ReadME

Author Information

- Name: Qianlai Yang, Katharine Huang
- Email: yang.qianl@northeastern.edu, huang,kat@northeastern.edu

High-Level Design Choices

During the development of this project, several design choices were made to ensure the program's efficiency and maintainability:

MVC Architecture: The Model-View-Controller pattern was chosen to separate concerns, making the codebase more modular and easier to maintain.

Also, in this time update, we add command pattern to our project.

Assumptions

The following assumptions were made during the development:

- The input images will always be in a supported format (e.g., ASCII PPM, PNG, JPG).
- The output images will always be in a supported format (e.g., ASCII PPM, PNG, JPG).
- The source files (image or script, and script) for input and output images file live in “res” folder.
- The command user enters in text, or each line of command in script file that provided is valid a command, otherwise throw “unknown command” message. User can continue typing. Note: if user enter “help”, the program will display commands for view, but it will not execute any feature of processing image.
- Enter ‘quit’ when user wants to quick the program.
- To run script file when jar executes, user enter “java jar… -file” follow script file name. Script file must be in same director of the jar file when running, the test image location too.

Special Features with commands

1. LOAD
    - Command: `load <path> <name>`
    - Description: Load an image from a specified path and assign it a name.
2. SAVE
    - Command: `save <path> <name>`
    - Description: Save the image with the given name to the specified path.
3. COLOR COMPONENT EXTRACTION
    - Red: `red-component <oldImage> <newImage>`
    - Green: `green-component <oldImage> <newImage>`
    - Blue: `blue-component <oldImage> <newImage>`
    - Description: Extract the respective color component of the image.
4. VALUE COMPONENT
    - Command: `value-component <oldImage> <newImage>`
    - Description: Extract the value component of the image.
5. INTENSITY COMPONENT
    - Command: `intensity-component <oldImage> <newImage>`
    - Description: Extract the intensity component of the image.
6. LUMA COMPONENT
    - Command: `luma-component <oldImage> <newImage>`
    - Description: Extract the luma component of the image.
7. IMAGE FLIPPING
    - Horizontal: `horizontal-flip <oldImage> <newImage>`
    - Vertical: `vertical-flip <oldImage> <newImage>`
    - Description: Flip the image horizontally or vertically.
8. BRIGHTEN/DARKEN

- Brighten: `brighten <value> <oldImage> <newImage>`
- Darken: `darken <num> <oldImage> <newImage>`
- Description: Brighten or darken the image by the given value.

9. RGB SPLIT AND COMBINE
    - Split: `rgb-split <oldImage> <imageR> <imageG> <imageB>`
    - Combine: `rgb-combine <imageR> <imageG> <imageB> <newImage>`
    - Description: Split the image into its RGB components or combine them into a single image.
10. BLUR/SHARPEN
    - Blur: `blur <oldImage> <newImage>`
    - Sharpen: `sharpen <oldImage> <newImage>`
    - Description: Apply a blur or sharpen effect to the image.
11. SEPIA
    - Command: `sepia <oldImage> <newImage>`
    - Description: Convert the image to sepia tone.
12. RUN SCRIPT
    - Command: `run <scriptFilePath> <image key>`
    - Description: Execute a script containing a sequence of commands.
13. HELP
    - Command: `help`
    - Description: Display a help message.
14. QUIT
    - Command: `quit`
    - Description: Quit the application.
15. LEVEL ADJUST SPLIT

- Command: `level-adjust-split <num b> <num m> <num w> <oldImage> <newImage> <num split>`
- Description: Level adjust split of an image.

