Limiarização Adaptativa com Imagem Integral

Felipe Adonai de Moraes

Sumário

- Objetivos
- Fundamentação Teórica
- Aplicações Práticas
- Algoritmo
- Exemplos
- Resultados

Objetivo

Implementar Técnica de Limiarização Adaptativa com Imagem Integral e comparar com outras técnicas de limiarização

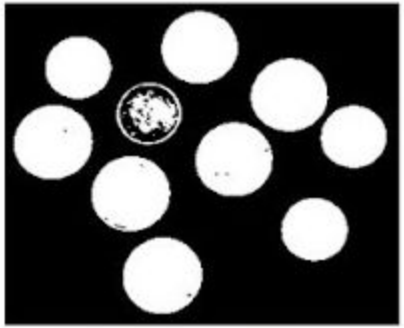
- Linguagem de programação: Python

Fundamentação Teórica

Limiarização

- Classificar pixels em duas categorias: claros ou escuros com base em um limiar
- Problema com iluminação
- Método de Otsu


Input Image

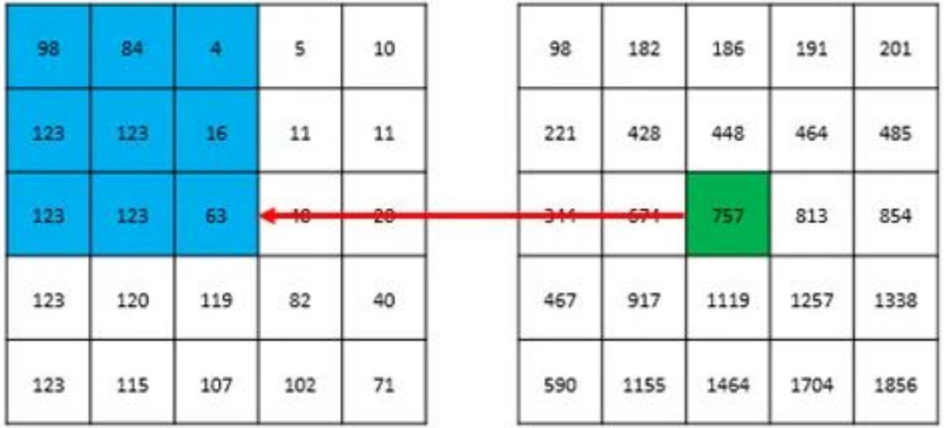

threshold output

Limiarização adaptativa

- Forma de limiarização onde cada pixel possui um limiar diferente com base em características locais, como a média dos pixels vizinhos

Imagem Integral (Tabela de Área Somada)

- Forma eficiente de calcular a área de uma parte de uma imagem
- Custo linear



(left) An original grayscale image and (right) an integral image

Fonte: https://miro.medium.com/v2/resize:fit:640/format:webp/1*Bj5dt4XsTfwp8w81haI8yQ.png

Cálculo Imagem Integral

$$I(x,y) = f(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1).$$

I é o valor da integral nos pontos (x,y)
f é a intensidade nos pontos (x,y)

98	84	4	5	10
123	123	16	11	11
123	123	63	40	20
123	120	119	82	40
123	115	107	102	71

98	182	186	191	201
221	428	448	464	485
344	674	757	813	854
467	917	1119	1257	1338
590	1155	1464	1704	1856

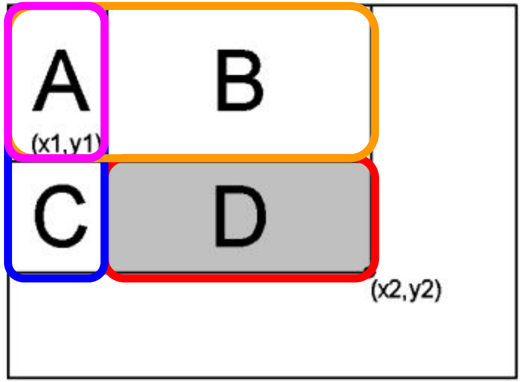
(left) An original grayscale image and (right) an integral image

Cálculo Área de um retângulo

$$\sum_{x=x_1}^{x_2}\sum_{y=y_1}^{y_2} f(x,y) = I(x_2,y_2) - I(x_2,y_1-1) - I(x_1-1,y_2) + I(x_1-1,y_1-1).$$

D B C A

98	182	186	191	201
221	428	448	464	485
~~314~~	~~674~~	757	813	854
467	917	1119	1257	1338
590	1155	1464	1704	1856



(A+B+C+D)-(A+B)-(A+C)+A.

Limiarização Adaptativa com Imagem Integral

- Maior robustez quanto a mudanças fortes de luminosidade
- Algoritmo de performance real-time
    - limiarização normalmente faz parte de um processo maior
- Pontos positivos:
    - fácil de implementar; produz mesma saída independente da ordem de processamento dos pixels; competitivo real time; ordem de complexidade linear
- Pontos negativos:
    - usa uma iteração a mais na imagem para gerar a saída
    - Menos flexível para definir a forma da região de vizinhança, se limita a regiões retangulares.

