Trabalho 3 de Introdução ao Processamento de Imagem Digital

Nome: Paulo Junio Reis Rodrigues - RA: 265674

01/12/2020

1 Introdução

O objetivo deste trabalho, será aplicar os métodos de limiarização global e local em imagens monocromáticas, muito utilizado para segmentação de objetos. A primeira técnica, a limiarização global, consiste em utilizar um único valor de limiar para utilizar na imagem. Já o método local, consiste em aplicar um limiar cara pixel da imagem, e para gerá-lo, é necessário verificar a informação contida em cada vizinho de cada pixel da imagem. Abaixo estão alguns exemplos de limiarização local:

1. Método de Bernsen.
2. Método de Niblack.
3. Método de Sauvola e Pietaksinen.
4. Método de Phansalskar, More e Sabale.
5. Método de Contraste.
6. Método de Média.
7. Método de Mediana.

Neste trabalho, está sendo considerado para os dois casos de limiarização que: se um pixel $p(x, y)$ na imagem de entrada possuir um valor mais alto que o limiar, então o pixel $p(x, y)$ é classificado como objeto (cor preta), caso contrário, ele é considerado como fundo (cor branca).

Portanto, o intuito deste trabalho é, a partir de várias técnicas de limiarização em imagens monocromáticas, comparar todos os métodos utilizados, e mostrar os resultados encontrados durante os vários testes feitos.

Junto com este relatório, está sendo enviado o arquivo trabalho_3_265674.zip que possui dentro dele todos os arquivos que irão ser citados neste relatório.

2 Programa

Todos os programas foram implementados usando Python 3.8.5, com as bibliotecas Numpy 1.19.2, OpenCV 4.4.0.42 e Matplotlib 3.3.2.

3 Como executar

O programa pode ser executado através do script *trabalho_3.py*. Quando executado, obrigatoriamente deve ser enviado a uma imagem monocromática, porém caso queira, poderá ser enviado como argumento, 6 valores:

`--limiar: valor do limiar global que será aplicado, caso não seja enviado, o padrão será 128.`

`--mascara: valor do tamanho da máscara que será utilizada em todos os métodos locais, caso não seja enviado, o padrão será 15.`

```
--k: valor do k para alguns métodos locais, caso não seja enviado, o padrão será 0.5.

--R: valor do R para alguns métodos locais, caso não seja enviado, o padrão será 0.25.

--p: valor do p para alguns métodos locais, caso não seja enviado, o padrão será 2.

--q: valor do q para alguns métodos locais, caso não seja enviado, o padrão será 10.
```

Logo abaixo é mostrado um exemplo de execução do script:

```
python trabalho_3.py --limiar 130 --mascara 7 imagens/wedge.pgm
```

3.1 Entrada

O programa possui 7 entradas, a primeira é a imagem monocromático no formato *PGM* que será utilizada para aplicar os métodos, as outras 6 entradas, são os valores dos parâmetros que serão utilizados na geração dos resultados.

3.2 Saída

As saídas do programa são 8 imagens monocromáticas no formato *PGM*, uma para cada método de limiarização aplicado. Além disso, é gerado 9 histogramas que representam o nível de cinza das imagens, no caso é feito 1 histograma para a imagem de entrada, e mais 8 histogramas para cada limiarização aplicada.

4 Parâmetros utilizados

Todas as imagens utilizadas para execução do programa estão presentes no diretório imagens/, e todas estão disponíveis em *https://www.ic.unicamp.br/ helio/imagens_pgm/*.

As saídas do script executado, terão o nome do trabalho "trabalho_3" e logo depois o nome da limiarização utilizada, exemplo: trabalho_3_Global.pgm e todas estas saídas estarão no diretório outputs/. No caso dos histogramas gerados, todos estarão na pasta histogramas/ e os arquivos gerados possuem o nome de "histograma" mais o método utilizado, exemplo: histograma_Global.pgm.

5 Solução do exercício

Como já foi dito nas seções anteriores, no trabalho foi implementado 8 métodos de limiarização, sendo 1 deles global e 7 locais. No começo da solução, utiliza-se a função *imread* do *OpenCV* para a leitura das imagens a partir do diretório informado. As imagens serão lidas via escala de cinza, e no momento que a imagem é carregada ele já é convertida para valores decimais (Float64) utilizando a função da biblioteca *Numpy*, para que, durante o cálculo de limiarização não ocorra nenhum erro.

