GAL: A Stepwise Model for Automated Cloud Shadow Detection in HICO Oceanic Imagery Utilizing Guided Filter, Pixel Assignment, and Geometric Linking

Jennerpher Renee Meyers
University of Southern Mississippi

The University of Southern Mississippi

GAL: A STEPWISE MODEL FOR AUTOMATED CLOUD SHADOW DETECTION IN HICO OCEANIC IMAGERY UTILIZING GUIDED FILTER, PIXEL ASSIGNMENT, AND GEOMETRIC LINKING

by

Jennerpher Renee Meyers

Abstract of a Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

May 2015

ABSTRACT

GAL: A STEPWISE MODEL FOR AUTOMATED CLOUD SHADOW DETECTION IN HICO OCEANIC IMAGERY UTILIZING GUIDED FILTER, PIXEL ASSIGNMENT, AND GEOMETRIC LINKING

by Jennerpher Renee Meyers

May 2015

Detection of cloud shadow pixels is an important step in image processing in several remote sensing ocean-color application domains, such as obtaining chlorophyll content. While shadow detection algorithms do exist, the vast majority are for over land which leaves few options for detection over water.

The detection of cloud shadow over water in HICO imagery is a unique problem. As its name implies, HICO (Hyperspectral Imager for the Coastal Ocean) imagery is produced for coastal and oceanic regions. Since land based algorithms remove water before processing, these approaches would not be applicable. The only currently published HICO shadow pixel detection algorithm [1] produces good results for predominantly homogeneous regions. It also involves hand-tuning of the parameters, which is not suitable for automation.

GAL is a fully automated stepwise model that starts by using satellite imagery and navigational data. The next step is applying the guided filter algorithm proposed by He, Sun, and Tang [2] to these images in order to filter and enhance the images before shadow detection. The third step classifies pixels into water, land, and clouds. The fourth step uses cloud shadow geometry to indicate possible shadow pixels. The final step is to reduce the amount of possible shadow pixels to the most probable shadow pixels.

This research combines the past techniques of cloud shadow geometry, edge detection, and thresholding, along with the new techniques of guided image filtering, in such a way that has never been done before. GAL works best with well-defined cloud shadows that contain a large contrast between water and shadow. Water type, coastal or deep ocean, does not affect GAL. Shadows with a large gradient may be under-detected. GAL can be applied to HICO data immediately, with the potential of being applied to all global high resolution ocean-color satellite imagery.

The University of Southern Mississippi

GAL: A STEPWISE MODEL FOR AUTOMATED CLOUD SHADOW DETECTION IN HICO OCEANIC IMAGERY UTILIZING GUIDED FILTER, PIXEL ASSIGNMENT, AND GEOMETRIC LINKING

by

Jennerpher Renee Meyers

A Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved:

Dr. Beddhu Murali
Committee Chair

Dr. Dia Ali

Dr. Chaoyang Zhang

Dr. Ras Pandey

Dr. Sean McCarthy

Dr. Karen Coats
Dean of the Graduate School

May 2015

DEDICATION

To my daughter Alexandria, Mommy is finally finished.

ACKNOWLEDGMENTS

Special thanks go to my committee chair, Dr. Beddhu Murali, for his immense patience, guidance, and advice. I could not have asked for a better major professor. I would like to thank Dr. Sean McCarthy for bringing this research to my attention, for talking me down from ledges, and for always being supportive. My thanks go to Dr. Dia Ali for patiently sitting through my presentation over and over again, for giving pages of notes until I was truly ready, for giving me a cubicle to hide in, and for always being there when I needed an ear. I also thank Dr. Joe Zhang, and Dr. Ras Pandey for their time serving on my committee, making sure I was well-rounded as a candidate.

I would also like to thank Dr. Tim Rehner and Dr. Daniel Bond for being the best bosses a graduate student could ever have. Thanks for the support, the encouragement, and the time off. Thanks to Dr. Glenn Bond for his support and encouragement, another ear that was truly needed and appreciated. Thanks to Dr. Ray Seyfarth, who allowed me to bother him incessantly, even though he is retired. Good professors never quit teaching.

