**Image Print Quality Controller**

**Fuzzy Sets and Membership Functions:**

|  |  |
| --- | --- |
| Input Variable: *Paper Type* | |
| **Linguistic Terms** | **Universe of Discourse** |
| Low | [0, 0, 15, 25] |
| Standard | [20, 30, 40, 50] |
| High | [50, 60, 70, 80] |

|  |  |
| --- | --- |
| Input Variable: *Image Sharpness* | |
| **Linguistic Terms** | **Universe of Discourse** |
| Blurry | [0, 0, 50] |
| Slightly Blurry | [0, 50, 100] |
| Sharp | [50, 100, 100] |

**Output Variable:** Print Quality

Rules:

|  |  |  |
| --- | --- | --- |
| **Image Sharpness** | **Paper Type** | **Print Quality** |
| Blurry | Low | draft |
| Standard | high |
| High | high |
| Slightly Blurry | Low | draft |
| Standard | high |
| High | standard |
| Sharp | Low | high |
| Standard | standard |
| High | standard |

Code:

import numpy as np

import skfuzzy as fuzz

import matplotlib.pyplot as plt

from skfuzzy import control as ctrl

# Universe of discourse (x-axis range)

img\_sharpness = ctrl.Antecedent(np.arange(0, 101, 1), 'img\_sharpness')

clr\_accuracy = ctrl.Antecedent(np.arange(0, 101, 1), 'clr\_accuracy')

print\_quality = ctrl.Consequent(np.arange(0, 101, 1), 'print\_quality')

# Image Sharpness Fuzzy Sets

img\_sharpness ['blurry'] = fuzz.trapmf(img\_sharpness.universe, [0, 0, 15, 25])

img\_sharpness ['slightly\_blurry'] = fuzz.trapmf(img\_sharpness.universe, [20, 30, 40, 50])

img\_sharpness ['sharp'] = fuzz.trapmf(img\_sharpness.universe, [50, 60, 70, 80])

img\_sharpness ['very\_sharp'] = fuzz.trapmf(img\_sharpness.universe, [75, 85, 100, 100])

# Color Accuracy Fuzzy Sets

clr\_accuracy ['inaccurate'] = fuzz.trapmf(clr\_accuracy.universe, [0, 0, 15, 25])

clr\_accuracy ['slightly\_inaccurate'] = fuzz.trapmf(clr\_accuracy.universe, [20, 30, 40, 50])

clr\_accuracy ['accurate'] = fuzz.trapmf(clr\_accuracy.universe, [50, 60, 70, 80])

clr\_accuracy ['very\_accurate'] = fuzz.trapmf(clr\_accuracy.universe, [75, 85, 100, 100])

# Output: Print Quality

print\_quality ['poor'] = fuzz.trimf(print\_quality.universe, [0, 0, 50])

print\_quality ['standard'] = fuzz.trimf(print\_quality.universe, [0, 50, 100])

print\_quality ['excellent'] = fuzz.trimf(print\_quality.universe, [50, 100, 100])

# Rules

rule1 = ctrl.Rule(img\_sharpness['blurry'] & clr\_accuracy['inaccurate'], print\_quality['poor'])

rule2 = ctrl.Rule(img\_sharpness['blurry'] & clr\_accuracy['slightly\_inaccurate'], print\_quality['poor'])

rule3 = ctrl.Rule(img\_sharpness['blurry'] & clr\_accuracy['accurate'], print\_quality['poor'])

rule4 = ctrl.Rule(img\_sharpness['blurry'] & clr\_accuracy['very\_accurate'], print\_quality['standard'])

rule5 = ctrl.Rule(img\_sharpness['slightly\_blurry'] & clr\_accuracy['inaccurate'], print\_quality['poor'])

rule6 = ctrl.Rule(img\_sharpness['slightly\_blurry'] & clr\_accuracy['slightly\_inaccurate'], print\_quality['poor'])

rule7 = ctrl.Rule(img\_sharpness['slightly\_blurry'] & clr\_accuracy['accurate'], print\_quality['standard'])

rule8 = ctrl.Rule(img\_sharpness['slightly\_blurry'] & clr\_accuracy['very\_accurate'], print\_quality['standard'])

rule9 = ctrl.Rule(img\_sharpness['sharp'] & clr\_accuracy['inaccurate'], print\_quality['standard'])

rule10 = ctrl.Rule(img\_sharpness['sharp'] & clr\_accuracy['slightly\_inaccurate'], print\_quality['standard'])

