Introdução ao Processamento de Imagem Digital - MC920
Trabalho 3

Daniel Henriques Pamplona - RA: 260401

1 Descrição geral

O objetivo deste trabalho é a análise de documentos através da conversão automática da informação contida nas imagens de documentos em texto editável. Para resolver o problema do desalinhamento de um certo documento na hora da digitalização, utilizamos dois tipos de técnicas para detecção e correção da inclinação de documentos: uma pela projeção horizontal e a outra pela transformada de Hough.

2 Projeção Horizontal

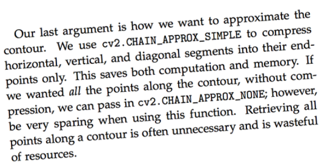
A projeção horizontal é um dos métodos utilizados para detectar a inclinação da imagem. Ele se baseia na análise do perfil de projeção horizontal, que é obtido somando os valores de intensidade de pixels ao longo de cada linha da imagem. Este perfil reflete a distribuição de texto ou elementos gráficos na imagem.

Para determinar o ângulo de inclinação, a imagem é rotacionada iterativamente em diferentes ângulos dentro de um intervalo pré-definido. Para cada ângulo, calcula-se o perfil de projeção horizontal e uma função objetivo, que mede a variação entre as linhas consecutivas do perfil. O ângulo que maximiza esta função objetivo é considerado o ângulo de inclinação da imagem.

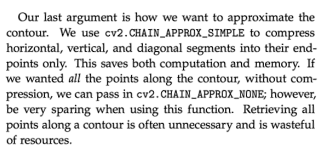
Após a detecção do ângulo, a imagem pode ser corrigida rotacionando-a no sentido oposto ao ângulo detectado, resultando em um documento alinhado horizontalmente.

2.1 Resultados para cada imagem:

2.1.1 Imagem neg_4


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_SIMPLE to compress horizontal, vertical, and diagonal segments into their endpoints only. This saves both computation and memory. If we wanted *all* the points along the contour, without compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(a) Imagem original


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_SIMPLE to compress horizontal, vertical, and diagonal segments into their endpoints only. This saves both computation and memory. If we wanted *all* the points along the contour, without compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(b) Imagem corrigida

```
Our last argument is how we want to approximate the
contour. We use v2. CHAIN_APPROX_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 cv2 \CHAIN_APPROX_NONE; however,
be very sparing when using this function. Retrieving all
points along a contour is often unnecessary and is wasteful

of resources.
```

Figura 2: Texto capturado pela imagem original

```
Our last argument is how we want to approximate the
contour. We use cv2.CHAIN_APPROX_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 cv2.CHAIN_APPROX_NONE; however,
be very sparing when using this function. Retrieving all
points along a contour is often unnecessary and is wasteful
of resources.
```

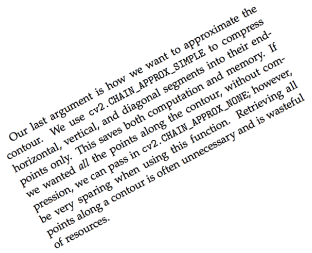
Figura 3: Texto capturado pela imagem corrigida

```
Skew angle: -4.00 degrees
```

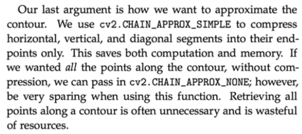
Ângulo de inclinação

Para uma imagem com uma inclinação baixa, podemos ver que a captura do texto na imagem original é basicamente a correta, com apenas um erro na penúltima linha que tem uma quebra de linha.

2.1.2 Imagem neg_28


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(a) Imagem original


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(b) Imagem corrigida

```
ot og or oe ane go xe) er we
oe we Rog ot oe oe one 0 om
woe \oo oe oe ro ap? oe ok
on oo: aye oe Se oe or %
we = a we on gore we oe oe co
|| oo ee ot oe ‘as ees
we wos ee \i oe we ee ≫
om, wo? to ˜ oe a oa \oe
3S ‘oe
```

Figura 5: Texto capturado pela imagem original

```
Our last argument is how we want to approximate the
contour. We use cv2.CHAIN_APPROX_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 cv2.CHAIN_APPROX_NONE; however,
be very sparing when using this function. Retrieving all
points along a contour is often unnecessary and is wasteful
of resources.
```

