

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

FACULDADE DE CIÊNCIAS - CAMPUS BAURU

DEPARTAMENTO DE COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LEONARDO KOOSUKE AZUMA

REDUÇÃO DE RUÍDO EM IMAGENS VIA GEOMETRIA FRACTAL

BAURU

Novembro/2025

LEONARDO KOOSUKE AZUMA

REDUÇÃO DE RUÍDO EM IMAGENS VIA GEOMETRIA FRACTAL

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru.
Orientador: Profa. Dra. Tatiana Miguel Rodrigues

BAURU
Novembro/2025

A997r Azuma, Leonardo Koosuke
Redução de ruído em imagens via Geometria Fractal /
Leonardo Koosuke Azuma. -- Bauru, 2025
68 p. : tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Ciência da
Computação) - Universidade Estadual Paulista (UNESP),
Faculdade de Ciências, Bauru
Orientadora: Tatiana Miguel Rodrigues

1. Redução de ruído. 2. Geometria Fractal. 3. IFS. I. Título.

Leonardo Koosuke Azuma

Redução de ruído em imagens via Geometria Fractal

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Profa. Dra. Tatiana Miguel Rodrigues

Orientador

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Ciência da Computação

Profa. Dra. Simone das Graças Domingues Prado

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Ciência da Computação

Prof. Dr. Antonio Carlos Sementille

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Ciência da Computação

Bauru, _____ de _____ de _____.

Resumo

O objetivo deste estudo é realizar a redução de ruído em imagens via Geometria Fractal através de uma modificação na etapa de identificação dos blocos mais semelhantes em uma imagem, substituindo a métrica original, o Erro Médio Quadrático (*Mean Squared Error* - MSE), pelo Índice de Similaridade Estrutural (*Structural Similarity Index Measure* - SSIM), e comparar os resultados obtidos com o algoritmo original e com o método *Block Matching and 3D Filtering* (BM3D). Para isso, foram realizados experimentos com diferentes tipos e intensidades de ruído aplicados a diversas imagens, e para quantificar a qualidade das imagens restauradas foram utilizadas as métricas MSE, Razão do Sinal-Ruído de Pico (*Peak Noise to Signal Ratio* - PSNR) e SSIM. Os resultados demonstram que a substituição do MSE pelo SSIM não melhora o desempenho do algoritmo fractal, apresentando, em alguns casos, leve perda de qualidade na restauração das imagens. O algoritmo BM3D apresentou desempenho superior em quase todos os testes, confirmando sua robustez e eficiência. No entanto, conclui-se que o método fractal ainda possui potencial, especialmente se forem exploradas formas mais eficazes de obtenção do Sistema de Funções Iteradas (IFS).

Palavras-chave: Redução de ruído, Geometria Fractal, IFS.

Abstract

The goal of this study is to perform image denoising using Fractal Geometry by modifying the step of identifying the most similar blocks in an image, replacing the original metric, the Mean Squared Error (MSE), with the Structural Similarity Index (SSIM), and comparing the results obtained with those of the original algorithm and the Block-Matching and 3D Filtering (BM3D) method. Experiments were conducted with different types and intensities of noise applied to various images, and the quality of the restored images was evaluated using the MSE, PSNR, and SSIM metrics. The results indicate that replacing MSE with SSIM does not improve the performance of the fractal algorithm and, in some cases, results in a slight loss of restoration quality. The BM3D algorithm achieved superior performance in almost all tests, confirming its robustness and efficiency. However, it is concluded that the fractal method still holds potential for improvement, especially if more effective approaches to obtaining the Iterated Function System (IFS) are explored.

Keywords: Denoising, Fractal Geometry, IFS.

Lista de figuras

Figura 1 – Representação dos pontos de uma sequência de Cauchy.	17
Figura 2 – Exemplo da copiadora.	18
Figura 3 – Iterações da copiadora.	19
Figura 4 – Exemplo da distância de um ponto à um conjunto.	20
Figura 5 – Exemplo da distância entre dois conjuntos.	20
Figura 6 – Exemplo do teorema do mapeamento contrativo para o exemplo 2.3.	23
Figura 7 – Triângulo de Sierpiński começando um quadrado	25
Figura 8 – Triângulo de Sierpiński começando um círculo	25
Figura 9 – Triângulo de Sierpiński gerado pelo algoritmo randômico com número de iterações crescente	27
Figura 10 – Densidade do fractal Samambaia de Barnsley gerado pelo algoritmo randômico	27
Figura 11 – Exemplo ruído Gaussiano	29
Figura 12 – Histograma da imagem com ruído Gaussiano	29
Figura 13 – Imagem corrompida pelo sal e pimenta	30
Figura 14 – Histograma da imagem com ruído Sal e Pimenta	30
Figura 15 – Imagem corrompida pelo ruído uniforme.	31
Figura 16 – Histograma da imagem com ruído uniforme	31
Figura 17 – Imagem corrompida pelo ruído Gamma.	32
Figura 18 – Histograma da imagem com ruído Gamma	32
Figura 19 – Imagem corrompida pelo ruído periódico.	33
Figura 20 – Histograma da imagem com ruído periódico	33
Figura 21 – Imagem corrompida pelo ruído <i>speckle</i>	34
Figura 22 – Histograma do ruído <i>speckle</i>	35
Figura 23 – Exemplo das métricas em diferentes intensidades de ruídos	37
Figura 24 – Transformações do bloco de domínio D_i para o bloco de imagem R_i	38
Figura 25 – Iterações da decodificação da Lena	40
Figura 26 – Fluxograma do Planejamento	41
Figura 27 – Resultados das métricas MSE, PSNR e SSIM para ruído Gaussiano com $\sigma = 0.12$	44
Figura 28 – Exemplo entre métodos de remoção de ruído.	45
Figura 29 – Resultados das métricas MSE, PSNR e SSIM para ruído Sal e Pimenta com $p_1 = 0.04$ e $p_2 = 0.04$	46
Figura 30 – Exemplo entre métodos de remoção de ruído.	47
Figura 31 – resultados das métricas MSE, PSNR e SSIM para ruído Uniforme com $a = 0.1$ e $b = 0.1$	48
Figura 32 – Exemplo entre métodos de remoção de ruído.	49

Figura 33 – resultados das métricas MSE, PSNR e SSIM para ruído gama com $a = 3$ e $b = 0.035$.	50
Figura 34 – Exemplo entre métodos de remoção de ruído.	51
Figura 35 – resultados das métricas MSE, PSNR e SSIM para ruído Periódico com $f_1(x, y)$.	52
Figura 36 – resultados das métricas MSE, PSNR e SSIM para ruído Periódico com $f_3(x, y)$.	53
Figura 37 – Exemplo entre métodos de remoção de ruído.	53
Figura 38 – resultados das métricas MSE, PSNR e SSIM para ruído uniforme com $a = 50$ e $\alpha = \frac{1}{50}$.	54
Figura 39 – Exemplo entre métodos de remoção de ruído.	55
Figura 40 – Melhor resultado do algoritmo fractal.	58
Figura 41 – Pior resultado do algoritmo fractal.	58
Figura 42 – Resultados das métricas MSE, PSNR e SSIM para ruído Gaussiano com $\sigma = 0.08$.	59
Figura 43 – Resultados das métricas MSE, PSNR e SSIM para ruído Gaussiano com $\sigma = 0.156$.	60
Figura 44 – Resultados das métricas MSE, PSNR e SSIM para ruído Gaussiano com $\sigma = 0.20$.	60
Figura 45 – Resultados das métricas MSE, PSNR e SSIM para ruído Sal e Pimenta com $p_1 = 0.02$ e $p_2 = 0.02$.	61
Figura 46 – Resultados das métricas MSE, PSNR e SSIM para ruído Sal e Pimenta com $p_1 = 0.08$ e $p_2 = 0.08$.	61
Figura 47 – Resultados das métricas MSE, PSNR e SSIM para ruído Sal e Pimenta com $p_1 = 0.16$ e $p_2 = 0.16$.	62
Figura 48 – resultados das métricas MSE, PSNR e SSIM para ruído Uniforme com $a = 0.05$ e $b = 0.05$.	62
Figura 49 – resultados das métricas MSE, PSNR e SSIM para ruído Uniforme com $a = 0.15$ e $b = 0.15$.	63
Figura 50 – resultados das métricas MSE, PSNR e SSIM para ruído Uniforme com $a = 0.2$ e $b = 0.2$.	63
Figura 51 – resultados das métricas MSE, PSNR e SSIM para ruído gama com $a = 2$ e $b = 0.035$.	64
Figura 52 – resultados das métricas MSE, PSNR e SSIM para ruído gama com $a = 3$ e $b = 0.045$.	64
Figura 53 – resultados das métricas MSE, PSNR e SSIM para ruído gama com $a = 4$ e $b = 0.065$.	65
Figura 54 – resultados das métricas MSE, PSNR e SSIM para ruído Periódico com $f_2(x, y)$.	65
Figura 55 – resultados das métricas MSE, PSNR e SSIM para ruído Periódico com $f_4(x, y)$.	66
Figura 56 – resultados das métricas MSE, PSNR e SSIM para ruído uniforme com $a = 100$ e $\alpha = \frac{1}{100}$.	66
Figura 57 – resultados das métricas MSE, PSNR e SSIM para ruído uniforme com $a = 25$ e $\alpha = \frac{1}{25}$.	67

Figura 58 – resultados das métricas MSE, PSNR e SSIM para ruído uniforme com $a = 10$ e $\alpha = \frac{1}{10}$.

Lista de tabelas

Tabela 1 – Parâmetros e probabilidades das transformações do fractal da samambaia de Barnsley.	26
Tabela 2 – Comparação das medias para o ruído Gaussiano.	45
Tabela 3 – Comparação das medias para o ruído Sal e Pimenta.	47
Tabela 4 – Comparação das medias para ruído Uniforme.	49
Tabela 5 – Comparação das medias para ruído Gama.	51
Tabela 6 – Comparação das medias para ruído Periódico.	54
Tabela 7 – Comparação das medias para ruído <i>Speckle</i>	55
Tabela 8 – Tempo de execução dos algoritmos fractais.	56

Lista de abreviaturas e siglas

BM3D	<i>Block-Matching and 3D Filtering</i>
IFS	Sistema de Funções Iteradas
MSE	Erro Médio Quadrático
PSNR	Razão do Sinal-Ruído de Pico
SSIM	Índice de Similaridade Estrutural

Sumário

1	INTRODUÇÃO	13
1.1	Problema	13
1.2	Justificativa	13
1.3	Objetivo	13
1.4	Objetivos específicos	14
2	FUNDAMENTAÇÃO	15
2.1	Geometria Fractal	15
2.1.1	Espaços métricos	15
2.1.2	Fractais	18
2.1.3	Espaço de Hausdorff	19
2.1.4	Mapeamentos Contrativos	21
2.1.5	Dois algoritmos para exibir fractais gerados por IFS	24
2.1.5.1	Algoritmo determinístico	24
2.1.5.2	Algoritmo randômico	25
2.2	Remoção de Ruído em Processamento de Imagens	27
2.2.1	Tipos de ruído	28
2.2.1.1	Ruído Gaussiano	28
2.2.1.2	Ruído Sal e Pimenta	29
2.2.1.3	Ruído Uniforme	30
2.2.1.4	Ruído Gamma	32
2.2.1.5	Ruído Periódico	33
2.2.1.6	Ruído <i>speckle</i>	34
2.2.2	Métricas	35
2.2.2.1	Erro Médio Quadrático	35
2.2.2.2	Razão do Sinal-Ruído de Pico	35
2.2.2.3	Índice de Similaridade Estrutural	36
2.2.3	BM3D	37
2.3	Algoritmo fractal	38
2.3.1	Codificação fractal	38
3	METODOLOGIA	41
3.1	Modificação no Algoritmo	41
3.2	Conjunto de Imagens	42
3.3	Execução dos Testes e Métricas	42

4	EXPERIMENTAÇÃO E RESULTADOS	44
4.1	Resultados	44
4.1.1	Resultados Ruído Gaussiano	44
4.1.2	Resultados Ruído Sal e Pimenta	45
4.1.3	Resultados Ruído Uniforme	48
4.1.4	Resultados Ruído Gama	50
4.1.5	Resultados Ruído Periódico	52
4.1.6	Resultados Ruído <i>Speckle</i>	54
4.1.7	Comparação de tempo	56
4.2	Discussão	56
5	CONCLUSÃO	58
A	RESULTADOS COMPLEMENTARES	59
A.1	Ruído Gaussiano	59
A.2	Ruído Sal e Pimenta	61
A.3	Ruído Uniforme	62
A.4	Ruído Gama	64
A.5	Ruído Periódico	65
A.6	Ruído <i>Speckle</i>	66
	REFERÊNCIAS	68

1 Introdução

1.1 Problema

A restauração de imagens é um problema fundamental no processamento de imagens, de forma que se tenta aproximar da imagem original a partir de uma imagem que sofreu perda de informação devido ao processo de captura, ao pós-processamento ou à fotografia em condições não ideais. Porém, os métodos tradicionais frequentemente têm dificuldades em alcançar uma boa acurácia em casos mais complexos. Um exemplo disso ocorre quando as características do ruído são desconhecidas e possuem uma natureza complexa.

Uma alternativa para superar essas limitações é a utilização de estruturas fractais, que oferecem um novo paradigma para o processamento de imagens. Segundo [Hamano, Tokuda e Kaneko \(1996\)](#), esse método não codifica diretamente os valores dos pixels, mas sim um "mapeamento" baseado em padrões auto semelhantes. Dessa forma, a decodificação de uma imagem pode ser interpretada como um processo de geração, permitindo uma abordagem diferente para a restauração de imagens.

1.2 Justificativa

O processamento de imagens é uma ferramenta cada vez mais utilizada em diversas áreas do conhecimento. Na medicina, por exemplo, algoritmos de inteligência artificial auxiliam na realização de previsões e diagnósticos precoces. Na indústria, a tecnologia pode ser aplicada na inspeção automatizada de produtos manufaturados. E até na exploração geológica, a formação de "imagens" é possível utilizando vibrações.

De acordo com [Song et al. \(2024\)](#), as características de auto-semelhança e multiescala dos fractais podem aprimorar a qualidade das imagens, além de melhorar o processo de reparação e reconstrução de imagens danificadas. Além disso, essas propriedades permitem preservar mais detalhes da imagem original, reduzindo a perda de informação durante o processamento.

Dessa forma, explorar métodos alternativos aos tradicionais pode ajudar a garantir uma melhor precisão na reconstrução de imagens corrompidas, nas quais os processos tradicionais não lidam de maneira satisfatória.

1.3 Objetivo

O objetivo deste trabalho foi modificar o algoritmo proposto por [Ghazel, Freeman e Vrscay \(2003\)](#), substituindo a métrica Erro Médio Quadrático (*Mean Squared Error* - MSE)

pelo Índice de Similaridade Estrutural (*Structural Similarity Index Measure* - SSIM), visando avaliar o impacto de uma métrica perceptualmente mais fiel à visão humana na qualidade final da imagem reconstruída.

1.4 Objetivos específicos

- Descrever os princípios do algoritmo de restauração de imagem baseado em codificação fractal;
- Preparar as imagens que serão usadas com ruídos para teste;
- Implementar o algoritmo fractal de restauração de imagem;
- Propor e validar uma modificação no cálculo da distância entre sub-blocos da imagem considerada;
- Verificar a eficácia do algoritmo por meio das métricas como MSE, Razão do Sinal-Ruído de Pico (*Peak Signal to Noise Ratio* - PSNR), SSIM;
- Avaliar como a substituição da métrica MSE pelo SSIM afeta o desempenho do algoritmo, comparando-os entre si e com o algoritmo BM3D em diferentes tipos de ruídos.

2 Fundamentação

Neste capítulo serão apresentadas noções matemáticas fundamentais, bem como os conceitos elementares da Geometria Fractal, que servirão de base para a compreensão dos capítulos posteriores. Em seguida, é discutida a remoção de ruído em imagens digitais, destacando os principais ruídos e métricas para análise. Por fim, descreve-se o algoritmo de codificação fractal, e como funciona.

2.1 Geometria Fractal

O conceito de geometria fractal foi introduzido por Benoit Mandelbrot em 1975. De acordo com (MANDELBROT, 1989), os fractais podem ser entendidos como uma linguagem geométrica voltada ao estudo de objetos irregulares e fragmentados, que apresentam as mesmas características em todas as escalas. O termo "fractal" deriva do latim *fractus*, que significa "quebrado" ou "fragmentado", refletindo a natureza desses objetos.

2.1.1 Espaços métricos

Os conceitos abaixo (DOMINGUES, 1982) serão importantes para o desenvolvimento deste trabalho.

Definição 2.1 (Espaço). *Um espaço X é um conjunto, e os pontos deste espaço são chamados de elementos do conjunto.*

Exemplo 2.1. *Um exemplo é $X = \mathbb{R}^2$, ou seja, todo par ordenado (x, y) com $x \in \mathbb{R}$ e $y \in \mathbb{R}$.*

Definição 2.2 (Espaço métrico). *Um espaço métrico (X, d) é um conjunto X com uma função*

$$d : X \times X \rightarrow \mathbb{R} \quad (2.1)$$

que mede a distância entre dois elementos $x, y \in X$. A função d deve satisfazer as seguintes propriedades para ser um espaço métrico:

1. $d(x, y) = d(y, x) \quad \forall x, y \in X$ (Simetria)
2. $0 < d < \infty \quad \text{se } x \neq y$ (Positividade)
3. $d(x, x) = 0 \quad \forall x \in X$ (Identidade)
4. $d(x, z) \leq d(x, y) + d(y, z) \quad \forall x, y, z \in X$ (Desigualdade triangular)

d é chamado de métrica.

Exemplo 2.2. Seja $X = \mathbb{R}^2$, cujos elementos são da forma (x_1, x_2) . E seja a função

$$d : X \times X \rightarrow \mathbb{R}, \quad d((x_1, x_2), (y_1, y_2)) = |x_1 - y_1| + |x_2 - y_2|. \quad (2.2)$$

Para ser um espaço métrico, d deve satisfazer as propriedades da definição 2.2.

Sejam $x, y, z \in X$, $x \neq y$.

Verificando a Propriedade 1:

$$\begin{aligned} d(x, y) &= |x_1 - y_1| + |x_2 - y_2| \\ &= |y_1 - x_1| + |y_2 - x_2| \\ &= d(y, x). \end{aligned}$$

Verificando a Propriedade 2: Como $x \neq y$, existe $i \in \{1, 2\}$ tal que $x_i \neq y_i$. Então:

$$\begin{aligned} d(x, y) &= |x_1 - y_1| + |x_2 - y_2| \\ &> |x_i - y_i| \\ &> 0. \end{aligned}$$

Verificando a Propriedade 3:

$$\begin{aligned} d(x, x) &= |x_1 - x_1| + |x_2 - x_2| \\ &= 0 + 0 \\ &= 0. \end{aligned}$$

Verificando a Propriedade 4:

$$\begin{aligned} d(x, z) &= |x_1 - z_1| + |x_2 - z_2| \\ &= |(x_1 - y_1) + (y_1 - z_1)| + |(x_2 - y_2) + (y_2 - z_2)| \\ &\leq |x_1 - y_1| + |y_1 - z_1| + |x_2 - y_2| + |y_2 - z_2| \\ &= d(x, y) + d(y, z). \end{aligned}$$

Portanto, (X, d) é um espaço métrico.

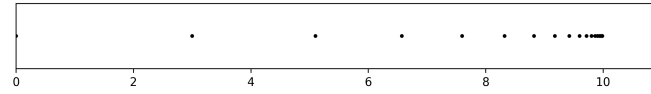
Definição 2.3. Duas métricas d_1 e d_2 no espaço X são equivalentes se existe constantes $0 < c_1 < c_2 < \infty$ tal que

$$c_1 d_1 \leq d_2 \leq c_2 d_1, \quad \forall (x, y) \in X \times X \quad (2.3)$$

Definição 2.4. Uma sequência $\{x_n\}_{n=1}^{\infty}$ de pontos no espaço métrico (X, d) é dita sequência de Cauchy se, para qualquer $\epsilon > 0$, existe $N > 0$ tal que

$$d(x_n, x_m) < \epsilon \quad \forall n, m > N. \quad (2.4)$$

Figura 1 – Representação dos pontos de uma sequência de Cauchy.



Fonte: Elaborada pelo autor.

Embora o exemplo da figura 1 mostre uma sequência que converge para um ponto, a convergência não é uma condição necessária para que uma sequência seja de Cauchy.

Definição 2.5. Uma sequência $\{x_n\}_{n=1}^{\infty}$ no espaço métrico (X, d) converge para um ponto $x \in X$ se, para qualquer $\epsilon > 0$, existe $N > 0$ tal que

$$d(x_n, x) < \epsilon, \quad \forall n > N \quad (2.5)$$

Neste caso, o ponto $x \in X$ é o ponto de convergência e é chamado de limite da sequência, o qual é representado por:

$$x = \lim_{n \rightarrow \infty} x_n. \quad (2.6)$$

Definição 2.6. Um espaço métrico (X, d) é dito completo se toda sequência de Cauchy em X possui um limite $x \in X$.

Pela definição 2.5, segue que, em um espaço métrico completo, toda sequência de Cauchy é convergente, e o seu limite pertence ao próprio espaço X .

Definição 2.7. Seja S um subconjunto de um espaço métrico (X, d) . S é dito compacto se toda sequência infinita $\{x_n\}_{n=1}^{\infty}$ contém uma subsequência com limite em S .

Definição 2.8. Seja S um subconjunto de um espaço métrico (X, d) . S é limitado se existe um número $R > 0$ tal que

$$d(x, y) \leq R \quad \forall x, y \in S. \quad (2.7)$$

Definição 2.9. Seja $S \subset X$ um subconjunto de um espaço métrico (X, d) . S é totalmente limitado se, para qualquer $\epsilon > 0$, existe um conjunto finito de pontos $\{s_1, s_2, \dots, s_n\} \subset S$ tal que, para qualquer $x \in S$, existe algum s_i com $d(x, s_i) < \epsilon$. O conjunto $\{s_1, s_2, \dots, s_n\}$ é chamado de $(\epsilon$ -net).

De forma intuitiva, é como se fosse possível cobrir o subconjunto S com bolas abertas de raio ϵ .