rule11 = ctrl.Rule(img\_sharpness['sharp'] & clr\_accuracy['accurate'], print\_quality['excellent'])

rule12 = ctrl.Rule(img\_sharpness['sharp'] & clr\_accuracy['very\_accurate'], print\_quality['excellent'])

rule13 = ctrl.Rule(img\_sharpness['very\_sharp'] & clr\_accuracy['inaccurate'], print\_quality['standard'])

rule14 = ctrl.Rule(img\_sharpness['very\_sharp'] & clr\_accuracy['slightly\_inaccurate'], print\_quality['standard'])

rule15 = ctrl.Rule(img\_sharpness['very\_sharp'] & clr\_accuracy['accurate'], print\_quality['excellent'])

rule16 = ctrl.Rule(img\_sharpness['very\_sharp'] & clr\_accuracy['very\_accurate'], print\_quality['excellent'])

# Control System and Simulation

qual\_indicator = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9, rule10, rule11, rule12, rule13, rule14, rule15, rule16])

qualInd\_sim = ctrl.ControlSystemSimulation(qual\_indicator)

# Labels and title

img\_sharpness.view()

plt.title('Image Sharpness Membership Functions')

plt.xlabel('Image Sharpness (%)')

plt.ylabel('Membership Value')

plt.legend()

clr\_accuracy.view()

plt.title('Color Accuracy Membership Functions')

plt.xlabel('Color Accuracy (%)')

plt.ylabel('Membership Value')

plt.legend()

print\_quality.view()

plt.title('Print Quality')

plt.xlabel('Image Quality (%)')

plt.ylabel('Membership Value')

plt.legend()

# Display the plot

plt.grid(True)

plt.show()

degOf\_img\_sharpness = float(input("Enter Degree of Image Sharpness [0-100]: "))

degOf\_clr\_accuracy = float(input("Enter Degree of Color Accuracy [0-100]: "))

# input Values

qualInd\_sim.input['img\_sharpness'] = degOf\_img\_sharpness

qualInd\_sim.input['clr\_accuracy'] = degOf\_clr\_accuracy

qualInd\_sim.compute()

print(qualInd\_sim.output['print\_quality'])

print\_quality.view(sim=qualInd\_sim)

scikit-fuzzy is a Python package that provides tools for projects involving fuzzy logic, also known as grey logic. [It is dependent on NumPy, SciPy, and NetworkX1](https://anaconda.org/conda-forge/scikit-fuzzy). [The skfuzzy.control submodule provides a Pythonic, object-oriented system for fuzzy control systems1](https://anaconda.org/conda-forge/scikit-fuzzy).

Fuzzy control systems are used to control complex systems that are difficult to model mathematically. [They use fuzzy logic to represent the system’s behavior and make decisions based on that behavior2](https://pythonhosted.org/scikit-fuzzy/user_guide.html).

To use scikit-fuzzy’s fuzzy control system, you need to define the following components:

* **Antecedents**: Input variables that affect the output.
* **Consequents**: Output variables that are affected by the input variables.
* **Rules**: A set of rules that define how the input variables affect the output variables.

Once you have defined these components, you can create a ControlSystem object and add your antecedents, consequents, and rules to it. [You can then create a ControlSystemSimulation object and use it to simulate your control system2](https://pythonhosted.org/scikit-fuzzy/user_guide.html).

In scikit-fuzzy, **ctrl.ControlSystemSimulation** is a class that allows you to simulate a fuzzy control system. It provides a way to input crisp values, perform computations based on the defined rules, and obtain output values. This class is used to interact with the fuzzy logic control system and see how it responds to different inputs.

The **ctrl.ControlSystem()** is a constructor in scikit-fuzzy that is used to create a control system. It takes a list of rules as an argument.

**Image Print Quality Indicator**

This passage is describing a special system that's used in printing. Its main purpose is to make sure that pictures and graphics look really good when they're printed on things like posters, packaging, and ads. What makes this system unique is that it uses a smart method called "fuzzy logic" to figure out how to make the prints even better. It takes into account lots of things that can affect how good the pictures look, and it gives a complete report on how good they are. And the best part is, it can make adjustments while it's in the middle of printing. This system not only guarantees that the prints will be consistently excellent, but it also helps save resources and money by reducing waste and production costs. This system uses both specialized equipment and computer programs to watch over the printing process. It keeps an eye out for any mistakes or problems in the printed pictures or text. It does this by using different methods to check things like color accuracy, density, and visual appearance. The main aim is to make sure that what comes out of the printer looks just right, without any issues like wrong colors or misalignment.