A MULTI THREADED FEATURE EXTRACTION TOOL FOR SAR IMAGES USING OPEN SOURCE SOFTWARE LIBRARIES

Bipin C[1,*] , C.V.Rao[1], P.V.Sridevi[2], Jayabharathi S[1], B.Gopala Krishna[1]

[1] National Remote Sensing Center, Hyderabad, India - (vu3bpn, cvrao909, jbharathi78, gkbarlasac)@gmail.com
[2] Department of ECE, Andhra University College of Engineering(A), Andhra Pradesh, India

Commission V, WG V/8

ABSTRACT:

In this paper, we propose a software architecture for a feature extraction tool which is suitable for automatic extraction of sparse features from large remote sensing data capable of using higher order algorithms (computational complexity greater than $O(n)$). Many features like roads, water bodies, buildings etc in remote-sensing data are sparse in nature. Remote-sensing deals with a large volume of data usually not manageable fully in the primary memory of typical workstations. For these reason algorithms with higher computational complexity is not used for feature extraction from remote sensing images. A good number of remote sensing applications algorithms are based on formulating a representative index typically using a kernel function which is having linear or less computational complexity(less than or equal to $O(n)$). This approach makes it possible to complete the operation in deterministic time and memory.
Feature extraction from Synthetic Aparture Radar (SAR) images requires more computationally intensive algorithm due to less spectral information and high noise. Higher Order algorithms like Fast Fourier Transform (FFT), Gray Level Co-Occurrence Matrix (GLCM), wavelet, curvelet etc based algorithms are not preferred in automatic feature extraction from remote sensing images due to their higher order of computational complexity. They are often used in small subsets or in association with a database where location and maximum extent of the features are stored beforehand. In this case, only characterization of the feature is carried out in the data.
In this paper, we demonstrate a system architecture that can overcome the shortcomings of both these approaches in a multi-threaded platform. The feature extraction problem is divided into a low complexity with less accuracy followed by a computationally complex algorithm in an augmented space. The sparse nature of features gives the flexibility to evaluate features in Region Of Interest (ROI)s. Each operation is carried out in multiple threads to minimize the latency of the algorithm. The computationally intensive algorithm evaluates on a ROI provided by the low complexity operation. The system also decouples complex operations using multi-threading.
The system is a customized solution developed completely in python using different open source software libraries. This approach has made it possible to carry out automatic feature extraction from Large SAR data. The architecture was tested and found giving promising results for extraction of inland water layers and dark features in ocean surface from SAR data.

1 INTRODUCTION

Remote sensing deals with large volume of data. Designing an algorithm for feature extraction from remote sensing data requires a high level of data planning. The algorithm developer needs to foresee the space and time constraints on available hardware resources. Microwave Data suffers from large speckle noise due to its mode of acquisition. When feature to be identified is sparse like water bodies or oil slicks, a lot of computation is used unnecessarily by looking at the less likely areas.[*]

Big O is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity. Collectively called Bachmannâ€“Landau notation or asymptotic notation. In computer science, big O notation is used to classify the algorithms according to how their running time or space requirements grow as the input size grows. In analytic number theory, big O notation is often used to express a bound on the difference between an arithmetical function and a better understood approximation. Big O notation characterizes functions according to their growth rates. Different functions with the same growth rate may be represented using the same O notation.

(Lee et al., 2011) describes the high-performance computing (HPC) infrastructure such as clusters, distributed networks and specialized hardware devices providing important architectural developments to accelerate the computations related with information extraction in remote sensing.

Convolution based image processing is a major class of image operations. Here the required transform is incorporated into a convolution kernel. The kernel represents the operations for the generation of single pixel in the output image. The entire output image is formed by convolution operation of the input image with the kernel. Convolution based operations are widely used due to its scalability and predictable complexity. Many fast implementation of convolution algorithm are available (Stockham, 1966)(Agarwal and Cooley, 1977).

Object Based Image Analysis (OBIA) (Blaschke, 2010) is a new emerging method in which the pixels are first segmented to form objects and the objects are further analyzed for spectral or topological signatures. It has been shown that this method has brought in radical changes in image processing for remote sensing. Softwares such as Orfeo-toolbox, eCognition,SAGA etc gives the support for OBIA are widely used for remote sensing applications.

