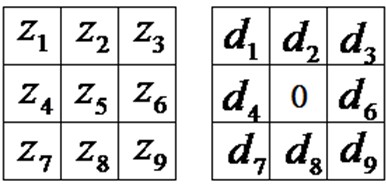
**使用模糊集合进行图像的边缘检测**

**班级：**控制工程9班

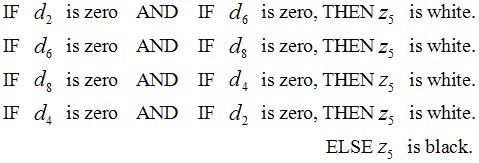
**姓名：**郑明航

**学号:**2018G0507026

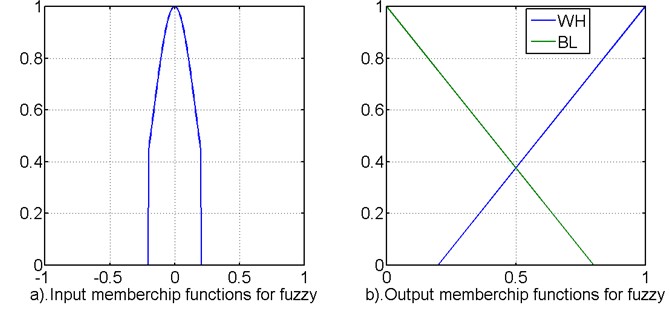
所谓边缘是指其周围像素灰度急剧变化的那些象素的集合，它是图像最基本的特征。若想进行一幅图像的边缘检测，在空间域上的想法是：“如果一个像素是处在平滑区域的，那么使得这个像素为亮，否则，则使得这个像素为暗。”本设计使用灰度差来表示像素的平滑程度。



 如上图所示，我们将一个像素8个相邻的像素，各减去这个像素的值，即https://img-blog.csdn.net/20131029123241937。我们可以得到如上图右侧的结果。然后，可以推出以下规则：



上述的zero，white与black都是模糊的概念。同样的，根据这三个规则模糊化之后，用重心法去模糊。同时，我们还希望，当两个点的灰度很接近的时候，去模糊能给予一个很强的响应，将灰度拉至很高(很亮)。所以，我们的输入隶属度函数需要再0处有一个较大的隶属度，这里，使用了高斯分布的一部分。其输出隶属度函数，也不是像上面的例子一样简单，我们希望若是白色的隶属度很高，其输出的灰度值就越高(越亮)，反之则越低(越暗)。根据上述，我指定的输入输出隶属度函数如下所示：



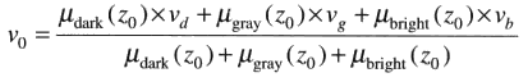
在用上述规则处理时， 使用AND进行集合运算，是求集合的最小部分，公式如下：

https://img-blog.csdn.net/20131029125447234

这里ELSE的语句，用如下公式 ：

https://img-blog.csdn.net/20131029125730125

 最后可以用中心法去模糊。



参考：《数字图像处理》冈萨雷斯 和 Nie and Barner[2006]

定义规则的IF函数：

Fuzzy\_Knowledge\_Filters.m

function [W1,W2,W3,W4,B] = Fuzzy\_Knowledge\_Filters(Intensity)

%W1,W2,W3,W4,B是输入dm的隶属度。

%Intensity是掩模差值矩阵。

for x = 1:1:3

for y = 1:1:3

if((Intensity(x,y) <= 0.2) &&(Intensity(x,y) >= -0.2))

Intensity(x,y) = exp(-20\*Intensity(x,y).\*Intensity(x,y));

else Intensity(x,y) = 0;

end

end

end

W1 = min(Intensity(1,2),Intensity(2,3));

W2 = min(Intensity(2,3),Intensity(3,2));

W3 = min(Intensity(3,2),Intensity(2,1));

W4 = min(Intensity(2,1),Intensity(1,2));

B = min(min(1-W1,1-W2),min(1-W3,1-W4));

end

**主函数：**

figure();

subplot(1,2,1);

x=-0.2:1/255:0.2;

y=exp(-20\*x.\*x);

x = -1:1/255:1;

y = [zeros(1,204) y zeros(1,204)];

plot(x,y);

axis([-1,1,0,1]),grid;

axis square;

xlabel('a).Input memberchip functions for fuzzy');

subplot(1,2,2);

x1 = [0,0.2,1];y1 = [0,0,1];

x2 = [0,0.8,1];y2 = [1,0,0];

plot(x1,y1,x2,y2);

axis([0,1,0,1]),grid;

axis square;

legend('WH','BL');

xlabel('b).Output memberchip functions for fuzzy');

f = imread('3.tif');

f = mat2gray(f,[0 255]);%图像归一化

[M,N]=size(f);

f\_Ex = zeros(M+2,N+2);%给图像加上像素为1的边框

for x = 1:1:M

for y = 1:1:N

f\_Ex(x+1,y+1) = f(x,y);

end

end

z = zeros (3,3);

g = zeros (M+2,N+2);

for x = 2:1:M+1

for y = 2:1:N+1 %掩模差值dm计算

z(1,1) = f\_Ex(x-1,y-1) - f\_Ex(x,y);

z(1,2) = f\_Ex(x-1,y) - f\_Ex(x,y);

z(1,3) = f\_Ex(x-1,y+1) - f\_Ex(x,y);

z(2,1) = f\_Ex(x,y-1) - f\_Ex(x,y);

z(2,2) = f\_Ex(x,y) - f\_Ex(x,y);

z(2,3) = f\_Ex(x,y+1) - f\_Ex(x,y);

z(3,1) = f\_Ex(x+1,y-1) - f\_Ex(x,y);

z(3,2) = f\_Ex(x+1,y) - f\_Ex(x,y);

z(3,3) = f\_Ex(x+1,y+1) - f\_Ex(x,y);

[W1,W2,W3,W4,B] = Fuzzy\_Knowledge\_Filters(z);

%通过输出隶属度函数计算像素点的灰度值

V1 = 0.8 \* W1 + 0.2;

V2 = 0.8 \* W2 + 0.2;

V3 = 0.8 \* W3 + 0.2;

V4 = 0.8 \* W4 + 0.2;

V5 = 0.8 - (0.8 \* B);

%重心法去模糊

g(x,y) = ((W1\*V1) + (W2\*V2) + (W3\*V3) + (W4\*V4) + (B\*V5))/(W1+W2+W3+W4+B);

end

end

figure();

subplot(1,2,1);

imshow(f,[0 1]);

xlabel('a).Original Image');

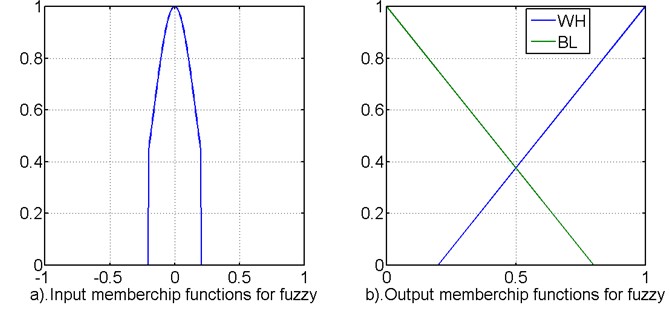
subplot(1,2,2);

imshow(g,[0 1]);

xlabel('b).Result of fuzzy');

**运行结果：**

模糊输入函数和模糊输出函数：



输入图片与处理后的图片：

