Homework 3

1. ## GaussianOp

   Write a class named `GaussianOp` that performs a Gaussian blurring.

   <<implements BufferedImageOp, pixeljelly.ops.PluggableImageOp>>
   GaussianOp

   ```
   + GaussianOp( double alpha, double sigma )
   + getDefault( BufferedImage src ) : BufferedImageOp
   + getAuthorName() : String
   + getStrength() : double
   + getAlpha() : double
   + getSigma() : double
   + setAlpha( double alpha ) : void
   + setSigma( double sigma ) : void
   ```

   a. `getDefault()` : returns a `GuassianOp` having an `alpha` of 2 and a `sigma` of 3.
   b. `filter` : This filter applies a Guassian blur filter to an image. This method must be efficient.

2. ## Fast Median

   Write a class named `FastMedianOp` that performs median filtering using an MxN region using the histogram-based technique.

   <<implements BufferedImageOp, pixeljelly.ops.PluggableImageOp>>
   FastMedianOp

   ```
   + FastMedianOp( int m, int n )
   + getDefault( BufferedImage src ) : BufferedImageOp
   + getAuthorName() : String
   + getM( ) : int
   + getN( ) : int
   ```

   a. `filter( BufferedImage src , BufferedImage dst)` . This method median filters the src by using the relatively fast histogram-based median-filtering technique.
   b. `getDefault()` : Returns a FastMedianOp using a 9x9 region.

3. ## Oriented Edge

   Write a class named OrientedEdgeOp. This filter applies a red highlighting to any selected edge in an image. Edges are selected by specifying an orientation angle, an epsilon (tolerance), and a strength value. A sample is on the edge if the MoG exceeds the strength value AND the orientation-of-gradient is within epsilon of the orientation.

   <<implements BufferedImageOp, pixeljelly.ops.PluggableImageOp>>
   OrientedEdgeOp

   ```
   + OrientedEdgeOp( double strength, double orientation, double epsilon )
   + getDefault( BufferedImage src ) : BufferedImageOp
   + getAuthorName() : String
   + getStrength() : double
   + getOrientation() : double
   + getEpsilon() : double
   + setStrength( double strength ) :void
   + setOrientation( double orientation ) : void
   + setEpsilon( double epsilon ) : void
   ```

   a. `getDefault` : returns an OrientedEdgeOp having a strength of .5, an orientaion of .75 and an epsilon of .15.
   b. `filter` : This filter applies a red highlighting to any selected edge in an image. An edge is selected by three controlling parameters: a `strength` given as a double in the range [0-1], an `orientation` given as a double in the range [0-1] and an `epsilon` given as a double in the range [0-1]. The orientation is a normalized angular measure such that 0 correspond to 0 degrees and 1 corresponds to 360 degrees. The source is copied to the destination unless the sample is an edge sample in which case the destination is set to red. A sample is an edge sample if the normalized magnitude of gradient is greater-than-or-equal-to the strength and the orientation of the gradient is within epsilon of the controlling orientation setting.

4. ## CBIR

   Write a command-line program named `ImageDatabase` that implements a **content based image recognition** (CBIR) system. A CBIR system maintains a database of images (i.e. a collection of images, not necessarily a SQL database). The database can be queried by providing an image Q. The response of the query is an ordered list of database images such that images at the beginning of the list are more visually similar to Q than those at the end of the response.

   ### Command Line Options

   The program is used by typing commands into a terminal. Those commands are described below.

   - `create` <Xn> <Yn> <Zn> <URL_FILENAME> <DB_FILENAME> <COLOR_MODEL>
   - `query` <Q_URL> <DB_FILENAME> <RESPONSE_FILENAME> <K>

   The meaning of each parameter is described below.

