**CHC6781 Machine Vision**

Dr Happy N. Monday

**Practical Experiment:** Image Manipulation Using Keras, Python, and OpenCV

**Objective:** The objective of this practical experiment is to familiarize students with fundamental image processing tasks using the Keras, Python, and OpenCV libraries. Through this experiment, students will learn how to load, display, resize, rename, convert color channels, perform edge detection, and apply image augmentation techniques.

**Learning Outcome:**

After completing this practical experiment, students should be able to:

* Load and display images using OpenCV and Matplotlib.
* Perform basic image manipulations, including resizing, color channel conversion, and edge detection.
* Understand the concept of data augmentation and apply it using Keras' `ImageDataGenerator`.
* Develop a comprehensive image processing pipeline using various libraries and techniques.

**Materials Needed:**

* Python environment with Keras, OpenCV, and required packages installed
* A set of sample images for experimentation

**Experiment Steps:**

**Step 1: Image Loading and Display**

1. Import necessary libraries: Import the required libraries: `opencv-python`, `matplotlib.pyplot`, and `os`.

2. Load an image: Use OpenCV's `cv2.imread()` function to load an image from the provided dataset.

3. Display the image: Use `plt.imshow()` from Matplotlib to display the loaded image.

4. Show the image: Use `plt.show()` to visualize the image.

**Step 2: Image Resize and Rename**

1. Resize the image: Use OpenCV's `cv2.resize()` to resize the loaded image to a specific size (e.g., 224x224 pixels).

2. Rename the image: Use `os.rename()` to rename the resized image file.

**Step 3: Image Color Channel Conversion**

1. Convert color channels: Use OpenCV's `cv2.cvtColor()` to convert the image from one color space to another (e.g., RGB to grayscale).

2. Display the converted image: Use Matplotlib `plt.imshow()` to display the converted image.

**Step 4: Image Edge Detection**

1. Apply edge detection: Utilize OpenCV's `cv2.Canny()` function to perform edge detection on the grayscale image.

2. Display the edges: Show the edges obtained from the edge detection process.

**Step 5: Image Augmentation**

1. Data augmentation: Use Keras' `ImageDataGenerator` to apply various data augmentation techniques such as rotation, flip, zoom, etc., to the original image.

2. Display augmented images: Display a few augmented images using Matplotlib.

**In-Class Observation Question:**

Why is data augmentation important in image processing tasks, and how can it improve the performance of machine learning models?

**Take Home Assignment:**

Write a Python script that combines all the above steps to create a pipeline for processing a batch of images. The script should loop through a folder containing multiple images, perform the defined operations on each image, and save the processed images in an output folder.

**Real-World Relevance:**

Image processing is a crucial component in various real-world applications such as computer vision, medical imaging, autonomous vehicles, and more. Understanding these fundamental image manipulation tasks lays the foundation for building advanced image processing and computer vision systems. The ability to pre-process and augment images effectively contributes to the enhancement of model performance and robustness in real-world scenarios.