Assignment 4

Part 1 code

参见 fuzzy.cpp。

Part 2 explanation

(1)fdm formula

这里主要参考的公式就是文章中提到的以下两条式子：

$$I(A) = \frac{2}{n}[\sum_{i=1}^{n}|\mu_A(x_i) - \mu_{\tilde{A}}(x_i)|]$$

Such a function is defined as

$$\mu_S(x) = S(x,a,b,c) = \begin{cases} 0, x < a \\ 2[(x-a)/(c-a)]^2, a < x \le b \\ 1 - 2[(x-c)/(c-a)]^2, b < x < c \\ 1, x > c \end{cases}$$

这里是具体实现的代码：

```
double fdmhelper(int a, int c,int x)
{
    double b = (a + c) / 2;

    if (x <= a)return 0;
    else if (x > a && x <= b)
    {
        double temp = (x - a) / (double)(c - a);
        return 2 * temp*temp;
    }
    else if (x > b && x < c)
    {
        double temp = (x - c) / (double)(c - a);
        return 1 - 2 * temp*temp;
    }
    else return 1;
}

double fdm(int a, int c)
{
    int n = MAX_GRAY_VALUE;
    double meanU = fdmhelper(a, c, (a+c)/2);

    double sum = 0;
    for (int i = 0; i < MAX_GRAY_VALUE; i++)
    {
        double ui = fdmhelper(a, c, i);
        sum += abs(ui - meanU);
    }

    return 2 * sum / (double)n;
}
```

这个 fdm 函数算的实际上就是 fuzzy densitity，用于后面阈值的划分。具体来说就是在 Xleft 和 Xright 之间取一个点 Xi，用 fdm 来判断 Xi 更接近于左边还是右边。

(2)constant variable

这是代码中设置的常量，MAX_GRAY_VALUE 表示最大灰度值，P1 和 P2 是设置的百分比。filename 和 resultname 分别是输入和输出图像。最后发现这里的 P1 和 P2 取值对结果的影响很大，因为他直接决定了 factor、xleft 和 xright，感觉可以对图片的灰度分布进行分析之后自动的设置 P1 和 P2 的取值，可能会比人工设定常数效果好。

```
const int MAX_GRAY_VALUE = 256;

const double P1 = 0.3964;
const double P2 = 0.2;

const string filename = "fuzzy.png";
const string resultname = "fuzzyresult.png";
```

(3)calculate threshold

这是 P1 和 P2 具体的应用：根据 0-128 和 128-256 的灰度频数与 P2*M*N 作比较，来判断是否需要做直方图均衡化，是灰度分布更平均一些。

```
//if we need to do histogram equalization
int pixelB = 0;
int pixelW = 0;
for (int i = 0; i < 128; i++)
{
    pixelB += hist[i];
}
pixelW = img->height*img->width - pixelB;
double Pmin = P2 * img->height * img->width;
if (pixelB < Pmin || pixelW < Pmin)
{
    histogramEqualization(img);//modify img->imageData by HE
}
```

然后计算出 minPixelB 和 minPixelW，用来确定 xleft 和 xright：

```
//recalculate because data has been changed
pixelB = 0;
pixelW = 0;
for (int i = 0; i < 128; i++)
{
    pixelB += hist[i];
    pixelW += hist[i + 128];
}
int minPixelB = P1*pixelB;
int minPixelW = P1*pixelW;

//calculate xl and xr
int xl = 0;
int xr = 128;
int tempSum = 0;
for (; xl < 128; xl++)
{
    tempSum += hist[xl];
    if (tempSum >= minPixelB)
    {
        break;
    }
}
for (tempSum = 0; xr < MAX_GRAY_VALUE; xr++)
{
    tempSum += hist[xr];
    if (tempSum >= minPixelW)
    {
        break;
    }
}
```

