SOFTWARE REQUIREMENTS SPECIFICATION

for

ICEBERG - LandCover use case

Version 1.0

Prepared by Ioannis Paraskevakos
RADICAL
Brad Spitzbart
Stony Brook University

September 12, 2018

Contents

Revision History

Name	Date	Reason For Changes	Version
Initial	8/30/2018		0.1
IP	9/12/2018	Added common section throughout the project	0.2

1 Introduction

1.1 Purpose

The purpose of this document is to capture the requirements of the ICEBERG: Seal Use Case. It will include functional, non-functional and User Interface requirements. It will be used as the reference document between the RADICAL Team and the Stony Brook team for the Seals use case development.

1.2 Document Conventions

The requested features are listed in section 4 and the non-functional requirements are listed in section 5. Each of these requirements have a priority from the set HIGH, MEDIUM, LOW. Based on the number of requirements and their priority, a timeline will be created with each requirement and its expected time-to-completion.

1.3 Intended Audience and Reading Suggestions

The document is edited and iterated between users and developers. It is intended to provide the developers as well as the project managers a complete understanding of the requirements as they are expected by the users.

An early use case document is provided in [1]. The current status of the project is provided by the use case Github repository [2].

1.4 Project Scope

LandCover is a pipeline for automated processing of satellite imagery, automated detection and removal of snow, ice, water, and shadows from the scene, automated atmospheric characterization and removal, and automated "stretching" of the scenes to provide spatial coverage of surveyed area, reasonable estimates on atmospheric contributions, and comparisons to a spectral library of known geologic materials.

1.5 References

[1] `https://github.com/iceberg-project/Use-Case-Descriptions/blob/master/LandCover/UseCase_Geology_Draft1_20171101.docx`
[2] `https://github.com/iceberg-project/LandCover`

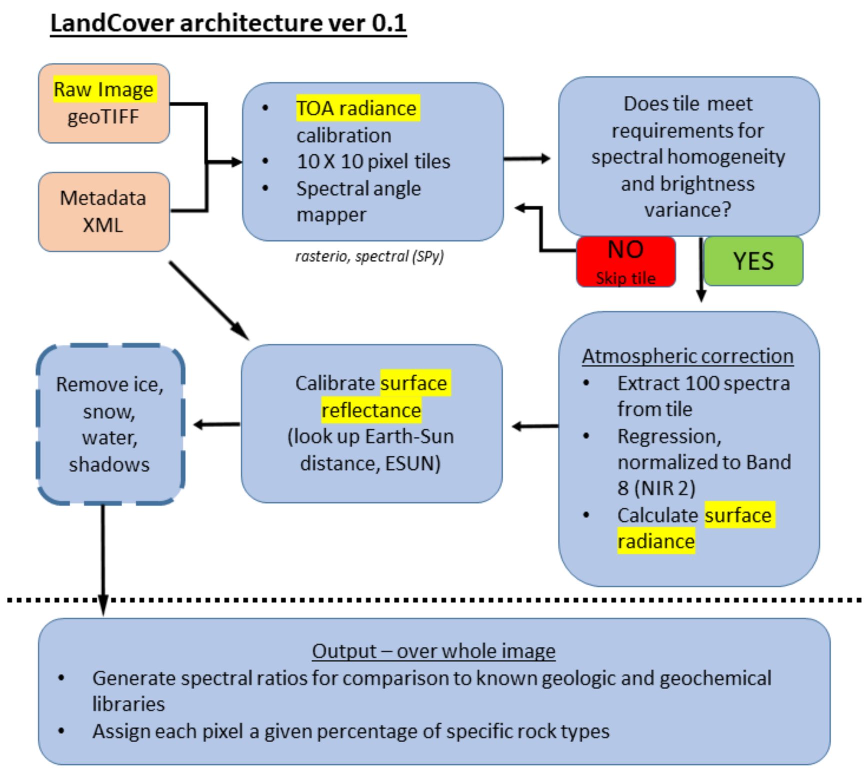
2 Overall Description

2.1 Product Perspective

ICEBERG is a multi-disciplinary, cyberinfrastructure, integration project to (1) develop open source image classification tools tailored to high-resolution satellite imagery of the Arctic and Antarctic to be used on HPDC resources, (2) create easy-to-use interfaces to facilitate the development and testing of algorithms for application specific geoscience requirements, (3) apply these tools through four use cases that span the biological, hydrological, and geoscience needs of the polar community, (4) transfer these tools to the larger (non-polar) EarthCube community for continued community-driven development.

