Mathematical Morphology

Mi Deng

Las Cruces High School
Las Cruces, NM 88001

Team 52

Mentor: Professor Joe Song, NMSU Computer Science Department

April 2, 2008

Abstract

Mathematical morphology is the analysis of images using mathematical functions and matrices. With the use of computers, these techniques can greatly reduce time spent in manually comparing images and can reveal unseen trends in pictures taken over time. With just the basic functions of mathematical morphology, my team analyzed the pictures of dyed fruit fly larvae to identify gene expression with Java. Through our program, we were able to identify changes in image sequences arranged in chronological order (over time). In the future, my program could be expanded into a larger program that incorporates the conversion from color to grayscale, the automated determination of orientation, and the writing of resultant images to a file all in one package instead of relying on user input steps through other software.

1. Project Background

1.1 Problem Statement

The analysis of photographs displaying gene expression over time in biological systems is time-consuming because the photographs must be compared manually. We would like to use mathematical morphology to automate the process, which would yield faster and more accurate results.

1.2 Project Objectives

With Java object-oriented programming, we would like to create a program that can read image sequences and convert them into matrices so mathematical morphology operations can be successfully to determine the change in each image from the one image in the sequence to the subsequent image in the sequence.

1.3 Definitions

Mathematical morphology changes an image (colors, shapes) into a matrix of numbers by converting each pixel's value into a matrix with the dimensions of the image. Then, a structural element (second, much smaller matrix) is chosen and functions based on the two matrices create a resultant matrix that replaces the data (input) matrix. Due to the non-linearity of the functions, one cannot truly predict the structure of the resultant if the operations are applied repeatedly (as is the case in my project). Thus, the structural element can be arbitrary, as long as it will induce a different resultant matrix from the input matrix and is significantly smaller than the inputted data matrix. The two most basic functions are erosion and dilation.

Because images also exist in color or shades of gray, two types of mathematical matrices with different erosion and dilation functions are employed.

The simplest type corresponds to the computer file type pbm (portable binary map). This file type requires each element of the matrix to be a binary bit (1 or 0) and dilation (A+B) is simply the creation of a matrix (C) where every 1 in C has the coordinates of $C_{(x+i),\,(y+j)}$ where $A_{x,\,y}$ and $B_{i,\,j}$ are also ones. Erosion (A-B) is the opposite function and causes every 1 in C to have the coordinates of every (x, y) in A if $A_{x,\,y}$ is a 1 and $A_{(x-i),\,(y-j)}$ is a 1 for every valid value of (i, j).

Unfortunately, the simple parity definition fails for grayscale images, which can have numbers run from 0 (black) to 255 or even to 65535 (white). To solve this problem,

a general, multidimensional method is employed on pgm files (portable grayscale map). The dilation function is defined as the superimposition of the data matrix over the structural element. This means that the upper left corner of A is placed on each valid (i, j) and the value of $C_{x+i, y+j}$ is $A_{x+i, y+j}$ plus $B_{x+i, y+j}$. Any matrix value where B is undefined (since B is smaller in dimensions than A) is set to be 0. If more than 1 value is defined at (x+i, y+j), the value of C is the maximum of the set of values.
Erosion is similarly defined through superimposition. The upper left corner of A is placed on each valid (i, j) and the value of $C_{x+i, y+j}$ is $A_{x+i, y+j}$ minus $B_{x+i, y+j}$. Any matrix value where B is undefined (since B is a smaller in dimensions than A) is set to be 0. If more than 1 value is defined at (x+i, y+j), the value of C is the minimum of the set of values.

Since the maximum value is limited by the resolution of the image (8-bit or 16-bit), the numbers must be readjusted to this after an erosion or dilation function because it is possible to have a negative matrix value or a value above the maximum.

Immediately, one begins to see differences between grayscale and binary operations. The grayscale operations are truly non-linear, since they employ minimum and maximum. Also, erosion shrinks the image in binary operations, but both erosion and dilation increase image size in grayscale operations when the structural element is nontrivial (larger than 1 by 1 and isn't all zeros).

Although the two functions seem like inverses, the two functions are not. In fact, two distinct composite functions exist: open and close. Open is erosion followed by dilation, while close is dilation followed by erosion. As their names suggest, close can close "holes", groups of 0s surrounded by 1s, while open can create more "holes".

1.4 Applications

We chose to apply our mathematical morphology program to the analysis of stained fruit fly gene expression over time.

The experiment, termed BDGP (Berkeley Drosophila Genome Project), and funded by UC Berkeley and numerous national organizations, intended to finish the sequencing of the genome of the fruit fly by analyzing complementary DNA, DNA that

code for active, mRNA producing genes. They accomplished this by collecting mRNA from rough endoplasmic reticulum.

