Image Browser Project Documentation

Elizabeth Chilcoat, Kelle Clark, Michael Heath

August 29, 2020

1 Usage and System Dependencies

1.1 Dependencies

The user should check to see if the following libraries are installed or do a batch install with

```
pip install -r requirements.txt
```

The libraries and versions used by the team include:

```
filetype v 1.0.7
opencv_python_headless v 4.4.0.40
numpy v 1.19.1
Pillow v 7.2.0
```

If only one or two libraries are missing, they can be added with:

```
pip install filetype\\
pip3 install filetype\\

pip install opencv-contrib-python\\
pip3 install opencv-contrib-python\\

pip install numpy\\
pip3 install numpy\\

pip install pillow\\
pip3 install pillow\\
```

1.2 Usage

To run in the command line:

```
>python image-browser.py [params] dir
```

where optional parameters are:

```
>python image_browser.py -r numrows -c numcols dir
```

and

```
image_browser  name of the executable
numrows  maximum number of rows in the display window
(default: 720)\\
numcols  maximum number of columns in the display window
(default: 1080)
dir  input directory
```

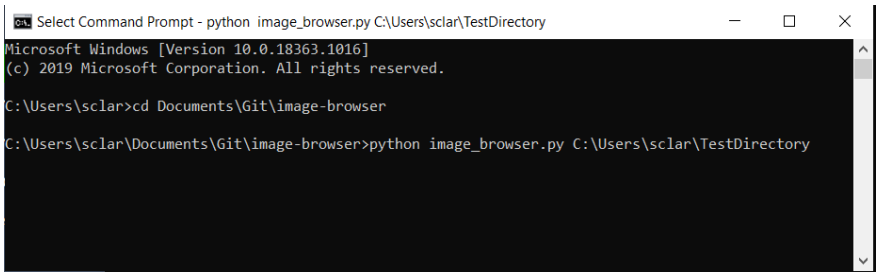

Figure 1: "Command line instructions for running the program"

2 User Requirements Log

The design, implementation and testing of the system are based on satisfying the following stated user requirements.

2.1 Requirements regarding reading all images in given directory/subdirectory:

1. Given a directory, display each picture in the directory as well as in its subdirectories.

2. The system should be able to access all files which are stored in a hierarchically-structured directory tree. Let us call the top-level directory as dirA. This directory will contain some images as well as other (sub)directories. Let us call these (sub)directories as dirA1, dirA2, and so on. 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.

3. The GUI should enable browsing of images which are stored in a hierarchically structured directory tree in depth-first order.

2.2 Requirements regarding execution, input and messages:

1. The system should be invoked using the following command:

```
>python image_browser.py -r numrows -c numcols dir
```

   where

```
image_browser  name of the executable
numrows  maximum number of rows in the display window
(default: 720)\\
numcols  maximum number of columns in the display window
(default: 1080)
dir  input directory
```

2. The system should provide help to the user on the command line with:

```
>python image-browser.py -help
```

   When invoked with -h option, it should display help similar to the following appropriate for MacOS and exit. If running on Windows, the default values need to be modified.

```
Image Browser v1.0
Usage: >python image-browser.py [params] dir

--?, -h, --help, --usage (value:true)
    print this message
--c          , --cols (value:1280)
    Max number of columns on screen
--r          , --rows (value:720)
    Max number of rows on screen
dir
    Directory that contains the pictures to browse
```

The parameters prefixed with the - (dash) are optional. You are free to use long parameter names such as –rows for -r

2.3 Requirements regarding the GUI functionality:

1. Include a simple Graphical User Interface, GUI, for browsing images.
2. The valid inputs from user will be: space bar or n (for next image), p (for previous image), and q (to stop the program).
3. The image information (name, size) should be displayed in the console window.
4. In addition to displaying images, the GUI should also display meta-data associated with the images. Available meta-data may include image file name, file path, image file type, image size (number of rows and columns), number of pixels (number of rows $\times$ number of columns), and image file size in bytes.
5. The system will ensure that the image fits within specified pixel dimensions while preserving aspect ratio for user's OS.
6. The system should make sure that the image completely fits in the display window. Note that this can be achieved by using the Affine transformation function of OpenCV that will preserve the aspect ratio.

