Aluno: Bruno Alexandre Krinski
Matéria: Processamento de Imagens

Tarefa 1 - Aplique diferentes tamanhos do filtro gaussiano que você implementou a uma imagem de grandes dimensões (ex. 4000X3000) e compare os resultados (imagem resultante e tempo de processamento) com o filtro GaussianBlur da bilbioteca OpenCV. Relate a suas conclusões.

Neste exercício foram realizados a comparação entre minha implementação da mascara gaussiana e a implementação do OpenCV. A conclusão obtida foi que, para valores pequenos de mascara e sigma, a minha implementação obteve um resultado muito similar com a implementação do OpenCV, porém com um tempo de execução um pouco maior. Para valores intermediários de mascara e sigma, a minha implementação não só conseguiu resultados muito similares com a implementação do OpenCV, como também conseguiu tempos de execução menores. Porém, para valores muito grandes de mascara e sigma, a minha implementação começou a obter resultados divergentes da implementação do OpenCV. E por fim, quando aumentou-se apenas o valor da mascara e manteve-se o valor do sigma, a minha implementação obteve um resultado completamente diferente do resultado da implementação do OpenCV.

A seguir, é apresentado mais detalhes de cada um dos testes executados. A função utilizada para gerar a mascara gaussiana utilizada pela minha implementação foi a função exibida a baixo:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

Figura 1: Fórmula utilizada para calcular o filtro da gaussiana.

A imagem utilizada para os experimentos foi a imagem apresentada na Figura 2. A imagem tem 4928 pixels de largura e 3264 pixels de altura.

Figura 2: Imagem utilizada nos experimentos do filtro da gaussiana e mediana.

Resultados:

Como é possível observar na Figura 3, visualmente, o filtro gaussiano implementado por mim obteve um resultado muito próximo ao resultado do filtro implementado pelo OpenCV. A comparação de histogramas das imagens resultantes dos filtros obteve um resultado de similaridade de 97%. Entretanto, o tempo de execução da função gaussiana (0.0316789150238) foi menor do que o tempo de execução da função gaussiana implementado por mim (0.0911040306091).

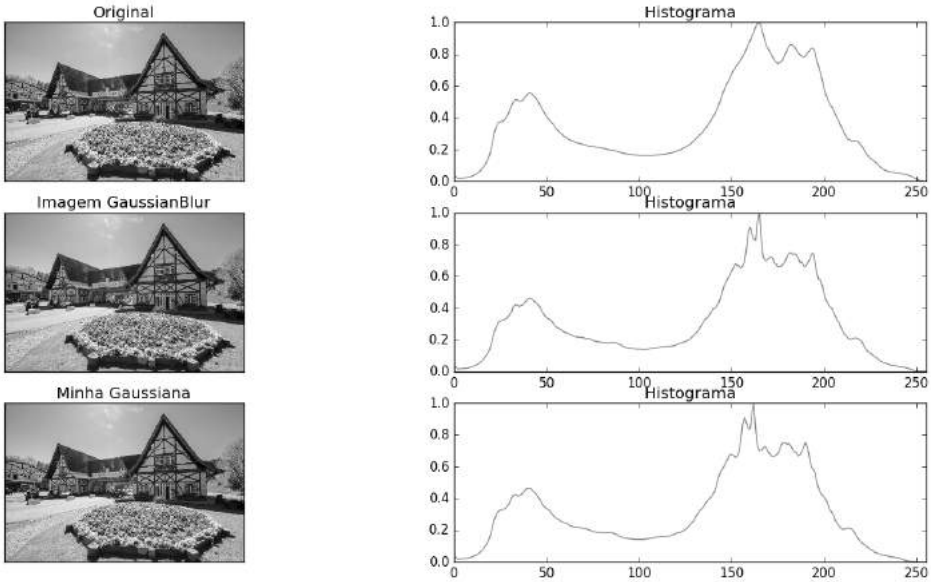

Figura 3: Resultado obtido com mascara de tamanho 5x5 e sigma 1.

