# Computer and Robot Vision

## Homework#5

R01944040 柳成蔭

這次的作業是對 gray scale image 進行 morphological operation，得到原 lena 圖 dilation, erosion, opening, closing 後的結果。

我使用 VS2012 編寫程式

先設置 morphology 使用的 kernel，dilation 和 erosion 使用 octogonal 3-5-5-5-3 的 kernel。

```
struct Kernel
{
    int kCols;
    int kRows;
    int anchorX;
    int anchorY;
    Mat values;
    Kernel(int cols, int rows, int ancx, int ancy, Mat val)
        :kCols(cols), kRows(rows), anchorX(ancx),
anchorY(ancy), values(val.clone())
    {
    }
};

    uchar kValArr[]={0,  255,255,255,0,
                   255,255,255,255,255,
                   255,255,255,255,255,
                   255,255,255,255,255,
                   0,  255,255,255,0  };
    Mat kVal=Mat(5,5,CV_8U,kValArr).clone();
    Kernel ker(5, 5, 2, 2, kVal);
```

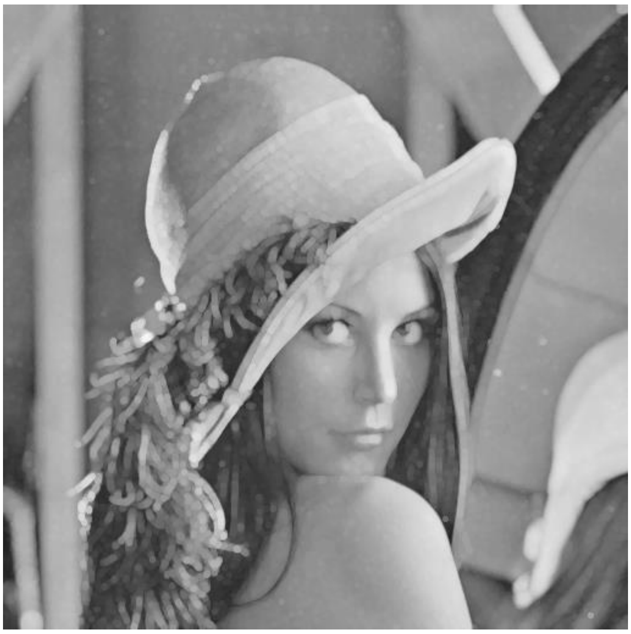
(a) Dilation

$$(f \oplus k)(x) = \max\{f(x-z)+k(z)|z \in K, x-z \in F\}$$

對原圖的每個 pixel，在新圖上進行 grayPixelDil。將新圖上原 pixel 位置的值設為原圖上 kernel 覆蓋的鏡像區域中最大的值。

```
void grayPixelDil( const Mat src, Mat dst, const Kernel ker,
int sI, int sJ )
{
    uchar grayMax=src.at<uchar>(sI,sJ);
    for (int kI = 0; kI < ker.kRows; kI++)
    {
        for (int kJ = 0; kJ < ker.kCols; kJ++)
        {
            int sX=sI-(kI-ker.anchorX);
            int sY=sJ-(kJ-ker.anchorY);
            if (sX>=0 && sX<=src.rows-1 &&
                sY>=0 && sY<=src.cols-1)
            {
                if (ker.values.at<uchar>(kI,kJ)==255)
                {
                    grayMax=(uchar)max( (int)grayMax,
(int)src.at<uchar>(sX,sY) );
                    dst.at<uchar>(sI,sJ)=grayMax;
                }
            }
        }
    }
}

void grayDilation( const Mat src, Mat dst, const Kernel ker )
{
    if(ker.values.empty())
    {
        printf("error");
        return;
    }

    for (int sI = 0; sI < src.rows; sI++)
    {
        for (int sJ = 0; sJ < src.cols; sJ++)
        {
            //dilation
            grayPixelDil(src, dst, ker, sI, sJ);
        }
    }
}

    Mat imgGrayDilation( img.cols, img.rows, CV_8U,
Scalar(0) );
    grayDilation( img, imgGrayDilation, ker );
```

