Algorithmic Implementation of Atmospheric Correction over Water

Josh Sixsmith & Passang Dorj

November 03 2019

1 Introduction

This document provides a brief overview of the implementation of atmospheric correction over water surfaces. The implementation is based on information provided in https://www.mssanz.org.au/modsim2017/G9/li.pdf.
The algorithm and workflow has been implemented into the *wagl* codebase (https://www.github.com/GeoscienceAustralia/wagl). A key component to the algorithmic implementation is the derivation and application of a point spread function representing the atmosphere.
The PSF is applied via image convolution, and depending on the resolution of the image, a kernel representing the PSF could be very large. This can present an issue for a production system when using standard image convolution. As such, two methods of convolution were implemented, and both versions will be compared and discussed.

2 Implementation details

2.1 Derivation of the atmospheric PSF

We have been supplied a build of MODTRAN (version 5.4) that has been internally modified in order in order to output the atmospheric PSF.
The wagl codebase makes use of MODTRAN 6.0, and rather than fall back to using MODTRAN 5.4, it was decided to keep using MODTRAN 6.0 for the radiative transfer component, and only use MODTRAN 5.4 for retrieving the PSF.
The supplied version of MODTRAN 5.4 was further modified in order to output the channel names that a calculated PSF refers to. The modification enabled a simpler workflow that relied on no assumptions about a given sensor, or for the channel order. The outputs are still appended to the TP7 file.
The PSF is read and stored (along with all the outputs produced by wagl) in an HDF5 file, as well as the 2D kernel representing the PSF.
The kernel is stored unnormalised; however, it is normalised during the convolution process.

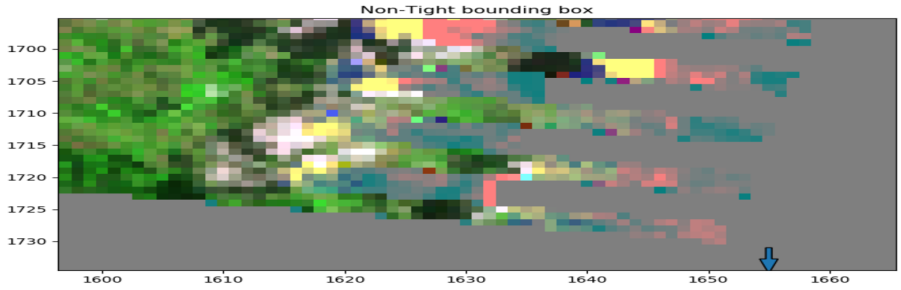
2.2 Aerosol type

The aerosol type is not enabled from the command line; instead, a simple boolean switch has been implemented that specifies that an acquisition or given a set of acquisitions is to be processed via the water atmospheric correction method instead. Internally, the workflow will then use a Maritime Navy model. This behaviour could be changed, enabling a user to specify 1 of several supported aerosol models.
For a production system to be designed and implemented, a logical process needs to be defined that enables automatic processing with a given aerosol. i.e. how does one determine the aerosol model for use within the workflow process? A quick solution is simply to pre-define locations to specifically use a given aerosol model. Considering the short time frames involved, the simplest approach is to process a batch job where a user specifies what model to use for each batch of processing.

2.3 Convolution (and preperation for)

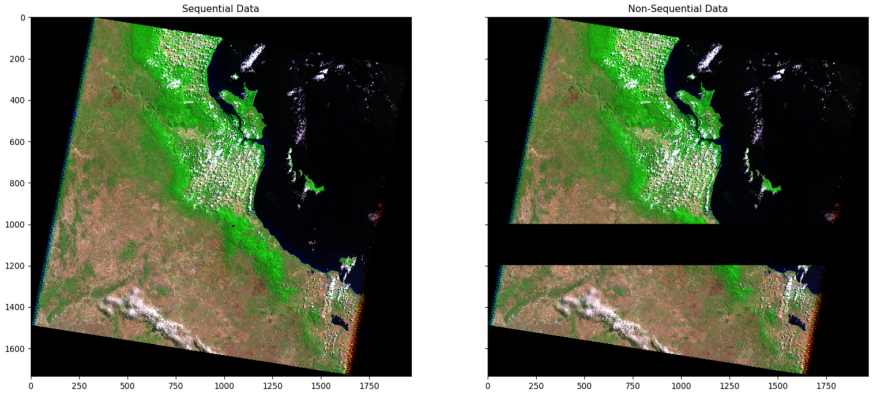
Convolution does not work effectively if the data contains null values. To avoid a null data issue, run length (or row length) averages are calculated and used as a fill value. This is a cheap and easy method to implement and compute; however, under certain conditions may not capture the spatial variability nor the context correctly. Alternate methods, such as spatial neighbourhood averaging have not been tested, but like anything, will require thorough testing and evaluation.
We did uncover an issue regarding image acquisitions that don't have a tight bounding box around the data extent, i.e. full rows of null data located at the top and bottom of the image. We now handle this issue by reducing the image to a tight bounding box before convolution and insert the result into the full array dimensions to preserve the original extent.