O mesmo resultado foi obtido com valores maiores da mascara e do sigma. Como é possível observar na Figura 4, para uma mascara de tamanho 125x125 e sigma 25, visualmente o filtro gaussiana implementado por mim e o filtro gaussiano implementado pelo OpenCV obtiveram resultados similares. O valor de similaridade obtido pela comparação dos histogramas das imagens resultantes de ambos os filtros foi de 96%. O tempo de execução da implementação do OpenCV (1.37114882469) foi maior que o tempo de execução da minha implementação (0.428194046021) superando as expectativas.

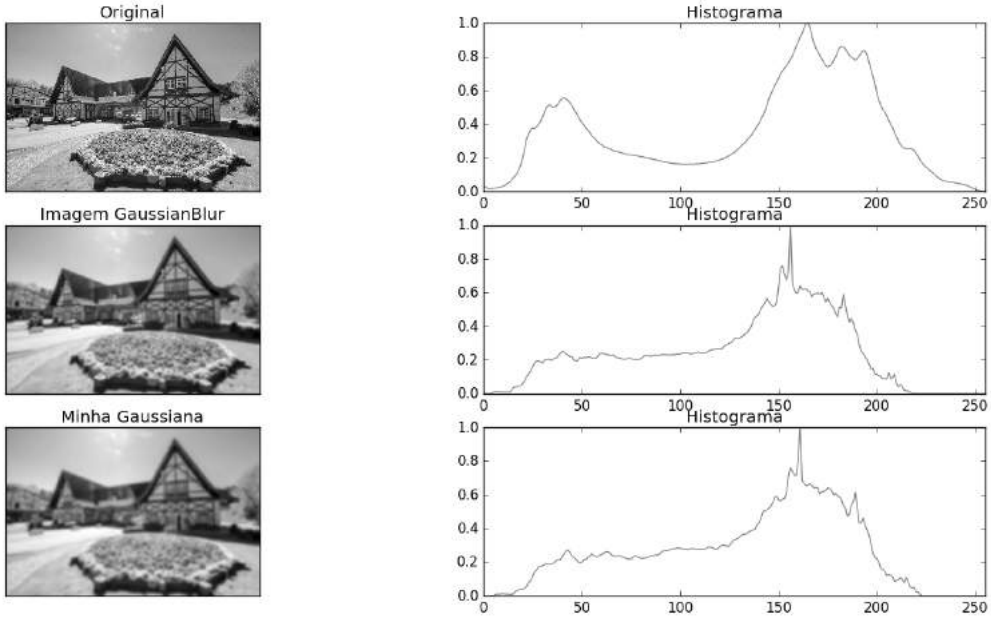

Figura 4: Resultado obtido com mascara de tamanho 125x125 e sigma 25.

Para valores muito grandes de mascara (225x225) e sigma (50), o filtro gaussiano implementado por mim obteve um borramento maior na imagem, como é possível visualizar na Figura 5. Porém, o valor de similaridade entre os histogramas foi de 97%. O tempo de execução obtido pela minha implementação foi de 0.617653846741 enquanto que o tempo de execução obtido pela implementação do OpenCV foi de 2.49992108345.

