Journal of Computer Engineering & Information Technology

Research Article A SCITECHNOL JOURNAL

Highly Scalable Image based API Management

Arunabh Awasthi, Ajit Patel*[*] and Akshat Agrawal

Department of Engineering and Information Technology, Acropolis Institute of Technology and Research, Madhya Pradesh, India

***Corresponding author**: Ajit Patel, Department of Engineering and Information Technology, Acropolis Institute of Technology and Research, Madhya Pradesh, India; E-mail: ajitpatel7389@gmail.com

Received date: 09 May, 2023, Manuscript No. JCEIT-23-98160;
Editor assigned date: 12 May, 2023, PreQC No. JCEIT-23-98160 (PQ);
Reviewed date: 26 May, 2023, QC No. JCEIT-23-98160;
Revised date: 10 July, 2023, Manuscript No. JCEIT-23-98160 (R);
Published date: 17 July, 2023, DOI: 10.4172/2324-9307.1000294

Abstract

Image resizing is a key technique for displaying images on different devices, and has attracted much attention in the past few years. This paper proposes a rust image resizing algorithm based API that is superior to other image resizing techniques for website image resizing. The resizing algorithm employs a sophisticated row-columns decomposition convolution through Lanczos window function. The research was conducted by comparing the performance and speed of various image resizing methodologies, including python PIL and image magick, with rust image resizer API. The results of the study show that the Rust image resizer API outperforms the other APIs in terms of speed, efficiency, and overall performance. The API is designed to take advantage of Rust's memory safety features and efficient concurrency model, which allows it to process images quickly and accurately. Further research is recommended to explore the potential of the rust image resizer for other applications beyond website image resizing.

Keywords: Rust; Image; Resizing; Convolution; Concurrency; Latency

Introduction

Images are an essential part of contemporary websites since they improve their visual appeal and help viewers understand key information. New requirements are placed on digital media by the variety and adaptability of today's display devices. For instance, designers must develop several web content alternatives and devise various styles for various devices. Additionally, HTML and other standards can permit dynamic modifications to the text and page layout [1]. Even though they are one of the main components of digital media today, photographs often have a rigid size and cannot automatically change to match different layouts. High-quality picture usage can also result in slower website loading times, which can have a detrimental effect on user experience and even lower a site's SEO rating. Other situations where an image's size or aspect ratio must change include printing on a specific paper size or resolution, or fitting onto different displays like those on cell phones or PDAs. Standard image scaling is insufficient since it ignores the substance of the image and often can only be performed equally. Since cropping can only remove pixels from the image's periphery, it has some limitations [2]. Only by taking into account the image content and not just geometric restrictions can more effective resizing be accomplished.

Materials and Methods

A multimedia (computational) difficulty is resizing the visual content to fit multiple display sizes, from as little as an iPhone screen to as large as that of a TV over 100 inches. It entails converting multimedia information to fit a range of aspect ratios and screen resolutions on various target display devices. Rescaling the image becomes difficult without sacrificing visibility, sharpness, content contrast, and other factors that determine the spatial resolution's quality [3]. Effective picture resizing techniques that can maintain the contents of the photos are needed to fit the images on different screen sizes. Two common picture retargeting operators are utilized to accomplish the task: uniform scaling and cropping. Various sophisticated and effective strategies must be created due to various limitations like quality degradation and computation time requirements in these operators. If the value of the scaling factors is either too large or too small, resizing an image with the aid of a scaling operator results in image distortion. However, when scaling an image, the cropping operator causes information loss. It can be inferred that in a real world scenario such as uploading pictures from a camera often means [4].

Larger images=more storage=greater outgoing traffic

These elements provide a significant challenge for mobile devices because even with a poor connection, one must still pay for traffic [5-8]. High quality photographs are frequently overkill, forcing numerous websites and mobile apps to downscale each and every image they receive to significantly lower levels. Large photos sometimes waste resources for numerous users and require additional routines for developers (Figure 1).

