Híbrido sobre reconocimiento de imagen proveniente de un API/REST

May 20, 2019

1 Introducción

La composición de imágenes hibridas puede sintetizarse en un proceso que consiste:

- Hacer pasar una primera foto por un filtro pasa bajos, que da como salida es una imagen borrosa
- La segunda imagen se la hace pasar por un filtro pasa altos, y este último entrega una imagen en la podemos pareciar el contorno de los objetos.
- Superponer las dos imágenes.

El observador de la imagen conseguirá una percepción distinta, dependiendo de la distancia que tome para apreciar la imagen Actualmente se han utilizado para la medicina, específicamente la medicina nuclear, asi lo demuestra la primera jornada nacional de imágenes hibridas en medicina nuclear, realizadas el 21 y 22 de noviembre de 2017.[1]

2 Experiencia

2.1 Módulo 1: Consumir API

Múltiples sitios webs ponen a disposición un número limitado de imágenes, las cuales pueden ser utilizadas con fines académicos. A su vez, estos sitios proporcionan su propia API/REST, con el fin de facilitar el trabajo. Con esto solo se debemos consumir los servicios que nos proporciona el proveedor de una manera responsable.

Actividades que realiza este módulo:

1. Realizará una solicitud de datos a un servidor
2. Obtendrá la respuesta
3. Guardará la imagen

```
def consumirAPI(dirFile):
    # Consumir API
    url='https://randomuser.me/api/'
    respuesta=requests.get(url)
    data=json.loads(respuesta.content)
    results=data["results"]
    src=results[0]["picture"]["large"]
    print(src)

    # Traer imagen del API
    respuesta = requests.get(src)
    if respuesta.status_code == 200:
        with open(dirFile, 'wb') as f:
            f.write(respuesta.content)
```

2.2 Módulo 2: Reconocimiento de Rostros

Aprovechando investigaciones previas, y soluciones existentes, la librería de openCV pre-instalados clasificadores del tipo Haar Cascade: es un método efectivo de detección de objetos propuesto por Paul Viola y Michael Jones en su artículo, "Detección rápida de objetos usando una cascada aumentada de características simples" en 2001. Es un enfoque basado en el machine learning donde el algoritmo es entrenado a partir de muchas imágenes positivas y negativas. Luego se utiliza para detectar objetos en otras imágenes.[2]

LBP Cascade: el concepto consiste en etiquetar los píxeles de una imagen mediante el umbral de la vecindad de 3 por 3 de cada píxel con el valor del píxel central y considerando el resultado como un número binario. Se confecciona un histograma con 256 intervalos, a cada intervalo le corresponde una etiqueta, que es la que le asignamos a los pixeles y esto puede ser usando como un descriptor de texturas.[3]

El clasificador del tipo Haar es mucho más preciso reconociendo rostros y tiene una taza muy baja de falsos de positivos, como aspectos negativos cabe destacar que es un algoritmo computacionalmente complejo y lento, además de que requiere un entrenamiento previo y tiene limitaciones respecto a la iluminación. El clasificado del tipo LBP es computacionalmente mucho más eficiente, se entrena en mucho menos tiempo y maneja mucho mejor la iluminación, sin embargo es menos preciso y tiene un taza de falsos positivos considerable. La elección del clasificador dependerá de aquellos aspectos que se estén dispuestos a renunciar.

Pasos que realiza el módulo:

1. Obtiene los datos para reconocimiento de un xml
2. Aplica el clasificador
3. Devuelve los puntos de origen del rostro (x,y) y tamaño del rostro (w,h)

```
def reconocer(imagen):
    # Reconocer rostro
    face_cascade = cv2.CascadeClassifier('xml/
        haarcascade_frontalface_default.xml')
    face = face_cascade.detectMultiScale(imagen, 1.3, 5)
    x=y=w=h=0
    if len(face):
        x= face[0][0]
        y= face[0][1]
        w= face[0][2]
        h= face[0][3]
    return x,y,w,h
```

2.3 Módulo 3: Generación de Imagen Hibrida

Este módulo :

