Fuzzy Logic System and Its Applications

Tianrun Qiu
12210829@mail.sustech.edu.cn
Southern University of Science and Technology
Shenzhen, Guangdong, China

ABSTRACT

This paper explores the concept of fuzzy logic system, which was introduced as an alternative to traditional Boolean logic due to its limitations in modern applications. Fuzzy logic, being a type of multivalued logic, allows for more flexibility in representing real-world scenarios. The properties of the fuzzy logic system, including complement, AND, OR, inference, and equivalence, are discussed based on traditional Boolean logic. Additionally, fuzzy set operations and defuzzification methods are explained. The paper presents two specific applications of fuzzy logic: controlling the color temperature of screens and optimizing image contrast. For each application, detailed definitions and methodologies are provided to demonstrate the effectiveness of fuzzy logic in addressing non-binary problems. The results showcase the practicality and adaptability of fuzzy logic in diverse domains.

KEYWORDS

Fuzzy logic, Boolean logic, multivalued logic, fuzzy set, affiliation function, defuzzification, color temperature control, image processing, image contrast

ACM Reference Format:
Tianrun Qiu. 2023. Fuzzy Logic System and Its Applications. In *Discrete Math, SUSTech, China.* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

In the first and second chapters of the textbook *Discrete Mathematics and Its Applications*[1], Boolean logic and set properties are introduced.

And it is also mentioned in the textbook, that in modern applications, traditional Boolean logic system has its own limitations[1], which resulted in the introduction of the fuzzy logic system. And it is encouraged by the book that we explore the subject of fuzzy logic by ourselves.

The concept of fuzzy logic originated when Lotfi Zadeh, an Iranian-Azerbaijani mathematician residing in the United States, put forth the idea of fuzzy set theory in 1965[2].

In this paper, we will focus on the properties of the fuzzy logic system based on traditional Boolean logic and two specific applications of this system.

2 PROPERTIES OF THE SYSTEM

In this section, we will show the formulas of which the fuzzy system follow, based on the traditional Boolean logic structure.

2.1 Fuzzy Logic Properties

In traditional logic, any Boolean value can have its value either *true* or *false*, and there is no other values available. However, fuzzy logic is a type of multivalued logic. In fuzzy logic, the value of the variable can be any real number in the interval [0,1][3], which suits this non-binary real world better in a variety of cases.

It is not a hard task to derive five basic rules below:

- Complement: $\overline{P} = 1 - P$
- And: $P \wedge Q = \min(P, Q)$
- Or: $P \vee Q = \max(P, Q)$
- Inference: $P \rightarrow Q = ((1 - P) \vee Q)$
- Equivalence: $P \leftrightarrow Q = (P \rightarrow Q) \wedge (Q \rightarrow P)$

Then, we can clearly show that the all five basic rules of traditional Boolean logic have its variation for fuzzy logic system. Then, it is assumable that other rules such as Idempotent Rule, Association Rule, etc., should also work for fuzzy logic system.

2.2 Fuzzy Set Operations

Similar to the fuzzy logic properties, we can also describe fuzzy set operations based on traditional set operations[3].

First, we may define μ as the degree of membership, which describe how much an item belongs to the set, as in fuzzy sets, elements are not always 100% a member of a set. The degree of membership is commonly calculated using a defined membership function of the fuzzy set, commonly a triangular function or Gaussian function[3].

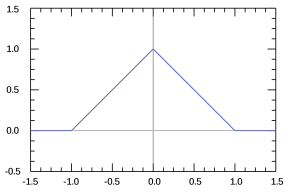

Figure 1: Triangular Function

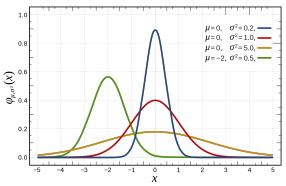

Figure 2: Gaussian Function

Then, we can describe a fuzzy set as

$$A = (u, \mu_A(u))|u \in U$$

and we may as well define the intersection, union and complement operations of fuzzy sets:

- Intersection: $\mu_A \cap B(u) = \min(\mu_A(u), \mu_B(u))$
- Union: $\mu_A \cup B(u) = \max(\mu_A(u), \mu_B(u))$
- Complement: $\mu_{\overline{A}}(u) = 1 - \mu_A(u)$

Therefore, other operations can as well be deduced.