To determine gene expression in the developing embryo over the hours of maturation, BDGP preserved fly embryos in various states of development from 1 to 11 hours (after egg-laying) in liquid nitrogen, followed by a methanol and then a formaldehyde solution. RNA probes and colored substrates (to detect the probes) then stained the fruit flies in areas where the gene was expressed. Finally, the embryos were photographed.

2. Project Approach

Taking the color pictures from the BDGP website, the image is converted to grayscale through ImageMagick. If any image has a different orientation from the first image, it is rotated by multiples of 90 degrees so that all have the same orientation. Then, in the appropriate folder, we created an extra file, marking the order of the list of grayscale pictures in chronological sequence.

The program is then run. It first reads the list, identifying each file. Then, it reads each file, one at a time, first by reading the file format and then the dimensions to generate an array of sufficient size. The file type also allows the program to decide which set of functions to apply. Then, the program reads the image's numbers and converts the string or characters into integers and sets the values to the array by setting each array value to each corresponding image coordinate. Lastly, the program prints each array value in order for the user to see.

Since we are measuring the change in time, we must adapt the existing functions. The program dilates one picture on of the fruit fly sequence a special structural element. The values on the perimeter of the "picture" of the structural element are 0 (black). The center of the element is gray or close to white (high value). By the definition of grayscale erosion, the dilated resultant image will only change in the places where the central, non-zero element was added to the inputted image. By keeping the structural element much smaller than the actual image, this problem becomes almost trivial. By subtracting the next element in the series (and setting undefined points at 0) to this resultant dilated result, one finds the gradient, or change over time in any area of the image. Naturally, these results are printed and are displayed as the final data since they depict the change in gene expression over time.

Since I used object oriented programming, each major step is a class organized into different Java image objects that depend on the image type (binary, grayscale, sequence). Each function that the program uses is a method.

3. Results and Discussion

We analyzed the apt gene on the fruit fly.

The original images were:

Image 1:

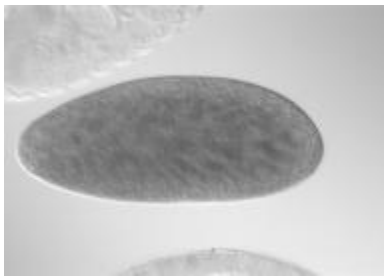
 (Stage 1-3; 0-2 Hours)

Image 2:

 (Stage 4-6; 2-4 Hours)

Image 3:

 (Stage 7-8; 4-4.5 Hours)

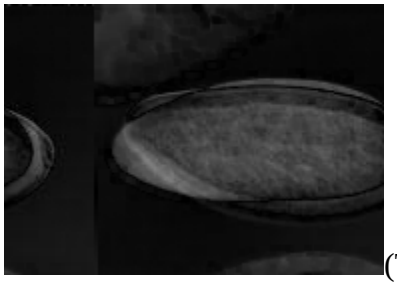
Image 4:

 (Stage 9-10; 4.5-5.5 Hours)

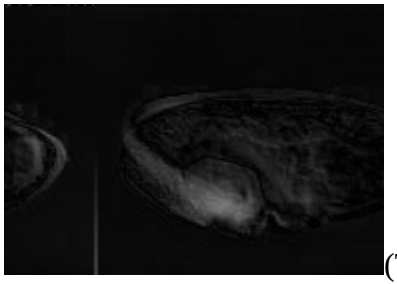
Image 5:

(Stage 11-12; 6-9.5 Hours)

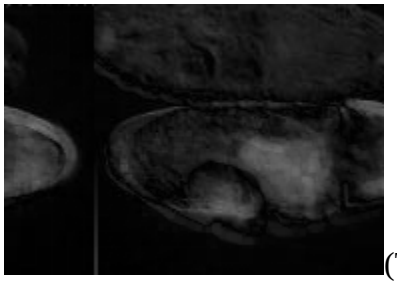
Using our dilation gradient, we found that:



(Transition from Image 1 to 2)



(Transition from Image 2 to 3)



(Transition from Image 3 to 4)

Clearly, our reader isn't perfect. It read the last fourth of the image and then read the image from the beginning. However, since our reader read every image that way, the result is not affected.

Dark areas represent areas of little change in gene expression. White areas represent areas of high change in gene expression.

4. Conclusions

Our program was successful in reading image sequences and taking the gradient. It was able to distinguish the differences in gene expression and expression it in color. In the future, more genetic data from other fruit fly genes will be analyzed through this method.

