbnails. Its speed of generating

thumbnails is about triple that of the old version of the thumbnail service.

References

[1] V. Patel, K. Mistree, "A review on different image interpolation techniques for image enhancement" *International Journal of Emerging Technology and Advanced Engineering*, pp. 129-33, 2013.
[2] L. Wang, *Video Image Magnification System based on FPGA*. Shenyang: Northeastern University, pp. 21-69, 2009.
[3] K. K, Parhi, *VLSI Digital Signal Processing Systems*, 3rd ed, Wiley-Interscience, pp. 34-451, 2004.
[4] R. C. Gonzalez, R. E. Woods. *Digital Image Processing*, 2nd ed. BEIJING: Publishing House of Electronics Industry, pp. 40-45, 2007.
[5] J. Huang, H. Chen, B. Wang, "Automatic Thumbnail Generation Based on Visual Representativeness and Foreground Recognizability," *International Conference on Computer Vision*, pp. 253-261, 2015.
[6] B. Saha, T. Dasgupta, S. Bhattacharya (2015, August), "An improved content aware image resizing algorithm based on a novel adaptive seam detection technique," In *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, pp. 2311-2316, Aug. 2015
[7] J. Wang, Z. Qu, Y. Chen, "Adaptive Content Condensation Based on Grid Optimization for Thumbnail Image Generation," *Transactions on Circuits & Systems for Video Technology*, vol. 23, no. 1, pp.1-1, 2016.
[8] K. Arai, H. Takei, H, Yamana, "Visual-attention-based thumbnail using two-stage GrabCut," *International Conference on Multimedia Computing and Systems*, pp. 96-101, 2012.
[9] W. Dong, N. Zhou, T. Y. Lee, "Summarization-Based Image Resizing by Intelligent Object Carving," *Transactions on Visualization & Computer Graphics*, 20(1), pp.111-24, 2014.
[10] R. Samadani, T. Mauer, D. Berfanger, J. Clark, S. H. Lim, and D. Tretter, "Honest image thumbnails: Algorithm and subjective evaluation," *Tech. Rep. HPL-2007–88, HP Labs*, 2007
[11] M. M. I. Lie, N. H. Vieira, G. B. Borba, "Automatic Image Thumbnailing Based on Fast Visual Saliency Detection," *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*, pp. 203-206, 2016.
[12] S. A. Esmaeili, B. Singh, and L. S. Davis, "Fast-at: Fast automatic thumbnail generation using deep neural networks," *CoRR*, vol. abs/1612.04811, 2016. 2
[13] Y. Sa, "Improved Bilinear Interpolation Method for Image Fast Processing," *International Conference on Intelligent Computation Technology and Automation*. pp. 308-311, 2014.
[14] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*, 2nd ed. BEIJING: Publishing House of Electronics Industry, pp. 40-45, 2007.
[15] X. D. Jiang, B. Sheng, W. Lin, W. Lu and L. Ma, "Image anti-aliasing techniques for Internet visual media processing: a review", *Journal of Zhejiang University SCIENCE C*, vol. 15, no. 9, pp. 717-728, 2014.
[16] R. Chandel and G. Gupta, Image Filtering Algorithms and Techniques: A Review, *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 10, 2013.
[17] P.A. Viola and M.J. Jones, Rapid object detection using a boosted cascade of simple features, in: CVPR, issue 1, pp. 511–518, 2001.
[18] J. Philbin, Paris dataset [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/parisbuildings/
[19] INRIA, Holidays dataset [Online]. Available: http://lear.inrialpes.fr/people/jegou/data.php
[20] A. Hore A and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," *International Conference on Pattern Recognition*, pp. 2366-2369, Aug. 2010.

TABLE III. THE TABLE OF NOTATIONS

Name	Definition
$gauss_{x,y}$	The weight in the Gaussian distribution matrix at column *x* and row *y*.
(x,y)or (i,j)	The formula of a coordinate. *x(i)* is the index of column; *y(j)* is the index of row.
$d\_blur_{i,j}$	The result of blur operation in the area with the center of *(i,j)*
r	The radius of filter window for sampling from the original image
$d_{i,j}$	The value of pixel at the coordinate *(i,j)*
H	The height of the original image
W	The width of the original image
ObjSize	The size of the objective image in Algorithm GBTI
ThreSize	The size of the threshold to determine when should use linear interpolation or GBTI
h	The height of the objective image
w	The width of the objective image
th	The height of the intermediate matrix
tw	The width of the intermediate matrix
gauss	The one dimension array of Gaussian distribution
$S_{si,sj}$	$S_{si,sj}$ represents the sum of the whole pixels on the region denoted by four coordinates: *(0,0), (0,sj), (si,0), (si,sj)*
P	*P* is the sum of pixels in a filter window
h_rate	The ratio of *H* to *h*
w_rate	The ratio of *W* to *w*