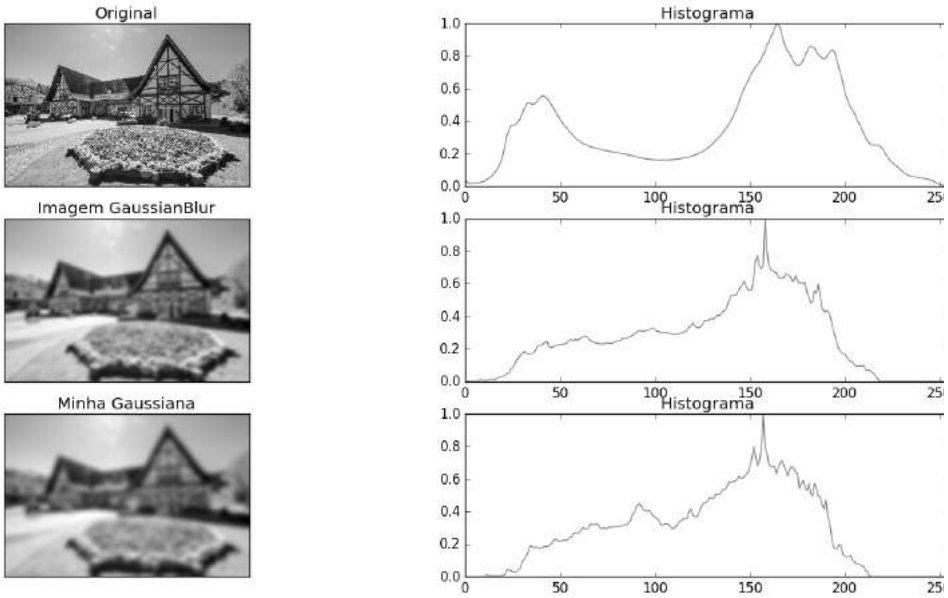

Figura 5: Resultado obtido com mascara de tamanho 225x225 e sigma 50.

Outro teste executado foi aumentar o tamanho da mascara (525x525) e mantendo o mesmo sigma (50). O resultado obtido pela minha implementação foi muito diferente do resultado obtido pela implementação do OpenCV. Como é possível visualizar na Figura 17, visualmente, tanto as imagens quanto seus histogramas são diferentes entre si. A comparação de similaridade entre os histogramas obteve um resultado de 23%. E o tempo de execução da minha implementação (1.24604010582) foi muito menor do que o tempo de execução da função do OpenCV (7.03612518311). Porém, mesmo com tempo muito inferior, como não obteve o mesmo resultado, não é uma implementação confiável.

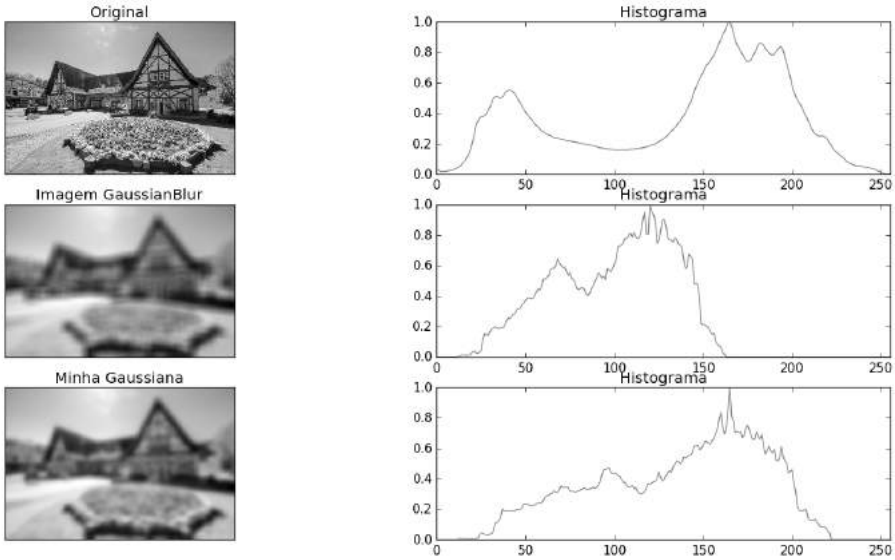

Figura 17: Resultado obtido com mascara de tamanho 525x525 e sigma 50.

Tarefa 2 - Analise diferentes tamanhos do filtro da mediana em diferentes níveis de ruído. Verifique o impacto na imagem resultante.

O valor da mascara utilizado nos testes foi 25x25. Os níveis de ruídos testados foram baixo = 50, médio = 100 e alto = 200. As Figuras 6,7 e 8 mostram os resultados obtidos em cada nível de ruído aplicado na imagem. Em ambos os casos, o filtro da mediana obteve um resultado muito bom em limpar os ruídos das imagens, como é possível visualizar nas figuras a baixo.

Na Figura 6 é possível visualizar o resultado da imagem que foi aplicado nível baixo de ruído e o resultado após o filtro da mediana limpar este ruído.