5.1 Limiarização global

O método de limiarização global é baseado em somente um valor de limiar, que é passado como parâmetro para a função. O método funciona da seguinte maneira, caso o valor do pixel $p(x, y)$ é maior que o limiar enviado, o pixel $p(x, y)$ resultante é considerado como objeto (cor preta) e caso o contrário, ele é considerado como fundo (cor branca). No programa o valor de limiar pode ser enviado como argumento na execução do script, ou caso nenhum valor seja enviado, o valor padrão será 128.

5.2 Limiarização local

Diferente do método global, na função local é gerada uma matriz de limiares antes da comparação com a imagem de entrada. Esta matriz é gerada da seguinte maneira, para cada pixel $p(x, y)$ é feito o cálculo (dependendo do método utilizado) do limiar considerando os seus vizinhos. Depois de gerar a matriz, é feita uma comparação pixel a pixel das duas imagens, e caso o valor da imagem

original for maior que a do limiar, o pixel $p(x, y)$ resultante será considerado como objeto (cor preta), caso o contrário, o pixel é considerado como fundo (cor branca).

Como o método local necessita dos vizinhos para calcular os limiares, foi feito um tratamento de borda para que no momento da geração dos valores dos limiares, não ocorra nenhum erro. O tratamento utilizado foi o de replicação de pixels, que consiste em replicar todos os pixels das bordas horizontais e verticais, para criar uma camada maior das bordas.

A seguir será mostrado todos os métodos utilizados para calcular os limiares da limiarização local. No método de Bernsen o limiar é calculado a partir desta fórmula:

$$T(x, y) = (z_{min} + z_{max})/2$$

No método de Niblack a fórmula é:

$$T(x, y) = \mu(x, y) + k * \sigma(x, y)$$

Já no método de Sauvola e Pietaksinen a formula é:

$$T(x, y) = \mu(x, y)[1 + k(\frac{\sigma(x, y)}{R} - 1)]$$

No método de Phansalskar, More e Sabale a formula é:

$$T(x, y) = \mu(x, y)[1 + p * \exp(-q\mu(x, y)) + k(\frac{\sigma(x, y)}{R} - 1)]$$

No método do constante, o valor do pixel é atribuindo como fundo ou objeto, dependendo se seu valor está mais próximo do máximo ou mínimo local, respectivamente. Porém, como o algoritmo trata esse valor como limiar, então é feita uma inversão dos valores para que quando a comparação do limiar for feita, não aconteça nenhum problema. Já os métodos da média e mediana, calculam o limiar baseado na média e na mediana dos vizinhos, respectivamente.

No final de cada método, é gerado o histograma e o valores de proporção de pixels pretos e brancos na imagem resultante. No caso, o histograma utiliza a função do *MatPlotLib* junto com a biblioteca *Numpy* para gerar os resultados, e os valores de proporção são calculados usando a biblioteca *Numpy*.

