**Computer and Robot Vision**

**Homework#4**

R01944040 柳成蔭

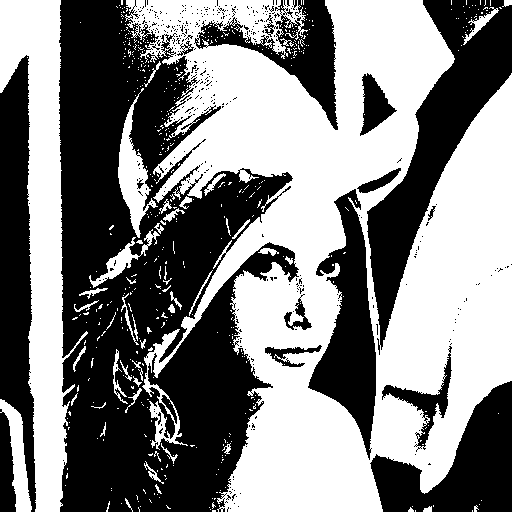
這次的作業是對binary image進行morphological operation，得到原lena圖dilation, erosion, opening, closing, 以及hit-and-miss transform後的結果。

我使用VS2012編寫程式

先得到原圖的binary圖。

|  |
| --- |
| Mat imgBinary;  img.copyTo(imgBinary);  for(int i=0;i<=imgBinary.rows-1;i++)  {  for(int j=0;j<=imgBinary.cols-1;j++)  {  if(imgBinary.at<uchar>(i,j)<=127)  imgBinary.at<uchar>(i,j)=0;  else  imgBinary.at<uchar>(i,j)=255;  }  } |

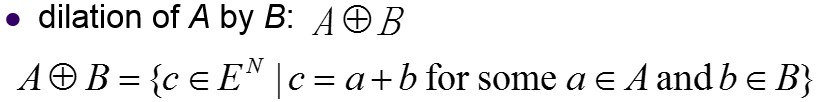
Binary的結果：



另外需要設置morphology使用的kernel，dilation和erosion使用octogonal 3-5-5-5-3的kernel。

|  |
| --- |
| struct Kernel  {  int kCols; //结构元素的行宽  int kRows; //列高  int anchorX; //结构原点位置水平坐标  int anchorY; //结构原点位置垂直坐标  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); |

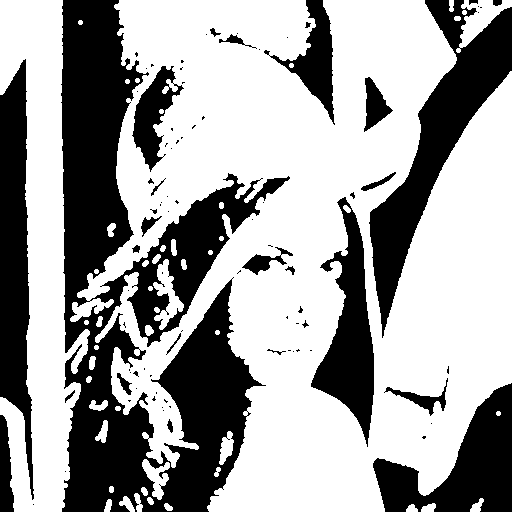
1. Dilation



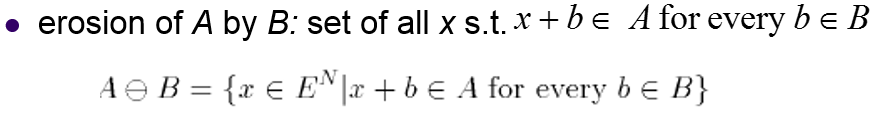
對原圖的每個pixel，如果值為255，就在新圖上進行pixelDil。對kernel上值為255的點，將新圖上對應位置的值設為255。

