SOFTWARE REQUIREMENTS SPECIFICATION

for

ICEBERG - LandCover use case

Version 1.0

Prepared by Ioannis Paraskevakos
RADICAL
Brad Spitzbart
Stony Brook University

August 30, 2018

Contents

Revision History

Name	Date	Reason For Changes	Version
Initial	8/30/2018		0.1

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.

2.3 User Classes and Characteristics

- Community users - web interface
- Expert users - local command line interface
- Superusers - direct XSEDE interface

2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

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

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

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.>

3.2 Hardware Interfaces

The software system requires High Performance Computing (HPC) resources for execution. The HPC resources should provide CPU and GPU node. Until now, PSC Bridges and SDSC Comet are the possible candidates.

3.3 Software Interfaces

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 3.1. *HARD* dependency to a library is restricted to the version shown. *SOFT* dependency to a library requires as a minimum version the one depicted.

Library	Version	Type
Python	3.5	*SOFT*
matplotlib	2.2.2	*SOFT*
opencv-python	3.4.1.15	*SOFT*
pandas	0.23.0	*SOFT*
spectral	0.19	*SOFT*
xml.etree.ElementTree		*SOFT*
rasterio		*SOFT*

Table 3.1: Software Dependencies.

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 System Feature 1

Calibrate to top-of-atmosphere radiance

4.1.1 Description

Input multispectral data, calibrate to top-of-atmosphere radiance using parameters drawn from image metadata file.

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1: REQ-2:

4.2 System Feature

Calculate surface reflectance by automatically applying an atmospheric correction algorithm

4.2.1 Description

1. Identify spectrally homogeneous regions with varying surface illumination and shadowing
2. Collect 200 spectra from these locations, ensuring that you are only cataloging variations in brightness (due to illumination) and not variations due to variable surface composition or other chemical/physical properties
3. Regress a line through these data at each band relative to Band 8, deriving a Y-intercept for each regression, which predicts the radiance value for each band when completely in shadow (attributed solely to atmospheric scattering)
4. Subtract these Y-intercept values from each image band to calculate surface radiance (atmosphere removed) from top-of-atmosphere radiance

4.3 System Feature

Convert to surface reflectance

4.3.1 Description

Use data from uploaded tables (Earth-Sun distance, ESUN values) to calibrate from surface radiance to surface reflectance

4.4 System Feature

Remove non-geological surfaces using derived spectral values (optional)

4.4.1 Description

1. Linearly "unmix" the image with a series of spectra from a given library that will include ice, snow, and shadowed and illuminated rock surfaces
2. Pixels meeting some threshold of non-illuminated geologic rock surfaces will be removed from the scene, leaving behind only illuminated rocky surfaces to be characterized geologically

OR, a shadow removal method to be determined

4.5 System Feature

Classify land cover geological composition

4.5.1 Description

1. Parameterize spectral data into known parameter mapping products
   a) Generate simple spectral ratios and parameter products that can be tied to known geologic and geochemical phenomena
   b) Relative values of these parameters can be used to understand the distribution of mafic compositions, oxidative weathering, and other surface properties

3. Linearly “unmix” these illuminated geological surfaces using a different endmember library, derived automatically from a library of hundreds of samples based on which library data were derived from a latitude/longitude contained within the scene, assigning each pixel a percentage of specifi rock types

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.>