FIT 3162 Computer Science Project 2
TEST REPORT
GPU Acceleration for Raster Filter Using APARAPI

Group 9

Christine 29392888

Wan Jack Lee 28848551

Zisong Liao 28418107

Table of content

1. Description of test approach

There are two test approaches in the project, which are the JUnitTest frameworks and manual testing. The JUnitTest will be conducted for each method of the classes, to assure the correctness of output and the correctness of error-catching, in order to fail-safe when incorrect input is given.

Manual testing will be conducted for integration testing and performance testing ,which is also our usability test, to test the software as a whole. It includes input several file types and comparison of the speed with the Eduard program.

2. Test

2.1. GradientOperator

2.1.1. createGradientOperatorObject

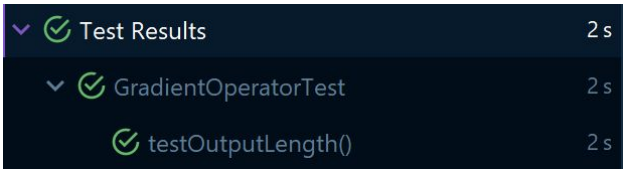
Purpose of Test	To ensure the correctness of creating GradientOperator object
Method of Test	Java JUnitTest
Input of the Test	-- (no argument needed for creating GradientOperator)
Expected output	GradientOperator object created without error (pass unit testcase)
Actual output	GradientOperator object created without error (pass unit testcase) 

2.1.2. testInvalidGrid

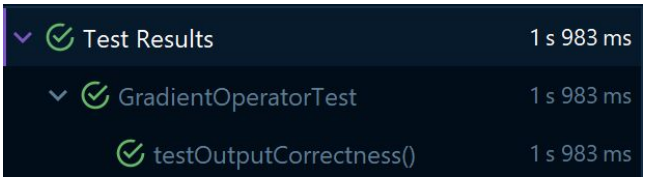
Purpose of Test	To ensure the error handling works perfectly (when the input Grid is not presenting in operate method in GradientOperator)
Method of Test	Java JUnitTest
Input of the Test	Null value for the operate method
Expected output	Error catched (pass unit testcase)
Actual output	Error catched (pass unit testcase) 

2.1.3. testOutputLength

Purpose of Test	To ensure that the output length of the GradientOperator(APARAPI) is same as the GradientOperator(Eduard)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to 4000; Grid(Eduard) object which fill with attributes of the

	Grid(APARAPI).
Expected output	Output length of the GradientOperator(APARAPI) object is equal to output length of the GradientOperator(Eduard) object (pass unit testcase)
Actual output	Output length of the GradientOperator(APARAPI) object is equal to output length of the GradientOperator(Eduard) object (pass unit testcase) 

2.1.4. testOutputCorrectness

Purpose of Test	To ensure that the output values of the GradientOperator(APARAPI) is totally equal to the GradientOperator(Eduard)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to 4000; Grid(Eduard) object which fills with attributes of the Grid(APARAPI).
Expected output	Output values of the GradientOperator(APARAPI) object is equal to output values of the GradientOperator(Eduard) object (pass unit testcase)
Actual output	Output values of the GradientOperator(APARAPI) object is equal to output values of the GradientOperator(Eduard) object (pass unit testcase) 

2.2. HorizontalTransposingLowPassFilter

2.2.1. createHorizontalOperatorObject

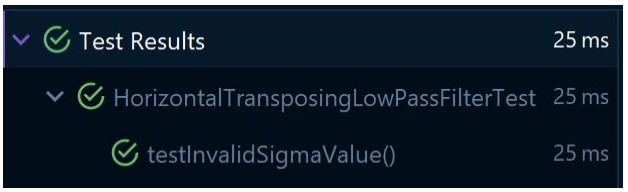
Purpose of Test	To ensure the correctness of creating HorizontalTransposingLowPassFilter object
Method of Test	Java JUnitTest
Input of the Test	For horizontalTransposingFirstPass instance, firstPass set to true and sigma set to 6f. For horizontalTranspoingSecondPass instance, firstPass set to false and sigma set to 6f.
Expected output	Both HorizontalTransposingLowPassFilter object created without error (pass unit testcase)
Actual output	Both HorizontalTransposingLowPassFilter object created without error (pass unit testcase) 

2.2.2. testInvalidGrid

Purpose of Test	To ensure the error handling works perfectly (when the input Grid is not presenting in operate method in HorizontalTransposingLowPassFilter)
Method of Test	Java JUnitTest
Input of the Test	Null value for the operate method
Expected output	Error catched (pass unit testcase)
Actual output	Error catched (pass unit testcase) 

2.2.3. testInvalidSigmaValue

Purpose of Test	To ensure the non-negative sigma value will be rejected when creating HorizontalTransposingLowPassFilter object
Method of Test	Java JUnitTest
Input of the Test	For horizontalTransposingFirstPass instance, firstPass set to true and sigma set to -100.

