Compreendo os Mecanismos de Processamento de Imagem no Gwyddion por Operações de Matrizes em Python

José David A. Sales, Diogo P. de L. Carvalho, Natália A. de Souza e Kayllany L. da S. Oliveira

Abstract O *Gwyddion* é um *software* de análise e visualização de dados em formato de imagens de microscopia de ponta de prova (SPM), sobretudo de microscopia de força atômica (AFM) e de microscopia de tunelamento com varredura (STM). Esse *software* apresenta diversas ferramentas padrão para o tratamento dos dados, as quais consistem em operações com matrizes, uma vez que imagens são descritas como matrizes computacionalmente. Dessarte, neste trabalho, almeja-se compreender os mecanismos de processamento de imagem por parte das principais funções do *Gwyddion* a partir de operações de matrizes, possibilitando a aplicação de tais funções em *Python*.

Keywords: Microscopia de Sonda (SPM); Processamento de Imagem; *Gwyddion*; *Python*; Operações de Matrizes.

José David Alves Sales
Ilum Escola de Ciência, Centro Nacional de Pesquisa em Energia e Materiais (CNPEM)
e-mail: jose23038@ilum.cnpem.br

Diogo Pereira de Lima Carvalho
Ilum Escola de Ciência, Centro Nacional de Pesquisa em Energia e Materiais (CNPEM)
e-mail: diogo23039@ilum.cnpem.br

Natalia Alcantara de Souza
Ilum Escola de Ciência, Centro Nacional de Pesquisa em Energia e Materiais (CNPEM)
e-mail: natalia23010@ilum.cnpem.br

Kayllany Lara da Silva Oliveira
Ilum Escola de Ciência, Centro Nacional de Pesquisa em Energia e Materiais (CNPEM)
e-mail: kayllany23042@ilum.cnpem.br

1 Introdução

O *Gwyddion* é um *software* de código aberto gratuito de análise e visualização de dados de microscopia, principalmente de microscopia de força atômica (AFM) e microscopia de tunelamento com varredura (STM). O software oferece diversas ferramentas padrão para o tratamento de dados, as quais são fundamentadas em operações com matrizes, dada a representação computacional das imagens.[1] Diante disso, o objetivo central do trabalho é compreender os mecanismos de processamento de imagem realizados pelo *G*wyddion, destacando as principais funções que envolvem operações matriciais. Para isso, a linguagem de programação escolhida para implementar e demonstrar tais operações é o *P*ython.

2 Processos matemáticos

Os processos matemáticos envolvidos no tratamento de imagens de SPM podem ser descritos, majoritariamente, com processos de operações de matrizes. Para além destes, também pode-se citar os processos de regressão linear, cálculo de mediana e normalização de valores. Cada processo de tratamento de imagem explorado nesse trabalho é matematicamente explicado nas três seções a seguir.

2.1 Processo de Planificação da Imagem

A princípio, o mecânismo de planificação de imagem (*Plane Level*) pode ser demonstrado como um processo de regressão linear para determinar uma função $z(i, j)$ que melhor represente a composição da imagem. Pode-se dizer que é impossível representar a complexidade de uma imagem apenas com uma função linear, porém, essa função permite a reprodução do gradiente da imagem, o qual, ao ser subtraído da imagem original, permite que a imagem deixe de apresentar um gradiente.

Os argumentos da função $z(i, j)$ representam as coordenadas da matriz que representa a imagem, assim como é mostrado na equação 1. É necessário ressaltar que essas coordenadas têm inicio no ponto 0 e não 1, diferentemento do que é usualmente mostrado em representações de matrizes genéricas. [5]

$$\text{Gradiente da imagem} = \begin{bmatrix} z(0,0) & \cdots & z(0,j) \\ \vdots & \ddots & \vdots \\ z(i,0) & \cdots & z(i,j) \end{bmatrix} \tag{1}$$

A função $z(i, j)$ é descrita seguinte maneira:

$$z(i, j) = a \cdot i + b \cdot j + c \tag{2}$$

A regressão linear permite que os melhores coeficientes a, b e c para a equação 2 sejam encontrados. É possível encontrar esses coeficientes através de vários métodos de regressão linear. Um dos métodos mais simples é o Coeficiente de Correlação de Pearson, o qual é usado para encontrar os coeficiente a e b nas equações 3 e 4.

