

## Histogram and Object Moment

Description of the algorithms/formulas used :

-Compute the histogram of the image and determine how many objects are in the image and the gray level of each.

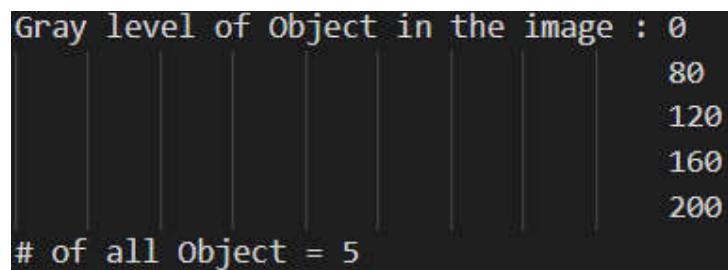
ใช้วิธีการหาค่า Gray level ของแต่ละค่าสีก่อน แล้วนำมาคำนวณ Gray level ที่มีค่ามากกว่า 1000 เพราะเนื่องจากเป็นก้อน Object จึงต้องมีค่ามากกว่า 1000 อญี่เหลว แต่เราจะไม่นับรวม 255 ที่เป็น Background

-Write a procedure and compute the central moments.

ใช้ตามสูตรที่อาจารย์ให้มาของ moment โดยในแต่ละกรวยคำนวน จะสร้าง Array มาใหม่ เราได้นำค่า Gray level ที่เป็นวัตถุใน ข้อ 1 มาใส่ใน stack โดย เรานำค่าแต่ละค่าของ Gray level มาคำนวน โดยถ้า Array ไหนของภาพ ตรงกับสีนั้นให้ แปลงค่าเป็น 1 และถ้าตรงไม่ตรงให้แปลงค่าเป็น 0 ก็จะได้ ภาพที่มี เพียง Object เดียว ออกมามาหลังจากนั้นก็เข้าไปคำนวนหาค่า Central moment ตามสูตรได้เลย ทำ while loop ตามขั้นตอนด้านบนนี้ ให้ครบจำนวน ของ Object และนำค่ามาเปลี่ยนเทียบ

**Results :**

-Compute the histogram of the image and determine how many objects are in the image and the gray level of each.



จะเห็นว่ามี แค่ 5 สีของ Gray level ซึ่งตรงตามรูปที่มี จำนวน 5 Object

-Write a procedure and compute the central moments.

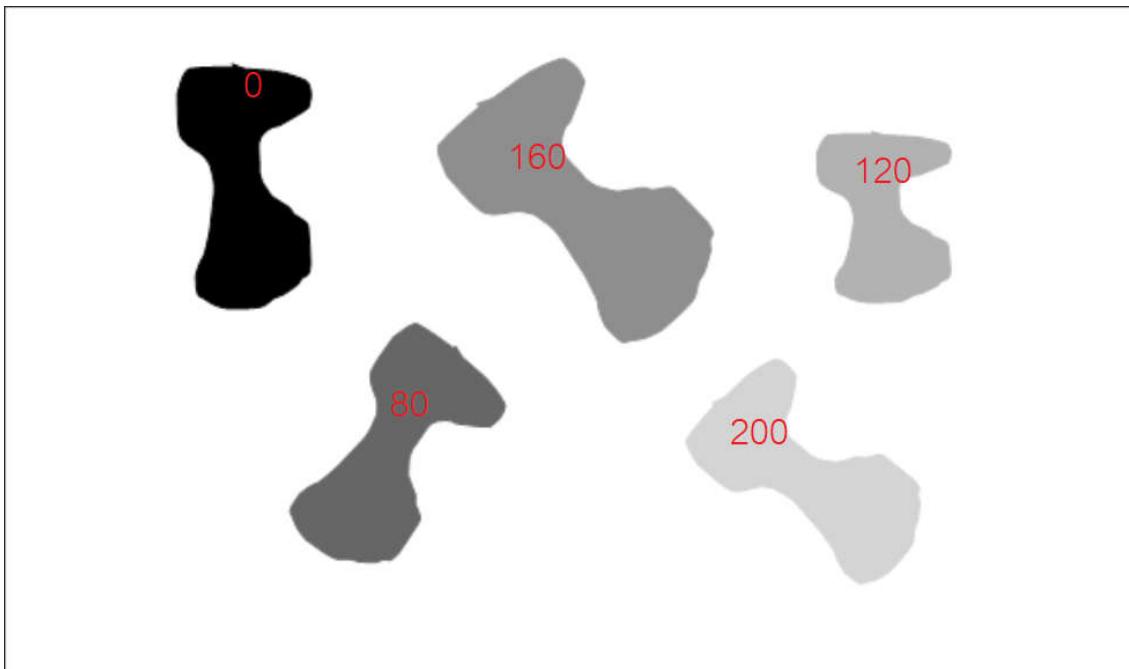
```

moment of an object Gray level (200) = 1.776457353016061
moment of an object Gray level (160) = 2.529928165323636
moment of an object Gray level (120) = 1.1686433557566196
moment of an object Gray level (80) = 1.7736852106169652
moment of an object Gray level (0) = 1.774040644748733

```

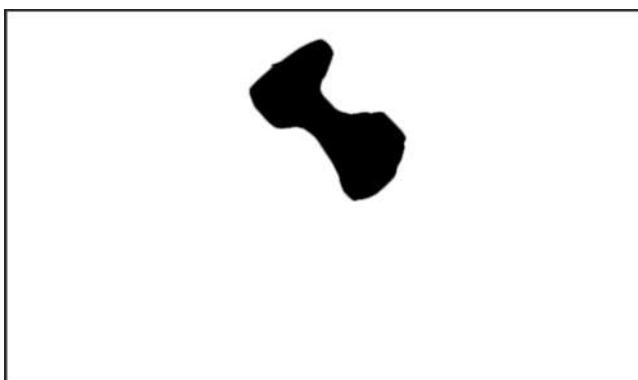
เมื่อนำไปคำนวนจะได้ค่า moment ออกมา ดังรูป และเมื่อนำไปเบรี่ยบเทียบกับรูปจริงก็จะเป็นไปตามทฤษฎี