Section 2 gives the details of the methodology used and various open source modules used in the design of this tool. Section 3 gives an overview of the program layout and their features. The results from the tool in two test cases are

[*] Corresponding author

described in section 4. A performance evaluation of the tool in GLCM calculation is discussed in Section 4.1.

2 METHODOLOGY

A lot of image processing algorithms are available in open source domain which can be used for remote sensing applications. Majority of these operations are inclined to a pixel based image processing approach where the operators are designed to map uniformly and efficiently into the entire image. When the feature of interest is small in area but distributed over the entire image, a computationally intensive algorithm to classify the feature when applied to the entire image brings in huge computation overhead. The computation can be optimized if a simple operation can select significant areas for analysis. The image processing task is split conveniently in a processing pipeline. The pipeline is designed to have gradually increasing computational complexity from input to the output. Combining connected segments into single object gives the opportunity to exploit parallel processing. A linear inequality at the input selects potential point of interest to start a segmentation operation. The segmentation step follows in the pipeline where connected pixels satisfying a goal function is grouped into a single segment. Computationally intensive algorithms follow in the pipeline for accurate classification of the segments. Parallel computing may be exploited at this stage of the pipeline and the results are gathered at the final stage of the pipeline.

2.1 FOSS modules used

The system was built completely using open software components in python language. The following are the different libraries used in the methodology.

2.1.1 Python

is a widely used open source programing language. Python is an interpreted high level programming language for general purpose programming. The language is extensively used for scientific computations due to its extensive support libraries, simple to learn syntax and clean object-oriented design. Python was selected for this task due to its support for Geospatial Data Abstraction Library (GDAL) libraries, rich set of image processing algorithms (Scikit-image) and support for parallel processing.

2.1.2 Parallel Processing

Current computer systems are multiprocessing systems where more than one Central Processing Unit (CPU) is present enabling executing of concurrent processes. Multiprocessing is a python library for process level parallelism in python programming. The library provides primitives for parallel execution, sharing objects, communication structures, parallel synchronization mechanisms etc. This module is the backbone of the entire architecture.

2.1.3 GDAL

A wide range of data formats and coordinate systems are being used by remote sensing systems. GDAL is a library for handling geospatial data. It has separate single abstract data model for handling raster and vector geospatial data for all recognized formats. It support over 155 raster drivers and 95 vector drivers. They also include utilities for Projections and Spatial Reference Systems (GDAL/OGR contributors, 2018).

2.1.4 OGR

Vector data provides a light weight mechanism for storing output of feature extraction algorithms especially when the output contain sparse spatial data with less information content. OGR is the submodule of GDAL for handling vector files. The OGR is a Simple Features Library proving read (and sometimes write) access to a variety of vector file formats. They have the capability to store descriptive information along with spatial data in a structured hierarchical database.

2.1.5 OSR

OGR Spatial Reference (OSR) provide services to represent coordinate systems (projections and datums) and to transform between them.

2.1.6 Numpy

Numpy is a fast matrix library package for scientific computing with Python. It supports a number of modules for linear algebra, Fourier transform, and random number capabilities. It also support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. (Jones et al., 2001)

2.1.7 Scikit-image

Set of image processing routines which includes many filtering and classification and other image processing routines. The library can operate directly on numpy arrays. There is a bunch of feature extraction algorithms like GLCM, Histogram of Oriented Gradients (HOG) etc. which we will be considered in the tool for performance evaluation.

2.1.8 Deep Neural Networks

These are the fine class of classifiers with exceptional accuracies. Many open source implementations are available among which Tensorflow (Abadi et al., 2015) (an open source machine learning framework from google), Keras (Chollet et al., 2015)(a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano), caffe (Jia et al., 2014) are few notable examples.

3 SYSTEM ARCHITECTURE

A feature extraction problem assumes that the feature of interest have the following characteristics.

- A simple operation can always detect a part of the feature.
- The feature is contiguous in space.
- The feature is sparsely distributed.
- The feature can be more precisely validated with a set of complex operations, auxiliary information or both.

3.1 Processing pipeline

The system is designed in a pipeline architecture where different processing elements are connected with queues. The input side of the pipeline is narrow with elements performing simple task. Towards the output side of the pipeline, the system is bulkier with complex algorithms and often work being shared with multiple threads for faster computation.

