SOFTWARE REQUIREMENTS SPECIFICATION

for

ICEBERG - Ice Wedge Polygon use case

Version 1.0

Prepared by Brad Spitzbart
Stony Brook University

July 2, 2019

Contents

Revision History

Name	Date	Reason For Changes	Version
BS	7/02/2019		0.1

1 Introduction

1.1 Purpose

The purpose of this document is to capture the requirements of the ICEBERG: ICE Wedge Polygon Use Case. It will include functional, non-functional and User Interface requirements. It will be used as the reference document between the RADICAL Team, Stony Brook team and UCONN team for the Ice Wedge Polygon 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.

The current status of the project is provided by the use case Github repository [1].

1.4 Project Scope

We provide a detection algorithm to map the location of ice wedge polygons from high-resolution imagery. This algorithm was developed by convolutional neural network training to detect ice wedge polygons. This is beneficial to understand the complex and interlinked processes responsible for the evolution of the pan-Arctic permafrost polygonal tundra.

1.5 References

[1] `https://github.com/iceberg-project/Ice_wedge_polygons`

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 Seals pipeline are tiling, detection, counting, and reporting the location of seals in a set of 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.
- 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
CUDA	8.0	predict_sealnet	*HARD*
Python	3.5	All	*SOFT*
matplotlib	2.2.2	Verify	*SOFT*
opencv-python	3.4.1.15	tile_raster	*SOFT*
pandas	0.23.0	predict_sealnet	*SOFT*
Pillow	5.1.0	predict_sealnet	*SOFT*
torch	0.4.0	predict_sealnet	*SOFT*
torchvision	0.2.1	predict_sealnet	*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

CUDA v8.0 is assumed to be preinstalled on the resource and provided via a module. 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 Seals.

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
Input Directory	-ip/–input_dir	String	/home/iparask/images	Required
Project	-p/–project	String	TG-MCB000000	Required
Queue	-q/–queueu	String	RM	Required
CPUs	-c/–cpus	Integer	1	Optional
GPUs	-g/–gpus	Integer	1	Optional
Output Directory	-op/–output_dir	String	’./’	Optional
Path to Model	-m/–model	String	model file	Optional
Walltime	-w/–walltime	Integer	60	Optional

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

<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.>

Not applicable as of now.

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.>

Not applicable as of now.

4 System Features

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

4.1 Analyzing images in bulk

4.1.1 Description and Priority

The Seals 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.
Priority: 9

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 table with the number of Seals per image.

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 output file

REQ-3: Provide an aggregated output file for all images with two columns: Image filename, number of seals

4.2 Output seasons

4.2.1 Description and Priority

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

4.2.2 Stimulus/Response Sequences

The stimulus is a flag during runtime. The response would be to output the results of images as they are produced. Those are shape files for each images as well as a single CSV file with the results of the season

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: Create an input defining a season. The images of the season are prioritized over the rest of the images.

REQ-2: Create a file with the results of the season with two columns: Image filename, number of seals

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.