Project 1

# Technical documentation

## **Introduction**

This application is an augmentation tool for images which provides six augmentation algorithms. The algorithms can be grouped in two categories: pixel-level (hue, brightness, saturation) and geometrical (rotate, translate, flip).

The application will generate a folder containing the processed images based on an input folder containing all the images to be processed and a configuration file describing the order (+combinations) of the augmentations.

## **Prerequisites**

* Python and opencv installed on local computer
* A Test directory containing 5 images with various content (e.g., saved from web) of the size 640 x 480.

## **Specification**

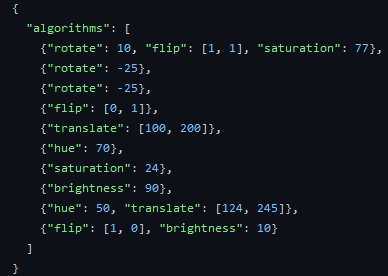
* Allows user to select a directory on local disk. (Using tkinter library)
* Read all .jpg images from this directory and, for each of them, apply a set of predefined augmentation algorithms with a set of predefined parameters.
* The augmentation algorithms and corresponding parameters to be applied will be loaded when the program starts from a configuration file
* The results of augmentation process will be saved on a new directory (output dir), having the same name with the original one plus the "\_aug" suffix.
* Each augmented image will be saved in the output dir having the name of augmentation algorithm as suffix followed by an incremental number starting with "\_1".

## **Configuration file**

The configuration file is a **config.json** file located where the application is running which will describe the augmentations to be applied for the current session. In the configuration you will be able to specify single or chained augmentations. The list of the augmentation algorithms that the application supports is the following:

* Pixel-level
  + Hue (e.g., “hue”: 50)
  + Brightness (e.g., “brightness”: 90)
  + Saturation (e.g., “saturation”: 24)
* Geometrical
  + Rotate (e.g., “rotate”: 10)
  + Translate (e.g., “translate”: [100, 200])
  + Flip (e.g., [1, 0])

**Input/output example:**

*Config file content*(config.json):  


chained

***Test*** *dir*:  
I1.jpg, I2.jpg, I3.jpg, I4.jpg, I5.jpg

***Test\_aug*** *dir*:  
I1\_rotate\_flip\_saturation\_1.jpg, I1\_rotate\_2.jpg, I1\_rotate\_3.jpg, I1\_flip\_4.jpg, I1\_translate\_5.jpg, I1\_hue\_6.jpg, I1\_saturation\_7.jpg, I1\_brightness\_8.jpg, I1\_hue\_translate\_9.jpg, I1\_flip\_brightness\_10.jpg, I2\_rotate\_flip\_saturation\_11.jpg, I2\_rotate\_12.jpg, I2\_rotate\_13.jpg, I2\_flip\_14.jpg, I2\_translate\_15.jpg, I2\_hue\_16.jpg, I2\_saturation\_17.jpg, I2\_brightness\_18.jpg, I2\_hue\_translate\_19.jpg, I2\_flip\_brightness\_20.jpg, I3\_rotate\_flip\_saturation\_21.jpg, I3\_rotate\_22.jpg, I3\_rotate\_23.jpg, I3\_flip\_24.jpg, I3\_translate\_25.jpg, I3\_hue\_26.jpg, I3\_saturation\_27.jpg, I3\_brightness\_28.jpg, I3\_hue\_translate\_29.jpg, I3\_flip\_brightness\_30.jpg, I4\_rotate\_flip\_saturation\_31.jpg, I4\_rotate\_32.jpg, I4\_rotate\_33.jpg, I4\_flip\_34.jpg, I4\_translate\_35.jpg, I4\_hue\_36.jpg, I4\_saturation\_37.jpg, I4\_brightness\_38.jpg, I4\_hue\_translate\_39.jpg, I4\_flip\_brightness\_40.jpg, I5\_rotate\_flip\_saturation\_41.jpg, I5\_rotate\_42.jpg, I5\_rotate\_43.jpg, I5\_flip\_44.jpg, I5\_translate\_45.jpg, I5\_hue\_46.jpg, I5\_saturation\_47.jpg, I5\_brightness\_48.jpg, I5\_hue\_translate\_49.jpg, I5\_flip\_brightness\_50.jpg

## **Augmentation algorithms**

## *Hue*

• Adjusts the hue of the image

• I make use of the HSV representation and modify the hue (H) matrix

• I switched off the remainder mechanism by avoiding to go past 255

• The function takes only one parameter: the value by which to adjust the hue

1. *Brightness*

• Adjusts the brightness of the image

• I make use of the HSV representation and modify the value (V) matrix

• I switched off the remainder mechanism by avoiding to go past 255

• The function takes only one parameter: the value by which to adjust the brightness

1. *Saturation*

• Adjusts the saturation of the image

• I make use of the HSV representation and modify the saturation (S) matrix

• I switched off the remainder mechanism by avoiding to go past 255

• The function takes only one parameter: the value by which to adjust the saturation

1. *Rotate*

• The function rotates an image and it takes one parameter: the degree by which the image is

rotated

• For this augmentation the position of each pixel in the new image is calculated with respect to

image center, using the formula presented in the course slides

1. *Translate*

• The function translates the image on x and/or y, and it takes 2 parameters: the value by which the images is translated on x axis and the value by which the image is translated on y axis

• Numpy indices selection property is used for solving this augmentation

1. *Flip*

• The function flips the image vertically and/or horizontally and it takes 2 parameters: the first for

horizontal flipping (can be 1 or 0) and the second for vertical flipping (can be 1 or 0)

• Numpy indices selection property is used for solving this augmentation