Trường ĐHSPKT TP.HCM
Khoa Công nghệ Thông tin
Bộ môn Tin học cơ sở

ĐÁP ÁN MÔN: CHUYÊN ĐỀ 1 – XỬ LÝ ẢNH SỐ
MÃ MÔN HỌC: 1279230
Học kỳ I – Năm học 2015-2016

Câu 1: (4 điểm)
Trình bày các bước và cho ví dụ xây dựng một hệ mờ thông dụng trong xử lý ảnh.
Trả lời:

We can use this figure and the preceding material to summarize the principal steps followed in the application of rule-based fuzzy logic:

1. *Fuzzify the inputs:* For each scalar input, find the corresponding fuzzy values by mapping that input to the interval [0, 1], using the applicable membership functions in each rule, as the first two columns of Fig. 3.52 show.
2. *Perform any required fuzzy logical operations:* The outputs of all parts of an antecedent must be combined to yield a *single* value using the max or min operation, depending on whether the parts are connected by ORs or by ANDs. In Fig. 3.52, all the parts of the antecedents are connected by

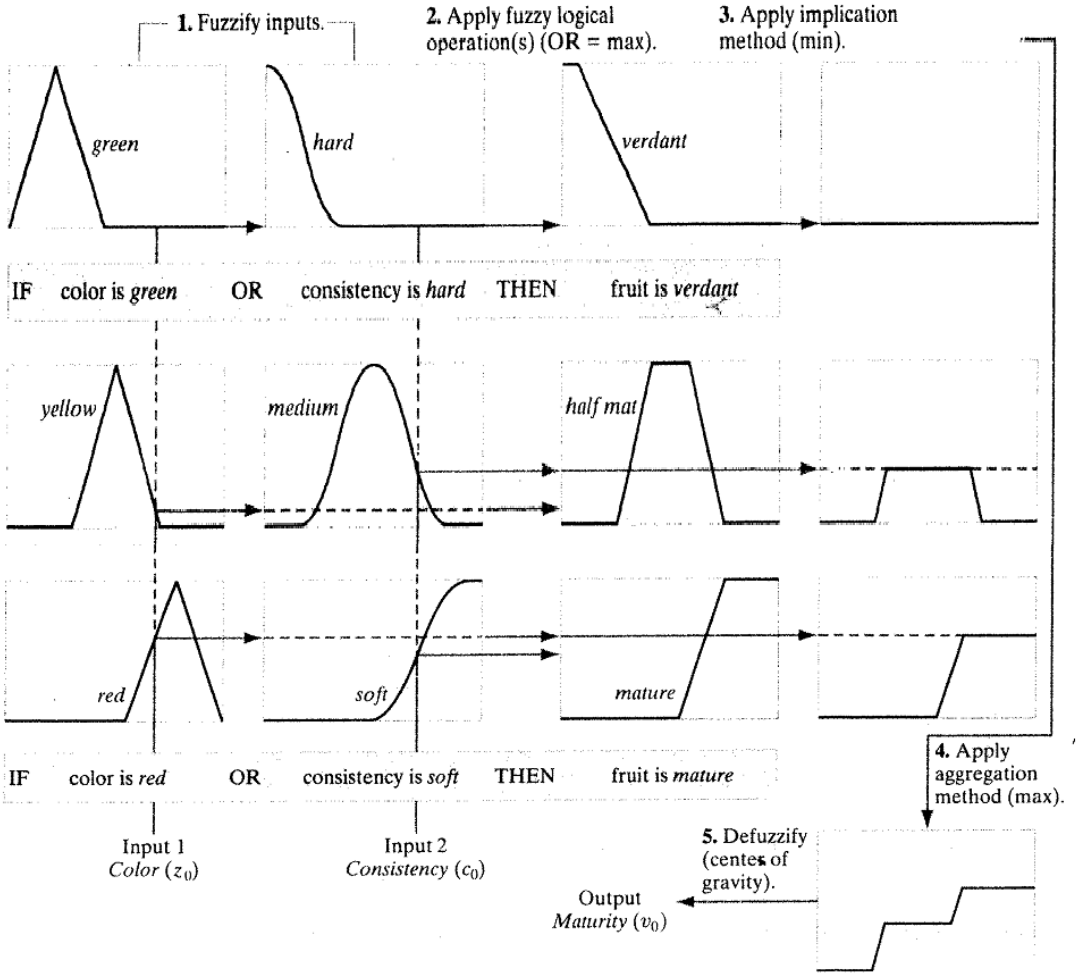


FIGURE 3.52 Example illustrating the five basic steps used typically to implement a fuzzy, rule-based system: (1) fuzzification, (2) logical operations (only OR was used in this example), (3) implication, (4) aggregation, and (5) defuzzification.

ORs, so the max operation is used throughout. The number of parts of an antecedent and the type of logic operator used to connect them can be different from rule to rule.

3. *Apply an implication method:* The single output of the antecedent of each rule is used to provide the output corresponding to that rule. We use AND for implication, which is defined as the min operation. This clips the corresponding output membership function at the value provided by the antecedent, as the third and fourth columns in Fig. 3.52 show.
4. *Apply an aggregation method to the fuzzy sets from step 3:* As the last column in Fig. 3.52 shows, the output of each rule is a fuzzy set. These must be combined to yield a single output fuzzy set. The approach used here is to OR the individual outputs, so the max operation is employed.