Aplicações práticas

- Limiarização normalmente faz parte de um processo maior
- Detecção de bordas
- Real-time video streams
- Realidade aumentada
    - Detecção de marcadores

Algoritmo

1. Calcula integral da imagem
2. Efetua limiarização:
   a. Parâmetros: Tamanho do bloco S; valor do limiar em porcentagem T
   b. Para cada pixel, realiza a média dos valores dentro do bloco, e efetua a comparação para limiarização: se o valor do pixel for T por cento menor que a média, é preto, senão é branco

Calcula integral da imagem

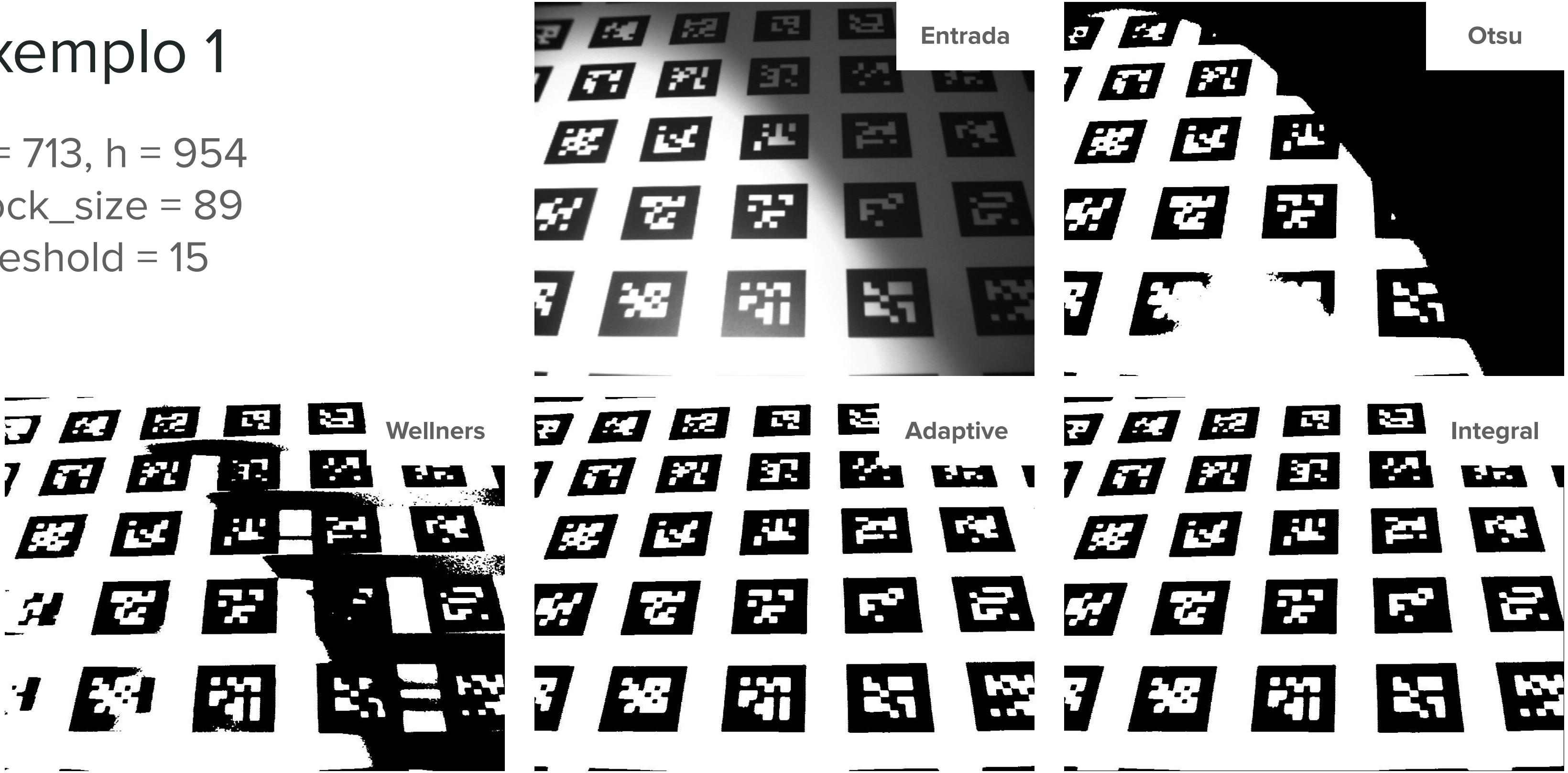
```python
intImg = np.zeros((w, h))
  for i in range(w):
    sum = 0
    for j in range(h):
      sum += in_img[i, j]
      if i == 0:
        intImg[i, j] = sum
      else:
        intImg[i, j] = intImg[i-1, j] + sum
```

Efetua limiarização

```
out_img = np.zeros((w, h))
  # Itera sobre cada pixel da imagem.
  for i in range(w - 1):
    for j in range(h - 1):
      # Define as coordenadas do bloco.
      x1 = max(0, i - block_size // 2)
      x2 = min(w - 1, i + block_size // 2)
      y1 = max(0, j - block_size // 2)
      y2 = min(h - 1, j + block_size // 2)
      # Calcula a soma dos pixels no bloco.
      block_sum = intImg[x2, y2] - intImg[x1, y2] - intImg[x2, y1] + intImg[x1, y1]
      # Calcula a quantidade de pixels no bloco.
      block_count = (x2 - x1) * (y2 - y1)
      # Compara o pixel com a média do bloco.
      if in_img[i, j] * block_count <= (block_sum * (100 - threshold) / 100):
        out_img[i, j] = 0
      else:
        out_img[i, j] = 255
```