Figure 1: Example of a non-tight bounding box



However, if a row of null data happens to occur between two valid rows, we call this phenomenon non-sequential and raise an Exception to stop the process. We don't anticipate coming across many acquisitions if any at all. So we felt it best to stop the process and record the acquisition id for future investigation.

Figure 2: Example of sequential and non-sequential data



Prior to convolution, both the data and the kernel are padded to an optimal size in both the x and y directions. The pad size is determined in two steps, 1. half the kernel size in each dimension + 1, 2. Find the next optimal power of 2 to enable an efficient Fourier transform. The actual padding method is *symmetric* for the data and *constant* for the kernel.
The convolution process, depending on the dimensions of the array and the resolution, can be very time-consuming. For example, processing a (10980, 10980) image with a (141, 141) kernel took approximately 50 minutes. So for Sentinel-2, there would be an additional three images of the same dimensions to process. The current design of the workflow isn't geared towards parallelising a single function; rather, it is geared towards parallelising across acquisitions (process more acquisitions at once).
As the PSF is designed to be radially symmetric, we've also enabled convolution to occur via Fourier. This has significantly reduced the computational time, at the expense of more system memory. Processing the same image example would take approximately 3 minutes via the Fourier method.

2.4 Glint correction

While the paper that the implementations ware based of included both a sun and sky glint correction. In consultation with Team Wombats, we have only implemented sky glint correction as an appropriate source of wind speed data has not been identified, as well as the fact that more work is required to develop an appropriate method that accounts for the high spatial variability of wind speed.
As for sky glint correction, a value of 1.34 for the refractive water index has been hardcoded. If desirable, the refractive water index can be added to the command line. Alternatively, wagl can be imported as a standard Python module, and the functions separately and define a value for the refractive water index as a keyword argument.
We were provided with outputs from a sample dataset to use as a comparison, but we were unable to get a satisfyingly close value on a simple image difference. As such, we then turned to the sample code that was provided and found that there is an additional variable *scat*.

The paper used this equation:

$$\rho_g = f_s \rho_{sung} + (1 - f_{skyg})\rho_{skyg} \tag{1}$$

Whereas the code used:

$$\rho_g = f_s \rho_{sung} + ((1 - f_{skyg}) + scat)\rho_{skyg} \tag{2}$$

Equation 2 yielded results that were slightly closer to the sample dataset, but still not the same. Despite the difference in results, equation 2 has been implemented and is in use by the workflow.

3 Outputs

3.1 Sample package

The current output package is not a representation of the final package. Additional work is required to determine the minimum requirements that meet the need for downstream applications relying on the atmospheric correction over water product.
The test output package contains:

- Lambertian Reflectance
- Lambertian reflectance with atmospheric adjacency correction
- NBAR (using adjacency corrected lambertian as input)
- NBART (using adjacency corrected lambertian as input)
- Fmask (classification schema identifing clear, cloud, cloud shadow, water and snow pixels)
- FV (Direct fraction in the view direction)
- Azimuthal incident and exiting angles
- Incident and exiting angles
- Relative azimuth and relative slope
- terrain shadow (cast and self shadow)
- satellite azimuth and satellite view
- solar azimuth and solar zenith
- time delta of the sensor acquisition

The outputs will be made available via an opendatacube instance, and the database location and product name will be provided soon. At this stage, the access will be via NCI, but if desireable, some work could be done to make the data available via an EC2 https://aws.amazon.com/ec2/ instance on Amazon Web Services https://aws.amazon.com/ running opendatacube.
While the paper mentioned not applying the point spread function to channels measuring across the SWIR domain, we decided not to include that logic here. The reasoning was mostly a consideration of timing constraints; however, if desired for the next phase of testing and validation, this work can be done. But it may also depend on defining what is required to be contained within the *atmospheric correction over water* product and what processes are to be applied to each channel.

