Curtin University — Computing Discipline

Programming Design and Implementation (COMP1007)

Semester 1, 2020

Assignment

Due: Sunday, 31st May 2020 at 5pm (local time)

Weight: 30% of the unit mark.

1 Introduction

This assignment builds upon your previous worksheets. These worksheets follow along with your current understanding of the concepts covered in the unit thus far and will develop on techniques, discussed in the lectures, each week. The practical worksheets are designed to encourage the application of your knowledge to a realistic context. Completion of the assignment task in each worksheet is vital to achieving a high mark in the assignment, as they are weighted against this assessment.

Note: This assignment will take significant time to implement. You cannot finish it in a single night or weekend. You will need to plan your time carefully and work on each part as the topic is covered.

You are working on a program to detect horizontal and vertical lines in a PNG (Portable Network Graphic). Your program will need to do the following:

- Be able to open both a `.png` and `.csv` file and retrieve a 2D array of greyscale values from it.
- Write the values from that array to a file, in `.csv` or `.png` format.
- Prompt the user to use an Image using values from User Input.
- Prompt the user to use a Kernel using values from User Input.
- Perform Convolutions based on the values within the array to detect lines.
- Perform Smoothing on a $s \times s$ area of pixels based on $\mathtt{x},\mathtt{y}$ coordinates.

Note: This assignment is subject to updates and clarifications. In the case that it does get updated, an announcement will go out and you will be required to adhere to the updated specification. It it your responsibility to monitor your email/announcements for updates.

2 Task Details

This Assignment specification is the overall guideline of how your assignment must function. The design choices as to how many model classes you need to write/use is left up to you, we recommend the minimum of what we taught you in the workshop. Each worksheet will take concepts within this document, and go into depth on how it must be developed. The **final** functionality of the assignment is as follows:

2.1 Data Validation

- The file that contains a Kernel/Mask[1] must be as follows:
  - The file must exist.
  - It must **not** throw an exception when read in.
  - Each element must be an Integer.
  - The size must be a square. If the height is k, the width must also be k. Therefore a $(k \times k)$ matrix.
- The file that contains an Image[2] must be as follows:
  - If it is a `.png`:
    - The file must exist.
    - It must **not** throw an exception when read in.
  - If it is a `.csv`:
    - The file must exist.
    - It must **not** throw an exception when read in.
    - Each element must be an Integer.
    - Each element must be $0 <= x <= 255$.
    - The size must be a rectangle. If the height is n, each width must be m. Therefore a $(n \times m)$ matrix.
- User Input for both the Image and the Kernel is the same as their respective `.csv` file reading validations.
- Smoothing:
  - The size (s) must be **odd** and no greater than the shortest side of the image.
  - The coordinates must be 1-index based.
  - The factor must be a Real.
  - The factor must be $0 < x <= 1$.

[1] A Kernel/Filter is the matrix that the file will be convoluted with. If an $(n \times m)$ matrix is convoluted with a (3×3) Kernel, the resulting matrix will be $((n-3+1) \times (m-3+1))$. Similarly with a $(k \times k)$ Kernel the resulting matrix will be an $((n-k+1) \times (m-k+1))$

[2] Either as a `.png` or `.csv`

2.2 Import Image

This option should provide a sub menu to the user, offering the option for either File (`.csv` or `.png`) or from User Input.

All other menu items should not be usable until an image has been added to your program.

2.2.1 File

This option of your menu will allow the user to import an image into your program, via File I/O. The imported file can consist of either `.png` or `.csv`. See Data Validation for the restrictions on these file types.

Refer to Worksheet 8 (File IO) for how to read in a `.png` image.

You are expected to handle all errors in your File reading, and display an understandable error to the user.

2.2.2 User Input

This option of your menu will first prompt the user for the **`x`** value of the image, then the **`y`** value. Then proceed to prompt the user for each coordinate on a separate line. For example:

```
Please enter in the x-dimension: 2
Please enter in the y-dimension: 3
These Coordinates are (x,y)
(1,1): 143
(1,2): 113
(1,3): 37
(2,1): 43
(2,2): 56
(2,3): 69
```

```
      0     1
  -------------
0 | 143 |  43 |
  -------------
1 | 113 |  56 |
  -------------
2 |  37 |  69 |
  -------------
```

You may swap the prompt order of **`x`** and **`y`** around if you wish, just ensure that you keep to the convention of `(x,y)`.

Note: Any time you need to call to the user to do or display something, that should be handled in the **`UserInterface`** class. You may make calls to this class to request user input and display user output from the methods within your program. Just remember that you should not be Inputting/Outputting from within model classes.

