Autor: Robson Tavares Nonato

Data: 09 de março de 2015.

**Apresentação do Problema**

Na área de processamento de imagens existem muitos interessantes problemas de pesquisas. Para ter bons resultados é importante que o desenvolvedor tenha domínio de filtros e transformações típicas desta área.

Além de conhecer a teoria por trás destes filtros é importante também que o desenvolvedor saiba procurar código e bibliotecas nativas da linguagem que já contém implementação destes filtros. Assim o desenvolvedor economiza tempo de implementação pois não precisa mais reinventar a roda, visto que esta já foi inventada e compartilhada por outros usuários.

**Data Limite de Entrega**

A data limite de entrega do código fonte é dia 31 de agosto de 2016 até as 23h59.

**Iniciação**

No Python uma importante biblioteca de processamento e parsing de imagens é o PyTesser. Você pode obter mais informações nos seguintes links:

<https://pypi.python.org/pypi/PyTesser/>

<http://stackoverflow.com/questions/15567141/installing-pytesser>

<https://github.com/RobinDavid/Pytesser>

**Exemplo**

Utilizando a seguinte imagem de entrada:

input image

Vamos construir um trecho de código para implementar um filtro para eliminar ruídos de fundo.

from PIL import Image

img = Image.open('1.gif')

img = img.convert("RGBA")

pixdata = img.load()

# Clean the background noise, if color != black, then set to white.

for y in xrange(img.size[1]):

for x in xrange(img.size[0]):

if pixdata[x, y][0] < 90:

pixdata[x, y] = (0, 0, 0, 255)

for y in xrange(img.size[1]):

for x in xrange(img.size[0]):

if pixdata[x, y][2] < 136:

pixdata[x, y] = (0, 0, 0, 255)

for y in xrange(img.size[1]):

for x in xrange(img.size[0]):

if pixdata[x, y][3] > 0:

pixdata[x, y] = (255, 255, 255, 255)

img.save("input-black.gif", "GIF")

A saída deste código é a seguinte imagem.

http://i.stack.imgur.com/0v0pe.gif

Após remover os ruídos eu posso ainda redimensionar a imagem conforme o seguinte código:

im\_orig = Image.open('input-black.gif')

big = im\_orig.resize((116, 56), Image.NEAREST)

ext = ".tif"

big.save("input-NEAREST" + ext)

O resultado deste código é a seguinte imagem:

enter image description here

Por fim, vamos chamar o PyTesser para fazer a decodificação deste capcha. Veja o código a seguir.

from pytesser import \*

image = Image.open('input-NEAREST.tif')

print image\_to\_string(image)

A saída ideal desse código deveria ser algo como 2ed1c5. Mas devido aos ruídos restantes o resultado vai ser algo como **%/ww**

Sua missão é melhor esse código a fim de alcançar o resultado correto **2ed1c5.**

**Desafio**

Você deve implementar ajustes neste código, filtros e transformações de modo a melhorar o processo de reconhecimento de caracteres para facilitar o trabalho do PyTesser. Para isso utilize seus conhecimentos e suas experiências para anteriores para implementar uma melhoria para este código e assim resolver o problema.

**Considerações**

Em anexo segue um pacote contendo 20 imagens diferentes.

Cada imagem resolvida por completo conta 1 ponto. Se o seu algoritmo resolver por completo 10 imagens, sua pontuação será de (10/20)\*100 = 50 pontos.

Não existe uma solução única.

Você utilizar outra biblioteca de OCR que não o PyTesser se você achar melhor.

Geralmente os algoritmos de iniciantes não resolvem 100% dos casos.

Você pode consultar a web, seus amigos e reutilizar códigos em Git públicos ou Stackoverflow!

A data final de entrega deste projeto é dia 31 de agosto de 2016.

Se você tiver dificuldade, procure ajuda. Entregue aquilo que conseguir entregar.