รูปจริงเมื่อนำค่า Gray level มาเปลี่ยนเทียบ จะได้รูป



Gray Level : 0

Moment : 1.774040644748733



Gray Level : 160

Moment : 2.529928165323636



Gray Level : 120

Moment : 1.1686433557566196



Gray Level : 80

Moment : 1.7736852106169652



Gray Level : 200

Moment : 1.776457353016061

และเมื่อนำไป เปรียบเทียบกับค่า moment ก็จะได้ว่า เมื่อ Object มีขนาดเล็กก็จะทำให้ค่า moment น้อย และในทางตรงข้าม เมื่อค่า moment มีค่ามาก ก็แสดงว่า วัตถุมีขนาดใหญ่ด้วยเห็นกัน และจะเห็นได้ว่า รูปที่มี ค่า Graylevel = 0 , 80 , 200 นั้นมีค่า moment เท่ากัน นั้นแหล่ะ เป็นเพียงการหมุนรูปเท่านั้นไม่มีการ ย่อหรือขยายรูป

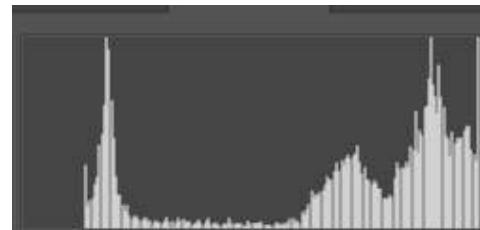
## Point Operations

Description of the algorithms/formulas used :

ใช้กราฟ equalization ปกติ โดยการ หาค่า gray level ของแต่ละค่าสีนั้นๆ จากนั้น นำค่า Gray level แต่ละอัน มาหารด้วย พื้นที่พังหมด และ นำไป บวกเพิ่มเรื่อยๆ ตามพังก์ชัน และ นำไปคูณค่า Dmax ของ รูป เมื่อได้ค่ามาแล้ว ไป round ก็จะได้ค่า Graylevel ใหม่ และก็นำไปแปลง แต่ละค่าในรูปให้เท่ากับค่าที่เราคำนวณออกมาได้

Results :

CAMERAMAN



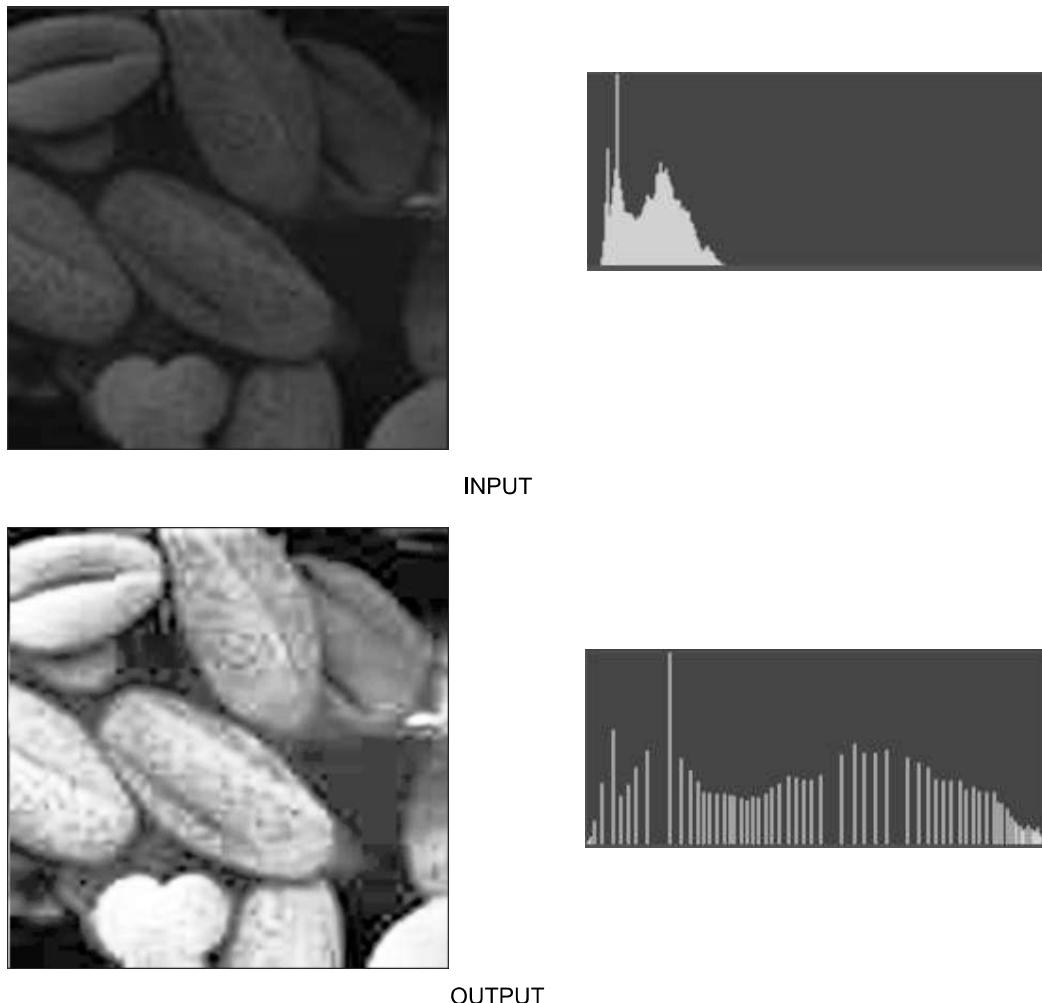
INPUT



OUTPUT

จะเห็นว่าค่า Output ที่ออกมานั้น เมื่อเปรียบเทียบกับ Input จะได้รูปที่ มีดarker เพราะรูป Input นั้นมีความสว่างของรูปสูง และเมื่อเปรียบเทียบตาม Histogram ก็จะเห็นได้ว่า จาก Input ที่ค่าต่ำ แต่สว่าง จะมีค่าสูง ก็จะแปลงมาเป็นค่าสี ที่เกลี่ยกันลงมาอยู่ แทนมีดบ้าง และ ค่า Gray level มีค่า ที่กระจาย และค่อนข้าง เท่ากันในแต่ละค่าด้วย

SEM256\_256



รูปนี้จะเห็นได้ชัดเลยว่า รูป input นั้นมีความมีดมาก และจาก Histogram จะเห็นว่า ค่า สี จะไปกองกันอยู่ตรงแบบมีดหมดเลย ทำให้ รูปออกมามีดมาก แต่เมื่อนำไปเข้าโปรแกรม ทำให้ รูปออกมาสว่างกว่า รูป Input อย่างมาก และเมื่อไปดู Histogram จะเห็นได้ชัดเลยว่า รูป มีการกระจายตัวของ ค่าสี ออกไป ในจำนวนที่เท่ากัน แต่ จะมีเพียงบางค่าที่มีค่าสูงเนื่องจาก เดิมมีค่าสูงอยู่แล้วด้วย

