**Fundamentals of Computer Vision – Project 1**

**Augmentation system for CV ML algorithms**

1. **Requirements**

Using Python and OpenCV write a program that:

* + Allows user to select a directory on local disk. (e.g. 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. (e.g. Rotation with +15 degree).
    - The augmentation algorithms and corresponding parameters to be applied will be loaded when the program starts from a configuration file (plain text, xml etc.)
    - 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".

1. **Configuration file structure**

* The configuration file is a JSON file that has the schema:

{

"type": "array",

"items": {

"type": "array",

"items": {

"type": "object",

"properties": {

"type": { "type": "string" },

"number1": { "type": "string" },

"number2": { "type": "string" },

"color": { "type": "string" }

}

}

}

}

* The file contains an array of arrays, each of the arrays containing objects that represent the augmentations
* Each augmentation has a type – the name, and 3 other variables, each representing different parameters for the specific augmentation (ex. for Rotate, the number1 parameter represents the degree of rotation)

1. **Framework design**

* **AugmentationTechnique** – class that stores an augmentation technique. The **augmentation\_type** parameter stores the name of the augmentation, while the other parameters store the different values required for each augmentation.
* **AugmentationSystem** – class that performs augmentations. It receives the augmentation array and the file path of the image folder. The class performs the following:
  + 1. Loads the images from the corresponding folder
  + 2. Creates the **\_aug** folder if it doesn’t exist, or clears it if it exists
  + 3. Performs the augmentations. It loops through the augmentation list and performs the corresponding augmentations for each image in the folder, and then stores the resulted images in the **\_aug** folder.

The application runs as follows:

1. The JSON data is read using the **JSON library** in python
2. The augmentation data is stored in an array: each element in the array is a list that contains **AugmentationTechnique** objects. These lists correspond to each chain augmentation that needs to be performed.
3. A GUI window is opened and the user is prompted to choose the folder where the images are stored. The **tkinter library** is used for selecting the folder from where we take the images.
4. After that, the user can click a button to perform the augmentations. The **AugmentationSystem** class is created.
5. The images are loaded using the **os** library and stored as **cv2** images. The **openCV** library is used for processing images and applying different augmentation techniques available.
6. The system loops through the augmentations and the images and performs all the augmentations needed. The **openCV** library is used for storing the augmented images in the correct folder.
7. **Augmentation algorithms**

The app allows the user to perform 6 different augmentation algorithms:

* Blur
  + - Description: Applies a Gaussian blur to the input image
    - Uses **numpy** tocreate a Gaussian kernel and the **cv2. filter2D()** function to apply the kernel to the image
* Sharpen
  + Description: Applies a sharpening filter to the input image, enhancing edge details.
  + Uses **numpy** tocreate a kernel with positive and negative weights that sum to 1, and the **cv2. filter2D()** function to apply the kernel to the image
* Tint
* Description: Applies a color tint to the input image, adjusting the intensity of a specific color channel.
* Taking as input a color choice (Red, Blue, Green) and a specified percentage, this function enhances the selected color channel for each pixel by the provided percentage.
* Flip
  + Description: Flips the input image horizontally, vertically, or both, altering its orientation
  + The method uses the **cv2.flip()** function to flip the image. This function takes as input 0 for vertical flip, 1 for horizontal flip, and -1 for both.
* Rotate
* Description: Rotates the input image by a specified angle.
* **cv2.getRotationMatrix2D()** is used to create a rotation matrix with the center in the center of the image, at the angle provided as input. Next, this matrix is passed to the function **cv2.warpAffine()**, that rotates the image.
* Translate
* Description: Translates (shifts) the input image by specified values in the x and y directions.
* An empty matrix with the dimensions of the input image is created. Each pixel is translated into the pixel with the offset of x and y, that are provided as inputs.