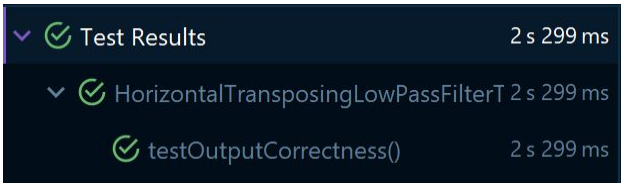
Expected output	Error catched (pass unit testcase)
Actual output	Error catched (pass unit testcase) 

2.2.4. testOutputLength

Purpose of Test	To ensure that the output length of the HorizontalTransposingLowPassFilter (APARAPI) is same as the AbstractFrequencyOperator(Eduard) **as the HorizontalTransposingLowPassFilter(Eduard) is a private class in AbstractFrequencyOperator(Eduard) thus unable to test them separately.
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to 4000; Grid(Eduard) object which fills with attributes of the Grid(APARAPI).
Expected output	Output length of the HorizontalTransposingLowPassFilter (APARAPI) object is equal to output length of the AbstractFrequencyOperator(Eduard) object (pass unit testcase)
Actual output	Output length of the HorizontalTransposingLowPassFilter (APARAPI) object is equal to output length of the AbstractFrequencyOperator(Eduard) object (pass unit testcase) 

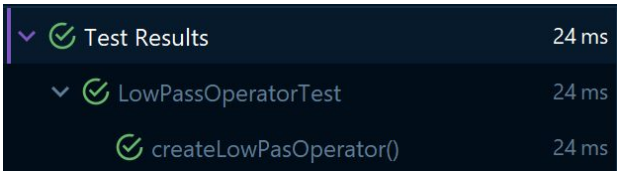
2.2.5. testOutputCorrectness

Purpose of Test	To ensure that the output values of the HorizontalTransposingLowPassFilter (APARAPI) is equal to the output values of AbstractFrequencyOperator(Eduard) **as the HorizontalTransposingLowPassFilter(Eduard) is a private class in AbstractFrequencyOperator(Eduard) thus unable to test them separately.
Method of Test	Java JUnitTest

Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to 4000; Grid(Eduard) object which fills with attributes of the Grid(APARAPI).
Expected output	Output values of the HorizontalTransposingLowPassFilter (APARAPI) object is equal to output values of the AbstractFrequencyOperator(Eduard) object (pass unit testcase)
Actual output	Output values of the HorizontalTransposingLowPassFilter (APARAPI) object is equal to output values of the AbstractFrequencyOperator(Eduard) object (pass unit testcase) 

2.3. LowPassOperator

2.3.1. createLowPassOperator

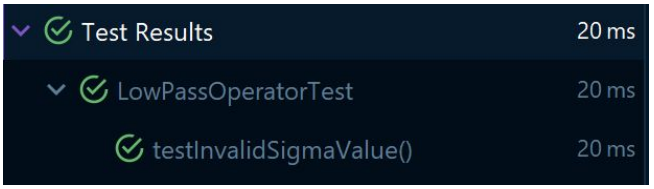
Purpose of Test	To ensure the correctness of creating LowPassOperator object
Method of Test	Java JUnitTest
Input of the Test	For lowPassOperatorSigmaBlur instance, sigma is set to 6f. For lowPassOperatorSigmaSmooth instance, sigma is set to 20f.
Expected output	Both LowPassOperator object created without error (pass unit testcase)
Actual output	Both LowPassOperator object created without error (pass unit testcase) 

2.3.2. testInvalidGrid

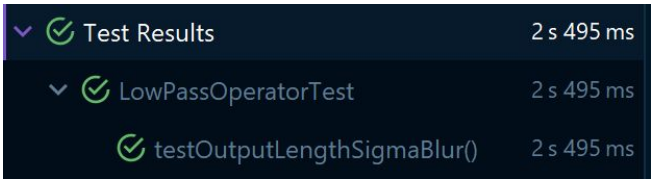
Purpose of Test	To ensure the error handling works perfectly (when the input Grid is not presenting in operate method in LowPassOperator)
Method of Test	Java JUnitTest
Input of the Test	Null value for the operate method
Expected output	Error catched (pass unit testcase)
Actual output	Error catched (pass unit testcase) 

2.3.3. testInvalidSigmaValue

Purpose of Test	To ensure the non-negative sigma value will be rejected when creating LowPassOperator object
Method of Test	Java JUnitTest
Input of the Test	For lowPassOperatorSigmaBlur instance, sigma is set to -100.
Expected output	Error catched (pass unit testcase)
Actual output	Error catched (pass unit testcase)