Teorema 2.1. Seja (X, d) um espaço métrico completo, então, um subconjunto $S \subset X$ é compacto se, e somente se, é fechado e totalmente limitado.

A demonstração completa deste resultado pode ser consultada em (BARNESLEY, 2014, p. 20). Este teorema é fundamental, pois garante condições em que certos mapeamentos contrativos admitem pontos fixos.

Definição 2.10. *Seja $S \subset X$ um subconjunto no espaço métrico (X, d) . S é aberto se para cada $x \in S$ existe um $\epsilon > 0$ tal que $B(x, \epsilon) = \{y \in X : d(x, y) \leq \epsilon\} \subset S$.*

Proposição 2.1. *Se (X, d) é um espaço métrico, então um conjunto $S \subset X$ é aberto se, e somente se, o seu complementar $X \setminus S$ é fechado.*

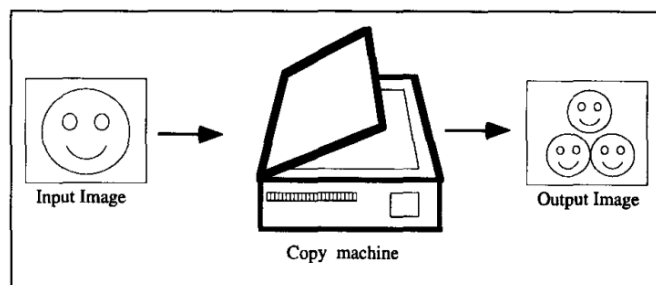
A demonstração desta proposição pode ser encontrada em (BARNESLEY, 2014, p. 21).

2.1.2 Fractais

Fractais são formas geométricas que apresentam estruturas detalhadas em escalas arbitrariamente pequenas e exibem autossimilaridade, ou seja, cada parte do objeto repete, de maneira exata ou aproximada, a forma original. Esses conjuntos surgem frequentemente em fenômenos naturais, como montanhas, nuvens e folhas, nos quais padrões semelhantes se manifestam em diferentes níveis de observação.

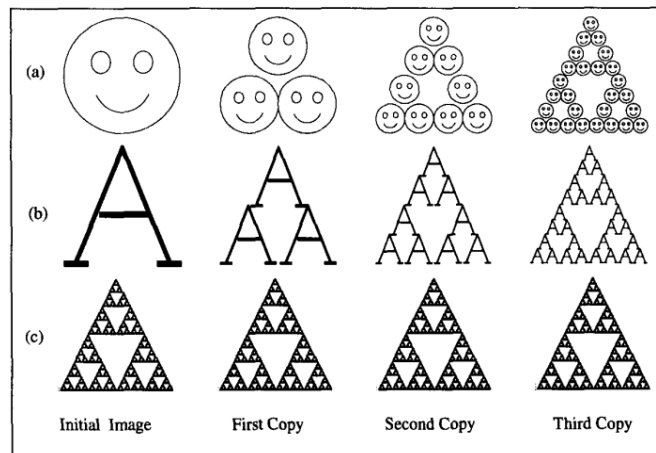
Um modo intuitivo de compreender esse processo foi proposto por Fisher (1994), que compara um IFS a uma copiadora. Essa copiadora é capaz de reduzir sua entrada e reproduzi-la em diferentes posições sobre uma folha. Se ela for retroalimentada com a própria saída repetidas vezes, o processo converge para uma imagem estável, que é o fractal. As figuras 2 e 3 demonstram esse processo.

Figura 2 – Exemplo da copiadora.



Fonte: (FISHER, 1994)

Figura 3 – Iterações da copiadora.



Fonte: (FISHER, 1994)

Nesta seção, serão definidos formalmente os fractais gerados por sistemas de funções iteradas (IFS), apresentando as bases matemáticas que descrevem sua construção e as propriedades que garantem a existência de seus atratores.

2.1.3 Espaço de Hausdorff

Para que os Fractais estejam bem definidos é necessário uma métrica e, conseqüentemente, um espaço, o que será feito a seguir.

Definição 2.11. *Seja (X, d) um espaço métrico completo. É definido $\mathcal{H}(X)$ como o espaço, o qual os pontos são os subconjuntos compactos não vazios de X*

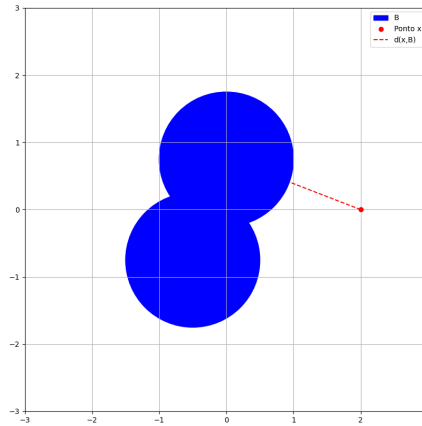
O espaço $\mathcal{H}(X)$ é chamado de **espaço de Hausdorff**.

Definição 2.12. *Seja (X, d) um espaço métrico completo, $x \in X$, e $B \in \mathcal{H}(X)$. A distância de um ponto x ao conjunto B :*

$$d(x, B) = \min\{d(x, y) | y \in B\}. \quad (2.8)$$

A figura 4 é um exemplo da distância entre um conjunto e um ponto.

Figura 4 – Exemplo da distância de um ponto à um conjunto.



Fonte: Elaborada pelo autor.

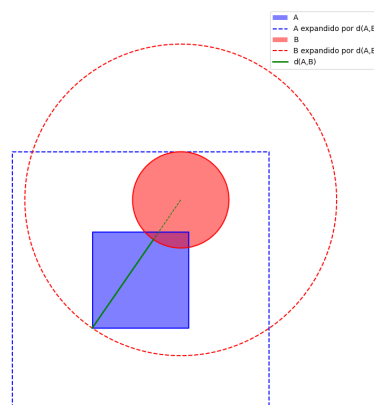
Definição 2.13. *Seja (X, d) um espaço métrico completo. A distância de $A, B \in \mathcal{H}(X)$ é definida como:*

$$d(A, B) = \max\{d(x, B) | x \in A\}. \quad (2.9)$$

De acordo com [Barnsley \(2014, p. 29\)](#), utilizando a compacidade de A e B pode-se provar que a distância esta bem definida e existem dois pontos $x \in A$ e $y \in B$, tal que $d(A, B) = d(x, y)$

A figura 5 da distância entre dois conjuntos $A, B \in \mathcal{H}(X)$:

Figura 5 – Exemplo da distância entre dois conjuntos.



Fonte: Elaborada pelo autor.

Definição 2.14. *Seja (X, d) um espaço métrico completo. A distância de Hausdorff entre dois*

pontos A e B em $\mathcal{H}(X)$ é definida como:

$$h(A, B) = \max\{d(A, B), d(B, A)\}. \quad (2.10)$$

Para que h seja uma métrica em $\mathcal{H}(X)$, ela deve satisfazer as quatro propriedades definidas em 2.2: simetria, não negatividade, identidade e desigualdade triangular. A demonstração de que a métrica satisfaz essas propriedades pode ser consultada em Barnsley (2014, p. 33).

Dessa forma, o par $(\mathcal{H}(X), h)$ constitui um espaço métrico completo, conhecido como "espaço dos fractais".

Teorema 2.2. *Seja (X, d) um espaço métrico completo. Então $(\mathcal{H}(X), h)$ é um espaço métrico completo. Dessa forma se $\{A_n \in \mathcal{H}(X)\}_{n=1}^{\infty}$ é uma sequência de Cauchy, então:*

$$A = \lim_{n \rightarrow \infty} A_n \in \mathcal{H}(X). \quad (2.11)$$

A demonstração completa desse resultado é extensa e pode ser consultada em Barnsley (2014, p. 35). Esse teorema é fundamental, pois garante que o espaço dos fractais possui as propriedades necessárias para a aplicação do Teorema do Ponto Fixo, que será abordado a próxima etapa.

2.1.4 Mapeamentos Contrativos

De acordo com Barnsley (2014, p. 42), a geometria fractal estuda conjuntos complexos definidos em espaços geometricamente simples, como $\mathbb{R}$, $\mathbb{R}^2$ e $\mathbb{C}$. Na geometria fractal determinística, esses conjuntos são caracterizados por possuírem propriedades de invariância sob transformações geométricas que, em geral, podem ser descritas por poucos parâmetros.

Entre essas transformações, destacam-se as **transformações afins** em $\mathbb{R}^2$, que podem ser representadas por uma matriz 2×2 e um vetor de translação. Como o foco deste trabalho está na construção de fractais por meio de *Sistemas de Funções Iteradas* (IFS), serão consideradas apenas transformações contrativas em $\mathbb{R}^2$.

Uma transformação afim em $\mathbb{R}^2$ é uma função $w : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ da forma

$$w(x) = Ax + b, \quad (2.12)$$

em que A é uma matriz 2×2 e b é um vetor em $\mathbb{R}^2$. O termo Ax representa uma transformação linear, podendo incluir: rotação; reflexão; cisalhamento; ou escala, enquanto b realiza uma translação.

Definição 2.15. *Uma transformação $f : X \rightarrow X$ definida em um espaço métrico (X, d) é uma contração se existe uma constante s , de modo que $0 \leq s < 1$, tal que:*

$$d(f(x), f(y)) \leq s \cdot d(x, y), \quad \forall x, y \in X. \quad (2.13)$$

Exemplo 2.3. Considere o espaço métrico $(\mathbb{R}^2, d)$, onde d é a métrica euclidiana, isto é, $d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ para $x = (x_1, x_2)$, $y = (y_1, y_2) \in \mathbb{R}^2$. Defina a transformação linear $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ pela matriz

$$A = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}, \quad (2.14)$$

de modo que, para $x = (x_1, x_2)$, temos $f(x) = A \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \frac{x_1}{2} \\ \frac{x_2}{2} \end{pmatrix}$.

Vamos verificar que f é uma contração. Sejam $x = (x_1, x_2)$, $y = (y_1, y_2) \in \mathbb{R}^2$. Representando x e y como vetores coluna, temos

$$f(x) = A \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \frac{x_1}{2} \\ \frac{x_2}{2} \end{pmatrix}, \quad f(y) = A \cdot \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \frac{y_1}{2} \\ \frac{y_2}{2} \end{pmatrix}. \quad (2.15)$$

A distância entre $f(x)$ e $f(y)$ é dada por

$$\begin{aligned} d(f(x), f(y)) &= \sqrt{\left(\frac{x_1}{2} - \frac{y_1}{2}\right)^2 + \left(\frac{x_2}{2} - \frac{y_2}{2}\right)^2} \\ &= \sqrt{\frac{(x_1 - y_1)^2}{4} + \frac{(x_2 - y_2)^2}{4}} \\ &= \frac{1}{2} \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \\ &= \frac{1}{2} d(x, y). \end{aligned}$$

Portanto, $d(f(x), f(y)) = \frac{1}{2} d(x, y)$, e como $s = \frac{1}{2}$ satisfaz $0 \leq s < 1$, a transformação f é uma contração com constante $s = \frac{1}{2}$.

Definição 2.16. Seja $f : X \rightarrow X$ uma função definida em um espaço X . Um ponto $x_0 \in X$ é chamado de ponto fixo de f se:

$$f(x_0) = x_0. \quad (2.16)$$

Teorema 2.3. (O Teorema do Mapeamento Contrativo) Seja $f : X \rightarrow X$ uma contração em um espaço métrico completo (X, d) . Logo f possui um único ponto fixo $x_f \in X$, e para qualquer ponto $x \in X$ a sequência $f^{on}(x) : n = 0, 1, 2, \dots$, onde $f^{on}(x)$ denota a aplicação iterada de f sobre x n vezes, isto é:

$$f^{on}(x) = f(f(\dots f(x) \dots)) \quad (2.17)$$

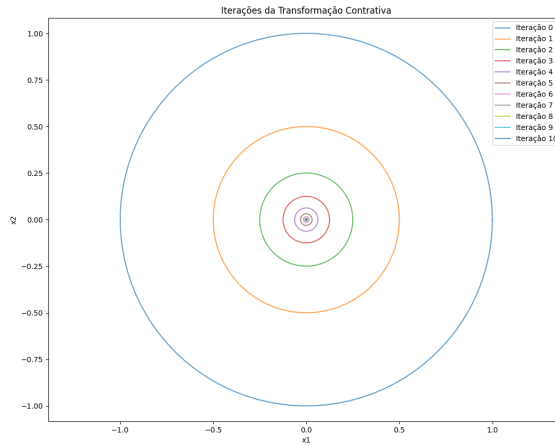
converge para x_f , isto é:

$$\lim_{n \rightarrow \infty} f^{on}(x) = x_f \quad (2.18)$$

A prova do Teorema do Mapeamento Contrativo pode ser encontrada em [Barnsley \(2014, p. 75\)](#).

A figura 6 mostra um exemplo do mapeamento contrativo para $f = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$.

Figura 6 – Exemplo do teorema do mapeamento contrativo para o exemplo 2.3.



Fonte: Elaborada pelo autor.

Lema 2.1. *Seja $w : X \rightarrow X$ uma contração em um espaço métrico (X, d) , então w é contínuo.*

Lema 2.2. *Seja $w : X \rightarrow X$ uma contração contínua no espaço métrico (X, d) , então $w : \mathbb{H}(X) \rightarrow \mathbb{H}(X)$.*

Lema 2.3. *Seja $w : X \rightarrow X$ uma contração no espaço métrico (X, d) , então $w : \mathbb{H}(X) \rightarrow \mathbb{H}(X)$.*

Definição 2.17. *Um Sistema de Funções Iteradas (IFS) consiste em um espaço métrico completo (X, d) junto com um conjunto finito de contrações $w_n : X \rightarrow X$, com os respectivos fatores de contração s_n , para $n = 1, 2, \dots, N$. A notação para o IFS é $\{X; w_n, n = 1, 2, \dots, N\}$ e o fator de contração s é dado por $s = \max\{s_1, s_2, \dots, s_N\}$.*

Conforme Barnsley (2014) o termo "hiperbólico" pode ser omitido em alguns contextos, uma vez que, na prática, a nomenclatura "IFS" é frequentemente usada para designar um conjunto finito de funções atuando em um espaço métrico, sem a exigência de que todas sejam contrações.

Teorema 2.4. *Seja $\{X; w_n, n = 1, 2, \dots, N\}$ um IFS hiperbólico com o fator de contração s . Então, a transformação $W : \mathcal{H}(X) \rightarrow \mathcal{H}(X)$ é definida por:*

$$W(B) = \bigcup_{n=1}^N w_n(B). \quad (2.19)$$

Para todo $B \in \mathcal{H}(X)$, é uma contração no espaço métrico completo $(H(X), h(d))$ com fator de contração $s = \max\{s_n \mid n = 1, 2, \dots, N\}$. Isto é,

$$h(W(B), W(C)) \leq s \cdot h(B, C), \quad \forall B, C \in \mathcal{H}(X). \quad (2.20)$$

Além disso, seu único ponto fixo, $A \in H(X)$, satisfaz

$$A = W(A) = \bigcup_{n=1}^N w_n(A). \quad (2.21)$$

Definição 2.18. O ponto fixo $A \in \mathcal{H}(X)$ é chamado de atrator do IFS.

2.1.5 Dois algoritmos para exibir fractais gerados por IFS

2.1.5.1 Algoritmo determinístico

A ideia é computar a sequência de iterações $A_n = W^n(A)$ começando de um conjunto inicial A_0 . Como o atrator é independente do conjunto inicial, então podemos começar com qualquer A_0 , como é mostrado no algoritmo 1.

Algoritmo 1: Algoritmo determinístico para computar o atrator de um IFS

Data: Conjunto inicial compacto $A_0 \in \mathcal{H}(X)$; funções contrativas $\{w_1, w_2, \dots, w_N\}$; número de iterações K .

Result: Aproximação do atrator A_K .

$A \leftarrow A_0$;

for $k \leftarrow 1$ **to** K **do**

$A \leftarrow \bigcup_{n=1}^N w_n(A)$;

return A ;

Esse algoritmo tem como complexidade de tempo de $O(N^K)$, onde N é a quantidade de transformações afins e K é o número de iterações. Isso se deve ao fato de que a cada iteração o conjunto A recebe K vezes pontos.

Considerando o fractal triângulo de Sierpiński que possui as seguintes transformações:

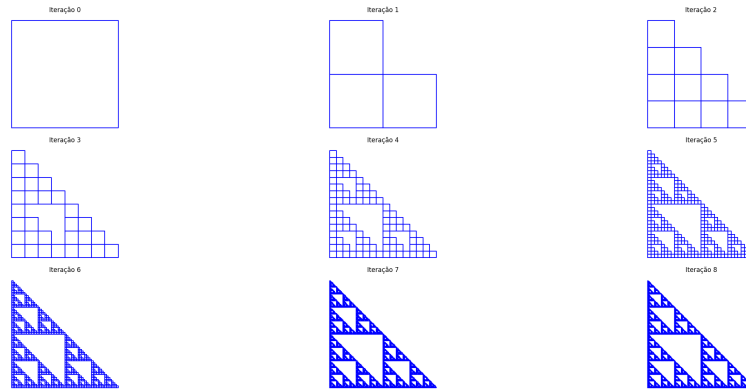
$$w_1(x, y) = \left(\frac{x}{2}, \frac{y}{2} + \frac{1}{2} \right)$$

$$w_2(x, y) = \left(\frac{x}{2} + \frac{1}{2}, \frac{y}{2} \right)$$

$$w_3(x, y) = \left(\frac{x}{2}, \frac{y}{2} \right)$$

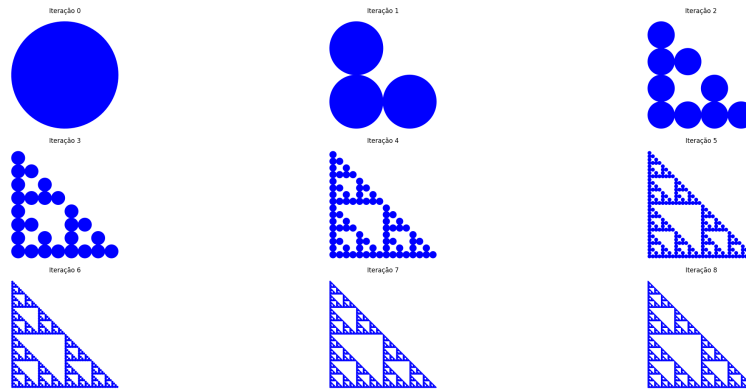
As figuras 7 e 8 mostram a aplicação do algoritmo em diferentes conjuntos iniciais.

Figura 7 – Triângulo de Sierpiński começando um quadrado



Fonte: Elaborada pelo autor.

Figura 8 – Triângulo de Sierpiński começando um círculo



Fonte: Elaborada pelo autor.

2.1.5.2 Algoritmo randômico

A ideia do algoritmo de randômico, também conhecido como "*Chaos Game*", é associar uma probabilidade p_i a cada transformação w_i do IFS, de modo que p_i represente a "densidade" ou a contribuição relativa de w_i para a formação do atrator. Dessa forma as probabilidades devem satisfazer $\sum_{i=1}^N p_i = 1$ e $p_i > 0$ e são usadas para selecionar aleatoriamente as transformações a cada iteração. Como resultado, os pontos gerados convergem para o atrator A , com uma distribuição de pixels que reflete a densidade do fractal, sendo mais concentrada nas regiões de maior "massa" (ou medida invariante).

Considere N transformações afins do tipo $w_i(x) = A_i x + b_i$, a probabilidade da transformação i é definida como:

$$p_i = \frac{|A_i|}{\sum_{j=1}^N |A_j|}. \quad (2.22)$$

Considerando somente o $\mathbb{R}^2$, então:

$$w_i(x) = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} \quad (2.23)$$

$$p_i = \frac{|a_i d_i - b_i c_i|}{\sum_{j=1}^N |a_j d_j - b_j c_j|}. \quad (2.24)$$

A seguir, apresenta-se um exemplo dos parâmetros utilizados para gerar o fractal da samambaia de Barnsley, assim como a probabilidade associada a cada transformação.

Tabela 1 – Parâmetros e probabilidades das transformações do fractal da samambaia de Barnsley.

w	a	b	c	d	e	f	p
$f1$	0	0	0	0.16	0	0	0.01
$f2$	0.85	0.04	-0.04	0.85	0	1.60	0.85
$f3$	0.20	-0.26	0.23	0.22	0	1.60	0.07
$f4$	-0.15	0.28	0.26	0.24	0	0.44	0.07

Fonte: Elaborado pelo autor.

O pseudocódigo a seguir descreve o funcionamento do algoritmo randômico.

Algoritmo 2: Algoritmo randômico para computar o atrator de um IFS

Data: funções contrativas $\{w_1, w_2, \dots, w_N\}$; número de iterações K .

Result: Aproximação do atrator A .

$soma_determinantes \leftarrow 0$;

for $i \leftarrow 1$ **to** N **do**

$p_i \leftarrow |w_i|$;
 $soma_determinantes \leftarrow soma_determinantes + p_i$;

for $i \leftarrow 1$ **to** N **do**

$p_i \leftarrow p_i / soma_determinantes$;

$A \leftarrow \emptyset$;

$x \leftarrow$ ponto inicial arbitrário em $\mathbb{R}^2$;

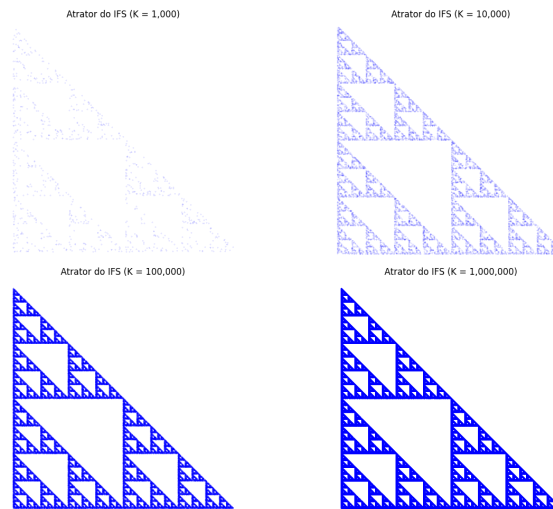
for $k \leftarrow 1$ **to** K **do**

Gere um número aleatório $r \in [0, 1]$;
 Escolha i tal que $\sum_{j=1}^{i-1} p_j < r \leq \sum_{j=1}^i p_j$;
 $x \leftarrow w_i(x)$;
 $A \leftarrow A \cup \{x\}$;

return A ;

As figuras 9 e 10 mostram a aplicação do algoritmo para diferentes IFS.