To the office staff of both the School of Computing and the School of Social Work along with my family and friends, thank you for everything you do every day. Without you none of this could have been done.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ABBREVIATIONS

APS	Automated Processing System
ASB	Adaptive Sliding Box
AVHRR	Advanced Very High Resolution Radiometer
CSDI	Cloud Shadow Detection Index
GF	Guided Filter
HICO	Hyperspectral Imager for the Coastal Ocean
ISS	International Space Station
IV	Integrated Value
MODIS	Moderate Resolution Imaging Spectroradiometer
NRL	Naval Research Laboratory
UTM	Universal Transverse Mercator

CHAPTER I

BACKGROUND

Satellite Image Processing

Since the late 1950's there have been thousands of satellites sent into space. The roles and research of each of these satellites can differ greatly. Satellite images can be used for research in agriculture, geology, forestry, landscape, education, intelligence, and warfare, to name a few. The satellite imagery used in this research is from the Hyperspectral Imager for the Coastal Ocean, which has many earthly applications.

Figure 1. HICO on ISS [3].

Entities such as the U.S. Navy and U.S. Marine Corps, as well as the EPA and other civilian researchers can utilize HICO data to determine end products such as water visibility, the ocean's depth and floor, and chlorophyll content, among others. These end products are useful when there is a need to test for water visibility for moving troops or to test for pollution through monitoring water quality. [4]

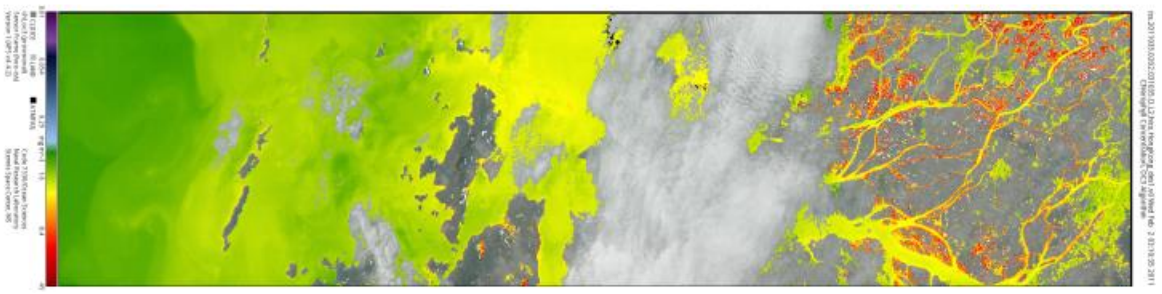

Figure 2. Chlorophyll Content of Hong Kong on February 2, 2011.

The information gained from each satellite through its sensors is tailored to the specific goals of each satellite. These sensors retrieve data along certain wavebands or range of wavebands of wavelengths, sometimes labeled as channels. Some satellite sensors have fewer channels and/or different wavelengths than HICO. One such sensor Moderate Resolution Imaging Spectroradiometer (MODIS) is located on both satellites Terra and Aqua. The data obtained from MODIS on these two satellites “will improve our understanding of global dynamics and processes occurring on the land, in the oceans, and in the lower atmosphere. MODIS is playing a vital role in the development of validated, global, interactive Earth system models able to predict global change accurately enough to assist policy makers in making sound decisions concerning the protection of our environment.” [5] MODIS has 36 channels covering wavelengths from 400 nm to 14,400 nm where HICO has 127 covering 400 nm to 1080 nm. Other satellites may also retrieve data that HICO was not designed to retrieve. Advanced Very High Resolution Radiometer (AVHRR), is used mainly for cloud detection and surface temperatures, HICO does not include surface temperatures. Although HICO is maintained by NRL, MODIS by NASA, and AVHRR by NOAA, many different sectors of the U.S. Government utilize the data from the satellites. [5] [6] Even within one entity, the diverse data collected from many different satellites can be used to obtain many similar, yet very different goals.

With the large difference in the type and amount of data received and processed by each sensor, at least one thing is common: clouds. If clouds are present in an image, there is always a possibility of cloud shadow within that image as well. Cloud shadows may hinder or help in end products. They may hinder by skewing end products; shadow can produce errors of 30% – 40% over land. It is expected that similar errors would happen over water as well, even though there has been no published research to date. [7] On the other hand, a recently published article by Amin et al. [8] states that shadow detection can aid in atmospheric correction algorithms in HICO. Identifying the clouds and their corresponding shadows is imperative when using satellite imagery and data in any research.

HICO and NRL

HICO was built by the Naval Research Laboratory (NRL), which is "the corporate research laboratory for the Navy and Marine Corps and conducts a broad program of scientific research, technology and advanced development." [10] HICO is located on the International Space Station (ISS). NRL is also responsible for maintaining, operating, and processing data retrieved from the HICO sensor. [9] Members of the Bio-Optical/Physical Processes and Remote Sensing Section of NRL are responsible for creating and maintaining the Automated Processing System (APS). This system contains the algorithms for calibrations and atmospheric corrections applicable to HICO data. It also contains the algorithms for processing the raw level 1 data, top-of-atmosphere satellite radiances, into the geo-referenced level 3 downstream optical properties. [11]

HICO was created to combat difficult constraints on retrieving useful sensor data of coastal areas. According to Corson et al. [12], to accomplish the goals of bathymetry, bottom type, chlorophyll content, and water inherent optical properties, a sensor must be hyperspectral and well-calibrated. The need for a hyperspectral sensor comes from the fact that the albedo, the portion of light or radiation reflected by a surface, is only a few percent. The low albedo is due to water being more absorbent of sunlight than land, snow, or ice as seen in Figure 3 below.

