Caio Teles Cunha

Implementação eficiente da propagação diagonal em algoritmos de frente de onda irregulares em GPUs
Projeto de Pesquisa

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Orientador: Prof. Dr. George Luiz Medeiros Teodoro
Coorientador: Willian Barreiros Jr.

Belo Horizonte, Minas Gerais
2025

1 Introdução

Histopatologia é o campo de estudo das doenças por meio da análise de tecidos, principalmente por meio do uso de lâminas. Com o desenvolvimento de técnicas de histopatologia digital, soluções de Diagnóstico Auxiliado por Computador (CAD) foram se tornando viáveis. Especificamente, a histopatologia digital foca no processamento e análise de imagens de lâminas histológicas inteiras, processo também conhecido como *Whole Slide Imaging* (WSIs)[1]. Por meio de scanners automáticos é possível a captura de diversas lâminas inteiras para WSIs. O resultado são imagens de altíssima resolução, comumente de $50K \times 50K$ até $100K \times 100K$ pixels[2]. Desde 2004, técnicas de WSIs têm sido amplamente utilizadas em CAD [1], focando em soluções de segmentação, detecção e identificação de características celulares.

Soluções de CAD são de grande importância, pois reduzem o custo de análise de imagens de tecido, tarefa laborosa que necessita de profissionais treinados, algo que pode ser escasso. No âmbito de pesquisa, soluções de CAD são desenvolvidas para auxiliar na identificação de padrões não-óbvios nessas lâminas a um custo razoável [1].Naturalmente, soluções de WSI se baseam em algoritmos de visão computacional e processamento de imagens.

Nesse domínio estão as operações morfológicas, que são uma classe importante de ferramentas de processamento de imagem, computadas em pixels individuais e sua vizinhança (geralmente componentes 4/8-conectados) usando uma abordagem complexa baseada em preenchimento (flood-filling). Exemplos dessas operações incluem reconstrução morfológica, máximos regionais, rotulagem de componentes conectados, transformada de distância e watershed [3, 4]. Elas são usadas em diversos domínios, incluindo a análise de imagens de patologia digital, que são nosso foco. O alto custo de processamento dessas operações tem historicamente limitado avanços no uso de WSIs. Acelerar essas operações é crucial para estudos de câncer, a fim de permitir a análise rápida de grandes conjuntos de dados [5, 1, 6, 7] com objetivo, por exemplo, de identificar padrões espaciais complexos celulares e sua correlação com sobrevida ou resposta a tratamento de pacientes [8].

2 Trabalhos Relacionados

A implementação de reconstrução morfológica é feita por meio de um operador pixel a vizinhança, aplicado à imagem inteira. Uma característica desse operador é a sua convergência, sendo que eventualmente a imagem final converge para estabilidade. Esse operador atualiza um dado pixel baseado em uma condição de propagação, avaliando a sua vizinhança com as duas imagens de entrada (máscara e marcador). Como o resultado da condição de propagação de cada pixel é imprevisível, observa-se uma irregularidade nos padrões de computação e de acesso à memória, tornando sua execução eficiente em

ambientes paralelos desafiadora.

Uma das implementações originais de reconstrução morfológica foi por meio de passadas raster/anti-raster na imagem, permitindo um padrão de acesso de dados com boa localidade [3]. Porém, essa estratégia resulta em uma alta dependência entre pixels, tornando sua paralelização complexa. Uma alternativa à esse padrão de processamento é o *Parallel Sequential Reconstruction* (PSR), onde são realizada quatro passadas (uma em cada direção da imagem) propagando em apenas uma direção, com dependência apenas do pixel anterior [9]. Embora paralelizável, essa abordagem resulta em um maior custo total de processamento e um padrão de acesso aos pixels com menor localidade. Uma forma de mitigar o custo de processamento quando poucos pixels estão ativos foi proposta no algoritmo *Fast Hybrid Reconstruction* (FH) que realiza as passadas raster/anti-raster inicialmente e, em seguida, trabalha com uma fila de pixels ativos. Permitindo à reconstrução morfológica convergir rapidamente.