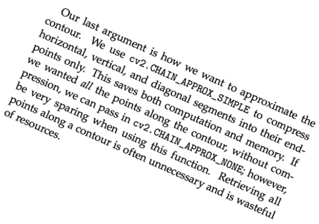
Figura 6: Texto capturado pela imagem corrigida

```
Skew angle: -28.00 degrees
```

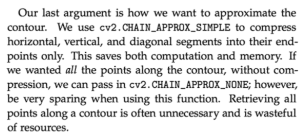
Ângulo de inclinação

Veja que, com uma inclinação maior, o Tesseract OCR não consegue captar direito o texto na imagem original, gerando um texto sem sentido. Ao corrigir a imagem com o ângulo de rotação calculado, é lido corretamente o texto.

2.1.3 Imagem pos_24


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(a) Imagem original


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(b) Imagem corrigida

```
Ou, fase 5 Bune, ds hoy, We Wang to Le ina the
\Rou, We Use cra Si fo ores
one, erty ng Gna, een, Lnty Ley, eng.
Poings Oniy, Thi Save, bu, 2 eta ang Hoon, ip
We antes ay the Poing, Pong he mtoyp Pithou: COn,
Lion ve Can Pass tn Cv, Rete rote,
be Very, Paring When "Sing this eretign \ent aly
Pings long a ntou, as ter, me 2g Is 8steg,
OF, Sore
```

Figura 8: Texto capturado pela imagem original

```
Our last argument is how we want to approximate the
contour. We use cv2.CHAIN_APPROX_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 cv2. CHAIN_APPROX_NONE; however,
be very sparing when using this function. Retrieving all
points along a contour is often unnecessary and is wasteful
of resources.
```

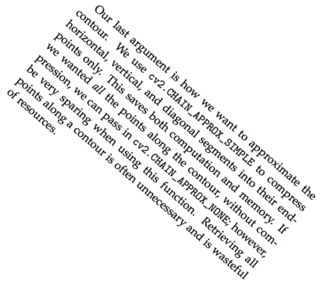
Figura 9: Texto capturado pela imagem corrigida

```
Skew angle: 24.00 degrees
```

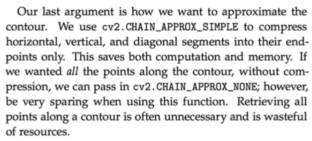
Ângulo de inclinação

Parecido com o item anterior, com uma angulação parecida, o texto é lido incorretamente.

2.1.4 Imagem pos_41


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(a) Imagem original


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(b) Imagem corrigida

```
gage
fon My, Q
fringing se
ae 86% hb Rm
& My, % yy bey Yi
ag ton We hee at
Pring a. eo
6g he Mao tage Gee
RU, Poy. & he Sop, ey My
ep Ng Rng ray 2 6% ha
eg Me jas, by p, og. Mr
$ %, 1%, 84, @ 63 Soy, ey
ne eg oy plan tg ®
ng ag? x 04 Sry ins
Be iy Te Be % \et
fog ae" Co, Men Na
Cat, a oy
SO ORION
0 We, He, Ge
ey, Uo, ee, % en
1 le, ago. °
an beg 4 tag Te 4%
2% 0, %, &
"2 Styg \Hey py,
tng Ye
ag ey
Gy,
Za
```

Figura 11: Texto capturado pela imagem original

```
Our last argument is how we want to approximate the
contour. We use cv2.CHAIN_APPROX_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 cv2.CHAIN_APPROX_NONE; however,
be very sparing when using this function. Retrieving all
points along a contour is often unnecessary and is wasteful
of resources.
```

Figura 12: Texto capturado pela imagem corrigida

```
Skew angle: 41.00 degrees
```

Ângulo de inclinação

Veja agora que, com uma inclinação maior ainda em relação as outras, o texto captado da imagem é maior ainda, e mais errado ainda. Ajustando com o ângulo de inclinação, o texto é lido perfeitamente.