## Algebraic Operations

Description of the algorithms/formulas used :

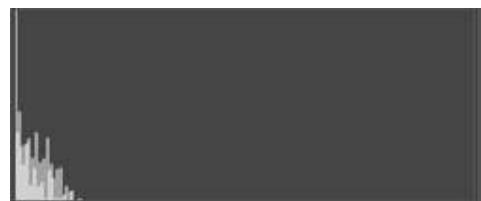
ใช้การบวกกลบทั่วไปของ Gray level เลย โดยเราจะ Input ภาพทั้ง 3 เส้นมาที่เดียวกัน และนำค่า Array ของแต่ละรูปไปดำเนินการตามที่เราต้องการได้เลย โดยถ้ามีค่า น้อยกว่า 0 ก็จะให้ค่านั้นเป็น 0 ไปเลย และเมื่อค่ามากกว่า 255 ก็จะให้ค่านั้นเป็น 255 เลยเช่นเดียวกัน

Results :

2G-R-B (excess green)



OUTPUT 1



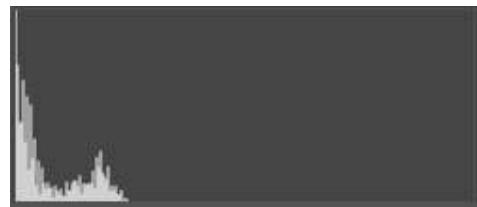
HISTOGRAM 1

จะเห็นว่า รูปที่ได้ จะเป็นเพียง ใบของต้นไม้ ซึ่งเป็นค่าที่ของ G ที่มีค่าเป็น 2 เท่า และเมื่อเทียบตาม Histogram จะได้ค่า ที่มากองกันตรง แคบมีด ซึ่งเป็นผลมาจากการที่ เราไป ลบ ออกจากทั้ง R และ B จึงทำให้เหลือเพียงค่า G อย่างเดียวตั้งรูป

R-B (red-blue difference)



OUTPUT 2



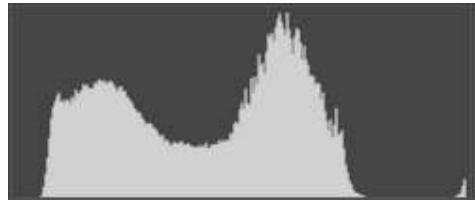
HISTOGRAM 2

จะเห็นว่า รูปที่ได้นั้น ส่วนใหญ่จะเป็นพื้นดิน และ ตัวลำต้นของต้นไม้ และใบไม้บางส่วนเท่านั้น ซึ่งเป็นผลมาจากการที่ ส่วนของ พื้นดินนั้นค่า R มีค่าสูง ทำให้ ยังคงอยู่มาก โดยที่ตอนลบออกไปคือค่า B ที่เป็นส่วนของค่าของความสว่างนั้นเอง โดย เมื่อดูตาม Histogram จะเห็นว่า ค่า มากของอยู่ในแถบมืดเข้มเดียวกับ Output 1 แต่จะมีค่าที่กระจายออกมากอยู่เกือบถึงช่วงกึ่งกลางของระหว่างแถบมืดและแถบสว่าง

$(R+B+G)/3$



OUTPUT 3



HISTOGRAM 3

จะเห็นว่า รูปที่ได้ออกมานั้น ค่อนข้างเหมือนรูปต้นฉบับ ทั้ง 3 แต่จะเป็นรูปที่ สว่างกว่าเท่านั้นเอง แต่เมื่อเรามา เทียบที่ Histogram จะเห็นได้ว่า รูปนี้มี Output ที่ค่อนข้างเหมือน G มากที่สุดแต่เพียงแค่ค่าจะเฉลี่ยน้อยลงมา ซึ่งเมื่อดู จากสมการที่ใช้ก็จะรู้ได้เลยว่า ยังไงผลลัพธ์จะต้องออกมาเป็นรูปที่สมบูรณ์ เนื่องจากเป็นการบวก และหารออกด้วย จำนวนที่เท่ากัน ไม่เหมือน รูปแบบที่ 1 และ 2 ที่เป็นการลบออกด้วยค่า สีน้ำเงิน ทำให้ค่าสีนั้นไม่เหลือค่าเลย

## Geometric Operations

Description of the algorithms/formulas used :

คำนวนหามุ่งต่างๆของ grid มาก่อน แล้วนำไปใส่ห้องในรูป Array ต่อมาคำนวนหาค่า มุ่งทั้ง 4 ของ grid และ นำมามา เทียบกับ มุ่งทั้ง 4 ของ distgrid หลังจากนั้น นำค่ามา มุ่งของทั้ง 2 รูป มา คำนวนหา  $w_1 - w_8$  ของ Bilinear Interpolation และเมื่อได้ค่ามา ให้นำกลับมาย้อนหา ค่าพิกัดใหม่ โดยนำมาย้อนกลับค่าของ ตำแหน่งของ grid จากที่ เราใส่ ของ grid ให้เปลี่ยนเป็นใส่ของ disgrid และ จากที่ใส่ของ disgrid เป็นใส่ของ grid แทน และเมื่อได้ตำแหน่งมา ก็ให้นำค่า ของ ตำแหน่งเดิมไปใส่ แต่ถ้าเป็นค่าที่ไม่ลงตัว ก็ให้นำไป round เพื่อหาเคียง แบบ Nearest Neighbor

Results :

