

Introducción a la Visión Computacional

Tarea 1

Nombre: Patricio Soto

Vision Computacional - Magister Data Science UDD

Se elige un set de imágenes de Fuego y Humo, estas imágenes provienen desde distintos dispositivos como cámaras y celulares de distintas marcas, tomadas por distintas personas. El objetivo es ver si es posible identificar el fuego para poder desarrollar algún algoritmo que sirva en la identificación de inicio o riesgos de incendios con cámaras de seguridad.

Link: <https://www.kaggle.com/datasets/dataloclusterabs/fire-and-smoke-dataset>

```
In [1]: import glob
from pathlib import Path
from PIL import Image, ImageFilter

In [153]:_RESIZE_PERCENT = 0.3
THRESHOLD = 200
INPUT_FOLDER = 'input_img'
OUTPUT_FOLDER = 'output_img'
TYPES_IMG = (*.jpg, *.png)

In [152]: img_metadata_files = []

def read_all_imgs():
    print('\n Leyendo archivos de Imagen y convirtiendo a objeto.')
    img_list = []
    for file in INPUT_FOLDER:
        img_list.extend(glob.glob(_INPUT_FOLDER+'\\*'+ext))
    for file in img_list:
        img_path = Path(file)
        img = Image.open(file)
        img_metadata = {
            'name': img_path.name,
            'ext': img_path.suffix,
            'path': file,
            'img': img,
            'img_processed': None,
            'size_file': img_path.stat().st_size,
            'size_img': img.size
        }
        img_metadata_files.append(type('img_metadata', (object,), img_metadata)())
    print('Guardando imagenes procesadas en [0]'.format(fullpath_output))
    for img in img_metadata_files:
        img_metadata.img_processed.save(fullpath_output+'\\{}\\'.format(img_metadata.name))

def resize_all_imgs():
    print('Procesando Imagenes ... Cambiando Tamaño ...')
    for img in img_metadata_files:
        x = img.metadata.size_img[0] * _RESIZE_PERCENT
        y = img.metadata.size_img[1] * _RESIZE_PERCENT
        img.metadata.img_processed = img.metadata.img.resize((int(round(x, 0)), int(round(y, 0))))
    print('...', end='')

def identify_fire_all_imgs():
    print('Procesando Imagenes ... Filtrado e Identificación del Fuego ...')
    for img in img_metadata_files:
        img.metadata = img.metadata.filter(ImageFilter.GaussianBlur(5))
        img.metadata = img.metadata.filter(lambda pxi: 255 if pxi > THRESHOLD else 0)
        img.metadata.img_processed = img_processed.convert('1')
    print('...', end='')

In [154]: read_all_imgs()
resize_all_imgs()
identify_fire_all_imgs()
save_processed_imgs('resized')
identify_fire_all_imgs()
save_processed_imgs('filtered')

Leyendo archivos de Imagen y convirtiendo a objeto.
.
.
.
Procesando Imagenes ... Cambiando Tamaño ...
.
.
.
Guardando imagenes procesadas en output_img\resized.
.
.
.
Procesando Imagenes ... Filtrado e Identificación del Fuego ...
.
.
.
Guardando imagenes procesadas en output_img\filtered.
.
.
.

En la primera parte se prefirió disminuir el tamaño en pixeles de forma porcentual, al darse cuenta que los tamaños no eran simétricos en las imágenes lo que daría como resultado la deformación de las mismas.

In [217]: for i in [10, 30, 45, 90]:
    print('Imagen [0] tamaño:{}'.format(i), img_metadata_files[i].img.size)

Imagen 10 tamaño: (1920, 4160)
Imagen 30 tamaño: (2576, 1932)
Imagen 45 tamaño: (4000, 1920)
Imagen 90 tamaño: (352, 640)

Ejemplo, en imagen 10 tiene un tamaño de 1920x4160 y la imagen 30 tiene un tamaño de 2576x1932. Estas se redujeron a un 30% de su tamaño. Quedando en 576x1248 y 773x580 respectivamente.

In [220]: for i in [10, 30, 45, 90]:
    print('Imagen [0] tamaño cambiado: {}'.format(i),
        img.open('..\output_img\\resized\\{}\\img_metadata_files[{}].name'.format(i)))

Imagen 10 tamaño cambiado: (576, 1248)
Imagen 30 tamaño cambiado: (773, 580)
Imagen 45 tamaño cambiado: (106, 553)
Imagen 90 tamaño cambiado: (106, 192)

En la segunda parte se intenta realizar operaciones y filtrado para ver la posibilidad de identificar solo las llamas de las imágenes. Tomaremos el ejemplo de la imagen 45 como éxito.

In [206]: img_metadata_files[45].img

Out[206]: 

In [215]: Image.open('..\output_img\\filtered\\{}\\img_metadata_files[45].name'.format(45))

Out[215]: 

Para llegar a este resultado lo primero que realizamos es convertir la imagen a blanco y negro, para luego realizar un filtro de desenfoque gaussiano para suavizar la imagen y luego se recorrieron los pixeles y se mantuvieron solo los pixeles con una luminosidad mayor a 200. Al finalizar se vuelve a convertir a 1-bit por pixel para eliminar lo suavizado.

Mostraremos otros ejemplo de éxito antes de seguir conversado de los inconvenientes.

In [207]: img_metadata_files[57].img

Out[207]: 

In [214]: img_metadata_files[93].img

Out[214]: 

Ahora de igual manera existen resultados no deseados, y sobre todo es en las imágenes de dia, o en las que existan partes con mucho brillo. Esto debido a la técnica utilizada, ya que se confunden estas zonas, con la luminosidad que emite el fuego. Esto se solucionó mediante el manejo de la imagen en si, o pasando por un entrenamiento de clasificación utilizando técnicas de "deep learning". Lo que podría ser un factor a considerar no uniformidad del fuego, ya que esta varía según los factores externos a él.

Nota: Repositorio github con imágenes input y output del proceso.
Link: https://github.com/pssotosa/MDS/tree/main/VISION\_COMPUTACIONAL/TAREA\_1
```

