Computer Vision hw_10

By R01922124 許彥彬

1. Intro of this homework:

   In this homework we are going to use zero crossing edge detector with different masks. They are “Laplacian”, “Minimum variance”, “Laplacian of Gaussian” and “Difference of Gaussian”.

   The following are functions and masks that going to be use.

```
int GetValue(vector<vector<int> >&, int, vector<vector<double> >&);
void ZeroCrossingEdgeDetector( cv::Mat&, cv::Mat&, int, vector<vector<double> >&, int);
void DifferenceOfGaussian ( cv::Mat&, cv::Mat&, int, int , int, int);

double laplacian_mask1[3][3] = {0,1,0,1,-4,1,0,1,0};
double laplacian_mask2[3][3] = {1/3.0,1/3.0,1/3.0,1/3.0,-8/3.0,1/3.0,1/3.0,1/3.0,1/3.0};
double mvl_mask[3][3] =  {2/3.0,-1/3.0,2/3.0,-1/3.0,-4/3.0,-1/3.0,2/3.0,-1/3.0,2/3.0};
double lg_mask[11][11] = {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};
```

2. Zero Crossing Edge Detectors

   I. Code:

```
void ZeroCrossingEdgeDetector ( Mat& src, Mat& dest,int mask_size, vector<vector<double> >& mask , int threshold) {
  vector<vector<int> > temp(src.rows,vector<int>(src.cols,0));
  vector<vector<int> > neighbour(mask_size,vector<int>(mask_size,0));
  Mat res(src.rows,src.cols,0);
  int rows = src.rows, cols = src.cols;
  uchar* row_pointer;
  int offset = mask_size/2;
  for ( int i=offset ; i<rows-offset ; i++ ){
    for ( int j=offset ; j<cols-offset; j++) {
      //get neighbours
      for ( int r=i-offset ; r<=i+offset ; r++ ) {
        for ( int c = j-offset; c<=j+offset; c++ ) {
          neighbour[r-i+offset][c-j+offset] = src.ptr(r)[c];
        }
      }
      temp[i][j] = GetValue(neighbour,mask_size,mask);
    }
  }

  for ( int i = 0 ; i<rows ; i++ ){
    row_pointer = res.ptr(i);
    for ( int j=0 ; j<cols; j++ ){
      if ( i<offset || j<offset || i>=rows-offset || j>=cols-offset ) row_pointer[j] = 255;
      else {
        int t = temp[i][j];
        if ( abs(t)<=threshold ) {
          row_pointer[j] = 255;
          continue;
        }
        int find = 0;
        for ( int r = i-offset ; r<=i+offset && !find ; r++ ) {
          for ( int c = j-offset ; c<=j+offset && !find; c++ ){
            int n = temp[r][c];
            if ( t>threshold && n<-threshold || t<-threshold && n>threshold ) {
              find=1;
              row_pointer[j]=0;
            }
            else
              row_pointer[j]=255;
          }
        }
      }
    }
  }
  dest = res;
}
```

II. Masks

i. Laplacian edge detectors with threshold 25 and 16

	1	
1	-4	1
	1	

$\frac{1}{3}$

1	1	1
1	-8	1
1	1	1

ii. Minimum Variance Laplacian with threshold 12

$\frac{1}{3}$

2	-1	2
-1	-4	-1
2	-1	2

iii. Laplacian of Gaussian with threshold 8000

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

iv. Difference of Gaussian with threshold 7

```
0.270 0.145 -0.000 -0.030 -0.023 -0.014 -0.007 -0.004 -0.002 -0.001 -0.000
0.145 0.070 -0.015 -0.029 -0.021 -0.013 -0.007 -0.003 -0.001 -0.001 -0.000
-0.000 -0.015 -0.029 -0.026 -0.018 -0.011 -0.006 -0.003 -0.001 -0.000 -0.000
-0.030 -0.029 -0.026 -0.020 -0.014 -0.008 -0.005 -0.002 -0.001 -0.000 -0.000
-0.023 -0.021 -0.018 -0.014 -0.009 -0.006 -0.003 -0.001 -0.001 -0.000 -0.000
-0.014 -0.013 -0.011 -0.008 -0.006 -0.003 -0.002 -0.001 -0.000 -0.000 -0.000
-0.007 -0.007 -0.006 -0.005 -0.003 -0.002 -0.001 -0.000 -0.000 -0.000 -0.000
-0.004 -0.003 -0.003 -0.002 -0.001 -0.001 -0.000 -0.000 -0.000 -0.000 -0.000
-0.002 -0.001 -0.001 -0.001 -0.001 -0.000 -0.000 -0.000 -0.000 -0.000 -0.000
-0.001 -0.001 -0.000 -0.000 -0.000 -0.000 -0.000 -0.000 -0.000 -0.000 -0.000
-0.000 -0.000 -0.000 -0.000 -0.000 -0.000 -0.000 -0.000 -0.000 -0.000 -0.000
```

3. Difference of Gaussian operator

I. Use $$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$ formula to create Gaussian filter and set into "Zero Crossing Edge Detectors" with given threshold.

