# SOFTWARE REQUIREMENTS SPECIFICATION

# for

# ICEBERG - ASIFT use case

**Version 1.0**

**Prepared by Aymen Alsaadi, Ioannis Paraskevakos**
**RADICAL**
**Brad Spitzbart**
**Stony Brook University**
**Michael MacFerrin**
**University of Colorado - Boulder**
**October 17, 2018**

# Contents

# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| IP | 9/12/2018 | Adding General information | 0.1 |

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to capture the requirements of the ICEBERG: ASIFT 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 and CU Boulder teams 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

The ASIFT provide the functionality for the user to take an airborne or satellite image anywhere that we have Worldview Ortho-rectified imagery available, and where the user knows the approximate location of the photo (within 100 km) but does not have geo-location information on it. The ASIFT Workflow be able to pin down where that photo was taken, automatically determine a likely set of geo-located ground control points for the image, and ortho-rectify the image if needed.

## 1.5 References

[1] `https://github.com/iceberg-project/Use-Case-Descriptions/blob/master/ASIFT/Use_Case_ASIFT_Draft1_2018.03.16.docx`
[2] `https://github.com/iceberg-project/ASIFT` [3] `https://www.ipol.im/`

# 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

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

## 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 | 3.5 | All | *SOFT* |
| EnTK | 0.7 | entk_script | *SOFT* |
| opencv-python | 3.2 | tile_raster | *SOFT* |
| gdal-python | 2.2.1 | tile_raster | *SOFT* |
| USAC | Any | All | *SOFT* |
| liblapack-dev | Any | All | *SOFT* |
| libblas-dev | Any | All | *SOFT* |
| libconfig-dev | Any | All | *SOFT* |
| libconfig++ | Any | All | *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 or GSISSH. 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 instead the user should execute the job through queue system.

## 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

OpenCV version 3.2.0 and GDAL v2.2.1 is assumed to be compiled and preinstalled on the resources.In order to use SIIM descriptors proposed by OpenCV and inorder to use GEOTIFF imagery. Also Python 2.7 and 3.5 should exist in the resource.

# 3 External Interface Requirements

## 3.1 User Interfaces

Provide an interface where users can upload an image and get back a table with Ground Control Points (GCPs) and ortho-rectified image.

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 and high amount of Memory(RAM). Any XSEDE resource is a good candidate.

## 3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## 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

<Don't really say "System Feature 1." State the feature name in just a few words.>

### 4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

### 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 2 (and so on)

# 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

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## 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 requirements 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.>