5. Acknowledgments

I would like to think Professor Joe Song and Mr. Greg Marez for assisting me with the programming of this project. I would also like to thank my former team member Sam Kester, who gave useful tips even after officially leaving my team due to schedule conflicts.

6. Sources

1. http://java.sun.com/javase/6/docs/api/
2. www.fruitfly.org
3. www.netpbm.sourceforge.net/doc/**pgm**.html
4. http://schmidt.devlib.org/jiu/introduction.html
5. www.imagemagick.org
6. Java—How to Program, Dietel & Associates, Inc. Prentice Hall, (2003)
7. Lecture Notes on Mathematical Morphology: Grayscale Images, Richard Alan Peters II, (2006)
8. http://en.wikipedia.org/wiki/Two's_complement

Appendix A (Source Code):

Class: BImageReader

```
package testMiDengsProgram;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;
/**
 *
 * @author Joe Song
 */
public class BImageReader{

   /* Creates a new instance of ImageReader */
   public BImageReader() {
   }

   public BinaryImage Scan(String filename) {
      int width = 0;
      int height = 0;
      int corner_row = 0;
      int corner_col = 0;
      int [][] array1 = null;
      BinaryImage image = null;

      // open the file for reading
      try {
         BufferedReader input = new BufferedReader( new FileReader( filename ) );
```

```
String text;
int section = 0;
int row = 0;
// scan the file to get image content
while ( ( text = input.readLine() ) != null ) {
    // section 1: pbm header
    // check for the pbm flag
    text = text.trim();
    if (text.length() == 0){
    continue;
    }

    switch(section) {
        case 0:
            if(text.contains("P1")) {
                System.out.println("Header:" + text);
                section = 1;
            }
            break;
        case 1:     // pbm
            // split(text);
            if (text.contains("# UpperLeft")){
                String s2[] = text.split(" ");
                corner_row = Integer.parseInt( s2[2] );
                corner_col = Integer.parseInt( s2[3] );
                continue;
            } else if (text.contains("#")) {
                continue;}
            String s0[] = text.split(" ");
            width = Integer.parseInt( s0[0] );
            height = Integer.parseInt( s0[1] );
```

```
                array1 = new int [height][width];
                image = new BinaryImage(width, height, corner_row, corner_col);
                System.out.println("Width:" + width);
                System.out.println("Height:" + height);
                System.out.println("corner_row:" + corner_row);
                System.out.println("corner_col:" + corner_col);
                section = 2;
                break;
            case 2:     // width, height
                String s1[] = text.split(" ");
                for(int col = 0; col < s1.length; col++){
                   image.Set(row, col, Integer.parseInt( s1[ col ] ));
                   array1 [row][col] = Integer.parseInt( s1[ col ] );
                }
                row++;
//                System.out.println("Pixels:" + text);
                // split(text);
                // section = 3;
                break;
            case 3:

                // pixel values in a row
                break;
            default:
                break;
          }
        }
//          image.Print();
        // MUST return an image object

      } catch( IOException ioException ) {
```

```
            System.out.println("ERROR: Cannot open file!");
            //JOptionPane.showMessageDialog( this, "FILE ERROR",
            //        "FILE ERROR", JOptionPane.ERROR_MESSAGE );
        }
        return image;
    }
}
```

Class: BinaryImage

```
/*
 * BinaryImage.java
 *
 * Created on November 2, 2007, 8:56 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package testMiDengsProgram;

/**
 *
 * @author Joe Song
 */
public class BinaryImage {

    private int m_pixels[][] = null;
    private int m_width = 0;
    private int m_height = 0;
    private int m_worlduleft_col = 0;
    private int m_worlduleft_row = 0;
    public int Width(){ return m_width; }
    public int Height(){ return m_height; }
    public int Corner_WCol(){ return m_worlduleft_col; }
    public int Corner_WRow(){ return m_worlduleft_row; }

    public void Set(int row, int col, int val){
        m_pixels[row][col] = val;
```

```
}
public int Get(int row, int col){
   return m_pixels[row][col];
}
public void SetUpperLeft(int wrow, int wcol){
   InitializeCorner(wrow, wcol);
}
public void Print(){
      for(int row = 0; row < m_height; row++){
         for(int col = 0; col < m_width; col++) {
            System.out.print( m_pixels[row][col]  + "\t");
         }
         System.out.print("\n");
      }
}

/** Creates a new instance of BinaryImage */
public BinaryImage(int width, int height, int corner_col, int corner_row) {
   Initialize(width, height);
   InitializeCorner(corner_col, corner_row);
}
public BinaryImage(int width, int height) {
   Initialize(width, height);
}
private void Initialize(int width, int height) {
   m_width = width;
   m_height = height;
   m_pixels = new int [height][width];

}
private void InitializeCorner(int corner_col, int corner_row){
```

```
        m_worlduleft_col = corner_col;
        m_worlduleft_row = corner_row;}
}
```

