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 26, 2018

Contents

Revision History

Name	Date	Reason For Changes	Version
IP	9/12/2018	Adding General information	0.1
AA,BS	10/26/2018	Adding product functions	0.2

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

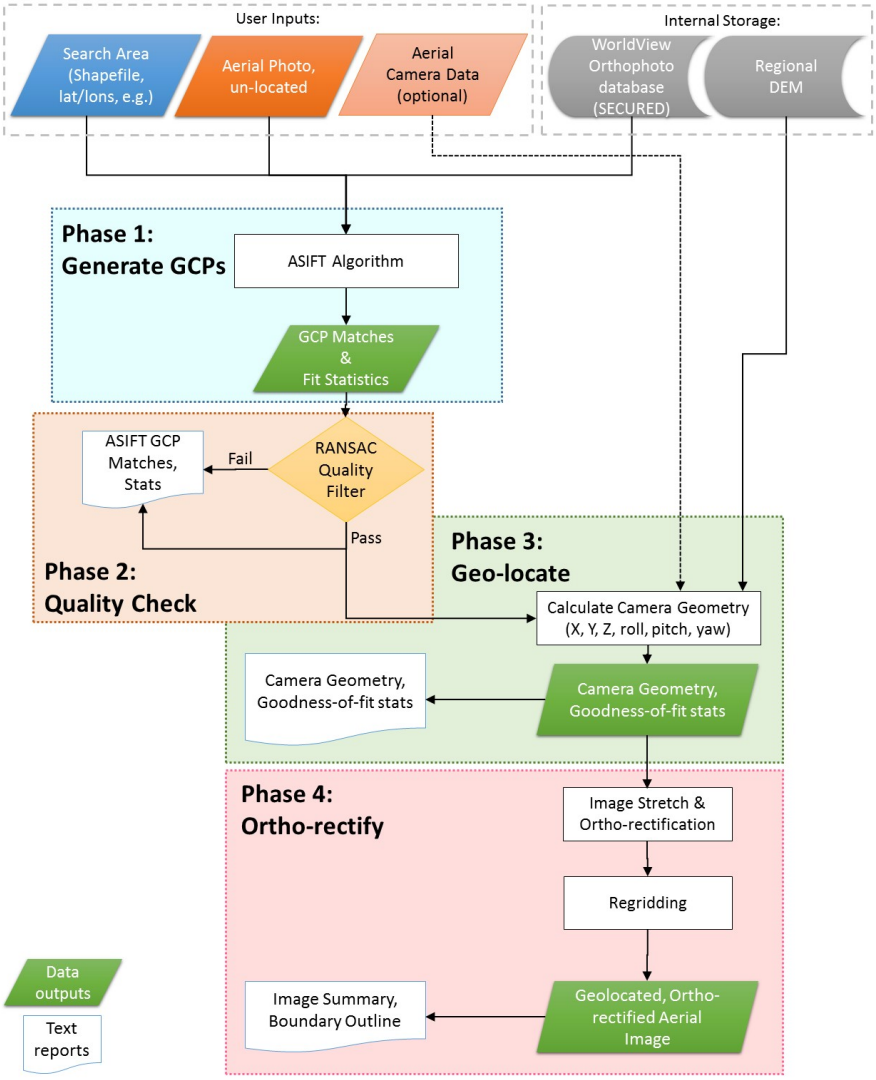
[1] `https://github.com/iceberg-project/Use-Case-Descriptions/blob/master/ASIFT/Use_Case_ASIFT_Draft1_2018.03.16.docx`
[2] `https://github.com/iceberg-project/ASIFT` [3] `https://www.ipol.im/`

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



The main functions of the ASIFT pipeline are generating ground control points (GCPs), quality checks (RANSAC), geo-location, ortho-rectification, and producing output images.

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.

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.

- 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.

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

There are no performance requirements as of now.

5.2 Safety Requirements

All input data are treated as read only. They should not be deleted.

5.3 Security Requirements

All Digital Globe (WorldView) imagery is proprietary and cannot be released publically. 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

The software should be accompanied with detailed documentation, and examples that demonstrate its usage. In addition, the source code should publicly available through Github.