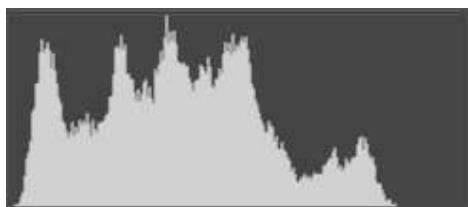


INPUT

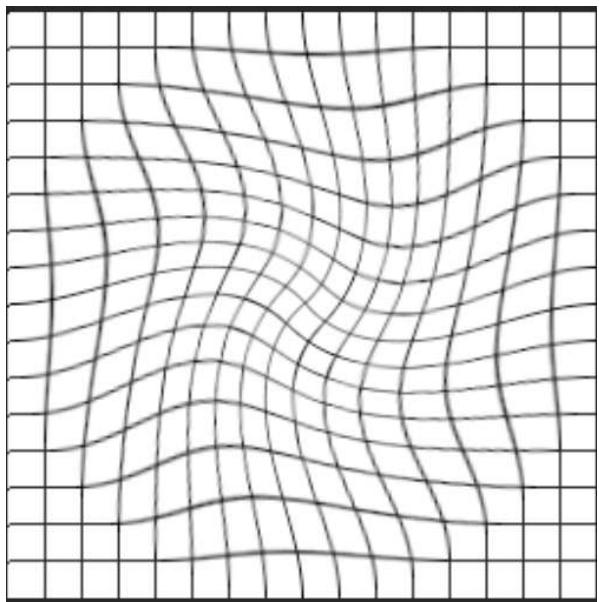
LENNA



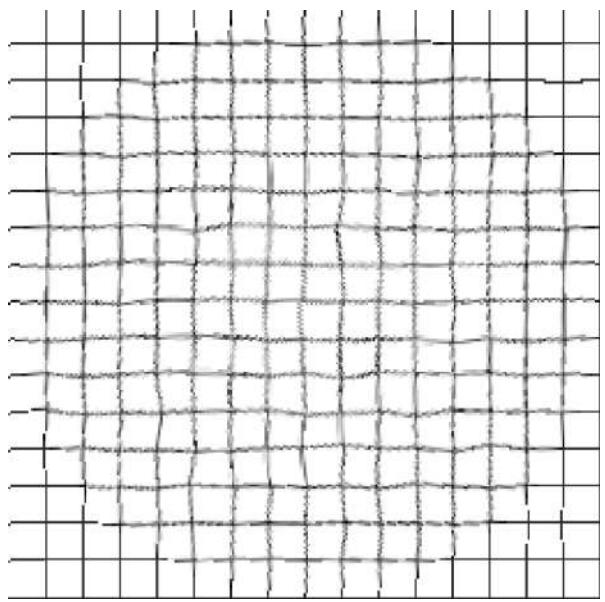
OUTPUT



จะเห็นว่า รูปที่ออกมานั้นกลับไปตรงเหมือนเดิม แต่เมื่อลองซูมดูใกล้ๆแล้วจะเกิดความไม่ชัดขึ้นแต่ถ้ามองรวมๆ ก็ ถือว่าผลลัพธ์ออกมารีดีตามคาด และจาก Histogram จะเห็นว่า ค่าแทบไม่เปลี่ยนแปลงเลยและเมื่อกลับไปดูตอนแปลง grid จะได้ Output ดังต่อไปนี้ ซึ่งถือว่า ทำออกมารีดีค่อนข้างตรง



INPUT



OUTPUT

## ภาคผนวก

### Histogram and Object Moment

```

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import java.lang.*;
import java.util.Stack;

public class ipcl {
    public static void main(String args[]) throws IOException {
        final BufferedInputStream stream_in = new BufferedInputStream(new
FileInputStream("scaled_shapes.pgm"));
        try {
            next(stream_in);
            final int row = Integer.parseInt(next(stream_in));
            final int col = Integer.parseInt(next(stream_in));
            final int max = Integer.parseInt(next(stream_in));

            final int[][] image = new int[row][col];
            final int[] D = new int[max+1];
            Stack<Integer> q = new Stack<>();
////////// นำค่า pgm เข้า array และ คำนวณค่า Graylevel //////////
            for (int i = 0; i < row; ++i) {
                for (int j = 0; j < col; ++j) {
                    final int p = stream_in.read();
                    image[i][j] = p;
                    D[p]++;
                }
            }

            System.out.print("Gray level of Object in the image : ");

```

```

////////// นำค่า Grayscale มาหาค่าที่มี มากกว่า 1000 เพื่อหาจำนวน
Object/////////
    for (int i=0 ; i < max ; i++) {
        if(D[i]>1000) {
            if(q.empty()) {
                System.out.println(i);
            }else{
                System.out.println(""
"+i);
            }
            q.add(i); //// เพิ่มค่าสี grayscale ที่เป็น Object ลงใน
stack
        }
    }
    System.out.println("# of all Object = "+q.size());
    /////// วนลูปตามจำนวน Object ///////////////
    while (q.size() != 0) {
        final int[][] m = new int[row][col];
        int removedele = q.pop();
        for (int i = 0; i < row; ++i) {
            for (int j = 0; j < col; ++j) {
                if(image[i][j]==removedele) {
                    m[i][j] = 1; ////มีค่าตรงตาม grayscale ของ Obj
                }else{
                    m[i][j] = 0;////มีค่าไม่ตรง grayscale ของ Obj
                }
            }
        }
        double theata = pqN(2, 0, m, row, col) + pqN(0, 2,
m, row, col);
        System.out.println("moment of an object Gray level (" +
removedele + ") = " + theata);
    }

} finally {
    stream_in.close();
}
////////// พิงก์ชั้นค่าน้ำณ pgm file ///////////

```

```

private static String next(final InputStream stream) throws
IOException {
    final List<Byte> bytes = new ArrayList<Byte>();
    while (true) {
        final int b = stream.read();

        if (b != -1) {
            final char c = (char) b;
            if (c == '#') {
                int d;
                do {
                    d = stream.read();
                } while (d != -1 && d != '\n' && d != '\r');
            } else if (!Character.isWhitespace(c)) {
                bytes.add((byte) b);
            } else if (bytes.size() > 0) {
                break;
            }
        }

        } else {
            break;
        }
    }

    final byte[] byteArray = new byte[bytes.size()];
    for (int i = 0; i < byteArray.length; ++i)
        byteArray[i] = bytes.get(i);
    return new String(byteArray);
}

////////// พังก์ชั่นคำนวณ moment //////////

private static int pqmoment(int p , int q , int[][] image,int row ,
int col) throws IOException{
    int m = 0;
    for (int x = 0; x < row; ++x) {
        for (int y = 0; y < col; ++y) {
            if(image[x][y] > 0) m += Math.pow(x,p)*Math.pow(y,q)*1;
            else m += 0;
        }
    }
}

```

```

        return m;
    }

    ////////////// พังค์ชั่นคำนวณ central moment //////////

private static double pqHu(int p , int q , int[][] image,int row , int col) throws IOException{
    int u = 0;
    int m10 = (pqmoment(1, 0, image,row,col));
    int m00 = (pqmoment(0, 0, image,row,col));
    int m01 = (pqmoment(0, 1, image,row,col));
    for (int x = 0; x < row; ++x) {
        for (int y = 0; y < col; ++y) {
            if(image[x][y] > 0) u +=
Math.pow((x-(m10/m00)),p)*Math.pow((y-(m01/m00)),q);
            else u += 0;
        }
    }

    return u;
}

```

### ////////// พังค์ชั่นคำนวณค่า $f_1$ //////////

```

private static double pqN(int p , int q , int[][] image,int row,int col) throws IOException{
    double n = 0;
    n = pqHu(p, q, image,row,col) / Math.pow(pqHu(0, 0,
image,row,col),((p+q/2)+1));
    return n;
}