Class BinaryMM:

```
/*
 * MM.java
 *
 * Created on November 9, 2007, 8:21 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */
package testMiDengsProgram;

/**
 *
 * @author Song and Li
 */
public class BinaryMM {

    public BinaryImage Dilate(BinaryImage A, BinaryImage B) {
        int col_corner = A.Corner_WCol() + B.Corner_WCol();
        int row_corner = A.Corner_WRow() + B.Corner_WRow();
        BinaryImage C = new BinaryImage(A.Width() + B.Width() - 1, A.Height() +
B.Height() - 1);
        for (int a_col = 0; a_col < A.Width(); a_col++) {
            for (int a_row = 0; a_row < A.Height(); a_row++) {
                for (int b_col = 0; b_col < B.Width(); b_col++) {
                    for (int b_row = 0; b_row < B.Height(); b_row++) {
                        if ((A.Get(a_row, a_col) == 1) && (B.Get(b_row, b_col) == 1)) {
                            C.Set(a_row + b_row, a_col + b_col, 1);
                        }
```

```
            }

        }
    }

}
C.SetUpperLeft(row_corner, col_corner);
return C;
}

public BinaryImage Erode(BinaryImage A, BinaryImage B) {
int counter = 0;
int counter2 = 0;
int col_corner = A.Corner_WCol() - B.Corner_WCol();
int row_corner = A.Corner_WRow() - B.Corner_WRow();
BinaryImage D = new BinaryImage(A.Width() - B.Width() + 1, A.Height() -
B.Height() + 1);
for (int b_col = 0; b_col < B.Width(); b_col++) {
for (int b_row = 0; b_row < B.Height(); b_row++) {
if (B.Get(b_row, b_col) == 1) {
counter2++;
}
}
}
for (int a_col = 0; a_col < A.Width(); a_col++, counter = 0) {
for (int a_row = 0; a_row < A.Height(); a_row++, counter = 0) {
for (int b_col = 0; b_col < B.Width(); b_col++) {
for (int b_row = 0; b_row < B.Height(); b_row++) {
if (A.Get(a_row, a_col) == 1 && B.Get(b_row, b_col) == 1) {
if ((a_row + b_row < A.Height()) && (a_col + b_col < A.Width())) {
if (A.Get(a_row + b_row, a_col + b_col) == 1) {
```

```
                        counter++;
                        if (counter == counter2) {
                            D.Set(a_row, a_col, 1);
                        }
                    }
                }

            }
        }
    }
    D.SetUpperLeft(row_corner, col_corner);
    return D;
}

public BinaryImage Close(BinaryImage A, BinaryImage B) {
    BinaryImage E;
    E = Dilate(A, B);
    BinaryImage F;
    F = Erode(E, B);
    return F;
}

public BinaryImage Open(BinaryImage A, BinaryImage B) {
    BinaryImage G;
    G = Erode(A, B);
    BinaryImage H;
    H = Dilate(G, B);
    return H;
}
```

```
}
/** Creates a new instance of MM */
```

Class: GrayscaleImage

```
/*
 * BinaryImage.java
 *
 * Created on November 2, 2007, 8:56 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package testMiDengsProgram;

/**
 *
 * @author Joe Song
 */
public class GrayscaleImage {

    private int m_pixels[][] = null;
    private int m_width = 0;
    private int m_height = 0;
    private int m_intensity = 0;
    private int m_worlduleft_col = 0;
    private int m_worlduleft_row = 0;
    public int Width(){ return m_width; }
    public int Height(){ return m_height; }
    public int Brightness() { return m_intensity; }
    public int Corner_WCol(){ return m_worlduleft_col; }
    public int Corner_WRow(){ return m_worlduleft_row; }
```

```
public void Set(int row, int col, int brightness){
   m_pixels[row][col] = brightness;
}
public int Get(int row, int col){
   return m_pixels[row][col];
}
public void SetUpperLeft(int wrow, int wcol){
   InitializeCorner(wrow, wcol);
}
public void Print(){
      for(int row = 0; row < m_height; row++){
         for(int col = 0; col < m_width; col++) {
            System.out.print( m_pixels[row][col]  + "\t");
         }
         System.out.print("\n");
      }
}

/** Creates a new instance of BinaryImage */
public GrayscaleImage(int width, int height, int corner_col, int corner_row) {
   Initialize(width, height);
   InitializeCorner(corner_col, corner_row);
}
public GrayscaleImage(int width, int height) {
   Initialize(width, height);
}
private void Initialize(int width, int height) {
   m_width = width;
   m_height = height;
   m_pixels = new int [height][width];
```

```
    }
    private void InitializeCorner(int corner_col, int corner_row){
        m_worlduleft_col = corner_col;
        m_worlduleft_row = corner_row;}
}
```

