Human Image Similarity Analysis using Fuzzy Logic

Name: Raman

Reg. No: 23MDT1048

1. Problem Statement:

We want to develop a system that can compare two human images and determine their similarity using fuzzy logic. The system should take into account the intensity difference and edge similarity between the images.

2. System Components:

- ***Image Similarity Fuzzy System****:* This is the core component of our system. It defines the fuzzy variables (intensity difference and edge similarity), membership functions, rules, and the defuzzification process to compute the similarity level between two images.
- ***Image Processing Functions****:* These functions are responsible for loading the images, computing features (such as intensity difference and edge similarity), and preparing the images for comparison.
- ***Main Function****:* This function orchestrates the overall process. It interacts with the user to input the paths of the images, calls the necessary functions to process the images and compute their similarity, and displays the results.

3. Workflow:

1. ***Input****:* The user provides the paths to two human images that they want to compare.
2. ***Image Processing****:* The system loads the images and computes two features for comparison:
    - ***Intensity Difference****:* The absolute difference in mean intensity between the two images.
    - ***Edge Similarity***: The similarity between the edges detected in the images.
3. ***Fuzzy System Computation****:* The system passes the computed features (intensity difference and edge similarity) to the fuzzy system. The fuzzy system uses predefined membership functions, rules, and the Mamdani inference method to compute the similarity level between the images.
4. ***Output****:* The system presents the similarity level and value to the user, indicating how similar the two images are.

4. Implementation:

- ***Image Similarity Fuzzy System****:* Implemented using the **skfuzzy** library in Python. It defines the fuzzy variables (**intensity_diff**, **edge_similarity**, **similarity**), membership functions, rules, and the defuzzification process.

- ***Image Processing Functions****:* Implemented using the **PIL** (Python Imaging Library) and numpy. These functions load the images, process them to compute the required features, and prepare them for comparison.
- ***Main Function****:* Orchestrates the process by interacting with the user, calling the necessary functions, and presenting the results.

5. Robustness and Error Handling:

- The system is designed to handle various types of human images, including differences in formats, sizes, and qualities.
- Error handling mechanisms are implemented throughout the system to catch and gracefully handle exceptions, such as invalid file paths, errors in image loading, and issues in fuzzy system computation.

By following this approach, we can develop a robust and effective system for comparing human images using fuzzy logic, providing users with insights into the similarity between the images.

CODE:

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
from PIL import Image, ImageFilter
import os


class ImageSimilarityFuzzySystem:
    def __init__(self):
        self.intensity_diff = ctrl.Antecedent(np.arange(0, 256, 1), 'intensity_diff')
        self.edge_similarity = ctrl.Antecedent(np.arange(0, 101, 1), 'edge_similarity')
        self.similarity = ctrl.Consequent(np.arange(0, 101, 1), 'similarity')
        self._setup_variables()
        self._setup_rules()

    def _setup_variables(self):
        names = ['low', 'medium', 'high']
        self.intensity_diff.automf(names=names)
        self.edge_similarity.automf(names=names)
        self.similarity.automf(names=names)

    def _setup_rules(self):
```

```
        self.rules = [
            ctrl.Rule(self.intensity_diff['low'] & self.edge_similarity['low'], self.similarity['high']),
            ctrl.Rule(self.intensity_diff['medium'] & self.edge_similarity['medium'], self.similarity['medium']),
            ctrl.Rule(self.intensity_diff['high'] & self.edge_similarity['high'], self.similarity['low'])
        ]

    def create_system(self):
        return ctrl.ControlSystem(self.rules)

    def compute_similarity(self, intensity_diff_input, edge_similarity_input):
        similarity_ctrl = self.create_system()
        similarity_estimator = ctrl.ControlSystemSimulation(similarity_ctrl)
        similarity_estimator.input['intensity_diff'] = intensity_diff_input
        similarity_estimator.input['edge_similarity'] = edge_similarity_input
        similarity_estimator.compute()
        similarity_value = similarity_estimator.output['similarity']
        return similarity_value

def compute_features(image1, image2):
    # Compute features for comparison (e.g., intensity difference, edge similarity)
    intensity_diff = np.abs(np.mean(image1) - np.mean(image2))

    # Compute edge similarity
    edge_similarity = compute_edge_similarity(image1, image2)

    return intensity_diff, edge_similarity


def compute_edge_similarity(image1, image2):
    if image1 == image2:
        # If both images are identical, return a default similarity value
        return 100.0

    # Apply edge detection filters
    edge_image1 = image1.filter(ImageFilter.FIND_EDGES)
    edge_image2 = image2.filter(ImageFilter.FIND_EDGES)
```

```
    # Resize images to have the same dimensions
    min_width = min(image1.width, image2.width)
    min_height = min(image1.height, image2.height)
    edge_image1 = edge_image1.resize((min_width, min_height))
    edge_image2 = edge_image2.resize((min_width, min_height))

    # Convert images to numpy arrays
    edge_array1 = np.array(edge_image1)
    edge_array2 = np.array(edge_image2)

    # Compute edge similarity
    similarity = np.sum(edge_array1 == edge_array2) / (min_width * min_height) * 100
    # Ensure a minimum threshold for similarity to avoid total area zero error
    min_similarity_threshold = 1.0  # You can adjust this threshold as needed
    edge_similarity = max(similarity, min_similarity_threshold)
    return edge_similarity
def main():
    # Create instance of the fuzzy system
    fuzzy_system = ImageSimilarityFuzzySystem()

    # Get input paths from the user
    image1_path = input("Enter path to first image: ").strip()
    image2_path = input("Enter path to second image: ").strip()

    # Check if files exist
    if not (os.path.isfile(image1_path) and os.path.isfile(image2_path)):
        print("One or both of the provided paths are invalid.")
        return

    # Load images
    try:
        image1 = Image.open(image1_path).convert("L")
        image2 = Image.open(image2_path).convert("L")
    except Exception as e:
        print(f"Error loading images: {e}")
```

```
        return

    # Check if images are identical
    if image1 == image2:
        print("The provided images are identical.")
        return

    # Compute features for comparison
    intensity_diff, edge_similarity = compute_features(image1, image2)
    similarity_value = fuzzy_system.compute_similarity(intensity_diff, edge_similarity)
    print("Similarity value:", similarity_value)


if __name__ == "__main__":
    main()
```

Output:

"C:\Users\Satoshi\OneDrive\Desktop\Data\PERSONAL_GROWTH\mini-projects\Images\Human iamge Fuzzy System\human_similarity.py"

Enter path to first image: C:\Users\Satoshi\Downloads\IMG_20201011_214611_894.jpg

Enter path to second image: C:\Users\Satoshi\Downloads\IMG_20201228_115441.jpg

Similarity value: 71.02626018028907