```

## Point Operations

```

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class ipc2 {

    public static void main(String args[]) throws IOException {
        final BufferedInputStream stream_in = new BufferedInputStream(new
FileInputStream("SEM256_256.pgm"));
        final BufferedOutputStream stream_out = new
BufferedOutputStream(new FileOutputStream("SEM256_256_out.pgm"));
        try {
            next(stream_in);
            final int row = Integer.parseInt(next(stream_in));
            final int col = Integer.parseInt(next(stream_in));
            final int max = Integer.parseInt(next(stream_in));

            final int[][] image = new int[row][col];
            final float[] HA = new float[max + 1];
            final float[] PA = new float[max + 1];
            final int[] DA = new int[max + 1];
            final float[] DP = new float[max + 1];
            final int[] DB = new int[max + 1];
            final int[] DBS = new int[max + 1];
/////////// รับค่าจาก pgm และนับจำนวนค่าสี ของแต่ละ graylevel //////////
            for (int i = 0; i < row; ++i) {
                for (int j = 0; j < col; ++j) {
                    final int p = stream_in.read();
                    image[i][j] = p;
                    DA[p]++;
                }
            }
/////////// คำนวณเตาแมสูตรของ equalization //////////
        }
    }
}

```

```

for (int i = 0; i <= max; i++) {
    HA[i] = (float) DA[i] / (row * col); // คูณค่าด้วย พื้นที่ทั้งหมด
    if (i == 0) {
        PA[i] = HA[i];
    } else {
        PA[i] = PA[i - 1] + HA[i]; // เพิ่มค่าเรื่อยๆ
    }
    DP[i] = max * PA[i]; // คูณค่าด้วย Dmax
    DB[i] = Math.round(DP[i]); // ปิดค่าที่ได้
}

////////// เขียนไฟล์ PGM //////////
stream_out.write("P5".getBytes());
stream_out.write("\n".getBytes());
stream_out.write(Integer.toString(image[0].length).getBytes());
stream_out.write(" ".getBytes());
stream_out.write(Integer.toString(image.length).getBytes());
stream_out.write("\n".getBytes());
stream_out.write(Integer.toString(255).getBytes());
stream_out.write("\n".getBytes());
for (int i = 0; i < image.length; ++i) {
    for (int j = 0; j < image[0].length; ++j) {
        final int p = image[i][j];
        image[i][j] = DB[p];
        stream_out.write(image[i][j]);
    }
}
}

} finally {
    stream_in.close();
    stream_out.close();
}
}

```

#### //// ฟังก์ชันอ่านไฟล์ pgm

```

private static String next(final InputStream stream) throws
IOException {
    final List<Byte> bytes = new ArrayList<Byte>();
    while (true) {

```

```
final int b = stream.read();

if (b != -1) {

    final char c = (char) b;
    if (c == '#') {
        int d;
        do {
            d = stream.read();
        } while (d != -1 && d != '\n' && d != '\r');
    } else if (!Character.isWhitespace(c)) {
        bytes.add((byte) b);
    } else if (bytes.size() > 0) {
        break;
    }

} else {
    break;
}

}

final byte[] byteArray = new byte[bytes.size()];
for (int i = 0; i < byteArray.length; ++i)
    byteArray[i] = bytes.get(i);
return new String(byteArray);
}

}
```

## Algebraic Operations

```

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;

public class ipc3 {

    public static void main(String args[]) throws IOException {
        final BufferedInputStream stream_in1 = new BufferedInputStream(new
FileInputStream("SanFranPeak_red.pgm"));
        final BufferedInputStream stream_in2 = new BufferedInputStream(new
FileInputStream("SanFranPeak_green.pgm"));
        final BufferedInputStream stream_in3 = new BufferedInputStream(new
FileInputStream("SanFranPeak_blue.pgm"));
        final BufferedOutputStream stream_out1 = new
BufferedOutputStream(new FileOutputStream("SanFranPeak_out1.pgm"));
        final BufferedOutputStream stream_out2 = new
BufferedOutputStream(new FileOutputStream("SanFranPeak_out2.pgm"));
        final BufferedOutputStream stream_out3 = new
BufferedOutputStream(new FileOutputStream("SanFranPeak_out3.pgm"));

        try {
            next(stream_in1);
            final int col = Integer.parseInt(next(stream_in1));
            final int row = Integer.parseInt(next(stream_in1));
            final int max = Integer.parseInt(next(stream_in1));

            int loop = 4;
            while(loop > 0) {
                loop--;
                next(stream_in2);
                next(stream_in3);
            }
        }
    }
}

```

```

final int[][] imageR = new int[row][col];
final int[][] imageG = new int[row][col];
final int[][] imageB = new int[row][col];
final int[][] image1 = new int[row][col];
final int[][] image2 = new int[row][col];
final int[][] image3 = new int[row][col];
    //// รับค่าของแต่ละ รูป ใส่ใน Array ของแต่ละตัว ////
for (int i = 0; i < row; ++i) {
    for (int j = 0; j < col; ++j) {
        final int p1 = stream_in1.read();
        imageR[i][j] = p1;
        final int p2 = stream_in2.read();
        imageG[i][j] = p2;
        final int p3 = stream_in3.read();
        imageB[i][j] = p3;
        int equ11 = 2*(p2)-p1-p3; ///// คำนวน 1
        if(equ11 > 255) equ11 = 255; // ถ้าค่าเกิน
        if(equ11 < 0) equ11 = 0; // ถ้าค่าน้อยกว่า 0
        image1[i][j] = equ11;
        int equ12 = p1 - p3; ///// คำนวน 2
        if(equ12 > 255) equ12 = 255;
        if(equ12 < 0) equ12 = 0;
        image2[i][j] = equ12;
        int equ13 = Math.round((p1+p2+p3)/3); /// คำนวน 3
        if(equ13 > 255) equ13 = 255;
        if(equ13 < 0) equ13 = 0;
        image3[i][j] = equ13;
    }
}

write(image1,stream_out1); // เขียนไฟล์ output 1
write(image2,stream_out2); // เขียนไฟล์ output 2
write(image3,stream_out3); // เขียนไฟล์ output 3

} finally {
    stream_in1.close();
    stream_in2.close();
    stream_in3.close();
}

