UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

SCC0251 - Processamento de Imagens

PROJETO FINAL

Identificação de nudez ou pornografia em imagens

Nome	**Número USP**
Felipe Kazuyoshi Takara	8921026
Leonardo Mellin Moreira Ferreira	7982767

1. Índice

2. Introdução

O objetivo do projeto é criar um programa capaz de identificar nudez e pornografia em imagens, usando como base a detecção de pele. A intenção é utilizar o reconhecimento automático no upload de imagens em redes sociais, facilitando o processo de remoção dessas imagens. O projeto foi implementado na linguagem Python, versão 3, e utiliza a biblioteca OpenCV para a manipulação das imagens.

3. Algoritmo

A parte principal do algoritmo é o método utilizado para detectar a pele. Foram utilizadas três etapas principais para esse processo. A primeira foi a calibragem das cores. Então foi feito removido tudo, menos a pele, da imagem. Isso foi feito com base nas cores mais comuns para tons de pele nos sistemas RGB e HSV. Por fim, foi feita a erosão seguida de dilatação para a remoção de pontos indesejados.

a. Calibragem das cores

Para a calibragem das cores, foi utilizada uma normalização da saturação das cores com base no sistema HLS.

```
(h, l, s)=cv2.split(img_hls)
#a imagem é separada nos canais de hue, luminosidade e saturação

max_s=np.amax(s)
min_s=np.amin(s)
max_h=np.amax(h)
s=(s-min_s)/(max_s-min_s)*255 #normalização da saturação
#h=(h-min_s)/(max_h-min_s) #normalização do hue, não utilizada

img_calibrated=cv2.merge((h, l, s)) #recombina os canais
```

Segundo um dos artigos de referência utilizado, também poderia ser utilizada a normalização do canal de cor (hue), mas, para o algoritmo implementado posteriormente (diferente do tratado pelo artigo), os resultados apresentados foram piores.

b. Filtro da pele

Para separar as possíveis áreas de pele de todo o restante da imagem, foi utilizado um filtro com base nas cores de cada pixel, nos sistemas RGB e HSV. Todos os pixels com cores aceitas como tons de pele passam a ser brancos, enquanto os demais são tonalizados de preto. Os valores utilizados para a filtragem foram baseados nos valores utilizados nos artigos de referência.

```
#as matrizes h, s e v os valores de hue, saturação e "valor" no sistema HSV
#as matrizes b, g e r são os valores de azul, verde e vermelho no sistema RGB
#x e y representam a posição do pixel na matriz
if (          (b[x, y]>160 and r[x, y]<180 and g[x, y]<180) or #muito azul
              (g[x, y]>160 and r[x, y]<180 and b[x, y]<180) or #muito verde
              (b[x, y]<100 and r[x, y]<100 and g[x, y]<100) or #muito escuro
              (g[x, y]>200) or #verde
              #(r[x, y]+g[x, y]>400) or #próximo ao amarelo
              #(g[x, y]>150 and b[x, y]<90) or #próximo ao amarelo
              (b[x, y]/(b[x, y]+g[x, y]+r[x, y])>0.4) or #muito azul
              (g[x, y]/(b[x, y]+g[x, y]+r[x, y])>0.4) or #muito verde
              (r[x, y]<102 and 100<g[x, y]<140 and 110<b[x, y]<160)): #oceano
       b[x, y]=r[x, y]=g[x, y]=0 #valores que se encaixam viram preto
#valores para pele no hsv:
elif not(0<=h[x, y]<=20 and 30<=s[x, y]<=255 and 80<=v[x, y]<=255):
       b[x, y]=r[x, y]=g[x, y]=0 #valores que não se encaixam viram preto
else:
       b[x, y]=r[x, y]=g[x, y]=255 #demais valores viram branco
```

Os filtros para a cor amarela não foram utilizados porque seu uso acarretou em resultados piores.

c. Erosão e dilatação

Para que sejam removidos alguns pontos isolados, que podem representar possíveis erros no filtro de cores, foi feita uma erosão seguida de dilatação.

```
kernel=np.ones((3, 3), np.uint8)
img_enhanced=cv2.erode(img, kernel, iterations=1)
img_enhanced=cv2.dilate(img_enhanced, kernel, iterations=1)
```

d. Etapas posteriores

As etapas seguintes à detecção da pele são simples. Primeiro, calcula-se a proporção de pele em relação ao todo.

```
(b, g, r)=cv2.split(img) #a imagem é separada nos canais de cores

skin=sum(sum(b/255))
#a área de pele é dada por todos os pixels onde qualquer canal de cor vale 255
total=len(b)*len(b[0])
#número total de pixels

percentage=skin/total*100 #calcula a porcentagem de branco
```

Por fim, é tomada a decisão quanto à presença de nudez utilizando-se um valor de porcentagem de pele base. Nos testes realizados, os melhores resultados

foram obtidos usando a proporção 23% (um valor menor que esse indica que provavelmente não há nudez).

```
return True if percentage>=23 else False
```

4. **Como executar**

Há três arquivos-fonte diferente: “skin_detection.py”, “analyze_image.py” e “nudity_detection.py”. Todos os arquivos podem ser executados em separado. Os exemplos de execução se baseiam em um sistema Linux e os comandos devem ser rodados a partir do terminal, no diretório desejado.