2.1.5 Imagem partitura

(a) Imagem original

(b) Imagem corrigida

```
= es Tox a
Ds, be "Py
Reg Mn
Xe =
re Fie Ry )
\SS AZ \SS
im,
=
tS, 2
ie
SQ, Ry. by
22> oO
2,
```

Figura 14: Texto capturado pela imagem original

```
Pasaaaccan
aa eee unaae.
cricthiid =taar
= Bd tong
Tumse
Pu Lereeeeric es
BB eeretetite
```

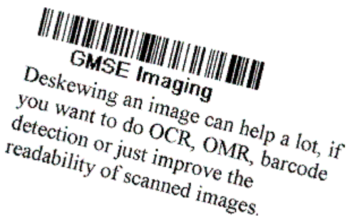
Figura 15: Texto capturado pela imagem corrigida

```
Skew angle: 25.70 degrees
```

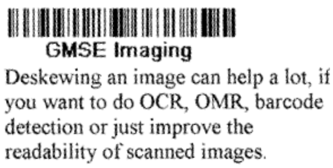
Ângulo de inclinação

Para essa imagem, como não há texto, não há o que ser lido. Porém, como existem as linhas da partitura, o código calcula o ângulo pra ajeitar as partituras na horizontal pra tentar ler algo. As imagens acima mostram o que ele interpreta da imagem, e o ângulo que ele rotaciona a imagem.

2.1.6 Imagem sample1



(a) Imagem original



(b) Imagem corrigida

```
HTM H UH ay
Mn my My
Deskewing an image can help a fox, ig
You Want to do OCR, OmrR, barcode
tection, OF jue improve the
"eAdabiity OF Scanned IMages
```

Figura 17: Texto capturado pela imagem original

```
Se

GMSE Imaging
Deskewing an image can help a lot, if
you want to do OCR, OMR, barcode
detection or just improve the
readability of scanned images.
```

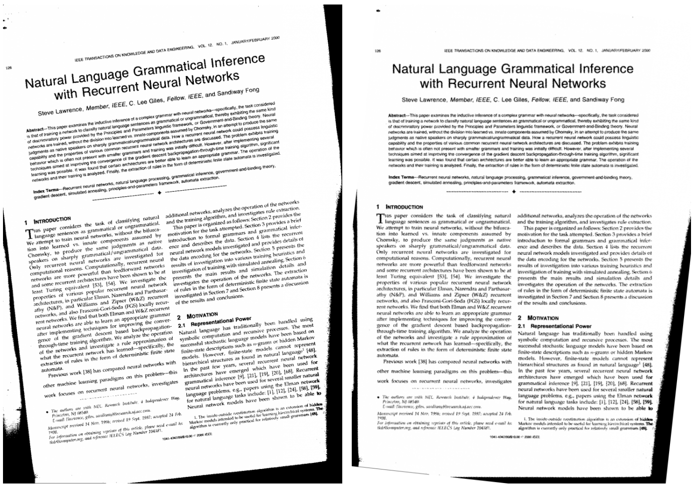
Figura 18: Texto capturado pela imagem corrigida

Skew angle: 14.10 degrees

Ângulo de inclinação

O ângulo de inclinação dessa imagem é um pouco mais inclinada que o a primeira imagem e menos inclinado que a segunda. Por isso, consegue ainda assim ler algumas coisas na imagem original. A sua rotação torna a leitura perfeita.

2.1.7 Imagem sample2


Natural Language Grammatical Inference with Recurrent Neural Networks

Steve Lawrence, Member, IEEE, C. Lee Giles, Fellow, IEEE, and Sandiway Fong

(a) Imagem original (b) Imagem corrigida

```
- eerie MONEE MO DN ON sexe non meen
Natural Language Grammatical Inference
Steve Lawrence, Member. IEEE, C. Lee Giles, Fellow. 1EEE, and Sandiway Fong:
pans T8008 oe  coneen te piiniyaaoae
poe are scope ra a ses Segoe
Satcnr te vac ee seo ecm = ere nr
omc sgt es rea dy OI vn at rea
rare wie aera ce a a a
oes rea ee moma neve oot)
sy tc ae amt a ee on
Sow enone at ect wi ee po
Som an sc a sr
caren gn a Face ot nem om min
vena nec or ron gone ge rg Oo
we ance se cen
eee
4 jwTRODUCTION
baad some recurrent Fropitectares Have DEST, ‘shown 40 De at cen igation of wang Mt matted annealing Sect 6
properties of Varin Tpopular fecucreny sewer netvork wwestigates UN ‘operation of (ME smetwrks. THe extraction
Peep tures in particular EINE ed Pathos of eS 6 re etree i INE tomate
rae enane 22 y oe Talent ea Pacem :
ner jamplementing WN IE Tor umproving the conver 2 MOTIVATION
Se gh gat Sa ming eat fapesectaon 7o
Througtrtime training Sigortam. We analy re operation Natural TAMU98Y Thas traditionally Peer handled vsing,
pare ee of ralles i the FOF fot deterministic inte ‘gate finite-state desception> ‘such as reprams Of ‘hidden Markov 4
automata, \ models. However finite-state ‘models cannot Ft yresent
or (8 has compare Pe) torts with Praeict Sucur SN se aoe
Ce eno cu us et se en Re Os Ut te gece
: a mea en A Sarurl
" \a The entra tt NEG rece tte. + degen tay, NBOABE problems, © PRPS using the Eman, network
a Tengen poe ene Ea 9
cuca mn
```