Figura 9 – Triângulo de Sierpiński gerado pelo algoritmo randômico com número de iterações crescente



Fonte: Elaborada pelo autor.

Figura 10 – Densidade do fractal Samambaia de Barnsley gerado pelo algoritmo randômico



Fonte: Elaborada pelo autor.

2.2 Remoção de Ruído em Processamento de Imagens

De acordo com [Jähne \(2005\)](#), uma imagem digital pode ser definida como uma função bidimensional $f(x, y)$, em que x e y representam as coordenadas espaciais no plano. O valor de f em cada par de coordenadas (x, y) é denominado intensidade ou nível de cinza da imagem nesse ponto, sendo esses valores finitos e discretos.

Além disso, $f(x, y)$ não representa um ponto isolado, mas sim uma pequena região retangular denominada *pixel*, que corresponde à menor unidade da imagem capaz de armazenar

informação de intensidade.

O ruído pode ser entendido como qualquer modificação indesejada no sinal da imagem, representada por $\eta(x, y)$.

Quando o ruído é aditivo, a imagem degradada $g(x, y)$ pode ser expressa por:

$$g(x, y) = f(x, y) + \eta(x, y). \quad (2.25)$$

Por outro lado, alguns tipos de ruído são multiplicativos, sendo modelados por:

$$g(x, y) = f(x, y) \cdot \eta(x, y). \quad (2.26)$$

No entanto, é possível transformar esse modelo em uma forma aditiva por meio da aplicação do logaritmo:

$$g(x, y) = f(x, y) \cdot \eta(x, y) \quad (2.27)$$

$$\log(g(x, y)) = \log(f(x, y) \cdot \eta(x, y)) \quad (2.28)$$

$$\log(g(x, y)) = \log(f(x, y)) + \log(\eta(x, y)) \quad (2.29)$$

$$g(x, y) = e^{\log(g(x, y))} \quad (2.30)$$

Por fim, nem todo tipo de ruído se enquadra nesses modelos, como é o caso do ruído Sal e Pimenta que será mostrado na sequência.

2.2.1 Tipos de ruído

Muitos tipos de ruído podem ser descritos matematicamente por uma função de densidade de probabilidade, que define a distribuição estatística dos valores de intensidade afetados. E a área sob a curva da função em um intervalo específico representa a probabilidade de que a variável aleatória assuma valores dentro desse intervalo.

2.2.1.1 Ruído Gaussiano

O ruído Gaussiano é um tipo de sinal que possui sua função de densidade de probabilidade igual a distribuição normal. E de acordo com [Boyat e Joshi \(2015\)](#) esse ruído é geralmente causado por fontes naturais, como a vibração térmica de átomos e a natureza discreta da radiação de objetos quentes.

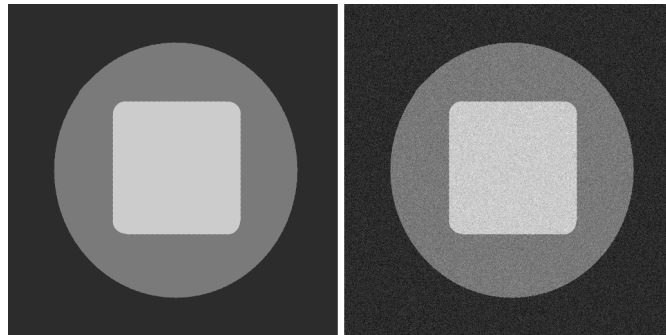
A função densidade de probabilidade do ruído Gaussiano é expressa por:

$$P(g) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(g-\mu)^2}{2\sigma^2}} \quad (2.31)$$

em que g representa o valor de intensidade ou nível de cinza, μ é a média dos valores de intensidade e σ é o desvio padrão, que determina a dispersão do ruído em torno da média.

As figuras [11](#) e [12](#) mostram, respectivamente, um exemplo do ruído e seu histograma.

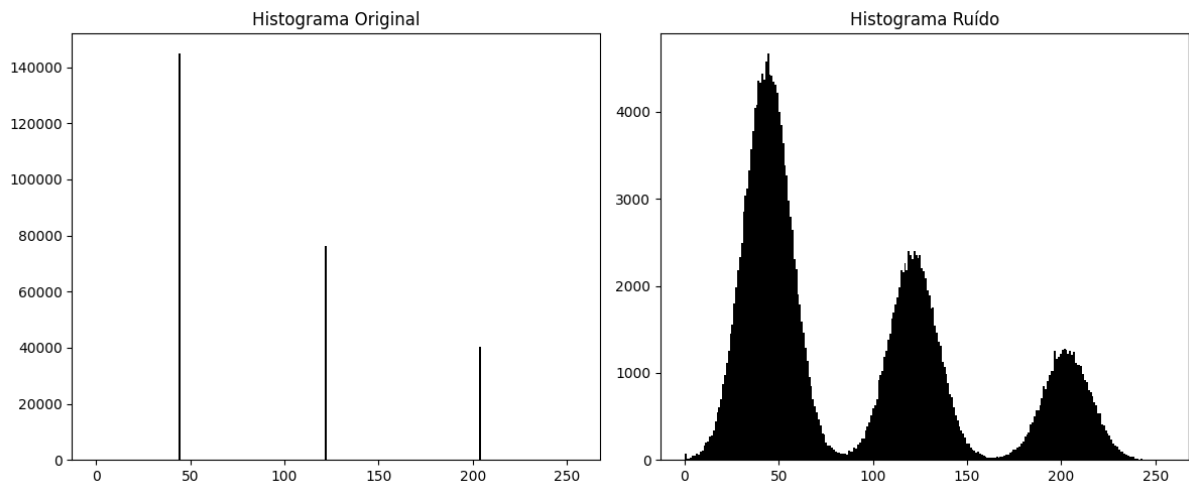
Figura 11 – Exemplo ruído Gaussiano



$$\mu = 0 \text{ e } \sigma = 0.05$$

Fonte: Elaborada pelo autor.

Figura 12 – Histograma da imagem com ruído Gaussiano



Fonte: Elaborada pelo autor.

2.2.1.2 Ruído Sal e Pimenta

No ruído do tipo Sal e Pimenta, apenas alguns pixels da imagem são afetados, apresentando valores significativamente diferentes dos seus vizinhos. Esses pixels corrompidos assumem, em geral, o valor mínimo (0) ou o valor máximo (255) de intensidade, o que gera pontos brancos e pretos dispersos pela imagem.

De acordo com [Boyat e Joshi \(2015\)](#), esse tipo de ruído é geralmente causado por falhas no processo de conversão do analógico para digital ou por erros na transmissão de bits durante a aquisição ou comunicação da imagem.

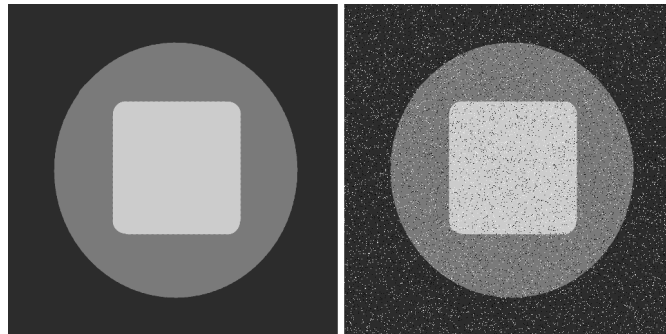
A função densidade de probabilidade do ruído Sal e Pimenta pode ser expressa como:

$$P(g) = \begin{cases} p_1, & \text{se } g = 0, \\ p_2, & \text{se } g = 255, \\ 0, & \text{caso contrário,} \end{cases} \quad (2.32)$$

em que p_1 e p_2 representam as probabilidades de ocorrência dos níveis de intensidade correspondentes ao preto(pimenta) e ao branco(sal), respectivamente.

As figuras 13 e 14 mostram, respectivamente, um exemplo do ruído e seu histograma.

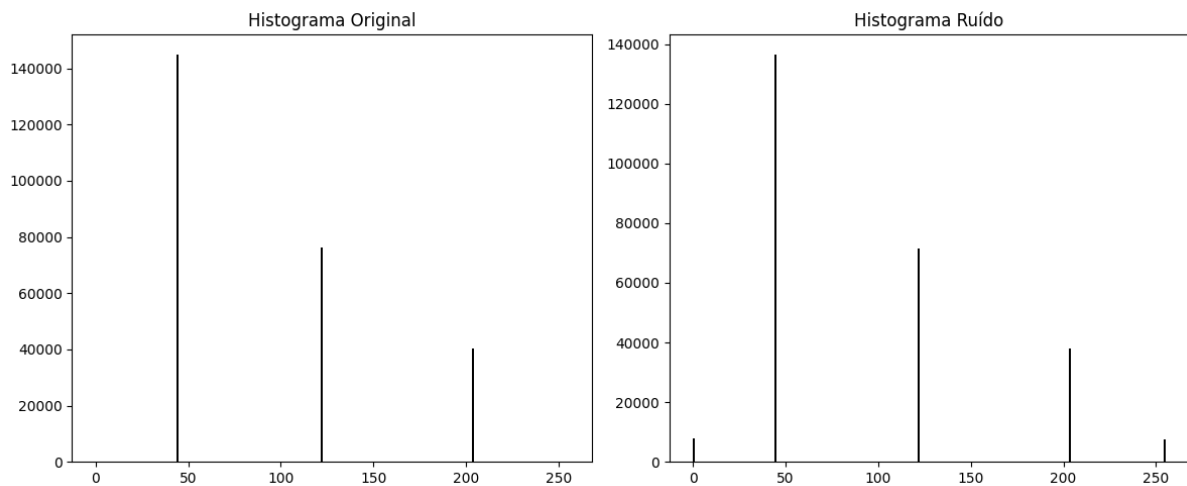
Figura 13 – Imagem corrompida pelo sal e pimenta



$$p_1 = p_2 = 0.03.$$

Fonte: Elaborada pelo autor.

Figura 14 – Histograma da imagem com ruído Sal e Pimenta



Fonte: Elaborada pelo autor.

2.2.1.3 Ruído Uniforme

O ruído uniforme é caracterizado por possuir a mesma probabilidade de ocorrência para todos os valores dentro de um intervalo definido. Dessa forma, a intensidade do ruído é

distribuída de forma constante entre o intervalo, resultando em variações aleatórias e igualmente prováveis na imagem.

De acordo com [Boyat e Joshi \(2015\)](#) esse tipo de ruído está geralmente presente quando dados analógicos são convertidos em dados digitais.

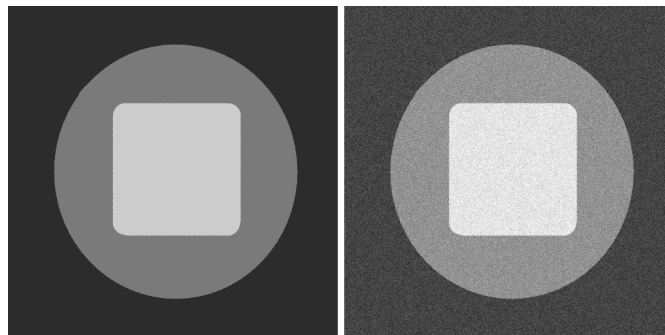
A função densidade de probabilidade desse tipo de ruído é dada por:

$$P(g) = \begin{cases} \frac{1}{b-a}, & \text{se } a \leq g \leq b, \\ 0, & \text{caso contrário,} \end{cases} \quad (2.33)$$

em que a e b representam respectivamente os valores mínimo e máximo de intensidade em que o ruído ocorre.

As figuras 15 e 16 mostram, respectivamente, um exemplo do ruído e seu histograma.

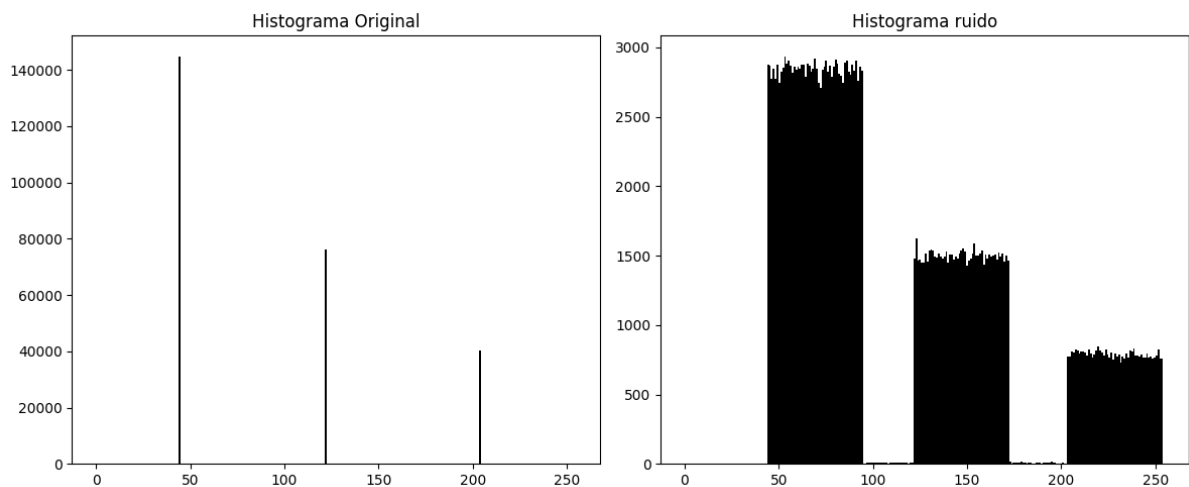
Figura 15 – Imagem corrompida pelo ruído uniforme.



$$a = 0, b = 0.2$$

Fonte: Elaborada pelo autor.

Figura 16 – Histograma da imagem com ruído uniforme



Fonte: Elaborada pelo autor.

2.2.1.4 Ruído Gamma

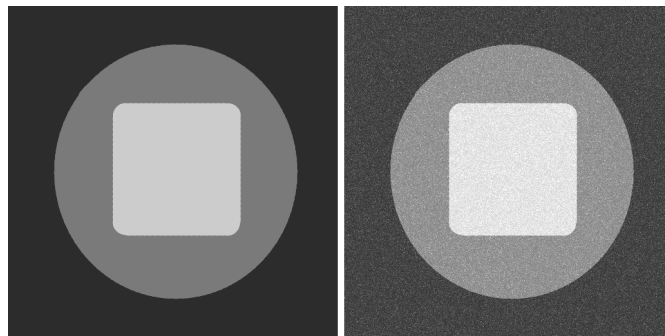
O ruído Gamma é um tipo de sinal que possui sua função de densidade de probabilidade igual a distribuição Gamma. E de acordo com [Boyat e Joshi \(2015\)](#) esse ruído geralmente ocorre em imagens geradas por laser.

A função de densidade de probabilidade da distribuição Gamma é expressa por:

$$P(g) = \begin{cases} \frac{a^b g^{b-1} e^{-ag}}{(b-1)!}, & \text{se } g \geq 0 \\ 0, & \text{se } g < 0 \end{cases} \quad (2.34)$$

As figuras 17 e 18 mostram, respectivamente, um exemplo do ruído e seu histograma.

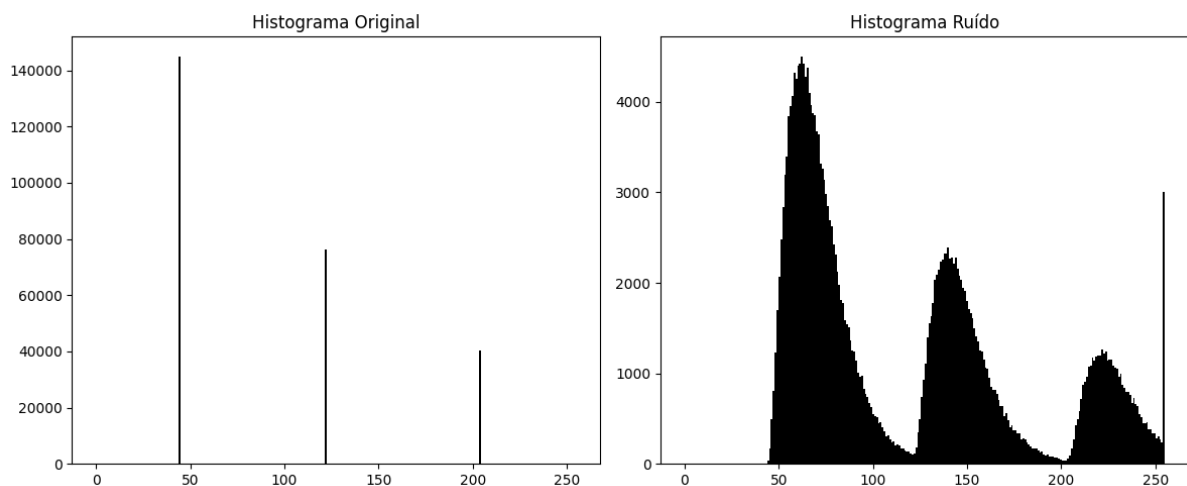
Figura 17 – Imagem corrompida pelo ruído Gamma.



$$a = 3.0, b = 0.035.$$

Fonte: Elaborada pelo autor.

Figura 18 – Histograma da imagem com ruído Gamma



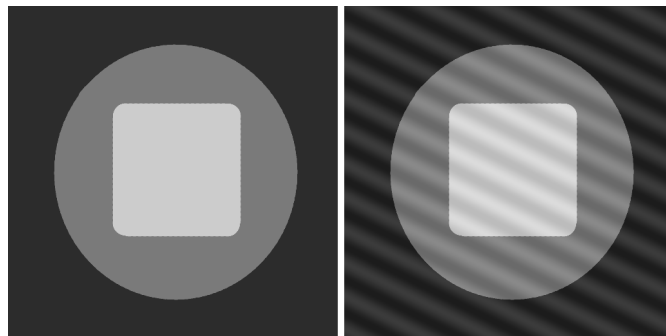
Fonte: Elaborada pelo autor.

2.2.1.5 Ruído Periódico

O ruído periódico, diferente dos anteriores, é espacialmente dependente, isto é, varia conforme a posição dos pixels na imagem. De acordo com [Boyat e Joshi \(2015\)](#) esse ruído é gerado por interferências eletrônicas no sinal de alimentação no processo de aquisição da imagem.

As figuras 19 e 20 mostram, respectivamente, um exemplo do ruído e seu histograma.

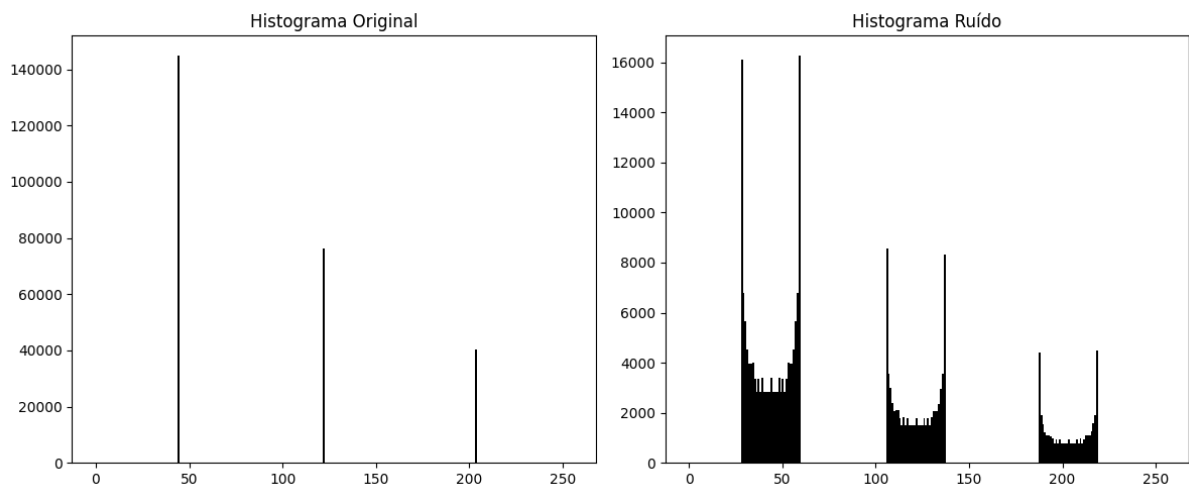
Figura 19 – Imagem corrompida pelo ruído periódico.



$$0.125 \cdot \sin(20\pi(0.5 * x - y))$$

Fonte: Elaborada pelo autor.

Figura 20 – Histograma da imagem com ruído periódico



Fonte: Elaborada pelo autor.

Nesse caso, o histograma não representa todas as possíveis variações do ruído periódico, mas apenas o comportamento específico observado na figura 20. Dessa forma, o histograma representa apenas a distribuição de intensidades para uma ocorrência específica do ruído, sem evidenciar seu comportamento espacial. Para capturar o comportamento espacial do ruído periódico, seria necessário usar ferramentas como a transformada de Fourier

2.2.1.6 Ruído *speckle*

O ruído *Speckle* é um tipo de ruído multiplicativo, que segundo [Boncelet \(2009\)](#), ocorre principalmente devido à utilização de luz coerente (por laser ou transmissor de radar) ou por efeitos atmosféricos durante o processo de aquisição da imagem. Esse tipo de ruído é considerado um dos mais complexos, pois é dependente do sinal e é espacialmente dependente, ou seja, sua intensidade e padrão estão relacionados às características locais da imagem.

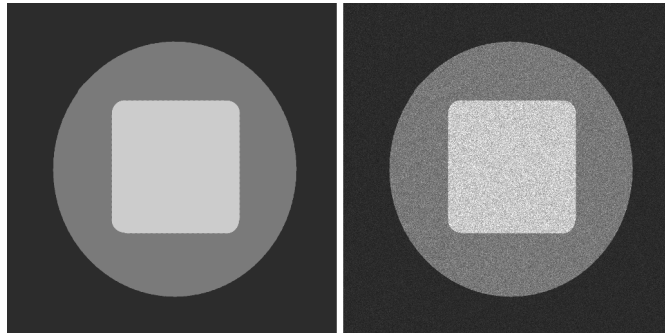
Além disso, [Boncelet \(2009\)](#) explica que esse fenômeno ocorre quando uma luz coerente incide sobre uma superfície e parte dela é refletida de volta. Devido às pequenas irregularidades microscópicas na rugosidade da superfície dentro de um mesmo pixel, o sinal refletido sofre flutuações aleatórias em sua fase e amplitude. Essas flutuações podem se combinar de forma construtiva, resultando em intensidades mais elevadas, ou de forma destrutiva, produzindo regiões com intensidade reduzida.