Class: GrayscaleMM:

```
/*
 * MM.java
 *
 * Created on November 9, 2007, 8:21 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */
package testMiDengsProgram;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;
import java.io.PushbackInputStream;

/**
 *
 * @author Song and Li
 */
public class GrayscaleMM {

    public GrayscaleImage Dilate(GrayscaleImage A, GrayscaleImage B) {
        int col_corner = A.Corner_WCol() + B.Corner_WCol();
        int row_corner = A.Corner_WRow() + B.Corner_WRow();
        int record1 = 0;
        GrayscaleImage C = new GrayscaleImage(A.Width() + B.Width() - 1, A.Height() +
B.Height() - 1);
        for (int b_col = 0; b_col < B.Width(); b_col++) {
```

```
            for (int b_row = 0; b_row < B.Height(); b_row++) {
               for (int a_col = 0; a_col < A.Width(); a_col++) {
                  for (int a_row = 0; a_row < A.Height(); a_row++) {
                     if ((a_row + b_row > B.Height() - 1) || (a_col + b_col > B.Width() - 1)) {
                        record1 = A.Get(a_row, a_col);
                     } else {
                        record1 = (A.Get(a_row, a_col) + B.Get(a_row + b_row, a_col +
b_col));
                     }
                     if (record1 > C.Get(a_row + b_row, a_col + b_col)) {
                        C.Set(a_row + b_row, a_col + b_col, record1);
                     }
                  }
               }
            }
         }
         C.SetUpperLeft(row_corner, col_corner);









         return C;
      }

      public GrayscaleImage Erode(GrayscaleImage A, GrayscaleImage B) {
         GrayscaleImage D = new GrayscaleImage(A.Width() + B.Width() - 1, A.Height() +
B.Height() - 1);
         GrayscaleImage I = new GrayscaleImage(B.Width(), B.Height());
```

```
        for (int b_col = 0; b_col < B.Width(); b_col++) {
            for (int b_row = 0; b_row < B.Height(); b_row++) {
                I.Set(b_row, b_col, (-B.Get((B.Height() - 1 - b_row), (B.Width() - 1 - b_col))));
            }
        }
        I.SetUpperLeft(-(B.Height() - 1 + B.Corner_WRow()), -(B.Width() - 1 +
B.Corner_WCol()));
        D = Dilate(A, I);









        return D;
    }

    public GrayscaleImage Close(GrayscaleImage A, GrayscaleImage B) {
        GrayscaleImage E;
        E = Dilate(A, B);
        GrayscaleImage F;
        F = Erode(E, B);
        return F;
    }

    public GrayscaleImage Open(GrayscaleImage A, GrayscaleImage B) {
        GrayscaleImage G;
        G = Erode(A, B);
        GrayscaleImage H;
```

```
    H = Dilate(G, B);
    return H;
  }

  public GrayscaleImage DGradient(String filename, GrayscaleImage B) {
    GrayscaleImage I;
    GrayscaleImage K;
    I = null;
    K = null;
    String file;
    int depth = -1;
    try {
      BufferedReader input = new BufferedReader(new FileReader(filename));
      int row = 0;
      int col = 0;
      // scan the file to get image content
      input = new BufferedReader(new FileReader(filename));
      while ((file = input.readLine()) != null) {
        // section 1: pbm header
        // check for the pbm flag


        depth++;
        GSImageReader reader = new GSImageReader();
        GrayscaleImage A = reader.Scan(file);
        GrayscaleImage J = new GrayscaleImage(A.Width() + B.Width() - 1,
A.Height() + B.Height() - 1);
        K = new GrayscaleImage(A.Width() + B.Width() - 1, A.Height() + B.Height() -
1);
        if (depth == 0) {
```

```
            I = new GrayscaleImage(A.Width() + B.Width() - 1, A.Height() + B.Height()
- 1);
        }
        for (int a_col = 0; a_col < A.Width() + B.Width() - 1; a_col++) {
            for (int a_row = 0; a_row < A.Height() + B.Height() - 1; a_row++) {
                if (a_col <= A.Width() - 1) {
                    if (a_row <= A.Height() - 1) {
                        J.Set(a_row, a_col, A.Get(a_row, a_col));
                    }
                }
                if (a_col > A.Width() - 1) {
                    if (a_row > A.Height() - 1) {

                        J.Set(a_row, a_col, 0);
                    }
                }
                K.Set(a_row, a_col, (J.Get(a_row, a_col) - I.Get(a_row, a_col)));
            }
        }
            K.Print();


        I = Dilate(A, B);

    }

} catch (IOException ioException) {

    System.out.println("ERROR: Cannot open file!");
//JOptionPane.showMessageDialog( this, "FILE ERROR",
//        "FILE ERROR", JOptionPane.ERROR_MESSAGE );
```

```
        }
        return null;
    }
}
/** Creates a new instance of MM */
```