O algoritmo FH foi implementado para GPUs em [10], sendo identificado esse padrão de computação de propagação de onda, chamado *Irregular Wavefront Propagation Pattern* (IWPP) que pode ser utilizado em várias operações morfológicas. Para GPUs, a implementação de IWPP por meio de FH apresentou alguns desafios. Essas operações possuem um padrão de acesso irregular à memória que não se alinham naturalmente às arquiteturas GPU, que favorecem o processamento regular. Isso resulta em acessos à memória não coalescidos e alta divergência na execução de *threads*, que degradam o desempenho. Adicionalmente, o uso de uma fila global de pixels resulta em contenção da mesma, um problema que se exacerba com a crescente capacidade computacional de GPUs modernas.

Tendo em vista esses gargalos de execução, [11] expandiu a literatura utilizando uma abordagem baseada em blocos, chamados de Megapixels (MPs). Ao invés de trabalhar com pixels ativos, a imagem é particionada em blocos (MPs) quadrados. MPs são considerados ativos, sendo enfileirados e processados independentemente em paralelo, enquanto trocam informação através das bordas sobrepostas com MPs vizinhos chamadas de *GhostZones*(GZs) no início e fim de cada processamento. Como as tarefas de processamento são tarefas mais caras computacionalmente em comparação com a estratégia original que rastreava e processava pixels [10], o *overhead* amortizado associado com o gerenciamento de tarefas é menor, visto que apenas pixels ou MPs que estão ativamente contribuindo para o resultado precisam ser processados e o tamanho máximo da fila também é significativamente menor, com menos operações de inserção e remoção. O uso de MPs melhora a localidade do algoritmo, também diminuindo a divergência de *threads*. Essas características são essenciais para execução eficiente em GPUs.

3 Trabalho Realizado

Embora a abordagem baseada e MPs tenha alcançado ganhos significativos em relação aos trabalhos anteriores [11], ainda existem oportunidades para melhorias. Primeiramente, ao se propagar valores em um MP, múltiplas threads (bloco de threads) são usadas cooperativamente. Elas executam iterativamente em quatro passadas no MP, da esquerda para direita, de cima para baixo, da direita para esquerda e, finalmente, de baixo para cima. Essas passadas são executadas até que não exista mais nenhuma modificação no MP, o que pode levar várias iterações. A proposta principal do trabalho foi criar, implementar e testar um novo padrão de execução mais eficiente, baseado em uma propagação diagonal, que consiste de duas passadas usando as threads da GPU em paralelo. A ideia desse padrão é permitir que os valores se propaguem mais rapidamente entre os pixels dentro do bloco e, assim, diminuir a quantidade de passadas necessárias para convergência.

A Figura 1a apresenta um MP de $3x3$, utilizando a reconstrução morfológica como exemplo onde, por simplicidade, o pixel de maior valor propaga para os seus vizinhos. Se executado em paralelo por três threads, por exemplo, as threads leriam os valores dos pixels da primeira coluna em paralelo, computariam propagação para vizinho (e.g., da esquerda para direita). Na propagação de referência [11] a passada da esquerda para a direita resultaria em 1a já que cada pixel depende apenas do pixel anterior na mesma linha. A propagação proposta percorre o bloco diagonalmente aumentando a mobilidade dos pixels em uma mesma passada. Por exemplo o maior valor em 1b que começa na posição (0,0) em uma mesma passada se propaga para todo o bloco.

(a) MP original. Propagação por passadas paralelas.

(b) Propagação proposta. Frente de onda diagonal.

Figura 1 – A figura apresenta o MP original de acordo com o estado-da-arte e a execução paralela, o resultado da propagação de referência e o processo de propagação proposto. Neste exemplo utilizamos o operador de reconstrução morfológica em que o pixel de maior valor se propaga para os vizinhos.

No entanto, a implementação eficiente dessa propagação apresenta diversos desafios. Primeiramente, como podemos ver na imagem 1b, o tamanho da frente de onda dentro do bloco varia entre as iterações. Essa variação não é ideal para a execução de blocos independentes, nos quais o ideal é manter a utilização máxima das threads a todo tempo de maneira independente. Além disso, o padrão de acesso à memória dentro do bloco não é eficiente.

Para que a execução mantenha a utilização máxima das threads da GPU a proposta modificará o formato do bloco de retângulo regular para paralelogramo. Essa abordagem de particionamento de problema e padrão de computação já foi utilizada em GPUs para acelerar a comparação de sequências genômicas [12].