1. Requiere imagen, máscara, puntos en donde se encuentra el rostro
2. Realizará los filtros pasa-altos (laplaciano), pasa-bajos (gaussiano)
3. Aplicará los filtros a las imágenes
4. Cambiará el tamaño de la máscara para adecuarlo al tamaño del rostro
5. Va a sobreponer la máscara sobre la imagen de fondo

```
def mezclar(imagen, mask, x, y, w, h):
    # filtros
    blur = cv2.blur(imagen,(17,17))
    laplacian = cv2.Laplacian(mask,cv2.CV_8U,ksize=17)
    # mezclas
    mezcla5= cv2.addWeighted(imagen,0.6,blur,0.4,0)
    mezcla6= cv2.addWeighted(mask,0.8,laplacian,0.2,0)

    parteMezcla= cv2.resize(mezcla6,(w,h))
    posx=x
    posy=y
    rows,cols=parteMezcla.shape

    mezcla5[posx:rows+posx:1, posy:cols+posy:1]=cv2.
        addWeighted(mezcla5[posx:rows+posx:1, posy:cols+posy
        :1],0.7,parteMezcla,0.3,0)

    return mezcla5
```

2.4 Módulo 4: Implementación

Para aplicar los módulos anteriores, se requerirá el siguiente código, que permitirá:

1. Obtener imagen (puede ser local o haciendo uso del módulo 1)
2. Crear imagen en gris (tanto para la imagen de fondo como para máscara)
3. Llamar al módulo 2 y 3
4. Probar si la imagen se ha logrado reconocer el rostro

```
# Obtener imagen
file="result/api-1.jpg"
consumirAPI(file)

# Crear gris del imagen base
imagen = cv2.imread(file)
rgbImagen= cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)

# Crear gris de la máscara
mask = cv2.imread('img/mask.jpg')
rgbMask= cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)

# Realizar el hibrido
x, y, w, h = reconocer(imagen)
if x!=0:
    hibrido= mezclar(rgbImagen, rgbMask, x, y, w, h)
    plt.imshow(hibrido, cmap='gray')
    cv2.imwrite("result/sv-4.png", hibrido)
else:
    print('buscar otra imagen')
```

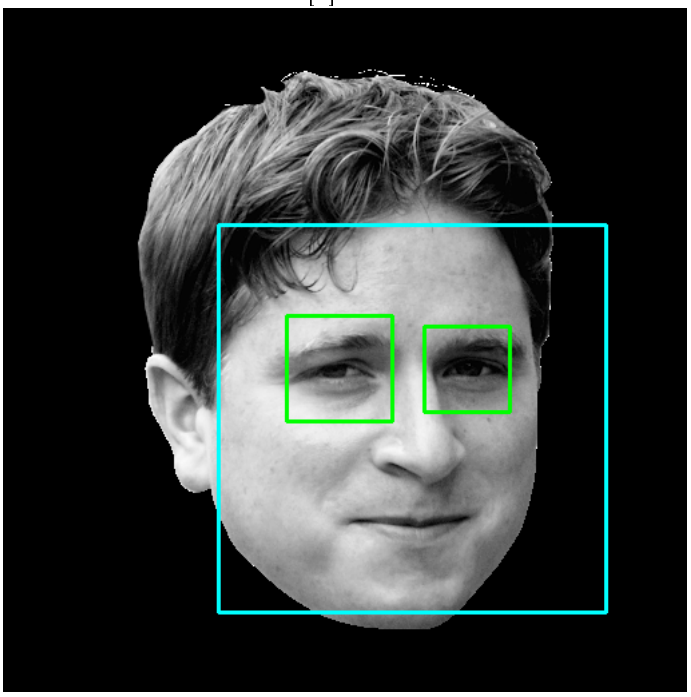
3 Resultados

Primeras pruebas [4]



Máscara

Reconocimiento de rostro [5]



Creando el hibrido de una imagen local [6]

Creando el hibrido de una imagen provista por un API [6]

4 Conclusiones

Con las capacidades tecnológicas actuales, y la facilidad de acceso a las mismas, permiten que tanto aficionados como investigadores colaboren en el desarrollo de soluciones de reconocimiento de rostros en campos tan importantes como son la medicina en el capo del diagnóstico por imagen o la forensia digital o como en este caso solo para entretenimiento.

La implementación realizada permite tener un acercamiento similar a las experiencias provistas por las grandes aplicaciones móviles como Snapchat ó Instagram.

5 Bibliografía

- [1] http://www.fuesmen.edu.ar/1-jornada-nacional-de-imagenes-hibridas-en-medicina-nuclear
- [2] https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html
- [3] http://cs229.stanford.edu/proj2008/Jo-FaceDetectionUsingLBPfeatures.pdf
- [4] https://github.com/jrodrigopuca/procesamiento-img/blob/master/hibrido.ipynb
- [5] https://github.com/jrodrigopuca/procesamiento-img/blob/master/reconocer_rostro.ipynb
- [6] https://github.com/jrodrigopuca/procesamiento-img/blob/master/dia_06.ipynb