2.4 Requirements regarding exception handling:

Any exceptions should be handled by the system, so that the program will always either execute as expected or exit. An error message of why the system exited is optional.

2.5 Requirement to use OpenCV library:

1. The project should demonstrate the team's level of familiar with OpenCV, specifically with the tasks of

(a) reading an image
(b) displaying an image
(c) allowing user interactivity with display window
(d) performing Affine transforms on the image

3 Software Model, Development Methodology

The team used the Iterative Software Engineering model with some features of the Integrate and Configure Model to perform the tasks of specification, development and testing. The team of 3 found it easy to communicate frequently through the Team's channel on Microsoft Team, all members were involved in understanding the stated user requirements, developing the implied system requirements, provided short feedback on progress, and worked together on coding and the architecture of the system through shared files. A workable piece of code was generated with the focus of meeting one function, and each of these sprints were finished in less than a day. These practices are in line with the Agile manifesto, but the team also relied heavily on scripts and libraries found in research to use in the development. For instance the initial requirement that the system be implemented using C++ was modified when the team found that their skill level was not appropriate at the time in this language and the processes involved in running C++ code. Research revealed support for OpenCV in Python, so the team changed the language of the system to Python given that the requirement could be modified to allow for any language except Matlab. The following represents the general architecture of the system which includes both implemented functionality and proposed future functionality:

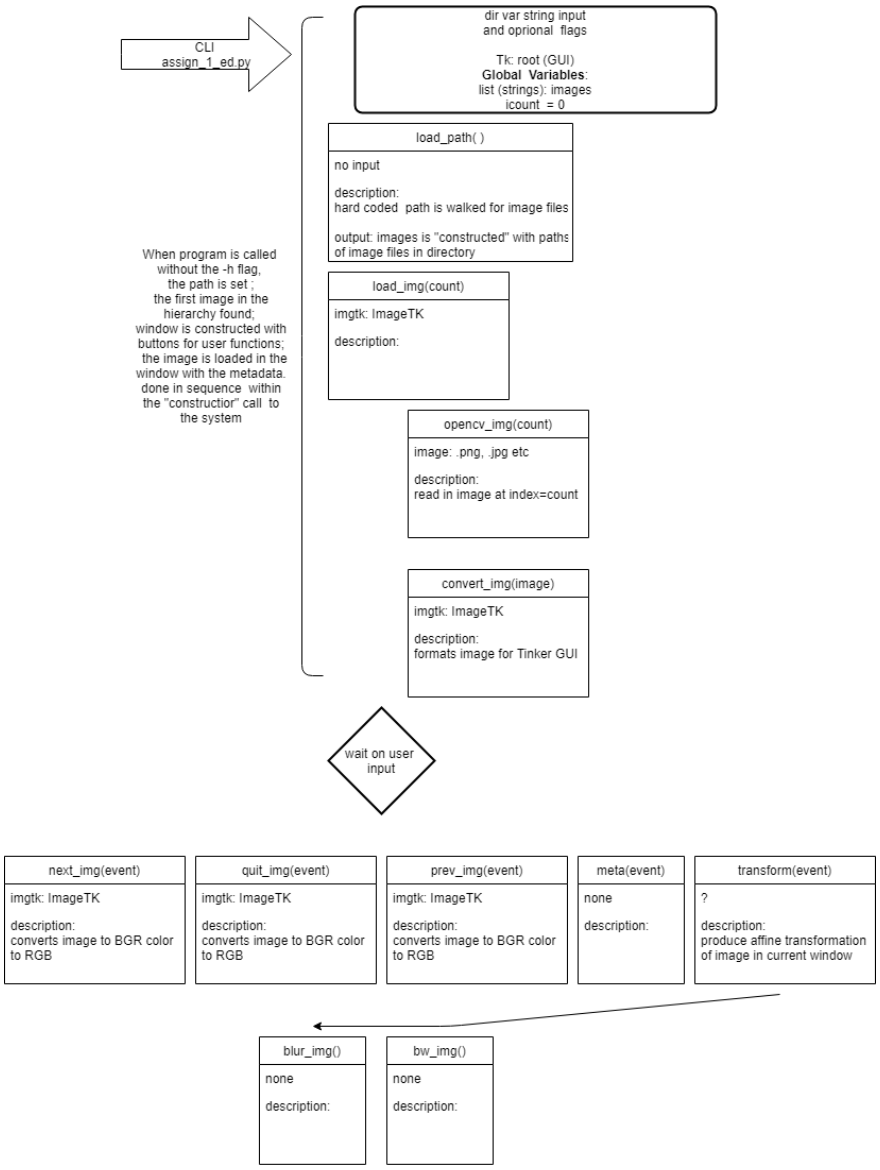

Figure 2: "Command line instructions for running the program"

4 Testing the System

A directory was created for testing that included empty files, files of varying types (HTML, PDF, Word) and images stored in different formats:

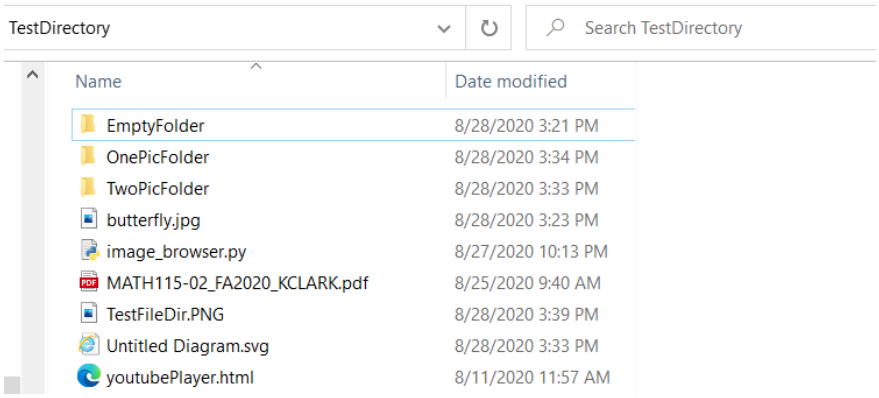

Figure 3: "Command line instructions for running the program"

4.1 Test Objective: Given a directory, does the system read in and display all images in the hierarchy in a top-down traversal?

Test ran

```
python image_browser.py C:\Users\sclar\TestDirectory
```

Test result: Image browser popped up with butterfly.png and traversed into directories and sub-directory in top-down order

4.2 Test Objective: Given a directory, does the system preserve the aspect ration of the users OS and ensure that the image fits inside the window. (Platforms checked: Windows and MacOS X)

Test ran

```
python image_browser.py C:\Users\sclar\TestDirectory
```

Test result: On both the Windows 10 and Mac OS used for testing, the images fit inside the window of the image browser and the aspect ration was preserved. (after finding that it did not and changes were made to the code :))

4.3 Test Objective: Is the metadata for each image displayed in the browser including name and size?

Test ran

```
python image_browser.py C:\Users\sclar\TestDirectory
```

Test result: Metadata is displayed on the split pane of the browser, including file path. A button is included that returns the content relevant to the operating system and the file.