3.2 System components

A module designed to work with this pipeline should follow the principles of the system. Since each module in the pipeline is working in a parallel thread, no module should print any debugging information directly to the console . The modules should use the debug queue to communicate debugging information which will be aggregated at one thread and displayed in the user console or in a GUI. All communications between the modules are through queues. Objects that can can be exchanged between processing elements is restricted by the queue implementation.

3.2.1 Programming guidelines

All modules in the pipeline should consist the following structure so as to work in the framework.

```
load(config,in_queue,out_queue,)
   #setuup variables and load files

work_from_queue()
   data = in_queue.get()
   while data is valid:
      result = process(data)
      out_queue.put(result)
      data = in_queue.get()
```

The load function initializes all the resources required for the algorithm to run. Which mostly includes opening files for input and output operations, initialize data structures and gathering configuration for the algorithm.

3.3 Shared objects

Queues from the multithreading module is used for thread-safe synchronized communication between processes. Data objects exchanged through queues are packed using a class structure or a dictionary. The class should not contain objects like opened files that cannot be converted to a stream to be communicated through the queue. Since each module is initialized in runtime, a multiprocessing manager(shared object which make sure that reading and writing into this object is safe from all threads) is used to exchange run time configurable information between the modules.

3.3.1 Data

This module deals with reading data from data sets. GDAL library is used primarily due to its selective reading and writing functions. Data module will read a chunk of data into a numpy array based on the current area being analyzed. This avoids the need of allocating primary memory proportional to the image being analyzed The entire data module get initialized with a JSON string provided during initialization(load routine) of the the program. The module gets initialized with a base directory to search for the data file and keywords that can be used to locate the file unambiguously. On initialization, the module finds the first occurrence of the file that matches the keywords and read metadata from the file which includes the coordinate reference system used in the image, the transform matrix for transforming from map coordinates to the image pixels. In the work_fron_queue routine, a point object is received from the queue and the corresponding pixel value attribute is added/updated in the point class and pushed to the output queue. The module also has modules for tightly coupled communication without a queue mode which is used in modules where a deterministic and fast response is expected.

3.3.2 Search

Search algorithms are used to retrieve a particular information from a given dataset. Based on the information to be retrieved, the search can be designed in many configurations. For searching a single seed point, it is wise to systematically traverse the entire data so that no data is missed. For retrieving a connected area for a given point, the search should traverse in all directions.

Linear search

A search operation in the image is a function with linear complexity (number of computation for $f(n)=O(n)$). The goal function could be typically a set of comparison operations on the input value to predefined threshold values. on finding a seed point, liner search stores the current position and direction of travel to the search context and starts another search in from the end point of the image. this is to avoid initializing another search in the same contiguous region.

Higher order search

A connected neighbor search can associate an expensive algorithm for segmentation of a connected region. This search can incorporate Markov random field (MRF) model based estimators(Moser et al., 2005) or a set of linear inequality based estimators as a goal function that will result in better quality segmentation. The search is implemented as a tree search with goal function on the immediate neighbor pixels to yield children for the next level of search (Black, n.d.).

Context management

As the search can traverse in arbitrary directions depending on the data, it is required to track the points a search has traversed. Initializing another point in the already searched area will invite a lot of computational overhead.

3.3.3 Filter

A set of higher order algorithms that tells accurately whether the selected region belongs to the specified class. A large number of techniques based on GLCM, FFT can be employed in the Area of interest (AOI) selected by search. Each filter algorithm adds their result as a labels to the search result. A final decision making algorithm aggregates the individual labels and give a tag whether the segment needs to to be considered for the output. Towards the end of the processing loop, if the tag is considered for output, the result is pushed to the out queue.

3.3.4 Configuration

JavaScript Object Notation (JSON) is a lightweight data-interchange format. It is easy for humans to read and write as well as for machines to parse and generate. The different parameters used in the algorithm can be stored as a JSON script which initializes from the default settings and can be edited to any prior to execution by the user.

3.3.5 Initialization

The thread initialization and configuration is the first step in execution. Each parallel module should be initialized from a corresponding JSON string. An initializer should parse the input JSON and configure each module in a separate thread. The initialization routine waits for all the threads to complete execution.