Com essa nova organização de bloco serão realizadas duas passadas por iteração, uma raster e uma anti-raster. Continuamos podendo processar cada linha em paralelo, mas, para manter a coerência, cada coluna deve terminar seu processamento antes da próxima coluna começar sua execução. Com isso esperamos melhorar o desempenho do atual estado-da-arte em [11]. No entanto, apesar dos potenciais ganhos, essa proposta traz diversos desafios quanto a sua implementação. Modificar o formato da divisão da imagem mantendo o acesso eficiente à memória pelas threads, bem como manter alta ocupação e baixa divergência para GPUs não é uma tarefa trivial.

4 Implementação

Para implementar a nova propagação é necessário alterar o formato do MP de quadrado para paralelogramo. Com o bloco nesse formato podemos concatenar as propagações para diminuir o número de passadas. A propagação de cima para baixo e da esquerda para a direita podem ser concatenadas, bem como a propagação de baixo para cima e da direita para a esquerda. A abordagem ingênua para implementar essa solução é alocar um MP com o dobro de largura e movimentar os pixels dentro do MP de forma que eles estejam diagonalemente colocados como exemplificado na figura 2b.

(a) MP original (b) MP modificado utilizando padding

Figura 2 – A figura apresenta uma implementação possível do bloco proposto e exemplifica a segunda iteração de uma propagação da esquerda para a direita

No exemplo mostrado na figura 2, temos a propagação da esquerda para direita como implementado no estado-da-arte em [11] e a propagação da esquerda para direita proposta utilizando o padding para a transformação do formato do bloco. As threads executam cada linha em paralelo de forma independente. No estado-da-arte, o pixel só é propagado lateralmente, já que cada thread verifica apenas o pixel à esquerda daquele que está processando no momento. Na implementação proposta, cada pixel verifica dois vizinhos, aumentando a mobilidade dos pixels dentro do bloco em uma mesma propagação. Isso é possível pois, ao alterar o formato do bloco e garantir que a coluna anterior tenha sido

completamente executada, temos a certeza que os dois vizinhos relevantes para o pixel já estão com os valores atualizados, o que mantém a coerência com a propagação anterior. Quando a nova frente for executada na propagação proposta, ambos os pixels que são pré-requisitos já terão seus valores atualizados possibilitando a propagação correta. Por exemplo, o pixel na posição (1,1) na figura 2a para a propagação da esquerda para a direita depende do pixel na posição (1,0) e para a propagação de cima para baixo depende do pixel em (0,1). Na figura 2b esse pixel ocupa a posição (1,2) e os pixels dos quais ele depende para a propagação da esquerda para a direita e de cima para baixo ocupam as posições (1,1) e (0,1) respectivamente. Mesmo nessa implementação simples podemos perceber a vantagem dessa propagação proposta ao analisarmos a mobilidade do pixel de valor 2. Enquanto no estado-da-arte o pixel se propagou apenas lateralmente, na propagação proposta, durante uma só passada, ele se propagou tanto lateralmente quanto verticamente.

Essa implementação com padding, mesmo que diminuindo o número de passadas, acaba se tornando ineficiente pelo maior espaço de memória gasto para cada um dos MPs. Diante disso foi implementado uma estratégia de preprocessamento da imagem. Essa estratégia consiste em rotacionar os pixels da imagem de forma que ao dividir o grid de megapixels, cada MP esteja com os pixels na posição correta para a execução da propagação proposta. Cada linha n da imagem nessa estratégia é considerada um array circular, os pixels são movidos n posições para frente no array como mostrado em 3b. A vantagem dessa abordagem é utilizar o mesmo espaço da abordagem do estado-da-arte em [11].

(a) Imagem original

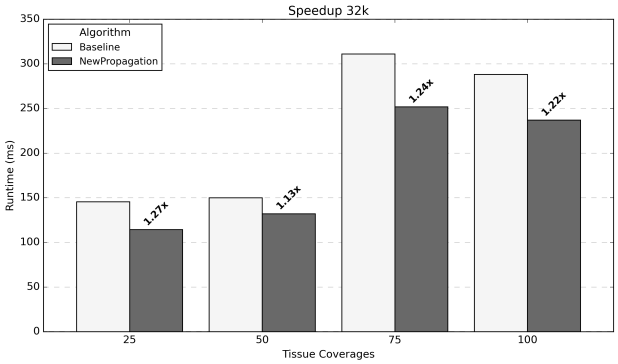
(b) Imagem modificada pelo pré-processamento

Figura 3 – A figura apresenta o pré-processamento aplicado a uma imagem 4x4 pixels dividida em 4 megapixels 2x2.

