Augmentation System Project

The framework of this project consists of reading, augmenting, and saving images based on configurations specified in an XML file.

1. **Imports and Setup**

* OpenCV (**cv2**): used for image manipulation and display.
* OS (**os**): used for file and directory operations.
* NumPy (**numpy**): used for efficient array manipulation.
* Tkinter (**tk**): used for opening a file dialog for user interaction.
* XML (**ET**): used for parsing the configuration XML file containing augmentation algorithms and corresponding parameters.

1. **Image Loading**

Function **load\_images\_from\_directory()** uses Tkinter to open dialog for selecting an image directory, then reads each image file in the directory into a list (**image\_list**). It also returns the directory path which is used for creating the output directory and list of file names which are used for saving augmented images along with the augmentation algorithm performed.

1. **Configuration Loading**

Function **load\_augmentation\_settings()** parses an XML configuration file (**augmentation\_config.xml**) and extracts augmentation operations and their parameters, storing them in an array. Each augmentation is a tuple of the operation name and its parameters.

1. **Image Saving**

Function **save\_augmented\_image()** saves the augmented images to an output directory with a unique filename based on the original name, augmentation type and index. This function helps organize augmented outputs.

1. **Augmentation Functionality**

Function **apply\_augmentation()** represents the core function that applies augmentations. Based on the aug\_name which represents the augmentation algorithm, it dispatches the operation to a specific augmentation function, passing the image and required parameters.Each augmentation function modifies the image using either low-level logic or OpenCV functions:

* **Brightness adjustment**: Adds a constant value to each pixel, making the image lighter or darker.
* **Contrast adjustment**: Adjusts the difference between the light and dark areas by scaling pixel values around the midpoint (128).
* **Color distortion**: Scales the intensity of each color channel (blue, green, red) to create a color shift.
* **Gamma correction**: Applies a non-linear transformation to adjust brightness, useful for correcting lighting variations.
* **Histogram equalization**: Redistributes pixel intensities to improve contrast, especially in low-contrast images.
* **Flip Image:** Flips the image horizontally or vertically, creating a mirror effect.
* **Scale Image:** Resizes the image by a given scale factor, either enlarging or shrinking it.
* **Rotation:** Rotates the image around its center by a specified angle.
* **Gaussian Noise Addition:** Adds random noise from a Gaussian distribution, simulating graininess or random image interference.
* **Logarithmic Point Transformation:** Applies a logarithmic function to intensities, enhancing darker areas.
* **Box Filter:** Averages pixel values in a neighborhood, creating a simple blurring effect.
* **Gaussian Filter:** Blurs the image using a Gaussian kernel, which smoothens and reduces noise.
* **Bilateral Filter:** Smoothens regions while preserving edges by filtering based on both spatial and intensity distance.
* **Median Filter:** Replaces each pixel with the median of its neighbors, effective for reducing salt-and-pepper noise.
* **High-Pass Filter:** Enhances edges by subtracting a blurred (low-pass) version from the original.
* **Grayscale Conversion:** Converts the image to grayscale, removing color information and retaining only intensity.
* **Binary Conversion:** Converts to binary (black and white) based on a threshold, highlighting specific intensities.
* **Translate Image:** Shifts the image horizontally and/or vertically by specified pixel amounts.
* **Shear Image:** Skews the image along the x or y-axis, distorting its shape.

1. **Execution Flow:**

The main flow of execution is started by the **main()** function which orchestrates the following process:

* Loads images from a selected directory
* Reads augmentation settings from the XML configuration file
* Creates an output directory to store augmented images
* Iterates through each image and applies each augmentation. The augmented image is saved with a unique filename.