2.2 Product Functions

The product functions consist of a series of algorithms to calibrate multispectral data from raw digital number through to spectral parameters derived from calibrated surface reflectance data. Intermediate steps include the derivation of top-of-atmosphere radiance, the estimation of atmospheric contributions and their removal from the scene, the calibration to surface reflectance, the removal of "non-geological" surfaces (e.g., ice, snow, water, shadow), the parameterization of reflectance data, and the "spectral unmixing" of orbital data using a library of spectral endmembers.



LandCover architecture ver 0.1
Raw Image geoTIFF
Metadata XML
TOA radiance calibration
10 X 10 pixel tiles
Spectral angle mapper
rasterio, spectral (SPy)
Does tile meet requirements for spectral homogeneity and brightness variance?
NO Skip tile
YES
Atmospheric correction
Extract 100 spectra from tile
Regression, normalized to Band 8 (NIR 2)
Calculate surface radiance
Calibrate surface reflectance (look up Earth-Sun distance, ESUN)
Remove ice, snow, water, shadows
Output – over whole image
Generate spectral ratios for comparison to known geologic and geochemical libraries
Assign each pixel a given percentage of specific rock types

2.3 User Classes and Characteristics

- Community users - Will use a web interface to access the capabilities of ICEBERG
- Expert users - Will use ICEBERG via command line interface to execute experiments and their use cases.
- Developers - Are users that are able to develop additional pipelines and/or change/update existing pipelines. They will be able to use the CLI or directly the resource interfaces.

2.4 Operating Environment

The software's middleware should be able to use Unix-based Operating Systems, such as Linux and MacOS. The software has library dependencies as listed in Table 2.1. *HARD* dependency to a library is restricted to the version shown. *SOFT* dependency to a library requires as a minimum version the one depicted.

The Command Line Interface should be used from a Virtual Machine that is in the Cloud and has constant Internet connection.

Library	Version	Executable	Type
Python	2.7+	All	*SOFT*
GDSAL	1.11	landcover	*SOFT*
matplotlib	2.2.2	landcover	*SOFT*
Pillow	5.1.0	landcover	*SOFT*
rasterio	Any	landcover	*SOFT*
EnTK	0.7	entk_script	*SOFT*

Table 2.1: Software Dependencies.

The Web interface should be hosted on resources with constant operation.

2.5 Design and Implementation Constraints

Access to the VM should be through SSH. The web interface should be under HTTPS protocol.

Users should have their image data uploaded to the resources via a secure data transfer system, like sftp/scp.

Users should not execute from the login nodes, unless otherwise specified explicitly by the resource documentation.

2.6 User Documentation

Users will be provided on-line documentation and help. Syntax, options, and error messages will be displayed via the web or command line interfaces.

2.7 Assumptions and Dependencies

Rasterio is assumed to be preinstalled on the resource and provided via a virtual environment. OMPI is installed and provided via a module for the launch method of RP. Python 2.7 should exist in the resource.

3 External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

Provide an interface where users can upload a set of images and get a set of images classified by landcove

The CLI interface should have the arguments based on table 3.1

Argument Name	Argument Flag(s)	Argument Type	Value	Required/ Optional
Resource	-r/–resource	String	xsede.bridges	Required
Output Directory	-op	String	'./'	Optional
Input Directory	-ip	String	/home/iparask/images	Required

Table 3.1: Command Line Interface Arguments.

3.2 Hardware Interfaces

The software system requires High Performance Computing (HPC) resources for execution. The HPC resources should provide CPU node. Any XSEDE resource is a good candicate.

3.3 Software Interfaces

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols,

electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4 System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 Selecting Parts of the Image for Atmospheric Correction

4.1.1 Description

As the image is partitioned, each partition is checked whether or not it will be used for the Atmospheric correction. If yes, it remains to the filesystem, otherwise it is discarded and removed.

4.1.2 Stimulus/Response Sequences

The stimulus is a set of partitions. The response sequence is a set of accepted partitions

4.1.3 Functional Requirements

- REQ 1: This stage can get a configurable number of partitions (1 / 10000)
- REQ 2: Read/Write data from/to local filesystem of the node.
- REQ 3: Memory usage per task should be below 4096 MBs.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

All Digital Globe (WorldView) imagery is proprietary and cannot be released publicly. Use of imagery must be in accordance with the guidelines and requirements of the Polar Geospatial Center and the NGA NextView License.

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional require-

ments in themselves, but they may imply certain functional requirements to enforce the rules.>

6 Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

6.1 Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

6.2 Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

6.3 Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>