FILTRO GAUSSIANO ADAPTATIVO IMPLEMENTADO EM AMBIENTE JULIA

B. M. Matosak[1], N. G. Medeiros[1], F. A. F. Rodrigues[1]

[1]Universidade Federal de Viçosa

Sensoriamento Remoto, Fotogrametria e Interpretação De Imagens

RESUMO

Este trabalho teve como finalidade a geração de uma função de filtragem gaussiana adaptativa no ambiente de programação Julia, assim como outras funções necessárias à comparação desta com o filtro gaussiano convencional – como o método de detecção de bordas de Sobel, Gradiente Morfológico, e Método de Otsu para a etapa de limiarização. A estratégia de implementação escolhida foi adaptar a função da filtragem gaussiana convencional a partir da utilização do inverso do desvio padrão da área sobreposta pelo operador do filtro (máscara), adotando-o como o valor de sigma para definir uma nova máscara para cada pixel filtrado. Com o emprego da metodologia adotada foi possível verificar que a função de filtragem gaussiana adaptativa apresentou melhores resultados que a filtragem gaussiana convencional, os quais são fundamentais para o processo subsequente de extração de feições em uma imagem. Com os resultados obtidos pode-se verificar também que o ambiente de programação Julia já possui a base necessária à criação de ferramentas inerentes ao processamento digital de imagens orbitais.

Palavras chave: Ambiente Julia, Filtro Adaptativo, Imagens Orbitais

ABSTRACT

The main goal of this work is the development of a function for the gaussian adaptive filter in the Julia language, as well as the creation of other functions needed for the comparison between the gaussian adaptive filter with the traditional gaussian filter – functions such Sobel's edge detection method, Morphological Gradient, and the Otsu Method for image thresholding. The implementation's strategy used was to adapt the traditional gaussian filter function, using the mathematical inverse of the standard deviation of the area covered by the filter's mask, using it as the sigma value, in order to define a new mask for each filtered pixel. The used methodology implied that the developed gaussian adaptive filter function returns better results than the traditional gaussian filter, which are fundamental for further usage in processes of edges acquisition. The obtained results also show that the Julia language and its programming environment already have the necessary base for the creation of tools commonly used in satellite images digital processing.

Keywords: Julia Language, Adaptive Filter, Satellite Images

1- INTRODUÇÃO

No Processamento Digital de Imagens (PDI) podem ser utilizados algoritmos de realce de imagens, a fim de tornar mais nítidos os objetos da, visando realizar postumamente a extração de feições presentes em cenas capturadas por sensores orbitais. Tais processos visam a eliminação de ruídos e o aumento no contraste dos objetos de interesse em relação ao restante da cena, que são fundamentais para possibilitar a minimização do excesso de feições extraídas, que não necessariamente são úteis ao interesse da aplicação. O processo a ser empregado para a obtenção das feições ótimas aos fins almejados deve ser escolhido pelo usuário dentre as opções disponíveis para que suas necessidades sejam atendidas.

Deve-se ressaltar que algo comum aos processamentos de realce e análise de imagens é a dificuldade dos algoritmos de serem executados de forma rápida. Isto decorre principalmente do fato de tais processos serem baseados em operações algébricas que exigem muito da capacidade computacional da máquina, além do emprego de dados que demandam, normalmente, grande capacidade de armazenamento. Decorrente desta característica dos algoritmos de PDI, torna-se necessária a otimização das funções empregadas, além do uso de ferramentas poderosas para a execução de tais algoritmos.

Uma linguagem de programação que tem chamado a atenção da comunidade acadêmica é a Julia, que em *benchmarks* realizados por Bezanson et al., 2017, mesmo em versão beta, demonstrou superioridade em vários aspectos quando comparada com linguagens consagradas como a MATLAB, R, Python, entre outras. No meio acadêmico, a linguagem Julia surge como uma grande aposta para a melhora no tempo de execução de algoritmos comuns do PDI, porém, por ainda ser recente, há muito a ser desenvolvido neste ambiente.

Com o uso de ferramentas capazes de manipular os dados geográficos presentes em imagens orbitais em maior velocidade, a disponibilização dos mesmos pode ser feita em intervalos de tempo menores, fornecendo a possibilidade do aumento da realização de testes, com intuito de obter informações mais acertadas às tomadas de decisões.

