Task Parallelism in WebAssembly: Porting and Evaluating OpenMP-driven Image Processing Pipelines

Algorithm Engineering 2026 Project

Daniel Motz
Friedrich Schiller University
Jena, Germany
daniel.motz@uni-jena.de

Leonard Teschner
Friedrich Schiller University
Jena, Germany
leonard.teschner@uni-jena.de

Mher Mnatsakanyan
Friedrich Schiller University
Jena, Germany
mher.mnatsakanyan@uni-jena.de

Abstract

TBC

Keywords

task parallelism, WebAssembly, OpenMP, image processing, C++

ACM Reference Format:

Daniel Motz, Leonard Teschner, and Mher Mnatsakanyan. 2026. Task Parallelism in WebAssembly: Porting and Evaluating OpenMP-driven Image Processing Pipelines: Algorithm Engineering 2026 Project. In *Proceedings of AEPRO 2026: Algorithm Engineering Projects (AEPRO 2026).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Possible Research Questions

(1) Easily extensible Open Source Pipeline, ready to use out of the box with default options giving a perfectly good image, completely written in C++ and licensed under MIT-0 and CeCILL-C, which are both commercially available (compared to e.g. unpaper) and has no big dependencies (unlike unpaper, which uses ffmpeg) and only uses CImg, which is portable (uses OpenMP), simple, self-contained, and lightweight.
(2) all of the above, and it can be compiled to run as WASM! The question would be: How easy is it to compile a C++ image processing pipeline with OpenMP and CImg to WASM? (This would show, that you can take a lightweight C++ library and just put it in the browser with great parallelisation.)
(3) We could additionally try and implement one of the filters or binarization methods in WASM SIMD (which is not platform specific) and show which of the two is faster. The conclusion could then be: is it worth it to manually write SIMD for WASM or is OpenMP good enough (which would be an interesting conclusion)? -> future work

What would be necessary to make our project run in WASM?

(1) Minimum working example of OpenMP can be found here: WASM OpenMP Examples
(2) Compile using O3 and look for automatic SIMD usage
(3) Put SIMD where it was not put by the compiler
(4) Some background on the compiler (WASI)?

1.1 Background

The execution of performance-critical applications in web browsers has changed with the introduction of WebAssembly (WASM). Although WASM offers near-native execution performance, utilising multi-core architectures with parallel programming is still a relatively new field. While native C++ programs use OpenMP to easily implement parallelism, support for OpenMP in WASM remains uncertain.

Was ist WASM?

Was machen wir mit unserer Pipeline?

(1) easy to use
(2) out of the box good results
(3) low dependencies (only CImg (doesnt have other dependencies) and OpenMP)
(4) only image enhancement, no OCR
(5) MIT-0 or CeCILL-C license (commercially usable)
(6) modular (users can choose which steps to apply)

In this paper, we examine whether and how well OpenMP-based parallelism is compatible with WebAssembly by using a lightweight image enhancement pipeline implemented in C++ with OpenMP. First, we demonstrate that the pipeline can be compiled for WASM, then we evaluate its performance in native execution and in WebAssembly. We then compare the results.

1.2 Related Work

Since the introduction of WASM in 2017 [9], research has progressed in a variety of ways. For instance, significant investment has gone into the 'internal' WASM runtime, either to add features or to investigate its performance and energy efficiency . Another area of research has investigated the use of WASM in various application [22] areas, such as web applications, the Internet of Things, and high-performance computing [22]. Of particular mention is the work of Chadha et al., who examine the use of WASM in the HPC domain in [7]. Since parallelisation is essential for HPC applications, they have developed a WASM embedder that enables the MPI library to be used in WASM applications.

When improving images and analyzing digitized documents using OCR text recognition and other recognition systems, segmenting the background and foreground is an essential step [21]. Segmentation is implemented using binarization. There are traditional methods for binarization, which use a global [14], local [4, 6, 15] or mixed thresholds [20, 21]. These are used to classify the pixels as foreground and background, pixel by pixel. Image feature methods such as edge detection [12, 21] and fuzzy logic [17, 21] have also been used for binarization. In recent years, deep learning binarization methods have been evolved in addition to the traditional methods. These are based on convolutional neural networks, generative adversarial networks, or attention mechanisms [21].