Exemplo 1

w = 713, h = 954
block_size = 89
threshold = 15



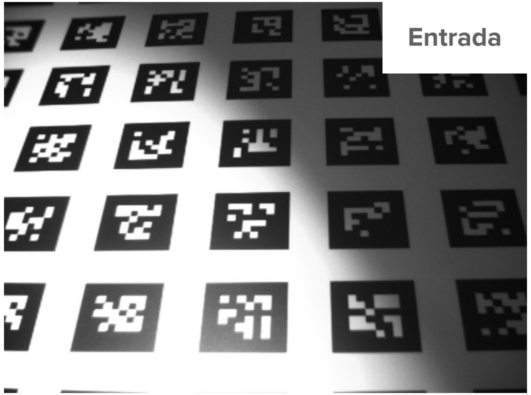
Algoritmo	Tempo	Num. Iterações
Otsu	1.5153 s	1360659
Wellners	0.6405 s	680202
Medium	11.2971 s	4981626810
Integral	3.0514 s	1358738

Exemplo 2

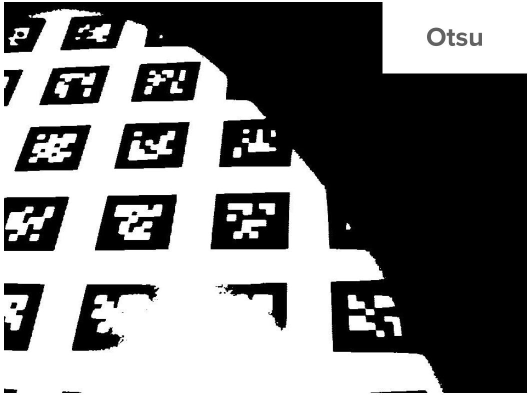
w = 713, h = 954

block_size = 10

threshold = 15

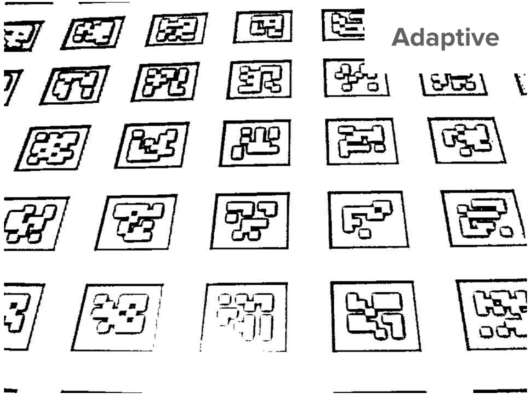


Entrada

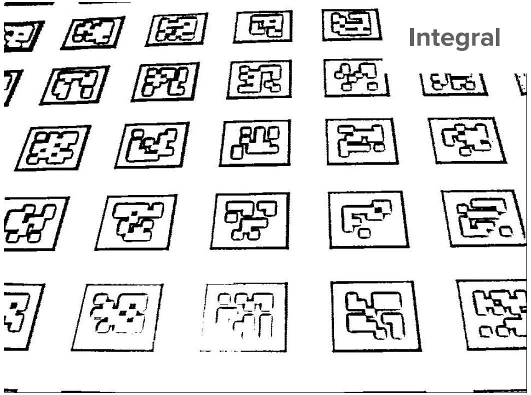


Otsu

Wellners



Adaptive

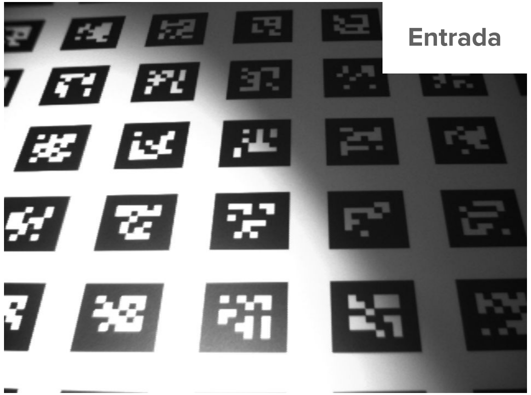


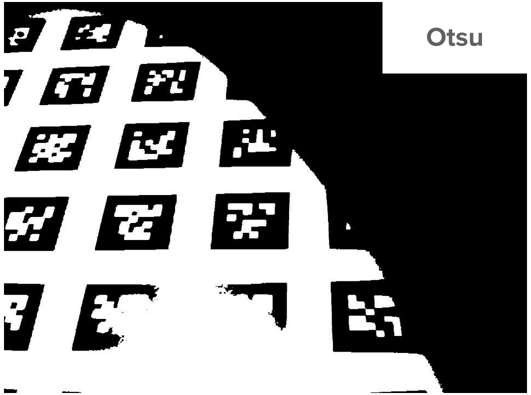
Integral

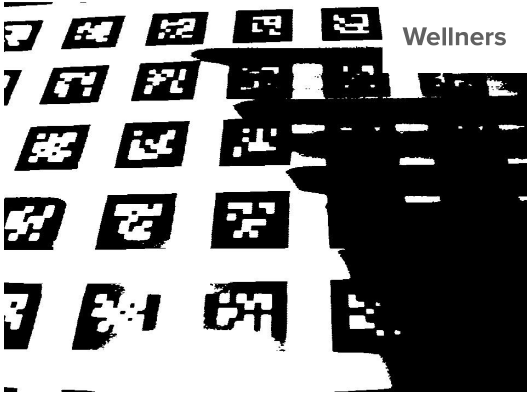
Algoritmo	Tempo	Num. Iterações
Otsu	1.6077	1360659
Wellners	0.6271	680202
Medium	11.0732	4981626810
Integral	3.0770	1358738