6 Resultados

6.1 Imagens de entrada

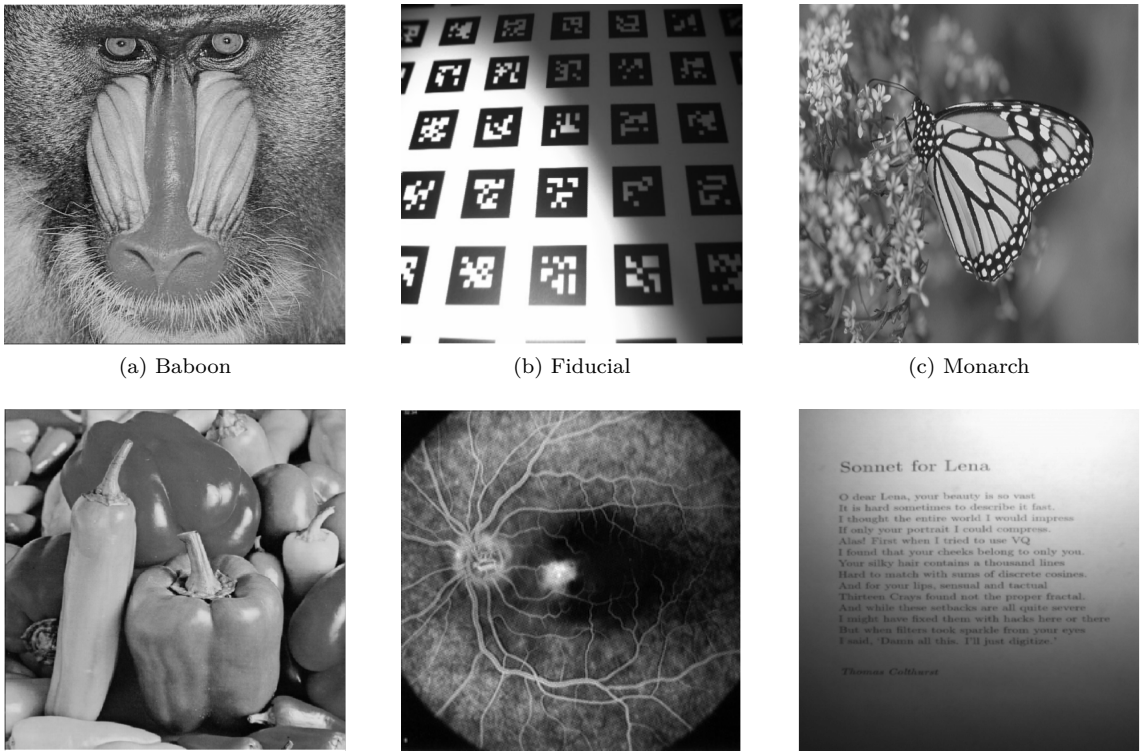
Como os resultados obtidos variam bastante de acordo com a imagem de entrada, primeiramente, necessita-se verificar qual é o padrão de cada imagem, para entendermos os resultados obtidos. Para verificar essas diferenças, os histogramas de níveis de cinza foram gerados para cada uma das 7 imagens de entrada, essas imagens podem ser encontradas na figura 1. Já os histogramas obtidos estão representados na figura 2.

Pode se notar algumas peculiaridades, a imagem do **baboon.pgm** por exemplo, possui um contraste bastante concentrado entre valores de 50 a 200, com alguns picos de níveis de cinza bem marcantes, sendo "fácil" a aplicação de uma limiarização global. Já imagem do **fiducial.pgm** por possui mais picos principais de níveis de cinza, é mais difícil aplicar uma limiarização global na imagem. Isso se dá pelo fato de haver uma má iluminação na imagem, causando esse efeito no histograma.

6.2 Resultados - limiarização global

Como já foi explicado nas seções anteriores, o método de limiarização global divide a imagem de acordo com um limiar. Para realizar alguns testes, o valor utilizado como limiar foi o 128, que é o valor padrão do algoritmo. Segue abaixo os valores de proporção de pixels pretos de cada imagem, após a limiarização:

1. Baboon: 0.5281
2. Fiducial: 0.3802



(a) Baboon (b) Fiducial (c) Monarch

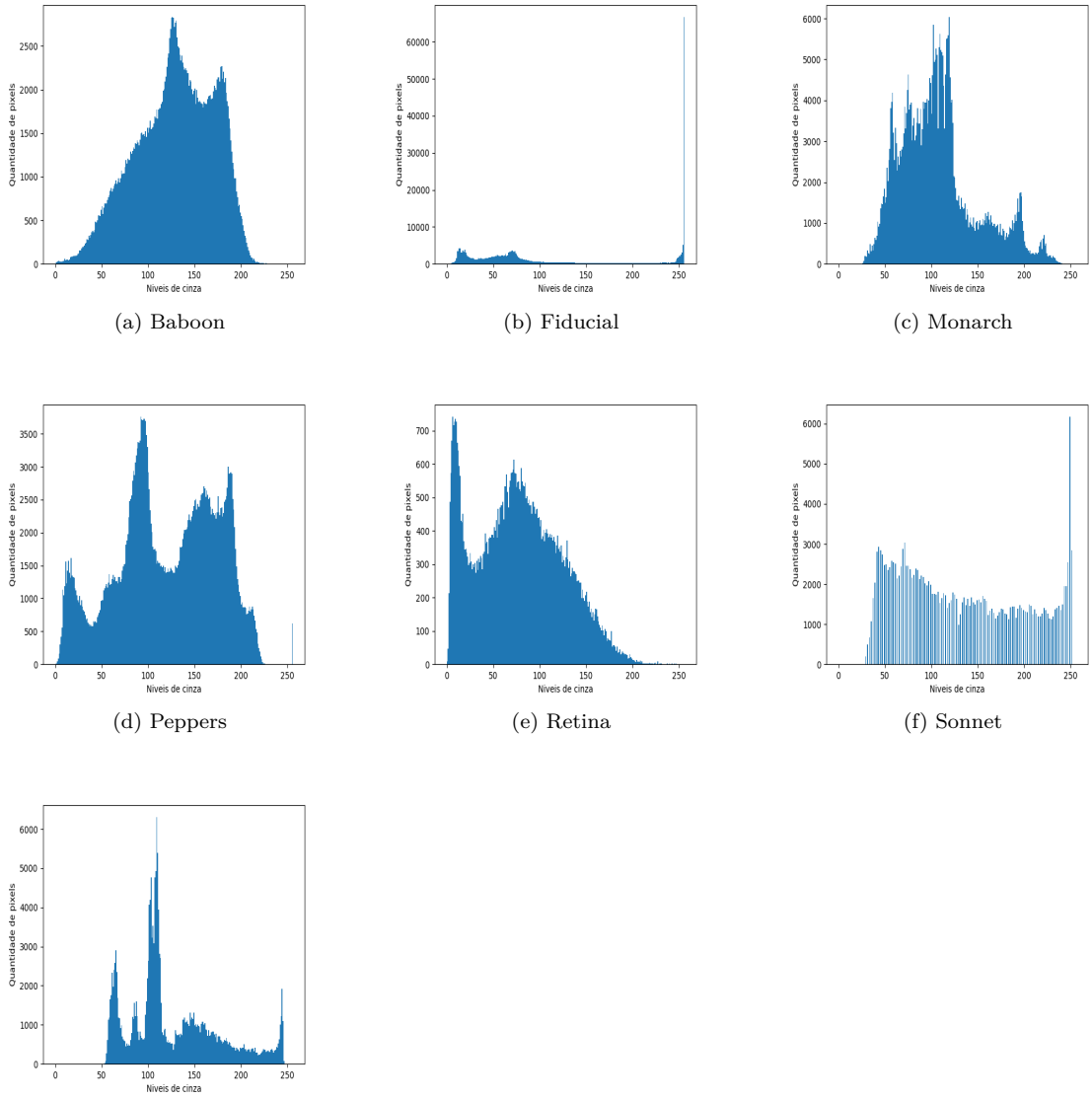
(d) Peppers (e) Retina (f) Sonnet

(g) Wedge

Figura 1: Imagens de entrada.

3. Monarch: 0.2206
4. Peppers: 0.4704
5. Retirna: 0.1525
6. Sonnet: 0.4794
7. Wedge: 0.3904

Como pode ser observado na figura 3, as imagens que possuem um problema de má iluminação *wedge.pgm, fiducial.pgm e sonnet.pgm* não obtiveram um bom resultado, pois a sombra gerada atrapalha o método a segmentar os objetos. Por outro lado, os resultados obtidos das outras imagens foi razoável. Entretanto, os resultados encontrados poderiam ser melhores, caso cada imagem tivesse seu próprio e único limiar, pois cada uma possui um contraste diferente.



(a) Baboon (b) Fiducial (c) Monarch

(d) Peppers (e) Retina (f) Sonnet

(g) Wedge

Figura 2: Histogramas para cada imagem de entrada.

6.3 Resultado - limiarização local

Como o objetivo dessa técnica é analisar cada região da imagem, podemos separar os testes em 2 seções, a primeira irá mostrar um resultado geral de todos os testes aplicados na imagem **peppers.pgm**, pois, esta imagem possui um melhor resultado nos testes realizados com todos os métodos. Logo depois, irá ser mostrado os resultado obtidos para as imagens com má iluminação com o método de **Sauvola e Pietaksinen**, no caso as imagens testadas foram *wedge.pgm, fiducial.pgm e sonnet.pgm.*

Em todos os testes, foram levados em consideração todos os valores padrões que os próprios autores indicaram para a execução.