2.3 Import Kernel

This option should provide a sub menu to the user, offering the option for either File (`.csv`) or from User Input.

2.3.1 File

This option of your menu will allow the user to import a kernel into your program, via File I/O. The imported file must be `.csv`. See Data Validation for the restrictions on this file type.

You are expected to handle all errors in your File reading, and display an understandable error to the user.

2.3.2 User Input

This option of your menu will prompt the user for the **k** size of the kernel. Then proceed to prompt the user for each coordinate on a seperate line. For example:

```
Please enter in the kernel size k: 3
These Coordinates are (x,y)
(1,1): 1
(1,2): 2
(1,3): 1
(2,1): 0
(2,2): 0
(2,3): 0
(3,1): -1
(3,2): -2
(3,3): -1
```

```
     0    1    2
   ----------------
0 |  1 |  0 | -1 |
   ----------------
1 |  2 |  0 | -2 |
   ----------------
2 |  1 |  0 | -1 |
   ----------------
```

You may swap the prompt order of **x** and **y** around if you wish, just ensure that you keep to the convention of `(x,y)`.

2.4 Convolution

This option of your menu will use the image currently stored in the program and attempt to convolute it with the kernel currently stored. If either one is not currently stored an error should be displayed to the user suggesting to import it.

The 2D convolution is a fairly simple operation at heart: you start with a kernel, which is simply a small matrix of weights. This kernel "slides" over the 2D input data, performing an elementwise multiplication with the part of the input it is currently on, and then summing up the results into a single output pixel.

It would allow us to detect the transition from a "light" side (some high value, representing colour) to a "dark" side (some low value, representing colour). Therefore showing us a line or edge in the resulting image.

If you want to see the process on a (5 x 5) Matrix with a (3 x 3) Kernel, converted to a (3 x 3) Matrix, this graphic is an easy way to understand how this Convolution operation would work.

Below is an exaggerated example of how this can detect a line. The left of the **`originalImage`** is considered very bright (a high value) and the right is dark (a low value). Using a vertically biased Kernel (**`Kernel_VERTICAL`**), you are able to produce a line in the **`convolutedImage`**.

`originalImage`

	0	1	2	3	4	5
0	10	10	10	0	0	0
1	10	10	10	0	0	0
2	10	10	10	0	0	0
3	10	10	10	0	0	0
4	10	10	10	0	0	0
5	10	10	10	0	0	0

(6 x 6)

$*$

`Kernel_VERTICAL`

	0	1	2
0	1	0	-1
1	1	0	-1
2	1	0	-1

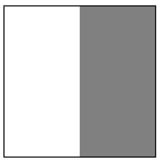
(3 x 3)

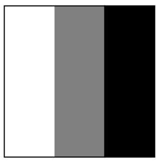
$=$

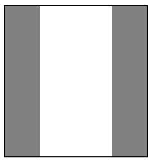
`resultArray`

	0	1	2	3
0	0	30	30	0
1	0	30	30	0
2	0	30	30	0
3	0	30	30	0

(4 x 4)







Your task is to take a Kernel (k x k) and perform a convolution operation on a 2D Matrix (n x m) to produce a $(n-k+1)$ x $(m-k+1)$ sized Matrix. The calculation for element $(0,0)$ if the Kernel is a (3 x 3) Matrix (K) and the image is a (n x m) Matrix (C) in the array would be:

$$(C_{0,0} * K_{0,0}) + (C_{0,1} * K_{0,1}) + (C_{0,2} * K_{0,2})$$
$$+(C_{1,0} * K_{1,0}) + (C_{1,1} * K_{1,1}) + (C_{1,2} * K_{1,2})$$
$$+(C_{2,0} * K_{2,0}) + (C_{2,1} * K_{2,1}) + (C_{2,2} * K_{2,2})$$

The general expression of a convolution is:

$$g(n,m) = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1}(x_{(n+i)(m+j)} * \omega_{(i)(j)})$$

where $x(n,m)$ is the original image, $\omega(k,k)$ is the filter kernel and $g((n-k+1),(m-k+1))$ is the filtered image.

Think about how we might use loops and submodules to calculate this. Make sure you are using a Modular approach, where each task that is needed is broken up into its sub-components

2.5 Export Image

This option should provide a series of prompts to the user to write the currently stored image to a file. See File Naming Convention for how to format the name of the output file.

The user will then choose between the option for writing the image as a **.png** or a **.csv**. Each should be as follows:

- For the process of writing a **.png** file, refer to the `writePNG()` method in Worksheet 8.
- For the process of writing a **.csv** file, refer to the `writeCSV()` method written in Worksheet 8.

