1 Exercises Chapter 4

1. Write (or just modify) code to compute the LoG of a greyscale image using two options: a) a single 5x5 kernel. b) two 3x3 kernels. Compare the images. Are they identical? Compare the runtime for the two approaches using a large image (use the time utility in Linux). **Answer:**

Listing 1: LoG

```
#define Mpixel(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(
    x)]

int LoG5x5[5][5]={1,3,4,2,1,  3,0,-6,0,3,  4,-6,-20,-6,4,  3,0,-6,0,3,  1,3,4,3,1};
int La[3][3]={1,1,1,  1,-8,1,  1,1,1};
int Ga[3][3]={1,2,1,  2,4,2,  1,2,1};
Mat imageinput;

int main( int argc, char** argv )
{
  namedWindow("original",0);
  namedWindow("LoG 1",0);
  namedWindow("LoG 2",0);
  if (argc != 2) {printf("needs an input image\n");exit(0);}
  imageinput = imread( argv[1], CV_LOAD_IMAGE_GRAYSCALE);
  Mat imagecopy( imageinput.rows+4,imageinput.cols+4,CV_8UC1, Scalar::all(0));
  Mat imagecopy2( imageinput.rows+4,imageinput.cols+4,CV_8UC1, Scalar::all(0));
  Mat imageLoG1(imageinput.rows,imageinput.cols,CV_8UC1, Scalar::all(0));
  Mat imageLoG2(imageinput.rows,imageinput.cols,CV_8UC1, Scalar::all(0));
  //copy data
  for(int x=0; x<imageinput.cols; x++){
    for(int y=0;y<imageinput.rows;y++){
      Mpixel(imagecopy,x+2,y+2)=Mpixel(imageinput,x,y);
    }
  }
/********** LoG 5x5 ************************************/
  for(int x=2; x<imagecopy.cols-2; x++){
    for(int y=2;y<imagecopy.rows-2;y++){
     float tempixel=0;
     tempixel=LoG5x5[0][0]*Mpixel(imagecopy,x-2,y-2)+
     LoG5x5[0][1]*Mpixel(imagecopy,x-1,y-2) + LoG5x5[0][2]*Mpixel(imagecopy,x,y-2)
         +
     LoG5x5[0][3]*Mpixel(imagecopy,x+1,y-2) + LoG5x5[0][4]*Mpixel(imagecopy,x+2,y
         -2) +
     LoG5x5[1][0]*Mpixel(imagecopy,x-2,y-1) + LoG5x5[1][1]*Mpixel(imagecopy,x-1,y
         -1) +
     LoG5x5[1][2]*Mpixel(imagecopy,x,y-1) +   LoG5x5[1][3]*Mpixel(imagecopy,x+1,y
         -1)+
     LoG5x5[1][4]*Mpixel(imagecopy,x+2,y-1) + LoG5x5[2][0]*Mpixel(imagecopy,x-2,y)
         +
```

```
35        LoG5x5[2][1]*Mpixel(imagecopy,x-1,y) +    LoG5x5[2][2]*Mpixel(imagecopy,x,y)+
36        LoG5x5[2][3]*Mpixel(imagecopy,x+1,y) +    LoG5x5[2][4]*Mpixel(imagecopy,x+2,y)
              +
37        LoG5x5[3][0]*Mpixel(imagecopy,x-2,y+1) + LoG5x5[3][1]*Mpixel(imagecopy,x-1,y