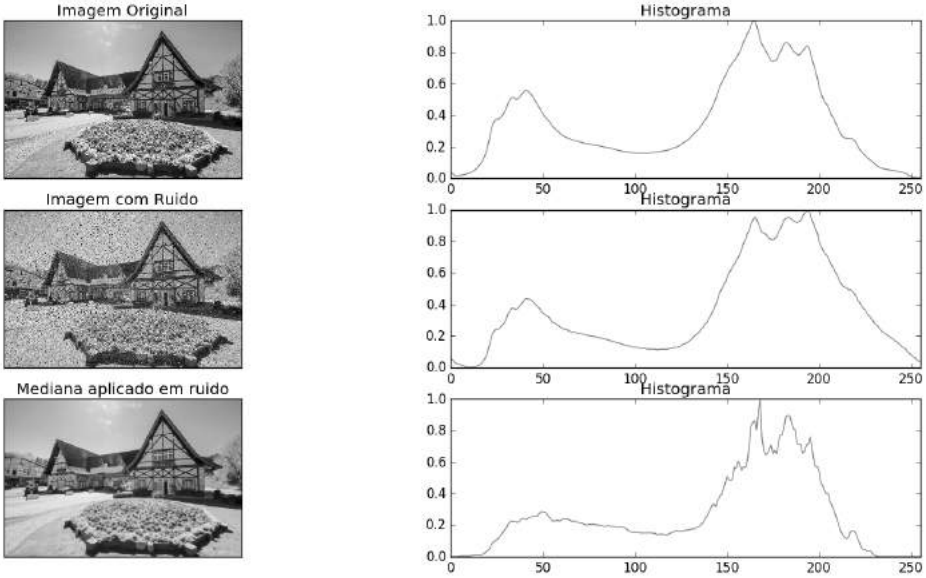

Figura 6: Imagem original, imagem com nível baixo de ruído e o resultado após o filtro da mediana remover o ruído.

Na Figura 7 é possível visualizar o resultado da imagem que foi aplicado nível médio de ruído e o resultado após o filtro da mediana limpar este ruído.

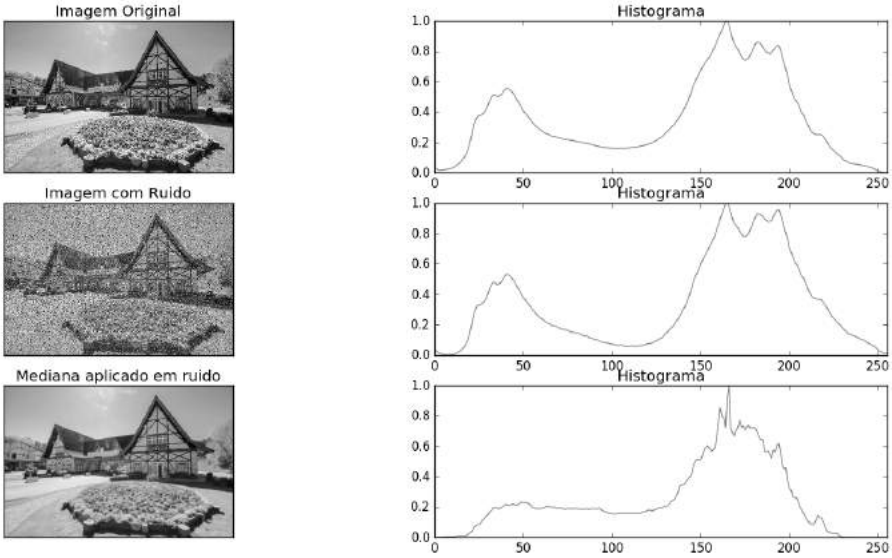

Figura 7: Imagem original, imagem com nível médio de ruído e o resultado após o filtro da mediana remover o ruído.

Na Figura 8 é possível visualizar o resultado da imagem que foi aplicado nível alto de ruído e o resultado após o filtro da mediana limpar este ruído.