$$a = \frac{\sum_{k=1}^{n}(i_k - \bar{i})(z_k - \bar{z})}{\sum_{k=1}^{n}(i_k - \bar{i})^2} \tag{3}$$

$$b = \frac{\sum_{k=1}^{n}(j_k - \bar{j})(z_k - \bar{z})}{\sum_{k=1}^{n}(j_k - \bar{j})^2} \tag{4}$$

$$c = \bar{z} - a\bar{i} - b\bar{j} \tag{5}$$

Resumindo as equações acima, o coeficiente a é calculado somando a diferença entre cada valor individual de i e a média de i ($\bar{i}$), multiplicada pela diferença entre o valor correspondente de z e a média de z ($\bar{z}$). Para o coeficiente a, esse somatório é então dividido pela soma dos quadrados das diferenças entre cada valor individual de i e a média de i. A mesma lógica se aplica para B, mas considerando a variável j, ao invés de i. Todo esse processo descrito nada mais é do que o processo estatístico de carcular a correlação. Para o coeficiente c na equação 5, o cálculo é bem mais simples: basta substrair $\bar{z}$, pela produto a com $\bar{i}$ substraído pela produto de b com $\bar{j}$.[4]

Ao fim, depois que a matriz gradiente da imagem é encontrada com a função $z(i, j)$, como descrito na equação 1, basta subtraí-la da matriz imagem original para conseguir obter uma matriz que representa a imagem planificada.[5]

$$M_{\text{imagem planificada}} = M_{\text{imagem original}} - M_{\text{gradiente da imagem}} \tag{6}$$

2.2 Processo de Filtragem

O mecanismo de filtragem explorado, denominado *Align Arows, Median Method*, consiste na criação de uma matriz plano de medianas, na qual todos os valores na linha i correspondem ao valor mediano dos valores da mesma linha i da matriz original já planificada($Med(M[i])$, assim como é descrito na matriz da equação 7.

$$\text{Plano de medianas} = \begin{bmatrix} Med(M[0])_{0,0} & \cdots & Med(M[0])_{0,j} \\ \vdots & \ddots & \vdots \\ Med(M[i])_{i,0} & \cdots & Med(M[i])_{i,j} \end{bmatrix} \tag{7}$$

Depois que a matriz plano de medianas é cálculada, apenas é necessário substraí-la da matriz planificada para se obter a matriz filtrada. [5]

$$M_{\text{imagem filtrada}} = M_{\text{imagem plafinificada}} - M_{\text{plano de medianass}} \tag{8}$$

2.3 Processo de Adição de Gradiente de Cor

O último processo a ser feito funciona de uma forma relativamente mais simples. Para que ocorra a adição de cor na imagem, é apenas necessário que cada item da matriz seja multiplicado por uma matriz de vetores RGB.

$$\vec{V} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{9}$$

$$\text{Matriz colorida} = \begin{bmatrix} a_{11} \cdot \mathbf{V} & \cdots & a_{1j} \cdot \mathbf{V} \\ \vdots & \ddots & \vdots \\ a_{i1} \cdot \mathbf{V} & \cdots & a_{ij} \cdot \mathbf{V} \end{bmatrix} \tag{10}$$

É interessante ressaltar que isso acaba formando uma matriz de matrizes, que também pode ser interpretado como uma matriz tridimensional. Para além disso, é importante dizer que o produto da matriz com o vetor RGB só pode ocorrer em uma matriz $i \times 3$ e este processo é bem diferente do processo executado na matriz da equação 10. [6]

3 Métodos

A partir dos processos envolvendo operações de matrizes descritas, definiram-se funções em *python* capaz de replicar os processos matemáticos das ferramentas *software Gwyddion* em um *Jupyter Notebook* com uma aplicação em um imagem de exemplo ilustrado mais posteriormente neste documento. As funções serão brevemente explicados neste documento, mas para melhores informações acerca deles, pode-se acessar o arquivo documentado no repositório do GitHub na referência [3].