2.3.4. testOutputLengthSigmaBlur

Purpose of Test	To ensure that the output length of the LowPassOperator(APARAPI, sigmaBlur) is same as the LowPassOperator(Eduard, sigmaBlur)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to 4000; Grid(Eduard) object which fills with attributes of the Grid(APARAPI).
Expected output	Output length of the LowPassOperator(APARAPI, sigmaBlur) object is equal to output length of the LowPassOperator(Eduard, sigmaBlur) object (pass unit testcase)
Actual output	Output length of the LowPassOperator(APARAPI, sigmaBlur) object is equal to output length of the LowPassOperator(Eduard, sigmaBlur) object (pass unit testcase) 

2.3.5. testOutputCorrectnessSigmaBlur

Purpose of Test	To ensure that the output values of the LowPassOperator(APARAPI, sigmaBlur) is equals to the LowPassOperator(Eduard, sigmaBlur)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to 4000; Grid(Eduard) object which fills with attributes of the Grid(APARAPI).
Expected output	Output values of the LowPassOperator(APARAPI, sigmaBlur) object is equal to output values of the LowPassOperator(Eduard, sigmaBlur) object

	(pass unit testcase)
Actual output	Output values of the LowPassOperator(APARAPI, sigmaBlur) object is equal to output values of the LowPassOperator(Eduard, sigmaBlur) object (pass unit testcase) 

2.3.6. testOutputLengthSigmaSmooth

Purpose of Test	To ensure that the output length of the LowPassOperator(APARAPI, sigmaSmooth) is same as the LowPassOperator(Eduard, sigmaSmooth)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to 4000; Grid(Eduard) object which fills with attributes of the Grid(APARAPI).
Expected output	Output length of the LowPassOperator(APARAPI, sigmaSmooth) object is equal to output length of the LowPassOperator(Eduard, sigmaSmooth) object (pass unit testcase)
Actual output	Output length of the LowPassOperator(APARAPI, sigmaSmooth) object is equal to output length of the LowPassOperator(Eduard, sigmaSmooth) object (pass unit testcase) 

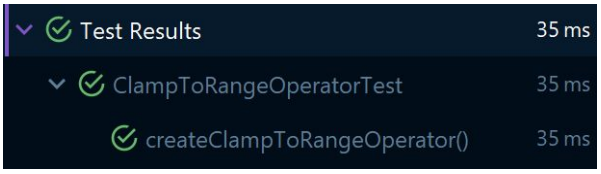
2.3.7. testOutputCorrectnessSigmaSmooth

Purpose of Test	To ensure that the output values of the LowPassOperator(APARAPI, sigmaSmooth) is equals to the LowPassOperator(Eduard, sigmaSmooth)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to

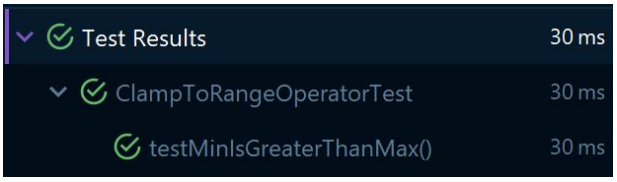
	4000; Grid(Eduard) object which fills with attributes of the Grid(APARAPI).
Expected output	Output values of the LowPassOperator(APARAPI, sigmaSmooth) object is equal to output values of the LowPassOperator(Eduard, sigmaSmooth) object (pass unit testcase)
Actual output	Output values of the LowPassOperator(APARAPI, sigmaSmooth) object is equal to output values of the LowPassOperator(Eduard, sigmaSmooth) object (pass unit testcase) 

2.4. ClampToRangeOperator

2.4.1. createClampToRangeOperator

Purpose of Test	To ensure the correctness of creating ClampToRangeOperator object
Method of Test	Java JUnitTest
Input of the Test	For the clampToTangeOperator instance, the minimum is 1 and the maximum is 99
Expected output	ClampToRangeOperator object created without error (pass unit testcase)
Actual output	ClampToRangeOperator object created without error (pass unit testcase) 

2.4.2. testMinIsGreaterThanMax

Purpose of Test	To ensure the error handling works perfectly (when the maximum value is less than the minimum value)
Method of Test	Java JUnitTest
Input of the Test	For the clampToTangeOperator instance, the minimum is 999 and the maximum is 111
Expected output	Error catched (pass unit testcase)
Actual output	Error catched (pass unit testcase) 

2.4.3. testInvalidGrid

Purpose of Test	To ensure the error handling works perfectly (when the input Grid is not presenting in operate method in ClampToRangeOperator)
Method of Test	Java JUnitTest
Input of the Test	Null value for the operate method
Expected output	Error catched (pass unit testcase)

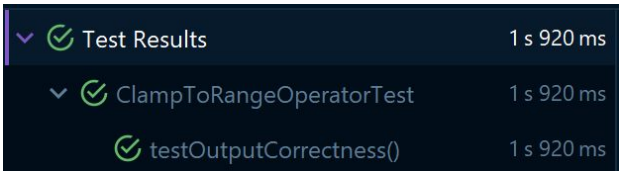
Actual output	Error catched (pass unit testcase) 