```

```

}

//// พังก์ชั่นอ่านค่า PGM ////

private static String next(final InputStream stream) throws
IOException {
    final List<Byte> bytes = new ArrayList<Byte>();
    while (true) {
        final int b = stream.read();

        if (b != -1) {

            final char c = (char) b;
            if (c == '#') {
                int d;
                do {
                    d = stream.read();
                } while (d != -1 && d != '\n' && d != '\r');
            } else if (!Character.isWhitespace(c)) {
                bytes.add((byte) b);
            } else if (bytes.size() > 0) {
                break;
            }
        }
    }
}

final byte[] bytesArray = new byte[bytes.size()];
for (int i = 0; i < bytesArray.length; ++i)
    bytesArray[i] = bytes.get(i);
return new String(bytesArray);
}

```

**////////// พังก์ชั่นเขียนไฟล์ PGM //////////**

```

private static void write(final int[][] image,final OutputStream
stream) throws IOException {
    stream.write("P5".getBytes());
    stream.write("\n".getBytes());
    stream.write(Integer.toString(image[0].length).getBytes());
    stream.write(" ".getBytes());

```

```
stream.write(Integer.toString(image.length).getBytes());
stream.write("\n".getBytes());
stream.write(Integer.toString(255).getBytes());
stream.write("\n".getBytes());
for (int i = 0; i < image.length; ++i) {
    for (int j = 0; j < image[0].length; ++j) {
        final int p = image[i][j];
        stream.write(p);
    }
}
stream.close();
}

}
```

## Geometric Operations

```

import java.io.BufferedReader;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
public class ipc4 {
    //////////////// กำหนด ค่าของ disgird //////////
    public static int[][][] disgird = new int[][][] {
        {{0, 0},{16, 0},{32, 0},{48, 0},{64, 0},{80, 0},{96,
0},{112, 0},{128, 0},{144, 0},{160, 0},{176, 0},{192, 0},{208, 0},{224,
0},{240, 0},{256, 0}},
        {{0, 16},{16, 16},{32, 16},{48, 16},{64, 16},{79, 16},{97,
17},{114, 19},{130, 18},{146, 19},{160, 18},{176, 17},{192, 16},{208,
16},{224, 16},{240, 16},{256, 16}},
        {{0, 32},{16, 32},{33, 32},{48, 32},{67, 31},{85, 35},{103,
37},{121, 40},{136, 42},{150, 43},{162, 41},{177, 37},{192, 35},{208,
32},{224, 32},{240, 31},{256, 32}},
        {{0, 48},{16, 48},{32, 48},{51, 49},{72, 49},{94, 53},{112,
56},{128, 60},{141, 63},{154, 65},{166, 65},{178, 62},{192, 57},{206,
52},{224, 48},{240, 48},{256, 48}},
        {{0, 64},{16, 64},{34, 64},{56, 63},{80, 66},{99, 68},{116,
72},{132, 76},{144, 80},{156, 84},{167, 85},{177, 83},{190, 80},{204,
74},{222, 66},{240, 64},{256, 64}},
        {{0, 80},{16, 80},{37, 78},{63, 78},{84, 78},{103, 81},{119,
85},{132, 89},{144, 94},{154, 100},{165, 103},{176, 102},{188,
100},{203, 94},{221, 85},{240, 80},{256, 80}},
        {{0, 96},{16, 96},{41, 93},{65, 91},{86, 90},{102, 90},{118,
96},{130, 102},{141, 108},{152, 116},{161, 117},{172, 119},{184,
116},{200, 112},{217, 105},{237, 97},{256, 96}},
        {{0, 112},{18, 110},{42, 106},{65, 103},{84, 101},{100,
102},{114, 105},{127, 112},{136, 119},{145, 126},{154, 130},{167,
132},{180, 132},{196, 128},{215, 122},{237, 115},{256, 112}},
        {{0, 128},{19, 126},{41, 120},{64, 113},{81, 112},{96,
112},{109, 115},{121, 120},{129, 128},{137, 135},{148, 141},{161,
143},{174, 144},{193, 142},{213, 137},{236, 131},{256, 128}}
    };
}

```

```

    {{0, 144}, {18, 141}, {40, 135}, {60, 129}, {76, 125}, {90,
124}, {101, 125}, {113, 129}, {121, 136}, {131, 144}, {142, 150}, {156,
154}, {172, 154}, {190, 154}, {212, 149}, {236, 145}, {256, 144}}),
    {{0, 160}, {17, 160}, {38, 151}, {57, 144}, {72, 140}, {85,
138}, {96, 138}, {106, 141}, {115, 148}, {126, 153}, {138, 161}, {153,
165}, {169, 167}, {190, 167}, {214, 163}, {238, 161}, {256, 160}}),
    {{0, 176}, {16, 177}, {34, 170}, {53, 162}, {66, 156}, {81,
153}, {92, 153}, {102, 156}, {112, 158}, {124, 165}, {137, 171}, {153,
174}, {171, 178}, {192, 178}, {217, 177}, {240, 176}, {256, 176}}),
    {{0, 192}, {17, 192}, {33, 191}, {51, 182}, {66, 175}, {78,
170}, {90, 169}, {101, 172}, {113, 176}, {124, 181}, {139, 184}, {155,
188}, {174, 189}, {198, 193}, {221, 192}, {240, 192}, {256, 192}}),
    {{0, 208}, {16, 208}, {31, 208}, {49, 204}, {64, 197}, {80,
193}, {89, 190}, {101, 190}, {113, 191}, {128, 195}, {144, 198}, {161,
203}, {182, 205}, {204, 206}, {224, 208}, {240, 208}, {256, 208}}),
    {{0, 224}, {16, 224}, {32, 224}, {48, 223}, {63, 221}, {80,
217}, {92, 213}, {106, 212}, {119, 212}, {133, 215}, {150, 217}, {168,
220}, {189, 222}, {208, 224}, {223, 224}, {241, 224}, {256, 224}}),
    {{0, 240}, {16, 240}, {32, 240}, {48, 240}, {64, 240}, {80,
239}, {95, 238}, {110, 237}, {125, 236}, {142, 237}, {158, 238}, {175,
239}, {192, 240}, {208, 240}, {224, 240}, {240, 240}, {256, 240}}),
    {{0, 256}, {16, 256}, {32, 256}, {48, 256}, {64, 256}, {80,
256}, {96, 256}, {112, 256}, {128, 256}, {144, 256}, {160, 256}, {176,
256}, {192, 256}, {208, 256}, {224, 256}, {240, 256}, {256, 256}})
};

///////////////// កំណត់ បុរាណ ងារ grid ///////////////