Class: GSImageReader:

```
/*
 * ImageReader.java
 *
 * Created on October 26, 2007, 8:47 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */
package testMiDengsProgram;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;
import java.io.PushbackInputStream;

/**
 *
 * @author Joe Song
 */
public class GSImageReader {

    /* Creates a new instance of ImageReader */
    public GSImageReader() {
    }


    public GrayscaleImage Scan(String filename) {
        int width = 0;
```

```
int height = 0;
int corner_row = 0;
int corner_col = 0;
int maxvalue = 0;
int[][] array1 = null;
GrayscaleImage image = null;

// open the file for reading
try {
   BufferedReader input = new BufferedReader(new FileReader(filename));
   String text;
   int section = 0;
   int row = 0;
   // scan the file to get image content
   while ((text = input.readLine()) != null) {
      // section 1: pbm header
      // check for the pbm flag


      switch (section) {
         case 0:
            if (text.contains("P2")) {
               System.out.println("Header:" + text);
               section = 1;
            }
            if (text.contains("P5")) {
               System.out.println("Header:" + text);
               section = 4;
            }
            break;
         case 1:     // pbm
```

```
    // split(text);
    if (text.contains("# UpperLeft")) {
       String s2[] = text.split(" ");
       corner_row = Integer.parseInt(s2[2]);
       corner_col = Integer.parseInt(s2[3]);
       continue;
    } else if (text.contains("#")) {
       continue;
    }
    String s0[] = text.split(" ");
    width = Integer.parseInt(s0[0]);
    height = Integer.parseInt(s0[1]);
    array1 = new int[height][width];
    image = new GrayscaleImage(width, height, corner_row, corner_col);
    System.out.println("Width:" + width);
    System.out.println("Height:" + height);
    System.out.println("corner_row:" + corner_row);
    System.out.println("corner_col:" + corner_col);
    section = 2;
    break;
case 2:     // width, height
    String s1[] = text.split("\\s+");
    if (s1.length == 1) {
       continue;
    }
    for (int col = 0; col < s1.length; col++) {
       image.Set(row, col, Integer.parseInt(s1[col]));
       array1[row][col] = Integer.parseInt(s1[col]);
    }
    row++;
    //System.out.println("Pixels:" + text);
```

```
    // split(text);
    // section = 3;
    break;
case 3:

    // pixel values in a row
    break;
case 4:
    if (text.contains("# UpperLeft")) {
        String s2[] = text.split(" ");
        corner_row = Integer.parseInt(s2[2]);
        corner_col = Integer.parseInt(s2[3]);
        continue;
    } else if (text.contains("#")) {
        continue;
    }
    String s10[] = text.split(" ");
    width = Integer.parseInt(s10[0]);
    height = Integer.parseInt(s10[1]);
    image = new GrayscaleImage(width, height, corner_row, corner_col);
    System.out.println("Width:" + width);
    System.out.println("Height:" + height);
    System.out.println("corner_row:" + corner_row);
    System.out.println("corner_col:" + corner_col);
    section = 5;
    break;
case 5:
    maxvalue = Integer.parseInt(text);
    System.out.println("maxvalue:" + maxvalue);
    section = 6;
    break;
```

```
                case 6:
                    int byteperpix = 0;
                    byte[] pushbackarray = null;
//                    char[] bufferedarray = null;
                    if (maxvalue > 255) {
                        byteperpix = 2;
                    } else {
                        byteperpix = 1;
                    }
                    pushbackarray = new byte[width * height * byteperpix];
//                    bufferedarray = new char[width * height * byteperpix];
                    PushbackInputStream input2 = new PushbackInputStream(new
FileInputStream(filename));
                    input2.read(pushbackarray, 0, width * height * byteperpix);
//                    input.read(bufferedarray, 0, width * height * byteperpix);
                    for (int col = 0; col < width; col++) {
                        for (int row2 = 0; row2 < height; row2++) {
                            int pixval = 0;
                            int offset = (col + row2 * width) * byteperpix;
//                            if (bufferedarray[offset] >= 256) {
//                                System.out.println("Error in (" + col + "," + row2 + "):" + ((int)
(bufferedarray[offset]) / 256) + "\t" + ((int) (bufferedarray[offset]) % 256));
//                            }
//                            if ((bufferarray[offset + 1] >= 256) && byteperpix==2) {
//                                System.out.println("Error in (" + row2 + "," + col + "):" + (int)
(bufferarray[offset + 1]));
//                            }

                            if (byteperpix == 2) {
//                                pixval = (int) (bufferarray[offset]);
```

```
                        pixval = (((int) (pushbackarray[offset]) << 8)) | (int)
(pushbackarray[offset + 1]);
                    }
                    if (byteperpix == 1) {
                        pixval = (int) (pushbackarray[offset]);
                    }
                    if (pixval < 0) {
                        pixval = 256+pixval;
                    }

                    image.Set(row2, col, pixval);
                  }
                }
                break;
              default:
                break;
            }
          }
//          image.Print();
      // MUST return an image object

      } catch (IOException ioException) {

         System.out.println("ERROR: Cannot open file!");
      //JOptionPane.showMessageDialog( this, "FILE ERROR",
      //       "FILE ERROR", JOptionPane.ERROR_MESSAGE );
      }
      return image;
   }
}
```