2.4.4. testOutputLength

Purpose of Test	To ensure that the output length of the ClampToRangeOperator(APARAPI) is same as the ClampToRangeOperator(Eduard)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to 4000; Grid(Eduard) object which fills with attributes of the Grid(APARAPI).
Expected output	Output length of the ClampToRangeOperator(APARAPI) object is equal to output length of the ClampToRangeOperator(Eduard) object (pass unit testcase)
Actual output	Output length of the ClampToRangeOperator(APARAPI) object is equal to output length of the ClampToRangeOperator(Eduard) object (pass unit testcase) 

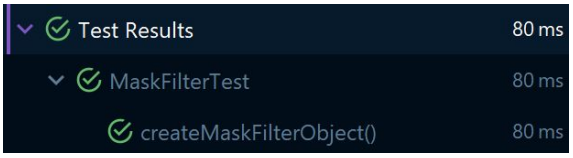
2.4.5. testOutputCorrectness

Purpose of Test	To ensure that the output values of the ClampToRangeOperator(APARAPI) is equals to the ClampToRangeOperator(Eduard)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to 4000; Grid(Eduard) object which fills with attributes of the Grid(APARAPI).
Expected output	Output values of the ClampToRangeOperator(APARAPI) object is equal to output length of the

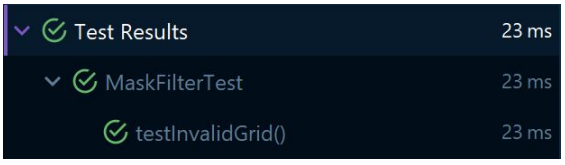
	ClampToRangeOperator(Eduard) object (pass unit testcase)
Actual output	Output values of the ClampToRangeOperator(APARAPI) object is equal to output length of the ClampToRangeOperator(Eduard) object (pass unit testcase) 

2.5. MaskFilter

2.5.1. createMaskFilterObject

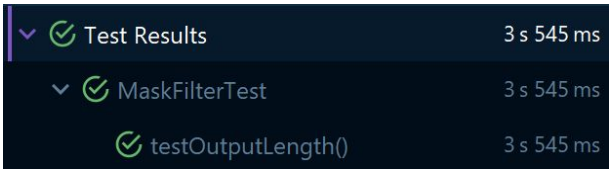
Purpose of Test	To ensure the correctness of creating MaskFilter object
Method of Test	Java JUnitTest
Input of the Test	-- (no argument needed for creating MaskFilter)
Expected output	MaskFilter object created without error (pass unit testcase)
Actual output	MaskFilter object created without error (pass unit testcase) 

2.5.2. testInvalidGrid

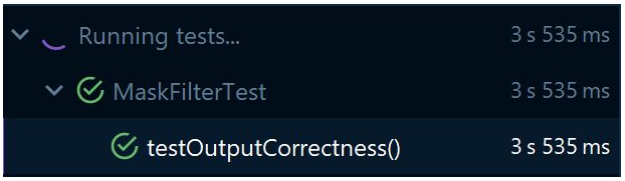
Purpose of Test	To ensure the error handling works perfectly (when the input Grid is not presenting in operate method in MaskFilter)
Method of Test	Java JUnitTest
Input of the Test	Null value for the constructor method
Expected output	Error catched (pass unit testcase)
Actual output	Error catched (pass unit testcase) 

2.5.3. testOutputLength

Purpose of Test	To ensure that the output length of the MaskFilter(APARAPI) is same as the maskFilter method in Main(Eduard)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to 4000; Grid(Eduard) object which fill with attributes of the Grid(APARAPI).
Expected output	Output length of the MaskFilter(APARAPI) object is equal to output length of the maskFilter method in Main(Eduard) object (pass unit testcase)
Actual output	Output length of the MaskFilter(APARAPI) object is equal to

	output length of the maskFilter method in Main(Eduard) object (pass unit testcase) 

2.5.4. testOutputCorrectness

Purpose of Test	To ensure that the output values of the MaskFilter(APARAPI) is equal to the maskFilter method in Main(Eduard)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns and 1500 rows which fill with random float values within range of 2500 to 4000; Grid(Eduard) object which fill with attributes of the Grid(APARAPI).
Expected output	Output values of the MaskFilter(APARAPI) object is equal to output values of the maskFilter method in Main(Eduard) object (pass unit testcase)
Actual output	Output values of the MaskFilter(APARAPI) object is equal to output values of the maskFilter method in Main(Eduard) object (pass unit testcase) 

2.6. Grid

2.6.1. shallowCopy

Purpose of Test	To ensure the shallow copy method of the Grid(APARAPI) works perfectly
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns, 1500 rows, cellSize of 0.5, north latitude of 0.6, south latitude of 0.8, east longitude of 0.8, west longitude of 0.8, which fill with random float values within range of 2500 to 4000
Expected output	A new Grid object which has the same attribute as the original Grid object.
Actual output	A new Grid object which has the same attribute as the original Grid object. 