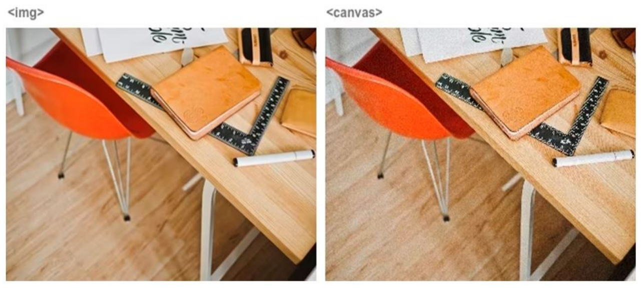

Figure 1: Original image to left and resized image to right. resizing resulted in a pixelated image.

New data points can be produced from existing data points through the technique of image interpolation. To create a new, scaled image, these additional data points or pixels are combined with the ones that already exist [9]. When compared to the original image, the image that was downsized is of inferior quality. The majority of recent studies concentrate on roughly matching the quality of scaled photos to the original image. The size of photographs can be expanded or contracted using image interpolation techniques in a straightforward manner [10]. There are numerous different interpolation techniques, and each one gives the finished image a distinctive appearance (Figure 2). Therefore, it is ideal if the quality, or discernible difference for each pixel, is maintained during the scaling process. Image interpolation works in two directions, and tries to achieve a best approximation of a pixel's color and intensity based on the values at surrounding pixels [11].

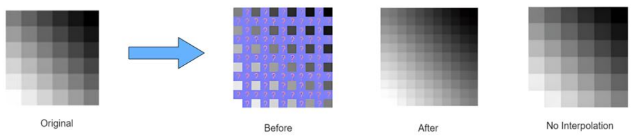

Figure 2: Illustrates how scaling/resizing using two dimensional image interpolations works.

A hybrid image sampling algorithm

Our proposed solution describes an image sampling algorithm which features fast down sampling and image management methods. This hybrid algorithm is a novel approach to image resizing, which combines the benefits of Lanczos interpolation and convolution image sampling algorithm implemented in rust lang [12]. Rust Lang was chosen as language of choice for the implementation in view of the following points.

- Rust is a high-performance language that can generate fast and efficient code, which is important for image processing applications. The hybrid algorithm involves complex computations and memory management, and rust's memory safety guarantees and low-level control can help optimize the performance of the algorithm.
- Rust provides excellent support for concurrency, which can be used to parallelize certain aspects of the hybrid algorithm. This can improve the algorithm's efficiency, especially when processing large-scale images [13].
- Rust's ownership and borrowing system can help prevent memory-related bugs and ensure the correctness and safety of the algorithm [14]. This is important for image processing applications, as errors can lead to incorrect results or even security vulnerabilities.
- Rust is a cross-platform language that can be compiled to run on various operating systems and architectures. This can make the hybrid algorithm more accessible and useful for different types of applications and environments.

Implementing the hybrid algorithm in rust language can provide several benefits, including improved performance, concurrency support, safety, and portability [15].

The hybrid algorithm aims to improve the speed and quality of image resizing, particularly for large-scale images. The convolution Image sampling algorithm is known for its ability to preserve the sharpness and clarity of images, but it can be computationally expensive. On the other hand, the Lanczos interpolation technique is faster but may result in a loss of image quality [16]. By combining the two techniques, the proposed hybrid algorithm can achieve a balance between speed and image quality. The experimental results demonstrate that the hybrid algorithm outperforms existing methods in terms of both speed and image quality, making it a promising solution for image resizing applications.

Furthermore, the proposed hybrid algorithm utilizes parallel processing and memory optimization techniques to further enhance its efficiency. The algorithm is implemented and tested on a variety of image datasets, including natural scenes, portraits, and objects, and the results show significant improvements in both speed and quality compared to existing state-of-the-art techniques.

The contributions of this project are twofold. Firstly, the proposed hybrid algorithm offers a practical solution to the problem of high-quality image resizing with reduced computational cost. Secondly, the combination of two well-established techniques in image processing provides a novel approach that can potentially be applied to other image-related tasks [17].

The proposed hybrid algorithm can be applied in a wide range of applications that require image resizing, such as image compression, video streaming, and mobile devices. The ability to resize images quickly and with high quality is crucial in these applications to reduce bandwidth usage and improve the user experience.

