Trabalho 3 de Introdução ao Processamento de Imagem Digital

Nome: Sabrina Beck Angelini - **RA**: 157240

16 de Maio de 2018

1 Introdução

O objetivo desse trabalho é implementar algoritmos para alinhamento automático de imagens de documentos.

Um problema comum que ocorre na digitalização é o desalinhamento do documento, ou seja, o posicionamento do papel com uma inclinação diferente do eixo do digitalizador. Na análise de documentos, que tem como objetivo converter a informação contida nas imagens em texto editável, o alinhamento correto do documento é essencial para o funcionamento adequado de sistemas de reconhecimento ótico de caracteres.

Junto desse relatório está sendo entregue o arquivo SABRINA_BECK_ANGELINI_157240.tar, que possui todos os arquivos citados nesse relatório. A entrega desses arquivos foi feita no Google Classroom.

2 O programa

O programa foi implementado com Python 3.5.2. As bibliotecas utilizadas e suas respectivas versões foram Skimage 0.10.1, Numpy 1.11.0 e Matplotlib 1.5.1.

2.1 Como executar

O programa pode ser executado através do script `alinhar.py`.

O script recebe como argumento uma imagem de entrada, o algoritmo que se deseja executar e o nome da imagem de saída. Um exemplo de execução é mostrado a seguir:

```
python3 alinhar.py img/neg_4.png hough out/neg_4_hough.png
```

O argumento que indica o algoritmo que se deseja executar pode assumir dois valores:

- horizontal: implementação baseada em projeção horizontal
- hough: implementação baseada na transformada de Hough

2.2 Entrada

As entradas do programa são: uma imagem colorida RGB ou monocromática, o algoritmo desejado usando uma das duas palavras chave *horizontal* ou *hough* e a imagem de saída onde o resultado deve ser salvo.

2.3 Saída

A saída do programa é uma imagem *PNG* rotacionada de forma a alinhar o documento presente na imagem de entrada. Além disso, na saída padrão é impresso o ângulo de inclinação detectado na imagem original do documento.

3 Parâmetros Utilizados

Todas as imagens utilizadas para execução do programa estão presentes no diretório `imgs/`, dentre elas estão as imagens informadas no enunciado do trabalho (http://www.ic.unicamp.br/ helio/imagens_inclinadas_png/). Ainda foram adicionadas as imagens *pos_ 90.jpeg* e *neg_ 90* que são documentos inclinados em 90º e -90º, respectivamente. Além disso, foram adicionadas as imagens *pos_ 204* e *neg_ 208.png* que são documentos inclinados em 204º e -208º, respectivamente. Por último, também foi testada uma imagem de documento não inclinado.

As saídas obtidas com essas imagens e textos foram armazenadas no diretório `examples/`.

4 Solução

4.1 Leitura da Imagem

A imagem de entrada é lida com a ajuda da função `skimage.io.imread` que armazena as diferentes bandas/canais de cores em um `numpy.ndarray` de três dimensões. A imagem colorida é armazenada em um `numpy.ndarray` de MxNx3 como RGB.

Depois de lida, a imagem é convertida para níveis de cinza através da função `skimage.color.rgb2gray`, após isso, a imagem em níveis de cinza é passada para cada implementação de alinhamento que faz seus próprios pré processamentos.

4.2 Técnica baseada em Projeção Horizontal

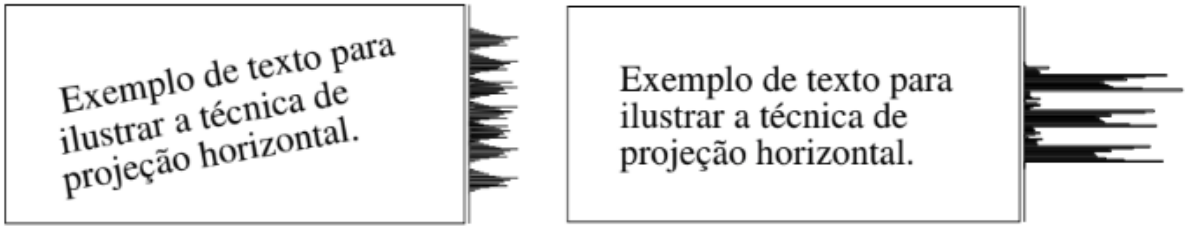
A projeção horizontal de uma imagem binária é definida como a soma dos pixels existentes em cada linha do objeto.

Como a projeção horizontal está definida como uma imagem binária, a imagem em níveis de cinza foi transformada em uma imagem binária através da aplicação de um limite. Todos os pixels com nível de cinza menor que esse limite foram transformados para a cor branca e todos os pixels com nível de cinza maior ou igual a esse limite foram transformados para a cor preta.

