Course: CS 825
Name: Chao Zhang
Student#: 200383834
Programming Language: JAVA

Programming 1

```
package ChaoZhang;

import java.io.*;

public class P1 {
	public static void main(String args[]) throws IOException {
		FileInputStream in = null;
		FileOutputStream out = null;
		try {
			in = new FileInputStream("C:/Users/CHAO/Desktop/building.raw"); // file
path to read
			out = new FileOutputStream("C:/Users/CHAO/Desktop/building+.raw"); //
file path to write
			int i = 0, j = 0;
			int[][] image_in = new int[420][560];
			int[][] image_out_x = new int[420][560]; // compute the horizontal edge
image
			int[][] image_out_y = new int[420][560]; // compute the vertical edge
image
			int[][] image_out_g = new int[420][560]; // compute the gradient image
			int[][] image_out_g_t = new int[420][560]; // compute the thresholded
gradient image

			for (i = 0; i < 420; i++) // here is to read the binary data into
                                      // array image[][]
				for (j = 0; j < 560; j++)
					image_in[i][j] = in.read();

			int x, y; // x is horizontal edge, y is vertical edge
			double g; // g is gradient

			for (i = 1; i < 419; i++)
				for (j = 1; j < 559; j++) {
```

```
                    x = image_in[i + 1][j + 1] - image_in[i - 1][j + 1] + 2 * image_in[i
+ 1][j]
                            - 2 * image_in[i - 1][j] + image_in[i + 1][j - 1] -
image_in[i - 1][j - 1];
                    // compute x according the Hpx in Sobel Operators

                    y = image_in[i - 1][j - 1] - image_in[i - 1][j + 1] + 2 * image_in[i][j
- 1]
                            - 2 * image_in[i][j + 1] + image_in[i + 1][j - 1] -
image_in[i + 1][j + 1];
                    // compute y according the Hpy in Sobel Operators

                    x = Math.abs(x);
                    image_out_x[i][j] = x; // put calculated x into the output array

                    y = Math.abs(y);
                    image_out_y[i][j] = y; // put calculated y into the output array

                    g = Math.sqrt(x * x + y * y);
                    g = Math.abs(g);
                    image_out_g[i][j] = (int) g; // put calculated g into the output array

                    if (g > 128)
                        image_out_g_t[i][j] = 0; // calculated g with a threshold of Te =
128 and put it into the output
                                                 // array
                    else
                        image_out_g_t[i][j] = 255;
                }

            /*
             * for (i = 0; i < 420; i++) for (j = 0; j < 560; j++) // write out the
             * processed array into the image out.write(image_out_x[i][j]); // this one is
             * for the horizontal edge image
             *
             * /* for (i = 0; i < 420; i++) //write out the processed array into the image
             * for (j = 0; j < 560; j++) // this one is for the vertical edge image
             * out.write(image_out_y[i][j]);
             *
             * for (i = 0; i < 420; i++) //write out the processed array into the image for
             * (j = 0; j < 560; j++) // this one is for the gradient image
```

```
                    * out.write(image_out_g[i][j]);
                    */
                for (i = 0; i < 420; i++) // write out the processed array into the image for
                    for (j = 0; j < 560; j++) // this one is for the thresholded gradient image
                        out.write(image_out_g_t[i][j]);

                // because it only contains one output stream, so we should write out the
image    (horizontal edge image, vertical edge image, gradient image, thresholded
gradient image) one by one

            } finally

            {
                if (in != null) {
                    in.close();
                }
                if (out != null) {
                    out.close();
                }
            }
        }
}
```

horizontal edge image:

vertical edge image:

gradient image:

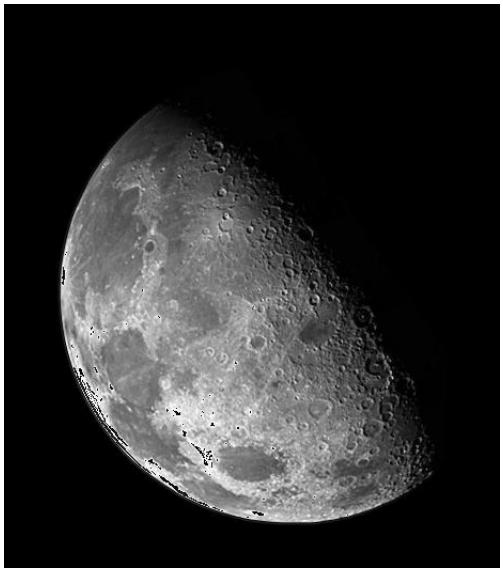
thresholded gradient image using a threshold of $T_E = 128$:

Programming 2

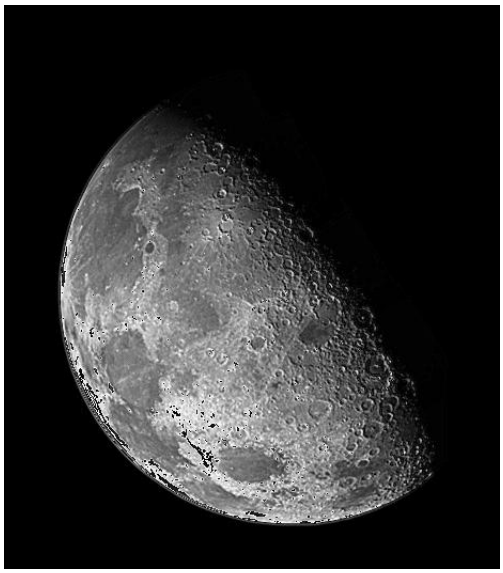
```
package ChaoZhang;

import java.io.*;

public class P2 {
	public static void main(String args[]) throws IOException {
		FileInputStream in = null;
		FileOutputStream out = null;
		try {
			in = new FileInputStream("C:/Users/CHAO/Desktop/moon.raw"); // file
path to read
			out = new FileOutputStream("C:/Users/CHAO/Desktop/moon+.raw"); // file
path to write
			int i = 0, j = 0;
			int[][] image_in = new int[528][464];
			int[][] image_out = new int[528][464];

			for (i = 0; i < 528; i++) // here is to read the binary data into
										// array image[][]
				for (j = 0; j < 464; j++)
					image_in[i][j] = in.read();

			int w = 1; // w is the coefficient and can be changed to different value for
the test
			int s;
			int[][] mask = new int[][] { { 0, 1, 0 }, { 1, -4, 1 }, { 0, 1, 0 } }; // mask is
the Laplacian sharpening

		// filter

			for (i = 1; i < 527; i++)
				for (j = 1; j < 463; j++) {
					s = 0; // s is the sum of the filter
					for (int a = -1; a < 2; a++)
						for (int b = -1; b < 2; b++) {
							s += image_in[i + a][j + b] * mask[1 + a][1 + b]; // filter
the pixel with filter
						}
```

```
                    s = image_in[i][j] - w * s; // calculate the new value according to the function in the book in
                                                // image sharping with laplacian sharping filter

                    image_out[i][j] = Math.abs(s); // s should be the absolute value before be writen into output array
                }

            for (i = 0; i < 528; i++)
                for (j = 0; j < 464; j++)
                    out.write(image_out[i][j]);

        } finally

        {
            if (in != null) {
                in.close();
            }
            if (out != null) {
                out.close();
            }
        }
    }
}
```