The following lists what channels will be output for each sensor as part of this initial implementation:

- Landsat-5 TM Channels:
  - BAND 1 (Blue)
  - BAND 2 (Green)
  - BAND 3 (Red)
  - BAND 4 (NIR)
  - BAND 5 (SWIR 1)
  - BAND 7 (SWIR 2)
- Landsat-7 ETM+ Channels:
  - BAND 1 (Blue)
  - BAND 2 (Green)
  - BAND 3 (Red)
  - BAND 4 (NIR)
  - BAND 5 (SWIR 1)
  - BAND 7 (SWIR 2)
  - BAND 8 (Panchromatic)
- Landsat-8 OLI Channels:
  - BAND 1 (Coastal Aerosol)
  - BAND 2 (Blue)
  - BAND 3 (Green)
  - BAND 4 (Red)
  - BAND 5 (NIR)
  - BAND 6 (SWIR 1)
  - BAND 7 (SWIR 2)
  - BAND 8 (Panchromatic)
- Sentinel-2 MSI Channels:
  - BAND 1 (Coastal Aerosol)
  - BAND 2 (Blue)
  - BAND 3 (Green)
  - BAND 4 (Red)
  - BAND 5 (Red Edge 1)
  - BAND 6 (Red Edge 2)
  - BAND 7 (Red Edge 3)
  - BAND 8 (NIR 1)
  - BAND 8A (NIR 2)
  - BAND 11 (SWIR 2)
  - BAND 12 (SWIR 3)

4 Results

4.1 Processing comparison of convolution methods

In total, 16 acquisitions across four different sensors were used for testing, not just the algorithmic additions, but also the framework and its ability to be applied generally across different satellite acquisitions and different resolutions. The acquisitions tested are listed below:

1. LT05_L1TP_093086_20041210_20161127_01_T1.tar
2. LT05_L1TP_093086_19991127_20161216_01_T1.tar
3. LT05_L1TP_095073_20080915_20161029_01_T1.tar
4. LE07_L1TP_093086_20081111_20161224_01_T1.tar
5. LE07_L1TP_093086_20010617_20170205_01_T1.tar
6. LE07_L1TP_095073_20070516_20170103_01_T1.tar
7. LC08_L1TP_093086_20141020_20170418_01_T1.tar
8. LC08_L1TP_093086_20151226_20170331_01_T1.tar
9. LC08_L1TP_095073_20130828_20170502_01_T1.tar
10. LC08_L1TP_097084_20171211_20171223_01_T1.tar
11. LC08_L1TP_110073_20150624_20170407_01_T1.tar
12. LC08_L1TP_112083_20151113_20170402_01_T1.tar
13. LC08_L1TP_115078_20130707_20170503_01_T1.tar
14. S2A_MSIL1C_20190401T002101_N0207_R116_T55HBU_20190401T014505.zip
15. S2B_MSIL1C_20190403T001109_N0207_R073_T55HBU_20190403T013318.zip
16. S2A_MSIL1C_20190723T020451_N0208_R017_T51KVA_20190723T033407.zip

The first comparison is looking at the processing efficiency and the memory overhead to get an idea of how much time is required for processing a batch of acquisitions, and the memory requirement which can limit how many acquisitions can be processed in parallel on a single node.
It should be noted that the following tables are not reporting the processing time and memory usage of convolution. The reported metrics are, in fact, of the entire workflow that outputs surface reflectance.

Table 1: Processing time and memory usage

Method	**Walltime**	**Memory**
Standard Convolution	07:32:18	64.0GB
Convolution via Fourier	00:50:30	55.45GB

As you can see, convolution via Fourier is much more efficient both in processing time, and only slightly more efficient in memory usage. But I would like to point out that this is merely comparing the time taken and memory used for processing the test acquisitions as a single batch job. The assumption is that for the standard convolution processing, the bulk of the time would be taken up by processing the higher-resolution channels.