2- REVISÃO BIBLIOGRÁFICA

2.1- Linguagem Julia

Julia é uma linguagem de programação dinâmica de alto nível e de alta performance para computação numérica, gratuita e *open source*. Apesar de ainda estar em sua versão beta (versão 0.6.1 em 08/09/2017), a linguagem já se mostra robusta e algumas vezes até mesmo mais rápida que C ou Python. O sistema Julia disponibiliza um compilador sofisticado, execução paralela distribuída, acurácia numérica, e uma biblioteca extensiva de funções matemáticas (JULIA, 2017).

A linguagem de programação Julia e todo o seu ambiente de desenvolvimento foi visionada para ser uma linguagem completa (BEZANZON et al., 2012):

> "*Nós queremos a velocidade do C com o dinamismo do Ruby. […] Nós queremos algo tão usável quanto o Python para programação geral, tão fácil para estatística quanto o R, […], tão poderoso para álgebra linear como o MATLAB.*"

Testes realizados por BEZANZON et al. (2017) resultaram em um comparativo desta linguagem com seus concorrentes, como mostra a Fig. 1.

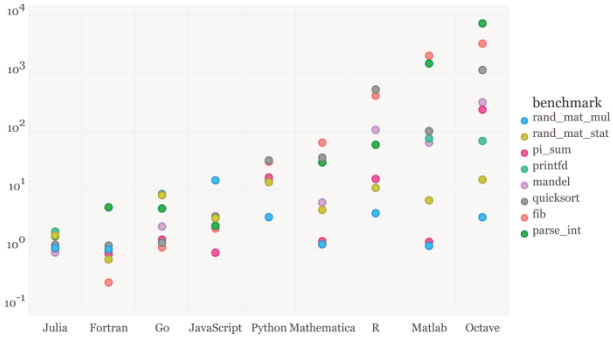


Fig. 1 – Comparação da performance de várias linguagens em *micro-benchmarks* simples. Tempo de execução do *benchmark* relativo a C (Quanto menor melhor; performance em C = 1.0) (Fonte: BEZANZON et al., 2017).

A partir da figura 1, verifica-se que os testes mostram que o ambiente Julia apresentou uma eficiência relativamente superior aos seus concorrentes para computação numérica, comprovando assim todo o potencial que esta ferramenta tem a oferecer no ramo do processamento de dados.

2.2- Filtragem Gaussiana

Uma das atividades mais comuns ao PDI é a aplicação de filtros, para que assim diferentes características da imagem sejam realçadas e outras suprimidas, facilitando a extração de feições ou mesmo a correção de ruídos. Comum em funções de filtragem é a convolução da imagem por uma máscara (também conhecida como operador) de elementos e dimensões pré definidas pelo usuário (GONZALEZ e WOODS, 2010).

A Filtragem Gaussiana é um processo onde a convolução é realizada na imagem visando a suavização da cena, com uma máscara construída de forma condizente com a distribuição normal bidimensional. Nixon e Aguado (2008) destacam que uma das formas de se implementar um filtro gaussiano é definindo a máscara através da equação 1.

$$h(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (1)$$

Onde x e y correspondem à distância do ponto da máscara equivalente ao pixel filtrado até o termo a ser definido, medida em unidades de pixel. Já σ é o desvio padrão a ser utilizado, definindo o abaulamento do "sino" formado pela distribuição.

2.3- Filtragem Adaptativa

A filtragem gaussiana age de forma independente à variação local dos pixeis da imagem. São desconsiderados fatores como, por exemplo, a mudança abrupta dos níveis de cinza, que é característica das bordas. Como dito, tal filtragem visa a suavização da cena, podendo assim ser diminuídos os ruídos indesejáveis contidos na cena, porém, a filtragem gaussiana também resulta na diminuição da precisão das bordas que definem as diferentes feições presentes nas imagens. Neste sentido, encontrar soluções que minimizem perda de resolução das bordas torna-se de grande importância para os processos subsequentes de extração de feições. Uma das soluções pode ser o uso de filtros adaptativos, que são aqueles que se moldam de acordo com características estatísticas da região sobreposta pela máscara de aplicação. Tais filtros demonstram ter um desempenho superior aos filtros convencionais, porém com maior complexidade de implementação e maior necessidade computacional para a execução. Um exemplo de filtro adaptativo é o filtro da mediana, onde o valor do pixel filtrado passa a ser correspondente a mediana para a região sobreposta pela máscara (GONZALEZ e WOODS, 2010).