   - <X_n> <Y_n> <Z_n> : each of these is an integer in [1, 8] that denotes the number of bits allocated to each of the three bands, given here as X, Y, and Z, in the color histogram. These bands will be red, green and blue if the <COLOR_MODEL> is RGB; othewise they will be hue, saturation and brightness if the <COLOR_MODEL> is HSB. The sum of X_n, Y_n, and Z_n must be less than or equal to 10. For example, the number of color histogram bins in the red dimension is given as 2^{R_n} when <COLOR_MODEL> is RGB.
   - <COLOR_MODEL> : The value of COLOR_MODEL will always be **RGB** for undergraduate students. For graduate students, you must support both **RGB** and **HSB**.
   - <URL_FILENAME> : the name of a URL file. A URL file is a plain-text file such that each line contains three URL's. The first is the URL of a flickr user. The second is the URL of a thumnail. The third is the URL of an image; this is the URL that our application will make use of.
   - <DB_FILENAME> : The name of a DB file. A DB file is a plain-text file such that the first line contains the resoultions used in the file (i.e. R_n, G_n, and B_n) and each subsequent line contains an image URL along with the color histogram of that image.
   - <Q_URL> : The URL of an image file.
   - <RESPONSE_FILENAME> : The name of a RESPONSE file. A RESPONSE file is an HTML file that, when viewed in a browser, displays the response of some `ImageDatabase` query. At the top of the page is the query image Q and the resolution used in the query (i.e. X_n, Y_n, and Z_n). Each subsequent image is one of the images in the database and they occur in the same order as the response. Each image is a link to some URL. Each of the reponse images also displays its distance from Q using the color-histogram similarity technique described in class.
   - <K> : The number of images in the query repsonse.

   ### Description

   The `create` command reads through the list of images given in the URL file. For each image in the file, its color histogram is computed (using the resolutions denoted as X_n, Y_n, Z_n) and saved into the DB file.

   The `query` command reads the Q image and constructs a color histogram having the same dimensions as histograms in the DB file. The program will then compute the similarity between the color hisogram of Q and the color histogram of every image in the DB file. It will produce an ordered list of URLs denoting the K images of the DB file that are most visually similar to Q. This list will be persisted as a RESPONSE file.

Required Testing

You must run your code using the following command-line arguments. Make sure to include the created files in your git submission. Also, an html file of the url-file.txt database may help to visualize the results.

1. java ImageDatabase create 1 1 1 url-file.txt db-1-1-1-rgb.txt rgb
2. java ImageDatabase create 2 2 2 url-file.txt db-2-2-2-rgb.txt rgb
3. java ImageDatabase create 3 4 3 url-file.txt db-3-4-3-rgb.txt rgb
4. java ImageDatabase create 3 5 2 url-file.txt db-3-5-2-rgb.txt rgb
5. java ImageDatabase query https://charity.cs.uwlax.edu/views/cs454/homeworks/hw3/red-frog.png db-1-1-1.txt query-result-1-1-1-rgb.html 150
6. java ImageDatabase query https://charity.cs.uwlax.edu/views/cs454/homeworks/hw3/red-frog.png db-2-2-2.txt query-result-2-2-2-rgb.html 150
7. java ImageDatabase query https://charity.cs.uwlax.edu/views/cs454/homeworks/hw3/red-frog.png db-3-4-3.txt query-result-3-4-3-rgb.html 150
8. java ImageDatabase query https://charity.cs.uwlax.edu/views/cs454/homeworks/hw3/red-frog.png db-3-5-2.txt query-result-3-5-2-rgb.html 150
9. java ImageDatabase query https://upload.wikimedia.org/wikipedia/commons/c/c5/Clanculus_pharaonius_01.JPG db-1-1-1.txt query-result-1-1-1-rgb-shell.html 150
10. java ImageDatabase query https://upload.wikimedia.org/wikipedia/commons/c/c5/Clanculus_pharaonius_01.JPG db-2-2-2.txt query-result-2-2-2-rgb-shell.html 150
11. java ImageDatabase query https://upload.wikimedia.org/wikipedia/commons/c/c5/Clanculus_pharaonius_01.JPG db-3-4-3.txt query-result-3-4-3-rgb-shell.html 150
12. java ImageDatabase query https://upload.wikimedia.org/wikipedia/commons/c/c5/Clanculus_pharaonius_01.JPG db-3-5-2.txt query-result-3-5-2-rgb-shell.html 150

Matrix Multiplication

You definitely don't want to write your own matrix-vector arithmetic classes. Perhaps use JAMA or some other third-party library.

Additional Requirements

1. You must submit all your work using GitLab using a project named `cs454`.
2. You must place all code into a package named "hw3".