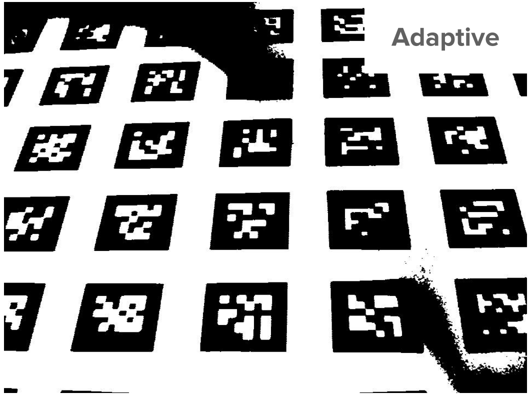
Exemplo 3

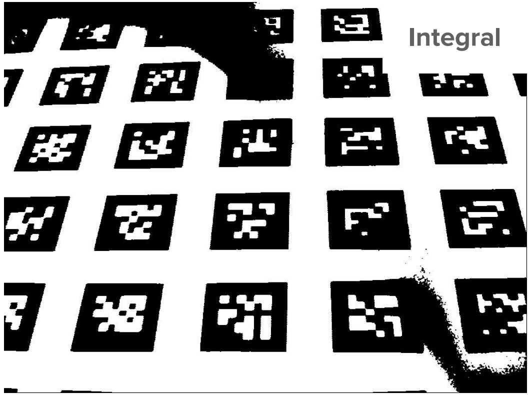
w = 713, h = 954
block_size = 500
threshold = 15







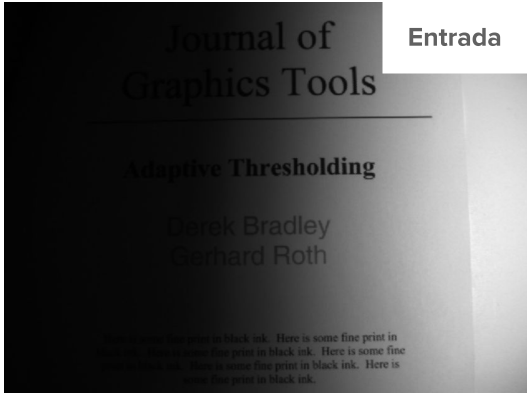




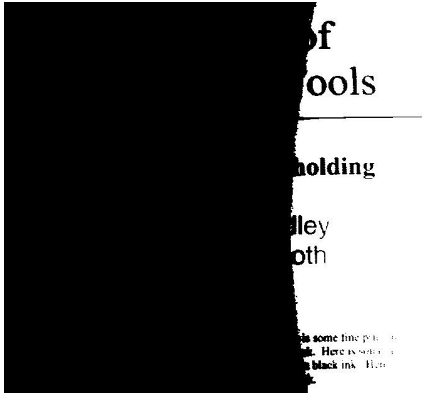
Algoritmo	Tempo	Num. Iterações
Otsu	1.5840	1360659
Wellners	0.6949	680202
Medium	110.6432	121686617702
Integral	3.3581	1358738

Exemplo 4

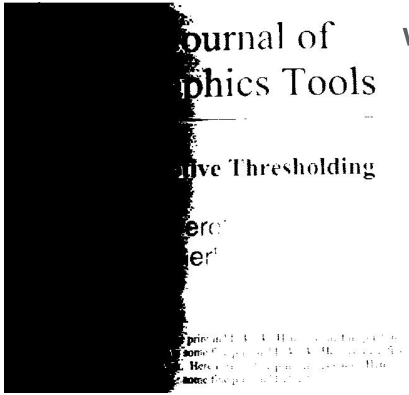
w = 624, h = 841
block_size = 78
threshold = 15