5. *Defuzzify the final output fuzzy set:* In this final step, we obtain a crisp, scalar output. This is achieved by computing the center of gravity of the aggregated fuzzy set from step 4.

In general, when dealing with M IF-THEN rules, N input variables, $z_1, z_2, \ldots z_N$, and one output variable, v, the type of fuzzy rule formulation used most frequently in image processing has the form

$$\begin{aligned}
&\text{IF } (z_1, A_{11}) \text{ AND } (z_2, A_{12}) \text{ AND} \ldots \text{AND } (z_N, A_{1N}) \text{ THEN } (v, B_1)\\
&\text{IF } (z_1, A_{21}) \text{ AND } (z_2, A_{22}) \text{ AND} \ldots \text{AND } (z_N, A_{2N}) \text{ THEN } (v, B_2)\\
&\ldots\ldots\\
&\text{IF } (z_1, A_{M1}) \text{ AND } (z_2, A_{M2}) \text{ AND} \ldots \text{AND } (z_N, A_{MN}) \text{ THEN } (v, B_M)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } (v, B_E)
\end{aligned} \tag{3.8-19}$$

where A_{ij} is the fuzzy set associated with the ith rule and the jth input variable, B_i is the fuzzy set associated with the output of the ith rule, and we have assumed that the components of the rule antecedents are linked by ANDs. Note that we have introduced an ELSE rule, with associated fuzzy set B_E. This rule is executed when none of the preceding rules is completely satisfied; its output is explained below.

As indicated earlier, all the elements of the antecedent of each rule must be evaluated to yield a single scalar value. In Fig. 3.52, we used the max operation because the rules were based on fuzzy ORs. The formulation in Eq. (3.8-19) uses ANDs, so we have to use the min operator. Evaluating the antecedents of the ith rule in Eq. (3.8-19) produces a scalar output, λ_i, given by

$$\lambda_i = \min\left\{\mu_{A_{ij}}(z_j); \quad j = 1, 2, \ldots, M\right\} \tag{3.8-20}$$

for $i = 1, 2, \ldots, M$, where $\mu_{A_{ij}}(z_j)$ is the membership function of fuzzy set A_{ij} evaluated at the value of the jth input. Often, λ_i is called the *strength level* (or *firing level*) of the ith rule. With reference to the preceding discussion, λ_i is simply the value used to clip the output function of the ith rule.

