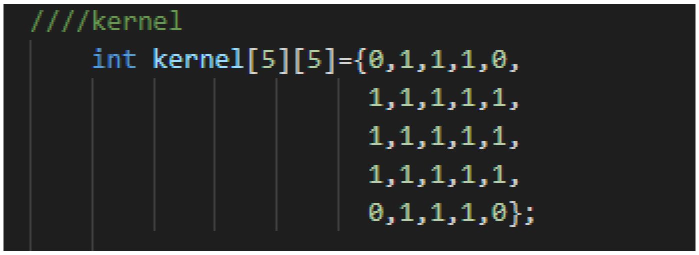
Computer Vision HW5

R10922A16 蔡家豪

Homework 5

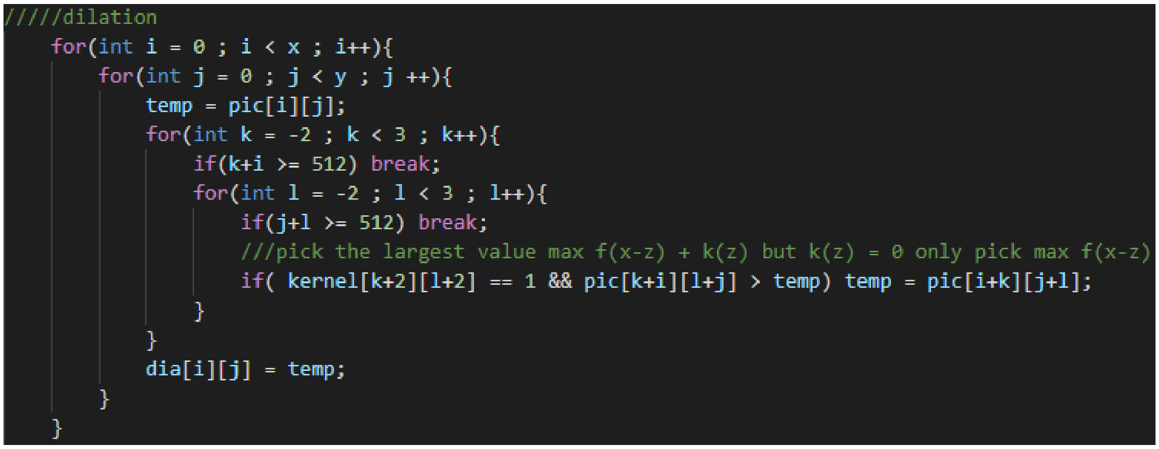
Mathematical Morphology - Gray Scaled Morphology

Kernel I use same as HW4
but actually value is 0(1 in kernel)
just because real value is 0
you can just skip +k(z) or -k(z) part and you will have the same effect as the real kernel



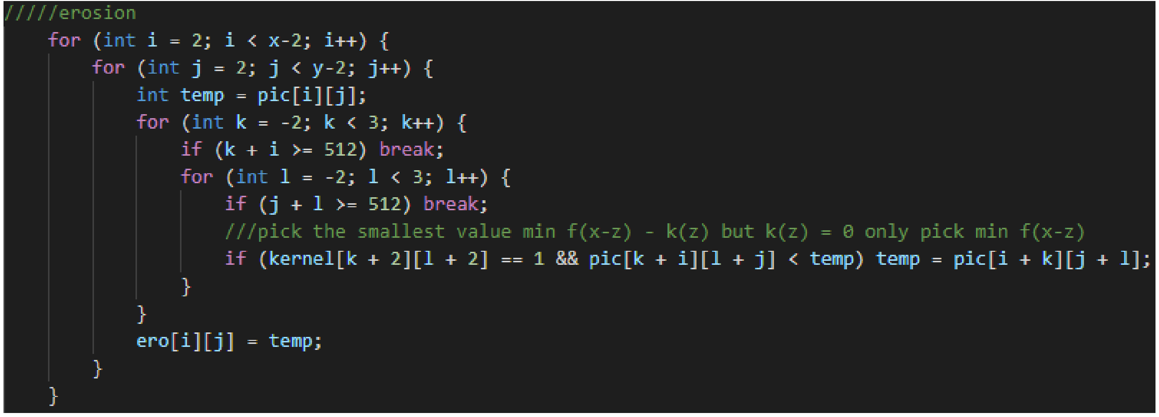
```
////kernel
    int kernel[5][5]={0,1,1,1,0,
                      1,1,1,1,1,
                      1,1,1,1,1,
                      1,1,1,1,1,
                      0,1,1,1,0};
```