|  |
| --- |
| void pixelDil( const Mat src, Mat dst, const Kernel ker, int sI, int sJ )  {  for (int kI = 0; kI < ker.kRows; kI++)  {  for (int kJ = 0; kJ < ker.kCols; kJ++)  {  int sX=sI-ker.anchorX+kI;  int sY=sJ-ker.anchorY+kJ;  if (sX>=0 && sX<=src.rows-1 &&  sY>=0 && sY<=src.cols-1)  {  if (ker.values.at<uchar>(kI,kJ)==255)  {  dst.at<uchar>(sX,sY)=255;  }  }  }  }  }  void Dilation( 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++)  {  if (src.at<uchar>(sI,sJ)==255)  {  //dilation  pixelDil(src, dst, ker, sI, sJ);  }  }  }  }  Mat imgDilation( imgBinary.cols, imgBinary.rows, CV\_8U, Scalar(0) );  Dilation( imgBinary, imgDilation, ker ); |

Dilation的結果：



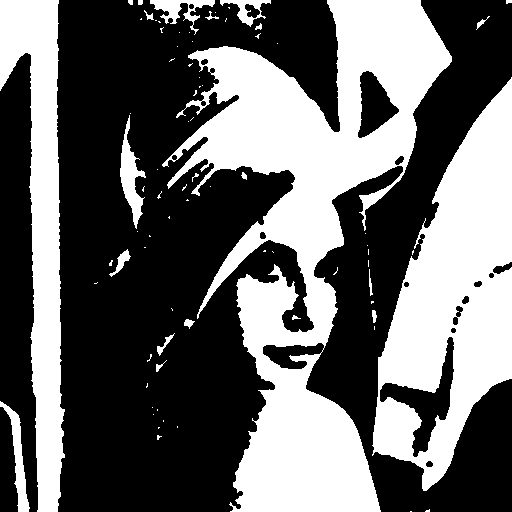
1. Erosion



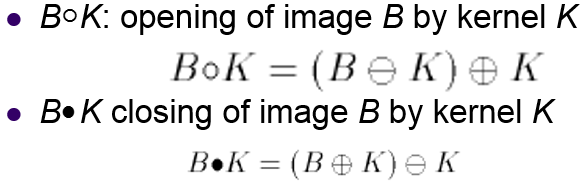
對原圖的每個pixel做pixelEro。對kernel上值為255，但是原圖上值為0的點，將新圖上kernel anchor位置的值設為0。如果所有kernel上值為255的點在原圖上值也為255，則將新圖上kernel anchor位置的值設為255。

|  |
| --- |
| void pixelEro( const Mat src, Mat dst, const Kernel ker, int sI, int sJ )  {  int kA=0;  for (int kI = 0; kI < ker.kRows; kI++)  {  for (int kJ = 0; kJ < ker.kCols; kJ++)  {  int sX=sI-ker.anchorX+kI;  int sY=sJ-ker.anchorY+kJ;  if (sX>=0 && sX<=src.rows-1 &&  sY>=0 && sY<=src.cols-1)  {  if ( ker.values.at<uchar>(kI,kJ)==255 &&  src.at<uchar>(sX,sY)==0)  {  dst.at<uchar>(sI,sJ)=0;  kI = ker.kRows;  kA=-1;  break;  }  }  }  }  if(kA == 0)  dst.at<uchar>(sI,sJ)=255;  }  void Erosion( 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  pixelEro(src, dst, ker, sI, sJ);  }  }  }  Mat imgErosion(imgBinary.cols, imgBinary.rows, CV\_8U, Scalar(0) );  Erosion( imgBinary, imgErosion, ker ); |