16. HISTOGRAM
    - Command: `histogram <oldImage key> <newImage>`
    - Description: Generate a histogram of the image.
17. COLOR CORRECT
    - Command: `color-correct <oldImage> <newImage>`
    - Description: Correct the coloring of an image.
18. COMPRESSION
    - Command: `compression <num> <oldImage> <newImage>`
    - Description: Apply compression to the image.
19. COLOR CORRECT SPLIT
    - Command: `color-correct-split <oldImage> <newImage> <num>`
    - Description: Apply color correction with a split effect.
20. LEVEL ADJUST
    - Command: `level-adjust <num b> <num m> <num w> <oldImage> <newImage>`
    - Description: Adjust the level of an image.
21. BLUR SPLIT
    - Command: `blur-split <oldImage> <newImage> <num>`
    - Description: Apply a blur effect with a split.
22. SHARPEN SPLIT
    - Command: `sharpen-split <oldImage> <newImage> <num>`
    - Description: Apply a sharpen effect with a split.
23. SEPIA SPLIT
    - Command: `sepia-split <oldImage> <newImage> <num>`

- Description: Convert the image to sepia tone with a split effect.

24. GREYSCALE SPLIT
    - Command: `greyscale-split <oldImage> <newImage> <num>`
    - Description: Convert the image to greyscale with a split effect.

UML Diagrams

Represents the structure of the system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

- UML diagrams for each package are in "res" folder : ConseloViewpackageUML.png, ControllerpackageUML.png,modelpackageUML.png and MVC.jpg
- The MVC contain the out look for our MVC structure, and rest UML contains the detail of function

Instructions for Running the Program

1. What to expect when before start?
   Execute “Main.java”.
2. What to expect when started?
   User is prompt for entering text.
   User must type input in lower-case.
   User must type in integer for some functions that need a number of degree in processing image.
   User can type 'help' to see the list of available commands.
   User can enter 'quit' to quick the program.
   Users must enter a valid command to start.
   User must assign a keyValue to image when try running with script file or display histogram.
   User can follow detailed guide from <Special Features with commands> section for command.
3. What to do when finished typing command?
   Press key to run the program after an valid command.
   The program will execute based on an valid command, then process the image, and output the image in “res” folder.

Brief Explain for Class And Interface

Model

FilePathHandle.class

This class helps us to find the file in the res direction.

`public static String getAbsolutePath(String filename)`

This method is change the user given filename to a absolute path to make sure our program will working in any environment.

ImageInterface.class

This is a interface that represents a image object. The image shall have a Pixel array which contain all pixel values and the calculation method.

`void setImage(Pixel[][] image)`

This method sets a 2D array of Pixel objects which contains all pixels in this image.

`Pixel[][] getImage() ;`

This method returns a 2D array of Pixel objects.

`int getWidth() ;`

This method is to get the width of the image.

`void setWidth(int width)`

This method sets the width for the image.

`int getHeight()`

This method is to get the height of the image.

`void SetHeight(int height)`

This method sets the height for the image.

`int[][] getValue()`

This method calculates the value component(the maximum value of the three components for each pixel) for each pixel.

`int[][] getIntensity()`

This method calculates the intensity component(the average of the three components for each pixel.) for each pixel.

`int[][] getLuma()`

This method calculates the value component(• the weighted sum 0.2126r+0.7152g+0.0722b) for each pixel.

Image.class

This class is implements by ImageInterface.class. In this method, we create a private method to help us calculate the value of value component, intensity component and luma component.

`private int[][] caluValue(String func)`

This method is help use calculate the value of value component, intensity component and luma component. And we will based on the different func the parameter send to us to imply different function.

ImageLoaderSaver.class

This class is an interface class which help us to load the image and save them.

`Image readPPM(String filename);`

This method is help us to read ppm image file.

`Image readPNGAndJPG(String filename);`

This method is help us to read png and jpg file.

`void saveImage(Image image, String path, String type) throws IOException`

This method is help us to save the image to the user-given path and user-given type.

ImageUtil.class

This class is a class implements the ImageLoaderSaver interface.

Pixel. class

This is an object class which represents the pixel's RGB which is r(red),g(green) and b(blue).

And we set the getter and setter method in there.

ImageManipulator.class

This is a class contain the method we shall use to modify the image.

