**Computer and Robot Vision**

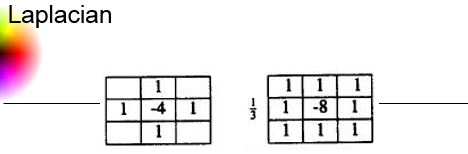
**Homework#10**

R01944040 柳成蔭

這次的作業是對原圖做Zero Crossing Edge Detection。

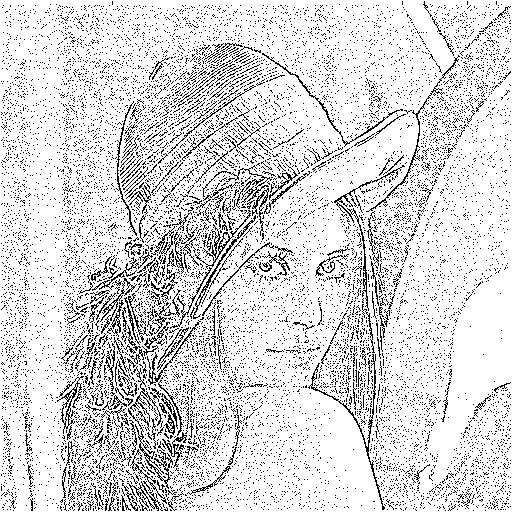
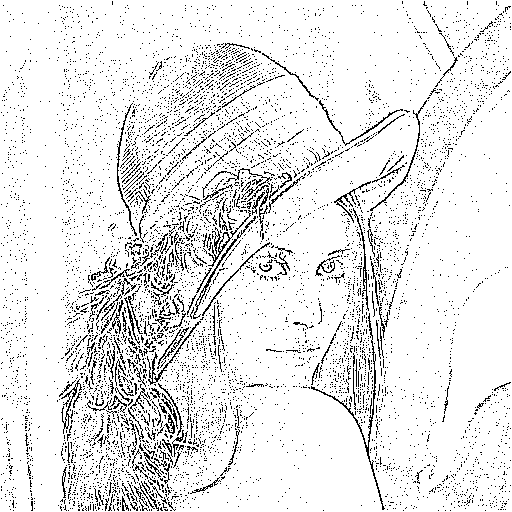
我使用VS2012編寫程式

1. Laplace Mask



|  |
| --- |
| void Laplace1(const Mat src, Mat res, int threshold)  {  //kernal  float m[]={0,1,0,  1,-4,1,  0,1,0 };  Mat M=Mat(3,3,CV\_32F,m).clone();  Kernel Mask(3, 3, 1, 1, M);  float ZeroCross;  for (int sI = 0; sI <= src.rows-1; sI++)  {  for (int sJ = 0; sJ <= src.cols-1; sJ++)  {  ZeroCross=ZeroCrossPixel(src, Mask, sI, sJ);  if(ZeroCross>=threshold)  res.at<uchar>(sI,sJ)=0;  else  res.at<uchar>(sI,sJ)=255;  }  }  }  void Laplace2(const Mat src, Mat res, int threshold)  {  //kernal  float m[]={1,1,1,  1,-8,1,  1,1,1 };  Mat M=Mat(3,3,CV\_32F,m).clone();  Kernel Mask(3, 3, 1, 1, M);  float ZeroCross;  for (int sI = 0; sI <= src.rows-1; sI++)  {  for (int sJ = 0; sJ <= src.cols-1; sJ++)  {  ZeroCross=ZeroCrossPixel(src, Mask, sI, sJ);  if( (ZeroCross/3.0) >= threshold )  res.at<uchar>(sI,sJ)=0;  else  res.at<uchar>(sI,sJ)=255;  }  }  } |