A função densidade de probabilidade do ruído *Speckle* pode ser expressa por uma distribuição do tipo gama:

$$F(g) = \frac{g^{\alpha-1} e^{-\frac{g}{a}}}{(\alpha-1)! a^{\alpha}} \quad (2.35)$$

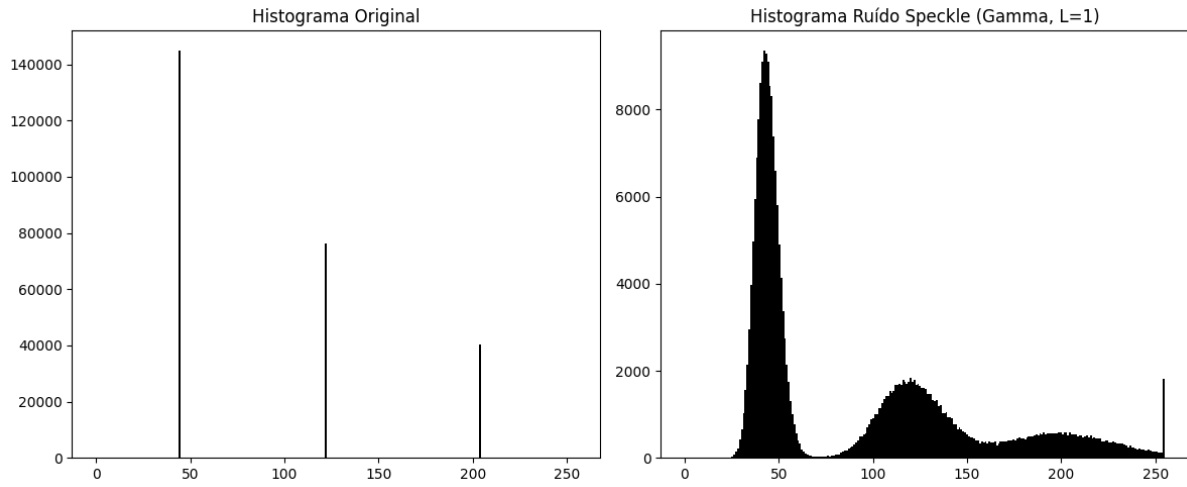
As figuras 21 e 22 mostram, respectivamente, um exemplo do ruído e seu histograma.

Figura 21 – Imagem corrompida pelo ruído *speckle*.



$$a = 50, \alpha = \frac{1}{50}$$

Fonte: Elaborada pelo autor.

Figura 22 – Histograma do ruído *speckle*

Fonte: Elaborada pelo autor.

Nesse caso, observa-se que o ruído é mais intenso nas regiões mais claras da imagem, evidenciando sua natureza multiplicativa

2.2.2 Métricas

A seguir serão apresentadas algumas métricas que serão utilizadas para analisar o desempenho dos algoritmos de restauração.

2.2.2.1 Erro Médio Quadrático

De acordo com [Najjar \(2024\)](#), o MSE é uma das métricas mais utilizadas para avaliar a qualidade de uma imagem. Ele quantifica a diferença média quadrática entre os pixels da imagem original e os da imagem degradada pelo ruído.

Quanto menor o valor do MSE, mais próxima a imagem processada está da original. Entretanto, embora seja amplamente empregado para medir a discrepância entre duas imagens, o MSE nem sempre apresenta correlação direta com a percepção humana de qualidade visual.

O cálculo do MSE é dado por:

$$MSE = \frac{1}{N} \sum_{i=1}^N (I_i - J_i)^2 \quad (2.36)$$

em que N é o número total de pixels da imagem, I_i é a intensidade do pixel i na imagem original I e J_i é a intensidade do pixel i na imagem degradada J .

2.2.2.2 Razão do Sinal-Ruído de Pico

A PSRN é uma medida matemática da qualidade de uma imagem baseada na diferença de intensidade entre os pixels de duas imagens.

De acordo com [Najjar \(2024\)](#), essa métrica é amplamente utilizada para avaliar a qualidade de imagens e vídeos comprimidos, pois indica o quão degradada uma imagem está em relação à original.

A PSNR é calculada a partir do MSE e expressa em decibéis (dB), conforme a seguinte equação:

$$PSNR = 10 \log_{10} \frac{MAX_I^2}{MSE} \quad (2.37)$$

em que MAX_I representa o maior valor possível de intensidade para um pixel, no caso de uma imagem de 8-bits $MAX_I = 255$.

Assim como o MSE, a PSNR nem sempre apresenta uma boa correlação com a percepção humana de qualidade visual, além de ser sensível a pequenas variações nos valores de intensidade dos pixels.

2.2.2.3 Índice de Similaridade Estrutural

De acordo com [Najjar \(2024\)](#), o SSIM é amplamente utilizado para avaliar a similaridade entre duas imagens, considerando aspectos perceptuais da visão humana. Diferentemente de métricas baseadas apenas em erro absoluto, como o MSE e o PSNR, o SSIM compara luminância, contraste e estrutura entre as imagens, retornando um valor entre -1 e 1 , em que 1 indica similaridade total.

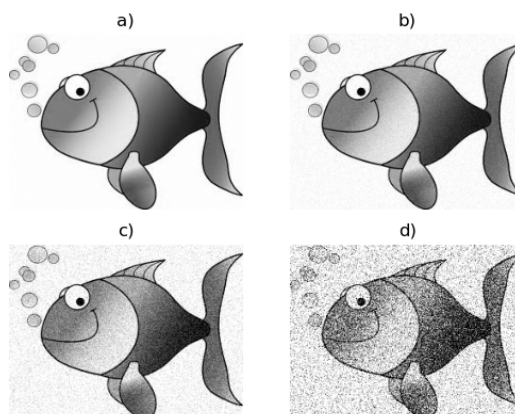
A métrica é normalmente calculada em blocos de tamanho fixo, cujos resultados são posteriormente combinados para formar o valor global de similaridade. Matematicamente, o SSIM entre dois blocos de imagem x e y é definido por:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (2.38)$$

em que μ_x e μ_y representam as médias das intensidades dos blocos x e y , respectivamente; σ_x e σ_y são os desvios padrão correspondentes; σ_{xy} é a covariância entre os valores dos pixels dos blocos x e y ; e c_1 e c_2 são constantes utilizadas para evitar instabilidades numéricas quando as médias ou variâncias são próximas de 0

A figura 23 mostram as métricas 2.36, 2.37 e 2.38 aplicadas a imagens com ruído.

Figura 23 – Exemplo das métricas em diferentes intensidades de ruídos



a) Imagem original; b) $MSE = 28.46$, $PSNR = 33.12$, $SSIM = 0.81$; c) $MSE = 63.72$, $PSNR = 23.97$, $SSIM = 0.47$; d) $MSE = 2547.30$, $PSNR = 14.07$, $SSIM = 0.21$.

Fonte: Elaborada pelo autor.

A Figura 23 ilustra os valores das três métricas aplicadas a um mesmo conjunto de imagens.

2.2.3 BM3D

O algoritmo Block-Matching and 3D Filtering (BM3D), proposto por (DABOV et al., 2006), é um marco no campo na redução de ruído em imagens. Sendo amplamente reconhecido como um método de referência devido ao seu elevado desempenho na redução de ruído e na preservação de detalhes (KONG; YANG; HE, 2020).

Além disso, o BM3D é frequentemente utilizado como base de comparação para novos algoritmos, e se destaca por sua robustez em diferentes casos de ruído. E embora alguns métodos modernos afirmem superar o BM3D, muitos foram avaliados em conjuntos limitados de dados, apresentando desempenho inferior em condições de ruído variadas, enquanto o BM3D mantém resultados semelhantes com métodos mais recentes (KONG; YANG; HE, 2020).

O BM3D introduziu um paradigma novo de filtragem colaborativa, baseado no agrupamento de blocos semelhantes (block matching) e no processamento de transformadas tridimensionais, por meio de janelas deslizantes (sliding window). Essa abordagem garante não apenas a remoção de ruído, mas também a preservação de detalhes da imagem (DABOV et al., 2006).

Dessa forma, o BM3D foi escolhido como uma das técnicas de comparação neste trabalho, tanto por sua relevância consolidada na literatura quanto pela disponibilidade em bibliotecas da linguagem Python.

2.3 Algoritmo fractal

De acordo com (FISHER, 1994), imagens do mundo real possuem autossimilaridade local, uma propriedade que as torna adequadas para técnicas de codificação fractal. Nesse contexto, Ghazel, Freeman e Vrscaj (2003) propõe o uso da codificação fractal como método para a remoção de ruído em imagens, partindo da observação de que, após o processo de codificação e decodificação, a imagem resultante apresenta menor quantidade de ruído. Isso ocorre porque a presença de ruído reduz a autossimilaridade existente na imagem, e, como a codificação fractal é um processo com perdas (*lossy*), ela tende a suprimir detalhes incoerentes como o ruído durante a reconstrução da imagem.

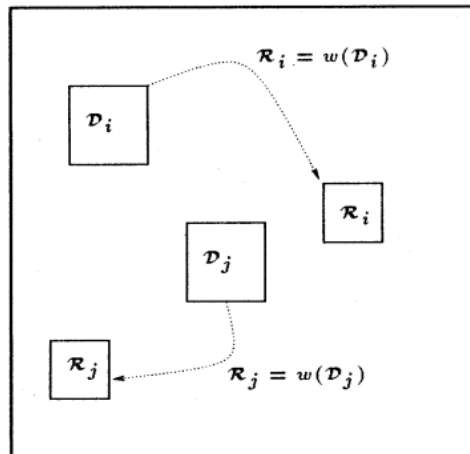
2.3.1 Codificação fractal

A codificação fractal apresentada na seção 2.1.2 é baseada na teoria do IFS, em que ocorre o problema inverso: ao invés de construir o fractal, deve-se determinar qual IFS gera um determinado atrator. Nesse contexto, uma imagem $X \in \mathbb{R}^2$ pode ser dividida em blocos de imagem (*range blocks*) não sobrepostos R_i e em um conjunto maior de blocos de domínio (*domain blocks*) $D_i \subset X$, de forma que $R_i = w_j(D_k)$, em que w_j é uma transformação afim do IFS.

Considerando que R_i e D_i sejam regiões quadradas, podem ser aplicadas as seguintes transformações: contração, rotação de 90° , 180° ou 270° , e reflexão horizontal, vertical ou diagonal, totalizando oito transformações geométricas possíveis.

A figura 24 exemplifica como ocorre essas transformações.

Figura 24 – Transformações do bloco de domínio D_i para o bloco de imagem R_i



Fonte: (GHAZEL; FREEMAN; VRSCAY, 2003).

também é possível ajustar a intensidade dos pixels por meio de uma função $\phi_i : \mathbb{R} \rightarrow \mathbb{R}$, definida como:

$$\phi(t) = \alpha t + \beta \quad (2.39)$$

onde t é a intensidade do pixel no bloco de imagem R_i , e α e β são parâmetros que ajustam o contraste e o brilho, definidos por:

$$\alpha = \frac{n \sum_{m=1}^n x_m y_m - (\sum_{m=1}^n x_m)(\sum_{m=1}^n y_m)}{n \sum_{m=1}^n x_m^2 - (\sum_{m=1}^n x_m)^2} \quad (2.40)$$

$$\beta = \frac{1}{n} \sum_{m=1}^n y_m - \alpha \frac{1}{n} \sum_{m=1}^n x_m \quad (2.41)$$

(GHAZEL; FREEMAN; VRSCAY, 2003).

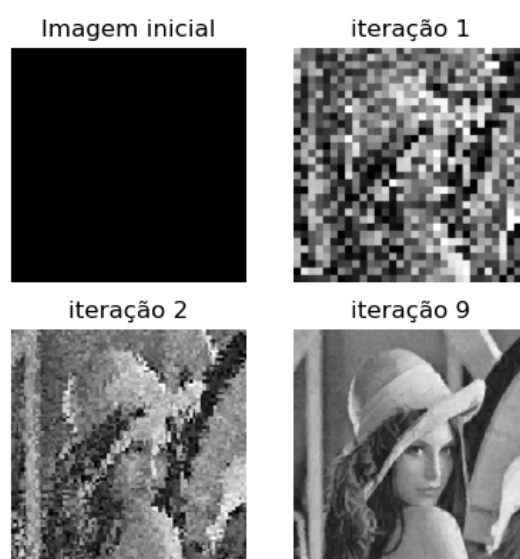
Dessa forma, o algoritmo de codificação fractal consiste nos seguintes passos:

1. Dividir a imagem original em blocos de imagem R_i e blocos de domínio D_i , sendo que cada bloco D_i possui o dobro do tamanho de R_i ;
2. Para cada bloco de intervalo R_i , buscar o bloco de domínio D_i que, ao se aplicar uma das oito possíveis transformações geométricas e uma transformação de brilho e contraste ϕ 2.40, minimize o MSE entre R_i e $w_j(D_i)$.

Embora a divisão uniforme em blocos quadrados seja o método mais simples, existem abordagens adaptativas, como o algoritmo *quadtree*, que divide os blocos de forma hierárquica de acordo com a quantidade de detalhes local da imagem. Dessa forma, regiões homogêneas são particionadas em blocos maiores, enquanto áreas com maior nível de detalhe são divididas em blocos menores.

Na etapa de decodificação, aplicam-se as transformações associadas a cada par (D_i, R_i) , conforme o processo iterativo descrito no algoritmo determinístico apresentado na seção 2.1.5.1. A figura 25 mostra iterações do processo de decodificação.

Figura 25 – Iterações da decodificação da Lena

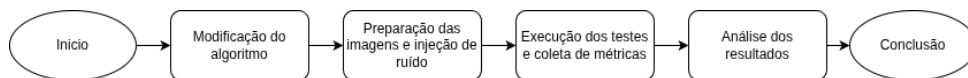


Fonte: Elaborada pelo autor.

3 Metodologia

Neste capítulo, serão apresentados os procedimentos metodológicos adotados para o desenvolvimento deste trabalho, a modificação proposta no algoritmo de codificação fractal o conjunto de imagens utilizados e o processo de testes. O planejamento metodológico deste trabalho segue as etapas apresentadas no fluxograma da Figura 26.

Figura 26 – Fluxograma do Planejamento



Fonte: Elaborada pelo autor.

3.1 Modificação no Algoritmo

O algoritmo proposto por Ghazel, Freeman e Vrscay (2003) utiliza o MSE como critério para determinar o melhor par formado por um bloco de domínio e sua respectiva transformação (D_i, w_i) para cada bloco de imagem R_i .

A modificação proposta neste trabalho consiste em substituir o MSE pela métrica SSIM. O objetivo dessa alteração é avaliar o impacto de uma métrica perceptualmente mais fiel à visão humana na qualidade final da imagem reconstruída. A parte do algoritmo que foi modificada está detalhada em 3.

Algoritmo 3: Algoritmo modificado

Data: Imagem com $X \in \mathbb{R}^2$, coordenadas do bloco de imagem (r_i, r_j) , tamanho do bloco de domínio $domain_size$, passo do domínio $domain_step$

Result: Transformação para (r_i, r_j)

$best_ssim \leftarrow -\infty$;

$best_transformation \leftarrow \emptyset$;

for $d_i \leftarrow 0$ **to** $h - domain_size$ **step** $domain_step$ **do**

for $d_j \leftarrow 0$ **to** $w - domain_size$ **step** $domain_step$ **do**

$D_k \leftarrow (d_i, d_j)$;

for $i \leftarrow 0$ **to** 7 **do**

$domain_transformation \leftarrow w_i(D_k)$;

$new_ssim \leftarrow ssim((r_i, r_j), domain_transformation)$;

if $new_ssim > best_ssim$ **then**

$best_transformation \leftarrow (D_k, w_i)$;

$best_ssim \leftarrow new_ssim$

return $best_transformation$;

3.2 Conjunto de Imagens

O conjunto de imagens utilizado para os testes foi obtido a partir da base de dados pública disponibilizada pelo *Signal and Image Processing Institute* (USC-SIPI), da University of Southern California (WEBER, 2018). Essa base contém imagens utilizadas em experimentos de processamento digital de imagens, com diferentes níveis de complexidade, texturas e contrastes. Esse conjunto de dados inclui tanto imagens em tons de cinza e coloridas e diferentes tamanhos de imagens. Foram utilizadas as 39 imagens presentes no volume 3.

Para padronizar o conjunto, todas as imagens foram convertidas para escala de cinza e redimensionadas para 512×512 pixels, utilizando a ferramenta ImageMagick (ImageMagick Studio LLC, 2025).

Os tipos de ruído aplicados às imagens durante os testes foram: ruído Gaussiano, ruído Sal e Pimenta, ruído Uniforme, ruído Gama, ruído Periódico e ruído Speckle, conforme descrito na Seção 2.2.1. Para aplicá-los, inicialmente foi realizada a normalização das imagens. Dessa forma, os parâmetros descritos posteriormente referem-se às imagens após a normalização.

3.3 Execução dos Testes e Métricas

As métricas utilizadas para a avaliação dos resultados, já apresentadas na Seção 2.2.2, são o MSE, a PSNR e o SSIM. Essas métricas foram selecionadas por serem amplamente utilizadas na literatura para quantificar a qualidade de imagens reconstruídas.

A execução dos testes ocorreu da seguinte forma: para cada imagem do conjunto de dados, foram aplicados quatro níveis diferentes de ruído, variando a intensidade a cada iteração. Em seguida, para cada par ruído e nível de ruído, os algoritmos foram aplicados e as métricas MSE, PSNR e SSIM foram calculadas e registradas.

Por fim, para cada métrica, foi calculada a média dos valores obtidos entre os diferentes níveis de ruído, conforme a seguinte expressão:

$$m = \frac{1}{n} \sum_{i=1}^n m_i \quad (3.1)$$

em que n representa o número total de imagens e m_i o valor da métrica correspondente na i -ésima execução.

Os parâmetros utilizados para a execução do algoritmo fractal e do algoritmo fractal modificado foram: tamanho do bloco de domínio = 32, tamanho do bloco de imagem = 16 e deslocamento da varredura = 32, pois apresentaram um desempenho adequado, aliado a um tempo de execução considerado viável. Idealmente, o deslocamento da varredura seria igual a 1, no entanto o tempo de execução se tornaria inviável, como será mostrado no capítulo 4.

4 Experimentação e Resultados

Neste capítulo, serão apresentados e analisados os resultados dos testes realizados, de acordo com as especificações do capítulo 3.

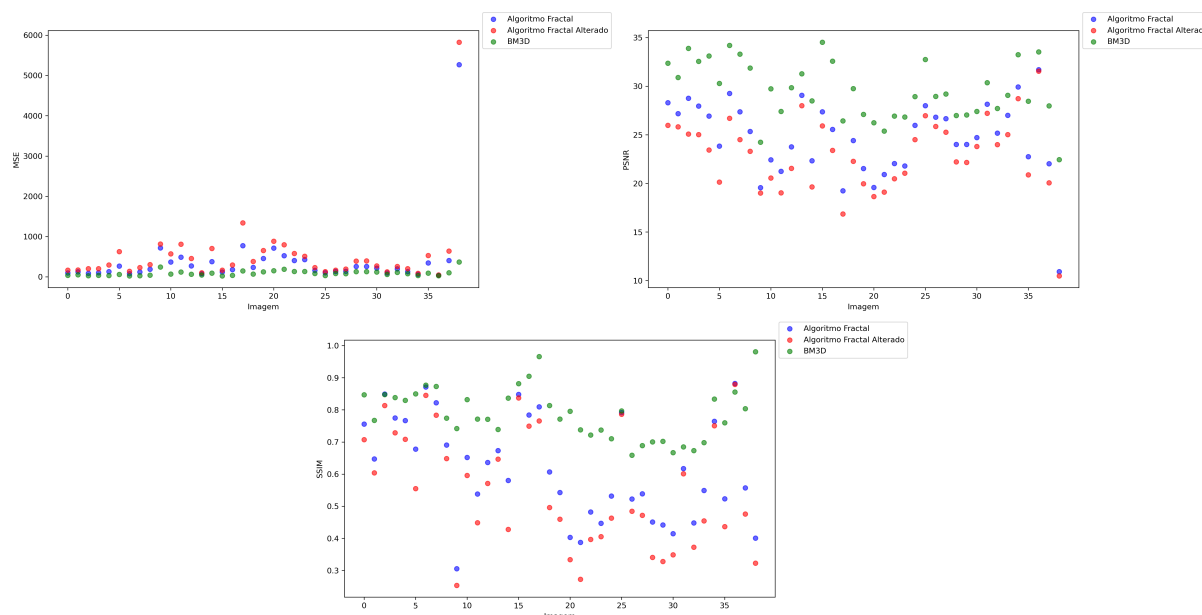
4.1 Resultados

Os parâmetros de ruído foram organizados em ordem crescente de intensidade, de modo a permitir a análise progressiva do comportamento dos algoritmos frente a diferentes níveis de degradação. Entretanto, como muitos dos resultados apresentaram comportamentos semelhantes, serão destacados apenas os mais relevantes nesta seção, enquanto os demais serão disponibilizados no Apêndice.

4.1.1 Resultados Ruído Gaussiano

Para o ruído Gaussiano, foram considerados quatro níveis de intensidade correspondentes aos valores $\sigma = 0.08$ 42, $\sigma = 0.12$ 27, $\sigma = 0.156$ 43 e $\sigma = 0.20$ 44. Os resultados obtidos para as métricas MSE, PSNR e SSIM são apresentados nas figuras seguintes.