              +1) +
38        LoG5x5[3][2]*Mpixel(imagecopy,x,y+1) +    LoG5x5[3][3]*Mpixel(imagecopy,x+1,y
              +1) +
39        LoG5x5[3][4]*Mpixel(imagecopy,x+2,y+1) + LoG5x5[4][0]*Mpixel(imagecopy,x-2,y
              +2) +
40        LoG5x5[4][1]*Mpixel(imagecopy,x-1,y+2) + LoG5x5[4][2]*Mpixel(imagecopy,x,y+2)
              +
41        LoG5x5[4][3]*Mpixel(imagecopy,x+1,y+2) + LoG5x5[4][4]*Mpixel(imagecopy,x+2,y
              +2);
42        tempixel=tempixel/16.0;
43        if (tempixel>255) tempixel=255;
44        if (tempixel<0) tempixel=0;
45        Mpixel(imageLoG1,x-2,y-2)=cvRound(tempixel);
46      }
47    }
48
49  /*********** Gaussian, then Laplacian **********************/
50
51    for(int x=2; x<imagecopy.cols-2; x++){
52      for(int y=2;y<imagecopy.rows-2;y++){
53        float tempixel=0;
54        tempixel=La[0][0]*Mpixel(imagecopy,x-1,y-1)+
55         La[0][1]*Mpixel(imagecopy,x,y-1) + La[0][2]*Mpixel(imagecopy,x+1,y-1)+
56         La[1][0]*Mpixel(imagecopy,x-1,y) + La[1][1]*Mpixel(imagecopy,x,y) +
57         La[1][2]*Mpixel(imagecopy,x+1,y) + La[2][0]*Mpixel(imagecopy,x-1,y+1) +
58         La[2][1]*Mpixel(imagecopy,x,y+1) + La[2][2]*Mpixel(imagecopy,x+1,y+1);
59        tempixel=tempixel/5.0;
60        if (tempixel>255) tempixel=255;
61        if (tempixel<0) tempixel=0;
62        Mpixel(imagecopy2,x-2,y-2)=cvRound(tempixel);
63      }
64    }
65
66    for(int x=2; x<imagecopy2.cols-2; x++){
67      for(int y=2;y<imagecopy2.rows-2;y++){
68        float tempixel=0;
69        tempixel=Ga[0][0]*Mpixel(imagecopy2,x-1,y-1)+
70         Ga[0][1]*Mpixel(imagecopy2,x,y-1) + Ga[0][2]*Mpixel(imagecopy2,x+1,y-1) +
71         Ga[1][0]*Mpixel(imagecopy2,x-1,y) + Ga[1][1]*Mpixel(imagecopy2,x,y) +
72         Ga[1][2]*Mpixel(imagecopy2,x+1,y) + Ga[2][0]*Mpixel(imagecopy2,x-1,y+1) +
73         Ga[2][1]*Mpixel(imagecopy2,x,y+1) + Ga[2][2]*Mpixel(imagecopy2,x+1,y+1);
74        tempixel=tempixel/16.0;
75        if (tempixel>255) tempixel=255;
```

```
76          if (tempixel<0) tempixel=0;
77          Mpixel(imageLoG2,x-2,y-2)=cvRound(tempixel);
78        }
79      }
80      imshow("original",imageinput);
81      imshow("LoG 1",imageLoG1);
82      imshow("LoG 2",imageLoG2);
83      waitKey(0);
84  }
```