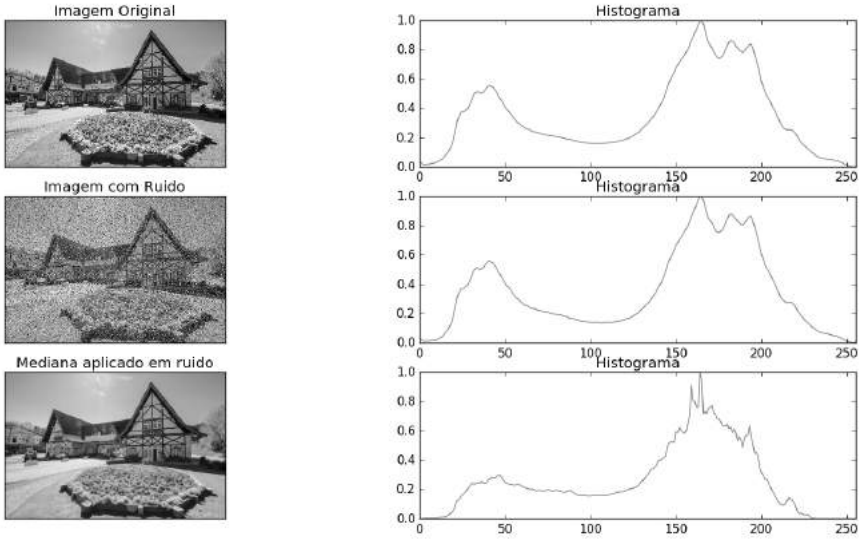

Figura 8: Imagem original, imagem com nível alto de ruído e o resultado após o filtro da mediana remover o ruído.

Tarefa 2.1 - Compare o desempenho e resultados do filtro da Mediana implementado por você e implementado na biblioteca OpenCV.

Como é possível visualizar na Figura 9, o resultado visual obtido pela mascara da mediana implementada por mim e implementada pela OpenCV foi muito semelhante. O resultado obtido pela similaridade de histogramas foi de 99%. Porém, enquanto o tempo de execução obtido pela função do OpenCV foi 0.0845239162445, a minha implementação obteve tempo 194.366774082.
O teste foi realizado com mascara de tamanho 5x5. Não foi possível testar com mascaras maiores devido ao tempo computacional muito alto.

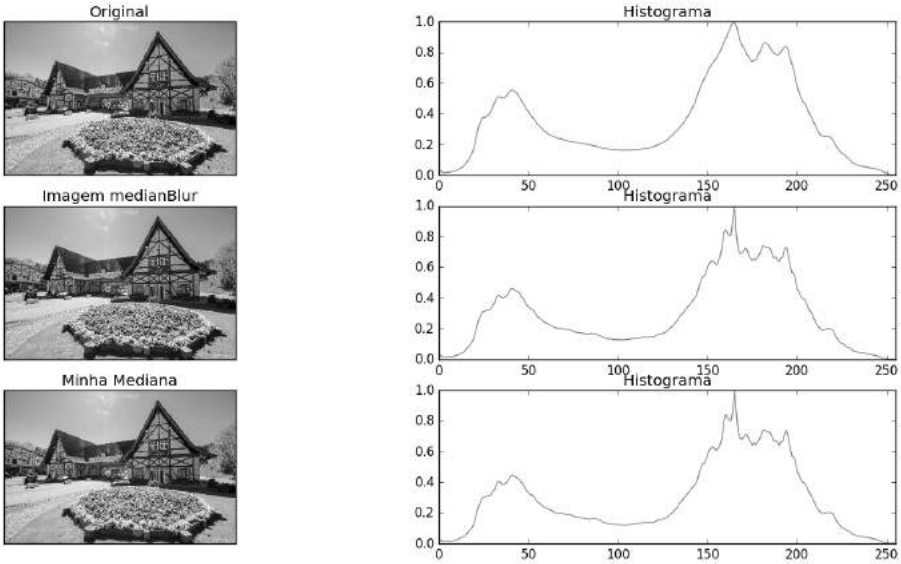

Figura 9: Comparação entre os filtros da mediana para mascara de tamanho 5x5

Tarefa 3 - Crie 100 imagens com ruído aleatório, empilhe essas imagens em lotes de 10 e analise o impacto em função do ruído. Repita o experimento para diferentes níveis de ruído (fraco, médio, forte).