2.6.2. fillWithRandomFloat

Purpose of Test	To ensure that the fill buffer with random float within a given range method works perfectly
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns, 1500 rows, cellSize of 0.5, north latitude of 0.6, south latitude of 0.8, east longitude of 0.8, west longitude of 0.8
Expected output	The buffer of the grid is filled with random float value within 2500 to 4000 (pass unit testcase)
Actual output	The buffer of the grid is filled with random float value within 2500 to 4000 (pass unit testcase) 

2.6.3. fillWithZero

Purpose of Test	To ensure that the fill buffer with 0 within a given range method works perfectly

Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns, 1500 rows, cellSize of 0.5, north latitude of 0.6, south latitude of 0.8, east longitude of 0.8, west longitude of 0.8
Expected output	The buffer of the grid is filled with 0 (pass unit testcase)
Actual output	The buffer of the grid is filled with 0 (pass unit testcase) 

2.6.4. getLength

Purpose of Test	To ensure the get length method of the grid works perfectly (the capacity of the buffer)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns, 1500 rows, cellSize of 0.5, north latitude of 0.6, south latitude of 0.8, east longitude of 0.8, west longitude of 0.8, which fill with random float values within range of 2500 to 4000
Expected output	The count of the item in the buffer is same as the output of get length method. The output of get length method is same as the total columns times total rows. (pass unit testcase)
Actual output	(The count of the item in the buffer is same as the output of get length method. The output of get length method is same as the total columns times total rows. (pass unit testcase) 

2.6.5. setGetBufferReceived

Purpose of Test	To ensure that the setBufferReceived and getBuffer method works perfectly
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns, 1500 rows, cellSize of 0.5, north latitude of 0.6, south latitude of 0.8, east

	longitude of 0.8, west longitude of 0.8. A separate float array (let say array A) which filled with 0.
Expected output	The buffer that set to the grid object is equal to the array A (pass unit testcase)
Actual output	The buffer that set to the grid object is equal to the array A (pass unit testcase) Test Results 30 ms GridTest 30 ms setGetBufferReceived() 30 ms

2.6.6. setBufferReceivedNull

Purpose of Test	To ensure the error handling works perfectly (when the float array is not exist)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns, 1500 rows, cellSize of 0.5, north latitude of 0.6, south latitude of 0.8, east longitude of 0.8, west longitude of 0.8. Null value for the setBufferReceived method
Expected output	Error catched (pass unit testcase)
Actual output	Error catched (pass unit testcase) Test Results 20 ms GridTest 20 ms setBufferReceivedNull() 20 ms

2.6.7. testGet

Purpose of Test	Test the get method of the grid (with 2d index and 1d index)
Method of Test	Java JUnitTest

Input of the Test	Grid(APARAPI) object which has 1500 columns, 1500 rows, cellSize of 0.5, north latitude of 0.6, south latitude of 0.8, east longitude of 0.8, west longitude of 0.8.
	`newBuffer[0] = 1;`
	`newBuffer[1] = 2;`
	`newBuffer[2] = 3;`
	
	`newBuffer[grid.getDirectIndex( col: 4, row: 5)] = 6;`
	`newBuffer[grid.getDirectIndex( col: 5, row: 6)] = 7;`
	`newBuffer[grid.getDirectIndex( col: 6, row: 7)] = 8;`

Expected output	All the value which hardcoded into the buffer can be get directly
	(pass unit testcase)
	`for (int i = 0; i < 3; i++)`
	`  assertEquals(grid.get(i), actual: i+1, delta: 2);`
	
	`for (int i = 4; i < 7; i++)`
	`  assertEquals(grid.get(i, row: i+1), actual: i+2, delta: 2);`

Actual output	All the value which hardcoded into the buffer can be get directly (pass unit testcase) Test Results 34 ms GridTest 34 ms testGet() 34 ms

2.6.8. testSet

Purpose of Test	Test the set method of the grid (with 2d index and 1d index)
Method of Test	Java JUnitTest

Input of the Test	Grid(APARAPI) object which has 1500 columns, 1500 rows, cellSize of 0.5, north latitude of 0.6, south latitude of 0.8, east longitude of 0.8, west longitude of 0.8.
	`grid.set(0,0);`
	`grid.set(0,1);`
	`grid.set(0,2);`
	
	`grid.set( value: 0, col: 3, row: 4);`
	`grid.set( value: 0, col: 4, row: 5);`
	`grid.set( value: 0, col: 5, row: 6);`

Expected output	All the values has set to grid
	(pass unit testcase)
	`for (int i = 0; i < 3; i++)`
	`  assertEquals(grid.get(i), actual: 0, delta: 2);`
	
	`for (int i = 3; i < 6; i++)`
	`  assertEquals(grid.get(i, row: i+1), actual: 0, delta: 2);`

Actual output	All the values has set to grid (pass unit testcase)