Class: SImageReader:

```
/*
 * ImageReader.java
 *
 * Created on October 26, 2007, 8:47 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */
package testMiDengsProgram;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;

/**
 *
 * @author Joe Song
 */
public class SImageReader {

    /* Creates a new instance of ImageReader */
    public SImageReader() {
    }

    public ImageSequence Scan(String filename) {
        int width = 0;
        int height = 0;
        int depth = 0;
```

```
        int corner_row = 0;
        int corner_col = 0;
        int corner_lvl = 0;
        String file;
        String text2;
        int [][][] array1 = null;
        ImageSequence image = null;

        // open the file for reading
        try {
           BufferedReader input = new BufferedReader(new FileReader(filename));
           String text;
           int dep = 0;
           int row = 0;
           int col = 0;
           // scan the file to get image content
           while ((text2 = input.readLine()) != null){
              depth++;
           }
           input.close();
           input = new BufferedReader(new FileReader(filename));
           while ((file = input.readLine()) != null) {
              // section 1: pbm header
              // check for the pbm flag



                    GSImageReader reader = new GSImageReader();
                    GrayscaleImage A = reader.Scan(file);
//                    A.Print();
                    if(dep == 0){
```

```
                width = A.Width();
                height = A.Height();
                corner_row = A.Corner_WRow();
                corner_col = A.Corner_WCol();
                array1 = new int [height][width][depth];
                image = new ImageSequence(width, height, depth);}
                for (row = 0; row < height; row++){
                for (col = 0; col < width; col++) {
                   image.Set(row, col, dep, A.Get(row, col));
                   array1[row][col][dep] = A.Get(row, col);
                }
                }

                   dep++;


                //System.out.println("Pixels:" + text);
                // split(text);
                // section = 3;


                // pixel values in a row

           }
//           image.Print();
        // MUST return an image object

        } catch (IOException ioException) {

           System.out.println("ERROR: Cannot open file!");
        //JOptionPane.showMessageDialog( this, "FILE ERROR",
```

```
        //        "FILE ERROR", JOptionPane.ERROR_MESSAGE );
        }
        return image;
    }
}
```