2.5.1 File Naming Convention

Whenever a file is written, you are required to ask the user for an input for the date that they wish to save with. This date <u>**must**</u> be in the format **DDMMYYYY** and you are required to split this date up into its sub-components and save the filename as "**<Year>-<Month>-<Day>_Processed_<OrignalFileName>.<FileExtension>**". You are required to use the same method as was taught in the workshops (e.g., **DIV** and **MOD**). If there was no **OriginalFileName** then you must prompt the user for a filename to save, and use that in place of the **<OrignalFileName>**.

For example, if this was a manual input of an image:

```
Please enter the File Name: MyImage
What filetype would you like to save with? (C)SV or (P)NG: p
Please enter a Date to save with: 20052020
File (2020-05-20_Processed_MyImage.png)
```

2.6 Smoothing

Note: This is not the standard way of applying image smoothing, this is more of a proof of concept.

This option of your menu will prompt the user to import an X and a Y value, then prompt for a smoothing value (See Data Validation) and perform a smoothing operation on the Image that is currently stored in your program. The smoothing operation consists of averaging out the elements of the $s \times s$ area around a given coordinate, and multiplying it by the smoothing factor.

```
Please enter a smoothing surface: 9
Please enter a pixel to smooth (x,y): 9,8
Please enter a smoothness factor: 0.763
```

Image

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(15 x 13)

Each of the cells shown in purple (including the original red one) will need to equal the `ceil`'d average of all $s \times s$ cells multiplied by the smoothing factor.

Note: These cells are all zeroes to illustrate the concept. In most cases you will have semi-random integer values in your image that will need this performed on them.

Ensure that you are checking your boundaries, if the user has entered a value too close to the edge of the image where the 9x9 would overlap the edge, then you must display an error.

3 General Housekeeping

3.1 User Input

All user input must be validated where appropriate, refer to Section 2.1 for validation of class-fields. All user input must be handled in the `UserInterface` class, unless otherwise stated.

3.2 User Output

All output to the user should be formatted in a pleasant, easy to read manner. This includes prompts for input. When outputting real numbers they should be formatted to 2 decimal places. All user output must be handled in the `UserInterface` class, unless otherwise stated.

Note: Only round to 2 decimal places at the output stage, so that we can maintain precision in our real numbers.

3.3 Math

All mathematic functions required must use your own made `PDIMath` Class, unless otherwise stated.

3.4 Error Handling

Under no circumstance should the user be aware of exception handling, this includes having un-handled exceptions, any output should be meaningful to a user with no coding experience.

All error messages displayed to the user should be meaningful to a non-programmer.

4 Documentation

You must thoroughly document your code using block comments (`/*...*/`) and inline comments (`//...`). For each Class, you must place a comment at the top of the file containing your name, the file name, the purpose of the class and any modification history. For each method you write you must place a comment immediately above it explaining its purpose, it's import and export values, and how it works. Other comments must be included as necessary. Refer to the Java Coding Standard document on Blackboard for more information on appropriate documentation.

4.1 Report

Your submission must include a two (2) page report. In the report you must cover the following:

- Justify your approach to data validation.
- Justify your design decisions in regards to what functionality was placed in the classes you decided to write.
- Explain why we separate our program into different static classes and model objects.
- Discuss the real world implications of using this algorithm. You don't have to go into full depth about this, just explain why we may need to detect/view edges in an image.
- Discuss any challenges you had in your implementation/design.

Note: The report must detail the *why* of your decisions not the *what*.

So what is the difference?
(This is an **example** of the differences between the what and why.)

What (not worth any marks): My menu has options for <x>, <y>, and <z>. For option <x> there is also a sub-menu. All my menus use integers for the selection.

Why (worth marks): I decided to implement a sub-menu for <x> so I could initially provide a simple menu to the user. My menu structures use a do-while loop as they must always be executed at least once, even if the first menu option is the exit option, I decided not to use a separate validation submodule for the menu integer and instead rely on the default part of the case to provide my error messages. This approach means I did not need to repeat the validation logic in both the individual cases and in the validation module, if I need to add more menu options I would only need to make a modification in one place. Initially, I tried to use characters for my menus as they are easier for the user to understand/remember but then I realised that several options would need to have the same character. I did not consider using strings as they are prone to user error, even though they would remove the necessary exception handling.

Warning: Copying the above text into your report is classed as Academic Misconduct and will be investigated by your Unit Coordinator.

If you do this, you will receive no marks for the comment as you have not done the required work.