```

```

public static int[][][] grid = new int[][][] {
    {{0, 0}, {16, 0}, {32, 0}, {48, 0}, {64, 0}, {80, 0}, {96, 0}, {112,
0}, {128, 0}, {144, 0}, {160, 0}, {176, 0}, {192, 0}, {208, 0}, {224, 0}, {240,
0}, {256, 0}},

    {{0, 16}, {16, 16}, {32, 16}, {48, 16}, {64, 16}, {80, 16}, {96,
16}, {112, 16}, {128, 16}, {144, 16}, {160, 16}, {176, 16}, {192, 16}, {208,
16}, {224, 16}, {240, 16}, {256, 16}},

    {{0, 32}, {16, 32}, {32, 32}, {48, 32}, {64, 32}, {80, 32}, {96,
32}, {112, 32}, {128, 32}, {144, 32}, {160, 32}, {176, 32}, {192, 32}, {208,
32}, {224, 32}, {240, 32}, {256, 32}},

    {{0, 48}, {16, 48}, {32, 48}, {48, 48}, {64, 48}, {80, 48}, {96,
48}, {112, 48}, {128, 48}, {144, 48}, {160, 48}, {176, 48}, {192, 48}, {208,
48}, {224, 48}, {240, 48}, {256, 48}}},

```

```

    {{0 , 64},{16, 64},{32,64},{48, 64},{64, 64},{80,64},{96,
64},{112, 64},{128, 64},{144, 64},{160, 64},{176, 64},{192, 64},{208,
64},{224, 64},{240, 64},{256, 64}}},

    {{0 , 80},{16, 80},{32,80},{48, 80},{64, 80},{80,80},{96,
80},{112, 80},{128, 80},{144, 80},{160, 80},{176, 80},{192, 80},{208,
80},{224, 80},{240, 80},{256, 80}}},

    {{0 , 96},{16, 96},{32,96},{48, 96},{64, 96},{80,96},{96,
96},{112, 96},{128, 96},{144, 96},{160, 96},{176, 96},{192, 96},{208,
96},{224, 96},{240, 96},{256, 96}}},

    {{0 , 112},{16, 112},{32,112},{48, 112},{64,
112},{80,112},{96, 112},{112, 112},{128, 112},{144, 112},{160,
112},{176, 112},{192, 112},{208, 112},{224, 112},{240, 112},{256,
112}}},

    {{0 , 128},{16, 128},{32,128},{48, 128},{64,
128},{80,128},{96, 128},{112, 128},{128, 128},{144, 128},{160,
128},{176, 128},{192, 128},{208, 128},{224, 128},{240, 128},{256,
128}}},

    {{0 , 144},{16, 144},{32,144},{48, 144},{64,
144},{80,144},{96, 144},{112, 144},{128, 144},{144, 144},{160,
144},{176, 144},{192, 144},{208, 144},{224, 144},{240, 144},{256,
144}}},

    {{0 , 160},{16, 160},{32,160},{48, 160},{64,
160},{80,160},{96, 160},{112, 160},{128, 160},{144, 160},{160,
160},{176, 160},{192, 160},{208, 160},{224, 160},{240, 160},{256,
160}}},

    {{0 , 176},{16, 176},{32,176},{48, 176},{64,
176},{80,176},{96, 176},{112, 176},{128, 176},{144, 176},{160,
176},{176, 176},{192, 176},{208, 176},{224, 176},{240, 176},{256,
176}}},

    {{0 , 192},{16, 192},{32,192},{48, 192},{64,
192},{80,192},{96, 192},{112, 192},{128, 192},{144, 192},{160,
192},{176, 192},{192, 192},{208, 192},{224, 192},{240, 192},{256,
192}}},

    {{0 , 208},{16, 208},{32,208},{48, 208},{64,
208},{80,208},{96, 208},{112, 208},{128, 208},{144, 208},{160,
208},{176, 208},{192, 208},{208, 208},{224, 208},{240, 208},{256,
208}}},

    {{0 , 224},{16, 224},{32,224},{48, 224},{64,
224},{80,224},{96, 224},{112, 224},{128, 224},{144, 224},{160,
224}}

```

```

224}, {176, 224}, {192, 224}, {208, 224}, {224, 224}, {240, 224}, {256,
224}},

    {{0 , 240},{16, 240},{32,240},{48, 240},{64,
240},{80,240},{96, 240},{112, 240},{128, 240},{144, 240},{160,
240},{176, 240},{192, 240},{208, 240},{224, 240},{240, 240},{256,
240}},

    {{0 , 256},{16, 256},{32,256},{48, 256},{64,
256},{80,256},{96, 256},{112, 256},{128, 256},{144, 256},{160,
256},{176, 256},{192, 256},{208, 256},{224, 256},{240, 256},{256, 256}}
};

public static void main(String args[]) throws IOException {
    final BufferedInputStream stream_in1 = new BufferedInputStream(new
FileInputStream("distlenna.pgm"));

    final BufferedOutputStream stream_out = new
BufferedOutputStream(new FileOutputStream("disgrid_out.pgm"));

    try {
        next(stream_in1);
        final int row = Integer.parseInt(next(stream_in1));
        final int col = Integer.parseInt(next(stream_in1));
        Integer.parseInt(next(stream_in1));
        int[][] image = new int[row][col];
        int[][] image_new = new int[row][col];
////// รับค่าเข้า array ตามปกติ // ////
        for (int i = 0; i < row; ++i) {
            for (int j = 0; j < col; ++j) {
                final int p1 = stream_in1.read();
                image[i][j] = p1;
            }
        }
////// เข้าฟังก์ชัน controlgrid // ////
        image_new = controlGrid(disgrid ,grid, image, row, col);
        stream_out.write("P5".getBytes());
        stream_out.write("\n".getBytes());

        stream_out.write(Integer.toString(image[0].length).getBytes());
        stream_out.write(" ".getBytes());
        stream_out.write(Integer.toString(image.length).getBytes());
        stream_out.write("\n".getBytes());
    }
}

```

```

        stream_out.write(Integer.toString(255).getBytes());
        stream_out.write("\n".getBytes());
        for (int i = 0; i < image.length; ++i) {
            for (int j = 0; j < image[0].length; ++j) {
                final int p = image_new[i][j];
                stream_out.write(p);
            }
        }
    } finally {
    stream_in1.close();
    stream_out.close();
}
}

//// พังก์ชั่นอ่าน pgm ////

private static String next(final InputStream stream) throws
IOException {
    final List<Byte> bytes = new ArrayList<Byte>();
    while (true) {
        final int b = stream.read();
        if (b != -1) {
            final char c = (char) b;
            if (c == '#') {
                int d;
                do {
                    d = stream.read();
                } while (d != -1 && d != '\n' && d != '\r');
            } else if (!Character.isWhitespace(c)) {
                bytes.add((byte) b);
            } else if (bytes.size() > 0) {
                break;
            }
        } else {
            break;
        }
    }

    final byte[] bytesArray = new byte[bytes.size()];
    for (int i = 0; i < bytesArray.length; ++i)
        bytesArray[i] = bytes.get(i);
}

