Software Requirement Specification

June 19, 2024

Project ID & Name	**Image Processing**		
Project Manager	Md. Mohoiminul Islam Chowdure		
Customer Request Reference			
Prepared by	Abu Darda	Date	23 May, 2024
Reviewed by		Date	

Contents

1 Requirement Scope Summary

This project involves creating a Python script utilizing OpenCV to perform various basic computer vision operations on images. The script will load an image, apply a series of processing steps, and save the resulting images to disk.

2 Functional Requirements Identification

FR1 Load and display images using OpenCV.

FR2 Convert the loaded image to grayscale.

FR3 Resize the image to a suitable size for computer vision models.

FR4 Apply various image processing techniques such as:

  a) Blurring
  b) Adding noise
  c) Denoising
  d) Augmentation

FR5 Detect edges in the image.

FR6 Perform histogram equalization.

FR7 Apply global and adaptive thresholding on the grayscale image.

FR8 Save each processed image to disk in a specified directory.

3 Functional Requirements Details

3.1 FR1: Load and Display Images

- **Description:** The system shall load an image from a specified file path and display it using OpenCV.
- **Input:** File path of the image.
- **Output:** Displayed image.

3.2 FR2: Convert the Loaded Image to Grayscale

- **Description:** The system shall convert the loaded image to grayscale.
- **Input:** Loaded image.
- **Output:** Grayscale image.

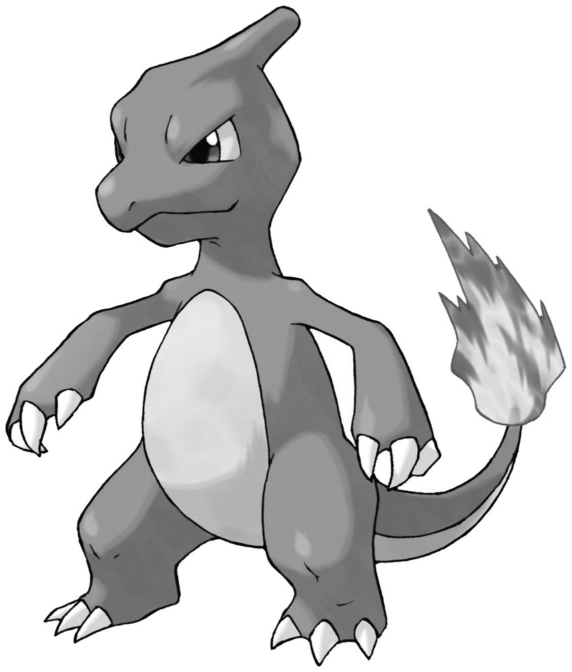

Figure 1: Grayscaled Image

3.3 FR3: Resize the Image

- **Description:** The system shall resize the image to a suitable size for computer vision models.
- **Input:** Grayscale image.
- **Output:** Resized image.

3.4 FR4: Apply Image Processing Techniques

- **Description:** The system shall apply various image processing techniques.
- **Input:** Resized image.
- **Output:** Processed images.
- **Techniques:**
  a) **Blurring:** Apply Gaussian or median blur.
  b) **Adding Noise:** Add Gaussian or salt-and-pepper noise.
  c) **Denoising:** Apply denoising filters.
  d) **Augmentation:** Apply random transformations.

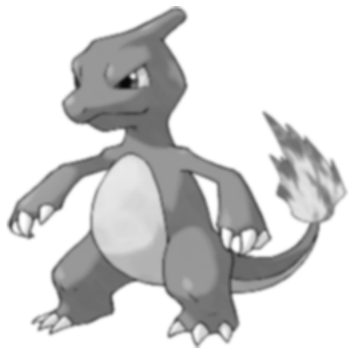

Figure 2: Blurred Image

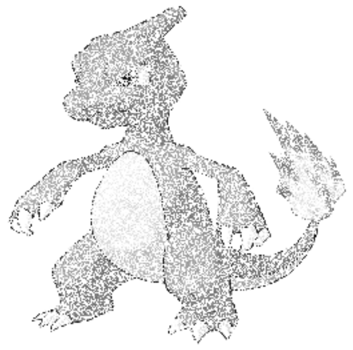

Figure 3: Noised Image

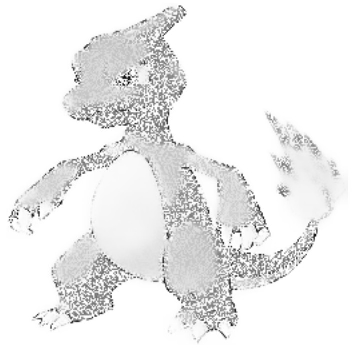

Figure 4: Denoised-noised Image

3.5 FR5: Detect Edges in the Image

- **Description:** The system shall detect edges in the image using edge detection techniques like Canny.
- **Input:** Processed image.
- **Output:** Image with detected edges.

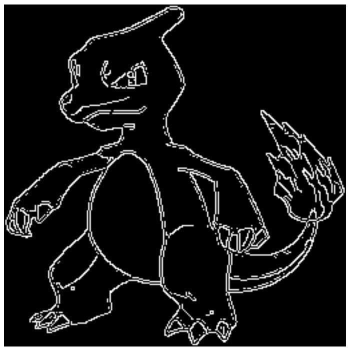

Figure 5: Edge detected Image

3.6 FR6: Perform Histogram Equalization

- **Description:** The system shall perform histogram equalization on the image.
- **Input:** Grayscale image.
- **Output:** Image with equalized histogram.

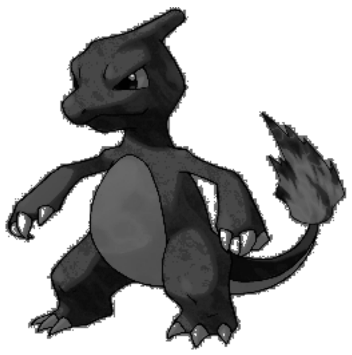

Figure 6: Histogram equalized Image

3.7 FR7: Apply Thresholding

- **Description:** The system shall apply global and adaptive thresholding on the grayscale image.
- **Input:** Grayscale image.
- **Output:** Thresholded image.

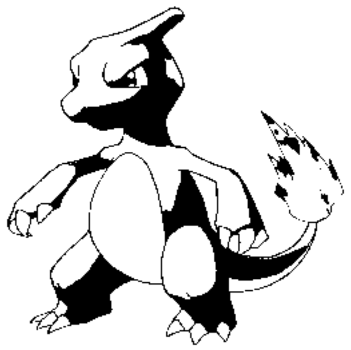

Figure 7: Global Thresholded Image

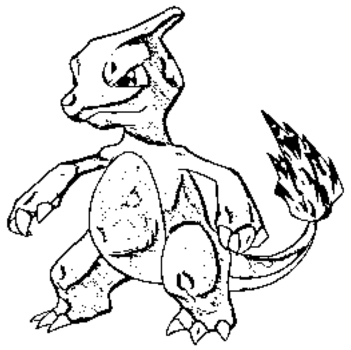

Figure 8: Adaptive Thresholded Image

3.8 FR8: Save Processed Images to Disk

- **Description:** The system shall save each processed image to a specified directory.
- **Input:** Processed images.
- **Output:** Images saved to disk in a directory named `cv_output`.

4 Deliverables

- A Python script implementing the described functionalities.
- A main section that processes an image using these functions and saves the results.
- Processed images saved in a directory named `cv_output`.