In addition to methods that solely rely on binarization, there are approaches that combine several image processing methods to improve the quality of text images. For example, Alqudah et al. [2] present a pipeline that combines entropy filters and morphological operations with binarization. In [13] Niblack's binarization is combined with a Laplace edge filter and global optimization. Another approach is described by Vlasceanu et al. [18]. This approach combines different binarization methods and uses a voting mechanism to decide which pixels should be assigned to the foreground or background.

1.3 Our Contribution

To improve scanned and photographed text images, we present a pipeline that combines several image processing methods. It includes steps such as deskewing, contrast enhancement, noise reduction, binarization, despeckle, and morphological operations. The binarized result of the pipeline can also be converted back into a color image.

The pipeline is modular and user-friendly. Users can select the steps they want to apply to customize the pipeline to their specific use cases. Although the method parameters can be customized individually, they are configured with best practice values by default to ensure a high level of user-friendliness.

The pipeline is provided both as a executable and as a C++ library containing the individual methods. To efficiently process large amounts of image data, the implementation uses parallelization via OpenMP and other optimization methods such as loop blocking. The CImg library [16] is used for the basic image processing methods.

1.4 Outline

This paper is structured as follows: Section 2 provides a detailed description of the developed pipeline and its methods. Section 3 demonstrates the performance of our pipeline using experiments. Finally, Section 4 summarises our results and provides an outlook on possible future work.

2 The Pipeline

Many approaches and best practices already exist for improving the quality of scanned images as seen in section 1.2. We have developed a pipeline that combines several of these methods in order to achieve **potentially** good results. Users can choose which steps to apply from the pipeline. The individual methods of the pipeline are shown in algorithm 1.

Algorithm 1 Image Text Enhance Pipeline

(1) convert image to grayscale
(2) Deskew (if requested)
(3) Contrast enhancement
(4) Denoising
(5) Binarization
(6) Despeckle (if requested)
(7) Morphological operations (if requested)
(8) Color passthrough

The individual methods of the pipeline are explained in more detail below.

2.1 Convert image to grayscale

All pipeline methods work on grayscale images. Therefore, the first step is to convert the input image into a grayscale image. This is achieved by applying the weighted sum $Y = 0.299R + 0.587G + 0.114B$, as standardised by the International Telecommunication Union [11], to each pixel. The result is a grayscale image in which the brightness value of each pixel, represented by Y, corresponds to that of the original RGB-pixel. All steps in the pipeline are performed on the converted grayscale image in-place after the conversion.

2.2 Deskew

To ensure that the text in the image is horizontally aligned, a deskew step is performed at the user's request. Text analysis methods such as OCR benefit in particular from horizontally aligned texts [3]. The deskewing algorithm uses the projection profile method, which is similar to the one described in [1]. First the image is binarized using Sauvola's method[1]. In the second step, the angle that maximizes the variance of the horizontal projections is determined. Finally, the image is rotated by this angle to correct the skew. The advantages of this method are that it is polarity-safe[2], uses a coarse-to-fine angle search[3] for efficiency, and employs Neumann boundary conditions [4] to avoid black corners.

2.3 Contrast enhancement

To correct contrast problems caused by varying lighting conditions while digitizing documents, a contrast enhancement step is performed. This helps with binarization in the further course [2]. A robust, linear contrast stretching algorithm is applied. The lower 1% and upper 1% of intensities are truncated to ignore outliers. The remaining range is then stretched to the full range from 0 to 255 (black to white).

2.4 Denoising

Digital images are subject to noise due to the way they are captured, compressed, or transmitted, resulting in the loss of image information. The presence of noise limits the effectiveness of image analysis steps, among other things [8]. Two simple and two adaptive filter methods are available for the user to choose from. *CImg* offers two simple filtering methods, which are also available in the Pipeline: a Gaussian filter with Neumann boundary conditions and a nonlinear median filter [16]. The Gaussian filter is a low-pass filter and thus blurs the image. The median filter is a non-linear filter that takes the pixel value that represents the median of the neighboring pixels in the window to contain edges [24]. In addition to the two simple filters, two adaptive filter methods are offered. An adaptive Gaussian filter that uses a variable standard deviation to blur less at edges and blur more in flat regions with high variance. In addition to the simple median filter, an adaptive median filter is provided.