Table 2: Per sensor processing time and memory usage (convolution via Fourier)

Sensor	**Walltime**	**Memory**
TM	00:22:22	1GB
ETM	00:52:19	7.68GB
OLI	00:43:23	9.02GB
MSI	00:56:10	4.61GB

While the wall times for ETM and MSI are longer than the 50 minutes wall time reported in Table 1, it should be noted that the times reported are from singular job runs. For a more realistic estimate, each job should ideally be run about half a dozen times each and an aggregate time reported. Unfortunately, with the timing constraints of this project, this will suffice. If a more accurate estimate is desired, this can be done at a more convenient time.
It should also be noted that the reported memory use comes directly from the PBS job status report and not from a memory profiling tool. The reported memory use is determined by a job monitoring tool managed by NCI and executed intermittently. As such, it will only be an estimate, and if a more accurate memory profile is desired, this can be done at a more convenient time.

Table 3: Per sensor processing time and memory usage per sensor (standard convolution)

Sensor	**Walltime**	**Memory**
TM	00:22:33	2.25GB
ETM	01:09:10	6.11GB
OLI	01:13:12	6.53GB
MSI	06:37:35	13.86GB

From Table 3, it is clear that there is a significant difference in wall time for those sensors that have higher resolution channels to process. With the MSI sensor topping the lot at just over 6.5 hours.
The processing times could be reduced in various ways, such as process each band independently; however, in simply changing the method, we can reduce the time significantly. Reducing the kernel size is another, however further testing on the impact of limiting the kernel size is required.

4.2 Output comparison of convolution methods

The following section is a simple difference comparison of the two different methods of applying convolution. The idea being that if the results come out vastly different, or simply not to an acceptable standard, then the alternate method (in this case using Fourier) will be abandoned.
The image datasets that used for comparison are all *atmoshperic adjacency corrected Lambertian reflectance*. If anything is wrong with this level of data output, then all datasets that use this as input are likely to be affected themselves.

Table 4: LT05_L1TP_093086_20041210_20161127_01_T1.tar

Channel name	**Min difference**	**Max difference**	**Percent different**
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 7	0	0	0

Table 5: LT05_L1TP_093086_19991127_20161216_01_T1.tar

Channel name	**Min difference**	**Max difference**	**Percent different**
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 7	0	0	0

Table 6: LT05_L1TP_095073_20080915_20161029_01_T1.tar

Channel name	**Min difference**	**Max difference**	**Percent different**
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 7	0	0	0

Table 7: LE07_L1TP_093086_20081111_20161224_01_T1.tar

Channel name	Min difference	Max difference	Percent different
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 7	0	0	0
BAND 8	0	0	0

Table 8: LE07_L1TP_093086_20010617_20170205_01_T1.tar

Channel name	Min difference	Max difference	Percent different
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 7	0	0	0
BAND 8	0	0	0

Table 9: LE07_L1TP_095073_20070516_20170103_01_T1.tar

Channel name	Min difference	Max difference	Percent different
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 7	0	0	0
BAND 8	0	0	0

Table 10: LC08_L1TP_093086_20141020_20170418_01_T1.tar

Channel name	Min difference	Max difference	Percent different
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 7	0	0	0
BAND 6	0	0	0
BAND 8	0	0	0

Table 11: LC08_L1TP_093086_20151226_20170331_01_T1.tar

Channel name	Min difference	Max difference	Percent different
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 6	0	0	0
BAND 7	0	0	0
BAND 8	0	0	0

Table 12: LC08_L1TP_095073_20130828_20170502_01_T1.tar

Channel name	Min difference	Max difference	Percent different
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 6	0	0	0
BAND 7	0	0	0
BAND 8	0	0	0

Table 13: LC08_L1TP_097084_20171211_20171223_01_T1.tar

Channel name	Min difference	Max difference	Percent different
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 6	0	0	0
BAND 7	0	0	0
BAND 8	0	0	0

Table 14: LC08_L1TP_110073_20150624_20170407_01_T1.tar

Channel name	Min difference	Max difference	Percent different
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 6	0	0	0
BAND 7	0	0	0
BAND 8	0	0	0

Table 15: LC08_L1TP_112083_20151113_20170402_01_T1.tar

Channel name	Min difference	Max difference	Percent different
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 6	0	0	0
BAND 7	0	0	0
BAND 8	0	0	0