然后计算左/右的比例因子，我看原文的意思好像是直接算 0 到 xleft 的 fdm1 和 xright 到 256 的 fdm2，然后 factor = fdm1 / fdm2。但是这样计算的话，factor 很容易偏大或者偏小，导致后面在取阈值的时候，阈值偏大或者偏小，产生的结果不尽人意。

这里我取得是 xmin 到 xleft 和 xright 到 xmax 的 fdm 值，xmin 是整幅图出现过的最小灰度值，xmax 是整幅图出现过的最大灰度值，这样产生的结果比较合理。

这是 factor 的计算过程：

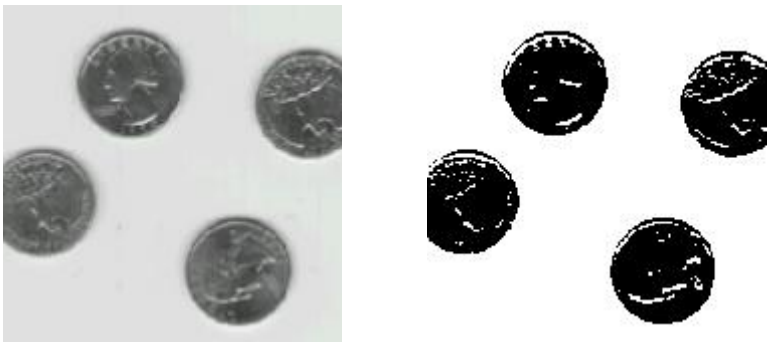
```
//calculate factor
int xmin, xmax;
for (int i = 0; i < MAX_GRAY_VALUE; i++)
{
    if (hist[i]>0)
    {
        xmin = i;
        break;
    }
}
for (int i = MAX_GRAY_VALUE - 1; i >= 0; i--)
{
    if (hist[i] > 0)
    {
        xmax = i;
        break;
    }
}
double factor = fdm(xr, xmax) / fdm(xmin, xl);
```

这是计算 threshold 的过程，从 xleft 到 xright 遍历：

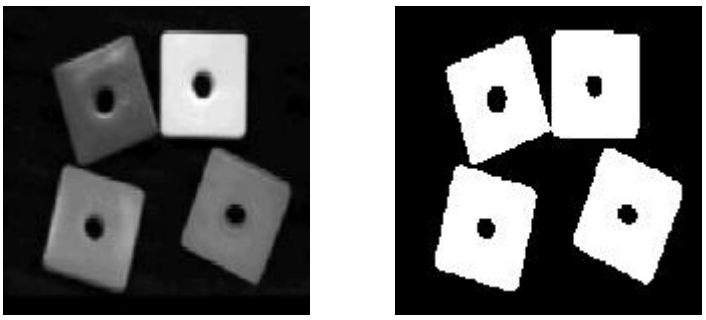
```
//calculate the threshold
int threshold;
for (int i = xl + 1; i < xr; i++)
{
    double left = fdm(xl, i);
    double right = fdm(i, xr);
    if (left * factor > right)
    {
        continue;
    }
    else
    {
        threshold = i;
        break;
    }
}
cout << "Threshold is: " << threshold << endl;
```

Part 3 output

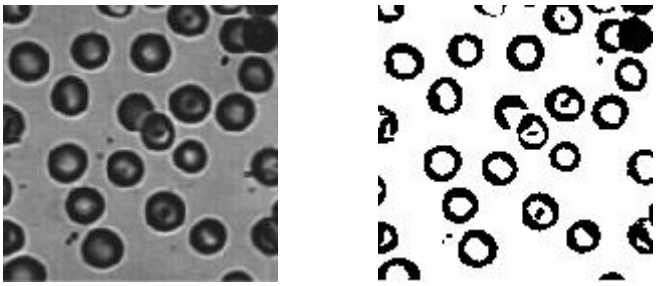
以下是对三幅图进行基于 fuzzy set 的阈值计算的结果：



（1）



（2）



（3）