6.3.1 Execução de todos os métodos

Como já discutido, os resultado obtidos da aplicação da limiarização local na imagem *peppers.pgm* se encontram na figura 4. Para gerá-los foi utilizado os seguintes parâmetros: Para o **Niblack** o valor $k = -0.2$, para o método do **Phansalskar, More e Sabale** foi utilizado os valores $k = 0.25, R = 0.5, p = 2$ e $q = 10$ e para o método **Sauvola e Pietaksinen** foi utilizado os valores

(a) Baboon (b) Fiducial (c) Monarch

(d) Peppers (e) Retina (f) Sonnet

(g) Wedge

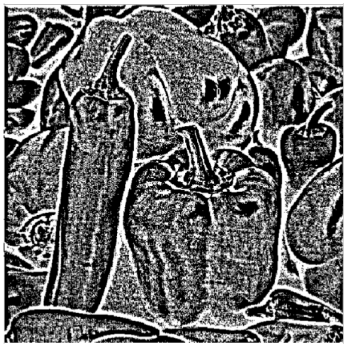
Figura 3: Execução do.

$k = 0.5$ e $R = 128$, o tamanho da mascara utilizada foi o padrão do algoritmo que é o valor 15. As proporções de níveis de pixels pretos estão descritos abaixo:

1. Bernsen: 0.5042
2. Niblack: 0.6138
3. Sauvola e Pietaksinen: 0.00000001
4. Phansalskar, More e Sabale: 0.00000001
5. Contraste:: 0.5042
6. Media: 0.5151
7. Mediana: 0.4590

Pelos resultados encontrados pode ser observado que cada um dos métodos gerou um resultado diferente, por exemplo, o resultado encontrado no método de **Bernsen** não conseguiu segmentar corretamente alguns dos objetos, deixando alguns bordas descontinuadas durante o processo.

(a) Bersen

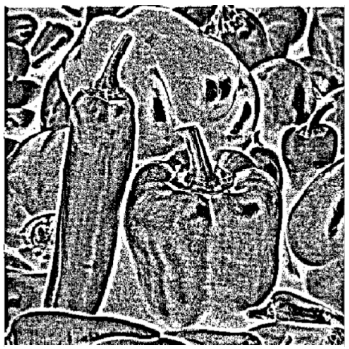


(b) Niblack

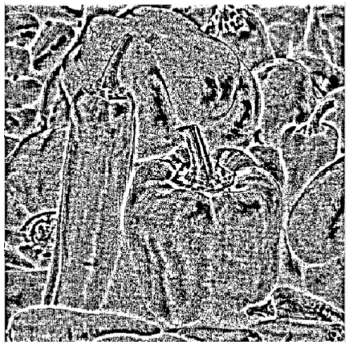
(c) Sauvola e Pietaksinen

(d) Phansalskar, More e Sabale

(e) Contraste



(f) Media



(g) Mediana

Figura 4: Execuções de métodos de limiarização local na imagem do **peppers.pgm**.

O método de **Niblack** obteve um resultado melhor do que o método anterior, deixando menos bordas descontinuadas. Porém o resultado apresenta mais ruídos pretos nos objetos. Já o método da média, aparentemente obteve um resultado parecido com o método de **Niblack**, porém com um pouco menos pixels pretos no resultado. Já o método da mediana obteve um resultado pior do que os demais, deixando a maioria das bordas na cor branca, não conseguindo segmentar bem os objetos.

