کد شبیه سازی مقاله و فراخوانی دیتاست

نرگس خاتمی

```
import cv2
import numpy as np
import tkinter as tk
from tkinter import filedialog
from PIL import ImageTk, Image

# Image enhancement functions
def histogram_equalization(img):
    return cv2.equalizeHist(img)

def clahe(img):
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    return clahe.apply(img)



def bpdhe(img):
    hist, bins = np.histogram(img.flatten(), 256, [0, 256])
    cdf = hist.cumsum()
    cdf_normalized = cdf * hist.max() / cdf.max()

    # Calculate the histogram equalization transform
    lut = np.interp(np.arange(256), bins[:-1], cdf_normalized).astype(np.uint8)
```

```
    # Apply the transform to the input image
    equalized_img = cv2.LUT(img, lut)

    return equalized_img


def agcwd(img, gamma=1, alpha=0.5, beta=0.25):
    # Calculate the image gradient
    gradient_x = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
    gradient_y = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
    gradient_magnitude = cv2.magnitude(gradient_x, gradient_y)

    # Calculate the adaptive gamma correction weights
    weight = np.power(gradient_magnitude, alpha)
    weight = cv2.normalize(weight, None, 0, 255, cv2.NORM_MINMAX)
    weight = np.power(weight, beta)

    # Apply gamma correction to the input image
    img_gamma = np.power(img / 255.0, gamma)

    # Apply the weighting distribution to the gamma-corrected image
    img_corrected = cv2.multiply(img_gamma, weight)
```

```
    # Normalize the corrected image to the range [0, 255]
    img_corrected = cv2.normalize(img_corrected, None, 0, 255, cv2.NORM_MINMAX)
    img_corrected = img_corrected.astype(np.uint8)

    return img_corrected


def fcce(img, alpha=0.25, beta=0.75, gamma=2.5, epsilon=1e-5):
    # Convert the input image to floating-point format
    img = img.astype(np.float32) / 255.0

    # Calculate the local mean and standard deviation
    local_mean = cv2.blur(img, (3, 3))
    local_std = cv2.sqrt(cv2.blur(img * img, (3, 3)) - local_mean * local_mean)

    # Calculate the contrast gain function
    contrast_gain = alpha + beta * (local_std / (local_std + epsilon))

    # Apply the contrast gain to enhance the image
    enhanced_img = np.power(img, gamma) * contrast_gain

    # Normalize the enhanced image to the range [0, 255]
```

```
    enhanced_img = cv2.normalize(enhanced_img, None, 0, 255, cv2.NORM_MINMAX)
    enhanced_img = enhanced_img.astype(np.uint8)

    return enhanced_img


def ecs(img, population_size=10, max_iterations=100, pa=0.25, alpha=1.5, sigma=0.2):
    # Convert the input image to floating-point format
    img = img.astype(np.float32) / 255.0

    # Initialize the cuckoo population with random solutions
    population = np.random.rand(population_size, *img.shape) * 255.0

    for iteration in range(max_iterations):
        print(f"Iteration {iteration+1}/{max_iterations}")

        # Evaluate the fitness of the population
        fitness = fitness_function(population, img)

        # Sort the population based on fitness in descending order
        sorted_indices = np.argsort(fitness)[::-1]
        population = population[sorted_indices]
        fitness = fitness[sorted_indices]
```

```
        # Generate new solutions using Levy flights
        new_population = levy_flight(population, alpha, sigma)

        # Evaluate the fitness of new solutions
        new_fitness = fitness_function(new_population, img)

        # Replace low-fitness solutions with new solutions
        num_replacements = int(pa * population_size)
        replace_indices = np.random.choice(population_size, size=num_replacements,
replace=False)
        population[replace_indices] = new_population[replace_indices]
        fitness[replace_indices] = new_fitness[replace_indices]

    # Return the best solution (cuckoo) as the enhanced image
    best_solution = population[0]
    enhanced_img = np.clip(best_solution, 0, 255).astype(np.uint8)

    return enhanced_img

# Fitness function for ECS (example: mean squared error)
def fitness_function(population, img):
    errors = np.mean((population - img) ** 2, axis=(1, 2))
```

```
    return -errors


# Levy flight function for generating new solutions
def levy_flight(population, alpha, sigma):
    levy = np.random.power(alpha, size=population.shape)
    levy = np.sign(np.random.randn(*population.shape)) * sigma * (levy ** (-1 / alpha))
    new_population = population + levy
    new_population = np.clip(new_population, 0, 255)
    return new_population


# GUI functions
def upload_image():
    file_path = filedialog.askopenfilename(filetypes=(("Image files", "*.jpg;*.jpeg;*.png"),
("All files", "*.*")))
    if file_path:
        global original_img_gray, original_img_tk
        img = cv2.imread(file_path)
        original_img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        original_img_pil = Image.fromarray(img)
        original_img_tk = ImageTk.PhotoImage(original_img_pil)
        input_label.config(image=original_img_tk)
        input_label.image = original_img_tk
```

```
def enhance_image():
    selected_method = method_var.get()
    if selected_method == "Histogram Equalization":
        processed_img = histogram_equalization(original_img_gray)
    elif selected_method == "CLAHE":
        processed_img = clahe(original_img_gray)
    elif selected_method == "BPDHE":
        processed_img = bpdhe(original_img_gray)
    elif selected_method == "AGCWD":
        processed_img = agcwd(original_img_gray)
    elif selected_method == "FCCE":
        processed_img = fcce(original_img_gray)
    elif selected_method == "ECS":
        processed_img = ecs(original_img_gray)
        return

    # Call the display_processed_image() function with the processed image and title
    display_processed_image(processed_img, selected_method)
    processed_img_label = None
    processed_img_rgb = cv2.cvtColor(processed_img, cv2.COLOR_GRAY2RGB)
    processed_img_pil = Image.fromarray(processed_img_rgb)
    processed_img_tk = ImageTk.PhotoImage(processed_img_pil)
```

```
    # Display the processed image in the new window
    processed_img_label.config(image=processed_img_tk)
    processed_img_label.image = processed_img_tk

    processed_img_label = tk.Label(output_window, image=processed_img_tk)
    processed_img_label.pack()



# Create the main window
window = tk.Tk()
window.title("Image Enhancement - Created by Narges Khatami")

# Upload image button
upload_button = tk.Button(window, text="Upload Image", command=upload_image)
upload_button.pack()

# Original image display
original_img_gray = None
original_img_tk = None
input_label = tk.Label(window)
input_label.pack()

# Enhancement method selection
```

```
method_var = tk.StringVar(window)
method_var.set("Histogram Equalization")
method_options = ["Histogram Equalization", "CLAHE", "BPDHE", "AGCWD", "FCCE", "ECS"]
method_menu = tk.OptionMenu(window, method_var, *method_options)
method_menu.pack()

# Apply enhancement button
enhance_button = tk.Button(window, text="Enhance", command=enhance_image)
enhance_button.pack()

# Processed image display
output_label = tk.Label(window)
output_label.pack()

def display_processed_image(processed_img, title):
    processed_window = tk.Toplevel()
    processed_window.title(title)

    processed_img_rgb = cv2.cvtColor(processed_img, cv2.COLOR_GRAY2RGB)
    processed_img_pil = Image.fromarray(processed_img_rgb)
    processed_img_tk = ImageTk.PhotoImage(processed_img_pil)
```

```
    processed_img_label = tk.Label(processed_window, image=processed_img_tk)
    processed_img_label.pack()

    processed_window.mainloop()



# Start the GUI main loop
window.mainloop()
```