SOFTWARE REQUIREMENTS SPECIFICATION

for

ICEBERG - Rivers use case

Version 1.0

Prepared by Ioannis Paraskevakos
RADICAL
Brad Spitzbart
Stony Brook University
Vena Chu
UC Santa Barbara

October 26, 2018

Contents

Revision History

Name	Date	Reason For Changes	Version
Initial	9/28/2018		0.1
Chu	10/26/2018		0.12

1 Introduction

1.1 Purpose

The purpose of this document is to capture the requirements of the ICEBERG: Rivers 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 UCSB teams for the Rivers 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

We provide a classification algorithm for ice surface features from high-resolution imagery. This algorithm was developed by convolutional neural network training to detect regions of large and small rivers and to distinguish them from crevasses and non-channelized slush. We also provide a detection algorithm to extract polyline water features from the classified high-probability river areas.

1.5 References

[1] `https://github.com/iceberg-project/Rivers/blob/master/HOWTO_StreamExtraction.txt`
[2] `https://github.com/iceberg-project/Rivers`

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 Rivers pipeline are conversion of orthorectified imagery into classified ice surface features including: Rivers (large rivers and lakes), Streams (small rivers), Crevasses, Slush. The Rivers and Streams class will mask areas for producing NDWI: Normalized difference water index images, extracting large stream profiles, and extracting lake profiles.

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*
matplotlib	2.2.2	Verify	*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

OMPI is installed and provided via a module for the launch method of RP. 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 a set of images and get back a table with the water features.

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/–output_dir	String	’./’	Optional
Input Directory	-ip/–input_-dir	String	/home/iparask/images	Required
Path to Model	-m/–model	String	model file	Optional
Walltime	-w/–walltime	Interger	60	Optional
User Name	-u/–username	String	’iparask’	Required

Table 3.1: Command Line Interface Arguments.

3.2 Hardware Interfaces

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

3.3 Software Interfaces

None at this time.

3.4 Communications Interfaces

None at this time.

4 System Features

This section describes the features of the Rivers use case. Each feature has a description, its stimulus—its input sequence or event, response sequence, and its functional requirements.

4.1 Classifying images in bulk

4.1.1 Description

The Rivers component should take as an input a set of images. These images will be organized in a single folder. Each analysis is independent of each other.

4.1.2 Stimulus/Response Sequences

The stimulus of this feature will be the path to a folder where all the images will exist. The response sequence is a set of rasters (one of each feature class) indicating the probabilities of each feature class

4.1.3 Functional Requirements

The following list displays the requirements of this feature. All requirements have the same priority unless otherwise stated.

REQ-1: Identify the individual images that exist in a single folder.

REQ-2: Each image should have an individual set of output raster files (one for each feature class) indicating class probabilities.

4.2 Extracting river and stream centerlines

4.2.1 Description

The Rivers component should take as an input a set of images and the associated raster probabilities for the Rivers and Streams classes. Each analysis is independent of each other.

4.2.2 Stimulus/Response Sequences

The stimulus of this feature will be the path to a folder where all the images will exist, and another path to the feature class raster probabilities. The response sequence is a set of shapefiles of River and Stream polylines.

4.2.3 Functional Requirements

The following list displays the requirements of this feature. All requirements have the same priority unless otherwise stated.

REQ-1: Identify the individual images that exist in a single folder, and their associated Rivers and Streams raster probabilities existing in two other folders.

REQ-2: Convert images to NDWI and perform linear feature extraction in high probability areas of Rivers and Streams.

REQ-3: Each image should have an individual output shapefile with attributes indicating a River polyline segment or a Stream polyline segment

4.3 Extracting lake polygons

4.3.1 Description

The Lakes component should take as an input a set of images and the associated raster probabilities for the Rivers class. Each analysis is independent of each other.

4.3.2 Stimulus/Response Sequences

The stimulus of this feature will be the path to a folder where all the images will exist. The response sequence is a set of rasters (one of each feature class) indicating the probabilities of each feature class.

4.3.3 Functional Requirements

The following list displays the requirements of this feature. All requirements have the same priority unless otherwise stated.

REQ-1: Identify the individual images that exist in a single folder, and their associated Rivers raster probabilities existing in another folder.

REQ-2: Convert images to NDWI and perform lake polygon extraction using high probability areas of Rivers.

REQ-3: Each image should have an individual output shapefile of lake polygons.

4.4 Output seasons

***brad: Shape files for each image as they are produced and an CSV with the results for one season.

4.4.1 Description and Priority

The Rivers component should output results per season if need be.
Priority: 5

4.4.2 Stimulus/Response Sequences

The stimulus is a flag during runtime. The response would be to output the results of images as they are produced.

4.4.3 Functional Requirements

The following list displays the requirements of this feature. All requirements have the same priority unless otherwise stated.

REQ-1: Create an input defining a season. The images of the season are prioritized over the rest of the images.***giannis: @Brad: how do we understand which images belong to a season?

REQ-2: Create a file with the results of the season with two columns: Image filename, id of water feature

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.