Nesta tarefa, foram aplicados diferentes níveis de ruído na imagem original, nível baixo = 50, nível médio = 100 e nível alto = 200. As imagens 11,13 e 15 mostram o resultado do nível de ruído aplicado na imagem. No total, 100 imagens foram criadas para cada nível de ruído e depois empilhadas em lotes de 10 imagens. As Figuras 12,14 e 16 mostram o resultado desde empilhamento.

No primeiro nível, nível baixo de ruído, foi possível conseguir um nível bom de limpeza de ruídos, porém ouve perda de algumas informações. Por exemplo, na imagem original (Figura 10) é possível visualizar a incidência da luz do sol no edifício. Esta informação é perdida na Figura 11. Nos níveis médio e alto foi maior ainda a perda de iluminação da imagem. O empilhamento funcionou melhor para o nível baixo de ruído. Nos demais níveis, a imagem resultante aparece ainda com certo nível de ruído.

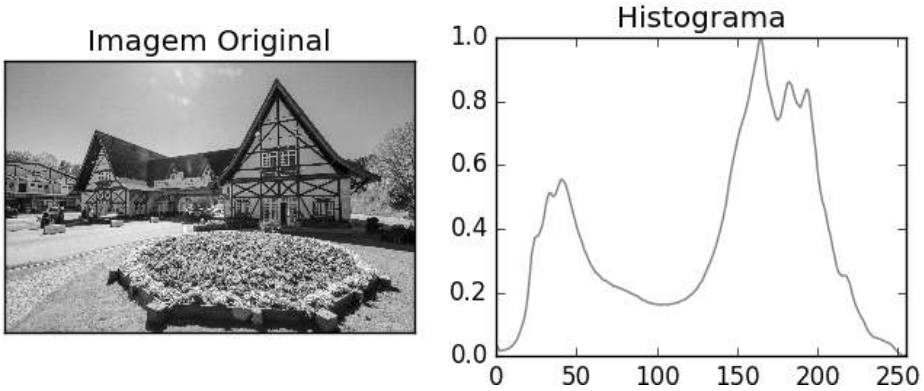

Figura 10: Imagem original sem ruido.

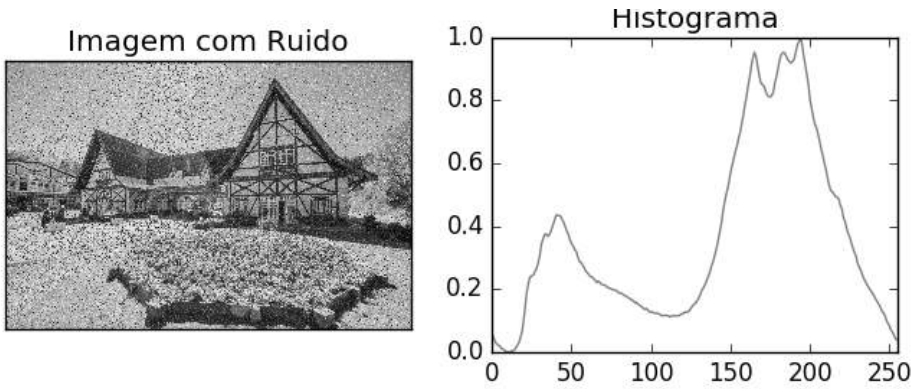

Figura 11: Imagem com nível baixo de ruído.