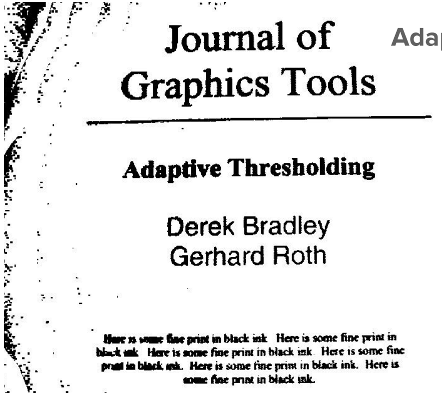
Entrada



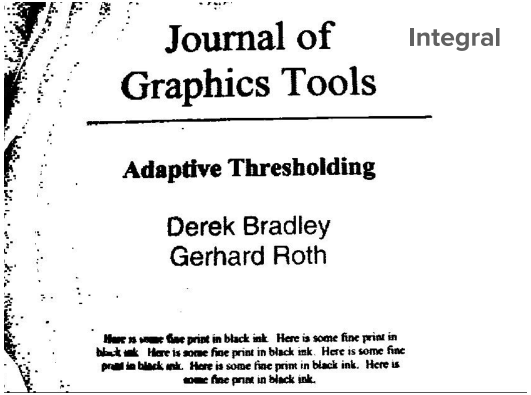
Otsu



Wellners



Adaptive

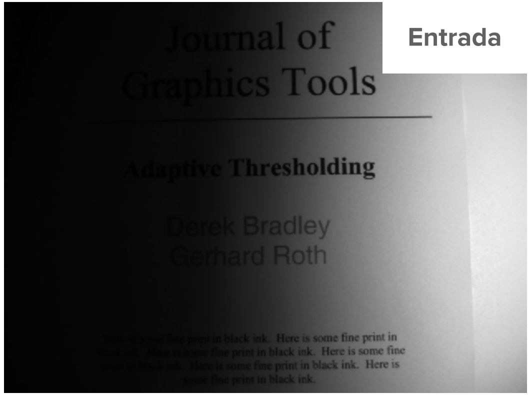


Integral

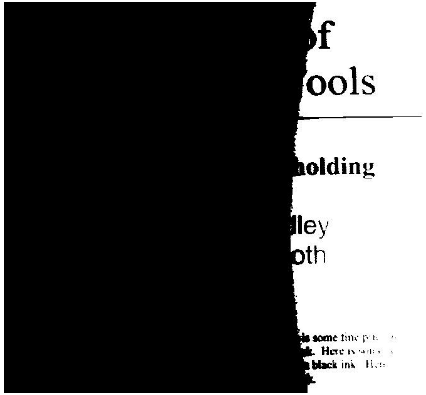
Algoritmo	Tempo	Num. Iterações
Otsu	1.1603	1049823
Wellners	0.4600	524784
Medium	8.0946	3017483040
Integral	2.4127	1048104

Exemplo 5

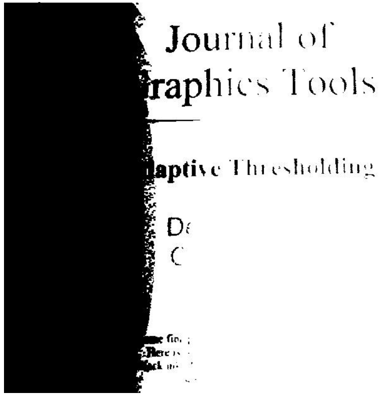
w = 624, h = 841
block_size = 78
threshold = 50



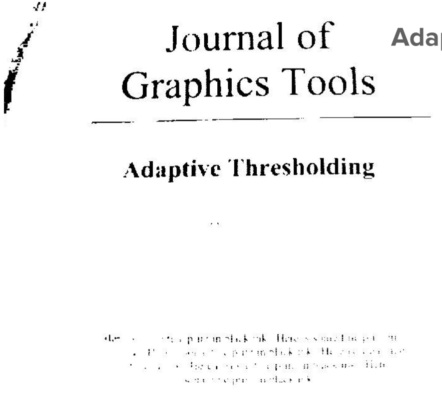
Entrada



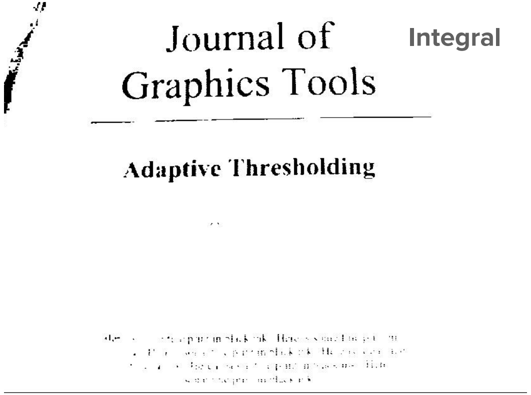
Otsu



Wellners



Adaptive

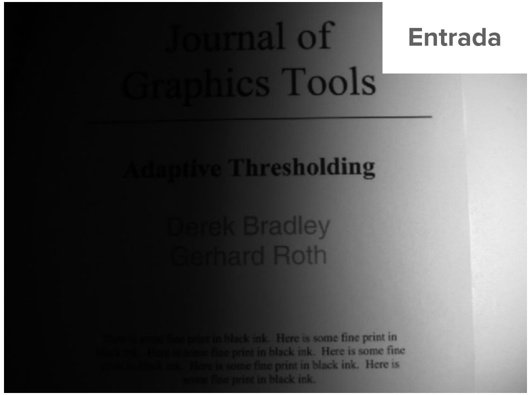


Integral

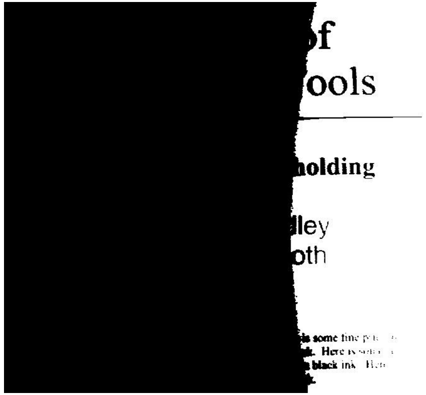
Algoritmo	Tempo	Num. Iterações
Otsu	1.2193	1049823
Wellners	0.5313	524784
Medium	9.6168	3017483040
Integral	2.6812	1048104