No Notebook, a função ”desentortar(imagem_qualquer)” é responsável pelo processo de planificação da imagem. Essa função recebe uma imagem em matriz bidimensional no formato de ”numpy.array” como argumento, com o qual realiza uma regressão linear a partir do método dos mínimos quadrados sobre a matriz para calcular o plano de fundo. Assim, a função retorna duas matrizes também no formato de ”numpy.array”, o ”plano_de_fundo” pelo plano calculado e a ”imagem_nivelada” pela subtração da matriz de entrada pelo plano de fundo.

A função ”filtro_mediana(imagem_qualquer)” é responsável pelo processo de filtragem pelo método de medianas. Essa função recebe uma imagem em matriz bidimensional no formato de ”numpy.array” como argumento, com o qual determina a mediana de cada linha e cria uma matriz de mesmo tamanho, porém com os elementos de cada linha possuindo o valor da mediana da linha determinada pela matriz original. A função então retorna duas matrizes, o ”plano_de_medianas” criada a partir dos cálculos das medianas das linhas e a ”imagem_filtrada” pela subtração da matriz de entrada pelo plano de medianas.

A função ”colorindo(imagem_cinza, matriz_cor)” é responsável pelo processo de adição de gradiente de cor. Para utilizar tal função, é necessário utilizar a função ”normalizar(matriz_qualquer)” anteriomente, o qual recebe uma imagem em matriz bidimensional no formato de ”numpy.array” como argmento e retorna uma imagem de mesmo formato com valores normalizados entre 0 e 255, conforme a escala RGB. Assim, a função ”colorindo(imagem_cinza, matriz_cor)” recebe dois argumentos, uma imagem normalizada na escala RGB em matriz bidimensional no formato de ”numpy.array” e uma matriz linha de cor com três valores no formato de ”numpy.array”. A matriz de cor é normalizada entre valores de 0 e 1 e, posteriomente, cada elemento da matriz de imagem de entrada é multiplicado à matriz de cor, retornando uma matriz ”matriz_colorida” como a matriz de entrada com a aplicação do gradiente de cor.

4 Resultados e Discussão

Através do conhecimento das diversas operações com matrizes, explicadas nos tópicos 2 e 3 do documento, foi possível elaborar um programa capaz de replicar as operações do *Gwyddion* e gerar um bom tratamento de imagem, proporcionanado uma imagem menos ruidosa e com melhor contraste, facilitando assim sua análise. A figura 1 abaixo mostra as imagens obtidas por meio de cada processo anteriormente discutido.

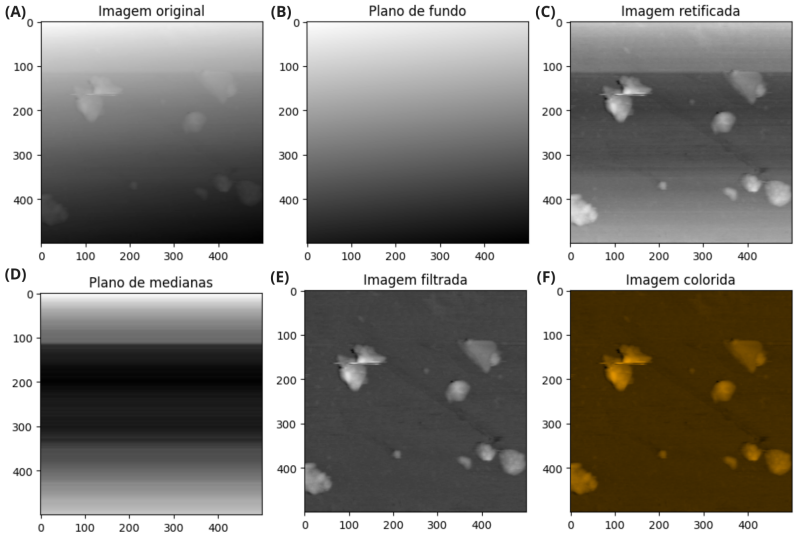

Fig. 1 (A) Imagem obtida por AFM. (B) Plano de fundo obtido por método de mínimos quadrados. (C) Imagem após nivelamento. (D) Plano de medianas obtido pela mediana das linhas. (E) Imagem após a aplicação do filtro de mediana. (F) Imagem filtrada colorida.