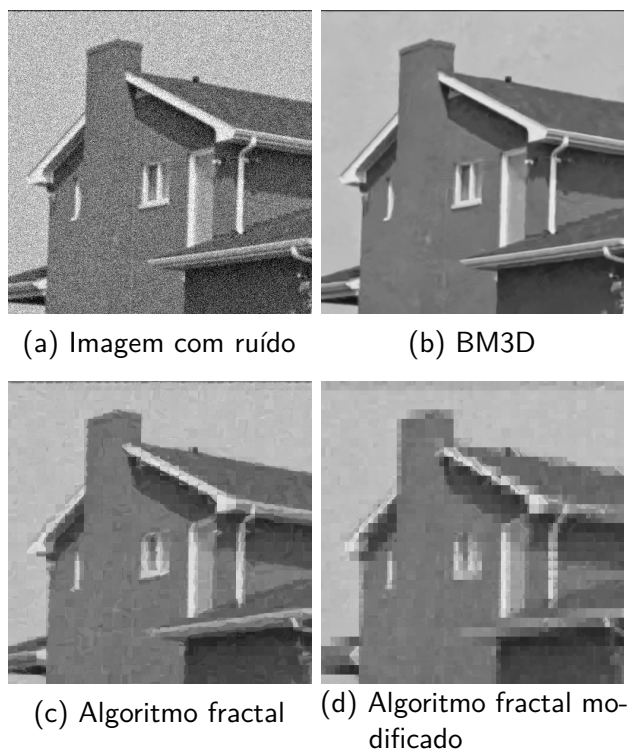
Figura 27 – Resultados das métricas MSE, PSNR e SSIM para ruído Gaussiano com $\sigma = 0.12$.



Fonte: Elaborado pelo autor.

A figura 28 mostra um exemplo dos algoritmos de restauração aplicados em uma imagem com ruído gaussiano.

Figura 28 – Exemplo entre métodos de remoção de ruído.



Fonte: Elaborado pelo autor.

A tabela 2 apresenta a comparação das médias das métricas MSE, PSNR e SSIM para os três algoritmos testados sobre imagens afetadas pelo ruído Gaussiano.

Tabela 2 – Comparação das medias para o ruído Gaussiano.

Parâmetro	Algoritmo Fractal			Algoritmo Fractal Modificado			BM3D		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
$\sigma = 0.08$	365.425	25.2033	0.641585	506.711	23.2698	0.570362	63.1261	31.8643	0.841708
$\sigma = 0.12$	389.448	24.7106	0.615056	536.421	22.9289	0.553044	90.4337	29.5731	0.788233
$\sigma = 0.156$	417.167	24.1848	0.591438	558.667	22.627	0.539987	110.508	28.9191	0.781444

Fonte: Elaborado pelo autor.

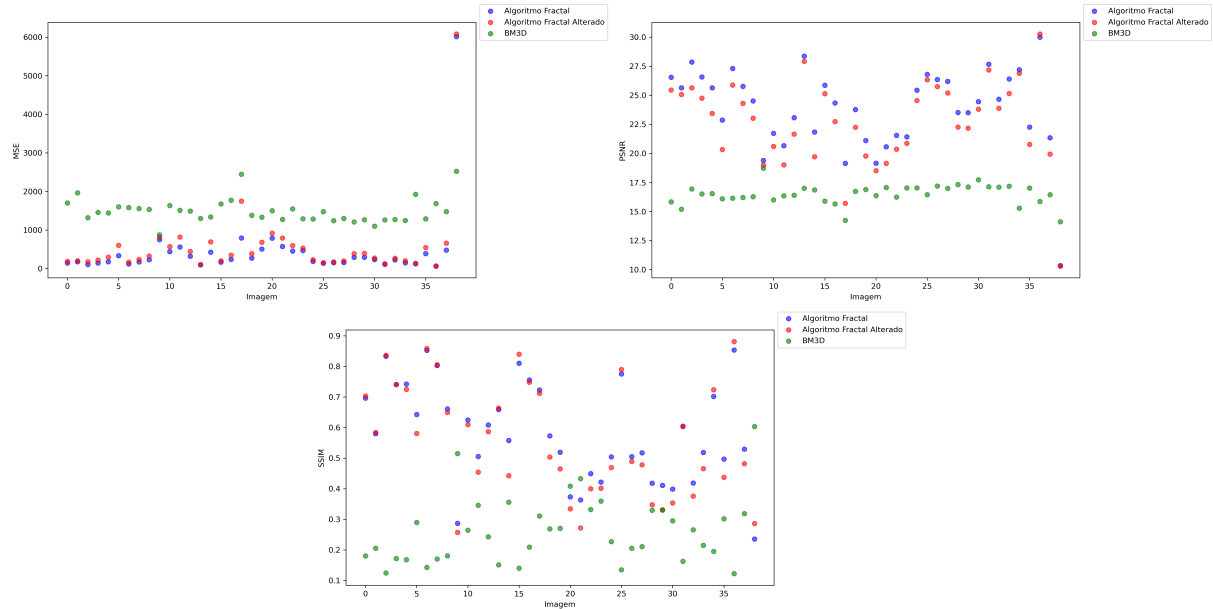
Observa-se que o algoritmo BM3D apresentou o melhor resultado em todos os testes. Além disso, a modificação proposta no algoritmo fractal apresentou impacto pequeno ou negativo em relação ao método original. A figura 27 também evidencia que o algoritmo fractal apresentou desempenho muito ruim na imagem 39. Dessa forma, a substituição da métrica MSE pelo SSIM não favoreceu o processo de restauração da imagem para o ruído Gaussiano.

4.1.2 Resultados Ruído Sal e Pimenta

Para avaliar o comportamento dos algoritmos diante do ruído Sal e Pimenta foram testados quatro níveis de intensidade diferentes. Os resultados obtidos para as métricas MSE,

PSNR e SSIM são apresentados nas figuras 45, 29, 46 e 47.

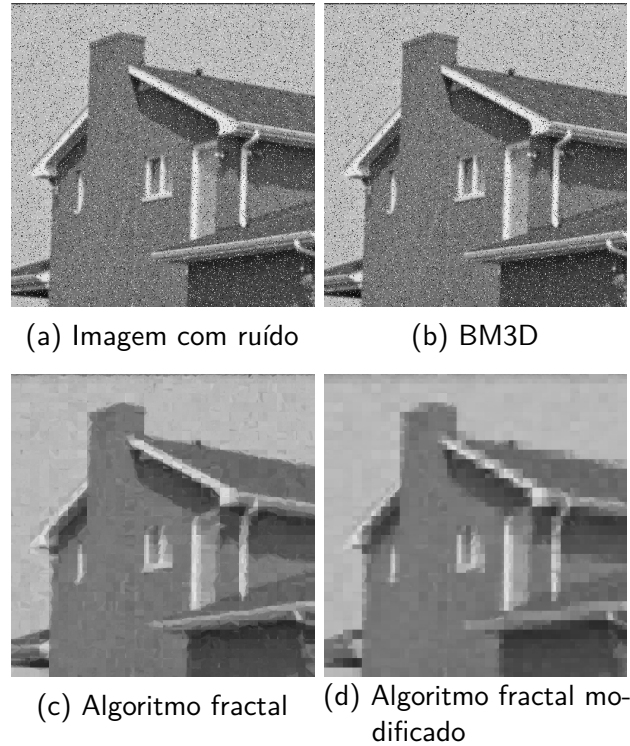
Figura 29 – Resultados das métricas MSE, PSNR e SSIM para ruído Sal e Pimenta com $p_1 = 0.04$ e $p_2 = 0.04$.



Fonte: Elaborado pelo autor.

A figura 30 mostra um exemplo dos algoritmos de restauração aplicados em uma imagem com ruído sal e pimenta.

Figura 30 – Exemplo entre métodos de remoção de ruído.



Fonte: Elaborado pelo autor.

A tabela 3 apresenta a comparação das médias das métricas MSE, PSNR e SSIM para os três algoritmos testados sobre imagens afetadas pelo ruído Sal e Pimenta.

Tabela 3 – Comparação das medias para o ruído Sal e Pimenta.

Parâmetro	Algoritmo Fractal			Algoritmo Fractal Modificado			BM3D		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
$p_1 = 0.02$ $p_2 = 0.02$	385.001	24.7183	0.614905	517.568	23.2646	0.573223	546.89	23.6124	0.595644
$p_1 = 0.04$ $p_2 = 0.04$	447.273	23.8669	0.581256	562.23	22.6868	0.556031	1488.92	16.4887	0.260426
$p_1 = 0.08$ $p_2 = 0.08$	562.763	22.4076	0.536197	656.29	21.6021	0.521546	2645.74	14.1355	0.160226
$p_1 = 0.16$ $p_2 = 0.16$	829.993	20.2845	0.487674	904.38	19.9192	0.482821	1563.6	20.0311	0.407954

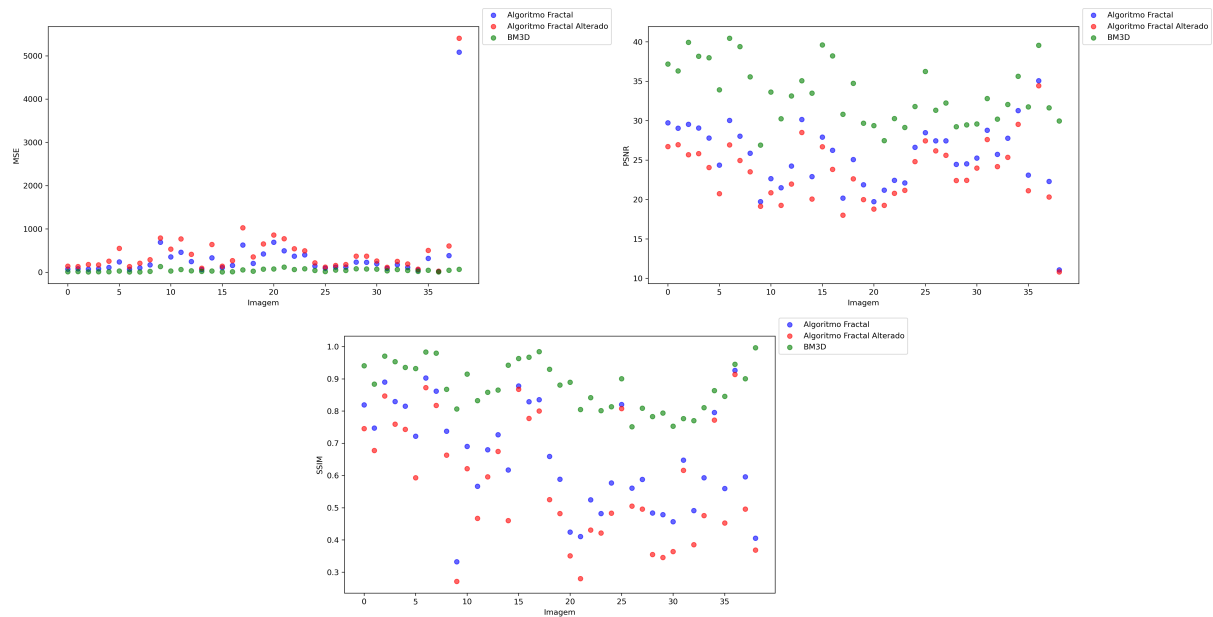
Fonte: Elaborado pelo autor.

Observa-se que para o ruído Sal e Pimenta, o algoritmo BM3D apresentou desempenho inferior em comparação aos métodos fractais. O algoritmo fractal original obteve os melhores resultados, especialmente em níveis intermediários de ruído, como nos parâmetros $p_1 = 0.4$, $p_2 = 0.4$ e $p_1 = 0.8$, $p_2 = 0.8$. Entretanto, a modificação proposta no algoritmo fractal resultou em uma leve perda de desempenho, portanto a substituição do MSE pelo SSIM não favoreceu a etapa de restaração nesse tipo de ruído.

4.1.3 Resultados Ruído Uniforme

Para avaliar o comportamento dos algoritmos diante do ruído Uniforme foram testados quatro níveis de intensidade diferentes. Os resultados obtidos para as métricas MSE, PSNR e SSIM são apresentados nas figuras 48, 31, 49 e 50.

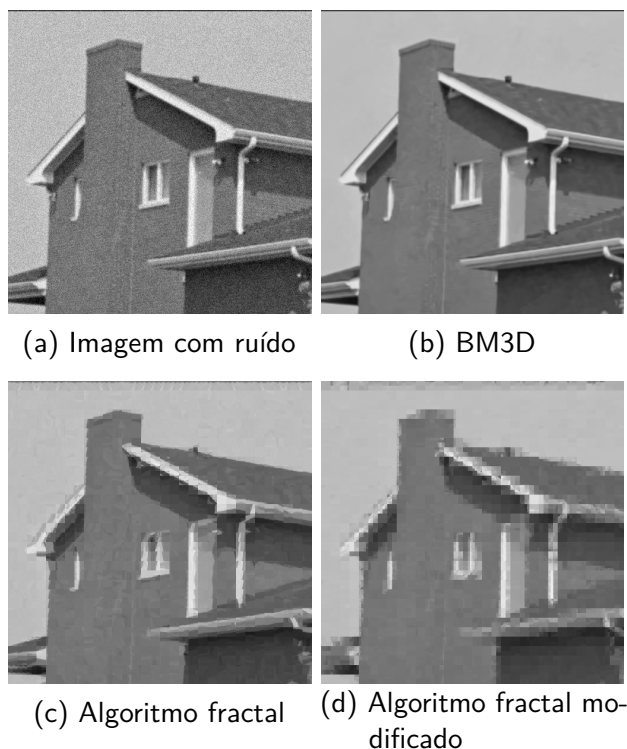
Figura 31 – resultados das métricas MSE, PSNR e SSIM para ruído Uniforme com $a = 0.1$ e $b = 0.1$.



Fonte: Elaborado pelo autor.

A figura 32 mostra um exemplo dos algoritmos de restauração aplicados em uma imagem com ruído uniforme.

Figura 32 – Exemplo entre métodos de remoção de ruído.



Fonte: Elaborado pelo autor.

A tabela 4 apresenta a comparação das médias das métricas MSE, PSNR e SSIM para os três algoritmos testados sobre imagens afetadas pelo ruído Uniforme.

Tabela 4 – Comparação das medias para ruído Uniforme.

Parâmetro	Algoritmo Fractal			Algoritmo Fractal Modificado			BM3D		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
$a = 0.05$ $b = 0.05$	350.069	25.6121	0.6706	474.613	23.6318	0.590128	22.5594	36.1787	0.920236
$a = 0.10$ $b = 0.10$	357.826	25.398	0.655072	493.473	23.3947	0.578982	40.8846	33.4391	0.877954
$a = 0.15$ $b = 0.15$	370.588	25.114	0.636836	510.028	23.1908	0.566864	60.532	31.6962	0.842569
$a = 0.20$ $b = 0.20$	386.76	24.732	0.616733	540.917	22.93	0.552967	80.926	30.3197	0.811921

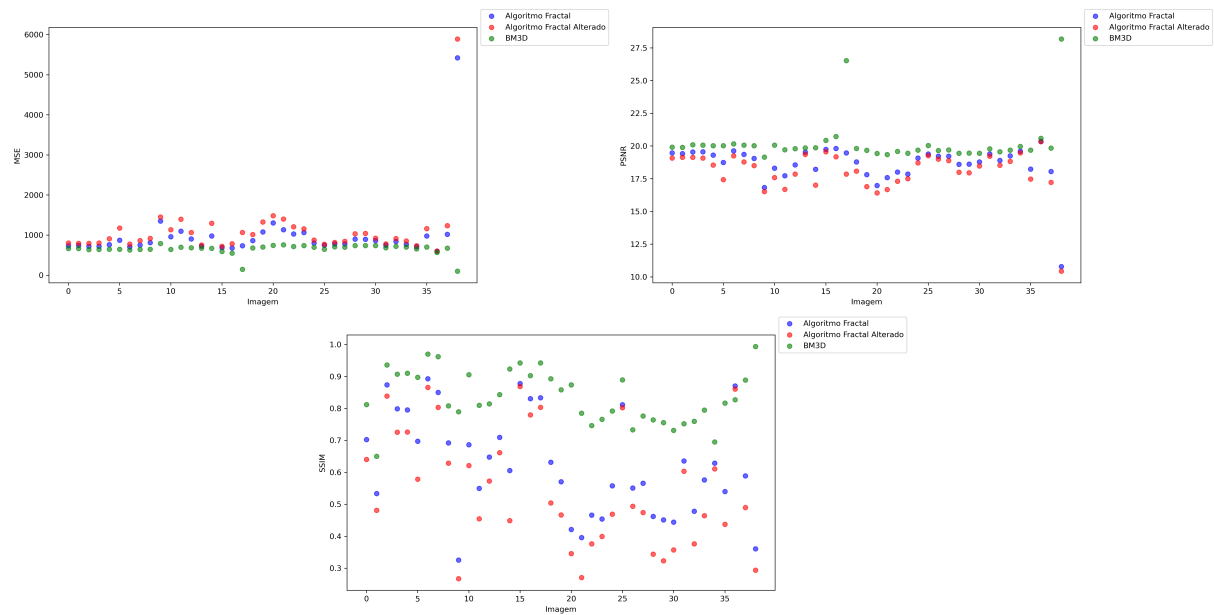
Fonte: Elaborado pelo autor.

Observa-se que para o ruído uniforme, o algoritmo BM3D apresentou o melhor desempenho entre todos os métodos avaliados, com resultados consistentes mesmo com o aumento da intensidade do ruído. De forma semelhante ao observado em outros tipos de ruído, apesar de que a modificação proposta no algoritmo fractal resultou em uma leve perda de desempenho, indicando que a substituição do MSE pelo SSIM não contribuiu para melhorar a restauração da imagem.

4.1.4 Resultados Ruído Gama

Para analisar o comportamento dos algoritmos diante do ruído Gama foram usados quatro níveis de intensidade diferentes. Os resultados obtidos para as métricas MSE, PSNR e SSIM são apresentados nas figuras 51, 33, 52 e 53.

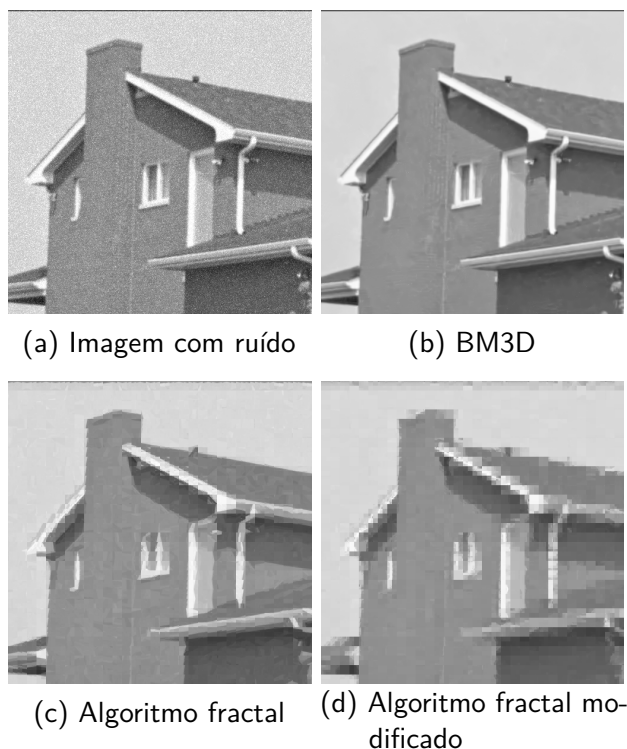
Figura 33 – resultados das métricas MSE, PSNR e SSIM para ruído gama com $a = 3$ e $b = 0.035$.



Fonte: Elaborado pelo autor.

A figura 34 mostra um exemplo dos algoritmos de restauração aplicados em uma imagem com ruído gama.

Figura 34 – Exemplo entre métodos de remoção de ruído.



Fonte: Elaborado pelo autor.

A tabela seguinte apresenta a comparação das médias das métricas MSE, PSNR e SSIM para os três algoritmos testados sobre imagens afetadas pelo ruído Gama.

Tabela 5 – Comparação das medias para ruído Gama.

Parâmetro	Algoritmo Fractal			Algoritmo Fractal Modificado			BM3D		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
$a = 2$ $b = 0.035$	619.68	21.075	0.642692	760.25	20.1936	0.566172	295.149	23.6369	0.868233
$a = 3$ $b = 0.035$	981.245	18.6341	0.624626	1116.98	18.0806	0.551897	650.132	20.2119	0.836146
$a = 3$ $b = 0.045$	650.132	20.2119	0.836146	1510.93	16.5506	0.539969	1057.18	18.1002	0.797313
$a = 4$ $b = 0.065$	3856.53	12.3842	0.546195	3989.96	12.2203	0.494841	3586.51	12.8316	0.663677

Fonte: Elaborado pelo autor.

Observa-se que para o ruído Gama, o algoritmo BM3D apresentou o melhor desempenho entre todos os métodos avaliados. Entretanto, à medida que a intensidade do ruído aumenta, todos os métodos passam a apresentar resultados significativamente piores. Além disso, a substituição do cálculo da diferença entre blocos, originalmente com MSE pelo SSIM no algoritmo fractal resultou na perda de desempenho na restauração da imagem.