Figura 20: Texto capturado pela imagem original

```
= me eer re NOERENONE OO OREN, Yh 0.1, sn ae
Natural Language Grammatical Inference
‘Steve Lawrence, Member, EEE, C. Lee Giles, Fellow, /EEE, and Sandiway Fong
De rere Sie ee ee ae
Rosaleerutans banaymmatiouety oun as pisesiote postal Seam astiegts meeioe
SSusreet pom pena by no Papeete ounces gsc tensed 6 Gomnoroe Bang ny ere
santa tntcins wnesbe Sisenveitemnive mom ecyntre enced ancy hercometepeare beta
fryeon ts rev penn cs rap pommaauapaneta don Nort mere ese os poe Nas
Sono pepo elas tr meen coraennt oom cand Te rauek ote Ta
baat et pen tema ganar et Vans ry BS. Noms penne sre
‘Sospon eda ncing ts cowpane soe pases asoh corpeeseecepeies nang spsesapanet
Perera ly ge hed are mepeeponpen cpap pont foe
Sane acer ecnnypvnaand recy ho asso ana artless We ae tacts ged
int Tames esa, ed wean peti, puma ec, govt eon,
porsipeteiorinpiclringenionetsh maisaren Leprol
ein se aay eR
1 IntRopucTiON
Tis re sete te a of casing strana nos ans boat he news
Coen melons me gromeicd ot eapetiatccl snd fe weeny dpmaten aad overages tide maacion
Weatttmpt ws tan neural newer, wiht te bree. "Ths papers obanand on flows Seton? pone
thn io"larcd ve ante compenans ssemed by medina rte ah ascnpnd Seton 9 prvi fd
Soop mensat toe mawerin Oe mmol Sar Saatentens ahae menipu el postions oot
```

```
Leqceies aie Compaen, weed patente os atoms ony te
networks are more powerful than feedforward networks results of investigation into various training heuristics and
Jeast Turing equivalent (53), 154). We investigate the presents the main results and simulation details and
Properties of various popular recurrent neural network investigates the operation of the networks. The extraction
Siy (NED): ane Wins ice OEE) ecren sls he fom of determin este awa
ete and ho Fase Gi-seo FO) louay cur rene n Section 7 nd
te retorts We find thet bth Elman aed Wc caren
after implementing, techniques for improving the conver- 2 MOTIVATION
fice i the patent decel based brclprpagaen- 2.1 Raprosantationel Powee
Uhrough-time training, algorithm. We analyze the operation Nistural language has traditionally been handled using,
of te retwors and Uren re eprocinatn ct Natal ngage hae wadlioaly bom bantled eng
wate mearen newark has eared pectcay tee mba empuation and recunine pres. The mort
Ginter cheno oro accent eee sn |
Previous work [38] has compared neural networks with nicrarchical structures as found in natural language' (48).
sla iee tee mega wank bee Seon toes Bt
work focuses on recurrent neural networks, investigates grammatical inference {9} [21], {19}, (20), [68]. Recurrent
: Scruton ave ben esd for fver serrata
© The autre are svidh NEC Resonch institute, + Iadqwrudence Way, language problems, eg. papers using the Elman network
Print, Wf S40 for natural language tasks include: (1), {12}, (241, (58), {99}
```

Figura 21: Texto capturado pela imagem corrigida

```
Skew angle: -6.30 degrees
```

Ângulo de inclinação

Nesse caso, como a imagem é muito grande e o texto também é, pouco se consegue detectar de escrita, tanto na imagem original quanto na corrigida.