The ELSE rule is executed when the conditions of the THEN rules are weakly satisfied (we give a detailed example of how ELSE rules are used in Section 3.8.5). Its response should be strong when all the others are weak. In a sense, one can view an ELSE rule as performing a NOT operation on the results of the other rules. We know from Section 3.8.2 that $\mu_{\text{NOT}(A)} = \mu_{\bar{A}}(z) = 1 - \mu_A(z)$. Then, using this idea in combining (ANDing) all the levels of the THEN rules gives the following strength level for the ELSE rule:

$$\lambda_E = \min\{1 - \lambda_i; \quad i = 1, 2, \ldots, M\} \tag{3.8-21}$$

We see that if all the THEN rules fire at "full strength" (all their responses are 1), then the response of the ELSE rule is 0, as expected. As the responses of the THEN rules weaken, the strength of the ELSE rule increases. This is the fuzzy counterpart of the familiar IF-THEN-ELSE rules used in software programming.

Câu 2: (4 điểm)
Xây dựng và cài đặt một hệ mờ làm tăng độ tương phản của ảnh dùng OpenCV.
Chứng minh công thức:

$$v_0 = \frac{\mu_{dark}(z_0) \times v_d + \mu_{gray}(z_0) \times v_g + \mu_{bright}(z_0) \times v_b}{\mu_{dark}(z_0) + \mu_{gray}(z_0) + \mu_{bright}(z_0)}$$

Trả lời:

Consider the general problem of contrast enhancement, one of the principal applications of intensity transformations. We can state the process of enhancing the contrast of a gray-scale image using the following rules:

IF a pixel is dark, THEN make it *darker*.

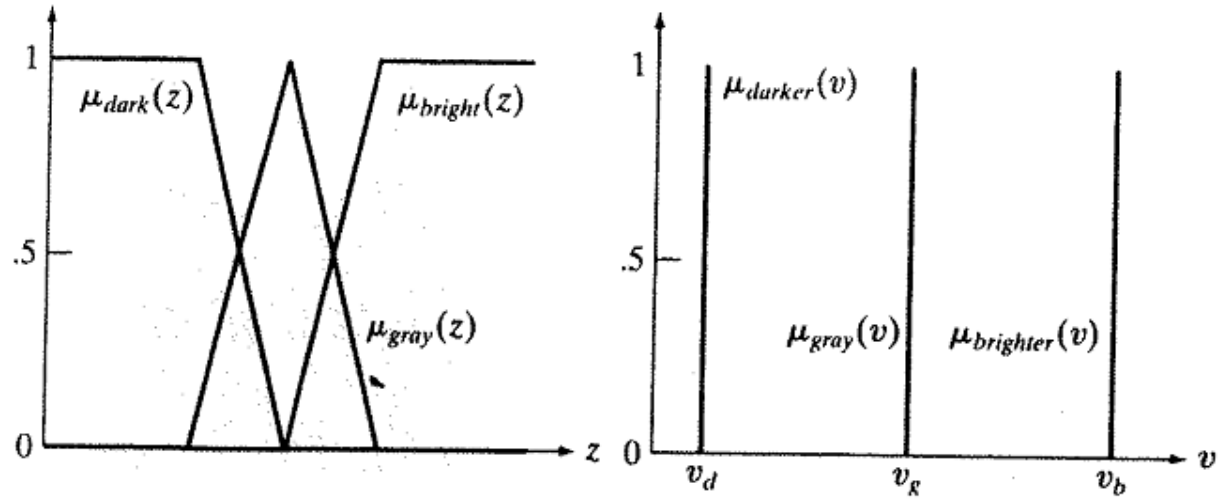
IF a pixel is gray, THEN make it gray.

IF a pixel is bright, THEN make it brighter.

Keeping in mind that these are fuzzy terms, we can express the concepts of *dark*, *gray*, and *bright* by the membership functions in Fig. 3.53(a).

In terms of the output, we can consider *darker* as being degrees of a dark intensity value (100% black being the limiting shade of dark), *brighter*, as being degrees of a bright shade (100% white being the limiting value), and *gray* as being degrees of an intensity in the middle of the gray scale. What we mean by

"degrees" here is the amount of one specific intensity. For example, 80% black is a very dark gray. When interpreted as *constant* intensities whose strength is modified, the output membership functions are *singletons* (membership functions that are *constant*), as Fig. 3.53(b) shows. The various degrees of an intensity in the range [0, 1] occur when the singletons are clipped by the strength of the response from their corresponding rules, as in the fourth column of Fig. 3.52 (but keep in mind that we are working here with only one input, not two, as in the figure). Because we are dealing with constants in the output membership functions, it follows from Eq. (3.8-18) that the output, v_0, to any input, z_0, is given by



a b

FIGURE 3.53
(a) Input and (b) output membership functions for fuzzy, rule-based contrast enhancement.