2.4- Detecção de Bordas

O processo de detecção de bordas engloba atividades que visam a definição dos limites dos objetos presentes na imagem. Dentre os processos empregados se destacam métodos como o método de Sobel, Roberts, Prewitt, Gradiente Morfológico, e etc.

O Método de Sobel para a detecção de bordas pode ser definido usando dois operadores derivativos, descritos na Fig. 2 (GONZALEZ e WOODS, 2010).

-1	-2	-1
0	0	0
1	2	1

(a)

-1	0	1
-2	0	2
-1	0	1

(b)

Fig. 2 – Operador de Sobel em x (S_x) (a) e em y (S_y) (b) (GONZALEZ e WOODS, 2010).

Definindo como * a aplicação de um operador em uma imagem A, a imagem que contém as bordas pode ser definida pelo método de Sobel (S) de acordo com a equação 2.

$$S = \sqrt{(S_x * A)^2 + (S_y * A)^2} \quad (2)$$

Outro método usado em PDI para a detecção de bordas é o Gradiente Morfológico (GM). Tal processo é formado pela combinação de operações morfológicas conhecidas como *Erosão* ($\ominus$) e *Dilatação* ($\oplus$). A obtenção das bordas através deste método é descrita pela equação 3 (MEDEIROS, 2002).

$$GM = (k \oplus A) - (k \ominus A) \quad (3)$$

Onde k é um elemento estruturante de ordem n x n, sendo A a imagem resultante, cujas feições tiveram as bordas detectadas.

2.5- Limiarização (Otsu)

O processo de limiarização consiste em definir um valor radiométrico (nível de cinza), onde os pixeis acima deste valor passarão a possuir o valor máximo possível estabelecido pela resolução radiométrica, e todos os pixeis abaixo ou iguais a este valor passam a possuir o valor mínimo possível. Gonzalez e Woods (2010) definem limiarização de acordo com a equação 4.

$$g(x,y) = \begin{cases} 1 & se\ f(x,y) > T \\ 0 & se\ f(x,y) \leq T \end{cases} \quad (4)$$

Onde T é o valor escolhido para definir em qual nível de cinza será dada a limiarização.

O Método de Otsu para a limiarização pode ser considerado como uma ferramenta para a definição de um nível ótimo para o limiar, visto que não é afetado pela subjetividade do usuário. Tal método pode ser definido da seguinte forma (OTSU, 1979), (GONZALEZ e WOODS, 2010):

1. Calcular o histograma normalizado da imagem de entrada. Designar os componentes do histograma como p_i, i = 0, 1, 2, … , L – 1.
2. Calcular as somas acumuladas, $P_1(k)$, para k = 0, 1, 2, … , L – 1.
3. Calcular as médias acumuladas, $m(k)$, para k = 0, 1, 2, … , L – 1.
4. Calcular a intensidade média global, m_G.
5. Calcular a variância entre classes, $\sigma_B^2(k)$, para k = 0, 1, 2, … , L – 1.
6. Obter o limiar de Otsu, k^*, como o valor de k. Se a máxima não for única, obter k^* pela média dos valores de k que correspondem aos diversos valores máximos detectados.
7. Obter a medida de separabilidade, η^*, avaliando $\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2}$ em $k = k^*$.

3- MATERIAIS E MÉTODOS

A metodologia empregada na execução das atividades deste trabalho visou a criação de um algoritmo no ambiente Julia para a filtragem gaussiana adaptativa em uma imagem orbital, assim como funções necessárias à comparação com a filtragem gaussiana convencional, bem como avaliar o potencial do ambiente Julia quanto ao processamento digital de imagens orbitais. A Tabela 1 apresenta as funções geradas.