2.3 Defuzzification

Defuzzification is a common step to derive fuzzy sets and corresponding membership degrees into a quantitative result in fuzzy logic system[5].

A trivial way to derive such a quantitative result is simply using the union function we discussed before to acquire the maximum membership degree as the result. However, this method is imperfect as it may cause information loss.

Therefore, computer scientists posed several methods of defuzzification, and the most famous ones might be the *adaptive integration method*, the *basic defuzzification distributions method* and the *center of gravity method*[5]. For convenience, we will use the *center of gravity method* in this paper.

This method can be represented in

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

where w_i is the weight, and x_i represent the position on the axis.

3 APPLICATION 1: COLOR TEMPERATURE CONTROLLING PROCESS

Then, we can utilize the tool of fuzzy logic system to several applications to suit the non-binary world better. The first original example is about better controlling process of screen color temperature, which is a typical problem which can be conquered better using fuzzy logic controlling method.

3.1 The traditional binary logic way

Studies have found that blue light emitted from computer screens may affect sleep quality. To solve this problem, operating system such as Windows, macOS, Android and iOS all carried out their own solution to this problem: an applet to alter the color temperature of the screen to be warmer[4]. However, these applets all come out with a Boolean toggle - either on or off.

In traditional Boolean logic, we can define this workflow as

$$(T \in S) \rightarrow K$$

where S is the time interval with Night Shift mode on, K is the condition of the mode switched on.

3.2 Using fuzzy logic

Then, using the fuzzy logic, we may redefine $K \in [0, 1]$ as the extent of extra warmness, $T = \mu_t(s)$ as the degree of affiliation, we can rewrite the formula as

$$T \rightarrow K \text{ (under fuzzy logic)}$$

to fit in the fuzzy logic system.

3.3 Detailed definition

For the time T, we can define three fuzzy sets Low, Mid and $High$ with triangular functions as their affiliation function.

It will be similar to the function below, where at $t = a$, the function starts to rise, at $t = b$ the function arrives at its peak and in $t = c$ drops back to 0:

$$\mu(t) = \begin{cases} 0 & \text{if } t \le a \\ \frac{t-a}{b-a} & \text{if } a \le t \le b \\ \frac{c-t}{c-b} & \text{if } b \le t \le c \\ 0 & \text{if } t \ge c \end{cases}$$

And we need to define the parameters $a, b, c \in [0, 24)$ for morning, daytime and evening, describing the hours of these periods. In this specific case, we may select the parameters below. These selection is based on the real sleeping data of a typical SUSTech CSE student.

- High 1: $a = b = 0, c = 4$
- Mid 1: $a = 2, b = 4.5, c = 7$
- Low: $a = 3.5, b = 10.5, c = 17.5$
- Mid 2: $a = 14, b = 18.5, c = 23$
- High 2: $a = 19, b = c = 24$

The function is steeper for the first half of the day, because high color temperature of the screen is generally not needed for that time. And it can be described in a graph:

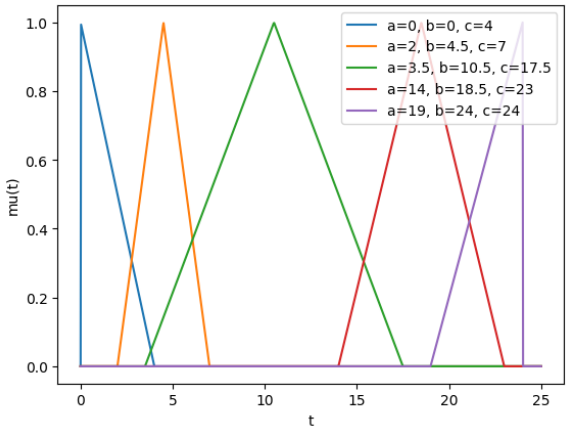

Figure 3: The five functions

And we may select $High = 3200\text{K}, Mid = 5100\text{K}, Low = 6400\text{K}$, where K is the unit of color temperature.