`public Image loadImagePath(String file) throws IllegalArgumentException`

This method loads the image by the given path, and returns an Image object to the controller. If the path is illegal, it will throw an exception.

```
public void saveImageByPath(String path , String type , Image img) throws IOException
```

This method saves the image by the given path and given type, If the type or path is illegal, it will throw an exception.

```
public Image filpHorizontally(Image image)
```

This method is filp the given image horizontally and return the new image after flip.

```
public Image filpVertically(Image image)
```

This method is filp the given image vertically, and returns the new image after flip.

```
public Image brighten(Image image , int num)
```

This method makes the image brighten based on the number and returns the new image after modification.

```
public Image greenComponent(Image image)
```

This method gets the green component from this image and return the new image.

```
public Image blueComponent(Image image)
```

This method gets the blue component from this image and returns the new image.

```
public Image redComponent(Image image)
```

This method gets the red component from this image and return the new image.

```
private Image colorComponent(Image image , String type)
```

This method is a helper method used to get the color component based on the type we pass. The purpose of this one is because we need reduce the duplicate code.

```
private Image helperOfLight(Image image , int num , boolean flag)
```

This method is a helper method used to make the image bright or dark,if the flag is true, make the image bright by the number, or else make it dark.

```
public Image darken(Image image , int num)
```

This method makes the image darken based on the number and returns the new image after modification.

```
public Image[] splitToThreeColorImages(Image image)
```

This method splits the image into three color components (red, green, and blue). And return an array that contains the three images to an Image array.

```
public Image combineToOneImage(Image r , Image g , Image b)
```

This method combines three color components (red, green, blue) into the image. And return the new image after combining.

`public Image valueComponent(Image image)`

This method is get the valueComponent based on what we calculate in Image class. And return the new image.

`public Image intensityComponent(Image image)`

This method is get the intensityComponent based on what we calculate in Image class. And return the new image.

`public Image lumaComponent(Image image)`

This method is to get the lumaComponent based on what we calculated in Image class. And return the new image.

`public Image blur(Image image)`

This method makes the image blur based on the kernel 3*3 matrix. And return the new image.

`public Image sharpen(Image image)`

This method makes the image sharpen based on the kernel 5*5 matrix. And return the new image.

`private Image filtering(double[][] kernel , Image image)`

This method is a helper method which based on the kernel matrix we have to filtering the image, and return the new image.

`public Image convertSepia(Image image)`

This method make the image grey by a sepia tone, and returns the new image.

`private double getPercentage(double[][] red , double[][] green , double[][] blue , int percentage)`

This method is get the limit value based on the percentage of compression.

`public Image imageSplit(Image image , int percentage , String str)`

This method is help to split image.We can apply different method and split the image to two part which help us to compare them.

`public Image levelAdjustmentSplit(Image image , int percentage , String str , int b , int m , int w)`

The method is help to the split the image with level adjustment.

`private void split(Image image , Image left , Image right)`

This method is a helper method for split the method to two part.

`public Image compression(int percentage , Image image)`

This method is compression the image to small size version.

`private double[][] compressionCal(double[][] imageValue)`

This method is a help method to calculation the conpression.

`private double[][] reverseCal(double[][] arrayOfComp , int width , int height , double limit)`

This method is a help method to calculate the reverse.

`private double[][] haar(double[][] array , int length)`

This method is a help method to calculate by haar.

`private double[][] invhaar(double[][] array , int length)`

This method is a help method to calculate by inverse haar.

`public double[] transform(double[] array , int len)`

This method is a help method is help inverse haar.

`public double[] inverse(double[] array , int len)`

This method is inverse the compression array.

`private Image imageCombine(Image left , Image right) throws IllegalStateException`

This method is combine the two image object to one image.

`private int getLeftWidthForSplit(int width , int percent)`

This method is a help method for get left image width

`public Image levelAdjustment(Image image , int b , int m , int w)`

This method is make the image to level adjustment.

`private int valueCal(int b , int m , int w , int v)`

This method is a help method for convert the b,m,w to fit the function.