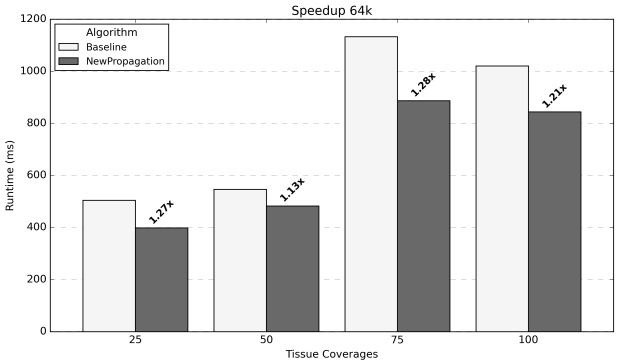
Entretanto, ao utilizar a imagem pré-processada, criamos outra condição de borda que deve ser tratada para manter a corretude das respostas. No grid de megapixels, aqueles pertencentes à diagonal principal fazem a ligação entre os pixels do início da imagem original e os pixels do final. Dessa forma, é necessário impedir a propagação de pixels de um lado para o outro dentro dos megapixels da diagonal principal. Isso foi feito criando-se uma propagação especial para esses megapixels de forma a evitar o vazamento.

5 Resultados

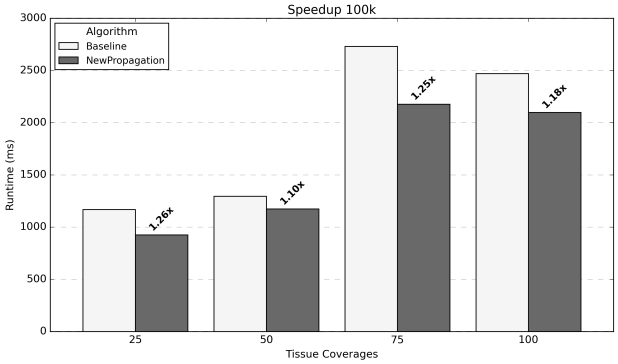
Os testes foram feitos no ambiente de cluster do laboratório SPEED. A máquina utilizada possuía uma placa de vídeo NVIDIA GeForce RTX 3090 Ti, que possui 24GB de memória e 10.752 núcleos CUDA. Os testes foram feitos comparando a versão proposta com a versão do estado-da-arte definida em [11]. Utilizamos imagens de lâminas histológicas inteiras *Whole Slide Imaging*(WSIs) de diferentes tamanhos e com diferentes percentuais de cobertura, de forma parecida ao que foi realizado em [11].



(a) Tamanho 32k



(b) Tamanho 64k

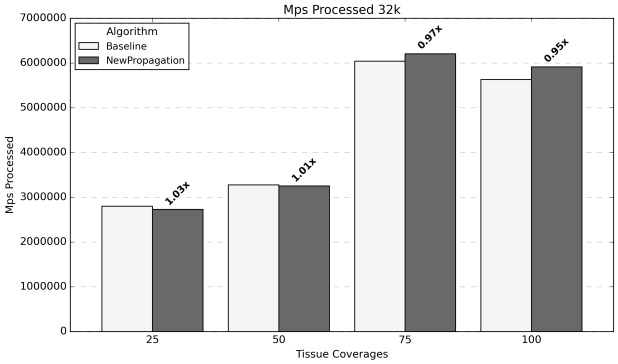


(c) Tamanho 100k

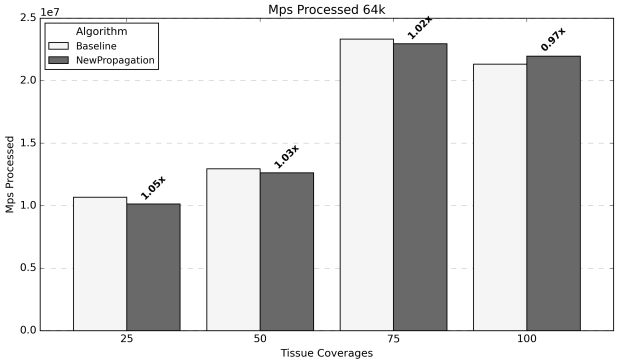
Figura 4 – Gráficos comparativos do speedup entre a versão proposta e o baseline