Exemplo 6

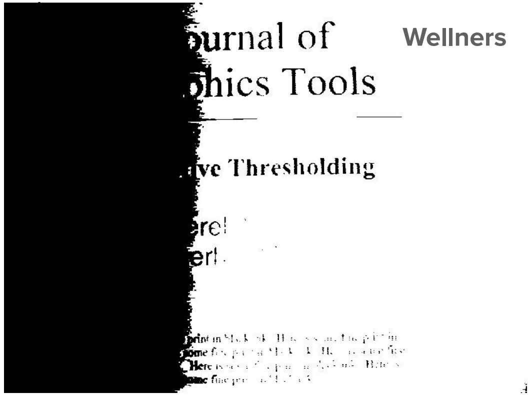
w = 624, h = 841
block_size = 78
threshold = 5



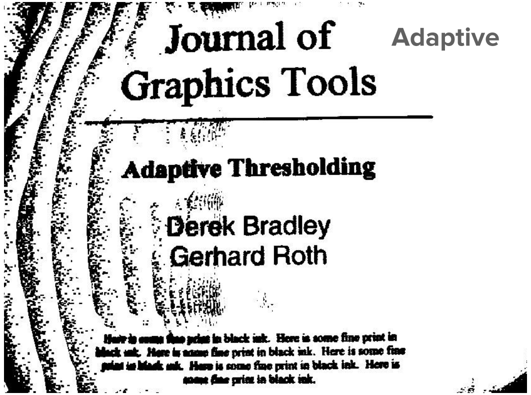
Entrada



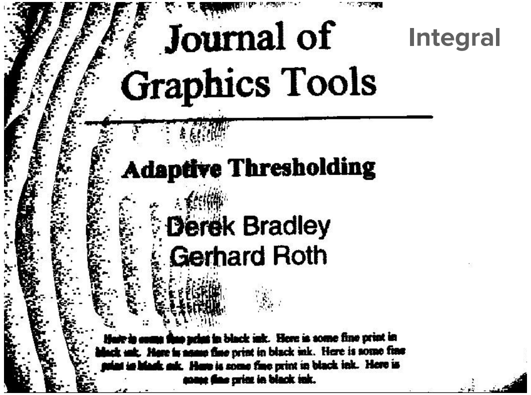
Otsu



Wellners



Adaptive

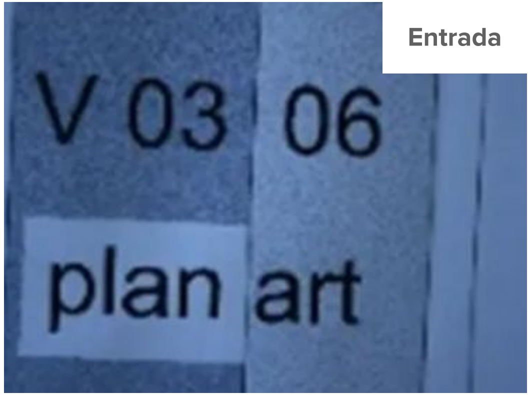


Integral

Algoritmo	Tempo	Num. Iterações
Otsu	1.2652	1049823
Wellners	0.4999	524784
Medium	7.6402	3017483040
Integral	2.4057	1048104