The resulting color temperature can be determined using $\mu_{High1}(t)$, $\mu_{Mid1}(t)$, $\mu_{Low}(t)$, $\mu_{Mid2}(t)$ and $\mu_{High2}(t)$ based on fuzzy logic axioms.

At last, we perform the defuzzification step using the Center of Gravity method[5]

$$T = \frac{\mu_{High1}(t) * High + \mu_{Mid1}(t) * Mid + \mu_{Low}(t) * Low + \mu_{Mid2}(t) * Mid + \mu_{High2} * High}{\mu_{High1}(t) + \mu_{Mid1}(t) + \mu_{Low}(t) + \mu_{Mid2}(t) + \mu_{High2}(t)}$$

for a good result.

3.4 Implementation report

In this part, a macOS app is developed using the Electron framework, in which the better color temperature control process we posed was implemented. Javascript is used as the main programming language.

The implementation basically includes two parts, a configuration window and a mask window, and the algorithm runs with time as the input.

Some implementations screenshot and the result color temperature curve with $t \in (0, 24)$ are shown below.

For more implementation technology details, please refer to the appendixes.

Figure 4: When $t = 00 : 47$

Figure 5: When $t = 15 : 47$

Figure 6: When $t = 21 : 47$

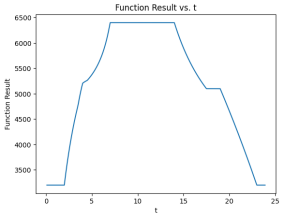

Figure 7: The output curve

It is not hard to perceive that we acquired a rather smooth curve using our procedure based on fuzzy logic knowledge.

4 APPLICATION 2: IMPROVING IMAGE CONTRAST

Fuzzy logic system also finds its usage in the field of image processing. This case is inspired by a Kaggle notebook[6], but the description and implementation are original. In this case, we want to explore the possibility to implement a image contrast enhancer using fuzzy inference system.

4.1 The way to conquer using fuzzy logic

The image contrast can be enhanced by darkening dark pixels and brightening light pixels. Therefore, we may derive any pixels into seven fuzzy sets, $Black$, $VeryDark$, $Dark$, $Medium$, $Light$, $VeryLight$ and $White$.

And we may infer obeying these logical inference rules:

- $VeryDark \longrightarrow Black$
- $Dark \longrightarrow VeryDark$
- $Light \longrightarrow VeryLight$
- $VeryLight \longrightarrow White$

And image contrast will be enhanced after such a procedure. At the end, we perform the defuzzification task to get the final result.

4.2 Detailed definition

In this implementation, we will use the Gaussian function to describe membership degrees here. The Gaussian function is of this type:

$$g(x, a, b, c) = a \exp\left(-\frac{(x-b)^2}{2c^2}\right)$$

where a is the peak height ($a = 1.0$ in this case, and will be omitted below), b is the position of peak, and c is the width of the bell-like graph. In this case, the width c should be dependent on the mean color (defined as $mc \in (0, 255)$ for convenience) for a more reliable result.

We may define the seven fuzzy sets specifically as:

- $Black$, membership function $g(x, 0, \frac{mc}{6})$
- $VeryDark$, membership function $g(x, \frac{mc}{3}, \frac{mc}{6})$
- $Dark$, membership function $g(x, \frac{2mc}{3}, \frac{mc}{6})$
- $Medium$, membership function $g(x, mc, 23)$
- $Light$, membership function $g(x, 255 - \frac{255-mc}{3}, \frac{255-mc}{6}))$
- $VeryLight$, membership function $g(x, 255-2(\frac{255-mc}{3}), \frac{255-mc}{6})$
- $White$, membership function $g(x, 255, \frac{255-mc}{6})$

We may demonstrate using a graph below:

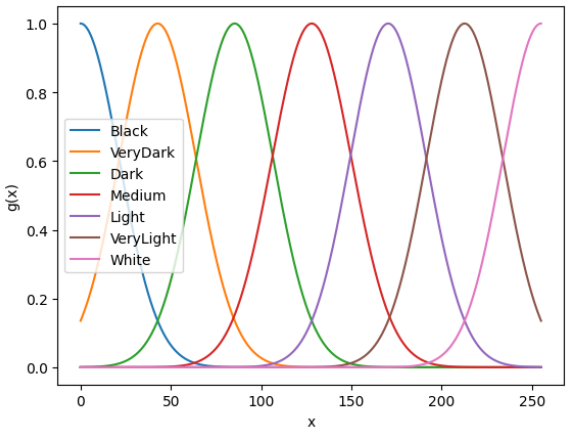

Figure 8: The seven fuzzy groups (when $mc = 128$)

Then, We may perform fuzzy logic operation $Black \cap VeryDark \cap Dark \cap Medium \cap Light \cap VeryLight \cap White$ to acquire which group the color at a specific pixel belongs to the most, and then perform the inference function we discussed before to generate the enhanced colors.

4.3 Implementation report

In the implementation:

- Python was used as the programming language in this case.
- OpenCV-Python was used to perform image operations.
- Numpy was used to do calculations.
- TQDM and Matplotlib are also used in our implementation.

The technological details can be found in the appendixes.

And below we will demonstrate some of our example images processed by our precedure:

In the first example, the sky is brighter and the shadow under the rooftop is darker. In the second example, the moon now is more vivid. In the third example, the snow is brighter and trees get darker.

To sum up, all three photos' contrast enhanced while preserving details and photographic quality.

Figure 9: Origin 1

Figure 10: Output 1

Figure 11: Origin 2

Figure 12: Output 2

Figure 13: Origin 3

Figure 14: Output 3

5 CONCLUSION

In conclusion, fuzzy logic proves to be a powerful tool for addressing complex and ambiguous problems that cannot be adequately represented using traditional Boolean logic.

By allowing for a range of values between true and false, fuzzy logic provides a more realistic representation of real-world circumstances. The properties of fuzzy logic, such as complement, AND, OR, inference, and equivalence, extend the capabilities of traditional Boolean logic and enable the handling of non-binary situations.

Fuzzy set operations, including intersection, union, and complement, facilitate the manipulation of fuzzy sets based on traditional set operations.

The defuzzification process, employing methods such as the Center of Gravity, allows for converting fuzzy sets into quantitative results.

The applications of fuzzy logic in controlling the color temperature of screens and optimizing image contrast demonstrate its practicality and effectiveness in different real-life domains.

Further research and exploration of fuzzy logic may lead to advancements in solving non-binary problems and improving a variety of processes across various fields.

6 ACKNOWLEDGEMENT

To Dr. S. Chen, for his Discrete Math classes.

REFERENCES

[1] Kenneth H. Rosen. *Discrete Mathematics and Its Applications.* McGraw-Hill. 2012.
[2] Stanford Encyclopedia of Philosophy. *Fuzzy Logic.* Bryant University. 23 July 2006.
[3] J. Zhang, Z. Zhan. *Computational Intelligence (Ji Suan Zhi Neng).* Tsinghua University Press. 1 November 2009.
[4] PCMag. *Ready For Bed? How to Stop Blue Light From Disturbing Your Sleep.* https://www.pcmag.com/how-to/how-to-stop-blue-light-from-disturbing-your-sleep. 17 January 2023.
[5] W. van Leekwijck, E. E. Kerre. (1999). *Defuzzification: criteria and classification.* Fuzzy Sets and Systems. 1999.
[6] V. Nguyên. *Fuzzy Logic - Image Contrast Enhancement.* https://www.kaggle.com/code/nguyenvlm/fuzzy-logic-image-contrast-enhancement/notebook. 2020.