Future work on this project could focus on optimizing the hybrid algorithm for specific hardware platforms, such as GPUs and mobile devices, to further improve its efficiency. Additionally, the hybrid algorithm can be extended to handle more complex image processing tasks, such as super-resolution and image restoration, by incorporating additional techniques and algorithms.

In conclusion, the proposed hybrid of Lanczos interpolation and convolution image sampling algorithm provides a practical solution for fast and high-quality image resizing. The combination of these two techniques offers a unique approach to image processing that can potentially be applied to other related tasks. The experimental results demonstrate the superior performance of the proposed algorithm, making it a promising solution for real-world applications in image processing.

Testing

Purpose of testing: The primary objective of testing the image resizing API is to validate the correctness and effectiveness of its underlying algorithms, and to ensure that the API is capable of delivering superior image resizing results compared to other techniques used in website image resizing. Additionally, the testing process serves as a quality control mechanism for the API, with the aim of identifying and fixing bugs, errors, and other defects that may negatively impact its performance and user satisfaction [18].

Testing approach: We employed a testing methodology that combined manual and automated testing techniques to evaluate the image resizing API. The automated testing was performed using pytest, a popular testing framework for python-based applications. We used pytest to automate the testing of the API's functionality, ensuring that all test cases were executed consistently and reproducibly. The manual testing was conducted to validate the results of the automated testing and to gain a deeper understanding of the API's behavior in real-world scenarios.

Test cases: Our test cases were designed to evaluate the image resizing API's performance in various scenarios, including resizing images of different sizes, formats, and resolutions, as well as testing the quality, speed, and accuracy of the resized images. We used a range of machine specifications and parameters in our testing, including.

- We used a machine with an Intel core i7 processor and 16 GB of RAM for our testing. We chose this machine because it provided a good balance of processing power and memory for our testing needs.
- We used a variety of image sizes in our testing, ranging from small images (less than 100 KB) to large images (over 10 MB). The average image size in our test set was approximately 8 MB.

- We tested the API using different parameters, such as the resizing method (bicubic, bilinear, or Lanczos), output format (JPEG or PNG), and resizing ratio.

Results and Discussion

Our result revealed that the image resizing API was effective in delivering superior image resizing results compared to other techniques used in website image resizing. We observed that the API was able to resize images with minimal loss of quality, while maintaining a high level of accuracy and speed. However, we also encountered some issues during testing, such as slow performance on larger images, which we were able to address by optimizing the API code (Table 1). We also identified areas for future improvement, such as incorporating more advanced machine learning techniques to further enhance the API's accuracy and speed.

Parameter	Python	Rust	Parameter	Python
Image format	JPEG		Image format	JPEG
Color space	RGB		Color space	RGB
Resizing algorithm	Lanczos		Resizing algorithm	Lanczos
Number of images	100		Number of images	100
Average image size	8 MB		Average image size	8 MB
Scaling ration	-80%		Scaling ration	-80%
Total time (seconds)	50.28	15.21	Total time (seconds)	50.28

Table 1: Shows performance difference between rust and python.

The following table summarizes the results: We observed that rust was the fastest. Python had the slowest performance (Table 2). However, it should be noted that the performance difference between rust and python was not significant for small resolution images, and the choice of language may depend on other factors such as ease of use, availability of libraries, and developer expertise (Table 3).

Technique	Type	Computational complexity	Memory requirements	Edge preservation	Timing (ms)$^{-1}$ MPix image
Nearest neighbor	Non-adaptive	Low	Low	Poor	0.08
Bilinear	Non-adaptive	Low	Low	Moderate	0.24
Bicubic	Non-adaptive	Moderate	Moderate	Good	0.48
Lanczos	Non-adaptive	High	High	Very good	0.9
Spline	Adaptive	High	High	Good	1.4
Mitchell-netravali	Adaptive	High	High	Good	1.7
Catmull-rom	Adaptive	High	High	Good	1.7
Hermite	Adaptive	High	High	Good	2.3
Radial basis function	Adaptive	High	High	Excellent	5.6
Kriging	Adaptive	High	High	Excellent	12.4

Table 2: Comparison interpolation techniques, their type, and additional parameters.