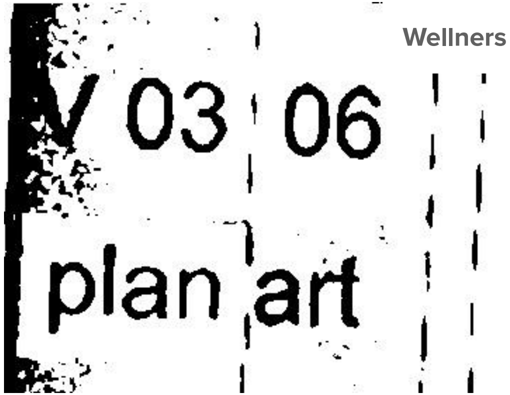
Exemplo 7

w = 278, h = 420
block_size = 34
threshold = 15

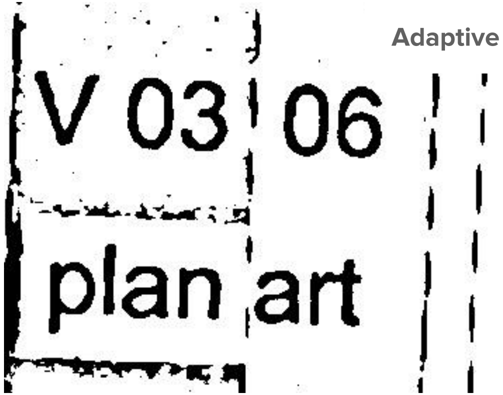


Entrada

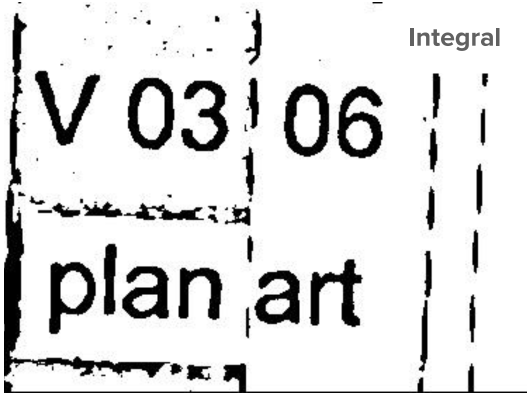
Otsu



Wellners



Adaptive

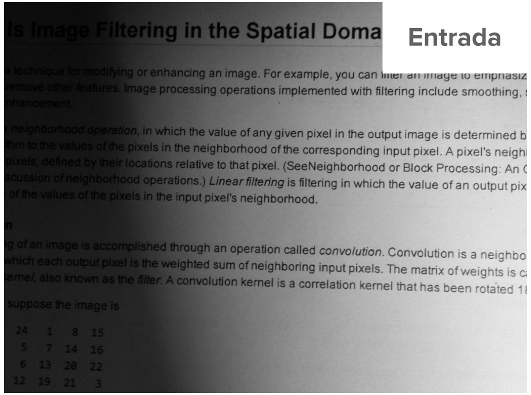


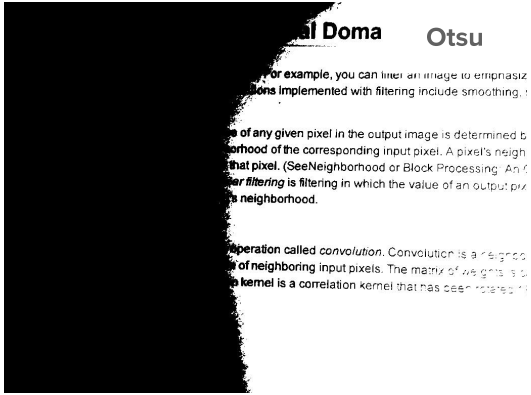
Integral

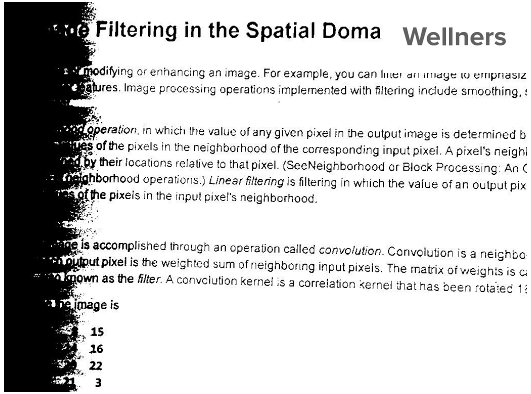
Algoritmo	Tempo	Num. Iterações
Otsu	0.3239	233775
Wellners	0.1099	116760
Medium	1.1802	127922964
Integral	0.6074	232823

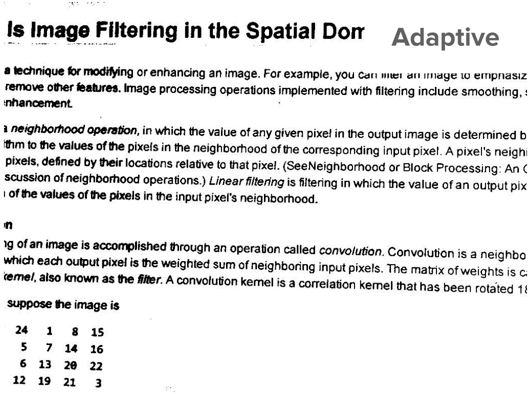
Exemplo 8

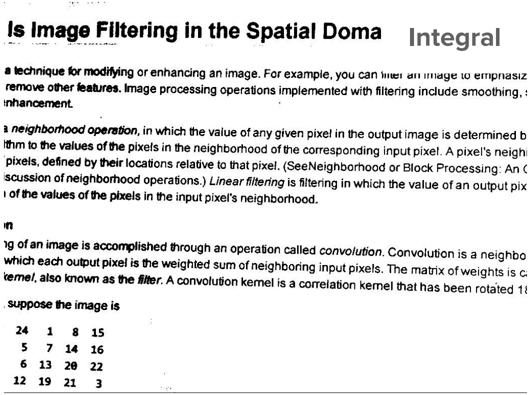
w = 773, h = 1378
block_size = 96
threshold = 15


Entrada


Otsu


Wellners


Adaptive


Integral

Algoritmo	Tempo	Num. Iterações
Otsu	2.2678	2130643
Wellners	0.9188	1065194
Medium	17.2660	9337746410
Integral	4.5984	2128238

Exemplo 9

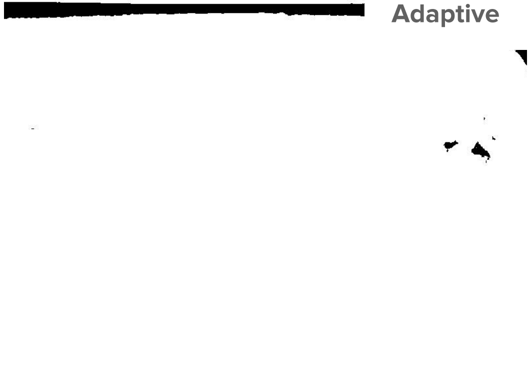
Imagem com alta intensidade
w = 599, h = 919
block_size = 74
threshold = 15