A CODES: GRAPH DRAWING

Listing 1: Application 1 Function Graph

```
import numpy as np
import matplotlib.pyplot as plt

def mu(t, a, b, c):
    if t <= a:
        return 0
    elif a <= t <= b:
        return (t - a) / (b - a)
    elif b <= t <= c:
        return (c - t) / (c - b)
    else:
        return 0

cases = [
    {'a': 0, 'b': 0, 'c': 4},
    {'a': 3, 'b': 6, 'c': 9},
    {'a': 8, 'b': 12.5, 'c': 17},
    {'a': 16, 'b': 19, 'c': 22},
    {'a': 21, 'b': 24, 'c': 24}
]

t = np.linspace(0, 25, 1000)
fig, ax = plt.subplots()

for case in cases:
    y = [mu(ti, case['a'], case['b'],case['c'])
for ti in t]
    ax.plot(t, y, label=f"a={case['a']}," +
f"b={case['b']},␣c={case['c']}")

ax.set_xlabel('t')
ax.set_ylabel('mu(t)')
ax.legend()
plt.show()
```

Listing 2: Application 1 Result Curve

```
## Application 2 Function Result Curve

High = 3200
Mid = 5100
Low = 6400

import numpy as np
import matplotlib.pyplot as plt


def mu(t, a, b, c):
    if t <= a:
        return 0
    elif a <= t <= b:
        return (t - a) / (b - a)
    elif b <= t <= c:
        return (c - t) / (c - b)
    else:
        return 0


def hi1(t):
    return mu(t, 0, 0, 4)


def mi1(t):
    return mu(t, 2, 4.5, 7)


def lo(t):
    return mu(t, 3.5, 10.5, 17.5)


def mi2(t):
    return mu(t, 14, 18.5, 23)


def hi2(t):
    return mu(t, 19, 24, 24)


def calculateT(t):
    mu_high1 = hi1(t)
    mu_mid1 = mi1(t)
    mu_low = lo(t)
    mu_mid2 = mi2(t)
    mu_high2 = hi2(t)

    numerator = (mu_high1 * High
+ mu_mid1 * Mid + mu_low * Low
+ mu_mid2 * Mid + mu_high2 * High)
    denominator = (mu_high1 + mu_mid1 + mu_low
+ mu_mid2 + mu_high2)

    if denominator != 0:
        return numerator / denominator
    else:
        return 0


t_values = np.linspace(0.1, 23.9, 10000)
result_values = [calculateT(t)
for t in t_values]

plt.plot(t_values, result_values)
plt.title('Function Result vs. t')
plt.xlabel('t')
plt.ylabel('Function Result')
plt.show()
```

Listing 3: Application 2 Function Graph

```
import numpy as np
import matplotlib.pyplot as plt


def g(x, b, c):
    return np.exp(-((x - b) ** 2)
/ (2 * c ** 2))


x = np.linspace(0, 255, 1000)
mc = 128

functions = [
    {'label': 'Black', 'b': 0,
'c': mc / 6},
    {'label': 'VeryDark', 'b': mc / 3,
'c': mc / 6},
    {'label': 'Dark', 'b': mc * 2 / 3,
'c': mc / 6},
    {'label': 'Medium', 'b': mc, 'c': 22},
    {'label': 'Light',
'b': 255 - (255 - mc) / 3 * 2,
'c': (255 - mc) / 6},
    {'label': 'VeryLight',
'b': 255 - (255 - mc) / 3,
'c': (255 - mc) / 6},
    {'label': 'White', 'b': 255,
'c': (255 - mc) / 6}
]

fig, ax = plt.subplots()
```

```
for function in functions:
    y = g(x, function['b'], function['c'])
    ax.plot(x, y, label=function['label'])

ax.set_xlabel('x')
ax.set_ylabel('g(x)')
ax.legend()
plt.show()
```