Threshold取15的處理結果：

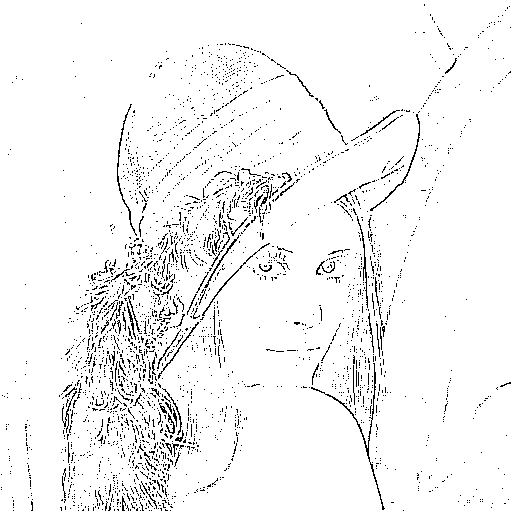
 

1. Minimum variance Laplacian

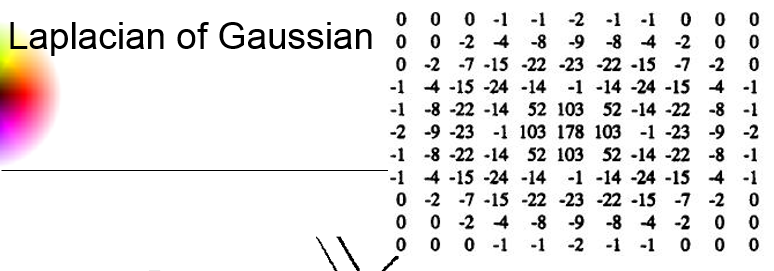


|  |
| --- |
| void MvLaplace(const Mat src, Mat res, int threshold)  {  //kernal  float m[]={2,-1,2,  -1,-4,-1,  2,-1,2 };  Mat M=Mat(3,3,CV\_32F,m).clone();  Kernel Mask(3, 3, 1, 1, M);  float ZeroCross;  for (int sI = 0; sI <= src.rows-1; sI++)  {  for (int sJ = 0; sJ <= src.cols-1; sJ++)  {  ZeroCross=ZeroCrossPixel(src, Mask, sI, sJ);  if( (ZeroCross/3.0) >= threshold )  res.at<uchar>(sI,sJ)=0;  else  res.at<uchar>(sI,sJ)=255;  }  }  } |

Threshold取20的處理結果：

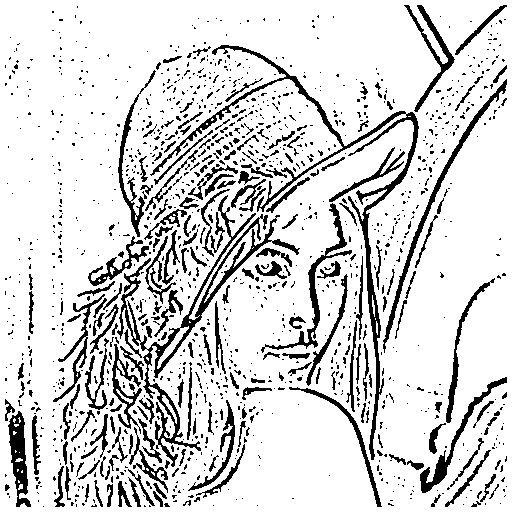


1. Laplace of Gaussian



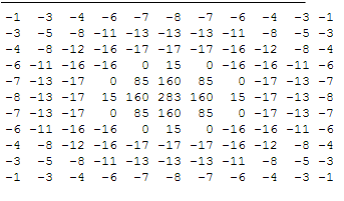
|  |
| --- |
| void LaplaceGaussian(const Mat src, Mat res, int threshold)  {  //kernal  float m[]={0, 0, 0, -1, -1, -2, -1, -1, 0, 0, 0,  0, 0, -2, -4, -8, -9, -8, -4, -2, 0, 0,  0, -2, -7, -15,-22,-23,-22,-15,-7, -2, 0,  -1, -4, -15,-24,-14,-1, -14,-24,-15,-4,-1,  -1, -8, -22,-14,52, 103,52, -14,-22,-8,-1,  -2, -9, -23,-1, 103,178,103,-1, -23,-9,-2,  -1, -8, -22,-14,52, 103,52, -14,-22,-8,-1,  -1, -4, -15,-24,-14,-1, -14,-24,-15,-4,-1,  0, -2, -7, -15,-22,-23,-22,-15,-7, -2, 0,  0, 0, -2, -4, -8, -9, -8, -4, -2, 0, 0,  0, 0, 0, -1, -1, -2, -1, -1, 0, 0, 0 };  Mat M=Mat(11,11,CV\_32F,m).clone();  Kernel Mask(11, 11, 5, 5, M);  float ZeroCross;  for (int sI = 0; sI <= src.rows-1; sI++)  {  for (int sJ = 0; sJ <= src.cols-1; sJ++)  {  ZeroCross=ZeroCrossPixel(src, Mask, sI, sJ);  if( ZeroCross >= threshold )  res.at<uchar>(sI,sJ)=0;  else  res.at<uchar>(sI,sJ)=255;  }  }  } |

Threshold取3000的處理結果：



1. Difference of Gaussian

Difference of Gaussian(inhibitory sigma=1, excitatory sigma=3, kernel size 11x11 [1][1])



|  |
| --- |
| void DifferenceGaussian(const Mat src, Mat res, int threshold)  {  //kernal  float m[]={-1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1,  -3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3,  -4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4,  -6, -11, -16, -16, 0, 15, 0, -16, -16,-11, -6,  -7, -13, -17, 0, 85, 160, 85, 0, -17,-13, -7,  -8, -13, -17, 15, 160, 283, 160, 15, -17,-13, -8,  -7, -13, -17, 0, 85, 160, 85, 0, -17,-13, -7,  -6, -11, -16, -16, 0, 15, 0, -16, -16,-11, -6,  -4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4,  -3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3,  -1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1 };  Mat M=Mat(11,11,CV\_32F,m).clone();  Kernel Mask(11, 11, 5, 5, M);  float ZeroCross;  for (int sI = 0; sI <= src.rows-1; sI++)  {  for (int sJ = 0; sJ <= src.cols-1; sJ++)  {  ZeroCross=ZeroCrossPixel(src, Mask, sI, sJ);  if( -ZeroCross >= threshold )  res.at<uchar>(sI,sJ)=0;  else  res.at<uchar>(sI,sJ)=255;  }  }  } |

Threshold取1的處理結果：