O limite é escolhido através do método `skimage.filter.threshold_otsu` que se baseia no método de Otsu. O método de Otsu busca exaustivamente por um limite que minimiza a variância intra-classe (variância dentro da classe). Sendo cada classe definida como os pixels abaixo do limite e os pixels acima do limite.

O cálculo da inclinação do documento baseado em projeção horizontal é feito variando o ângulo de rotação da imagem de entrada de 1º em 1º começando de -90º indo até 90º e calculando a projeção horizontal da imagem em cada um dessas configurações. O ângulo de inclinação do documento é aquele que otimiza uma função objetico calculada em cima do perfil da projeção horizontal.

A imagem de exemplo a seguir mostra um documento originalmente inclinado em 10º e sua versão após uma correção de alinhamento. É possível ver como a projeção horizontal da imagem resultante possui maior amplitude que a original, além de ter sido compactada em um intervalo menor de linhas da imagem.

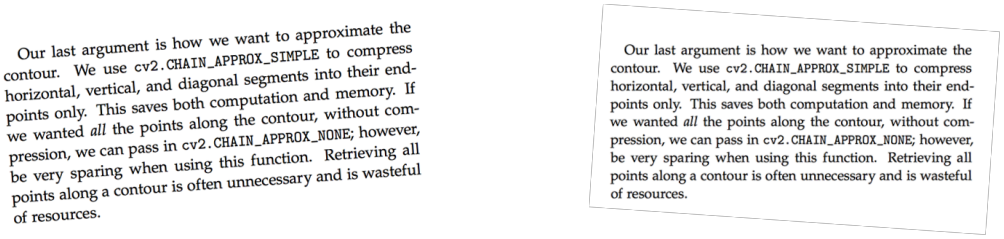


(a) Original: inclinada em 10º (b) Após correção

Figura 1: Projeção Horizontal de um documento originalmente inclinado e após o alinhamento automático.

A função objetivo utilizada é a soma dos quadrados das diferenças dos valores em células adjacentes do perfil de projeção.

O resultado da técnica pode ser visto na imagens mostradas a seguir:



(a) Original (b) Alinhado

Figura 2: Alinhamento de documento inclinado em -4º utilizando a técnica baseada em Projeção Horizontal.

(a) Original (b) Alinhado

Figura 3: Alinhamento de documento inclinado em 41º utilizando a técnica baseada em Projeção Horizontal.

Podemos ver que a técnica funciona bem para ambas as imagens que foram alinhadas de forma horizontal.

Ao utilizar o Tesseract OCR (Optical Character Recognition) para transformar as imagens ajustadas em texto editável temos os seguintes resultados para a imagem originalmente inclinada em -4º:

O texto editável extraído da imagem original com inclinação de 4º:

```
Our last argummt is how we want to approximate the
contour. We use cv2.ClIAIN_APPRDX,SIHPLE to compress
horizontal, venical, and diagonal segments into their end–
points only. This saves both computation and memory. if
```

```
we muted all the points along the contour, without com-
e can pass in cv2.Cl-1AIN_APPRUX_NUNE; however,

pression, w
be very sparing when using this function. Retrieving all

points along a contour is often unnecessary and is wasteful
of resources
```

O texto editável extraído da imagem após alinhamento automático com técnica baseada em projeção horizontal:

```
Our last argument is how we wam to approximate the
contour. We use cv2.CHAIN.APPRDX\_SIMPLE to compress
horizontal, vertical, and diagonal segments into their end-
points only. This saves both computation and memory. If
we wanted all the points along the contour, without com-
pression, we can pass in (NZ. CEAILAP'PRDXJIDNE; however,
be very sparing when using this function. Retrieving all
poinis along a contour is oflen unnecessary and is wasteful
of resources.
```

Podemos notar uma melhora enorme no resultado do Tesseract OCR após o alinhamento automático, apesar de ainda ter cometido alguns erros.

4.3 Técnica baseada na Transformada de Hough

A técnica baseada na Transformada de Hough localiza as linhas formadas pelas regiões de texto. A transformada de Hough converte pares de coordenadas *(x, y)* da imagem em curvas nas coordenadas polares (ρ, θ).

Os picos no plano de Hough são formados por pixels pretos alinhados na imagem (linhas dominantes) e permitem identificar o ângulo θ de inclinação do documento.

Antes de aplicar a técnica baseada na transformada de Hough, é aplicado o detector de bordas Canny (`skimage.filter.canny`) que gera um mapa de bordas. Esse mapa de bordas é utilizado posteriormente pela transformada para encontrar linhas na imagem.