3 Transformada de Hough

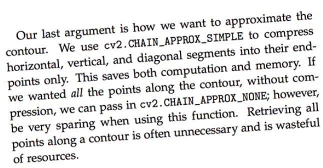
A Transformada de Hough é empregada para identificar linhas predominantes na imagem, que podem ser usadas para calcular o ângulo de inclinação. O processo começa com a aplicação de um detector de bordas, como o algoritmo de Canny, para destacar as bordas da imagem. Em seguida, a Transformada de Hough é utilizada para identificar linhas retas nas bordas detectadas. Cada linha é representada por dois parâmetros: a distância perpendicular da linha à origem (ρ) e o ângulo de inclinação (θ).

Os ângulos das linhas detectadas são então analisados, e a mediana desses ângulos é calculada para determinar o ângulo de inclinação predominante da imagem. Este ângulo é usado para corrigir a inclinação, rotacionando a imagem no sentido oposto.

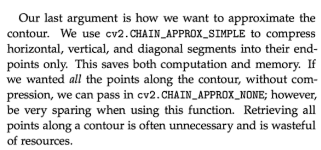
A análise de cada imagem é basicamente a mesma feita anteriormente pois os outputs são os mesmos (com os mesmos argumentos pois são os mesmos ângulos), com exceção das imagens partitura, sample1 e sample2 que são discutidos ao fim dos resultados da transformada.

3.1 Resultados para cada imagem:

3.1.1 Imagem neg_4


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_SIMPLE to compress horizontal, vertical, and diagonal segments into their endpoints only. This saves both computation and memory. If we wanted *all* the points along the contour, without compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(a) Imagem original


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_SIMPLE to compress horizontal, vertical, and diagonal segments into their endpoints only. This saves both computation and memory. If we wanted *all* the points along the contour, without compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(b) Imagem corrigida

```
Our last argument is how we want to approximate the
contour. We use v2. CHAIN_APPROX_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 cv2 \CHAIN_APPROX_NONE; however,
be very sparing when using this function. Retrieving all
points along a contour is often unnecessary and is wasteful

of resources.
```

Figura 23: Texto capturado pela imagem original

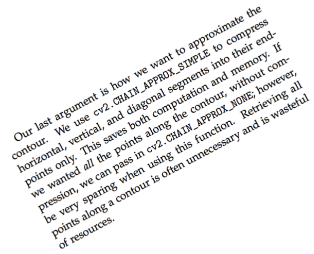
```
Our last argument is how we want to approximate the
contour. We use cv2.CHAIN_APPROX_SIMPLE to compress
horizontal, vertical, and diagonal segments into their end-
points only. This saves both computation and memory. If
we wanted all the points along the contour, without com-
pression, we can pass in cv2. CHAIN_APPROX_NONE; however,
be very sparing when using this function. Retrieving all
points along a contour is often unnecessary and is wasteful
of resources.
```

Figura 24: Texto capturado pela imagem corrigida

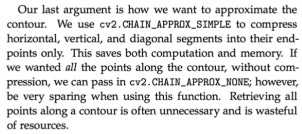
```
Skew angle: -4.00 degrees
```

Ângulo de inclinação

3.1.2 Imagem neg_28


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(a) Imagem original


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(b) Imagem corrigida

```
ot og or oe ane go xe) er we
oe we Rog ot oe oe one 0 om
woe \oo oe oe ro ap? oe ok
on oo: aye oe Se oe or %
we = a we on gore we oe oe co
|| oo ee ot oe ‘as ees
we wos ee \i oe we ee ≫
om, wo? to ~ oe a oa \oe
3S ‘oe
```

Figura 26: Texto capturado pela imagem original

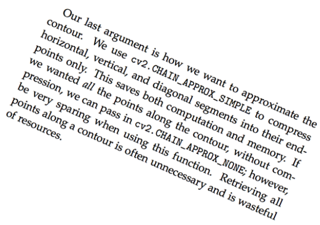
```
Our last argument is how we want to approximate the
contour. We use cv2.CHAIN_APPROX_SIMPLE to compress
horizontal, vertical, and diagonal segments into their end-
points only. This saves both computation and memory. If
we wanted all the points along the contour, without com-
pression, we can pass in cv2. CHAIN_APPROX_NONE; however,
be very sparing when using this function. Retrieving all
points along a contour is often unnecessary and is wasteful
of resources.
```