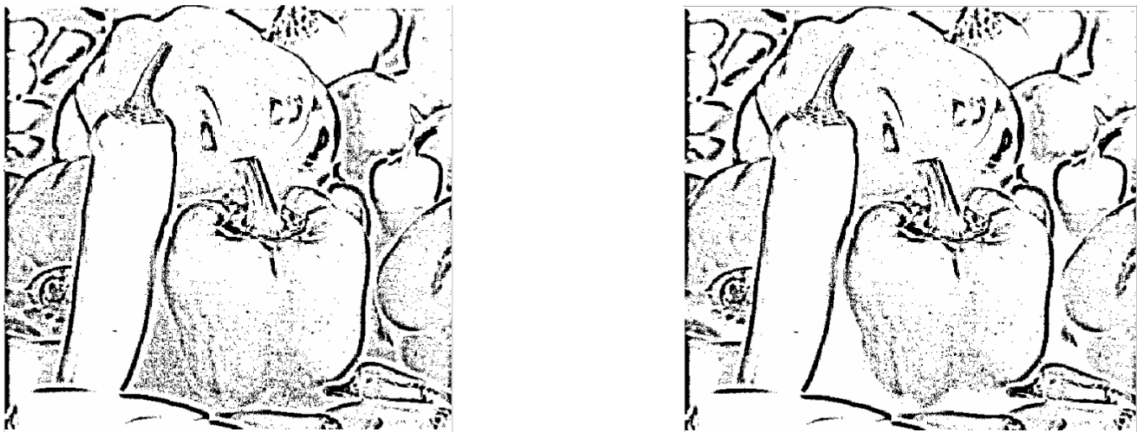
Outro método analisado é o de contraste, que por sinal obteve um resultado muito parecido com o método **Bernsen**, as proporções de pixels pretos estão bastante parecidas e as imagens geradas estão visualmente idênticas.

E por último temos os métodos de **Sauvola e Pietaksinen** e o de **Phansalskar, More e Sabale**, que neste caso gerarão resultados muito ruins, gerando somente 2 imagens em branco. Isso se dá pelo fato de que este primeiro método foi feito para imagens que possuem má iluminação, que não é o caso da imagem do **peppers.pgm**. Já o outro método foi feito para imagens de baixo contraste, porém a imagem de entrada possui um alto contraste. Entretanto, foram feitos outros testes que variam alguns valores de k e q, no caso -0.1 e 0.1 respectivamente, que por sua vez, geraram alguns resultados bem satisfatórios, fazendo uma segmentação muito boa nas imagens. A proporção de preto podem ser encontrada abaixo:

1. Sauvola e Pietaksinen: 0.1502

2. Phansalskar, More e Sabale: 0.1343

Os dois resultados foram bastante parecidos, porém é necessário verificar que o segundo método gerou menos ruído na imagem, deixando o resultado com menos pixels na cor preta na imagem. Os resultados podem ser observados na figura 5.



(a) Sauvola e Pietaksinen (b) Phansalskar, More e Sabale

Figura 5: Execuções de métodos de limiarização local na imagem do **peppers.pgm**, porém com valores diferentes da referencia bibliográfica.

Contudo, os métodos de limiarização local obtiveram resultados melhores, se comparado com o método de limiarização global.

6.3.2 Métodos para soluções de má iluminação

Para estes testes foram utilizados o método de **Sauvola e Pietaksinen**, pois o método foi feito para ser aplicado em imagens com má iluminação como: *wedge.pgm, fiducial.pgm e sonnet.pgm.* Para a primeira rodada de testes, foi utilizados os valores padrões de cada sugestão dos autores. E os resultados podem ser verificados na figura 6, e suas proporções de preto são essas:

(a) Fiducial (b) Wedge (c) Sonnet

Figura 6: Execução do método **Sauvola e Pietaksinen** para a limiarização local.