	Test Results 50 ms GridTest 50 ms testSet() 50 ms

2.6.9. getRow

Purpose of Test	Test the getRow method (convert 1d representation to 2d row index)
Method of Test	Java JUnitTest

Input of the Test	Grid(APARAPI) object which has 1500 columns, 1500 rows.
	`assertEquals(grid.getRow( directIndex: 1500), actual: 1);`
	`assertNotEquals(grid.getRow( directIndex: 1499), actual: 1);`
	
	`try{`
	`  grid.getRow( directIndex: -100);`
	`  fail();`
	`}`
	`catch(IllegalArgumentException ex){`
	
	`}`

Expected output	(pass unit testcase)
Actual output	(pass unit testcase)

2.6.10. getCol

Purpose of Test	Test the getCol method (convert 1d representation to 2d col index)
Method of Test	Java JUnitTest
Input of the Test	Grid(APARAPI) object which has 1500 columns, 1500 rows.

	`assertEquals(grid.getCol( directIndex: 1500), actual: 0);`
	`assertNotEquals(grid.getCol( directIndex: 1499), actual: 0);`
	
	`try{`
	`  grid.getCol( directIndex: -100);`
	`  fail();`
	`}`
	`catch(IllegalArgumentException ex){`
	
	`}`

Expected output	(pass unit testcase)
Actual output	(pass unit testcase) Test Results 33 ms GridTest 33 ms getCol() 33 ms

2.6.11. getDirectIndex

Purpose of Test	Test the getDirectIndex method (convert 2d row and col index to 1d representation)
Method of Test	Java JUnitTest

Input of the Test	Grid(APARAPI) object which has 1500 columns, 1500 rows.
	`assertEquals(grid.getDirectIndex( col: 2, row: 3), actual: 4502);`
	`assertNotEquals(grid.getDirectIndex( col: 1, row: 1), actual: 1);`
	
	`try{`
	`  grid.getDirectIndex( col: -100, row: 0);`
	`  grid.getDirectIndex( col: 0, row: -100);`
	`  fail();`
	`}`
	`catch(IllegalArgumentException ex){`
	`}`

Expected output	(pass unit testcase)
Actual output	(pass unit testcase) Test Results 25 ms GridTest 25 ms getDirectIndex() 25 ms

2.7. OverallTest

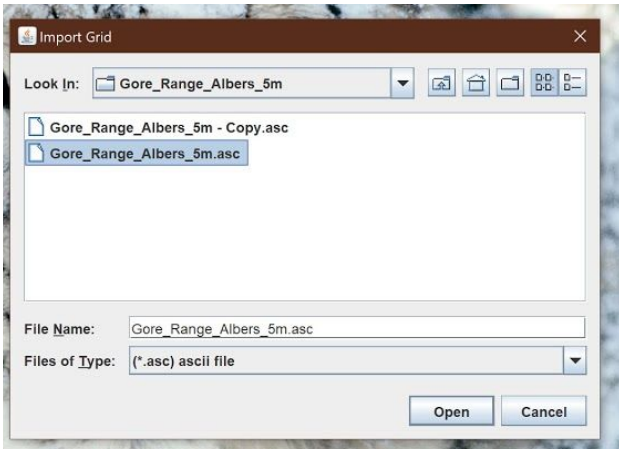
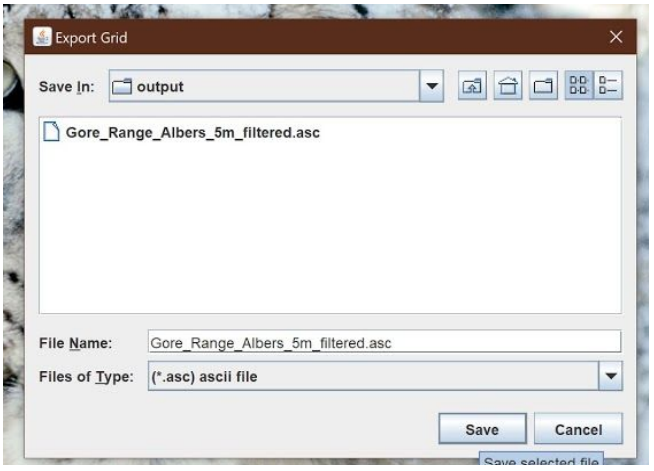
The Integration testing and Performance testing are related to the usability test and will be discussed in Section 3(Usability Test).