A transformada de Hough e os picos no plano de Hough são calculados através das funções `skimage.transform.hough_line` e `skimage.transform.hough_line_peaks`. Dos ângulos encontrados pela função `skimage.transform.hough_line_peaks`, selecionamos a moda, ou seja, o ângulo de maior frequência encontrado. Esse ângulo indica a inclinação do documento.

No entanto, aparentemente, as linhas detectadas pela transformada foram linhas no sentido vertical do texto. Isso se deve à natureza do alfabeto latino, cuja maioria de seus caracteres (33/52) tem pelo menos um traço vertical. Essa característica fez com que os picos no plano de Hough fossem as linhas orientada verticalmente em relação ao texto. A imagem a seguir ilustra essa característica do alfabeto:

Para identificar a inclinação com relação à horizontal foi realizado o seguinte cálculo: ângulos negativos encontrados adiciona-se 90^o e ângulos positivos encontrados subtrai-se 90^o.

O resultado da técnica pode ser visto na imagens mostradas a seguir:

Podemos ver que a técnica funciona bem para ambas as imagens que foram alinhadas de forma horizontal.

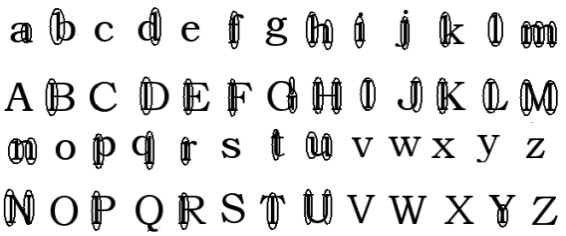
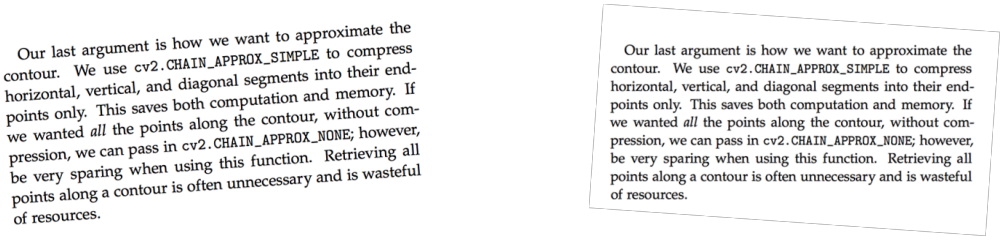


Figura 4: Alfabeto latino com maioria das letras possuindo pelo menos um traço vertical.



(a) Original (b) Alinhado

Figura 5: Alinhamento de documento inclinado em -4° utilizando a técnica baseada na Transformada de Hough.

Ao utilizar o Tesseract OCR (Optical Character Recognition) para transformar as imagens ajustadas em texto editável temos os seguintes resultados para a imagem originalmente inclinada em -4°:

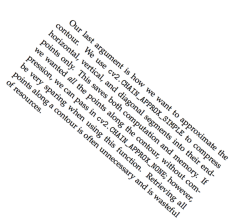
O texto editável extraído da imagem original com inclinação de 4°:

```
Our last argummt is how we want to approximate the
contour. We use cv2.ClIAIN_APPRDX,SIHPLE to compress
horizontal, venical, and diagonal segments into their end-
points only. This saves both computation and memory. if
we muted all the points along the contour, without com-
e can pass in cv2.Cl-1AIN_APPRUX_NUNE; however,

pression, w
be very sparing when using this function. Retrieving all

points along a contour is often unnecessary and is wasteful
of resources
```

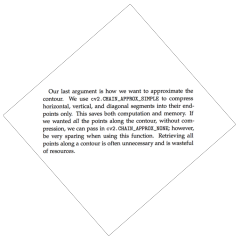
O texto editável extraído da imagem após alinhamento automático com técnica baseada em projeção horizontal:

```
Our last argument is how we wam to approximate the
contour. We use cv2.CHAIN.APPRDX\_SIMPLE to compress
horizontal, vertical, and diagonal segments into their end-
points only. This saves both computation and memory. If
we wanted all the points along the contour, without com-
pression, we can pass in (NZ. CEAILAP'PRDXJIDNE; however,
be very sparing when using this function. Retrieving all
poinis along a contour is oflen unnecessary and is wasteful
of resources.
```

Podemos notar uma melhora enorme no resultado do Tesseract OCR após o alinhamento automático, apesar de ainda ter cometido alguns erros.



(a) Original



(b) Alinhado

Figura 6: Alinhamento de documento inclinado em 41º utilizando a técnica baseada na Transformada de Hough.