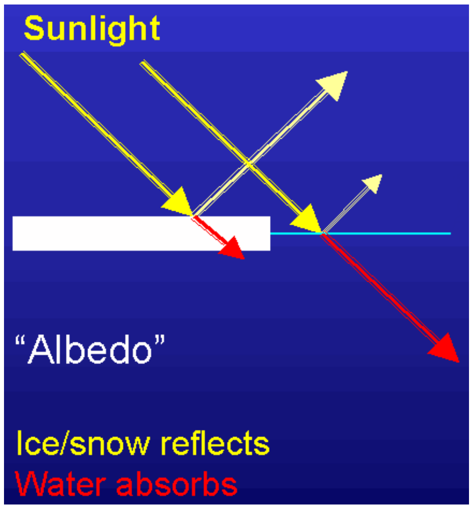

Figure 3. Visual Representation of Albedo: It should be noted that land reflects sunlight similarly to ice and snow shown in this illustration. [13].

HICO was designed and calibrated to handle the issue of the atmosphere being significantly brighter than the water surface. HICO data is processed through APS to adjust for any remaining atmospheric distortions. APS has and is continually updated with code to handle atmospheric corrections, among other transformations and end products. One addition APS could greatly benefit from is a fully automated shadow detection algorithm that works over coastal and homogenous water.

Previous Shadow Detection

The available literature on shadow pixel detection can be divided into two categories: detecting shadows over land or over water. While shadow detection algorithms do exist, the vast majority are for land. For example, in [14], a thresholding technique is used to identify shadow, as well as water in the image must be identified and separated from shadow scenes. This algorithm uses the small variance of water in their images to identify the difference between shadow and water. Once the water is identified it is removed from the scene. Another algorithm designed to detect and remove cloud shadow over land was designed for AVHRR data. [7] This algorithm uses shadow geometry with an iterative tile projection. Simpson states that all water pixels should be removed before inputting the image, which, again, means it cannot be applied to HICO. Most of the available literature applies either one or more algorithms that include geometry, thresholding, and/or classification techniques. [14] [15] [7] These algorithms do not apply to HICO data because these algorithms, like most land algorithms, remove water before applying the algorithm or as in [14] treat water in such a way that is not conducive with coastal regions.

One of two cloud shadow detection over water algorithms was published by Jiang and Wang. [16] This algorithm is not on HICO data but on MODIS – Aqua and SeaWiFS. Although this algorithm will not be applicable to HICO in its current form, it is important to mention it. The intent of Jiang was to extend the cloud shadow detection algorithm already contained in IDPS, a program similar to NRL's APS. They use a static size box to identify pixels around a cloud. Their box size is about 25x25 pixels around the cloud itself. This value was chosen by incorporating shadow geometry. Due to the resolution size of MODIS it was calculated that the shadow path would be about 12 pixels. This would be very different for HICO where shadow paths can be up to 100 pixels. They use the shadow geometry to identify the direction in which to test for shadow pixels. Once the direction was calculated, they could determine if the pixels were shadows or not by using radiance values from the wavelength 551. HICO does not have this wavelength; however, it does have wavelengths 547 and 553.

The only algorithm currently published addressing HICO imagery shadow detection is “Optical Algorithm for Cloud Shadow Detection Over Water” by Ruhul Amin et al. [1] The algorithm that is presented uses a constant threshold by normalizing the integrated value of radiance data of the blue and green spectra of a chosen pixel by the mean of the integrated values of the pixels in an Adaptive Sliding Box (ASB). Static boxes have been used prior to Amin’s et al. work. The chosen pixel lies in the center of the box. The algorithm works well but has some restrictions. First, the box size must be chosen, so the box is larger than the shadow area. The next constraint is that the water must be homogenous. The difficulty with heterogeneous waters, which are quite often the case of coastal waters, is that shadow over a light section of water may be lighter than a darker area in another part of the image.