`public int[][] generateData(Image image)`

This method generates the image to the histogram value.

`public Image correction(Image image)`

This method is make the image to color correction.

Controller

`ImageControllerInterface.class`

This is an interface for the image controller, we will use this to control the program base on the message we get from the user or script.

`void load(String path , String name):`

this method is to load the image and set the name to this image to help us store it.

`void save(String path , String name) throws IOException ;`

this method is help us save the image to a suitable format. We will based on the name find the image object we store.

`public void redComponent(String oldImage , String newImage)`

This method is get the red component from the image , and we will set new name for the result we get based on the user set.

`public void greenComponent(String oldImage , String newImage)`

This method is get the green component from the image , and we will set new name for the result we get based on the user set.

`public void blueComponent(String oldImage , String newImage)`

This method is get the blue component from the image , and we will set new name for the result we get based on the user set.

`public void valueComponent(String oldImage , String newImage)`

This method is get the value component (the maximum value of the three components for each pixel) from the image , and we will set new name for the result we get based on the user set.

`public void intensityComponent(String oldImage , String newImage)`

This method is get the intensity component (the average of the three components for each pixel.) from the image , and we will set new name for the result we get based on the user set.

`public void lumaComponent(String oldImage , String newImage)`

This method is to get the luma component (the weighted sum 0.2126r+0.7152g+0.0722b) from the image, and we will set a new name for the result we get based on the user set.

`public void filpHorizontally(String oldImage , String newImage)`

This method is to filp image to horizontally. And save the new image with the name user set.

`public void filpVertically(String oldImage , String newImage)`

This method is to filp image to vertically. And save the new image with the name user set.

`public void brighten(int num , String oldImage , String newImage)`

This method is to make image brighten by given number. And save the new image with the name user set.

`public void rgbSplit(String oldImage , String imageR , String imageG , String imageB)`

This method is to split image to the red, green,blue component image. And save the new image with the name user set.

`public void rgbCombine(String imageR , String imageG , String imageB , String newImage)`

This method is to combine the red, green,blue component image to a new image. And save the new image with the name user set.

`public void darken(int num , String oldImage , String newImage)`

This method is to make the image darken by give num. And save the new image with the name user set.

`public void blur(String oldImage , String newImage)`

This method is to make the image darken. And save the new image with the name user set.

`public void sharpen(String oldImage , String newImage)`

This method is to make the image darken. And save the new image with the name user set.

`public void convertSepia(String oldImage , String newImage)`

This method is to make the image grey tone. by sepia And save the new image with the name user set.

`void compression(int percentage , String oldImage , String newImage) ;`

This method is to compression image.

`void blurSplit(String oldImage , String newImage , int split) ;`

This method is to make the image split.And make the left part blur,and combine it together. And save the new image with the name user set.

`void sharpenSplit(String oldImage , String newImage , int split) ;`

This method is to make the image split. And make the left part sharpen,and combine it together. And save the new image with the name user set.

`void sepiaSplit(String oldImage, String newImage, int split);`

This method is to make the image split. Make the left part sepia,and combine it together. And save the new image with the name user set.

`void greyscaleSplit(String oldImage, String newImage, int split);`

This method is to make the image split. Make the left part grey scale, and combine it together. And save the new image with the name user set.

`void levelAdj(int b, int m, int w, String oldImage, String newImage);`

This method is to make the image modify by level adjustment. And save the new image with the name user set.

`void histogram(String oldImage, String newImage) throws IOException;`

This method is create histogram for image.

`void colorCorrect(String oldImage, String newImage);`

This method is to make the image modify by color correction. And save the new image with the name user set.

`public void runScript(String scriptFilePath) throws IOException`

This method will call by view method which run the command script to execute our program.

`void levelAdjSplit(int b, int m, int w, String oldImage, String newImage, int split);`

This method is to make the image split. Make the left part level adjustment, and combine it together. And save the new image with the name user set.

`void colorCorrectSplit(String oldImage, String newImage, int split);`

This method is to make the image split. Make the left part color correction, and combine it together. And save the new image with the name user set.