[1] see section 2.5 or [15]
[2] detects light from dark backgrounds
[3] tbc
[4] tbc

This is particularly well suited for removing impulse noise (salt and pepper noise) while preserving edges and fine details. It starts with a 3×3 window, which is expanded to a defined maximum window size when impulse noise is detected, while non-impulse pixels remain unchanged. This makes it ideal for removing scan speckle[5] in text images.

2.5 Binarization

Segmenting the foreground and background is a good approach to improve image quality. It also represents the first step in recognition systems such as OCR [2, 21]. Since binarization methods have been well researched in the literature, several are available in the pipeline for the users to choose from. All methods are thresholding methods that calculate a threshold T for each pixel (T_g for global and T_w for local terms) and compare the pixel value $i(x, y)$ with the threshold. This type of binarization is simple and efficient [5].

The simplest method is the Otsu method [14], which calculates a global threshold while minimizing the intraclass variance. This is shown in equations (1) and (2), where $w1(t)$ and $w2(t)$ are the probabilities of the two classes (foreground and background) and $\sigma^2\_1(t), \sigma^2\_2(t)$ are the variances of the two classes. This makes the method efficient, but also susceptible to overlaps and poor intensity distributions [5].

$$\sigma^2\_w(t) = w1(t) * \sigma^2 1(t) + w2(t) * \sigma^2\_2(t) \tag{1}$$

$$T_g = \underset{t}{\operatorname{argmin}}\ \sigma^2\_w(t) \tag{2}$$

One adaptive local threshold approach is the Sauvola method [15]. This method calculates a local threshold T_w for each pixel in a window around that pixel. This method is resistant to uneven lighting. The local mean m_w and local standard deviation σ_w of the window are used, as well as the parameter R, which represents the dynamic range of the standard deviation. The sensitivity of the threshold can be corrected using the parameter k. The threshold is represented in equation (3) [15, 21]. An optional parameter $delta$ has been added to further fine-tune the threshold.

$$T_w = m_w * \left(1 + k * \left(\frac{\sigma_w}{R} - 1\right)\right) - delta \tag{3}$$

Another adaptive method that dynamically adjusts the required window size is the method developed by Bataineh et al. [4]. This method first calculates a global threshold T_{con} (4), which classifies the pixel values into foreground (black), background (white), and confusion values (red) (5). Based on the ratio p of foreground to confusion values and the global standard deviation σ_g, a primary window size PW_{size} is selected (6). If the number of confusion values in the window exceeds the number of foreground values, half the window size SW_{size} is used.

$$T_{con} = m_g - \frac{m_g^2 * \sigma_g}{(m_g + \sigma_g) * (0.5 max_{level} + \sigma_g)} \tag{4}$$

$$I = \begin{cases} \text{black}, & i(x,y) \le T_{con} - \left(\frac{\sigma_g}{2}\right), \\ \text{red}, & T_{con} - \left(\frac{\sigma_g}{2}\right) < i(x,y) < T_{con} + \left(\frac{\sigma_g}{2}\right), \\ \text{white}, & i(x,y) \ge T_{con} + \left(\frac{\sigma_g}{2}\right), \end{cases} \tag{5}$$

$$PW_{size} = \begin{cases} \left(\frac{I_h}{4}, \frac{I_w}{6}\right), & p \ge 2.5 \text{ or } (\sigma_g < 0.1 * max_{level}), \\ \left(\frac{I_h}{30}, \frac{I_w}{20}\right), & 1 < p < 2 - 5 \text{ or } (I_h + I_w < 400), \\ \left(\frac{I_h}{40}, \frac{I_w}{30}\right), & p \le 1, \end{cases} \tag{6}$$