5 Submission Requirements

Warning: Electronic copies of all assignment documents must be kept in your Curtin user accounts, and not edited after the due date as the assignment will be tested in your tutorial sessions. These should be in a directory called **`PDIAssignment`**, located immediately under your home directory. Failure to do this will result in 0 marks being allocated to the testing portion of the marking.

Submit your assignment electronically, via Blackboard (OASIS -> My Studies), before the deadline.

Your assignment should be submitted as a single .tar.gz file containing:

A declaration of originality – whether scanned or photographed or filled-in electronically, but in any case *complete*.

Your implementation – including all your pseudo code, java code, readme, etc . . .
Do not submit your .class files

Your report – a single PDF report, as outlined above. The specified length of the report is a guideline! If your report fails to address and explain any of the specified topics you will be penalised (regardless of length).

Note: do not use .zip, .zipx, .rar, .7z, etc, they will be taken as **non-submissions** and instantly receive zero (0) for the assignment.

You are responsible for ensuring that your submission is correct and not corrupted. You may make multiple submissions, but only your latest submission will be marked. You MUST download your submission, extract the contents and ensure it still compiles/runs. If your submission is corrupted you will receive a mark of **zero (0)** with no chance of resubmission.

Please note that assessments submitted in the Discipline of Computing must comply with the Discipline of Computing Coding Standard, which can also be found on Blackboard. All code must be fully documented.

Please ensure you read through the Coding and Academic Integrity document on Blackboard before you start your assignment to ensure that you adhere to it.

6 Class Requirements

You are required to make good use of Model Classes as well as static classes. You should have a minimum of: (**`DetectEdges`**, **`Menu`**, **`FileIO`**, **`UserInterface`**, **`PDIMath`**) for Static classes.

The Model classes are left ambiguous so that we can see your design practices. You will need to use at least 2 Model classes (maybe more).

7 Extensions & Late Assessments

No extensions will be granted. If exceptional circumstances prevent you from submitting an assignment on time, contact the Unit Coordinator ASAP with relevant documentation. After the due date and time is too late.

If you submit your assignment after the specified due date/time you will lose all of the marks for the demonstration. However, you will still have to demonstrate your assignment and successfully complete the question and answer portion in order to receive any marks for the submission. Anything after the due date is considered a late submission, even if only a minute late.

8 Demonstration

You will be required to demonstrate each of the worksheet submissions in your practical session. The final assignment submission will also be demonstrated in the practical session in the week starting the 1st June 2020. If you cannot attend your registered practical session you must make arrangements with the lecturer by Monday the 25th May 2020. You may be asked several questions about your assignment in this demonstration.

Failure to demonstrate the assignment during your registered practical will result in a mark of ZERO (0) for the entire assignment. During the demonstration you will be required to answer questions about your design. If you cannot answer the questions satisfactorily, you will receive a mark of zero for the assignment.

9 Assignment Worksheets and Marking

The pseudocode algorithm and Java implementation will both be marked. Your java code will be assessed by its suitability as a valid implementation of your supplied pseudocode. If you do not supply pseudocode, your Java will **not** be marked.

Each of the assignment tasks should have been signed off in their respective weeks and will have given you feedback on how your design is going. Any feedback provided in these weekly sign-offs should be noted and applied to your assignment before its final submission. In the final sign-off you will only be demonstrating the functionality, your lecturer/tutor will mark your assignment outside of class.

Marks will be allocated for the appropriate choices of static classes / model objects, the overall functionality, and adhering to best programming practices as discussed in the lectures and practicals.

10 Misconduct – Plagiarism and Collusion

You must read the Academic Integrity In Coding pdf and abide by it or you will risk misconduct.

This is an assessable task, and as such there are strict rules. You must not ask for or accept help from *anyone* else on completing the tasks. You must not show your work to another student enrolled in this unit who might gain unfair advantage from it.

These things are considered **plagiarism** or **collusion**.

Staff can provide assistance in helping you understand the unit material in general, but nobody should help you solve the specific problems posed in this document. The purpose of the assignment is for *you* to solve them *on your own*.

Please see both the Academic Integrity In Coding pdf (attached to the assignment specification on blackboard) and Curtin's Academic Integrity website for information on academic misconduct (which includes plagiarism and collusion).

The unit coordinator may require you to provide an oral justification of, or to answer questions about, any piece of written work submitted in this unit. Your response(s) may be referred to as evidence in an Academic Misconduct inquiry. In addition, your assignment submission may be analysed by Turnitin and/or other systems to detect plagiarism and/or collusion.

End of Assignment