Class: SequenceMM:

```
/*
 * MM.java
 *
 * Created on November 9, 2007, 8:21 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */
package testMiDengsProgram;

/**
 *
 * @author Song and Li
 */
public class SequenceMM {

    public ImageSequence Dilate(ImageSequence A, ImageSequence B) {
        int col_corner = A.Corner_WCol() + B.Corner_WCol();
        int row_corner = A.Corner_WRow() + B.Corner_WRow();
        int lvl_corner = A.Corner_WLvl() + B.Corner_WLvl();
        int record1 = 0;
        ImageSequence C = new ImageSequence(A.Width() + B.Width() - 1, A.Height() +
B.Height() - 1, A.Depth() + B.Depth() - 1);
        for (int b_dep = 0; b_dep < B.Depth(); b_dep++) {
            for (int b_col = 0; b_col < B.Width(); b_col++) {
                for (int b_row = 0; b_row < B.Height(); b_row++) {
                    for (int a_dep = 0; a_dep < A.Depth(); a_dep++) {
                        for (int a_col = 0; a_col < A.Width(); a_col++) {
                            for (int a_row = 0; a_row < A.Height(); a_row++) {
```

```
                        if ((a_row + b_row > B.Height() - 1) || (a_col + b_col > B.Width() -
1) || (a_dep + b_dep > B.Depth() - 1)) {
                            record1 = A.Get(a_row, a_col, a_dep);
                        } else {
                            record1 = (A.Get(a_row, a_col, a_dep) + B.Get(a_row + b_row,
a_col + b_col, a_dep + b_dep));
                        }
                        if (record1 > C.Get(a_row + b_row, a_col + b_col, a_dep + b_dep)) {
                            C.Set(a_row + b_row, a_col + b_col, a_dep + b_dep, record1);
                        }
                    }
                }
            }
        }
    }
    C.SetUpperLeft(row_corner, col_corner, lvl_corner);









    return C;
  }

  public ImageSequence Erode (ImageSequence A, ImageSequence B) {
    ImageSequence D = new ImageSequence(A.Width() + B.Width() - 1, A.Height() +
B.Height() - 1, A.Depth() + B.Depth() - 1);
    ImageSequence I = new ImageSequence(B.Width(), B.Height(), B.Depth());
```

```
        for (int b_dep = 0; b_dep < B.Depth(); b_dep++){
        for (int b_col = 0; b_col < B.Width(); b_col++) {
            for (int b_row = 0; b_row < B.Height(); b_row++) {
                I.Set(b_row, b_col, b_dep, (-B.Get((B.Height() - 1 - b_row), (B.Width() - 1 -
b_col), (B.Depth() - 1 - b_dep))));
            }
        }

        }
        I.SetUpperLeft(-(B.Height() - 1 + B.Corner_WRow()), -(B.Width() - 1 +
B.Corner_WCol()), -(B.Depth() - 1 + B.Corner_WLvl()));
        D = Dilate(A, I);









        return D;
    }

    public ImageSequence Close(ImageSequence A, ImageSequence B) {
        ImageSequence E;
        E = Dilate(A, B);
        ImageSequence F;
        F = Erode(E, B);
        return F;
    }
```

```
    public ImageSequence Open(ImageSequence A, ImageSequence B) {
        ImageSequence G;
        G = Erode(A, B);
        ImageSequence H;
        H = Dilate(G, B);
        return H;
    }

}
/** Creates a new instance of MM */
```

Class: ImageSequence

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package testMiDengsProgram;

/**
 *
 * @author Joe Song
 */
public class ImageSequence {

    private int m_pixels[][][] = null;
    private int m_width = 0;
    private int m_height = 0;
    private int m_depth = 0;
    private int m_intensity = 0;
    private int m_worlduleft_col = 0;
    private int m_worlduleft_row = 0;
    private int m_worlduleft_lvl = 0;

    public int Width() {
        return m_width;
    }

    public int Height() {
        return m_height;
    }
```

```
public int Depth() {
   return m_depth;
}

public int Brightness() {
   return m_intensity;
}

public int Corner_WCol() {
   return m_worlduleft_col;
}

public int Corner_WRow() {
   return m_worlduleft_row;
}

public int Corner_WLvl(){
   return m_worlduleft_lvl;
}

public void Set(int row, int col, int lvl, int brightness) {
   m_pixels[row][col][lvl] = brightness;
}

public int Get(int row, int col, int lvl) {
   return m_pixels[row][col][lvl];
}

public void SetUpperLeft(int wrow, int wcol, int wlvl) {
   InitializeCorner(wrow, wcol, wlvl);
}
```

```
    public void Print() {
        for (int lvl = 0; lvl < m_depth; lvl++) {
            for (int row = 0; row < m_height; row++) {
                for (int col = 0; col < m_width; col++) {
                    System.out.print(m_pixels[row][col][lvl] + "\t");
                }

                System.out.print("\n");
            }
            System.out.print("\n");
        }
    }

    /** Creates a new instance of BinaryImage */
    public ImageSequence(int width, int height, int depth, int corner_col, int corner_row,
int corner_lvl) {
        Initialize(width, height, depth);
        InitializeCorner(corner_col, corner_row, corner_lvl);
    }

    public ImageSequence(int width, int height, int depth) {
        Initialize(width, height, depth);
    }

    private void Initialize(int width, int height, int depth) {
        m_width = width;
        m_height = height;
        m_depth = depth;
        m_pixels = new int[height][width][depth];
```

```
    }

    private void InitializeCorner(int corner_col, int corner_row, int corner_lvl) {
        m_worlduleft_col = corner_col;
        m_worlduleft_lvl = corner_lvl;
        m_worlduleft_row = corner_row;
    }
}
```