3. Usability Test

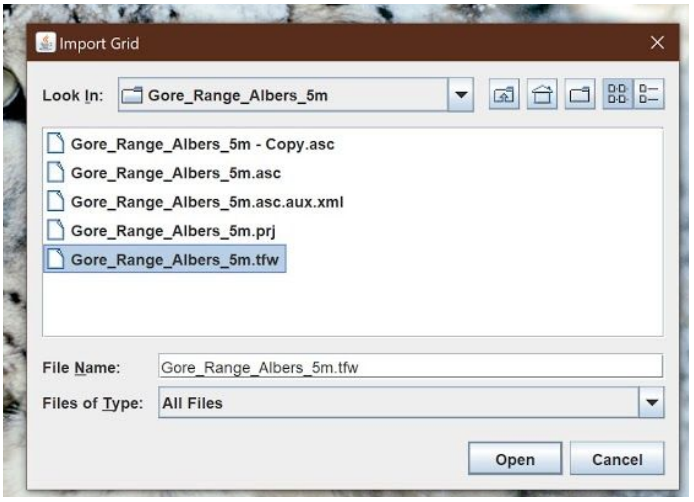
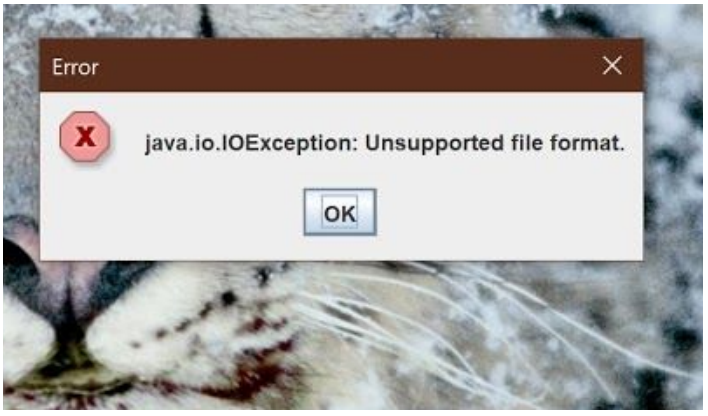
3.1. Integration Testing

The integration testing will test through the APARAPI version of the raster filter generator. It will take in three types of input. To start the program, please refer to the Code Report - User Guide.

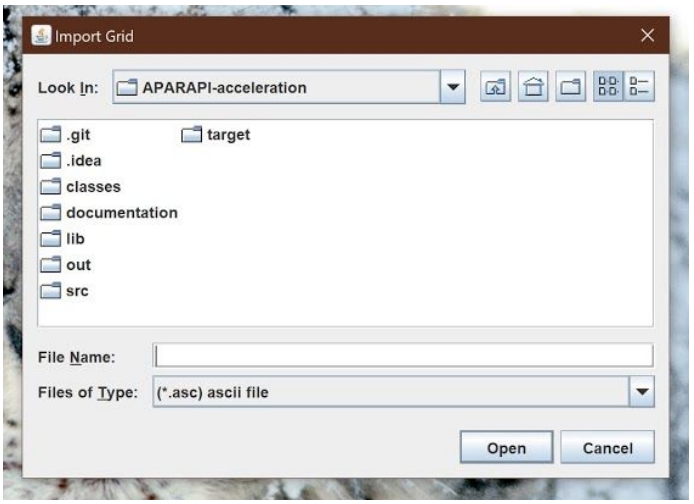
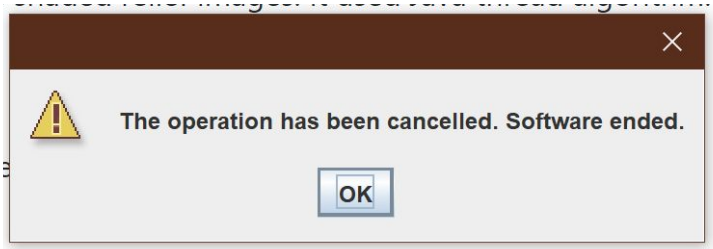
3.1.1. Correct file type

Grid Input	Gore_Range_Albers_5m.asc The asc file is the raster grid file. 
Grid Output	Gore_Range_Albers_5m_filtered.asc * The asc file is the raster grid file which applies the mask filter algorithm. Gore_Range_Albers_5m_filtered.png * The png file is the preview of the post-processed raster grid file Gore_Range_Albers_5m_aparapi.txt * The txt file will store the benchmark of each algorithm. (i.e algorithm runtime).  ** After outputting all the files, a window will pop-out and shows the runtime of each algorithm. Software execution done

3.1.2. Invalid file type test

Grid Input	A file which is not asc raster file. In this case we input a random file 
Grid Output	No output file
Error Message	A message which indicates the file type is not unsupported. The program will end after that. 

