Super_Resolution_CI_Preprocessed

February 9, 2021

1 Super-resolución 4x:

1.1 1.- Preprocesado de imágenes:

Ejercicio de curso para la asignatura de Computación Inteligente perteneciente al Máster Universitario en Sistemas Inteligentes y Aplicaciones Numéricas para la Ingeniería (MUSIANI) en el curso 2020/21, realizado por Juan Sebastián Ramírez Artiles.

En este notebook se preprocesarán las imágennes del dataset DIV2X de acceso libre para que puedan ser usadas por el modelo de super-resolución implementado en el notebook `Super_Resolution_CI_Model.ipynb`.

[1]:

```
import cv2
from os import listdir
from os.path import isfile, join
import numpy as np
from pathlib import Path
import os
```

Se usarán las imágenes de alta resolución del track1. Son originariamente 800 imágenes que se seleccionaran y dividirán en imágenes para entrenamiento e imágenes para validación.

[2]:

```
small = "x4"
big = "x1"

path_small = "img_" + small
path_big = "img_" + big

cut_small = "cut_small"
cut_big = "cut_big"

not_used_small = "not_used_" + small
not_used_big = "not_used_" + big

Path(cut_small).mkdir(parents=True, exist_ok=True)
Path(cut_big).mkdir(parents=True, exist_ok=True)

Path(not_used_small).mkdir(parents=True, exist_ok=True)
Path(not_used_big).mkdir(parents=True, exist_ok=True)
```

Se seleccionarán las imágenes que concuerden con unas dimensiones superiores a 1300 píxeles de alto y 1900 píxeles de ancho.

[3]:
```
def filter_images():

    for f_small in listdir(path_small):
        f_big = f_small.replace(small, "")

        img_small = cv2.imread(join(path_small, f_small))
        img_big = cv2.imread(join(path_big, f_big))

        if img_big.shape[0] < 1300 or img_big.shape[1] < 1900:
            cv2.imwrite(join(not_used_small, f_small), img_small)
            cv2.imwrite(join(not_used_big, f_big), img_big)

            os.remove(join(path_small, f_small))
            os.remove(join(path_big, f_big))
```

Las imágenes con estilo retrato se voltearán para que queden en estilo panorámico.

[4]:
```
def rotate_images():

     for f_small in listdir(path_small):
        f_big = f_small.replace(small, "")

        img_small = cv2.imread(join(path_small, f_small))
        img_big = cv2.imread(join(path_big, f_big))

        if img_small.shape[0] > img_small.shape[1]:
            aux_img_small = cv2.rotate(img_small, cv2.ROTATE_90_CLOCKWISE)
            cv2.imwrite(join(not_used_small, f_small), img_small)
            cv2.imwrite(join(path_small, f_small), aux_img_small)

            aux_img_big = cv2.rotate(img_big, cv2.ROTATE_90_CLOCKWISE)
            cv2.imwrite(join(not_used_big, f_big), img_big)
            cv2.imwrite(join(path_big, f_big), aux_img_big)
```

Este método será el que establezca las dimensiones finales que tendrán las imágenes.

[5]:
```
def get_min_rows_cols(path):

    min_rows = np.Inf
    min_cols = np.Inf

    for f in listdir(path):
        img = cv2.imread(join(path, f))

        if img.shape[0] < min_rows:
             min_rows = img.shape[0]

        if img.shape[1] < min_cols:
            min_cols = img.shape[1]

    return min_rows, min_cols
```

Este otro método comprobará que las imágenes reducidas y las de alta resolución mantienen su proporción.

[6]:

```
def compare_images():

    for f_small in listdir(path_small):
        f_big = f_small.replace(small, "")

        img_small = cv2.imread(join(path_small, f_small))
        img_big = cv2.imread(join(path_big, f_big))

        if img_small.shape[0] * 4 != img_big.shape[0]:
            print(f_small)
            return False

        if img_small.shape[1] * 4 != img_big.shape[1]:
            print(f_small)
            return False

    return True
```

En este método es en donde se cortarán finalmente las imágenes.

[7]:

```
def cut_images(min_row_size, min_col_size):

    for f_small in listdir(path_small):
        f_big = f_small.replace(small, "")

        img_small = cv2.imread(join(path_small, f_small))
        img_big = cv2.imread(join(path_big, f_big))

        center_y = img_small.shape[0] // 2
        center_x = img_small.shape[1] // 2

        h = min_row_size // 2
        w = min_col_size // 2

        crop_img_small = img_small[center_y-h:center_y+h, center_x-w:center_x+w]

        center_y *= 4
        center_x *= 4
        h *= 4
        w *= 4

        crop_img_big = img_big[center_y-h:center_y+h, center_x-w:center_x+w]

        cv2.imwrite(join(cut_small, f_small), crop_img_small)
        cv2.imwrite(join(cut_big, f_big), crop_img_big)
```

[8]:

```
if compare_images():
    rotate_images()
    filter_images()
    min_rows, min_cols = get_min_rows_cols(path_small)
    cut_images(min_rows, min_cols)
```