TABELA 1 – FUNÇÕES CRIADAS E RESPECTIVA DESCRIÇÃO

Função	Descrição
`filter_gauss()`	Filtragem gaussiana
`filter_gauss_adapt()`	Filtragem gaussiana adaptativa
`sobel_edges()`	Detector de bordas de Sobel
`grad_morf()`	Detector de bordas por gradiente morfológico
`contrast()`	Aplicação de contraste simples
`otsu_thresholding()`	Método de limiarização de Otsu

As funções foram desenvolvidas usando o editor de texto Atom (versão 1.19.5), executadas no ambiente Julia (versão 0.6.0) rodando diretamente na janela de comandos do sistema operacional Ubuntu (versão 16.04).

As cenas da Fig. 5 foram submetidas aos processos de filtragem gaussiana com $\sigma=1$ (`filter_gauss()`) e filtragem gaussiana adaptativa (`filter_gauss_adapt()`), com o tamanho da máscara de aplicação variando de 3 a 11 com passo igual a 2: os resultados destas aplicações são então submetidos aos métodos de detecção de bordas de Sobel e Gradiente Morfológico. Depois desta etapa, ainda é necessário submeter os resultados do gradiente morfológico a uma operação de melhora do contraste que toma o valor máximo do histograma e o interpola para que passe a ser igual ao máximo da resolução radiométrica, para que assim na próxima etapa sejam alcançados melhores resultados. Depois disto, as imagens contendo a informação das bordas são submetidas ao processo de

limiarização de Otsu, para que sejam obtidas as bordas finais a serem comparadas.

Um fluxograma que exemplifica a metodologia adotada se encontra na Fig. 3.

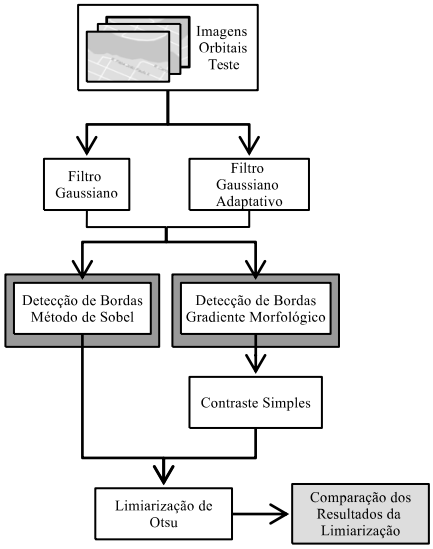


Fig. 3 – Fluxograma que elucida a metodologia adotada na execução das atividades.

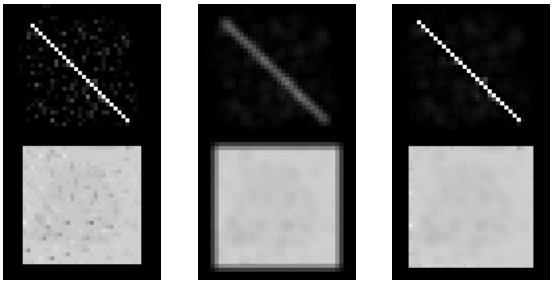
No ambiente Julia podem ser desenvolvidos algoritmos pelos próprios usuários, assim como o uso de funções de pacotes já disponibilizados por terceiros. O pacote *Images.jl* permite a entrada e saída de imagens e o seu processamento, e também contém funções como o detector de bordas de Canny (HOLY, 2017).

Ao carregar as imagens são atribuídos os valores 1.0 para o máximo e 0.0 para o mínimo adotados pela resolução radiométrica (HOLY, 2017), prática comum no processamento digital de imagens, que compatibiliza imagens de diferentes resoluções radiométricas com algoritmos previamente implementados. Desta forma, todas as funções implementadas neste trabalho não dependem do intervalo de níveis de cinza das imagens inseridas para funcionar corretamente.