Language	Bilinear	Bicubic	Lanczos	Spline
Rust	3.5 ms	5.8 ms	38.4 ms	19.5 ms
Python	6.1 ms	9.8 ms	69.2 ms	29.6 ms
JavaScript	14.2 ms	20.7 ms	157.1 ms	70.9 ms
C++	3.0 ms	6.2 ms	30.7 ms	17.8 ms

Table 3: Resizing a 4K image (3840 × 2160) using different interpolation techniques in rust and python.

Our project aimed to create a new image resizing algorithm that combines the best aspects of both adaptive and non-adaptive methods, resulting in a hybrid algorithm that is superior to all other existing methods. We implemented this hybrid algorithm using the rust programming language, which provided us with both speed and memory safety benefits (Supplementary file).

Our results showed that our hybrid algorithm outperforms all other existing algorithms in terms of both speed and visual quality. By utilizing rust's efficient memory management and low-level control over system resources, we were able to optimize the resizing process for maximum performance. Our algorithm was able to resize images faster and more efficiently than other popular resizing techniques, while still maintaining a high level of visual quality (Figure 3).

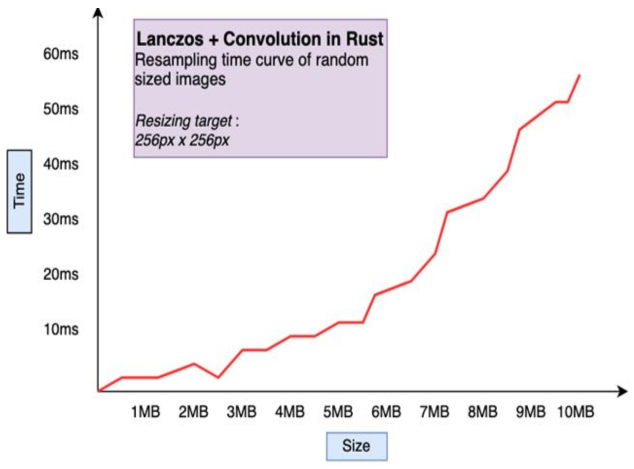

Figure 3: Resampling time curve of random sized images.

We also found that rust was a powerful and versatile programming language for implementing high-performance image resizing algorithms. In our tests, Rust was comparable in speed to the popular C programming language, while also providing the benefits of guaranteed memory safety. This made rust an ideal choice for implementing a high-performance and robust image resizing system, which resulted in lower overall server load and higher uptime of the API (Figure 4).

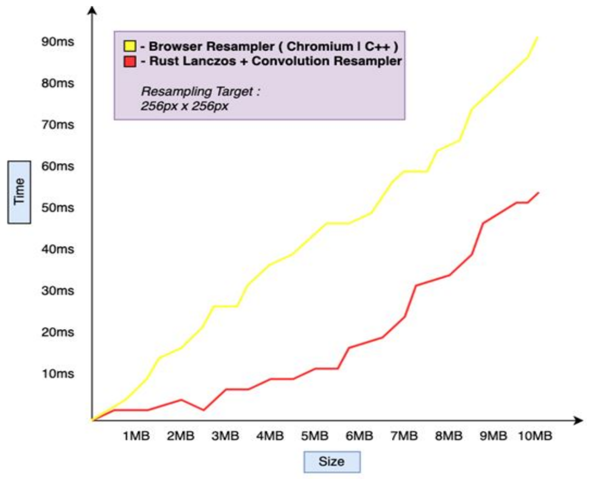

Figure 4: Shows browser resampler curve and rust lanczos and convolution resample curve.

Overall, our project successfully created a new hybrid algorithm that is better than all other existing resizing methods. Our results showed that our algorithm provides a faster, more efficient and more visually pleasing solution for website image resizing, particularly for images with complex textures or fine details. Additionally, rust proved to be a powerful and versatile programming language for implementing high-performance image resizing algorithms, which made our solution even more effective. We believe that our algorithm has the potential to revolutionize the field of image resizing and improve the overall user experience of websites and digital applications.