Erosion的結果：



1. Opening & Closing



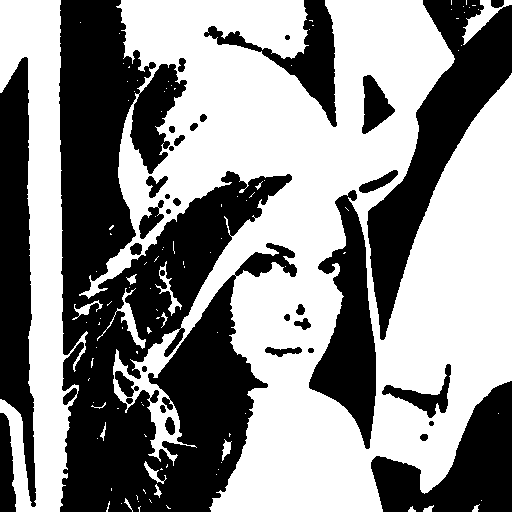
Opening會先對影像進行一次erosion，然後對得到的結果進行一次dilation。

|  |
| --- |
| Mat imgOpening(imgBinary.cols, imgBinary.rows, CV\_8U, Scalar(0) );  Erosion( imgBinary, imgErosion, ker );  Dilation( imgErosion, imgOpening, ker ); |

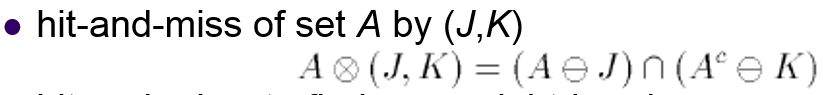
Closing則相反，先對影像進行一次dilation，然後對得到的結果進行一次erosion。

|  |
| --- |
| Mat imgClosing(imgBinary.cols, imgBinary.rows, CV\_8U, Scalar(0) );  Dilation( imgBinary, imgDilation, ker );  Erosion( imgDilation, imgClosing, ker ); |

Opening & Closing的結果：

1. Hit-and-miss



先設置hit-and-miss中J和K使用的kernel。

|  |
| --- |
| uchar kerHitMissValArr[]={255,255,  0, 255};  Mat kerHitMissVal=Mat(2,2,CV\_8U,kerHitMissValArr).clone();  Kernel kerJ(2, 2, 0, 1, kerHitMissVal);  Kernel kerK(2, 2, 1, 0, kerHitMissVal); |

然後對binary圖取contrary。

|  |
| --- |
| void Contrary( const Mat src, Mat dst )  {  for (int sI = 0; sI < src.rows; sI++)  {  for (int sJ = 0; sJ < src.cols; sJ++)  {  if ( src.at<uchar>(sI,sJ)==255 )  {  dst.at<uchar>(sI,sJ)=0;  }  else //src.at<uchar>(sI,sJ)==0  {  dst.at<uchar>(sI,sJ)=255;  }  }  }  }  Mat imgBinContrary(imgBinary.cols, imgBinary.rows, CV\_8U, Scalar(0) );  Contrary( imgBinary, imgBinContrary ); |

之後分別用J和K對原binary圖和取contrary之後的圖進行erosion。

|  |
| --- |
| Mat imgHitMissEroJ(imgBinary.cols, imgBinary.rows, CV\_8U, Scalar(0) );  Erosion( imgBinary, imgHitMissEroJ, kerJ );  Mat imgHitMissEroK(imgBinary.cols, imgBinary.rows, CV\_8U, Scalar(0) );  Erosion( imgBinContrary, imgHitMissEroK, kerK ); |

最後對erosion得到的兩個結果取intersection。

|  |
| --- |
| void intersection( const Mat src1, const Mat src2, Mat dst )  {  for (int sI = 0; sI < src1.rows; sI++)  {  for (int sJ = 0; sJ < src1.cols; sJ++)  {  if ( src1.at<uchar>(sI,sJ)==255 &&  src2.at<uchar>(sI,sJ)==255)  {  dst.at<uchar>(sI,sJ)=255;  }  else  {  dst.at<uchar>(sI,sJ)=0;  }  }  }  }  Mat imgHitMiss(imgBinary.cols, imgBinary.rows, CV\_8U);  intersection( imgHitMissEroJ, imgHitMissEroK, imgHitMiss ); |

Hit-and-miss檢測到的upper-right corner的結果：