2. Write (or modify) the code for edge detection. Take an argument to choose from: Sobel, Laplacian or Canny. Note that the resulting image may be of different nChannels for different OpenCV functions. **Answer:**

Listing 2: Sobel,Laplacian and Canny

```
1  #include "cv.h"
2  #include "highgui.h"
3  #include <stdio.h>
4  #include <cstdio>
5  #include <iostream>
6
7  using namespace cv;
8  using namespace std;
9
10 IplImage *image = 0, *image2 = 0, *auximage = 0;
11 uchar* pixel;
12 int main( int argc, char** argv )
13 {
14     if(argc!=3) {printf("needs an image and 1 for Laplacian, 2 for Sobel or 3 for
           Canny\n"); exit(0);}
15     if( (image = cvLoadImage( argv[1], 1)) == 0 )
16     return -1;
17     if(atoi(argv[2])==1){
18       cvNamedWindow("Original",1);
19       cvNamedWindow("Laplacian",2);
20       image2 = cvCreateImage(cvSize(image->width, image->height), IPL_DEPTH_16S,
             1);
21       auximage = cvCreateImage(cvSize(image->width, image->height), IPL_DEPTH_8U,
             1);
22       cvCvtColor(image, auximage, CV_BGR2GRAY);
23       cvLaplace(auximage,image2,3);
24       cvConvertScaleAbs( image2, auximage );
25       cvShowImage("Original",image);
26       cvShowImage("Laplacian",auximage);
27       cvWaitKey(0);
28       cvReleaseImage(&image);
```

```
29          cvReleaseImage(&image2);
30          return 0;
31        }
32        if(atoi(argv[2])==2){
33          cvNamedWindow("Original",1);
34          cvNamedWindow("Sobel",2);
35          image2 = cvCreateImage(cvSize(image->width, image->height), IPL_DEPTH_16S,
                1);
36          auximage = cvCreateImage(cvSize(image->width, image->height), IPL_DEPTH_8U,
                1);
37          cvCvtColor(image, auximage, CV_BGR2GRAY);
38          cvSobel(auximage,image2,1,1,5);
39          cvConvertScaleAbs( image2, auximage );
40          cvShowImage("Original",image);
41          cvShowImage("Sobel",auximage);
42          cvWaitKey(0);
43          cvReleaseImage(&image);
44          cvReleaseImage(&image2);
45          return 0;
46        }
47        if(atoi(argv[2])==3){
48          cvNamedWindow("Original",1);
49          cvNamedWindow("Canny",2);
50          auximage = cvCreateImage(cvSize(image->width, image->height), IPL_DEPTH_8U,
                1);
51          image2 = cvCreateImage(cvSize(image->width, image->height), IPL_DEPTH_8U, 1)
                ;
52          cvCvtColor(image, auximage, CV_BGR2GRAY);
53          cvCanny(auximage,image2,1,100,3);
54          cvShowImage("Original",image);
55          cvShowImage("Canny",image2);
56          cvWaitKey(0);
57          cvReleaseImage(&image);
58          cvReleaseImage(&image2);
59          return 0;
60        }
61    }
```

3. Test the Hough transform using cvHoughLines (or the equivalent) using different images. What happens when the thresholds are changed? **Answer:**

Listing 3: Hough Transform from OpenCV

```
1  #include <cv.h>
2  #include <highgui.h>
3  #include <math.h>
4  #include <stdio.h>
5  int main(int argc, char** argv)
```

```
6  {
7      if (argc!=3) {printf("usage: houghlines2 image method(1:standard or 2:
           statistical\n");exit(0); }
8      const char* filename = argv[1];
9      IplImage* src = cvLoadImage( filename, 0 );
10     IplImage* dst;
11     IplImage* color_dst;
12     CvMemStorage* storage = cvCreateMemStorage(0);
13     CvSeq* lines = 0;
14     int i;
15
16     if( !src )
17         return -1;
18
19     dst = cvCreateImage( cvGetSize(src), 8, 1 );
20     color_dst = cvCreateImage( cvGetSize(src), 8, 3 );
21
22     cvCanny( src, dst, 50, 200, 3 );
23     cvCvtColor( dst, color_dst, CV_GRAY2BGR );
24     if (atoi(argv[2])==1){
25       lines = cvHoughLines2( dst, storage, CV_HOUGH_STANDARD, 1, CV_PI/180, 125,
             150, 20 );
26
27       for( i = 0; i < MIN(lines->total,100); i++ )
28       {
29         float* line = (float*)cvGetSeqElem(lines,i);
30         float rho = line[0];
31         float theta = line[1];
32         CvPoint pt1, pt2;
33         double a = cos(theta), b = sin(theta);
34         double x0 = a*rho, y0 = b*rho;
35         pt1.x = cvRound(x0 + 1000*(-b));
36         pt1.y = cvRound(y0 + 1000*(a));
37         pt2.x = cvRound(x0 - 1000*(-b));
38         pt2.y = cvRound(y0 - 1000*(a));
39         cvLine( color_dst, pt1, pt2, CV_RGB(255,0,0), 3, CV_AA, 0 );
40       }
41     }
42     if (atoi(argv[2])==2){
43       lines = cvHoughLines2( dst, storage, CV_HOUGH_PROBABILISTIC, 1, CV_PI/180,
             50, 50, 10 );
44       for( i = 0; i < lines->total; i++ )
45       {
46         CvPoint* line = (CvPoint*)cvGetSeqElem(lines,i);
47         cvLine( color_dst, line[0], line[1], CV_RGB(255,0,0), 3, CV_AA, 0 );
48       }
49     }
```

```
50      cvNamedWindow( "Source", 1 );
51      cvShowImage( "Source", src );
52      cvNamedWindow( "Hough", 1 );
53      cvShowImage( "Hough", color_dst );
54      cvWaitKey(0);
55      return 0;
56  }
```