$$v_0 = \frac{\mu_{dark}(z_0) \times v_d + \mu_{gray}(z_0) \times v_g + \mu_{bright}(z_0) \times v_b}{\mu_{dark}(z_0) + \mu_{gray}(z_0) + \mu_{bright}(z_0)} \quad (3.8\text{-}22)$$

The summations in the numerator and denominator in this expressions are simpler than in Eq. (3.8-18) because the output membership functions are constants modified (clipped) by the fuzzified values.

Fuzzy image processing is computationally intensive because the entire process of fuzzification, processing the antecedents of all rules, implication, aggregation, and defuzzification must be applied to *every* pixel in the input image. Thus, using singletons as in Eq. (3.8-22) significantly reduces computational requirements by simplifying implication, aggregation, and defuzzification. These savings can be significant in applications where processing speed is an important requirement.

Công thức (3.8.22) dễ dàng chứng minh bằng đồ thị.

Code:

```
void FuzzyIntensity(Mat imgin, Mat imgout)
{
	double MuyDark[L], MuyGray[L], MuyBright[L];
	int z;
	double a, b, c, d;
	a = 0;	c = 0;	b = 75;	d = 52;
	for (z = 0; z < L; z++)
	if (a - c <= z && z < a)
		MuyDark[z] = 1 - (a - z) / c;
	else if (a <= z && z < b)
		MuyDark[z] = 1;
	else if (b <= z && z <= b + d)
		MuyDark[z] = 1 - (z - b) / d;
	else
		MuyDark[z] = 0;

	a = 127; b = 52; c = 52;
	for (z = 0; z < L; z++)
	if (a - b <= z && z < a)
		MuyGray[z] = 1 - (a - z) / b;
	else if (a <= z && z <= a + c)
		MuyGray[z] = 1 - (z - a) / c;

	a = 179; c = 52; b = 255; d = 0;
	for (z = 0; z < L; z++)
	if (a - c <= z && z < a)
		MuyBright[z] = 1 - (a - z) / c;
	else if (a <= z && z < b)
		MuyBright[z] = 1;
```

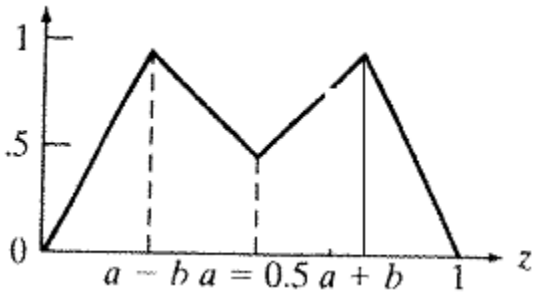
```
        else if (b <= z && z <= b + d)
            MuyBright[z] = 1 - (z - b) / d;
        else
            MuyBright[z] = 0;

        int vd = 15, vg = 127, vb = 240;

        int M = imgin.size().height;
        int N = imgin.size().width;
        int x, y;
        uchar z0;
        double v0;
        for (x = 0; x < M; x++)
        for (y = 0; y < N; y++)
        {
            z0 = imgin.at<uchar>(x, y);
            v0 = (MuyDark[z0] * vd + MuyGray[z0] * vg +
MuyBright[z0] * vb) / (MuyDark[z0] + MuyGray[z0] +
MuyBright[z0]);
            imgout.at<uchar>(x, y) = (uchar)v0;
        }
        return;
}
```

Câu 3: (2 điểm)

Hãy viết phương trình hàm thuộc của hình vẽ sau:



Trả lời:

Dựa vào phương trình đường thẳng đi qua 2 điểm (x_1, y_1) và (x_2, y_2) là:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

Ta lần lượt viết phương trình của các đoạn thẳng trong hình trên.

----------HẾT----------

TP. HCM, ngày 27/12/2015

Bộ môn duyệt