The local threshold T_w is then calculated for each window (7). This uses an adaptive standard deviation value $\sigma_{adaptive}$ based on the maximum and minimum values of the standard deviation of all windows (8) [4]. Due to the adaptive window size and adaptive threshold value, which are based on the image features, this method is robust against various challenges such as thin pen strokes and low-contrast images. However, excessive background remains unavoidable [21].

$$T_w = m_w - \frac{m_W^2 - \sigma_W}{(m_g + \sigma_W) \times (\sigma_{adaptive} + \sigma_W)} \tag{7}$$

$$\sigma_{adaptive} = \frac{\sigma_W - \sigma_{min}}{\sigma_{max} - \sigma_{min}} \tag{8}$$

2.6 Despeckle

A despeckle step is offered to remove small spots that arise or remain during binarization. It removes smaller, connected components (speckles) from the binarized image. The method *get_label*() from the CImg library [16] is used to detect the connected components. It calculates the connected components using the algorithm by Hesselinks et al. [10, 16].

2.7 Morphological operations

After segmenting the foreground and background, small holes or islands may appear. These can be removed using opening and closing operations [23]. The pipeline offers the option of applying the morphological operations dilation and erosion. Dilation expands bright (white) areas. In binary images, this can connect broken characters or thicken strokes. Erosion reduces bright areas [6]. In binary images, this can remove small noise points or make strokes thinner [19, 23].

2.8 Color passthrough

As the final step in the pipeline, the binarized image can be used to obtain the color values of the original image. To do this, the binarized and enhanced image is used as a mask. All pixels that were classified as foreground (black) in the binarized image are replaced by the color of the underlying pixel in the original image. As shown in equation (9), $I'(x, y, z)$ describes the result, $I_{original}(x, y, z)$ describes the colored original image, and $i(x, y)$ describes the binarized image.

[5] tbc

[6] and expands dark areas

$$I'(x,y,z) = \begin{cases} I_{original}(x,y,z), & \text{if } i(x,y) = \text{black}, \\ \text{white}, & \text{else} \end{cases} \tag{9}$$

3 Experiments

4 Conclusions

References

[1] Teruo Akiyama and Norihiro Hagita. 1990. Automated entry system for printed documents. *Pattern Recognition* 23, 11 (1990), 1141–1154. doi:10.1016/0031-3203(90)90112-X

[2] Musab Kasim Alqudah, Mohammad F. Bin Nasrudin, Bilal Bataineh, Mashal Alqudah, and Arwa Alkhatatneh. 2015. Investigation of binarization techniques for unevenly illuminated document images acquired via handheld cameras. In *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*. 524–529. doi:10.1109/I4CT.2015.7219634

[3] Wuzhida Bao, Cihui Yang, Shiping Wen, Mengjie Zeng, Jianyong Guo, Jingting Zhong, and Xingmiao Xu. 2022. A Novel Adaptive Deskewing Algorithm for Document Images. *Sensors* 22, 20 (2022). doi:10.3390/s22207944

[4] Bilal Bataineh, Siti Abdullah, Khairuddin Omar, and Mohammad Faidzul Nasrudin. 2011. Adaptive Thresholding Methods for Documents Image Binarization, Vol. 6718. 230–239. doi:10.1007/978-3-642-21587-2_25

[5] Bilal Bataineh, Mohamed Tounsi, Nuha Zamzami, Jehan Janbi, Waleed Abdel Karim Abu-ain, Tarik AbuAin, and Shaima Elnazer. 2025. A Comprehensive Review on Document Image Binarization. *Journal of Imaging* 11, 5 (2025). doi:10.3390/jimaging11050133

[6] Derek Bradley and Gerhard Roth. 2007. Adaptive Thresholding using the Integral Image. *J. Graphics Tools* 12 (01 2007), 13–21. doi:10.1080/2151237X.2007.10129236

[7] Mohak Chadha, Nils Krueger, Jophin John, Anshul Jindal, Michael Gerndt, and Shajulin Benedict. 2023. Exploring the Use of WebAssembly in HPC. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming* (Montreal, QC, Canada) *(PPoPP '23)*. Association for Computing Machinery, New York, NY, USA, 92–106. doi:10.1145/3572848.3577436