Dilation 的結果：



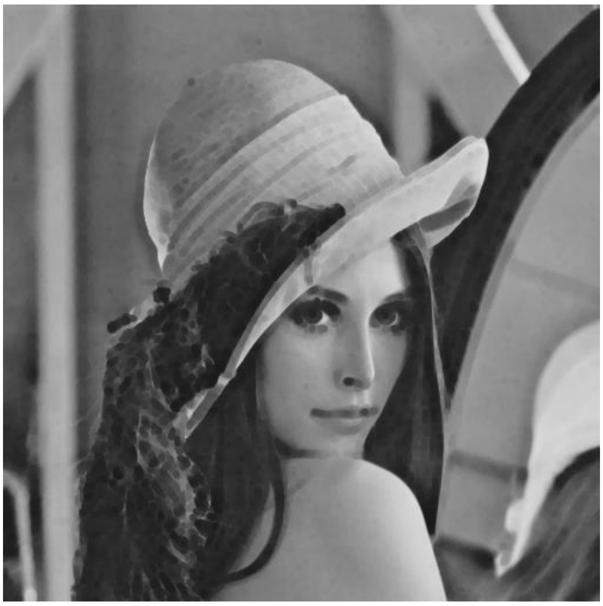
(b) Erosion

$$(f \ominus k)(x) = \min\{f(x+z) - k(z)\}$$

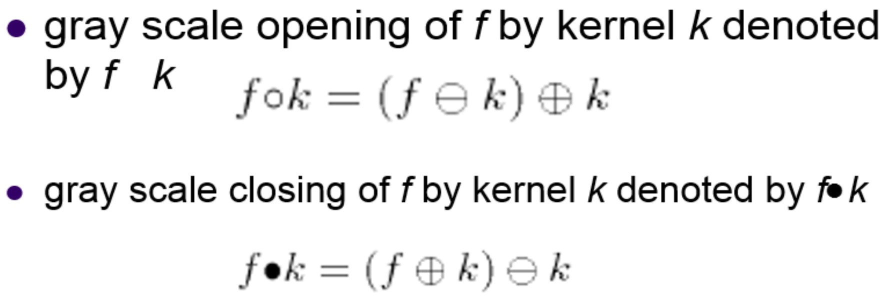
對原圖的每個 pixel 做 grayPixelEro。將新圖上原 pixel 位置的值設為原圖上 kernel 覆蓋區域中最小的值。

```
void grayPixelEro( const Mat src, Mat dst, const Kernel
ker, int sI, int sJ )
{
    uchar grayMin=src.at<uchar>(sI,sJ);
    for (int kI = 0; kI < ker.kRows; kI++)
    {
        for (int kJ = 0; kJ < ker.kCols; kJ++)
        {
            int sX=sI+(kI-ker.anchorX);
            int sY=sJ+(kJ-ker.anchorY);
            if (sX>=0 && sX<=src.rows-1 &&
                sY>=0 && sY<=src.cols-1)
            {
                if (ker.values.at<uchar>(kI,kJ)==255)
                {

    grayMin=(uchar)min((int)grayMin,(int)src.at<uchar>(sX,s
Y));
                    dst.at<uchar>(sI,sJ)=grayMin;
                }
            }
        }
    }
}

void grayErosion( const Mat src, Mat dst, const Kernel
ker )
{
    if(ker.values.empty())
    {
        printf("error");
        return;
    }

    for (int sI = 0; sI < src.rows; sI++)
    {
        for (int sJ = 0; sJ < src.cols; sJ++)
        {
            //erosion
            grayPixelEro(src, dst, ker, sI, sJ);
        }
    }
}

    Mat imgGrayErosion(img.cols, img.rows, CV_8U,
Scalar(0) );
    grayErosion( img, imgGrayErosion, ker );
```

Erosion的結果：



(c) Opening & Closing



- gray scale opening of *f* by kernel *k* denoted by *f*   *k*

$$f \circ k = (f \ominus k) \oplus k$$

- gray scale closing of *f* by kernel *k* denoted by *f*•*k*

$$f \bullet k = (f \oplus k) \ominus k$$

Opening會先對影像進行一次erosion，然後對得到的結果進行一次

dilation。

```
    Mat imgGrayOpening(img.cols, img.rows, CV_8U,
Scalar(0) );
    grayErosion( img, imgGrayErosion, ker );
    grayDilation( imgGrayErosion, imgGrayOpening, ker );
```

Closing則相反，先對影像進行一次dilation，然後對得到的結果進行一次erosion。

```
    Mat imgGrayClosing(img.cols, img.rows, CV_8U,
Scalar(0) );
    grayDilation( img, imgGrayDilation, ker );
    grayErosion( imgGrayDilation, imgGrayClosing, ker );
```

Opening & Closing 的結果：