Figura 27: Texto capturado pela imagem corrigida

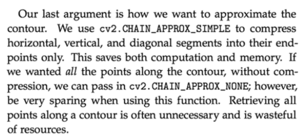
```
Skew angle: -28.00 degrees
```

Ângulo de inclinação

3.1.3 Imagem pos_24


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(a) Imagem original


Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(b) Imagem corrigida

```
Ou, fase 5 Bune, ds hoy, We Wang to Le ina the
\Rou, We Use cra Si fo ores
one, erty ng Gna, een, Lnty Ley, eng.
Poings Oniy, Thi Save, bu, 2 eta ang Hoon, ip
We antes ay the Poing, Pong he mtoyp Pithou: COn,
Lion ve Can Pass tn Cv, Rete rote,
be Very, Paring When "Sing this eretign \ent aly
Pings long a ntou, as ter, me 2g Is 8steg,
OF, Sore
```

Figura 29: Texto capturado pela imagem original

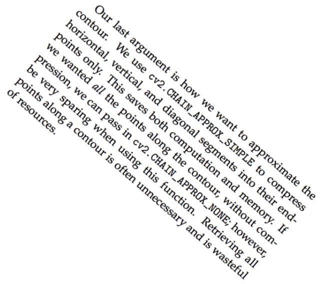
```
Our last argument is how we want to approximate the
contour. We use cv2.CHAIN_APPROX_SIMPLE to compress
horizontal, vertical, and diagonal segments into their end-
points only. This saves both computation and memory. If
we wanted all the points along the contour, without com-
pression, we can pass in cv2. CHAIN_APPROX_NONE; however,
be very sparing when using this function. Retrieving all
points along a contour is often unnecessary and is wasteful
of resources.
```

Figura 30: Texto capturado pela imagem corrigida

```
Skew angle: 24.00 degrees
```

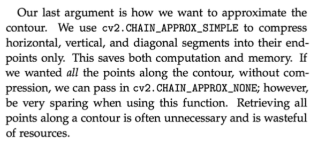
Ângulo de inclinação

3.1.4 Imagem pos_41



Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(a) Imagem original



Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_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 cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

(b) Imagem corrigida

```
gage
fon My, Q
fringing se
ae 86% hb Rm
& My, % yy bey Yi
ag ton We hee at
Pring a. eo
6g he Mao tage Gee
RU, Poy. & he Sop, ey My
ep Ng Rng ray 2 6% ha
eg Me jas, by p, og. Mr
$ %, 1%, 84, © 63 Soy, ey
ne eg oy plan tg ®
ng ag? x 04 Sry ins
Be iy Te Be % \et
fog ae" Co, Men Na
Cat, a oy
SO ORION
0 We, He, Ge
ey, Uo, ee, % en
1 le, ago. °
an beg 4 tag Te 4%
2% 0, %, &
"2 Styg \Hey py,
tng Ye
ag ey
Gy,
Za
```

Figura 32: Texto capturado pela imagem original

```
Our last argument is how we want to approximate the
contour. We use cv2.CHAIN_APPROX_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 cv2.CHAIN_APPROX_NONE; however,
be very sparing when using this function. Retrieving all
points along a contour is often unnecessary and is wasteful
of resources.
```

Figura 33: Texto capturado pela imagem corrigida

```
Skew angle: 41.00 degrees
```

Ângulo de inclinação

3.1.5 Imagem partitura

(a) Imagem original

(b) Imagem corrigida

```
= es Tox a
Ds, be "Py
Reg Mn
Xe =
re Fie Ry )
\SS AZ \SS
im,
=
tS, 2
ie
SQ, Ry. by
22> oO
2,
```

Figura 35: Texto capturado pela imagem original

```
eee asaee
Beaune.
iether
== Diet fog
Bvesirs sic Sori ras
iG Cree eric
Beeecetel itr
```

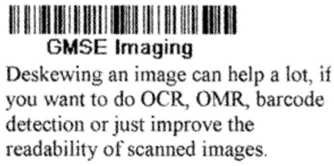
Figura 36: Texto capturado pela imagem corrigida

```
Skew angle: 26.00 degrees
```

Ângulo de inclinação

3.1.6 Imagem sample1

(a) Imagem original



(b) Imagem corrigida

```
HTM H UH ay
Mn my My
Deskewing an image can help a fox, ig
You Want to do OCR, OmrR, barcode
tection, OF jue improve the
"eAdabiity OF Scanned IMages
```