Em algoritmos convencionais de suavização gaussiana, é criada uma máscara de acordo com o valor para o desvio padrão (σ) inserido, sendo esta usada em todo o processo de convolução, sem levar em consideração se uma região com ruído ou de borda está sendo suavizada. A abordagem tomada para a criação do filtro gaussiano adaptativo foi usar uma modificação da equação 1 de acordo com a equação 4, porém, para cada pixel filtrado, uma máscara é criada para sua suavização, levando em consideração a estatística do seu entorno. O valor de σ a ser adotado nesta adaptação foi o inverso do desvio padrão dos níveis de cinza dos pixeis da área sobreposta pela máscara, multiplicado por uma constante, neste caso adotado após testes empíricos o valor de 0,05 – denominado aqui de σ'. É também realizada a normalização das máscaras, para que o somatório de seus termos seja sempre igual a 1.

$$h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (4)$$

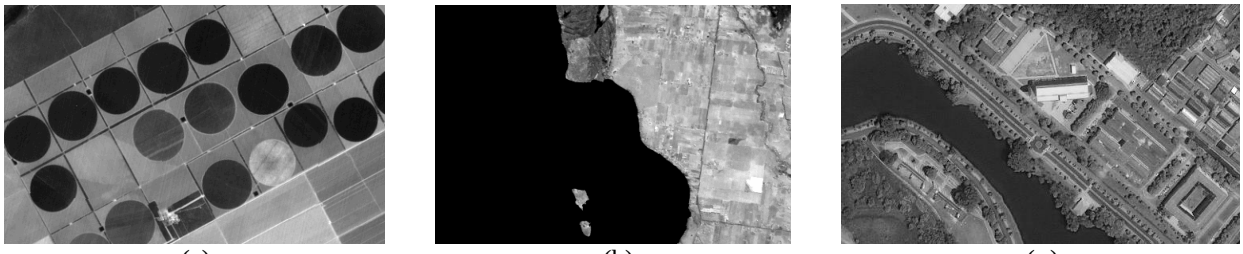
Em regiões de borda, o valor de σ' tende a diminuir devido ao alto desvio padrão entre os pixeis, ocasionando um baixo efeito de borramento. Em regiões sem bordas de feições e com ruído, o desvio padrão tende a ser menor e diferente de zero, o que torna σ' maior, causando o efeito de borramento de forma adaptativa. Uma condição foi criada, para que o valor original do pixel seja retornado, caso o valor do desvio padrão seja nulo. Um exemplo da aplicação do filtro pode ser visto na Fig. 4.



(a) (b) (c)

Fig. 4 – Imagem sintética sem tratamento (40 x 70 pixeis) (a), imagem tratada com `filter_gauss()` máscara 3x3 e σ=1 (b), e imagem tratada com `filter_gauss_adapt()` máscara 3x3 (c).

As imagens testadas representam duas cenas diferentes obtidas por meio do satélite Resourcesat-2 sensor LISS-3. Uma das imagens é de uma região com pivôs centrais no sul da Bahia e outra de um trecho do rio Paraná, com uma resolução espacial de 24 metros (Fig. 5a e 5b). A terceira cena é da Avenida P. H. Rolfs na Universidade Federal de Viçosa obtida por meio de um veículo aéreo não tripulado, cuja resolução espacial é de 1 metro (Fig. 5c).



(a) (b) (c)

Fig. 5 – Imagens usadas para os testes com as funções: Pivôs centrais no sul baiano (400 x 250 pixeis) (a), trecho do Rio Paraná (450 x 300 pixeis) (b), e avenida P. H. Rolfs no Campus Viçosa da UFV (500 x 300 pixeis) (c).

4- RESULTADOS E CONCLUSÕES

Os resultados obtidos gerados a partir das funções criadas descritas pela Tabela 1 foram definidos após a aplicação da detecção de bordas de Sobel e Gradiente Morfológico, assim como limiarizações. Alguns dos resultados são mostrados nas figuras de 6 a 8.

(a) (b)

Fig. 6 – Ampliação do resultado da filtragem gaussiana (a) e filtragem gaussiana adaptativa (b) sobre a imagem bruta da Fig. 5c com máscara 5 x 5.