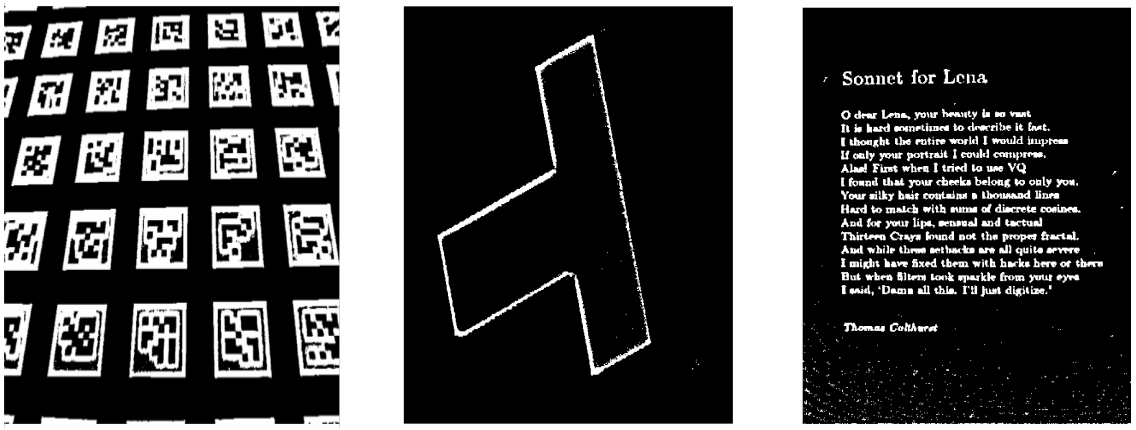
1. Fiducial: 0.8232

2. Wedge: 1.0

3. Sonnet: 0.9998

Os resultados encontrados não foram muito bons de acordo com os parâmetros sugerido pelos autores. Somente a imagem **fiducial.pgm** obteve um resultado satisfatório. Entretanto, em alguns outros testes foram feitos a partir de outros valores para k, e os resultados podem ser observados na figura 7. E suas proporções de preto são essas:

1. Fiducial ($n = 15, k = 0,6, R = 128$): 0.6985
2. Wedge ($n = 15, k = 0,75, R = 128$): 0.9730
3. Sonnet ($n = 15, k = 0,4, R = 128$): 0.9030



(a) Fiducial (b) Wedge (c) Sonnet

Figura 7: Execução do método **Sauvola e Pietaksinen** para a limiarização local, porém agora, com parâmetros mais calibrados.

Agora com os novos parâmetros, podemos observar que o método realmente funciona, as imagens geradas apresentam alguns pequenos ruídos, porém como os testes não foram exaustivos, ao mudar alguns valores de outras variáveis, estes ruídos podem ser reduzidos. O texto da imagem **sonnet.pgm** não é muito legível, com isso, foi feito outro teste utilizando uma mascara menor no valor de 5 e com um k um pouco maior no valor de 0.075, e o resultado obtido pode ser observado na figura 8. Já a imagem **wedge.pgm** obteve um ótimo resultado, deixando quase todo objeto segmentado. Outros parâmetros como N e R foram modificados, porém os resultados encontrados não foram muito diferentes.

7 Limitações e conclusões

Neste trabalho podemos tirar várias conclusões sobre os métodos de limiarização aplicados. Primeiramente o método de limiarização global é bastante útil se o limiar escolhido separar bem em duas partes do contraste da imagem, entretanto, como em algumas imagens possuem má iluminação, o efeito gerado não será muito bom por causa do sombreamento.

Nos métodos de limiarização local, vale destacar o método de **Sauvola e Pietaksinen**, que conseguiu limiarizar e assim segmentar alguns dos objetos nas imagens que a limiarização global não conseguiu. Porém, estas funções locais possuem um tempo computacional maior do que a global, e além disso, possuem vários parâmetros, para se calcular os limiares, deixando mais difícil de encontrar um resultado perfeito para cada imagem específica.

Por fim, alguns testes foram feitos, porém não é possível realizar todas as combinações dos parâmetros, para encontrar o melhor resultado para cada imagem.

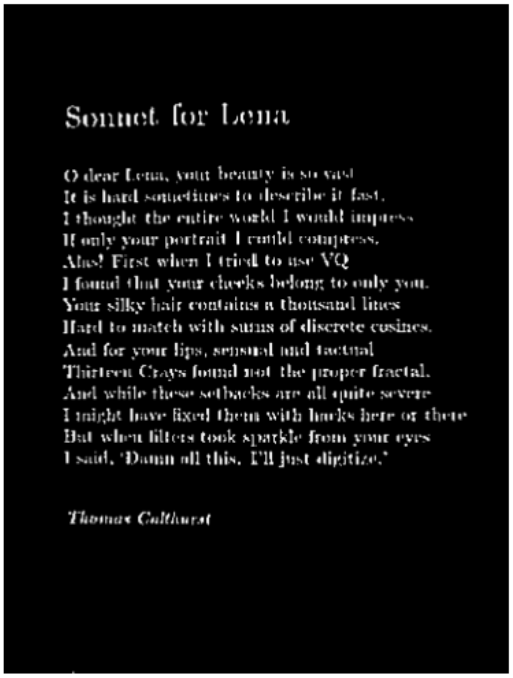


Figura 8: Execução do método **Sauvola e Pietaksinen** para a limiarização local, na imagem do **sonnet.pgm** com os valores de $k = 0.075$, $n = 5$ e $R = 128$.