3.3.6 Debugging

Errors and debugging is inevitable in the development of programs. A scrolling list of information should be displayed in a debugging console to know the state of execution of the algorithm. Each module at a convenient stage of execution, send a text message giving execution state or encountered values to the processing engine with a tag indicating the source(module name) and urgency of the message. A debugging task should reside in the root thread displaying and logging messages from every module base.

3.3.7 Output generation

Vector files are a convenient form of output when the results are sparse. spatial information is stored as polygons, lines or points in vector formats. It is required to convert the image data which is structured in systematic grid to be converted to lines or polygons. Delaunay Triangulation is structure to hold spatial dataset on arbitrary dimension and can be used efficiently for vector operations. Shapely (Gillies et al., 2007) is a Python package for set-theoretic analysis and manipulation of planar features.

When results needs to be in raster format, gdal libraries are used. The coordinate reference system and transform vector for defining Geo-transform is gathered from input file if not given explicitly.

3.4 System topology

Figure 1 gives a minimal setup for feature extraction using the proposed tool. the different components of the proposed system and their dependence are shown. Figure 2 shows a typical network using queue. The less latent modules are connected is series and the modules with high latency(Filters) are connected in parallel to achieve better perfomance.

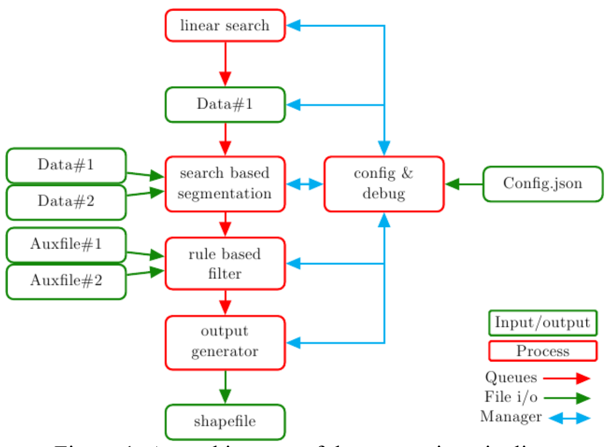

Figure 1: An architecture of the processing pipeline

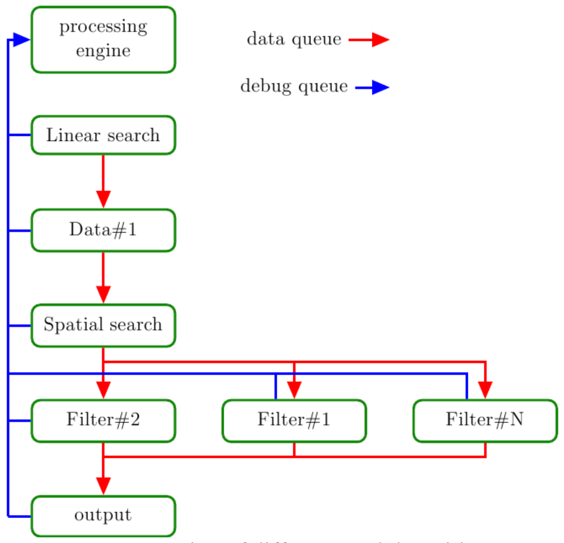

Figure 2: Interconnection of different modules with queues

4 RESULTS

This tool is used to extract water bodies from Sentinel-1A data using a level based segmentation followed by a series of GLCM shape based filters. A subset of the result is shown in figure 3 the various parameters estimated by the different filters is summarized in Table 1. The scene also includes a linear feature (stream) which is detected with discontinuity . A water extraction algorithm was tested with the proposed tool and validated against water bodies delineated from LISS-IV multispectral data. The algorithm was able to detect water bodies with commission and omission errors less than 15%.

The results of dark region extraction in RISAT-1 A data for oil spill on ocean surface is shown in figure 4. The tool was effectively used for segmentation followed by evaluation of GLCM and shape parameters for characterizing the detected segment.