4. Write your own Hough transform code for a straight line. What considerations are you making regarding the size of the accumulators? How these decisions affect the performance of the code? **Answer:**

Listing 4: Hough Transform Showing the Accumulator

```
1   //DEMO of accumulators for hough transforms
2   #include <cv.h>
3   #include <highgui.h>
4   #include <math.h>
5   #include <stdio.h>
6
7   #define pixel(image,x,y) ((uchar *)(image->imageData + (y)*image->widthStep))[(x)*
        image->nChannels]
8
9   void myhoughtrans (IplImage * im2)
10  {
11          int center_x, center_y, r, omega, i, j, rmax, tmax;
12          double conv;
13          double tmval=0.0;
14          conv = CV_PI/180.0;
15          center_x = im2->width/2;
16          center_y = im2->height/2;
17          rmax = (int)(sqrt((double)(im2->width*im2->width+im2->height*im2->height))
                /2.0);
18          printf("rmax is %d \n",rmax);
19          double houghspace[180][2*rmax+1];
20          IplImage * accumulator;
21          printf("image size is %d,%d \n",180,(int)2*rmax+1);
22          accumulator = cvCreateImage( cvSize(180,(int)2*rmax+1), 8, 1 );
23
24          for (r = 0; r < 2 * rmax+1; r++)
25             for (omega = 0; omega < 180; omega++)
26                  houghspace[omega][r] = 0;
27          tmax = 0; tmval = 0;
28          for (i = 0; i < im2->height; i++) {
29            for (j = 0; j < im2->width; j++) {
30                  if (pixel(im2,i,j) == 255) {
31                     for (omega = 0; omega < 180; ++omega)
32                     {
33                          r = (int)((i - center_y) * sin((double)(omega*conv))
```

```
34                              + (j - center_x) * cos((double)(omega*conv)));
35                          houghspace[omega][rmax+r] += 1;
36
37                      }
38                  }
39              }
40          }
41          int imax=0,jmax=0;
42          for (i=0; i<180; i++){
43            for (j=0; j<2*rmax+1; j++){
44              if ( (houghspace[i][j]) > tmval)
45              {
46                  tmval = houghspace[i][j];
47                  printf("tmval %lf houghspace %d %d = %lf \n",tmval,i,j,houghspace[
                        i][j]);
48                  imax = i;
49                  jmax = j;
50              }
51            }
52          }
53          printf("maxro %d largest accumulator at  %d %d =  %lf \n",rmax,imax,jmax,
                tmval);
54          for (i=0; i<180; i++){
55            for (j=0; j<2*rmax+1; j++){
56               pixel(accumulator,i,j)=cvRound(houghspace[i][j]*(255/tmval));
57            }
58          }
59          cvNamedWindow( "Accumulator", 1 );
60          cvShowImage( "Accumulator", accumulator );
61  }
62
63  int main(int argc, char** argv)
64  {
65      if (argc!=2) {printf("usage: houghtransDEMO image \n");exit(0); }
66      const char* filename = argv[1];
67      IplImage* src = cvLoadImage( filename, 0 );
68      IplImage* dst;
69      int i;
70      if( !src )
71          return -1;
72      dst = cvCreateImage( cvGetSize(src), 8, 1 );
73      cvCanny( src, dst, 100, 200, 3 );
74      myhoughtrans(dst);
75
76      cvNamedWindow( "Source", 1 );
77      cvShowImage( "Source", src );
78
```

```
79      cvNamedWindow( "Canny", 1 );
80      cvShowImage( "Canny", dst );
81
82      cvWaitKey(0);
83
84      return 0;
85  }
```