II. Code

```
void DifferenceOfGaussian ( Mat& src, Mat& dest, int kernel_size,int var1, int var2,int threshold ) {
  //create gaussian mask
  vector<vector<double> > k1(kernel_size,vector<double>(kernel_size,0));
  vector<vector<double> > k2(kernel_size,vector<double>(kernel_size,0));
  vector<vector<double> > kernel(kernel_size,vector<double>(kernel_size,0));
  double sum1=0,sum2=0;
  for ( int i=0 ; i<kernel_size; i++ ) {
    for ( int j=0 ; j<kernel_size; j++ ){
      k1[i][j] = 1.0/(2*PI*var1*var1)*exp(-1.0*(i*i+j*j)/(2.0*var1*var1));
      k2[i][j] = 1.0/(2*PI*var2*var2)*exp(-1.0*(i*i+j*j)/(2.0*var2*var2));
      sum1+=k1[i][j];
      sum2+=k2[i][j];
    }
  }
  //normalize and minus
  for ( int i=0 ; i<kernel_size ; i++ ){
    for (int j=0 ; j<kernel_size ; j++ ){
      if ( j ) printf(" ");
      kernel[i][j] = k1[i][j]/sum1-k2[i][j]/sum2;
      printf("%.3lf",kernel[i][j]);
    }
    printf("\n");
  }
  ZeroCrossingEdgeDetector(src,dest,kernel_size,kernel,threshold);
}
```

4. "GetValue" function

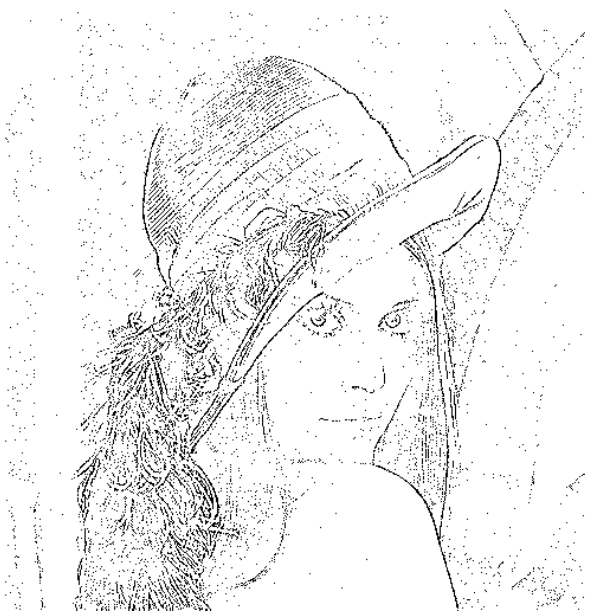
I. This function is to get the value calculated by neighbor and a given mask.

II. Code

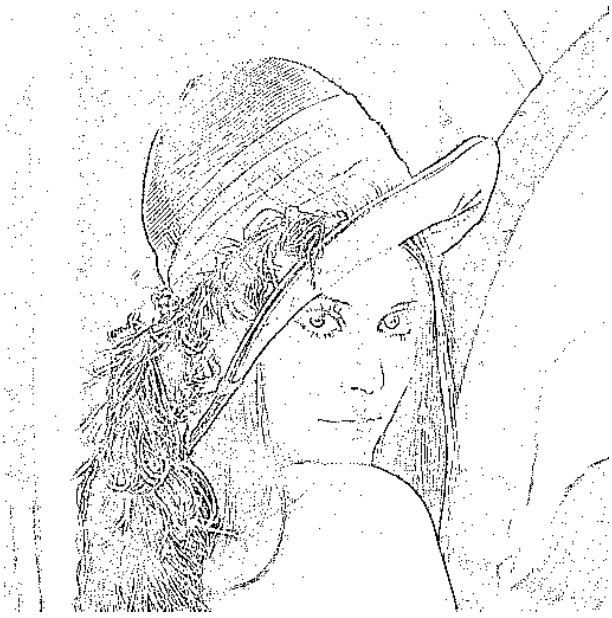
```
int GetValue(vector<vector<int> >& neighbour, int mask_size, vector<vector<double> >& mask){
  double res=0;
  for ( int i=0 ; i<mask_size ; i++ ){
    for ( int j=0 ; j<mask_size ; j++ ){
      res+= neighbour[i][j]*mask[i][j];
    }
  }
  return (int)res;
}
```