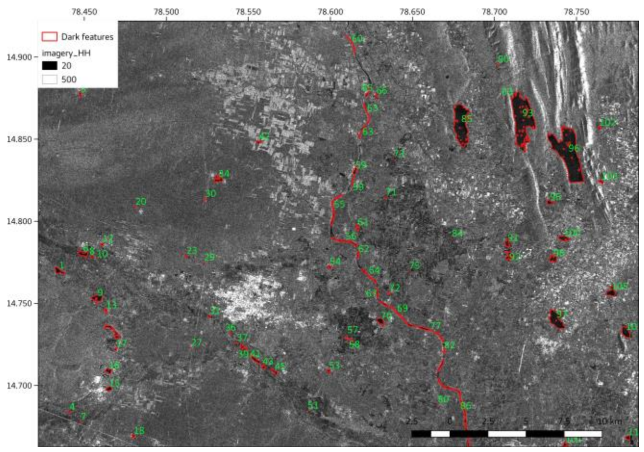

Figure 3: Output vector overlaid on input for a dark feature extraction over land using the proposed tool

ID	Area (hectare)	Perimeter (dam)	Mean		Dissimilarity	
			VV	VH	VV	VH
96	419.860	1591.46	66.41	45.96	12.56	8.71
93	340.010	2082.57	68.08	45.71	16.12	10.83
93	340.010	2082.57	68.08	45.71	16.12	10.83
85	209.190	1060.73	63.79	45.19	17.60	12.26
74	155.645	933.97	65.61	48.46	17.19	12.80
86	97.775	2266.69	61.73	47.83	1.54	1.18
97	77.110	501.42	64.89	47.12	14.87	10.96

1Table 1: Area, Perimeter, Mean and GLCM based dissimilarity measured over dark regions for significant polygons

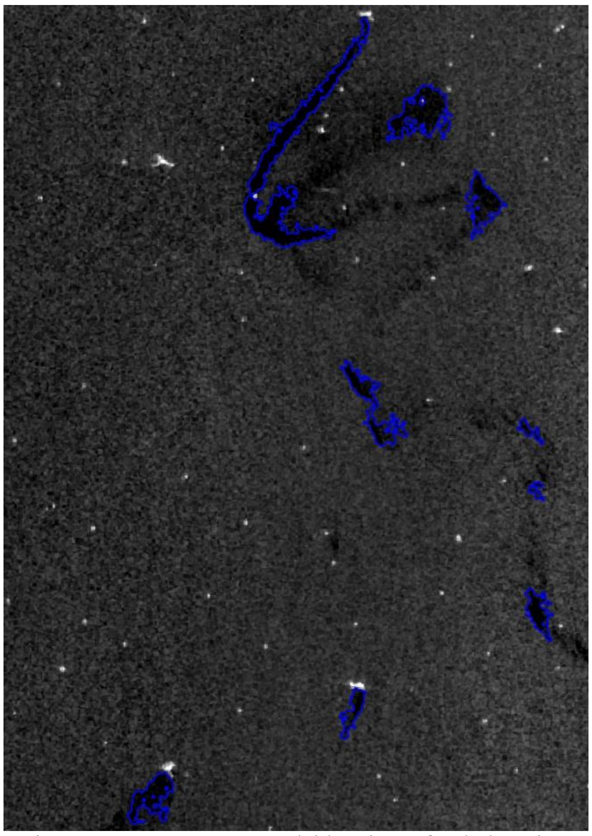

Figure 4: Output vector overlaid on input for dark region extraction from ocean using the proposed tool

4.1 Performance test

A simple task is designed to evaluate the performance of the algorithm. Calculating the GLCM matrix and find the dissimilarity along four directions at distances upto five in a 5x5 sliding window over the image where pixel value is less than 700 in RISAT 1A image. For the normal operation mode, we calculated GLCM matrix in 5x5 sliding window over the entire image and computed similarity from the GLCM matrix. The mask is calculated by comparing the dn values to a threshold. The mask is multiplied by the similarity matrix to give the required output. In the proposed pipeline, input is filtered by a comparison operation to which is send as seed points to the segmentation algorithm. A connected neighbor search collects all connected pixels satisfying the selection criteria giving the whole connected segmented region to the filter. The filter calculates the GLCM matrix and similarity measure for the input segments and writes to the output file.

The results are shown in figure 5 and the processing time for the test is shown th the table 2. It is evident that the proposed algorithm gives results in less time for the computation of the GLCM.