B CODES: APPLICATION 1

Listing 4: main.js

```
const {app, BrowserWindow} = require('electron')
const path = require('node:path')
const {ipcMain} = require("electron");
let win = null, mask = null;
const createWindow = () => {
    mask = new BrowserWindow({
        alwaysOnTop: true,
        transparent: true,
        focusable: false,
        frame: false,
        webPreferences: {
            preload: path.join(__dirname,
'preload.js'),
        }
    });

    win = new BrowserWindow({
        alwaysOnTop: true,
        width: 600,
        height: 300,
        webPreferences: {
            preload: path.join(__dirname,
'preload.js'),
        }
    });



    win.loadFile('index.html');
    mask.loadFile('mask.html');
    mask.maximize();
    mask.setIgnoreMouseEvents(true);
}

ipcMain.on('color', (event, arg) => {
    mask.webContents.send("color-toggle",
"rgba(" + arg.red + "," + arg.green + ","
+ arg.blue + ", 0.2)");
});

app.whenReady().then(() => {
    createWindow();
});
```

Listing 5: preload.js

```
const {contextBridge, ipcRenderer} =
require('electron');

contextBridge.exposeInMainWorld(
    'ipcRenderer',
    {
        send: (channel, arg) =>
ipcRenderer.send(channel, arg),
        on: (event, data) =>
ipcRenderer.on(event, data)
    }
)
```

Listing 6: renderer.js

```
const High = 3200, Mid = 5100, Low = 6400;

function mu(t, a, b, c) {
    if (t <= a) {
        return 0;
    } else if (a <= t && t <= b) {
        return (t - a) / (b - a);
    } else if (b <= t && t <= c) {
        return (c - t) / (c - b);
    } else {
        return 0;
    }
}

function hi1(t) {
    return mu(t, 0, 0, 4);
}

function mi1(t) {
    return mu(t, 2, 4.5, 7);
}

ma

function lo(t) {
    return mu(t, 3.5, 10.5, 17.5);
}

function mi2(t) {
    return mu(t, 14, 18.5, 23);
}

function hi2(t) {
    return mu(t, 19, 24, 24);
```

```
}

function calculateT(t) {
    let mu_high1 = hi1(t);
    let mu_mid1 = mi1(t);
    let mu_low = lo(t);
    let mu_mid2 = mi2(t);
    let mu_high2 = hi2(t);

    let numerator = (mu_high1 * High
+ mu_mid1 * Mid + mu_low * Low
+ mu_mid2 * Mid + mu_high2 * High);
    let denominator = (mu_high1
+ mu_mid1 + mu_low
+ mu_mid2 + mu_high2);

    if (denominator !== 0) {
        return Math.ceil(numerator
/ denominator);
    } else {
        return 0;
    }
}


function getColorFromTemperature(temperature) {
    temperature = temperature / 100;

    let red, green, blue;

    if (temperature <= 66) {
        red = 255;
        green = Math.min(Math.max(99.4708025861 *
Math.log(temperature) - 161.1195681661, 0), 255);
    } else {
        red = Math.min(Math.max(329.698727446 *
Math.pow(temperature - 60, -0.1332047592), 0)
, 255);
        green = Math.min(Math.max(288.1221695283
* Math.pow(temperature - 60, -0.0755148492), 0)
, 255);
    }

    if (temperature >= 66) {
        blue = 255;
    } else {
        if (temperature <= 19) {
            blue = 0;
        } else {
            blue =
Math.min(Math.max(138.5177312231 *
Math.log(temperature - 10) - 305.0447927307, 0)
, 255);
        }
    }

    return {red: Math.round(red),
green: Math.round(green),
blue: Math.round(blue)};
}

function getResult() {
    let hour = $("#hour").val();
    let minute = $("#minute").val();
    let result = calculateT(Number(hour) +
(Number(minute) / 60));
    $("#result").text(to2digit(hour) + ":" +
to2digit(minute) + ", " + result + "K");
    $("body").css("background-color", "rgb(" +
getColorFromTemperature(result).red + "," +
getColorFromTemperature(result).green + "," +
getColorFromTemperature(result).blue + ")");
    window.ipcRenderer.send("color",
getColorFromTemperature(result));
}

function to2digit(num) {
    return ("0" + num).slice(-2);
}

$(document).ready(function () {
    $("#hour").on("input", function () {
        getResult();
    });
    $("#minute").on("input", function () {
        getResult();
    });
    $("#hour").val(new Date().getHours());
    $("#minute").val(new Date().getMinutes());
    getResult();
});
```