ImageController.class

This class is implements by ImageControllerInterface. It will fill the method we create in the interface. And we create a map<String, Image> to store the image we create, the key is name of the image which is set by user, and the image is the Image object modify by our program.

Command.class

This class helps us to split the command from the user by two part: action and argument. we will save the argument as a list which we can pass to the controller easier.

`public String getAction()`

return an action string after split

`public List<String> getArguments()`

return an argument string after split

ICommand interface

This interface is a commend interface, in this interface, we contain the execute method which to execute the fit method.

All Other classes in controller

Based on what we learned from the preview class, we apply the command pattern to this class. We create some command class for the main controller. All other class is implement from ICommand.

View

ConsoleView.class

This class is help us to read commands from user. And we implement AutoCloseable.class to help use handle BufferReader. In the constructor, we will figure out is the user writes the command or let the program run the script.

`public void setImageController(ImageControllerInterface controller)`

this method is set for the controller interface.

`public Command readCommand() throws IOException`

This method helps us to read commands and if the command is illegal, we will throw an exception.

`public void displayHelp()`

This method shows the help menus for the user once they type the “help”

`public void determineAction(String action, List<String> args)`

This is the method to get the user’s command. Based on what they give to us, we will call different methods from the controller to modify the image.

`public void executeScript (String scriptFilePath)`

This method will help us execute the script which the user provide.

`public void displayMessage(String message)`

This method will show the message after operation.

`public void close() throws IOException`

This method will let the user stop the program.

DisplayHistogram.java

This class is help us to save the histogram.

`public BufferedImage createHistogramImage(int[][] data)`

Creates a histogram image from the given RGB data.

`private void initializePath(Path2D redPath , Path2D greenPath , Path2D bluePath)`

Initializes the paths for the RGB histogram lines.

`private int findMaxValue(int[][] data)`

Finds the maximum value in the RGB data.

`private void drawBackgroundAndGrid(Graphics2D g2d)`

Draws the background and grid for the histogram.

SwingFeaturesFrame:

this method is create the GUI for the view. In the panel, we have the image view.At the controller panel, we can add the file, save the file, percentage of split and the modify button

`public SwingFeaturesFrame(ImageController controller)`

This class is the constructor, in this constructor , we can create the panel which contain all the component in the page.

`public void actionPerformed(ActionEvent arg0)`

This class is the override the method which based on action command modify the panel.

`private void modifyMethod()`

This method is a helper method, which help panel do the modify based on the selection we made.

`private void updateImage()`

this is the helper method for update the image panel.

`public JLabel getFileOpenDisplay()`

`public JLabel getFileSaveDisplay()`

`public JLabel getOptionDisplay()`

`public JLabel getItemStateLabel()`

These method is getter method to help us to do the GUI test.

ModelHandler package

ModelHandler.java

This class is handle which mode to use.

`public void startApplication(String[] args)`

A class handle which mode to start.

`private void initializeGuiMode(ImageManipulator model , SwingFeaturesFrame guiView)`

Helper class handle when each mode start.

`private void initializeCommandLineMode(ImageManipulator model , ConsoleView consoleView)`

Helper class handle when each mode start.

`private void initializeScriptMode(ImageManipulator model , ConsoleView consoleView , String scriptPath)`

Helper class handle when each mode start.

`private void setLookAndFeel()`

Helper class handle when each mode start with GUI

`private void handleInvalidArguments()`

Helper class handle when error in mode start.

`public boolean isGuiInitialized()`

Helper checks if the GUI has been initialized.

`public boolean isCommandLineInitialized()`

Helper checks if the Command mode has been initialized.

`public boolean isScriptModeInitialized()`

Helper checks if the running script mode has been initialized.

`public boolean isInvalidArgumentsHandled()`

Helper check if the error opening any mode.

Citiation:

This picture we use is token by Katharine Huang which is one of our teammate.

And the manhattan and koala image is provide from professor.