**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); |

1. Dilation



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：



1. Erosion



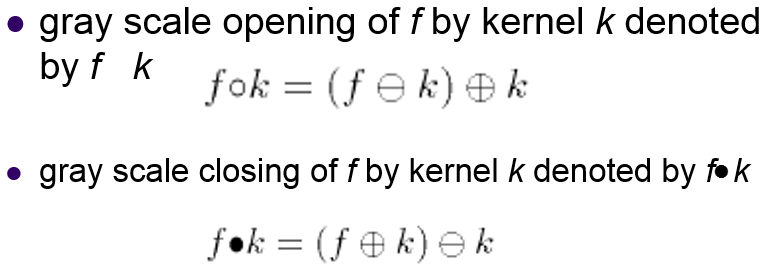
對原圖的每個pixel做grayPixelEro。將新圖上原pixel位置的值設為原圖上kernel覆蓋區域中最小的值。

|  |
| --- |
| 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,sY));  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：



1. Opening & Closing



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：