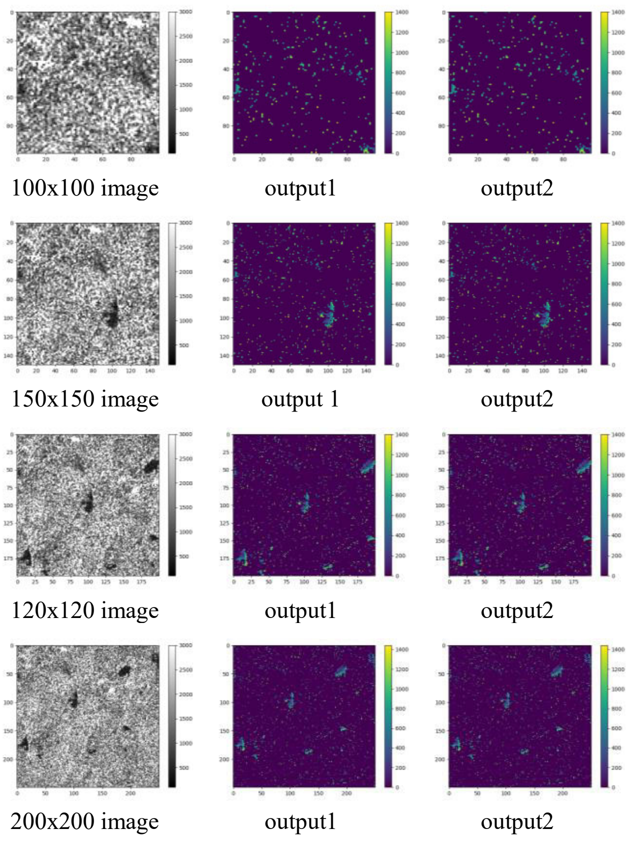

Figure 5: Output of GLCM computations, output1 is from normal matrix based operation and output2 is from the proposed tool

5 CONCLUSIONS

This approach allows to evaluate algorithms of arbitrary complexity in very large data effectively. It is giving better performance in detecting sparse features by using algorithms of high complexity in like GLCM in large data. It also helps to make algorithm execution time sensitive to valid output rather

Algorithm	Image size	Execution time (s)		Output points
		Normal	Proposed	
GLCM	100x100	82.014	3.057	369
GLCM	150x150	180.149	6.754	837
GLCM	200x200	320.939	16.083	1907
GLCM	250x250	500.714	21.543	2642

Table 2: Comparison of algorithm performance for calculation of GLCM parameters for images at different sizes

than the input data size.

6 ACKNOWLEDGMENTS

The authors sincerely thank Shri. Santanu Chowdhury, Director National Remote Sensing Center (NRSC) and Dr. Vinod M Bothale, Deputy Director, Data Processing And Web Applications Area, NRSC for their encouragement and support to carry out this work.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Agarwal, R. and Cooley, J., 1977. New algorithms for digital convolution. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25(5), pp. 392–410.

Black, P. E., n.d. Dictionary of algorithms and data structures.

Blaschke, T., 2010. Object based image analysis for remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing* 65(1), pp. 2 – 16.

Chollet, F. et al., 2015. Keras. https://keras.io.

GDAL/OGR contributors, 2018. GDAL/OGR geospatial data abstraction software library. http://gdal.org.

Gillies, S. et al., 2007. Shapely: manipulation and analysis of geometric objects.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T., 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Jones, E., Oliphant, T., Peterson, P. et al., 2001. SciPy: Open source scientific tools for Python. http://www.scipy.org/.

Lee, C. A., Gasster, S. D., Plaza, A., Chang, C. I. and Huang, B., 2011. Recent developments in high performance computing for remote sensing: A review. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* 4(3), pp. 508–527.

Moser, G., Serpico, S. B. and Causa, F., 2005. MRF model parameter estimation for contextual supervised classification of remote-sensing images. *Int. Geosci. Remote Sens. Symp.* 1, pp. 308–311.

Stockham, Jr., T. G., 1966. High-speed convolution and correlation. In: *Proceedings of the April 26-28, 1966, Spring Joint Computer Conference*, AFIPS '66 (Spring), ACM, New York, NY, USA, pp. 229–233.