```

```

        return new String(bytesArray);
    }

    public static int[][] controlGrid( int[][][] disgridXY_new
,int[][][] gridXY , int[][] pic,int row , int col) {
        int[] x_new = new int[4];
        int[] y_new = new int[4];
        double[] x = new double[4];
        double[] y = new double[4];
        double[] w = new double[8];
        double[][] xy = new double[4][4];
        double[][] inverse = new double[4][4];
        int[][] image_new = new int[row][col];
        ///////////////// วนรับค่า ของแต่ละ ช่อง /////////////////
        for (int i = 0; i < gridXY.length-1; ++i) {
            for (int j = 0; j < gridXY[0].length-1; ++j) {
                ///////////////// รับค่า x y ของแต่ละมุม ของ grid ///////////////
                x_new[0] = gridXY[i][j][0];
                x_new[1] = gridXY[i][j+1][0];
                x_new[2] = gridXY[i+1][j][0];
                x_new[3] = gridXY[i+1][j+1][0];

                y_new[0] = gridXY[i][j][1];
                y_new[1] = gridXY[i][j+1][1];
                y_new[2] = gridXY[i+1][j][1];
                y_new[3] = gridXY[i+1][j+1][1];
                ///////////////// นำไปทำเป็น bilinear ///////////////
                for(int num = 0 ; num < 4 ; num++){
                    xy[num][0] = x_new[num];
                    xy[num][1] = y_new[num];
                    xy[num][2] = x_new[num]*y_new[num];
                    xy[num][3] = 1;
                }
                ///////////////// หา inverse ///////////////
                inverse = invert(xy);
                ///////////////// รับค่า x y ของแต่ละมุม ของ distgrid ///////////////
                x[0] = disgridXY_new[i][j][0];
                x[1] = disgridXY_new[i][j+1][0];
                x[2] = disgridXY_new[i+1][j][0];
                x[3] = disgridXY_new[i+1][j+1][0];
            }
        }
    }
}

```

```

y[0] = disgridXY_new[i][j][1];
y[1] = disgridXY_new[i][j+1][1];
y[2] = disgridXY_new[i+1][j][1];
y[3] = disgridXY_new[i+1][j+1][1];
////// หาค่าตัวแปร w ทั้งหมด /////
w[0] = (inverse[0][0] * x[0]) + (inverse[0][1] * x[1]) +
(inverse[0][2] * x[2]) + (inverse[0][3] * x[3]);
w[1] = (inverse[1][0] * x[0]) + (inverse[1][1] * x[1]) +
(inverse[1][2] * x[2]) + (inverse[1][3] * x[3]);
w[2] = (inverse[2][0] * x[0]) + (inverse[2][1] * x[1]) +
(inverse[2][2] * x[2]) + (inverse[2][3] * x[3]);
w[3] = (inverse[3][0] * x[0]) + (inverse[3][1] * x[1]) +
(inverse[3][2] * x[2]) + (inverse[3][3] * x[3]);
w[4] = (inverse[0][0] * y[0]) + (inverse[0][1] * y[1]) +
(inverse[0][2] * y[2]) + (inverse[0][3] * y[3]);
w[5] = (inverse[1][0] * y[0]) + (inverse[1][1] * y[1]) +
(inverse[1][2] * y[2]) + (inverse[1][3] * y[3]);
w[6] = (inverse[2][0] * y[0]) + (inverse[2][1] * y[1]) +
(inverse[2][2] * y[2]) + (inverse[2][3] * y[3]);
w[7] = (inverse[3][0] * y[0]) + (inverse[3][1] * y[1]) +
(inverse[3][2] * y[2]) + (inverse[3][3] * y[3]);
////// นำค่า w ที่ได้มา เข้าสมการใหม่ เพื่อย้อนสมการไปหาจอก /////
for (int k = y_new[0]; k < y_new[2]; k++) {
    for (int l = x_new[0]; l < x_new[1]; l++) {
        int xp = (int)Math.round(w[0]*l + w[1]*k +
w[2]*l*k + w[3]);
        int yp = (int)Math.round(w[4]*l + w[5]*k +
w[6]*l*k + w[7]);
        image_new[k][l] = pic[yp][xp];
    }
}
return image_new;
}

//////// สมการหา invers /////
public static double[][] invert(double a[][][])
{

```

```

int n = a.length;
double x[][] = new double[n][n];
double b[][] = new double[n][n];
int index[] = new int[n];
for (int i=0; i<n; ++i)
    b[i][i] = 1;

gaussian(a, index);

for (int i=0; i<n-1; ++i)
    for (int j=i+1; j<n; ++j)
        for (int k=0; k<n; ++k)
            b[index[j]][k]
                -= a[index[j]][i]*b[index[i]][k];

for (int i=0; i<n; ++i) {
    x[n-1][i] = b[index[n-1]][i]/a[index[n-1]][n-1];
    for (int j=n-2; j>=0; --j) {
        x[j][i] = b[index[j]][i];
        for (int k=j+1; k<n; ++k) {
            x[j][i] -= a[index[j]][k]*x[k][i];
        }
        x[j][i] /= a[index[j]][j];
    }
}
return x;
}

public static void gaussian(double a[][], int index[])
{
    int n = index.length;
    double c[] = new double[n];

    for (int i=0; i<n; ++i)
        index[i] = i;

    for (int i=0; i<n; ++i){
        double c1 = 0;
        for (int j=0; j<n; ++j) {
            double c0 = Math.abs(a[i][j]);

```

```
    if (c0 > c1) c1 = c0;
}
c[i] = c1;
}

int k = 0;
for (int j=0; j<n-1; ++j) {
    double pil = 0;
    for (int i=j; i<n; ++i) {
        double pi0 = Math.abs(a[index[i]][j]);
        pi0 /= c[index[i]];
        if (pi0 > pil) {
            pil = pi0;
            k = i;
        }
    }
    int itmp = index[j];
    index[j] = index[k];
    index[k] = itmp;
    for (int i=j+1; i<n; ++i){
        double pj = a[index[i]][j]/a[index[j]][j];
        a[index[i]][j] = pj;
        for (int l=j+1; l<n; ++l)
            a[index[i]][l] -= pj*a[index[j]][l];
    }
}
}
```