Como podemos perceber pelos gráficos na figura 4, a implementação proposta superou o baseline em todos os tamanhos de imagem testados, bem como em todos os percentuais

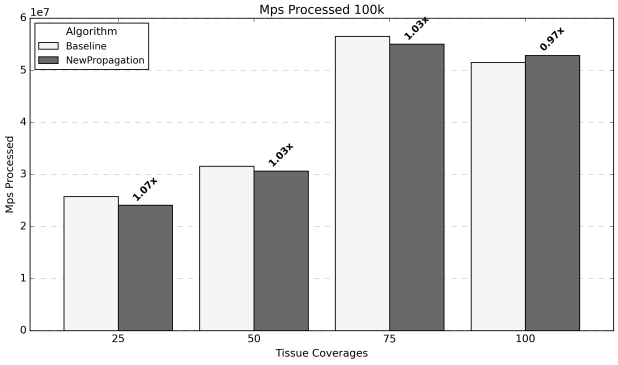
de cobertura testados. Para as imagens de tamanho 32k 4a, o menor speedup foi **1.13x** enquanto o maior foi de **1.27x**. Para as imagens de tamanho 64k 4b, o menor speedup foi **1.13x** enquanto o maior foi de **1.28x**. Para as imagens de tamanho 100k 4c, o menor speedup foi **1.10x**, enquanto o maior foi de **1.26x**. Os resultados de speedup foram bem consistentes entre as imagens, indicando pouca variação entre o número de propagações que foram diminuidas em cada caso.



(a) Tamanho 32k



(b) Tamanho 64k



(c) Tamanho 100k

Figura 5 – Gráficos comparativos do número de megapixels processados entre a versão proposta e o baseline

Outro teste realizado foi a comparação entre o número de megapixels que foram processados em cada uma das versões. Como a implementação proposta altera somente

o número de passadas para a realização da propagação, era esperado que o número de megapixels permanecesse o mesmo. Caso o número de megapixels diminuisse, o speedup poderia ser decorrência disso e não da propagação modificada. Se a propagação modificada aumentasse o número de megapixels processados, deveríamos avaliar se o tradeoff valeria a pena. Nesse caso, como mosta a figura 5, o número de megapixels permaneceu quase o mesmo em todos os tamanhos de imagem e em todos os percentuais de cobertura.

Como a propagação proposta altera o número de passadas para a metade da original, esperavamos um ganho teórico máximo de **2x**. No entanto, a implementação não alcançou esses resultados. Foi feito um trabalho de *profiling* na aplicação usando ferramentas da NVIDIA como *Nsight Compute*. Que indicou um problema de acesso concorrente na memória compartilhada. Cada thread no novo padrão de propagação, ao executar um pixel, terá que compará-lo a dois pixels vizinhos, no exemplo da propagação diagonal de cima para baixo com o vizinho superior e da esquerda, e esses pixels estão armazenados em posições diferentes de bancos diferentes na memória compartilhada da GPU. Isso faz com que a GPU serialize o acesso das threads a esses dados,já que cada banco só pode responder em um ciclo a uma leitura. Esse problema é comum em aplicações para GPU e em outros contextos, como na multiplicação de matrizes, pode ser resolvido com a computação através da transposta. Acreditamos que solucionar esse problema possa deixar a solução mais próxima do máximo teórico.

6 Conclusão

Nesse trabalho apresentamos uma nova proposta de propagação de pixels para executar o Padrão de Frente de Onda Irregular em GPUs aumentando a eficiência da abordagem de megapixels proposta por [11]. A proposta se baseia em modificar a posição relativa entre os pixels na imagem de forma a possibilitar utilizar apenas duas passadas para a propagação. O trabalho mostra que, embora a abordagem baseada e MPs tenha alcançado ganhos significativos em relação aos trabalhos anteriores [11], ainda existem oportunidades para melhorias. Esse trabalho demandou o entendimento profundo da aplicação e do método de propagação, bem como da arquitetura interna da GPU e seus padrões de programação.