4.1.5 Resultados Ruído Periódico

Para analisar o comportamento dos algoritmos diante do ruído periódico, foram utilizadas quatro funções diferentes, definidas como:

$$f_1(x, y) = 0.125 \sin(10\pi(0.5x - y)),$$

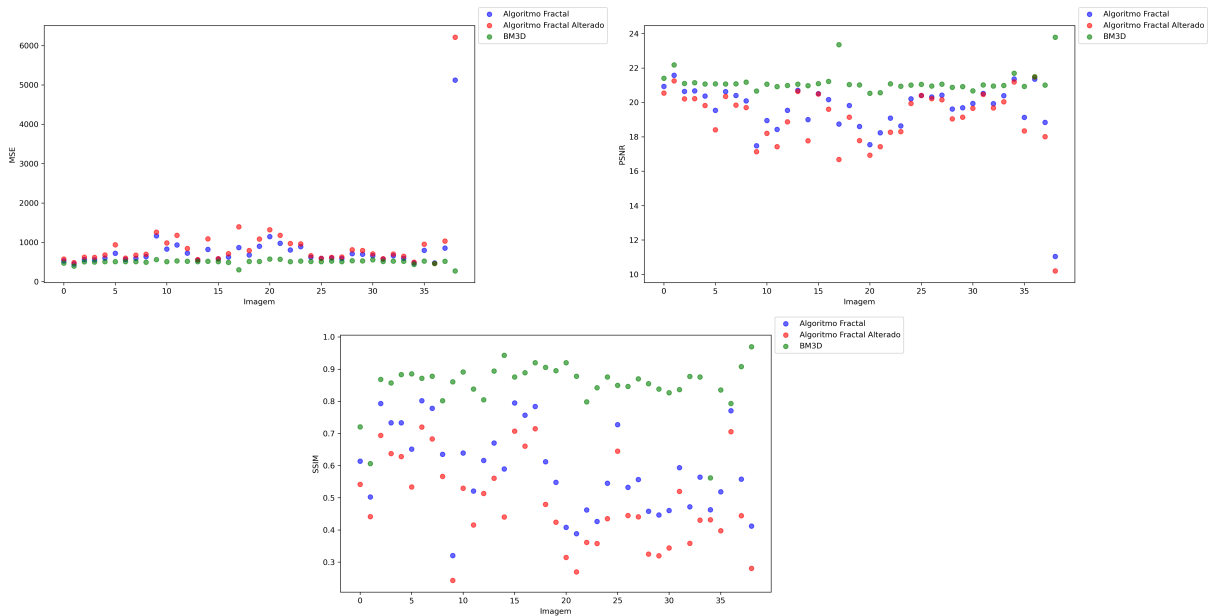
$$f_2(x, y) = 0.125 \sin(20\pi(0.5x - y)) + 0.0625 \sin(50\pi(x - y)),$$

$$f_3(x, y) = 0.125 \sin(20\pi(0.5x - y)) + 0.125 \sin(50\pi(x - y)),$$

$$f_4(x, y) = 0.15 \sin(20\pi(0.5x - y)) + 0.15 \sin(50\pi(x - y)) + 0.05 \sin(80\pi(x - y)).$$

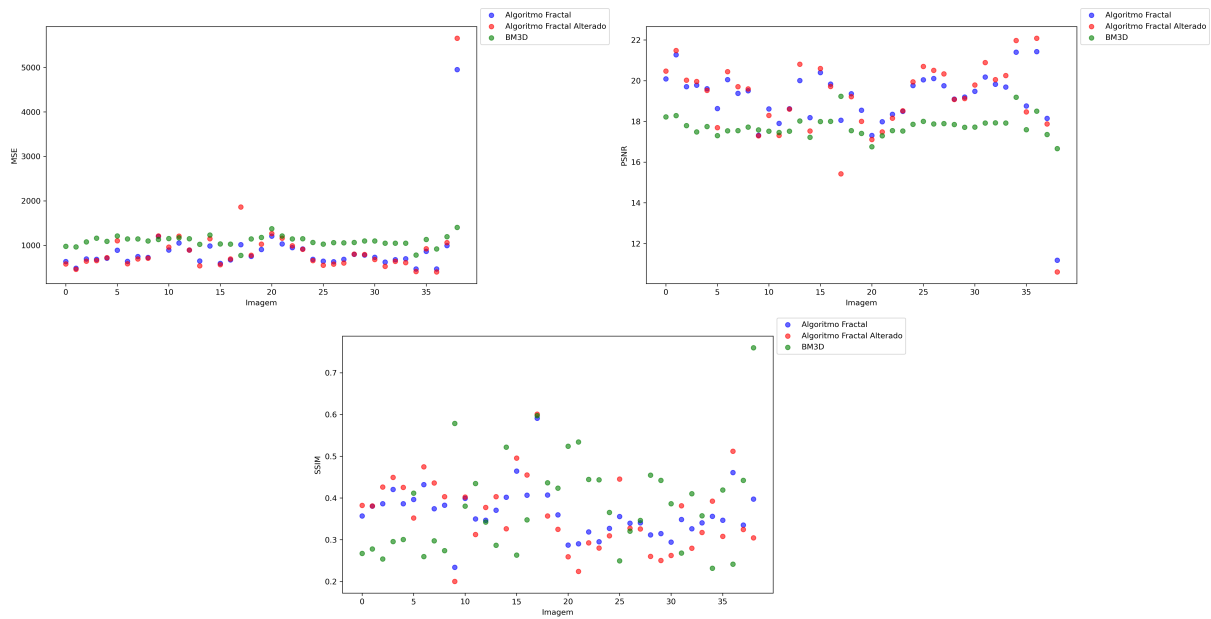
Os resultados obtidos para as métricas MSE, PSNR e SSIM são apresentados nas figuras 35, 54, 36 e 55 respectivamente.

Figura 35 – resultados das métricas MSE, PSNR e SSIM para ruído Periódico com $f_1(x, y)$.



Fonte: Elaborado pelo autor.

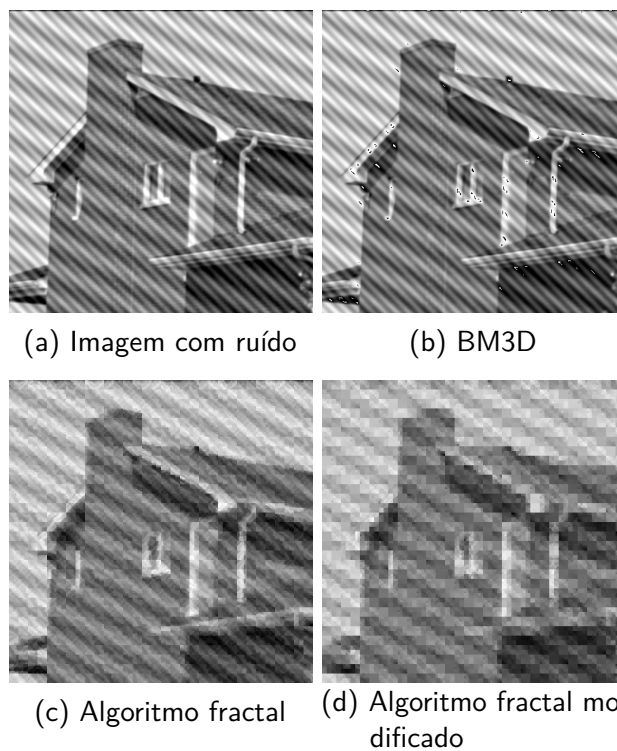
Figura 36 – resultados das métricas MSE, PSNR e SSIM para ruído Periódico com $f_3(x, y)$.



Fonte: Elaborado pelo autor.

A figura 37 mostra um exemplo dos algoritmos de restauração aplicados em uma imagem com ruído periódico.

Figura 37 – Exemplo entre métodos de remoção de ruído.



Fonte: Elaborado pelo autor.

A tabela 6 apresenta a comparação das médias das métricas MSE, PSNR e SSIM para os três algoritmos testados sobre imagens afetadas pelo ruído Periódico.

Tabela 6 – Comparação das medias para ruído Periódico.

Parâmetro	Algoritmo Fractal			Algoritmo Fractal Modificado			BM3D		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
f_1	813.393	19.5738	0.586121	940.465	19.0489	0.486254	499.657	21.1829	0.849769
f_2	499.657	21.1829	0.849769	882.222	19.3575	0.395882	639.63	20.1072	0.60429
f_3	889.779	19.1027	0.365118	931.689	19.0925	0.360213	1098.56	17.7495	0.381946
f_4	1114.73	17.9943	0.309477	1119.93	18.1714	0.313372	1701.76	15.8425	0.309921

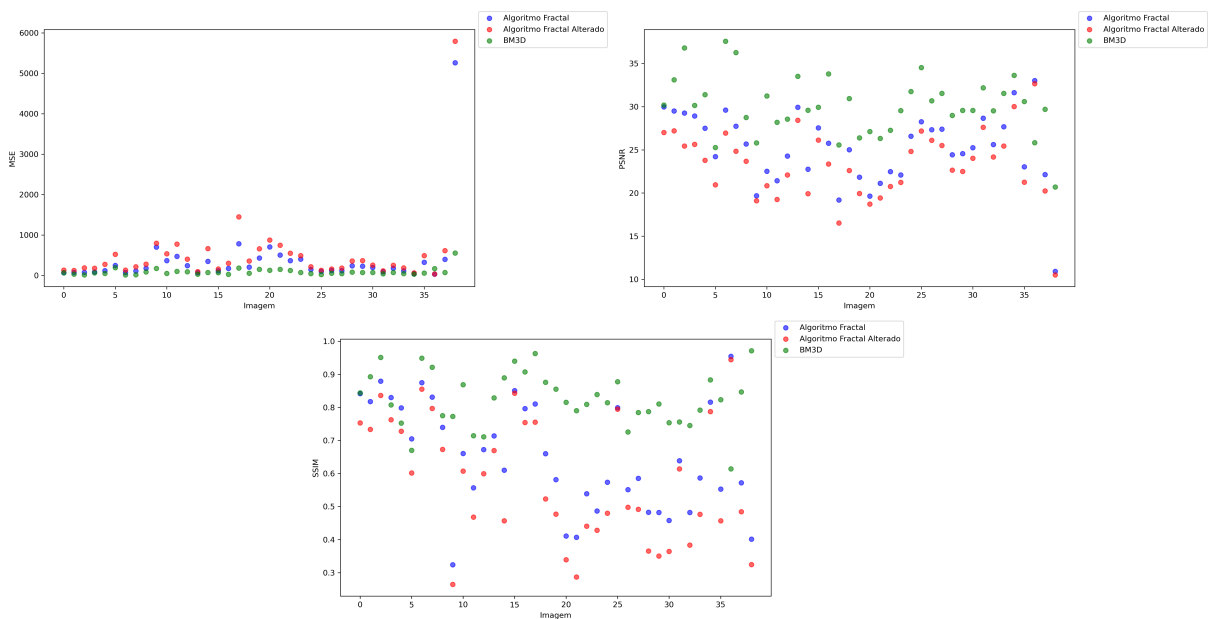
Fonte: Elaborado pelo autor.

Observa-se que para o ruído periódico, o algoritmo BM3D apresentou um resultado superior apenas quando o ruído foi gerado pela função $f_1(x, y)$. Para as demais funções, os métodos fractais obtiveram melhor desempenho. Esse resultado se deve ao fato de que o método fractal conseguiu identificar com maior facilidade as transformações presentes nesses tipos de ruído. Da mesma forma, a substituição do cálculo da diferença entre blocos, do MSE para o SSIM, no algoritmo fractal resultou em perda de desempenho.

4.1.6 Resultados Ruído *Speckle*

Para analisar o comportamento dos algoritmos diante do ruído *Speckle*, foram utilizados quatro níveis diferentes de intensidade. Os resultados obtidos para as métricas MSE, PSNR e SSIM são apresentados nas figuras 56, 38, 57 e 58.

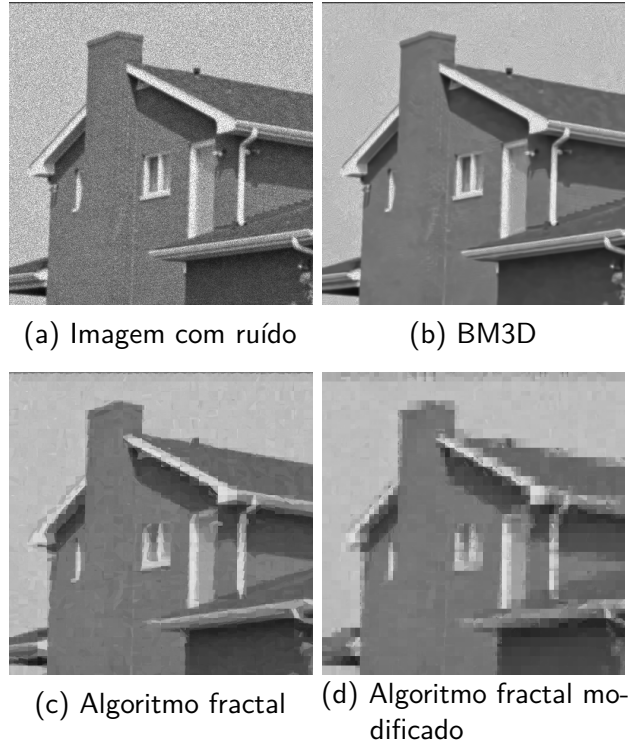
Figura 38 – resultados das métricas MSE, PSNR e SSIM para ruído uniforme com $a = 50$ e $\alpha = \frac{1}{50}$.



Fonte: Elaborado pelo autor.

A figura 39 mostra um exemplo dos algoritmos de restauração aplicados em uma imagem com ruído speckle.

Figura 39 – Exemplo entre métodos de remoção de ruído.



Fonte: Elaborado pelo autor.

A tabela 7 apresenta a comparação das médias das métricas MSE, PSNR e SSIM para os três algoritmos testados sobre imagens afetadas pelo ruído *Speckle*.

Tabela 7 – Comparação das medias para ruído *Speckle*.

Parâmetro	Algoritmo Fractal			Algoritmo Fractal Modificado			BM3D		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
$a = 100$ $\alpha = \frac{1}{100}$	359.556	25.4495	0.661162	496.453	23.4595	0.584105	58.1168	31.9935	0.866421
$a = 50$ $\alpha = \frac{1}{50}$	370.583	25.2353	0.649549	514.775	23.2937	0.576113	87.4786	30.0937	0.823797
$a = 25$ $\alpha = \frac{1}{25}$	398.265	24.8327	0.631221	547.942	23.018	0.565492	133.386	28.1848	0.773359
$a = 10$ $\alpha = \frac{1}{10}$	483.133	23.8591	0.596692	617.363	22.4317	0.547454	239.591	25.5638	0.700682

Fonte: Elaborado pelo autor.

Observa-se que, para o ruído *Speckle*, o algoritmo BM3D apresentou desempenho superior em todos os diferentes parâmetros de ruído. No entanto, é possível identificar um outlier, em que o algoritmo fractal se sobressaiu. Esse comportamento ocorreu porque se trata de uma imagem em que é relativamente fácil encontrar uma transformação capaz de gerar

o atrator de um bloco de imagem. Além disso, a substituição do cálculo da diferença entre blocos do MSE para o SSIM, no algoritmo fractal resultou em perda de desempenho.

4.1.7 Comparação de tempo

O tempo de execução será comparado nesta seção. No entanto, a implementação do algoritmo BM3D não foi incluída, uma vez que foi utilizada a biblioteca "bm3d" em Python. Essa escolha impede a mensuração direta do tempo de processamento interno do método, pois a função é executada de forma encapsulada, sem acesso às etapas intermediárias do algoritmo. Assim, a análise de desempenho em tempo de execução foi restrita apenas às implementações desenvolvidas neste trabalho.

O processador usado para esse teste foi um Intel Core i5-9400F.

Tabela 8 – Tempo de execução dos algoritmos fractais.

Parâmetro	Algoritmo Fractal	Algoritmo Fractal Modificado
	Tempo	Tempo
$N = 32, M = 16, step = 32$	24.398s	1m3.016s
$N = 32, M = 16, step = 16$	1m26.195s	3m56.143s

Fonte: Elaborado pelo autor.

onde N é o tamanho do bloco de domínio, M é o tamanho do bloco de imagem e $step$ é o deslocamento utilizado na varredura.

Pela tabela, pode-se concluir que o tempo de execução do algoritmo fractal modificado é relativamente maior. Isso ocorre porque o SSIM é uma métrica baseada nos valores de um conjunto de pixels, o que aumenta o número de operações em função do tamanho de M . Além disso, é interessante notar que, conforme o $step$ é reduzido pela metade, o tempo de execução aproximadamente quadruplica, o que se explica pelo fato de a imagem testada possuir formato quadrado.

4.2 Discussão

De maneira geral, os resultados mostram que ambos os algoritmos fractais são inferiores ao BM3D, exceto em casos muito específicos, nos quais é relativamente fácil encontrar uma transformação afim que tenha como atrator o bloco de imagem. Da mesma forma, observa-se que o algoritmo fractal original apresenta desempenho superior ao algoritmo modificado, não apenas em eficiência na restauração, mas também em tempo de execução.

Além disso, é importante notar que a limitação do algoritmo fractal está relacionada ao problema inverso à construção fractal por IFS, não existe uma maneira simples de determinar a transformação do atrator, exceto por meio de um algoritmo de força bruta. Isso resulta,

provavelmente, em tempos de execução superiores aos do BM3D, mesmo que ambos os métodos fossem implementados na mesma plataforma.

Dessa forma, caso venha a existir um método rápido para resolver o problema da inversa, é esperado que o algoritmo fractal apresente desempenho significativamente melhor.

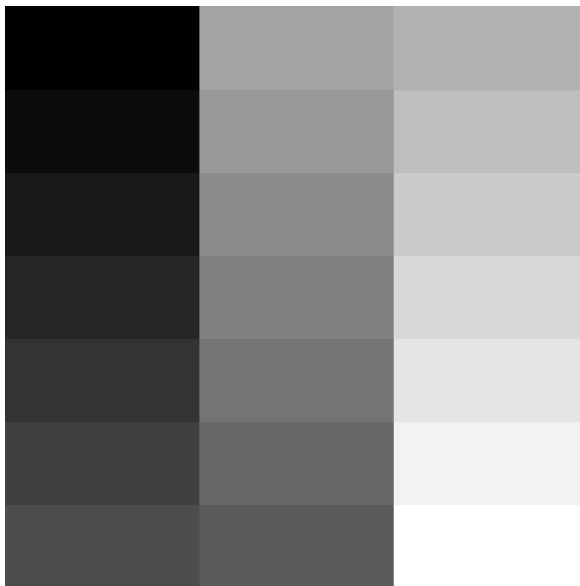
5 Conclusão

A partir do exposto pela presente monografia é possível concluir que a modificação na métrica do MSE para o SSIM entre os blocos no algoritmo fractal apresenta uma piora na qualidade da restauração da imagem e no tempo de processamento.

Além disso, o algoritmo *Block Matching and 3d Filtering* (BM3D) é superior significativamente aos algoritmos fractais na maioria das ocasiões, entretanto a diferença pode ser reduzida caso haja uma forma de resolver o problema de determinar um IFS mais eficiente.

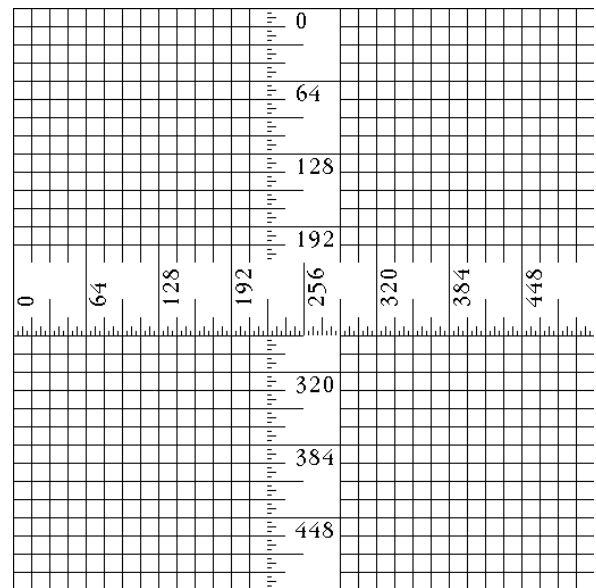
Isso ocorre porque a imagem 40 apresentou o melhor resultado, enquanto a imagem 41 apresentou o pior. Essa diferença está diretamente relacionada às escolhas dos parâmetros descritos no Capítulo 3, que podem facilitar ou dificultar a identificação do melhor IFS pelo algoritmo.

Figura 40 – Melhor resultado do algoritmo fractal.



Fonte: (WEBER, 2018)

Figura 41 – Pior resultado do algoritmo fractal.



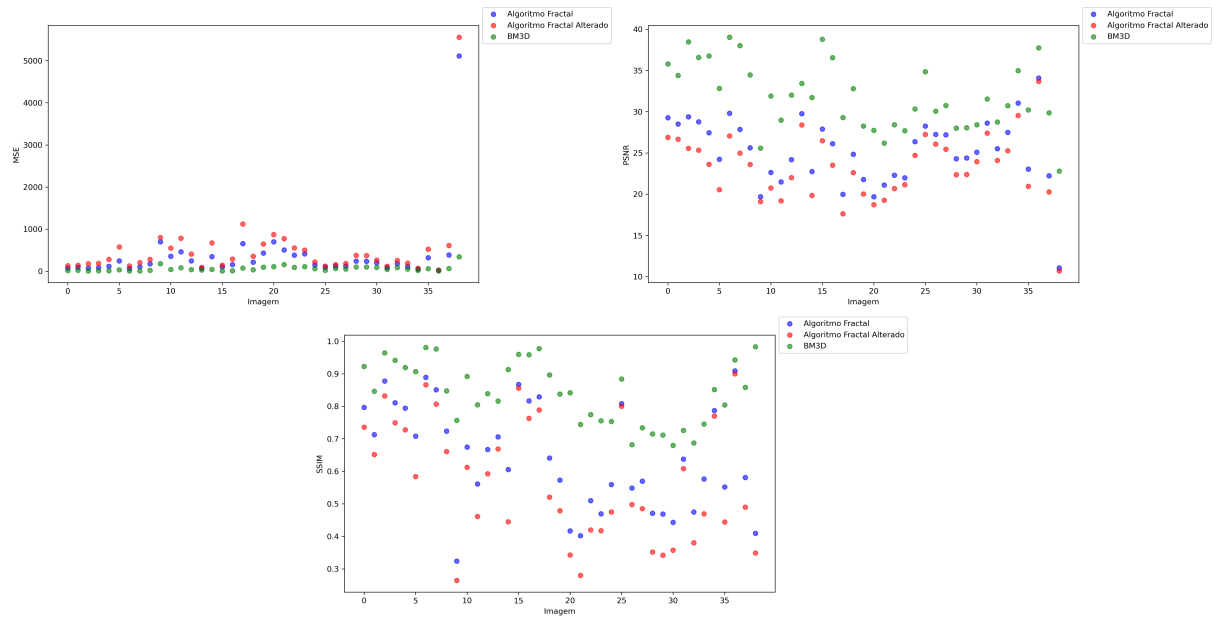
Fonte: (WEBER, 2018)

Dessa forma, possíveis trabalhos futuros são: explorar variações na divisão de blocos de domínio e imagem; analisar o desempenho em abordagens mais recentes que utilizam aprendizado de máquina.