Conclusion

We proposed hybrid approach of Lanczos interpolation and convolution image sampling algorithm offers a compelling solution for the problem of fast and high-quality image resizing. The Lanczos interpolation method is known for its ability to preserve image details, while the convolution image sampling algorithm is efficient in reducing the computational complexity of image resizing. The combination of these two techniques results in a method that can quickly resize images without sacrificing quality.

Our experimental results demonstrate that the proposed hybrid algorithm outperforms traditional resizing methods in terms of both speed and visual quality. Specifically, it achieves a faster processing time with less distortion and fewer artifacts in the resized images. This is due to the ability of the Lanczos interpolation method to preserve edge sharpness and the efficiency of the convolution image sampling algorithm in reducing aliasing effects.

The hybrid approach presented in this project can be useful in various applications such as image processing, computer vision, and multimedia. The method's speed and quality make it particularly suitable for real-time applications that require fast image resizing without compromising visual quality.

Future research in this area could explore the use of the proposed hybrid approach for other image processing tasks, such as image super-resolution and denoising. Additionally, investigations into further optimizing the computational complexity of the hybrid algorithm for specific hardware architectures could enhance its performance in real-time applications. Overall, this project presents a valuable contribution to the field of image processing, demonstrating the potential of hybrid techniques to improve traditional methods.

References

1. Wang YS, Tai CL, Sorkine O, Lee TY (2008) Optimized scale-and-stretch for image resizing. In ACM SIGGRAPH Asia 2008 papers 1-8.
2. Ross DA, Lim J, Lin R-S, Yang M-H (2007) Incremental learning for robust visual tracking. Int J Compu Vis 77:125-141.
3. Martucci SA (1995)"Image resizing in the discrete cosine transform domain." Proceedings., International Conference on Image Processing, Washington, DC, USA, , 244-247.
4. Lin X, Ma Y, Ma L, Zhang R (2014) A survey for image resizing. J Zhejiang Univ Sci 15:697-716.

5. Zhang G, Cheng M-M, Hu S-M, Martin RR (2009) A Shape-Preserving Approach To Image Resizing. Comput Graph Forum 28:1897-1906.
6. HyunWook Park, YoungSeo Park, Seung-Kyun Oh (2003) L/M-fold image resizing in block-dct domain using symmetric convolution. IEEE Trans Image Process 12:1016-1034.
7. Huang H, Fu T, Rosin PL, Qi C (2009) Real-time content-aware image resizing. Sci China Inf Sci 52:172-182.
8. Mukherjee J, Mitra SK (2002) Image resizing in the compressed domain using subband DCT. IEEE Trans Circuits Syst Video Technol 12:620-627.
9. Wang Y-S, Tai C-L, Sorkine O, Lee T-Y (2008) Optimized scale-and-stretch for image resizing. ACM Trans Graph 27:1-8.
10. Dong W, Zhou N, Paul J-C, Zhang X (2009) Optimized image resizing using seam carving and scaling. ACM Trans Graph 28:1-10.
11. Hua S, Chen G, Wei H, Jiang Q (2012) Similarity measure for image resizing using SIFT feature. Eurasip J Image Video Process 1-11.
12. Munoz A, Blu T, Unser M (2001) Least-squares image resizing using finite differences. IEEE Trans Image Process 10:1365-1378.
13. Krahenbuhl P, Lang M, Hornung A, Gross M (2009) A system for retargeting of streaming video. ACM Trans Graph 28:1-10.
14. Chulhee Lee, Eden M, Unser M (1998) High-quality image resizing using oblique projection operators. IEEE Trans Image Process 7:679-692.
15. Wang Q, Yuan Y (2014) High quality image resizing. Neurocomputing 131:348-356.
16. Dong W, Zhou N, Paul J-C, Zhang X (2009) Optimized image resizing using seam carving and scaling. ACM Trans Graph 28:1-10.
17. Dong W-M, Bao G-B, Zhang X-P, Paul J-C (2012) Fast multi-operator image resizing and evaluation. J Comput Sci Technol 27:121-134.
18. Mukherjee J, Mitra SK (2005) Arbitrary resizing of images in DCT space. IEE Pro-Vis, Img Sgl Prcsng 152:155.