***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*