A Resultados Complementares

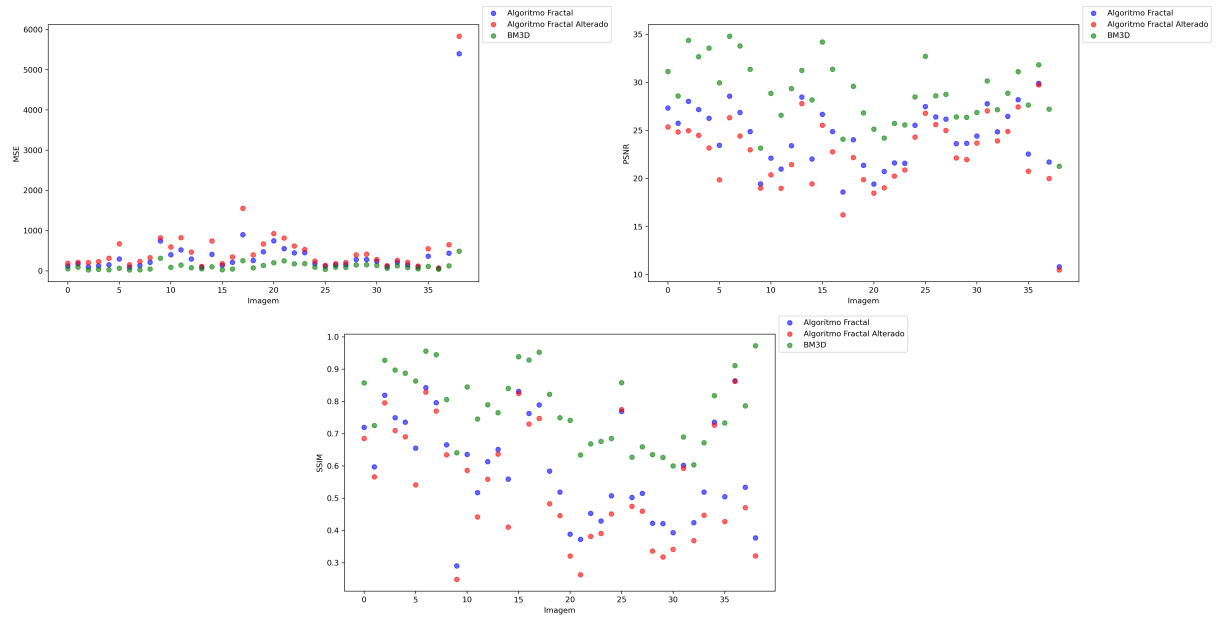
A.1 Ruído Gaussiano

Figura 42 – Resultados das métricas MSE, PSNR e SSIM para ruído Gaussiano com $\sigma = 0.08$.



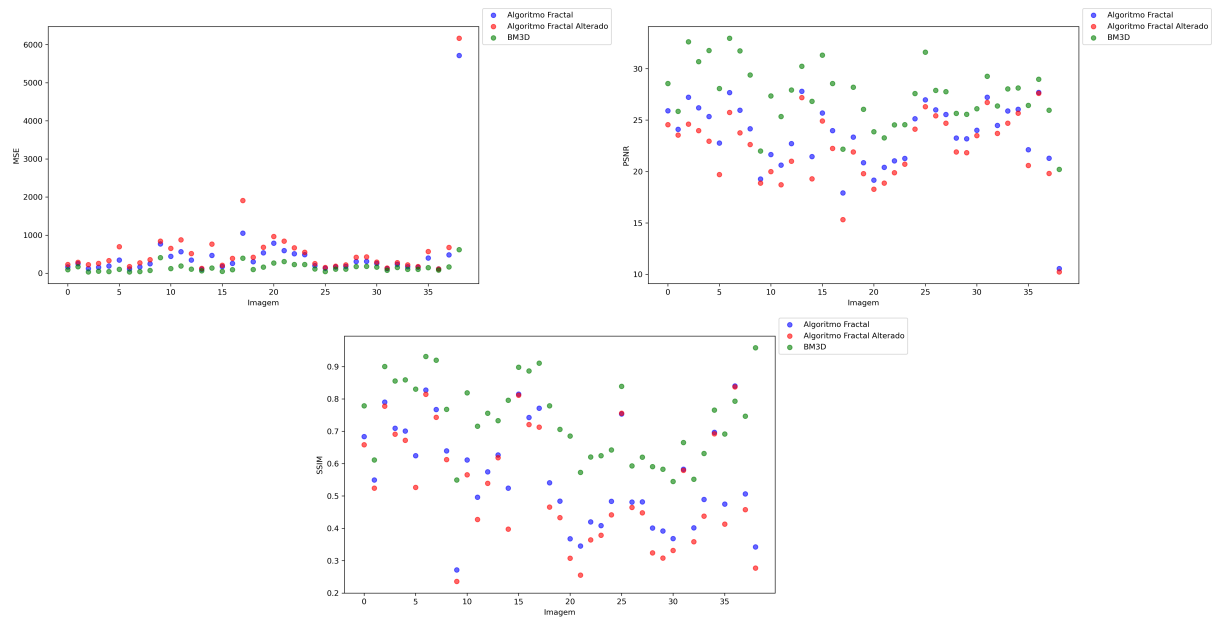
Fonte: Elaborado pelo autor.

Figura 43 – Resultados das métricas MSE, PSNR e SSIM para ruído Gaussiano com $\sigma = 0.156$.



Fonte: Elaborado pelo autor.

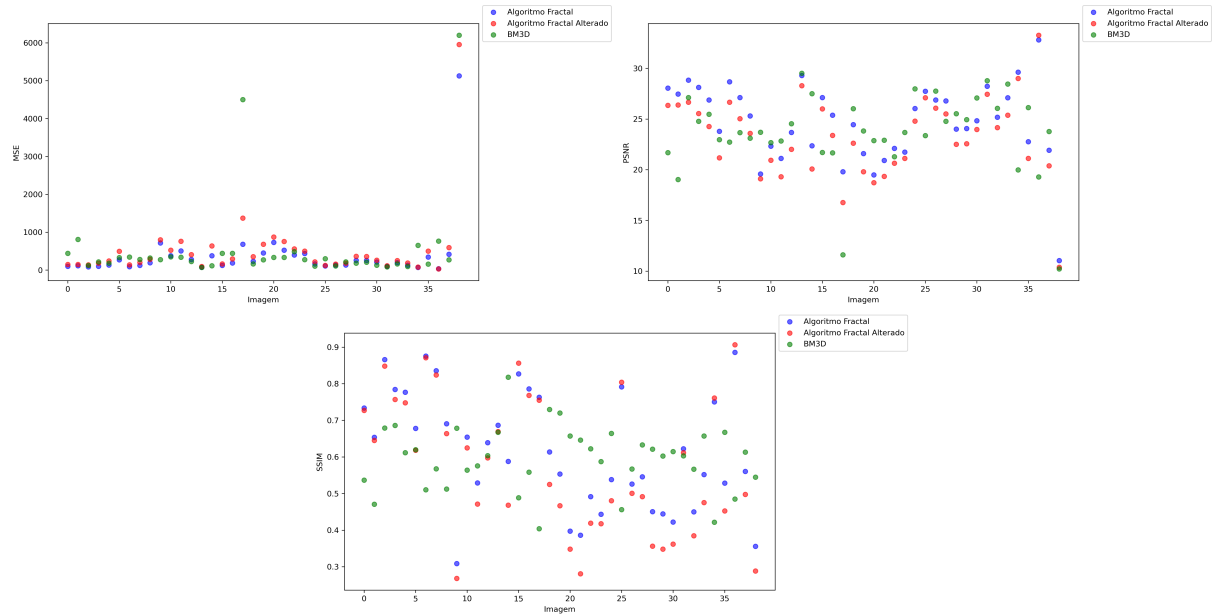
Figura 44 – Resultados das métricas MSE, PSNR e SSIM para ruído Gaussiano com $\sigma = 0.20$.



Fonte: Elaborado pelo autor.

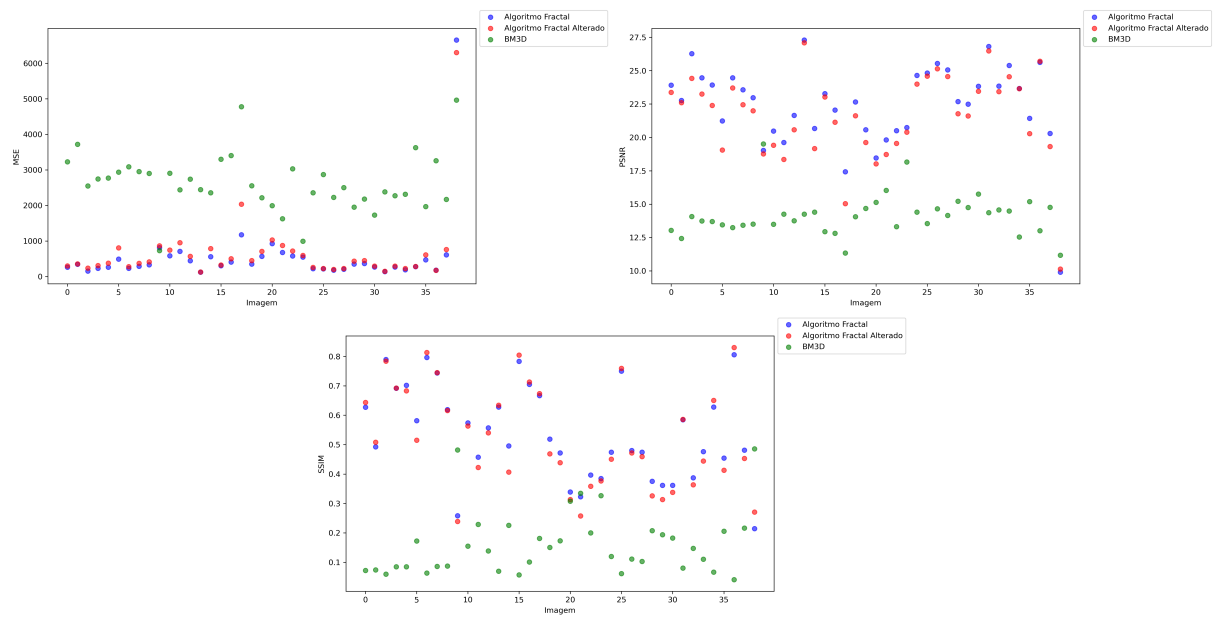
A.2 Ruído Sal e Pimenta

Figura 45 – Resultados das métricas MSE, PSNR e SSIM para ruído Sal e Pimenta com $p_1 = 0.02$ e $p_2 = 0.02$.



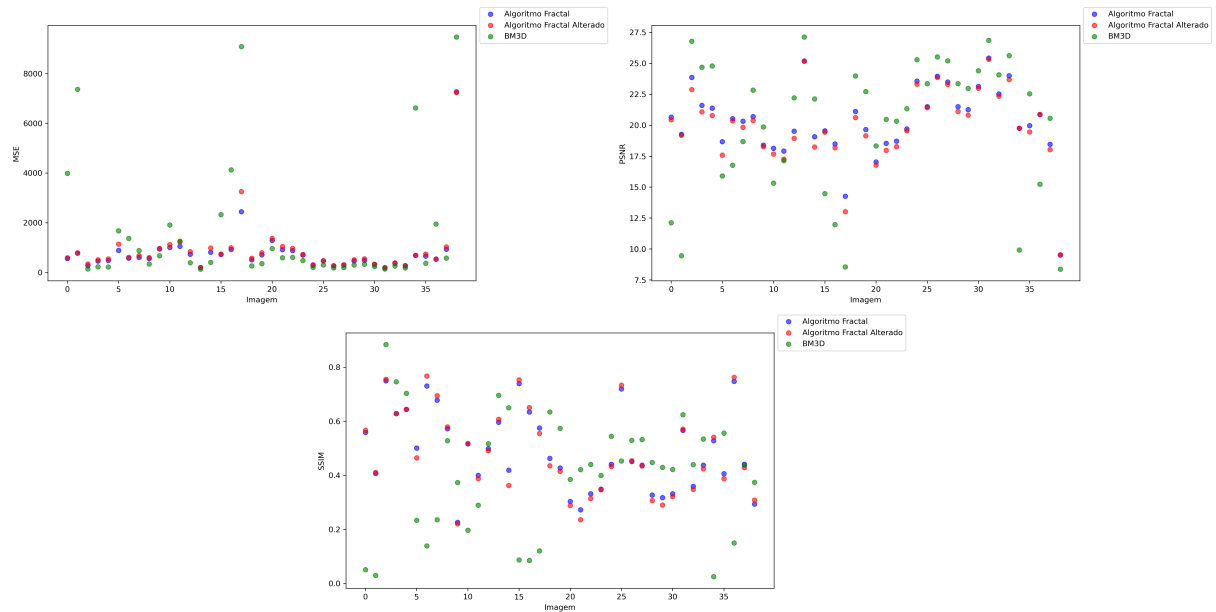
Fonte: Elaborado pelo autor.

Figura 46 – Resultados das métricas MSE, PSNR e SSIM para ruído Sal e Pimenta com $p_1 = 0.08$ e $p_2 = 0.08$.



Fonte: Elaborado pelo autor.

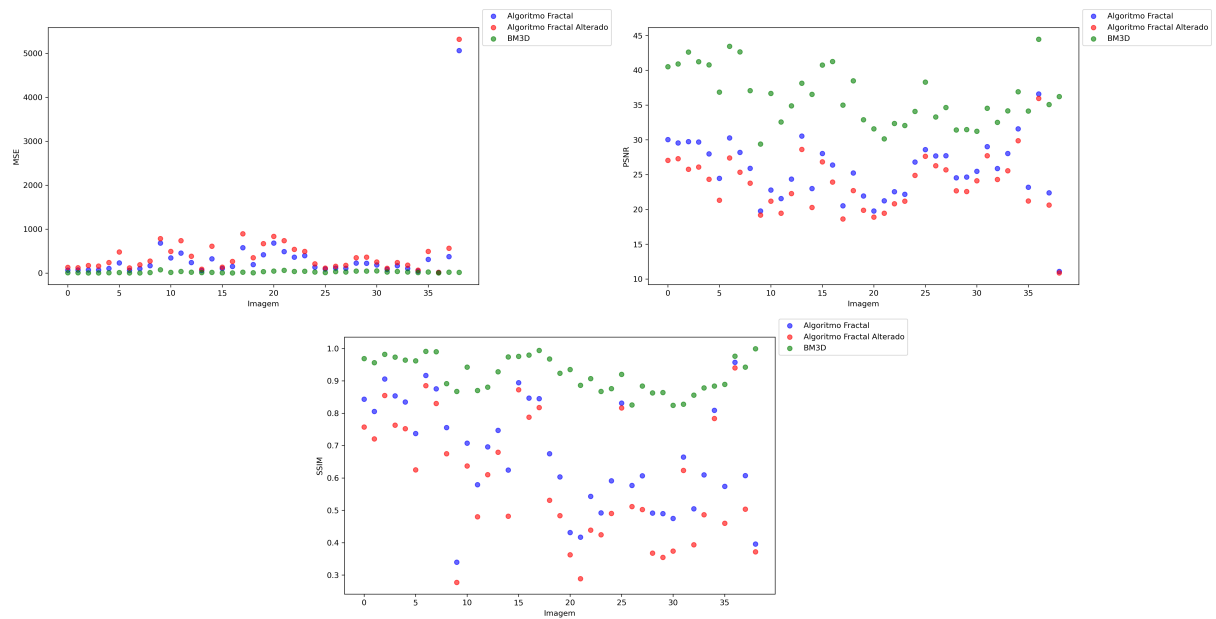
Figura 47 – Resultados das métricas MSE, PSNR e SSIM para ruído Sal e Pimenta com $p_1 = 0.16$ e $p_2 = 0.16$.



Fonte: Elaborado pelo autor.

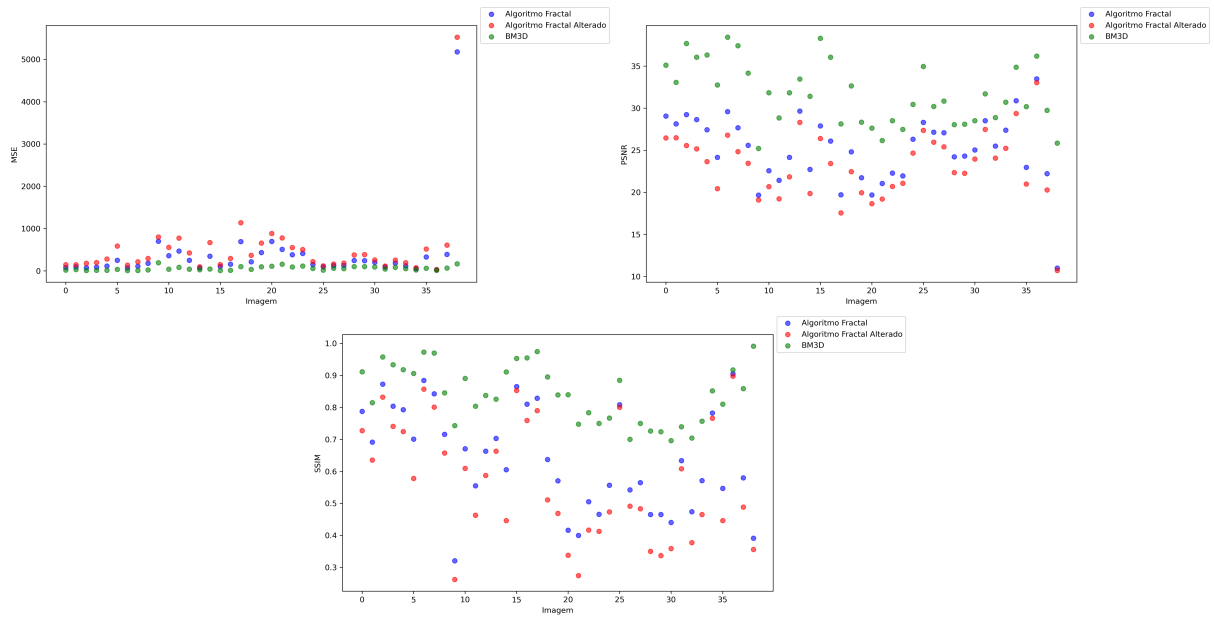
A.3 Ruído Uniforme

Figura 48 – resultados das métricas MSE, PSNR e SSIM para ruído Uniforme com $a = 0.05$ e $b = 0.05$.



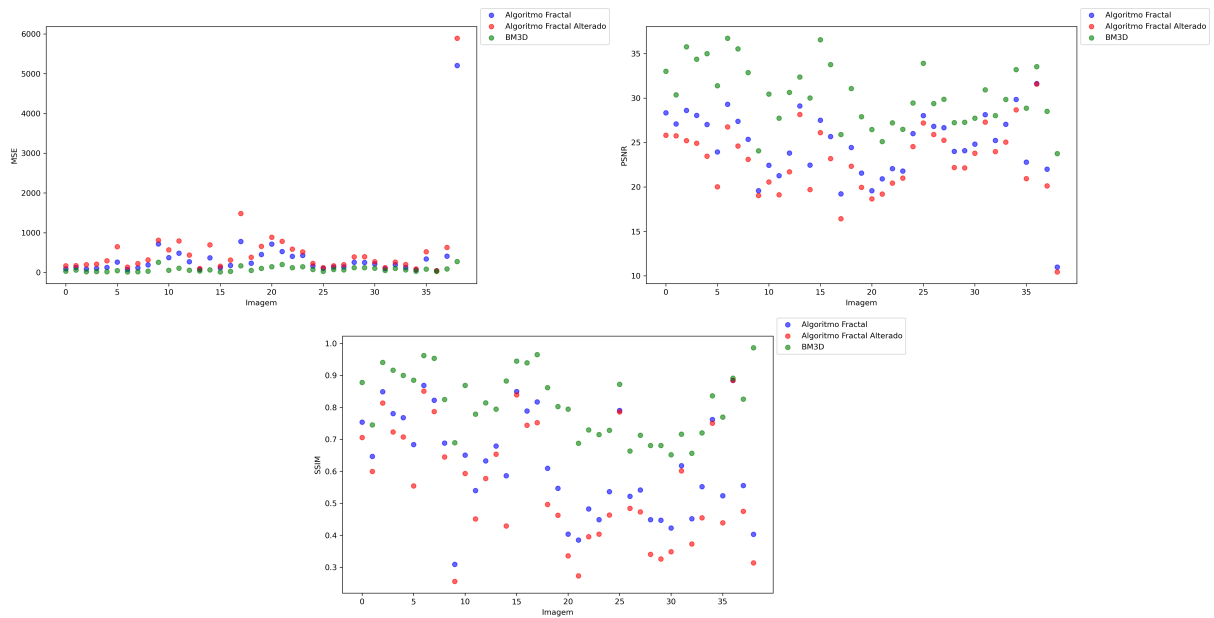
Fonte: Elaborado pelo autor.

Figura 49 – resultados das métricas MSE, PSNR e SSIM para ruído Uniforme com $a = 0.15$ e $b = 0.15$.



Fonte: Elaborado pelo autor.

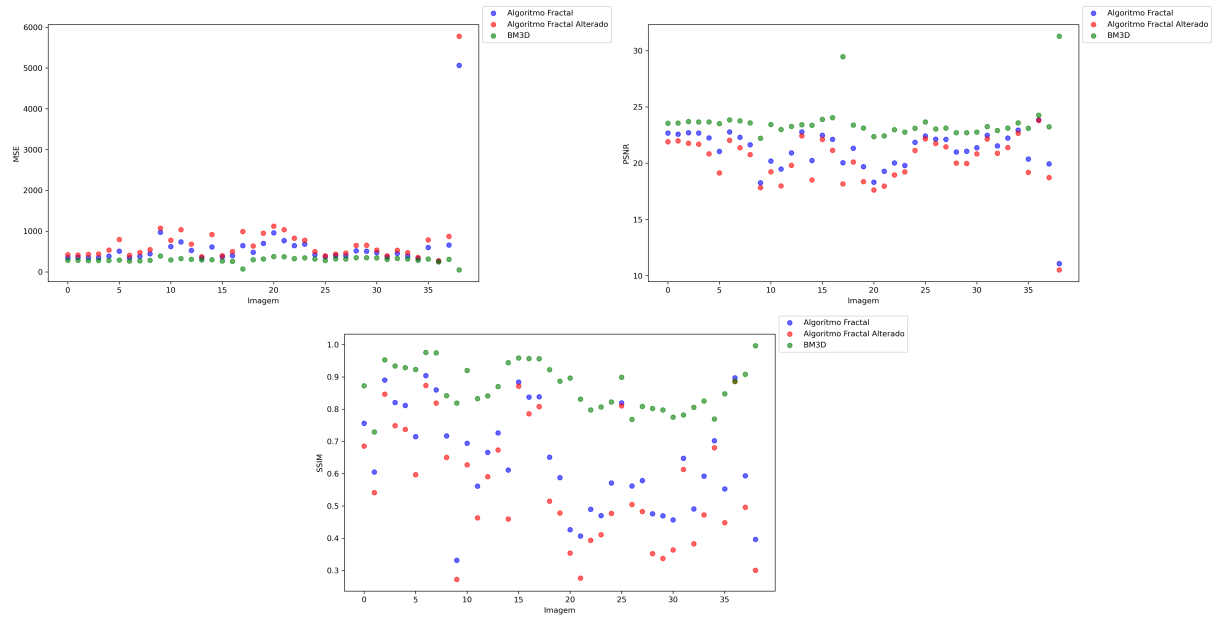
Figura 50 – resultados das métricas MSE, PSNR e SSIM para ruído Uniforme com $a = 0.2$ e $b = 0.2$.