Figura 38: Texto capturado pela imagem original

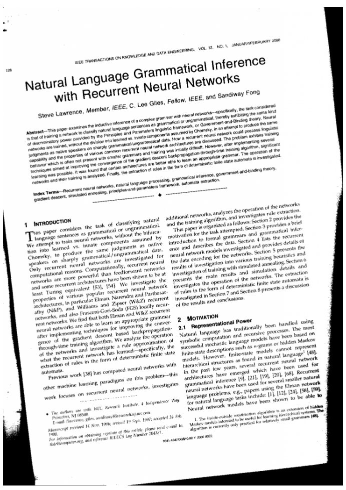
```
TV IARECEMI AAT

GMSE Imaging
Deskewing an image can help a lot, if
you want to do OCR, OMR, barcode
detection or just improve the
readability of scanned images.
```

Figura 39: Texto capturado pela imagem corrigida

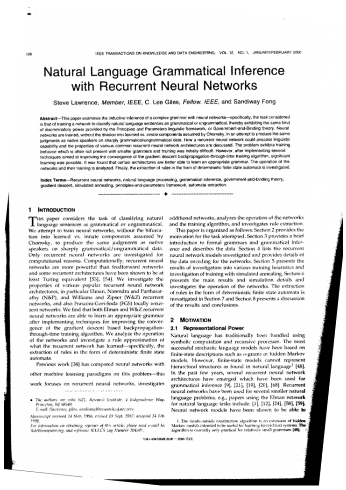
```
Skew angle: 14.00 degrees
```

Ângulo de inclinação

3.1.7 Imagem sample2



(a) Imagem original



(b) Imagem corrigida

```
- eerie MONEE MO DN ON sexe non meen
Natural Language Grammatical Inference
Steve Lawrence, Member. IEEE, C. Lee Giles, Fellow. 1EEE, and Sandiway Fong:
pans T8008 oe  coneen te piiniyaaoae
poe are scope ra a ses Segoe
Satcnr te vac ee seo ecm = ere nr
omc sgt es rea dy OI vn at rea
rare wie aera ce a a a
```

```
oes rea ee moma neve oot)
sy tc ae amt a ee on
Sow enone at ect wi ee po
Som an sc a sr
caren gn a Face ot nem om min
vena nec or ron gone ge rg Oo
we ance se cen
eee
4 jwTRODUCTION
baad some recurrent Fropitectares Have DEST, ‘shown 40 De at cen igation of wang Mt matted annealing Sect 6
properties of Varin Tpopular fecucreny sewer netvork wwestigates UN ‘operation of (ME smetwrks. THe extraction
Peep tures in particular EINE ed Pathos of eS 6 re etree i INE tomate
rae enane 22 y oe Talent ea Pacem :
ner jamplementing WN IE Tor umproving the conver 2 MOTIVATION
Se gh gat Sa ming eat fapesectaon 7o
Througtrtime training Sigortam. We analy re operation Natural TAMU98Y Thas traditionally Peer handled vsing,
pare ee of ralles i the FOF fot deterministic inte ‘gate finite-state desception> ‘such as reprams Of ‘hidden Markov 4
automata, \ models. However finite-state ‘models cannot Ft yresent
or (8 has compare Pe) torts with Praeict Sucur SN se aoe
Ce eno cu us et se en Re Os Ut te gece
: a mea en A Sarurl
" \a The entra tt NEG rece tte. + degen tay, NBOABE problems, © PRPS using the Eman, network
a Tengen poe ene Ea 9
cuca mn
```