Table 16: LC08_L1TP_115078_20130707_20170503_01_T1.tar

Channel name	Min difference	Max difference	Percent different
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 6	0	0	0
BAND 7	0	0	0
BAND 8	0	0	0

Table 17: S2A_MSIL1C_20190401T002101_N0207_R116_T55HBU_20190401T014505.zip

Channel name	Min difference	Max difference	Percent different
BAND 1	0.000 0	0.000 0	0.000 0
BAND 2	$-1.862\,6 \cdot 10^{-09}$	0.000 0	$8.294\,6 \cdot 10^{-07}$
BAND 3	0.000 0	0.000 0	0.000 0
BAND 4	0.000 0	0.000 0	0.000 0
BAND 5	0.000 0	0.000 0	0.000 0
BAND 6	0.000 0	0.000 0	0.000 0
BAND 7	0.000 0	0.000 0	0.000 0
BAND 8	0.000 0	0.000 0	0.000 0
BAND 8A	0.000 0	0.000 0	0.000 0
BAND 11	0.000 0	0.000 0	0.000 0
BAND 12	0.000 0	0.000 0	0.000 0

Table 18: S2B_MSIL1C_20190403T001109_N0207_R073_T55HBU_20190403T013318.zip

Channel name	Min difference	Max difference	Percent different
BAND 1	0	0	0
BAND 2	0	0	0
BAND 3	0	0	0
BAND 4	0	0	0
BAND 5	0	0	0
BAND 6	0	0	0
BAND 7	0	0	0
BAND 8	0	0	0
BAND 8A	0	0	0
BAND 11	0	0	0
BAND 12	0	0	0

Table 19: S2A_MSIL1C_20190723T020451_N0208_R017_T51KVA_20190723T033407.zip

Channel name	Min difference	Max difference	Percent different
BAND 1	0.000 0	0.000 0	0.000 0
BAND 2	0.000 0	$3.725\,3 \cdot 10^{-09}$	$8.294\,6 \cdot 10^{-07}$
BAND 3	0.000 0	0.000 0	0.000 0
BAND 4	0.000 0	0.000 0	0.000 0
BAND 5	0.000 0	0.000 0	0.000 0
BAND 6	0.000 0	0.000 0	0.000 0
BAND 7	0.000 0	0.000 0	0.000 0
BAND 8	0.000 0	0.000 0	0.000 0
BAND 8A	0.000 0	0.000 0	0.000 0
BAND 11	0.000 0	0.000 0	0.000 0
BAND 12	0.000 0	0.000 0	0.000 0

Of all the channels from each Satellite/Sensor combination, only two bands returned a difference in the comparison of atmospheric adjacency corrected Lambertian reflectance using two different methods, *standard convolution* and *convolution via fourier*. The difference itself is minimal, as is the tally of pixels that are different.
I would put these differences down to standard floating-point calculations, and if a 10000 scale factor is applied, the values will truncate to zero.

5 Recommendations

From the difference comparisons on the sample datasets, it is clear that we can achieve the same or otherwise a very very similar result to standard convolution when using convolution via Fourier.
The metrics gathered from each of the batch processing jobs indicate the advantage in using convolution via Fourier over standard convolution. Considering the size of the kernels could be quite large, we need to be considerate of the time it takes to process numerous datasets in an efficient manner. We can get a massive saving on CPU time, thus reducing the expenditure of KSU's. The additional memory required is not that significant, and in some cases, less.
Kicking off a test job is fairly trivial, so much so that other users could do this themselves if they have additional test areas of interest.

6 Next steps

Sample data can now be produced on a large scale, with different configurations. To progress the development of a standard packaged product, additional validation and testing are required. I would highly recommend that the previous validation exercise undertaken with CSIRO be re-visited and re-run against data produced by the *wagl* codebase. This work should involve the Calibration/Validation team, who would be expected to document the procedure and enable the work to be more easily re-evaluated when required, as well as to collate and manage any ground truth data.
For the product to be finalised for a production environment, work is needed to determine the required level of processing for the majority of downstream applications to take advantage of an Analysis Ready Data concept.
Requirements such as:

- *what datatypes?*
- *Is float32 required for sky glint corrected data in order to allow accurate sun glint correction?*
- *Does the image need to be subsetted by half the kernel size to avoid image edge effects*
- *Does the kernel need to be capped at a pre-determined x/y size?*
- *Different image padding method*
- *different method to replace null values?*