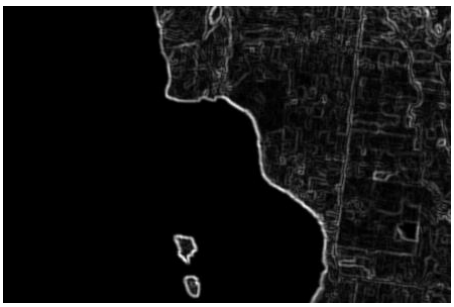
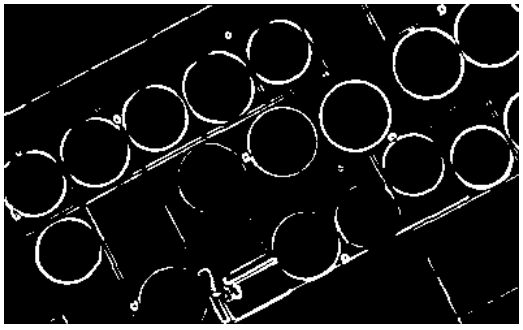


Fig. 7 – Gradiente morfológico da aplicação da função `filter_gauss_adapt()` com uma máscara 5 x 5 sobre a imagem bruta da Fig. 3c.

(a)



(b)

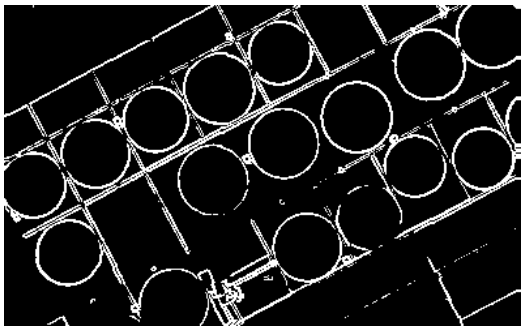


Fig. 8 – Bordas obtidas com o método de Sobel e limiarização de Otsu sobre a imagem da Fig. 5a usando: filtro gaussiano (a) e filtro gaussiano adaptativo (b).

Comparando os resultados apresentados nas figuras 8a e 8b, nota-se que as imagens limiarizadas pelo método de Otsu, processadas a partir do resultado da função gaussiana adaptativa, retornaram melhores resultados para o processo de extração de bordas, com relação ao uso da filtragem gaussiana convencional, visto que mais bordas foram detectadas, considerando um nível de ruído similar ao uso da metodologia convencional.

Outra conclusão percebida a partir dos experimentos é a capacidade do ambiente Julia em processar imagens orbitais, se mostrando suficiente mesmo que ainda em estágio de desenvolvimento (sem versão estável disponível).

AGRADECIMENTOS

Os autores agradecem ao Departamento de Engenharia Civil (DEC) da Universidade Federal de Viçosa (UFV) pela disponibilização da imagem do campus Viçosa (Fig. 5c), ao Instituto Nacional de Pesquisas Espaciais (INPE) pelo fornecimento das imagens Resourcesat-2 (Fig. 5a e 5b), e por fim à Coordenadoria de Educação Aberta e à Distância (CEAD) por recurso financeiro, com a concessão de uma bolsa de iniciação científica.

REFERÊNCIAS

Bezanzon, J.; A. Edelman; S. Karpinski e V. B. Shah, 2017. Julia: A Fresh Approach to Numerical Computing, SIAM Review, Vol. 59, N° 1, pp. 65-98.

Bezanzon, J.; A. Edelman; S. Karpinski e V. B. Shah, 2017. Why we created Julia, Disponível em: https://julialang.org/blog/2012/02/why-we-created-julia, Acesso em: 3 de Setembro de 2017.

Gonzalez, R. C.; R. E. Woods, 2010. Processamento Digital de Imagens 3ª edição, Pearson, São Paulo, Brasil, 624 páginas.

Holy, T., 2017. JuliaImages: image processing and machine vision for Julia, Disponível em: http://juliaimages.github.io, Acesso em 3 de Setembro de 2017.

Julia, The Julia Language, Disponível em: https://Julialang.org/, Acesso em: 3 de Setembro de 2017.

Medeiros, N. G.; E. A. Silva e J. R. Nogueira, 2002. Segmentação Morfológica De Imagens Utilizando O Gradiente Morfológico Multi-Escala, Revista Brasileira de Cartografia, Vol. 54, N° 1, pp.77-85.

Nixon, M.; A. Aguado, 2008. Feature Extraction & Image Processing Second Edition, Elsevier, Oxford, Reino Unido, 423 páginas.

Otsu, N., 1979. A Threshold Selection Method from Gray-Level Histograms, IEEE Transactions On Systems, Man, And Cybernetics, Vol. SMC-9, N° 1, pp. 62-66.