a. skin_detection.py

Para executar apenas este arquivo, faz-se:

```
$ python3 skin_detection.py file_name
```

Em que “file_name” é o nome da imagem que se deseja analisar (a imagem deve estar no mesmo diretório do arquivo-fonte). Após o código ser executado, o programa exibe a imagem original e a nova imagem, com a pele destacada em branco.

b. analyze_image.py

Para executar apenas este arquivo, faz-se:

```
$ python3 analyze_image.py
```

Este programa analisa as imagens presentes nos subdiretórios “nudity” e “non_nudity” da pasta “image_samples” (cada pasta com quinze imagens, nomeadas de “image1.jpg” a “image15.jpg”). A execução deste arquivo depende do arquivo “skin_detection.py” e de todas as pastas de imagens. Ao se executar o programa, são geradas listas chamadas “percentages.txt”, salvas nas pastas de imagens, que representam a proporção de pele em cada imagem analisada. A execução pode levar alguns minutos, pois faz a análise das 30 imagens de teste.

c. nudity_detection.py

Para executar este arquivo, faz-se:

```
$ python3 nudity_detection.py file_name
```

Em que “file_name” é o nome da imagem que se deseja analisar (a imagem deve estar no mesmo diretório do arquivo-fonte). Esse é o programa mais importante, pois reúne as funções de todos os anteriores (consequentemente, todos devem estar no mesmo diretório para sua execução). Ao ser executado, podem ser exibidas uma das duas mensagens (que indicam a provável ausência ou presença de nudez, respectivamente):

```
$ python3 nudity_detection.py image.jpg
This image probably does not contain nudity.
```

```
$ python3 nudity_detection.py image.jpg
This image probably contains nudity.
```

5. Resultados

a. Exemplo

Para a criação deste exemplo, foi usada a seguinte imagem:

Após a execução de “skin_detection.py”, a imagem retornada foi a seguinte:

A porcentagem de pele atribuída para esta imagem pelo programa “analyze_image.py” foi de 11,64%. Sendo assim, o programa “nudity_detection.py” informa que essa imagem provavelmente não contém nudez.

b. Testes

Para a realização dos testes, foram utilizadas as imagens presentes na pasta “image_samples”. Essas imagens foram obtidas a partir de arquivos pessoais e buscas na internet (por termos como “photo”, “bikini photo” etc). Não foram utilizadas imagens com nudez explícita, deu-se preferência por fotos de pessoas com roupas de baixo ou biquíni (com exceção de uma imagem de nudez, censurada por barras pretas).

Após a execução do programa para todas as imagens presentes, os seguintes resultados foram obtidos:

	Positivo para nudez	Negativo para nudez	Total
Correto	12	12	24 (80%)
Incorreto	3	3	6 (20%)
Total	15	15	30 (100%)

c. Conclusão

Nos testes realizados, o programa apresentou bons resultados (80%), porém o algoritmo possui muitas falhas e não funcionaria bem em um ambiente real.

Umal falha é o método que de detecção da pele, que se baseia exclusivamente na cor de cada ponto. Esse método é falho porque elimina alguns pontos corretos (como áreas muito claras por causa da luz ou escura demais pela sombra), mas, principalmente, destaca muitos pontos que não representam nenhuma pele (especialmente cabelos, roupas claras e areia).

Soma-se a isso o fato de que apenas a proporção de pele em uma imagem não é um bom indicativo para nudez. Fotos com muita pele, especialmente as próximas ao rosto (como selfies) são inofensivas, mas apresentam uma porcentagem grande de pele. Por outro lado, fotos de nudez a uma determinada distância (um ensaio fotográfico na natureza, por exemplo), possuem uma proporção baixa.

Quanto à implementação, houve um problema com a eficiência do algoritmo de detecção de pele. Há um loop que itera por todos os pontos da imagem, para

decidir se determinado ponto representa pele. Esse loop apresentou eficiência muito baixa, demorando alguns segundos para a análise de cada imagem. A causa mais provável é que o Python não é uma linguagem eficiente para se executar laços aninhados e percorrer grandes matrizes. Possíveis soluções englobam o uso de outra linguagem (como C ou Fortran) ou o uso de Cython, uma extensão que permite que códigos com a sintaxe de Python sejam compilados para C. Como o programa foi feito apenas para testar um conceito, nenhuma dessas soluções foi implementada.

6. Referências

Base para a calibragem das cores no HLS e filtro do fundo com base no sistema RGB:
http://www.inf.pucrs.br/~pinho/CG/Trabalhos/DetectaPele/Artigos/Improved%20Automatic%20Skin%20Detection%20in%20Color%20Images.pdf

Base para o filtro do fundo com base no sistema HSV:
http://www.pyimagesearch.com/2014/08/18/skin-detection-step-step-example-using-python-opencv/

Possíveis causas e soluções para a baixa eficiência do algoritmo ao iterar por todos os pontos na imagem:
https://stackoverflow.com/questions/36353262/i-need-a-fast-way-to-loop-through-pixels-of-an-image-stack-in-python