Figura 41: Texto capturado pela imagem corrigida

```
. {ees mmAcnON8. ON OWLEDNE O DATA NOREEN VEL 1. 10.1, MANYFEBRY 200
Natural Language Grammatical Inference
with Recurrent Neural Networks
Steve Lawrence, Member, JEEE, C. Lee Giles, Fellow, IEEE, and Sandiway Fong
panrechTe pr umes tudinal aconpls gant wh co tea petly bea sed
a ae ere tay ncralojagy arto: partake o copunancl foray anion oe ta oe
eee ate wine Prep oa Pantawr apne tones @ Gonrewrt nes Beary nosy, Ne
See rena siacireieed es was crgrara enredhy Goma naman oars be ame
jit cpsien ss not poorascouegemeaiel dan, Nort weer rool ah Gd pasts ROSE
Se eer ener Seer eeat pp teama samcoom or Sarat Te pases Danie a
See er Tae nr cen ene goomer ae room bes ea roe ne opens sat
eae ae eee cmrpans ate poset cn mcepluanvcaprins Yon oars op teat
esis cere aneats tans: can canoes ov tens sae sao cope pene Te egoen oh
SSre’ranr nase ped Sac cums cotta bent tie i ea sige
i  e
pecans gravalepenarmperireloena-weprigermatanr
ss ana encg epIaMSa
1 IntropucTiON
TT re ose tl ning natant ann option te ne
fn ino"lamcd vt ina components steed by trata forthe tack aeeped Sten proves oid
speakers on sharply grammatical/ungrammatical data. ence and describes the data. Section 4 lists the recurrent
Grip recent novel networks are investigated for neural network made invenigned and provides ete
computational reasons. Computationally, recurrent neural the data encoding for the networks. Section 5 presents the
networks are more powerful than feedforward networks results of iwestigation into various training heuristics and
Jeast Turing equivalent [53], [54]. We investigate the presents the main results and simulation details and
ee ne ees a
architectures, in particular Elman, Narendra and Parthasar- of rules in the form of deterministic finite state automata is
Sniy (NGM), and Willan ed Zips OWEZ) Ure qreipated Seton Jad Secon 8 poems session
oe ee
Toor nctcet. Wend ut bt Ean sed Wz eure
her implementing technics for ueprong the conver. 2 MOTIVATION
ee eee ee
Traagruune Sning deren Weanajor he Opeaton Manurt capeees her estealy ben hand xing
of the networks and investigate a rule approximation of symbolic computation and recursive processes. The most
Shot near newect hes amet "pecaly me mba cemutaion and recunine pres, The most
<atracton of resin the form of deterministic Fite sate Faiesatedexceptions such as grams ot huden Matkow
| models. However, finite-state models cannot represent
Previous work (38] has compared neural networks With hierarchical structures as found in natural language’ (48).
other machine learning, paradigms on this problem |this {nthe past few years, several recurrent neural network
Ichnehures ave emerged whch have been wed fae
ee ee ee ee
" [sary apenas fetid omar
‘O The authors ure vwih NEC Research Dustitute. 4 Independence Way, language problems, e.g... papers using the Elman network
Princeton, NP OSH for natural language tasks inctude: [1], {12}, (24). (58), (99)
```

Figura 42: Texto capturado pela imagem corrigida

```
Skew angle: -6.00 degrees
```

Ângulo de inclinação

4 Discussão final

Para as imagens neg_4, neg_28, pos_24 e pos_41, esperamos valores iguais de ângulo de inclinação pois as linhas das imagens são bem definidas, então independente do modelo utilizado, os resultados serão os mesmos. Porém, em imagens que tem mais conteúdo e ”ruído”, os modelos diferentes apresentam ângulos de inclinação diferentes:

Tabela 1: Valores de ângulo de inclinação para cada um dos modelos

Modelo	partitura	sample1	sample2
Projeção	25.7°	14.1°	−6.3°
Hough	26.0°	14.0°	−6.0°

Alguns pontos podem ser destacados:

1. Como as variações entre os ângulos são muito pequenas, não há percepção de diferença a olho nu.

2. As imagens discutidas apresentam altos ruídos, imagens e quantidade de informação com fontes pequenas.

3. Apesar de as linhas da partitura serem bem definidas, elas podem não ser perfeitamente horizontais devido a distorções na digitalização ou pequenas imperfeições na impressão. Isso pode levar a pequenas diferenças na detecção do ângulo entre os métodos. Além disso, a partitura possui elementos adicionais (como notas musicais e símbolos) que podem interferir ligeiramente no cálculo do perfil de projeção ou na detecção de linhas pela transformada de Hough.

4. Para as samples1 e 2, a transformada de Hough pode ter priorizado linhas mais destacadas (como bordas ou margens), enquanto a projeção horizontal pode ter sido influenciada por variações na intensidade dos pixels.

Por serem diferenças muito pequenas entre os valores, podemos afirmar que ambos os métodos são eficientes para a correção do ângulo de rotação de imagens, independente do conteúdo das imagens. Por fim, a captação de texto por meio de Tesseract OCR se mostra possível de fato em apenas imagens com conteúdo textual.