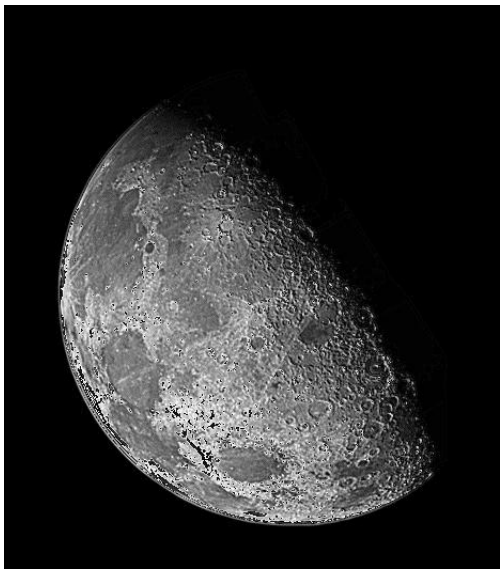
w=1



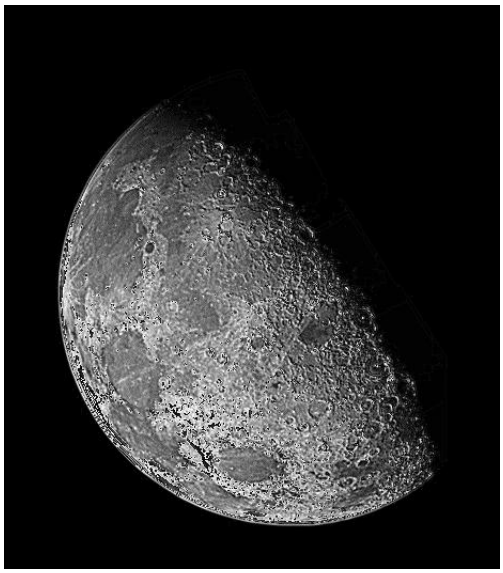
w=2



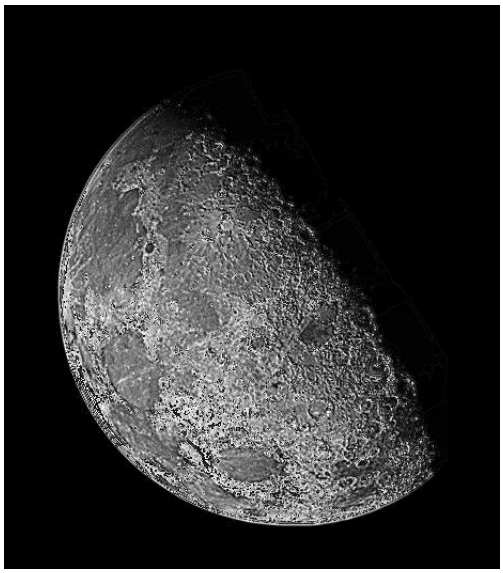
w=3



w=4



w=5



Programming 3

```
package ChaoZhang;

import java.io.*;

public class P3 {
	public static void main(String args[]) throws IOException {
		FileInputStream in = null;
		FileOutputStream out = null;
		try {
			in = new FileInputStream("C:/Users/CHAO/Desktop/lines.raw"); // file path
to read
			out = new FileOutputStream("C:/Users/CHAO/Desktop/lines+.raw"); // file
path to write

			int i, j, u, v;
			int x, y;
			int[][] image_in = new int[256][256];
			int xCtr, yCtr, nAng, nRad, cRad;
			double dAng, dRad;
			int[][] houghArray;

			xCtr = 128;	//center x is 128
			yCtr = 128;	//center y is 128
			nAng = 256;	//total steps of angle are 256
			dAng = Math.PI / nAng;	//the change of angle
			nRad = 256;	//total steps of radial are 256
			cRad = 100 / 2;	//the center of the radial
			double rMax = Math.sqrt(xCtr * xCtr + yCtr * yCtr);
			dRad = (2.0 * rMax) / nRad;
			houghArray = new int[nAng][nRad];	// hough accumulator

			for (i = 0; i < 256; i++) // here is to read the binary data into
						// array image[][]
				for (j = 0; j < 256; j++)
					image_in[i][j] = in.read();

			for (u = 0; u < 256; u++)
				for (v = 0; v < 256; v++)
					if (image_in[u][v] < 255) {
```

```
                        x = u - xCtr;
                        y = v - yCtr;

                        for (int k = 0; k < nAng; k++) {
                            double theta = dAng * k;
                            int r = cRad + (int) Math.rint((x * Math.cos(theta) + y *
Math.sin(theta)) / dRad);    //get r which is    x · cos(θ) + y · sin(θ)
                            if (r >= 0 && r < nRad)
                                houghArray[k][r]++;    //accumulate the value
                        }
                    }

            for (i = 0; i < 256; i++)
                for (j = 0; j < 256; j++)
                {
                    if (houghArray[i][j]<50) houghArray[i][j]=0;      //set the threshold
value to 50
                    out.write(255-houghArray[i][j]);    //invert the image for better
analyzing
                }


        } finally

        {
            if (in != null) {
                in.close();
            }
            if (out != null) {
                out.close();
            }
        }
    }
}
```

The dark point is the largest peak point in the accumulator in parameter space

Original hough accumulator (the image has been inverted for better looking)

To analyse the hough space, after setting threshold with 50 to get the new processed image, we can see 2 dark points