Fonte: Elaborado pelo autor.

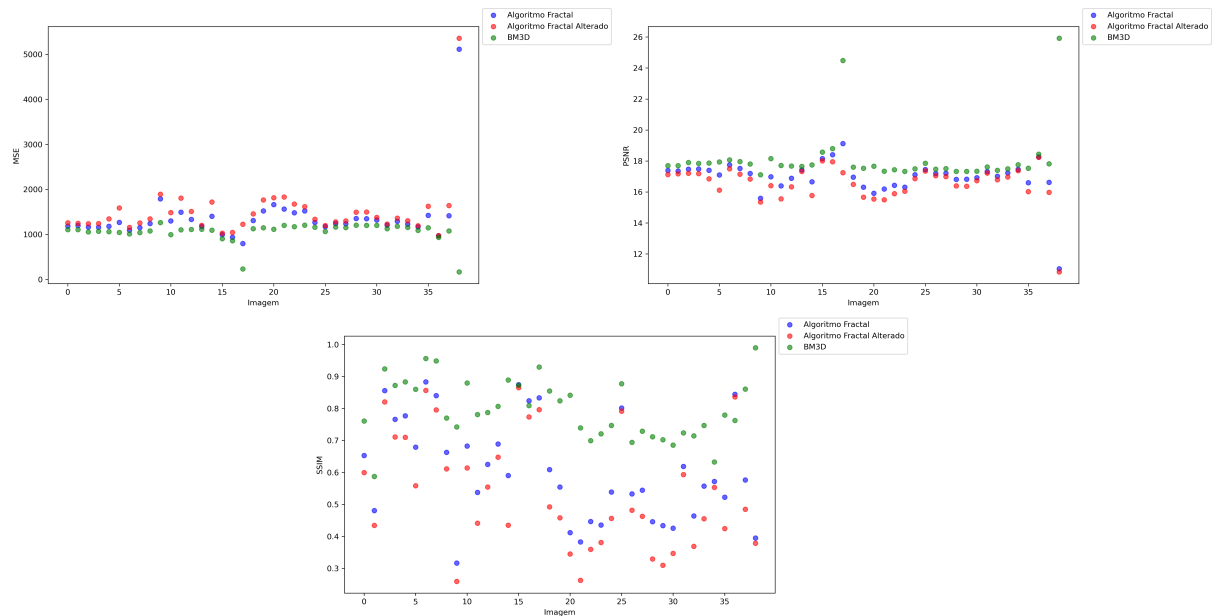
A.4 Ruído Gama

Figura 51 – resultados das métricas MSE, PSNR e SSIM para ruído gama com $a = 2$ e $b = 0.035$.



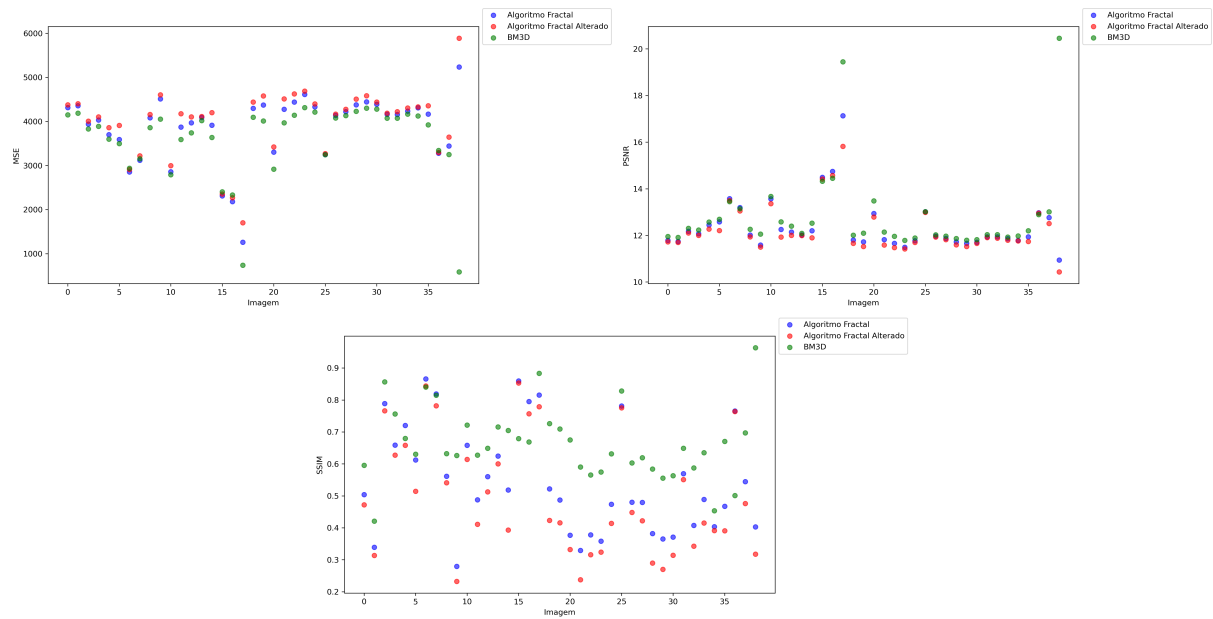
Fonte: Elaborado pelo autor.

Figura 52 – resultados das métricas MSE, PSNR e SSIM para ruído gama com $a = 3$ e $b = 0.045$.



Fonte: Elaborado pelo autor.

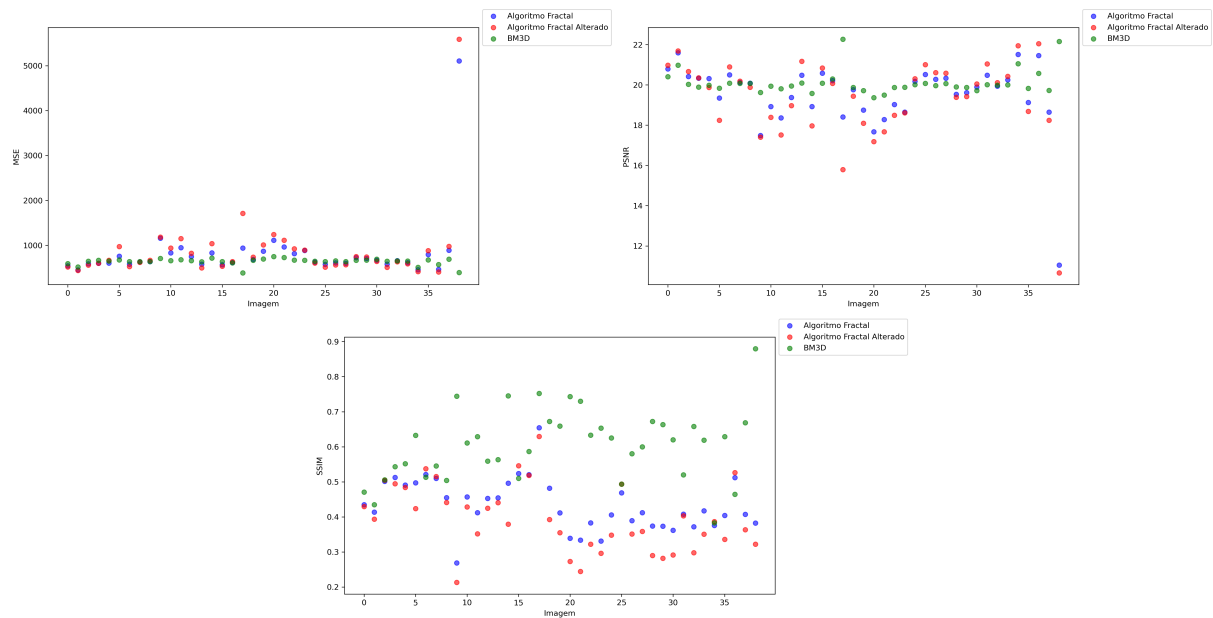
Figura 53 – resultados das métricas MSE, PSNR e SSIM para ruído gama com $a = 4$ e $b = 0.065$.



Fonte: Elaborado pelo autor.

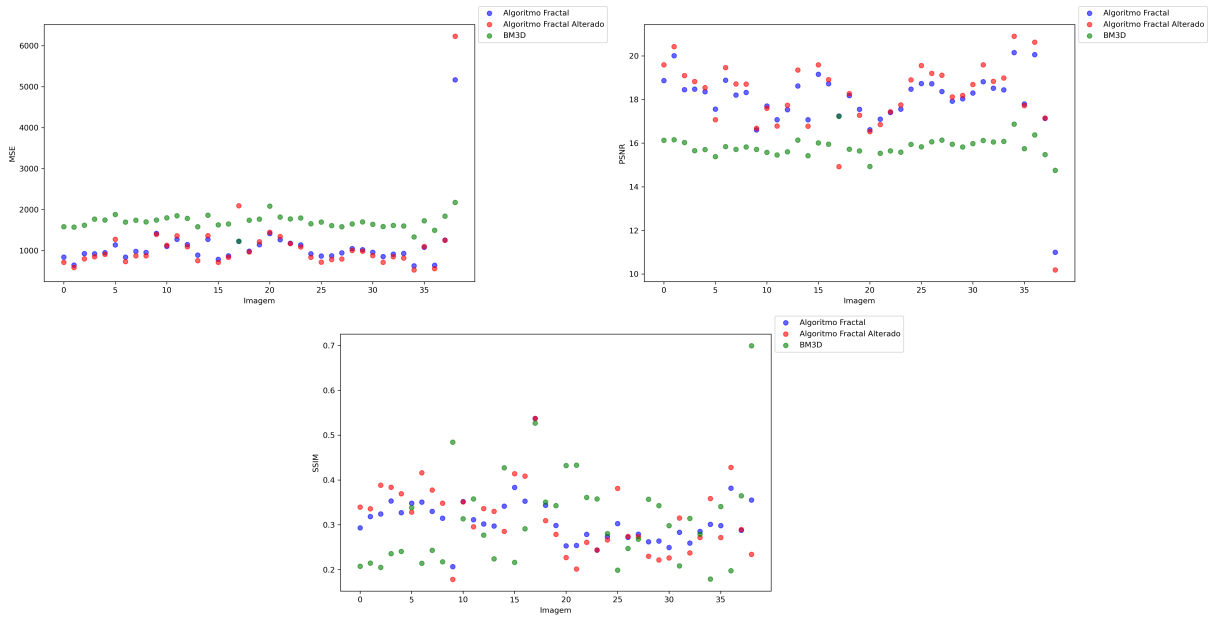
A.5 Ruído Periódico

Figura 54 – resultados das métricas MSE, PSNR e SSIM para ruído Periódico com $f_2(x, y)$.



Fonte: Elaborado pelo autor.

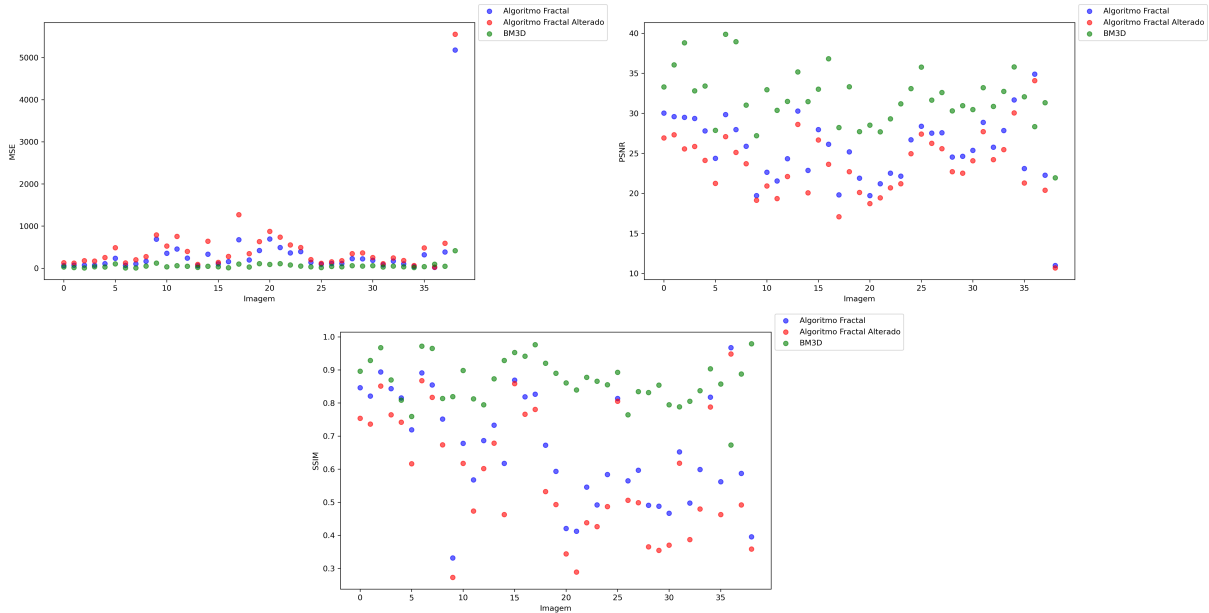
Figura 55 – resultados das métricas MSE, PSNR e SSIM para ruído Periódico com $f_4(x, y)$.



Fonte: Elaborado pelo autor.

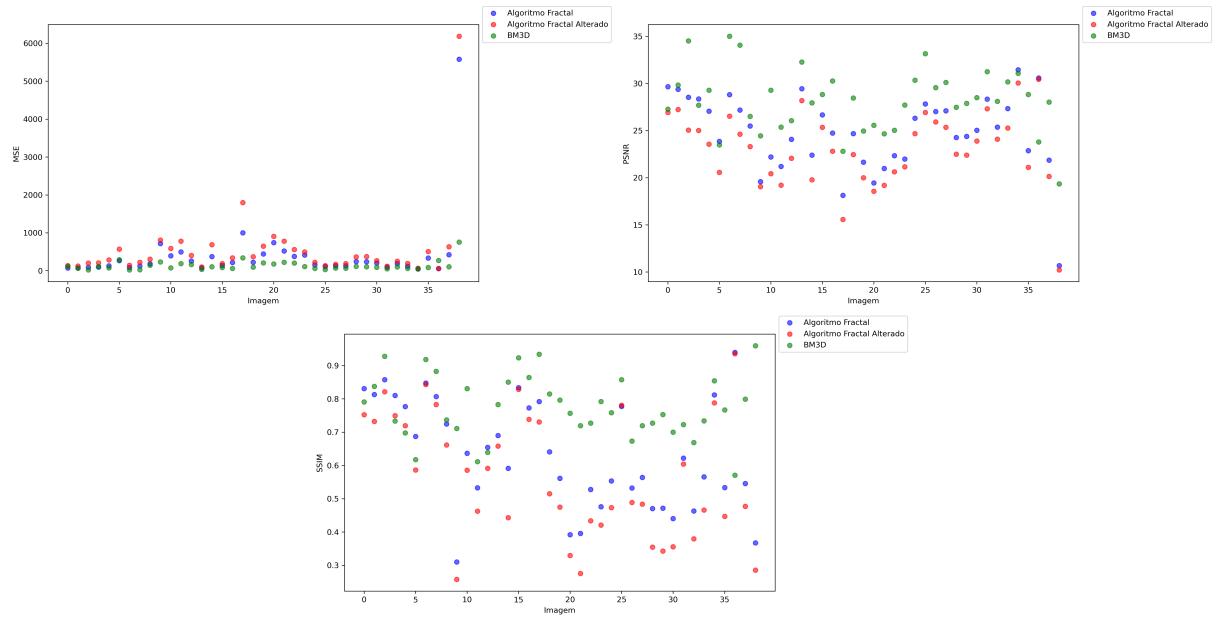
A.6 Ruído Speckle

Figura 56 – resultados das métricas MSE, PSNR e SSIM para ruído uniforme com $a = 100$ e $\alpha = \frac{1}{100}$.



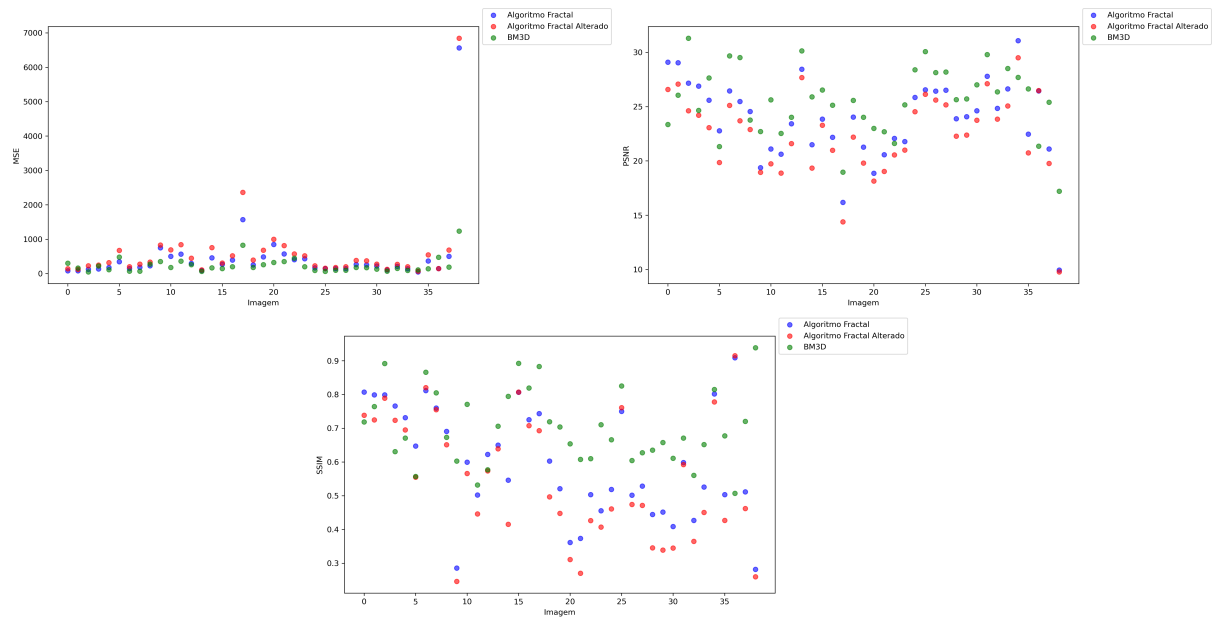
Fonte: Elaborado pelo autor.

Figura 57 – resultados das métricas MSE, PSNR e SSIM para ruído uniforme com $a = 25$ e $\alpha = \frac{1}{25}$.



Fonte: Elaborado pelo autor.

Figura 58 – resultados das métricas MSE, PSNR e SSIM para ruído uniforme com $a = 10$ e $\alpha = \frac{1}{10}$.



Fonte: Elaborado pelo autor.

Referências

- BARNESLEY, M. F. *Fractals everywhere*. [S.l.]: Academic press, 2014.
- BONCELET, C. Chapter 7 - image noise models. In: BOVIK, A. (Ed.). *The Essential Guide to Image Processing*. Boston: Academic Press, 2009. p. 143–167. ISBN 978-0-12-374457-9. Disponível em: <<https://www.sciencedirect.com/science/article/pii/B978012374457900007X>>.
- BOYAT, A. K.; JOSHI, B. K. A review paper: noise models in digital image processing. *arXiv preprint arXiv:1505.03489*, 2015.
- DABOV, K.; FOI, A.; KATKOVNIK, V.; EGIAZARIAN, K. Image denoising with block-matching and 3d filtering. In: SPIE. *Image processing: algorithms and systems, neural networks, and machine learning*. [S.l.], 2006. v. 6064, p. 354–365.
- DOMINGUES, H. H. *Espaços métricos e introdução à topologia*. [S.l.]: Atual, 1982.
- FISHER, Y. Fractal image compression. *Fractals*, World Scientific, v. 2, n. 03, p. 347–361, 1994.
- GHAZEL, M.; FREEMAN, G. H.; VRSCAY, E. R. Fractal image denoising. *IEEE Transactions on image processing*, IEEE, v. 12, n. 12, p. 1560–1578, 2003.
- HAMANO, T.; TOKUDA, K.; KANEKO, M. Image restoration based on estimation of fractal structure. In: *Proceedings of Digital Processing Applications (TENCON '96)*. [S.l.: s.n.], 1996. v. 1, p. 311–316 vol.1.
- ImageMagick Studio LLC. *ImageMagick*. 2025. Disponível em: <<https://imagemagick.org>>.
- JÄHNE, B. *Digital image processing*. [S.l.]: Springer, 2005.
- KONG, Z.; YANG, X.; HE, L. A comprehensive comparison of multi-dimensional image denoising methods. *arXiv preprint arXiv:2011.03462*, 2020.
- MANDELBROT, B. B. Fractal geometry: what is it, and what does it do? *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, The Royal Society London, v. 423, n. 1864, p. 3–16, 1989.
- NAJJAR, Y. A. Comparative analysis of image quality assessment metrics: Mse, psnr, ssim and fsim. *International Journal of Science and Research (IJSR)*, v. 13, n. 3, p. 110–114, 2024.
- SONG, J.; WANG, B.; JIANG, Q.; HAO, X. Exploring the role of fractal geometry in engineering image processing based on similarity and symmetry: A review. *Symmetry*, MDPI, v. 16, n. 12, p. 1658, 2024.
- WEBER, A. G. *The USC-SIPI Image Database: Version 6*. Los Angeles, CA, USA, 2018. Acesso em: 18 out. 2025. Disponível em: <<https://sipi.usc.edu/database/>>.