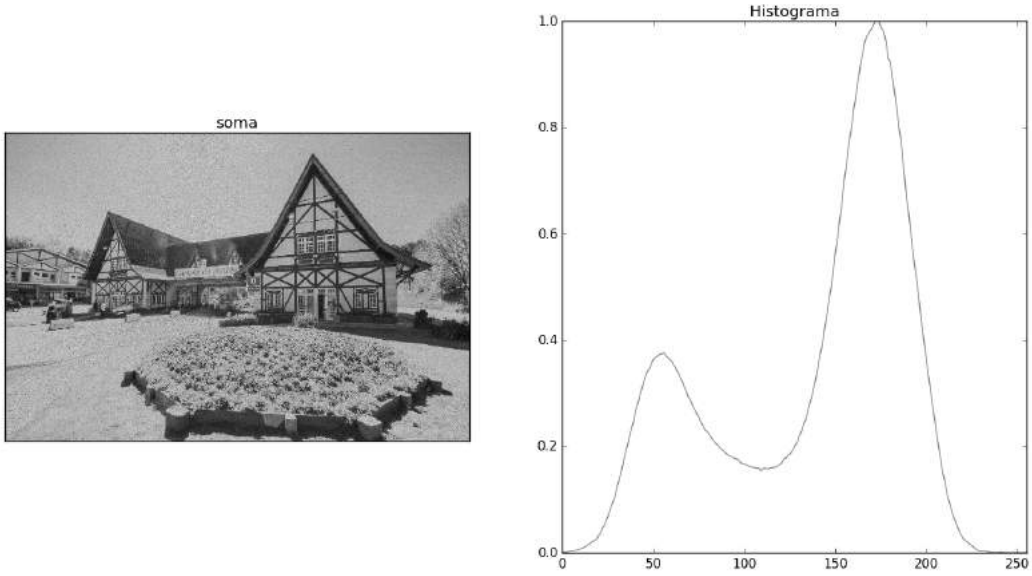

Figura 12: Lote de 10 imagens com nível baixo de ruído somadas.

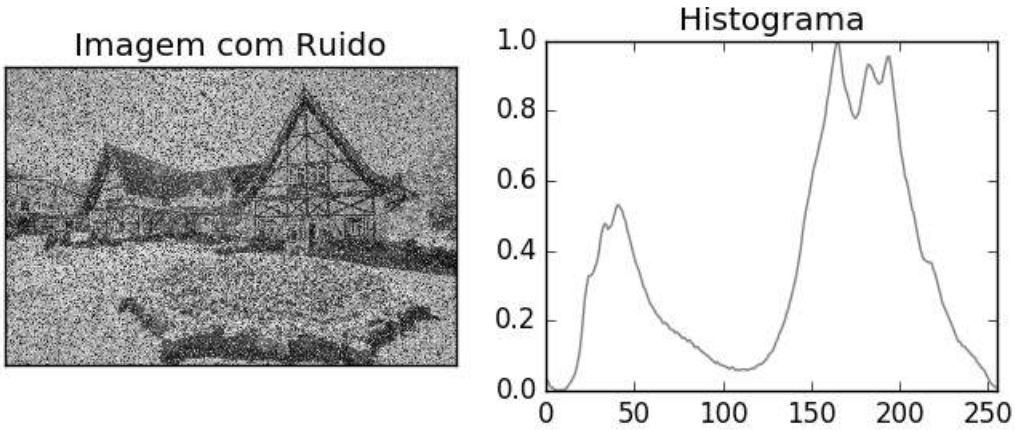

Figura 13: Imagem com nível médio de ruído.