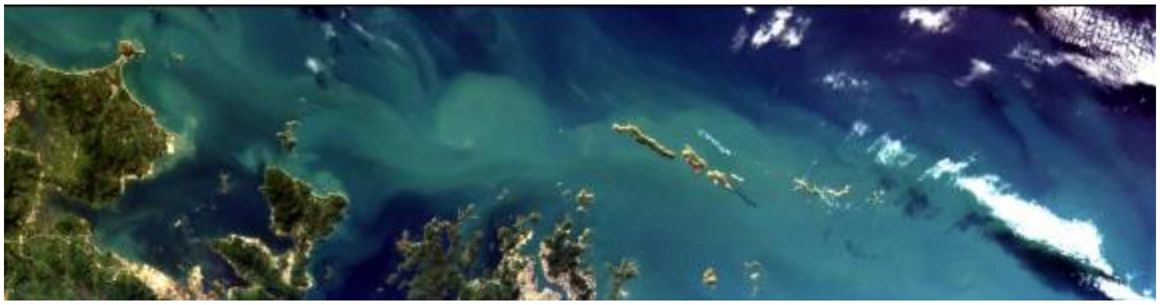

Figure 4. Hong Kong Raw Image.

Figure 4 is a perfect example of this situation. Note that some of the shadow on the right side of the image is much lighter than some of the dark water on the left of the image. This is the reason a threshold over the entire image will not work.

The algorithm designed in [1], starts by using the raw data from HICO. Each pixel has 127 radiance values. Amin et al. utilizes the radiance values from the bands that range 400 nm to 600 nm. These 35 bands represent the values contained within the blue and green spectrum. He represents this as the parameter IV. Amin et al. states that although the IV can visually show a difference between shadowed and non-shadowed regions, it is not strong enough for a constant threshold to be applied to the image.

$$IV = \int_{400\,nm}^{600\,nm} Lt(\lambda)d\lambda$$

Figure 5. IV: The integrated radiance values from bands 400 nm to 600 nm. [1]

Once the radiance values are extracted, the ASB is selected. This box is used to select pixels around the pixel in question. The size is arbitrary and is chosen so that both sunlit and shadow pixels are included. One of the drawbacks of this algorithm is the user interaction needed to complete this step of the algorithm. Once the ASB has been chosen (32x32 to 128x128), the IV of the ASB must be calculated. This allows for the CSDI, or cloud shadow detection index, to be calculated. The CSDI, as seen in the figure below, is used with a threshold to identify cloud shadow pixels.

$$CSDI = \frac{IV_c}{\langle IV_{ASB} \rangle}$$

Figure 6. CSDI: IV of the selected pixel divided by the mean of the IVs of the ASB. [1]

Amin et al. states that this index may break down in coastal waters. This breakdown is the reason for the specification of homogenous waters, which is another drawback for this method. Once the CSDI has been calculated, the threshold is applied. Amin states that through a visual inspection CSDI <= 0.95 is too low to detect relatively thin parts of the shadows while CSDI => 0.97 is a little high and gives a false signal. This leads to the threshold being <=0.96. The figure below is the results of [1].