(a) Dilation



```
/////dilation
    for(int i = 0 ; i < x ; i++){
        for(int j = 0 ; j < y ; j ++){
            temp = pic[i][j];
            for(int k = -2 ; k < 3 ; k++){
                if(k+i >= 512) break;
                for(int l = -2 ; l < 3 ; l++){
                    if(j+l >= 512) break;
                    ///pick the largest value max f(x-z) + k(z) but k(z) = 0 only pick max f(x-z)
                    if( kernel[k+2][l+2] == 1 && pic[k+i][l+j] > temp) temp = pic[i+k][j+l];
                }
            }
            dia[i][j] = temp;
        }
    }
```

(b) Erosion



```
/////erosion
    for (int i = 2; i < x-2; i++) {
        for (int j = 2; j < y-2; j++) {
            int temp = pic[i][j];
            for (int k = -2; k < 3; k++) {
                if (k + i >= 512) break;
                for (int l = -2; l < 3; l++) {
                    if (j + l >= 512) break;
                    ///pick the smallest value min f(x-z) - k(z) but k(z) = 0 only pick min f(x-z)
                    if (kernel[k + 2][l + 2] == 1 && pic[k + i][l + j] < temp) temp = pic[i + k][j + l];
                }
            }
            ero[i][j] = temp;
        }
    }
```

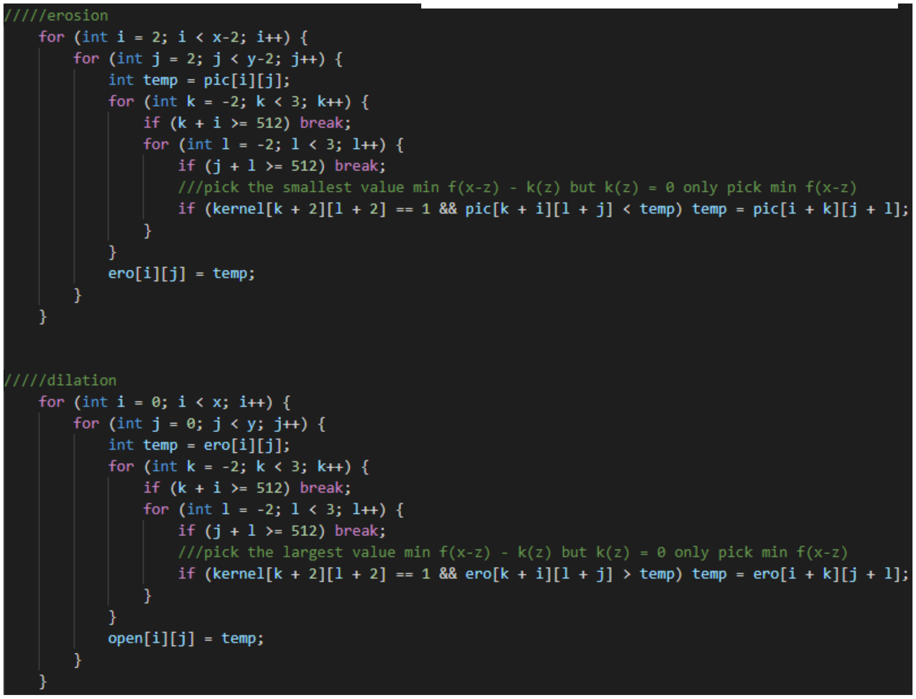
(c) Opening

step:

1.erosion

2.dilation

$$B \circ K = (B \ominus K) \oplus K$$



```
/////erosion
    for (int i = 2; i < x-2; i++) {
        for (int j = 2; j < y-2; j++) {
            int temp = pic[i][j];
            for (int k = -2; k < 3; k++) {
                if (k + i >= 512) break;
                for (int l = -2; l < 3; l++) {
                    if (j + l >= 512) break;
                    ///pick the smallest value min f(x-z) - k(z) but k(z) = 0 only pick min f(x-z)
                    if (kernel[k + 2][l + 2] == 1 && pic[k + i][l + j] < temp) temp = pic[i + k][j + l];
                }
            }
            ero[i][j] = temp;
        }
    }


/////dilation
    for (int i = 0; i < x; i++) {
        for (int j = 0; j < y; j++) {
            int temp = ero[i][j];
            for (int k = -2; k < 3; k++) {
                if (k + i >= 512) break;
                for (int l = -2; l < 3; l++) {
                    if (j + l >= 512) break;
                    ///pick the largest value min f(x-z) - k(z) but k(z) = 0 only pick min f(x-z)
                    if (kernel[k + 2][l + 2] == 1 && ero[k + i][l + j] > temp) temp = ero[i + k][j + l];
                }
            }
            open[i][j] = temp;
        }
    }
```

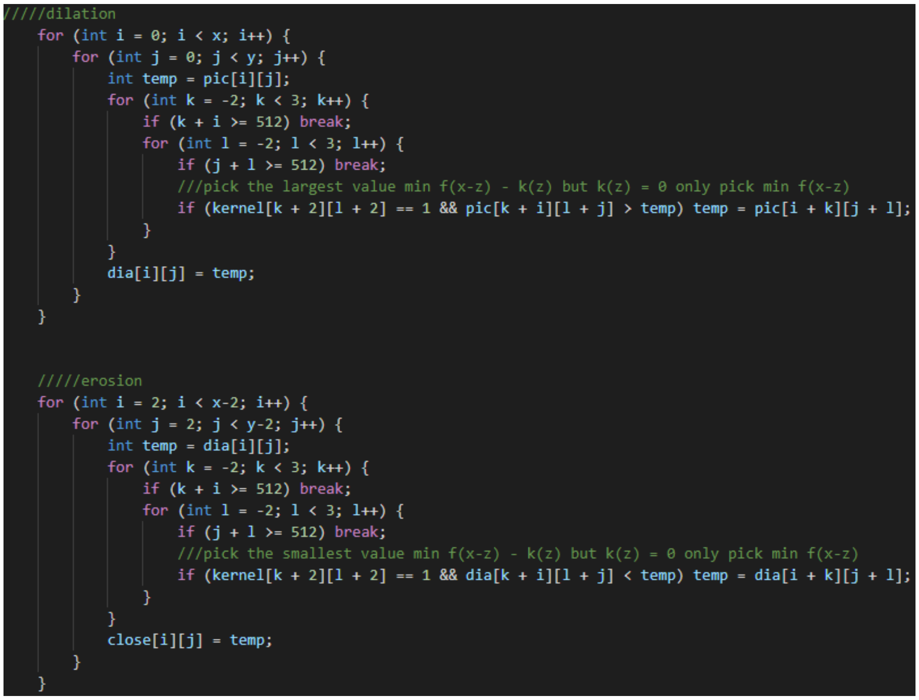
(d) Closing

step:

1.dilation

2.erosion

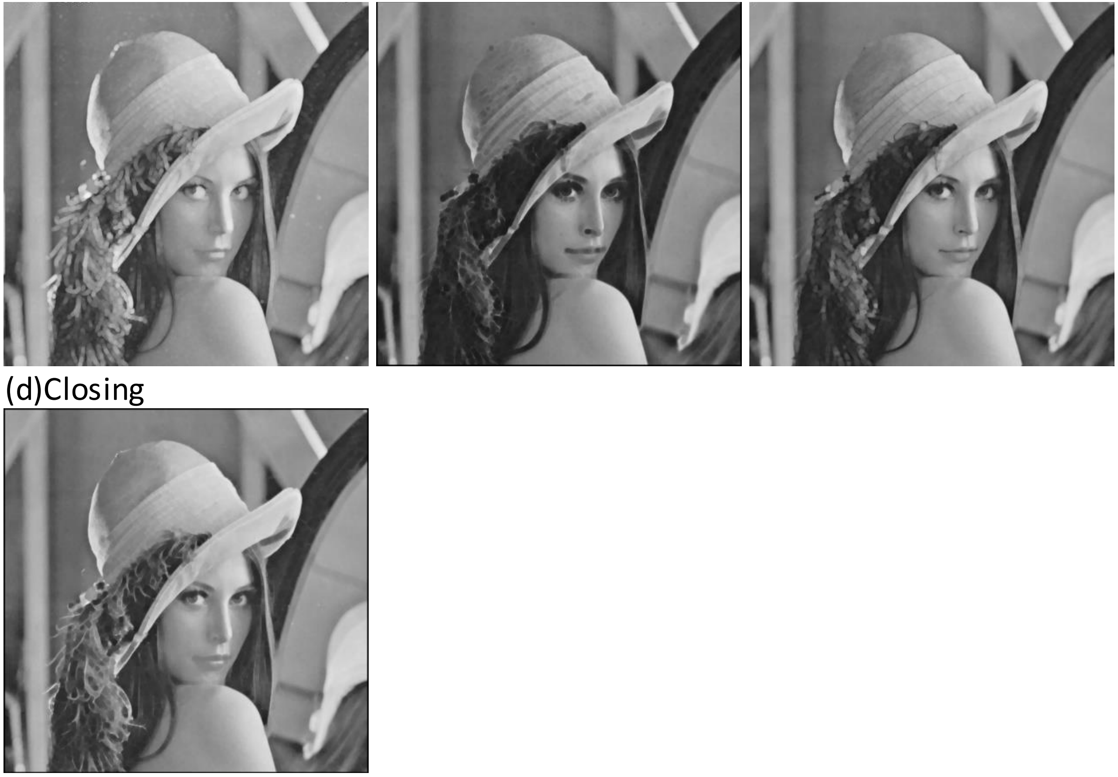
$$B \bullet K = (B \oplus K) \ominus K$$



```
/////dilation
    for (int i = 0; i < x; i++) {
        for (int j = 0; j < y; j++) {
            int temp = pic[i][j];
            for (int k = -2; k < 3; k++) {
                if (k + i >= 512) break;
                for (int l = -2; l < 3; l++) {
                    if (j + l >= 512) break;
                    ///pick the largest value min f(x-z) - k(z) but k(z) = 0 only pick min f(x-z)
                    if (kernel[k + 2][l + 2] == 1 && pic[k + i][l + j] > temp) temp = pic[i + k][j + l];
                }
            }
            dia[i][j] = temp;
        }
    }


    /////erosion
    for (int i = 2; i < x-2; i++) {
        for (int j = 2; j < y-2; j++) {
            int temp = dia[i][j];
            for (int k = -2; k < 3; k++) {
                if (k + i >= 512) break;
                for (int l = -2; l < 3; l++) {
                    if (j + l >= 512) break;
                    ///pick the smallest value min f(x-z) - k(z) but k(z) = 0 only pick min f(x-z)
                    if (kernel[k + 2][l + 2] == 1 && dia[k + i][l + j] < temp) temp = dia[i + k][j + l];
                }
            }
            close[i][j] = temp;
        }
    }
```

result image:



(a)Dilation

(b)Erosion

(c)Opening

(d)Closing