CMP SCI 4420/5420 **Image Processing and Computer Vision**
Assignment # 2 Due Date: September 23, 2024

Building an Image Corpus

Purpose

In this project, you will start to develop an image corpus. An image corpus is a large collection of images which is created for a specific purpose. For example, an image corpus may be created to evaluate algorithms for a specific image processing task such as classification using supervised or unsupervised machine learning algorithms. For an idea of how people develop a corpus for their research, read the paper on detection of disease paying attention to the data development portions in sections labeled "Method" and "Results". [I want you to just skim this paper and not spend too much time on trying to understand all the nuances in there.]

Even though the minimum requirements of the project should be fulfilled by the due date, I'll expect you to continue building the corpus for the rest of the semester, and use it in other projects.

Task

Images in a corpus share certain characteristics that are same across all the images contained therein. For instance, they will all have the same size (i.e., same number of rows and columns), are either color or gray scale images, are consistently named, and each image is associated with certain meta data, if any – date of acquisition, modality of acquisition, copyright owner, and image annotation in natural language text.

For your project, you will search the Internet for pictures that meet a certain criteria that will be chosen by you. We want to procure pictures that are free of copyright issues. You will note in the metadata the location where you received the picture. You are also free to take pictures yourself and add those to the corpus.

The data for this project are images that are stored in a hierarchically structured directory. Let us call the top-level directory as `dirA` (replace `dirA` with a suitable and meaningful directory name, such as `sports` or `travel`. This directory will contain some images as well as other (sub)directories. Let us call these (sub)directories as `dirA1` and `dirA2`. Each of these subdirectories in turn will contain some images and may contain other (sub)directories. This (sub)directory structure may extend to an arbitrary number of levels.

The images in the directories can be of different sizes as well as different image file types (e.g., tiff, png, jpeg, pbm). I'll suggest that the images be of uniform size but they should not be larger than 640×480 pixels.

Invoking the solution

Your solution will be invoked using the following command:

```
corpus -h -a -g -r numrows -c numcols -t type indir outdir
```

`corpus`	Name of your executable
`a`	If specified, preserve the aspect ratio of the images
`g`	Save the output image as grayscale [default: save as input]
`numrows`	Maximum number of rows in the output image [default: 480]
`numcols`	Maximum number of columns in the output [default: 640]
`type`	Output image type (can be `jpg`, `tif`, `bmp`, or `png` If type is not specified, original file type is retained
`indir`	Input directory
`outdir`	Output directory [default: `indir.corpus`]

The parameters prefixed with - as well as `outdir` are optional. You are free to use long parameter names such as `--rows` for `-r`.

Suggested implementation steps

You will first collect a few images and organize them in the form of a hierarchy in a single directory. This will become your input directory. You will then perform the steps as follows:

1. Parse the command line. You can create your own parser or use the class `CommandLineParser` provided by OpenCV. Each of the optional arguments may have a default value. If the user specifies the option `-h`, print a help message and exit. Otherwise, assign the number of rows and number of columns from user inputs or default values.
2. Processing an image consists of either up-sampling or down-sampling the image so that its size conforms to the specified size. Also, the up-sampled/down-sampled image must be converted to specified image file type while saving. Furthermore, processing may require converting a color image to grayscale/monochrome or binary.
3. Save metadata in the output directory in a file called `metadata`. This file will contain one line of record for each image in the directory. The processing should also preserve any meta-data associated with the original image, except for the image dimensions.
4. Make sure that you catch any exceptions thrown by OpenCV functions. They will help to save you a lot of trouble.

Criteria for Success

You are free to use your own methods to solve the problem. You are also free to device your own mechanism to build a structure for the metadata files. One suggestion will be to use XML to save metadata; of course, in this case the record for each file will be longer than one line. You can look at the XML facilities provided in OpenCV to save data.

You are free to use the affine transform function from your previous assignment to normalize the source images.

Grading

I'll use the following rubric to assess your submission.

1. *Overall submission; 40pts* You will provide at least 25 images curated from different sources, including taking the pictures yourself. The metadata for each image will include the source of the image. Each image should be free of copyright restrictions. You are free to use sources such as Wikimedia Commons or any other source that does not impose any restriction on their use.
2. *Command line parsing; 10pts* Program is able to parse the command line appropriately, assigning defaults as needed; issues help if needed.
3. *Image size standardization; 10pts* Each image should be downsampled/upsampled to the specified size. Make sure to preserve the aspect ratio, if required.
4. *Metadata Description; 10pts* Provide metadata description for each image as requested, including the source of the image.
5. *Code readability; 10pts* The code must be readable, with appropriate comments. Author and date should be identified.
6. *Code reusability; 10pts* The code is structured in the form of functions, in separate files with their own headers, to facilitate reusability.
7. *Exception handling; 5pts* The code should handle exceptions, thrown from within the code or OpenCV.
8. *Meta-files, 5pts* Meta-files such as `README`, `Makefile`, and revision history are provided and are appropriately written.

Submission

Submit an electronic copy of all the sources, `README`, Makefile(s), and results. Create your programs in a directory called *username*.2 where *username* is your login name on delmar. This directory should be located in your `$HOME`. Once you are done with everything, *remove the executables and object files*, and issue the following commands:

```
% cd
% chmod 755 ~
% ~bhatias/bin/handin cs5420 2
% chmod 700 ~
```

Do not copy-paste these commands from the PDF; type in those commands.