Para trabalhos futuros pretendemos explorar a modificação dinâmica do tamanho dos megapixels, que na implementação atual é estático, baseado nas características da imagem. Isso permitiria um uso mais eficiente dos recursos da GPU, alocando megapixels menores em regiões altamente detalhadas ou quando a carga é baixa, e maiores em áreas homogêneas, reduzindo computações redundantes.

Referências

[1] LI, X.; LI, C.; AL. et. A comprehensive review of computer-aided whole-slide image analysis: from datasets to feature extraction, segmentation, classification and detection approaches. *Artificial Intelligence Review*, Springer, v. 55, n. 6, p. 4809–4878, 2022.

[2] The Cancer Genome Atlas Research Network. *The Cancer Genome Atlas Program (TCGA)*. 2006–2018. National Cancer Institute. Disponível em: https://www.cancer.gov/ccg/research/genome-sequencing/tcga. Disponível em: <https://www.cancer.gov/ccg/research/genome-sequencing/tcga>.

[3] SOILLE, P. *Morphological Image Analysis: Principles and Applications*. 2. ed. Berlin Heidelberg New York: Springer-Verlag, 2003.

[4] HŸTCH, M.; HAWKES, P. W. *Morphological image operators*. [S.l.]: Academic Press, 2020. v. 216.

[5] MARDANI, K.; MAGHOOLI, K. Enhancing retinal blood vessel segmentation in medical images using combined segmentation modes extracted by DBSCAN and morphological reconstruction. *Biomedical Signal Processing and Control*, Elsevier, v. 69, p. 102837, 2021.

[6] MCGENITY, C. et al. Artificial intelligence in digital pathology: a systematic review and meta-analysis of diagnostic test accuracy. *npj Digital Medicine*, v. 7, n. 1, p. 114, May 2024. ISSN 2398-6352. Disponível em: <https://doi.org/10.1038/s41746-024-01106-8>.

[7] TAUQEER, A.; ASIF, A.; SADEGHI-NAINI, A. Detection, localization, and staging of breast cancer lymph node metastasis in digital pathology whole slide images using selective neighborhood attention-based deep learning. *Scientific Reports*, v. 15, n. 1, p. 37847, Oct 2025. ISSN 2045-2322. Disponível em: <https://doi.org/10.1038/s41598-025-21787-9>.

[8] BRUMMEL, K. et al. Tumour-infiltrating lymphocytes: from prognosis to treatment selection. *British Journal of Cancer*, v. 128, n. 3, p. 451–458, Feb 2023. ISSN 1532-1827. Disponível em: <https://doi.org/10.1038/s41416-022-02119-4>.

[9] KARAS, P. Efficient computation of morphological greyscale reconstruction. In: SCHLOSS DAGSTUHL–LEIBNIZ-ZENTRUM FÜR INFORMATIK. *Sixth Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS'10)–Selected Papers (2011)*. [S.l.], 2011. p. 54–61.

[10] TEODORO, G. et al. Efficient irregular wavefront propagation algorithms on hybrid CPU–GPU machines. *Parallel Computing*, v. 39, n. 4–5, p. 189–211, 2013. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167819113000343>.

[11] OLIVEIRA, M. et al. The Megapixel Approach for Efficient Execution of Irregular Wavefront Algorithms on GPUs. *IEEE TPDS*, 2025.

[12] SANDES, E. F. O.; MELO, A. C. M. de. CUDAlign: using GPU to accelerate the comparison of megabase genomic sequences. p. 137–146, 2010.

[13] KONG, J. et al. Machine-based morphologic analysis of glioblastoma using whole-slide pathology images uncovers clinically relevant molecular correlates. *PloS one*, Public Library of Science San Francisco, USA, v. 8, n. 11, p. e81049, 2013.

[14] MEIRELLES, A. L. et al. Effective and efficient active learning for deep learning-based tissue image analysis. *Bioinformatics*, Oxford University Press, v. 39, n. 4, p. btad138, 2023.

[15] ROUT, R. et al. Skin lesion extraction using multiscale morphological local variance reconstruction based watershed transform and fast fuzzy c-means clustering. *Symmetry*, MDPI, v. 13, n. 11, p. 2085, 2021.