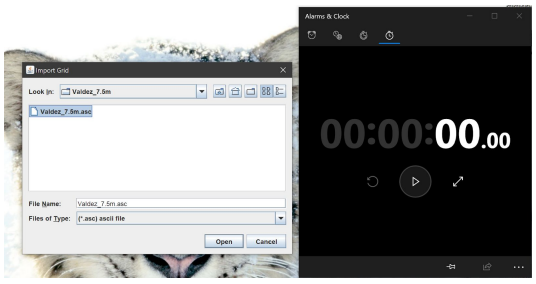
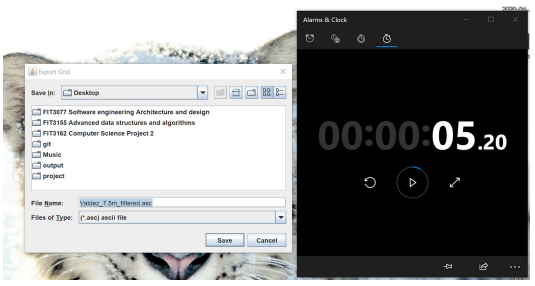
3.1.3. File not present

Grid Input	No file is given. 
Grid Output	No output file
Warning Message	A message which indicates no input file given. The program will end after that. 

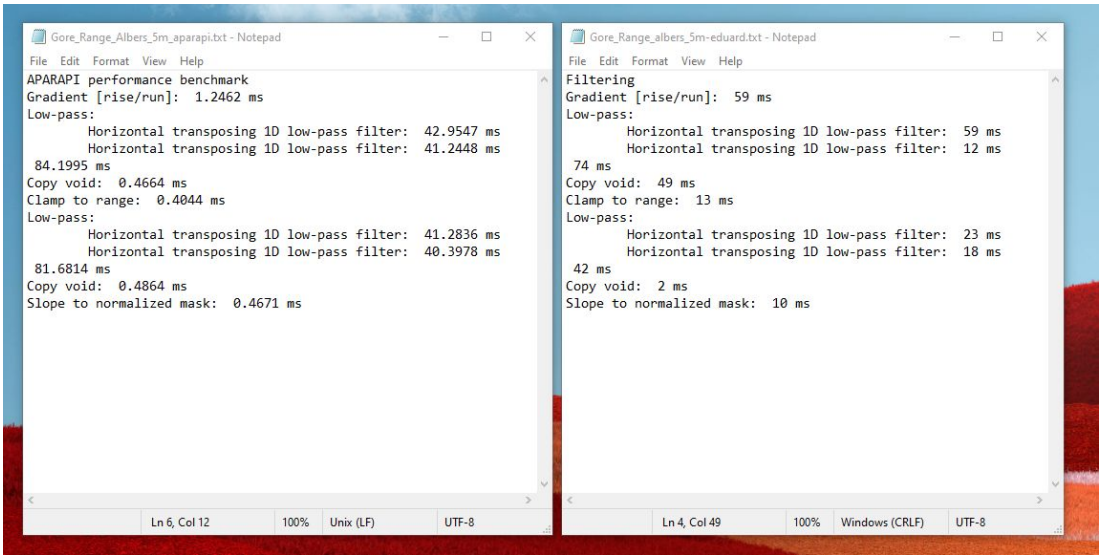
3.2. Performance Testing

The execution environment of the program is in Windows 10 with the configured processor Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz with 8 Logical Processor(s) and 8GB ram. The Graphics Processing Unit of the execution environment is NVIDIA GeForce GTX 1050 with dedicated GPU memory of 2GB and shared GPU memory of 4GB.

The average execution time of the software is 5 seconds (after selecting the grid file and before exporting the grid file).

Based on the output benchmark comparison, we know that most of the APARAPI operator has achieved speed up compared to Eduard operator except Horizontal transposing operator. The overall APARAPI raster filter software is slightly slower than Eduard software. For a more detailed explanation, refer to [Final Report - Outcomes - Results](#).



4. Limitations of software

Based on the unit test cases, it reveals some limitations of the software. For instance, the overall program runtime is not optimized as compared with the Eduard program. The reason for dragging the runtime is caused by the limitation of APARAPI. As APARAPI doesn't support read-in multiple rows of the array at the same time, and the horizontal transposing operator will have to execute by chunk-basic, thus the algorithms will have to execute in sequential order. The disadvantages of executing in sequential order will be a drawback of the speedup.

Furthermore, APARAPI requires to copy the array directly to GPU memory from memory before processing the data and copy back the data to CPU memory, the communication overhead and the workload of the GPU will moreover slow down the program.

For the detailed explanation, refer to [Final Report - Outcomes - Limitations of project outcomes](#).

5. Recommendation for Improvements

There are several suggestions to overcome the limitations of the software. The first suggestion is to replace APARAPI grid class with Eduard grid class, it requires refactorization of classes and modification of the original Eduard class (which we try to avoid). The second suggestion is to design a new algorithm for the horizontal transposing operator which doesn't require reading multiple rows at the same time.

For a detailed explanation refer to Final Report - Outcomes - Future works and improvements.

6. Testcase excluded

The test case which is excluded is the testing of the horizontal transposing algorithm of Eduard. The reason is that the horizontal transposing operator of Eduard is a private class inside Eduard Abstract Frequency Operator thus it will be unable to check the correctness of the algorithm by creating the instance of the object. Though, it is mitigated by checking the results of applying horizontal transposing operator of APARAPI twice with the results of Eduard Abstract Frequency Operator.