O programa desenvolvido para replicar os processos do *Gwyddion* e e gerar as imagens da figura 1 pode ser acessado através do nosso repositório no site Github, disponível no site da referência [3]. Vale ressaltar que na função de planificação, o uso de uma pronta para cálculo de coeficientes por regressão linear foi feito, mas é compreende-se a possibilidade de encontrar os coeficientes pelo método matemático citado na subseção 2.1.

5 Sobre as Referências Bibliográficas

A referência [1] apenas foi necessária para dar descrição formal da microscopia de varredura por ponta de prova. A fonte [2] foi de grande utilidade para compreender as representações de imagens como matrizes e os processos que ocorrem com elas em sua exibição e tratamento. A fonte [3] é apenas o repositório criado por nós para que pudéssemos compartilhar um material extra ao leitor e também para nos autorreferenciar. O artigo da fonte [4] explica bastante sobre as etapas e os processo de regressão linear em imagens. A referência [5] é um guia completo de usuário do *gwyddion*, nele foi possível compreender a fundo sobre os processos que envolvem cada ferramenta do *software*. A fonte [6] explora bastante o processamento de imagem com *python* e suas bibliotecas, esta foi de grande utilizadade para elaboração do nossa programa.

6 Considerações Finais

Considerando todo o trabalho desenvolvido e a pesquisa realizada para compreensão do *software* de tratamento de imagem, é possível concluir que alcançamos completamente nossos objetivos. Conseguimos replicar perfeitamente as principais funções do *Gwyddion* através dos vários processos algébricos aprendidos durante a execução do trabalho. O tema desenvolvido também foi de grande utilidade à medida que nos fez resgatar uma série de conteúdos relacionados com a computação e com a estatística, como a uso de funções do módulo *Linalg* da biblioteca *Numpy* e o cálculo da covariância. Para além disso, o trabalho feito mostra como a compreensão dos processos matemáticos por trás das operações computacionais, juntamente com o conhecimento de sintaxes de uma linguagem de programação, possibilitam a reprodutibilidade de métodos computacionais em qualquer linguagem.

7 Contribuições dos autores

José David Alves Sales: Investigação, Processamento em Python, Redação - Processos Matemáticos, Resultados e Discussão, Considerações Finais e Revisão.
Diogo Pereira de Lima Carvalho: Investigação, Processamento em Python, Redação - Resumo e Métodos, Análise formal e Revisão
Natalia Alcantara de Souza: Investigação, Análise Formal, Organização, Redação - Introdução e Revisão.
Kayllany Lara da Silva Oliveira: Investigação, Redação - Resultados e Discussão e Revisão.

Agradecimentos

Gostariamos de agradecer ao técnico de laboratório Alessadro Mourato de Souza, da Ilum - Escola de Ciência, pela ajuda e pelas explicações sobre a necessidade do tratamento de imagens em função dos próprios mecanismo de funcionamento do AFM educacional.

References

1. Bruker. *What is an SPM?*. Year of Access 2023. Disponível em: https://www.bruker.com/en/products-and-solutions/microscopes/materials-afm/what-is-an-spm.html
2. CORRÊA, J. et al. MICROSCOPIA CONFOCAL BÁSICA. Universidade de Brasília Microscopia Confocal Básica. 2012.
3. José David. Repositório *do_gwyddion_ao_python*. GitHub, 2023. Disponível em: https://github.com/joseDavid23038/do_gwyddion_ao_python.git.
4. HE, Kaiming; SUN, Jian; TANG, Xiaoou. Guided image filtering. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 35, n. 6, p. 1397-1409, 2012.
5. KLAPETEK, Petr; NECAS, David; ANDERSON, Christopher. Gwyddion user guide. Czech Metrology Institute, v. 2007, p. 2009, 2004.
6. NAG, H. Applications of Linear Algebra in Image Filters [Part I] - Operations, 2020.