Listing 7: index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <script type="text/javascript"
src="jquery.min.js"></script>
    <script type="text/javascript"
src="bootstrap.bundle.min.js">
</script>
    <link rel="stylesheet"
href="bootstrap.min.css">
```

```
    <title>Color Temperature Control</title>
    <style>
        html, body {
            height: 100%;
        }

        .container-fluid {
            height: 100vh;
        }

        #result {
            font-size: 2rem;
        }
    </style>
</head>
<body>
<div class="container-fluid text-center
w-100 h-100">
    <div class="row row align-items-center
h-100">
        <div class="col-2"></div>
        <div class="col-8">
            <label for="hour">Hour: </label>
<input id="hour" type="range" min="0" max="23"
value="12" class="form-range"/>
            <label for="minute">Minute: </label>
<input id="minute" type="range" min="0" max="59"
value="30" class="form-range"/>
            <div id="result"></div>
        </div>
        <div class="col-2"></div>
    </div>
</div>
<script src="renderer.js"></script>
</body>
</html>
```

Listing 8: mask.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8"/>
    <script type="text/javascript"
src="jquery.min.js"></script>
    <script type="text/javascript"
src="bootstrap.bundle.min.js">
</script>
    <link rel="stylesheet"
href="bootstrap.min.css">
    <title>Mask</title>
    <style>
        html, body {
            height: 100%;
        }
    </style>
</head>
<body>
<div
class="container-fluid text-center w-100 h-100">
</div>
<script src="renderer.js"></script>
<script>
    window.ipcRenderer.on('color-toggle',
(event, arg) => {
        document.body.style.backgroundColor
= arg;
    });
</script>
</body>
</html>
```

C CODES: APPLICATION 2

Listing 9: Application 2 Code

```
import cv2

image = cv2.imread('ori.png')

image = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
import matplotlib.pyplot as plt

rgb_image = cv2.cvtColor(image, cv2.COLOR_LAB2RGB)
plt.axis('off')
plt.imshow(rgb_image)

cnt = 0
l_sum = 0

height, width, channels = image.shape
for y in range(height):
    for x in range(width):
        h, s, l = image[y, x]
        cnt += 1
        l_sum += l

mc = l_sum / cnt

import numpy as np


def gaussian(x, b, c):
    return np.exp(-0.5 * ((x - b) * (x - b)
/ (2 * c * c)))
```

```
def bk(x):
    return gaussian(x, 0, mc / 6)


def vd(x):
    return gaussian(x, mc / 3, mc / 6)


def dk(x):
    return gaussian(x, mc * 2 / 3, mc / 6)


def md(x):
    return gaussian(x, mc, 23)


def lt(x):
    return gaussian(x, 255 - (255 - mc) / 3 * 2,
(255 - mc) / 6)


def vl(x):
    return gaussian(x, 255 - (255 - mc) / 3,
(255 - mc) / 6)


def wt(x):
    return gaussian(x, 255, (255 - mc) / 6)


from tqdm import tqdm

height, width, channels = image.shape
for y in tqdm(range(height)):
    for x in range(width):
        L, A, B = image[y, x]
        maxi = max(bk(L), vd(L), dk(L), md(L),
lt(L), vl(L), wt(L))
        if (vd(L) == maxi):
            L -= mc / 10
            maxi = max(bk(L), vd(L), dk(L),
md(L), lt(L), vl(L), wt(L))
        elif (dk(L) == maxi):
            L -= mc / 10
            maxi = max(bk(L), vd(L), dk(L),
md(L), lt(L), vl(L), wt(L))
        elif (lt(L) == maxi):
            L += (255 - mc) / 10
            maxi = max(bk(L), vd(L), dk(L),
md(L), lt(L), vl(L), wt(L))
        if (L < 0):
            L = 0
        elif (L > 255):
            L = 255
        image[y, x] = [L, A, B]


rgb_image = cv2.cvtColor(image, cv2.COLOR_LAB2RGB)
plt.axis('off')
plt.imshow(rgb_image)
```