[8] Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. 2019. Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art* 2, 1 (2019), 7. doi:10.1186/s42492-019-0016-7

[9] Andreas Haas, Andreas Rossberg, Derek L. Schuff, Ben L. Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. 2017. Bringing the web up to speed with WebAssembly. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Barcelona, Spain) *(PLDI 2017)*. Association for Computing Machinery, New York, NY, USA, 185–200. doi:10.1145/3062341.3062363

[10] Wim H. Hesselink, Arnold Meijster, and Coenraad Bron. 2001. Concurrent determination of connected components. *Science of Computer Programming* 41, 2 (2001), 173–194. doi:10.1016/S0167-6423(01)00007-7

[11] International Telecommunication Union. 2011. *Recommendation ITU-R BT.601-7: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*. Recommendation BT.601-7. ITU Radiocommunication Sector (ITU-R), Geneva, Switzerland. https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-E.pdf

[12] Shijian Lu, Bolan Su, and Chew Lim Tan. 2010. Document image binarization using background estimation and stroke edges. *International Journal on Document Analysis and Recognition (IJDAR)* 13, 4 (2010), 303–314. doi:10.1007/s10032-010-0130-8

[13] Sergey Milyaev, Olga Barinova, Tatiana Novikova, Pushmeet Kohli, and Victor Lempitsky. 2013. Image Binarization for End-to-End Text Understanding in Natural Images. In *2013 12th International Conference on Document Analysis and Recognition*. 128–132. doi:10.1109/ICDAR.2013.33

[14] Nobuyuki Otsu. 1979. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics* 9, 1 (1979), 62–66. doi:10.1109/TSMC.1979.4310076

[15] J. Sauvola and M. Pietikäinen. 2000. Adaptive document image binarization. *Pattern Recognition* 33, 2 (2000), 225–236. doi:10.1016/S0031-3203(99)00055-2

[16] Jean-Philippe Tarel. 2024. CImg: A Simple C++ Toolkit for Image Processing. https://cimg.eu. Version 3.x.

[17] Li-Jing Tong, Kan Chen, Yan Zhang, Xiao-Ling Fu, and Jian-Yong Duan. 2009. Document Image Binarization Based on NFCM. In *2009 2nd International Congress on Image and Signal Processing*. 1–5. doi:10.1109/CISP.2009.5305330

[18] Giorgiana Violeta Vlăsceanu, Caraman Ghenadie, Răzvan Nițu, and Costin-Anton Boiangiu. 2022. A voting method for image binarization of text-based documents. In *2022 21st RoEduNet Conference: Networking in Education and Research (RoEduNet)*. 1–5. doi:10.1109/RoEduNet57163.2022.9921086

[19] Martin Werner. 2020. *Digitale Bildverarbeitung*. Springer Vieweg Wiesbaden. doi:10.1007/978-3-658-22185-0

[20] You Yang. 2008. OCR Oriented Binarization Method of Document Image. In *2008 Congress on Image and Signal Processing*, Vol. 4. 622–625. doi:10.1109/CISP.2008.262

[21] Zhengxian Yang, Shikai Zuo, Yanxi Zhou, and Jianwen He, Jinlong andShi. 2024. A Review of Document Binarization: Main Techniques, New Challenges,and Trends. *Electronics* 13, 7 (2024). doi:10.3390/electronics13071394

[22] Yixuan Zhang, Mugeng Liu, Haoyu Wang, Yun Ma, Gang Huang, and Xuanzhe Liu. 2025. Research on WebAssembly Runtimes: A Survey. *ACM Trans. Softw. Eng. Methodol.* 34, 8, Article 239 (Oct. 2025), 47 pages. doi:10.1145/3714465

[23] Yujin Zhang and Tsinghua University Press. 2017. *Image Engineering. Vol 2, Image Analysis* (1st ed. ed.). De Gruyter,, Berlin ;.

[24] Yu-Jin Zhang. 2017. *Image engineering*. De Gruyter, Berlin.