Entrada

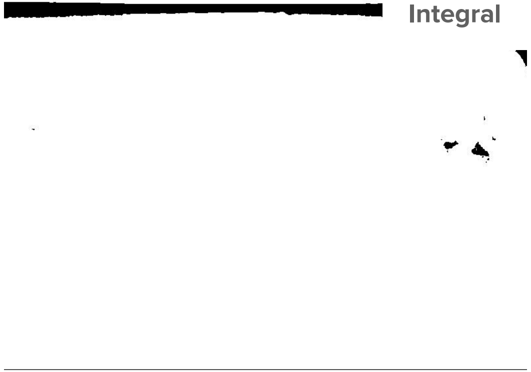
Otsu

Wellners

Adaptive



Integral



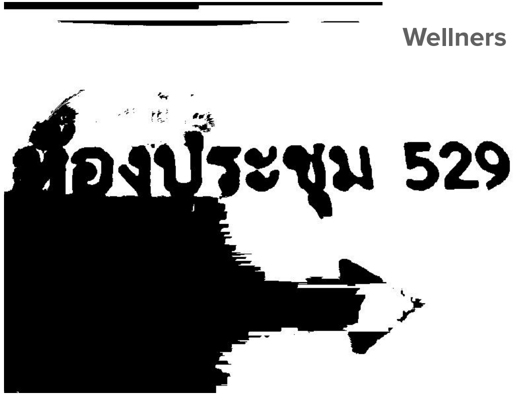
Algoritmo	Tempo	Num. Iterações
Otsu	1.2851	1101217
Wellners	0.4971	550481
Medium	9.8246	2859022481
Integral	2.4825	1099445

Exemplo 10

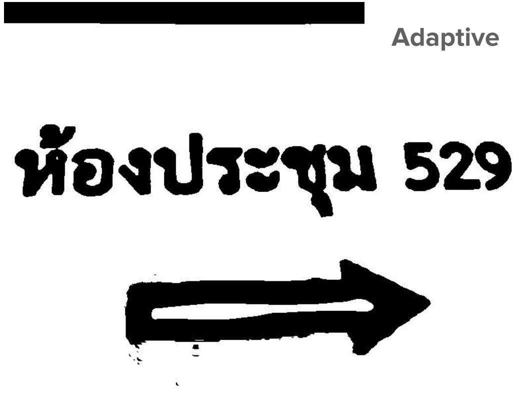
Imagem com baixa intensidade
w = 599, h = 919
block_size = 74
threshold = 15

Entrada

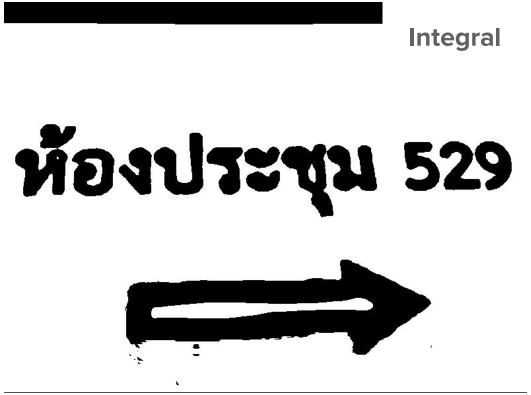
Otsu



Wellners



Adaptive



Integral

Resultados

Algoritmo	Vantagens	Desvantagens	Quando Usar
Otsu	Fácil implementação Baixo custo	Problemas ao lidar com iluminação	Histogramas com distribuição bimodal de intensidade Fundo uniforme
Wellners	Fácil implementação Adapta-se melhor com variação da iluminação	Depende da ordem de leitura dos pixels Baixa representação dos vizinhos	Pequenas variações de iluminação
Medium	Grande representação dos vizinhos Robusto quanto a variação de iluminação Grande parametrização	Custo computacional	Imagens com grande variação de iluminação
Integral	Eficiência computacional	Menos flexível Necessita de uma iteração a mais	Necessidade de alta performance: performance real-time

Limitações

- Baixa taxa de recall quando objeto de interesse está em áreas de alta intensidade
- Caso objeto de interesse seja maior do que S x S, ele pode ser classificado como fundo. Para resolver, basta aumentar o tamanho de S, porém pode perder detalhes da imagem
- Ajustes dos parâmetros Block size e Threshold interferem, porém percebeu-se que utilizar block_size = ⅛ da largura da imagem e threshold = 15 apresentou bons resultados
- Menos eficiente para imagens grandes:
    - Cálculo linear da imagem integral
    - Acesso a memória

Ponto de melhoria

- Computar métricas de qualidade da segmentação (taxa recall)
- Otimização do algoritmo
    - Podemos realizar a limiarização simultaneamente ao cálculo da imagem integral, pois a limiarização de um pixel, não exige que toda a matriz da imagem integral esteja completa

Referências

Bradley, D., & Roth, G. (2004). Adaptive thresholding using the integral image. Journal of Graphics Tools, 9(1), 11-17.

Pintaric, T. (2023, 14 de junho). Concept of Modified Adaptive Thresholding Using Integral Image to Decompose Text Under Illumination. Kittipop-P. https://kittipop-p.medium.com/concept-of-modified-adaptive-thresholding-using-integral-image-to-decompose-text-under-illumination-f8ced99e271