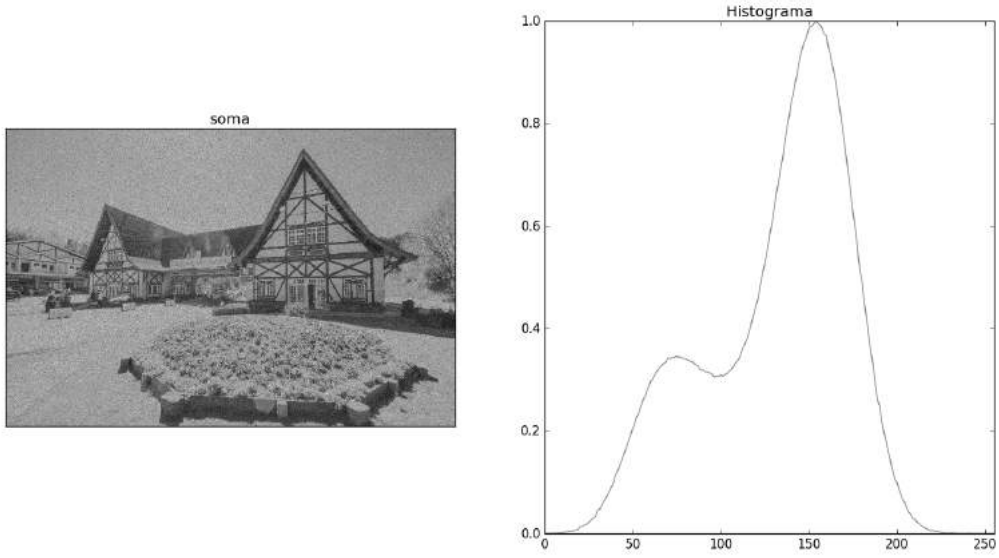

Figura 14: Lote de 10 imagens com nível médio de ruído somadas.

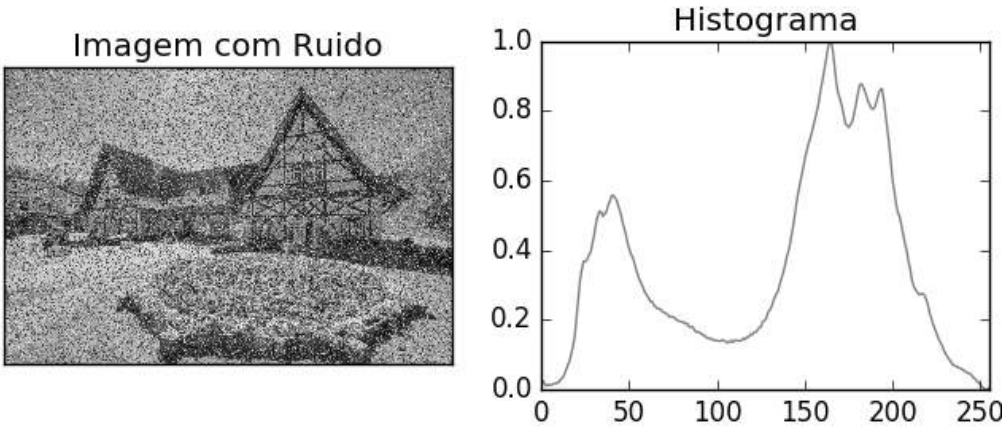

Figura 15: Imagem com nível alto de ruído

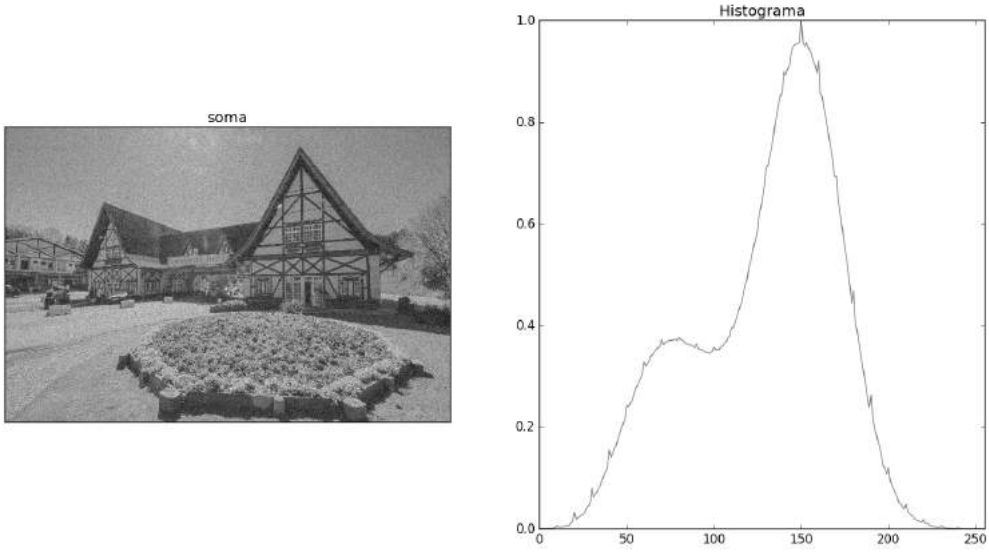

Figura 16: Lote de 10 imagens com nível alto de ruído somadas.