4.4 Comparação das duas técnicas

A seguinte tabela exibe os ângulos, em graus, encontrados para as imagens de teste utilizadas:

Imagem	Ângulo Esperado	Projeção Horizontal	Transformada de Hough
neg_4	-4	-4	-4.02234636872
neg_28	-28	-28	-28.156424581
pos_24	24	24	24.1340782123
pos_41	41	41	41.2290502793
sample1	14	14	14.0782122905
sample2	-6	-6	-6.03351955307
pos_90	90	-90	89.4972067039
neg_90	-90	-90	89.4972067039
neg_208	208	-28	-28.156424581
pos_204	204	24	24.1340782123
0	0	0	0.0

Tabela 1: Ângulos esperados e calculados para cada imagem testada.

Comparando as duas técnicas através da tabela anterior vemos que os resultados de ambas são bem parecidos para todas as imagens testadas. Além disso, se compararmos os textos gerados pelo Tesseract OCR mostrados nas duas subseções anteriores para a imagem originalmente inclinada em -4º, vemos que o resultado também é o mesmo para ambas as técnicas. Em questões de resultado, temos que ambas as técnicas aplicadas são satisfatórias para o alinhamento automático de documentos e variam muito pouco em seus resultados.

Com relação ao tempo de execução de cada implementação, a seguinte tabela mostra o tempo de execução, em segundos, do programa para cada técnica e cada imagem de teste utilizada:

Imagem	Projeção Horizontal	Transformada de Hough
neg_4	1s	0s
neg_28	2s	0s
pos_24	2s	0s
pos_41	3s	0s
sample1	0s	0s
sample2	7s	0s
pos_90	1s	0s
neg_90	1s	0s
neg_208	3s	0s
pos_204	2s	0s
0	0s	0s

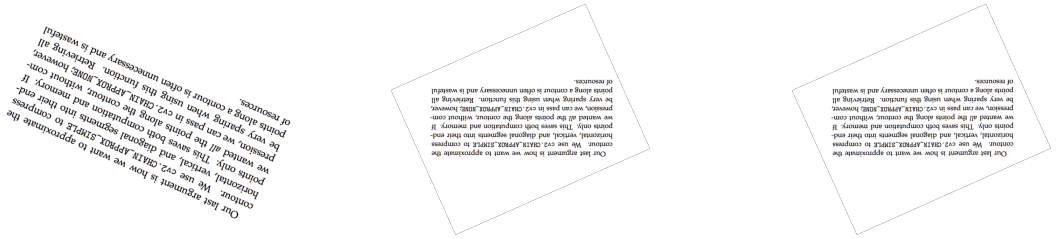
Tabela 2: Tempo de execução de cada técnica para cada imagem testada.

Pela tabela acima podemos ver que a técnica baseada na Transformada de Hough é mais rápida que a técnica baseada na Projeção Horizontal. Isso é evidenciado no caso da imagem *sample1* que é a maior imagem no conjunto de testes e leva 7s no alinhamento automático por projeção horizontal contra 0s com a transformada de Hough.

5 Limitações

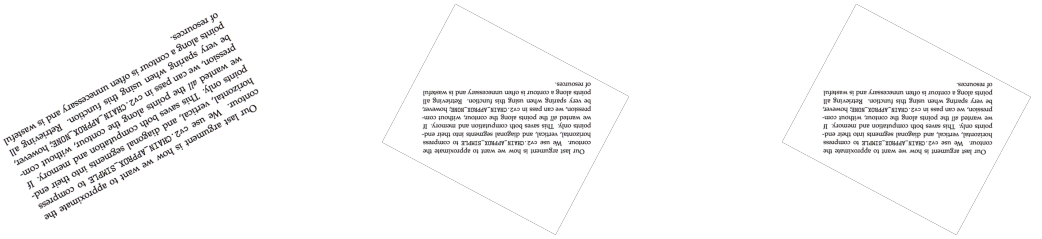
O programa aceita imagens *PNG* e *JPEG* coloridas ou monocromáticas contendo documentos de textos.

Os algoritmos implementados conseguem com sucesso alinhar o documento em uma imagem na horizontal. No entanto, os algoritmos são incapazes de identificar se o texto está de ponta cabeça ou não, além disso, inclinações sempre são identificadas no intervalo [-90º, 90º], ou seja, imagens inclinadas fora desse intervalo acabam de ponta de cabeça. As imagens a seguir ilustam essa limitação:



(a) Original (b) Alinhado com Projeção Horizontal (c) Alinhado com Transformada de Hough

Figura 7: Alinhamento de documento inclinado em 204º.



(a) Original (b) Alinhado com Projeção Horizontal (c) Alinhado com Transformada de Hough

Figura 8: Alinhamento de documento inclinado em -208º.

6 Bibliografia

A. Papandreou and B. Gatos, "A Novel Skew Detection Technique Based on Vertical Projections", <http://www.iapr-tc11.org/archive/icdar2011/fileup/PDF/4520a384.pdf>