5. Results

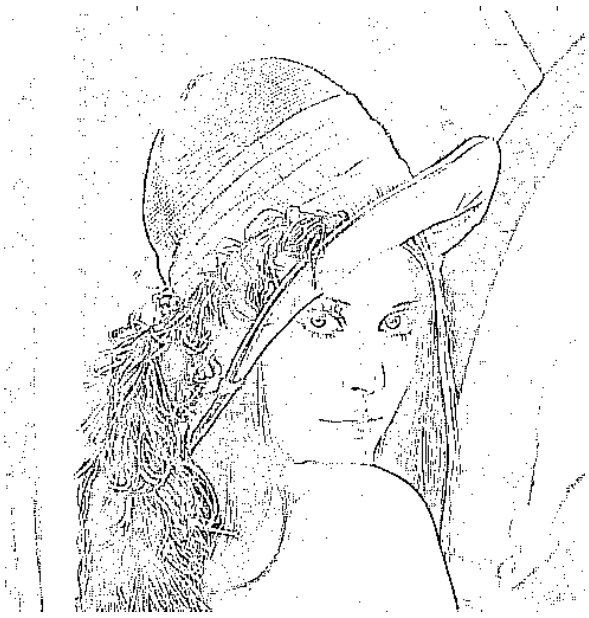
lena.bmp



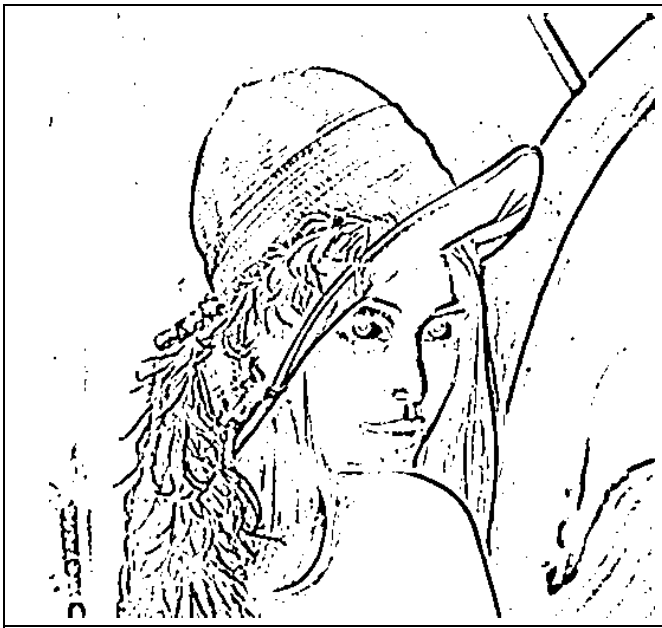
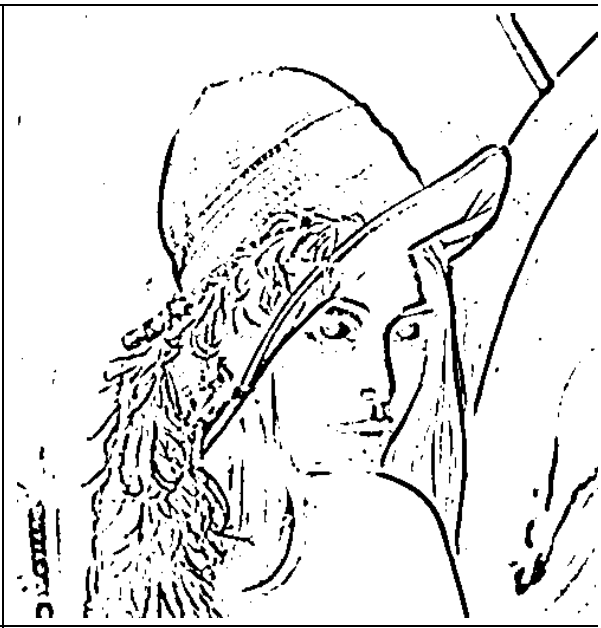
Lap_mask1_25.bmp



Lap_mask2_16.bmp



mv_Lap_12.bmp

	
Lap_of_Gaussian_8000	Diff_of_Gaussian_7.bmp

6. Appendix
   I. build_all.sh
      “sh build_all.sh” will automatically compile the code in terminal.
   II. R01922124_HW10.cpp
      source code
   III. lena.bmp
      original lena image
   IV. Many result images
   V. R01922124_HW10.pdf
      report