4.4 Test Objective: Determine what happens at the end case where a file is given as dir that is not an image.

Test ran

```
python image_browser.py C:\Users\sclar\TestDirectory\requirements.txt
```

Test result:

```
Invalid path or there are no images in path
```

4.5 Test Objective: Determine what happens if user does not enter in required parameter dir

Test ran

```
python image_browser.py
```

Test result:

```
usage: image_browser.py [-h] [--rows ROWS] [--cols COLS] dir
image_browser.py: error: the following arguments are required: dir
```

4.6 Testing Objective: What is the result of using the optional -h flag?

Test ran

```
C:\Users\sclar\Documents\Git\image-browser>python image_browser.py -h
```

Test result:

```
usage: image_browser.py [-h] [--rows ROWS] [--cols COLS] dir

Image browser v1.0

positional arguments:
  dir           The root directory to view photos in

optional arguments:
  -h, --help    show this help message and exit
  --rows ROWS   Max number of rows on screen (Default is 720)
  --cols COLS   Max number of columns on screen (Default is 1080)
```

4.7 Testing Objective: Is the user able to traverse the images using 'n' for next, 'p' for previous including handling going back to the beginning of the slides when at the end and to the beginning of the slide show when at the end. In addition can the user quit with q?

Test ran

```
python image_browser.py C:\Users\sclar\TestDirectory
```

Test results: Successful traversal from beginning to end using 'n', going backwards with 'p', moving from end to beginning, moving from beginning to end. The application terminates with 'q.'

4.8 Testing Objective: Did the team use OpenCv techniques and Affine transformations in their implementation?

Test ran

```
python image_browser.py C:\Users\sclar\TestDirectory
```

Test results: Affine functionality is included as a button on the browser window. When user selects the button AffineT, the image is transformed to a new image that is "sheared" so that the pixels neighbors are preserved with but the locations are all shifted relative to a new position.

4.9 Testing Objective: Can the user enter in a specified max number of columns and rows with the optional flags?

Test ran

```
python image_browser.py --cols 90 --rows 90 C:\Users\sclar\TestDirectory
```

Test result: Using such a small maximum row and col size (both equal), it was noted that the image was always square and very small...but the entire image stayed in the browser.

5 Reflecting on the learning experience and teamwork

Reflect on your solution to the problem and the learning experience through this project. Trust building, cohesion, and psychological safety are the foundations elements of teamwork. Reflect on the team dynamic experienced in this project.

5.1 Team member contribution/effort assessment

The following rubric is used by the members of the team in order to rate themselves and their teams members on a scale of 1 to 5 about their individual contribution to the project (a rating of 1 being poor and 5 being outstanding). Rationale should be provided for each rating. These documents are seperate.

Table 1: Team Assessment.

	Self-assessment	**Team member 1**	**Team member 2**
attended meetings			
communicated			
participated			
contributed			
provided feedback			
positive attitude			

The rubric for scoring is given in the following image:

	Good	Fair	Poor
Conformance to Specifications (40 points)	The program works correctly and meets all of the specifications. (40 points).	The program works correctly but implements less than 50% of the specifications. (30 points)	The program produces incorrect results or does not compile. (10 points)
Data Structures and Algorithms (20 points)	Appropriate and efficient data structures and algorithms are used. (20 points)	The data structures and algorithms used get the job done but they are neither a natural fit nor efficient. (10 points)	Solution is based on brute force approach. No consideration is given to selection of suitable data structures and algorithms. (5 points)
Testing (20 points)	Coverage of test cases is comprehensive. Following information is provided for test cases: inputs, expected results, pass/fail, and remarks.(20 points)	Coverage of test cases is low. Test case documentation is incomplete. (10 points)	Cursory treatment of testing. No documentation of test cases. (5 points)
Coding (10 points)	The code is compact without sacrificing readability and understandability. (10 points)	The code is fairly compact without sacrificing readability and understandability. (5 points)	The code is brute force and unnecessarily long. (2 points)
Readability (5 points)	The code is well organized and very easy to follow. (5 points)	The code is readable only by someone who knows what it is supposed to be doing. (3 points)	The code is poorly organized and very difficult to read. (1 points)
Documentation (5 points)	The code is self-documenting and obviates the need for elaborate documentation. It is well written and clearly explains what the code is accomplishing and how. (5 points)	The documentation is simply comments embedded in the code with some simple header comments separating functions/methods. (3 points)	The documentation is simply comments embedded in the code and does not help the reader understand the code. (2 points)

Figure 4: "Team member assessment rubric"