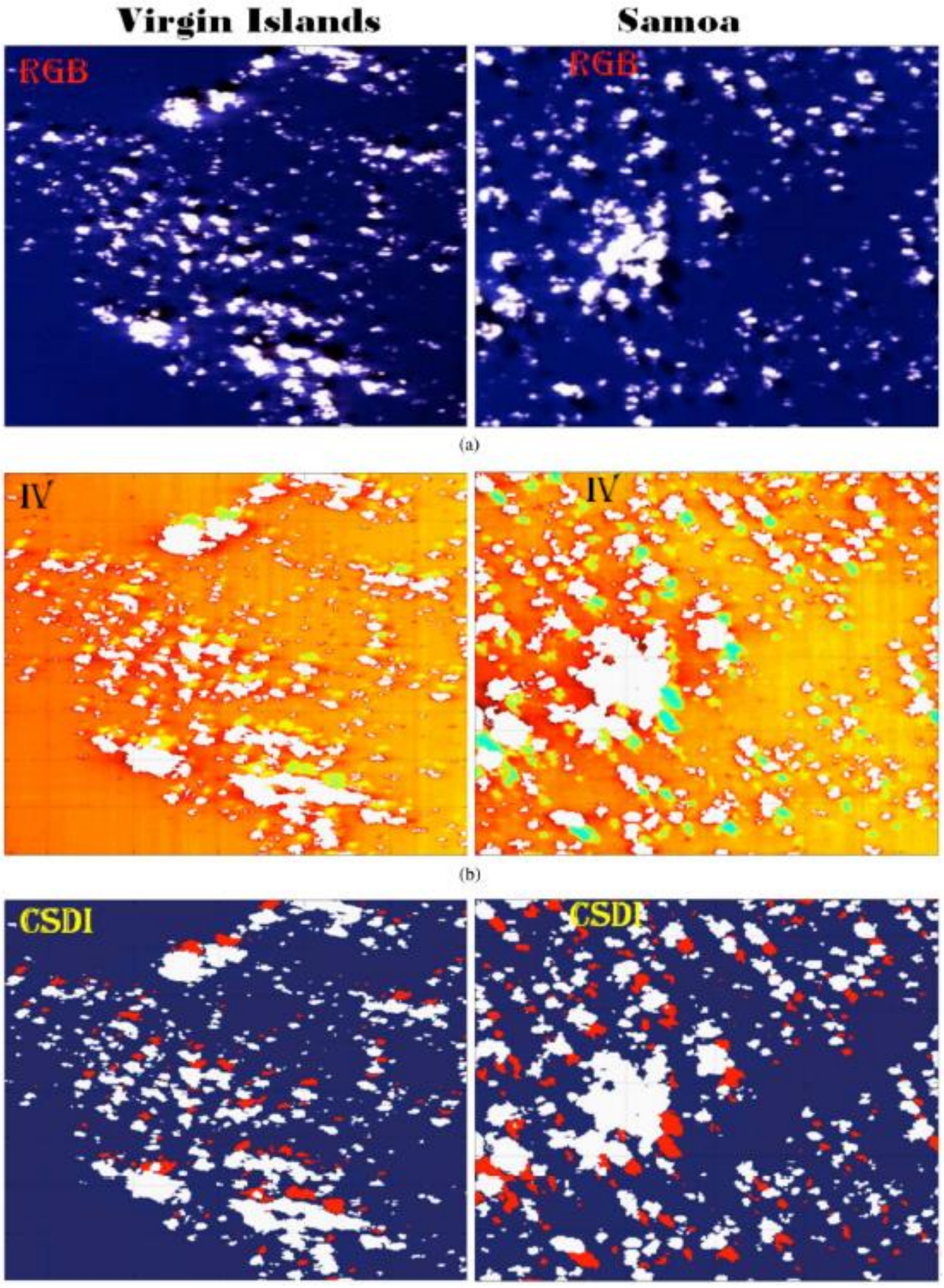

Figure 7. Results Optical Algorithm for Cloud Shadow [1].

As stated before, Amin’s et al. algorithm seems to yield decent results in homogenous water with arbitrary parameters set by a user. The method contained in this research will be able to handle both types of water and be fully automated, which should not only improve previous research, but will also extend that research into new areas.

These two algorithms show that research in this area is being done, and automated cloud shadow detections over water are in great demand. This highlights an area of research that needs further expansion. GAL does just that. It takes the methods that have shown to work in previous research and combines them with new techniques in a unique way to produce a model of cloud shadow detection over water that does not exist. Because of the potential to work with any satellite data and full automation, GAL is a contribution that is sorely needed.

CHAPTER II

RESEARCH ALGORITHM

Introduction

The algorithm designed for this research was constructed for use with HICO data. Although HICO data was the initial intent, the algorithm has the potential to work with any satellite data as long as an image and navigational data is available. Using three images for the development phase seen in Figure 8, Figure 9, and Figure 10, GAL was designed to take a stepwise approach to detecting cloud shadow.

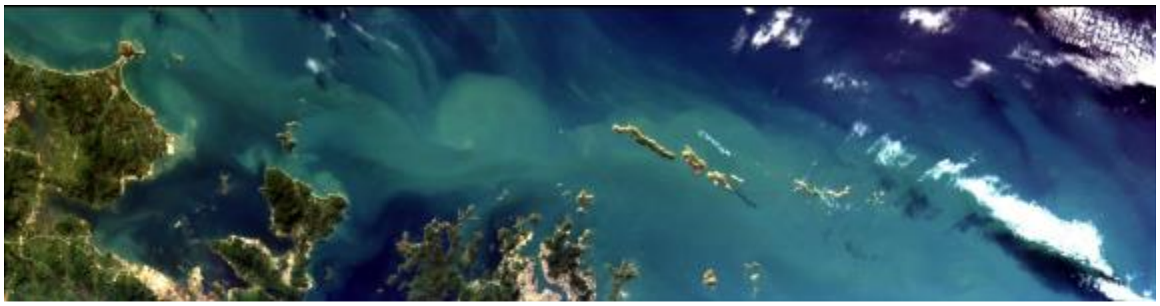

Figure 8. Hong Kong Raw Image 2009.

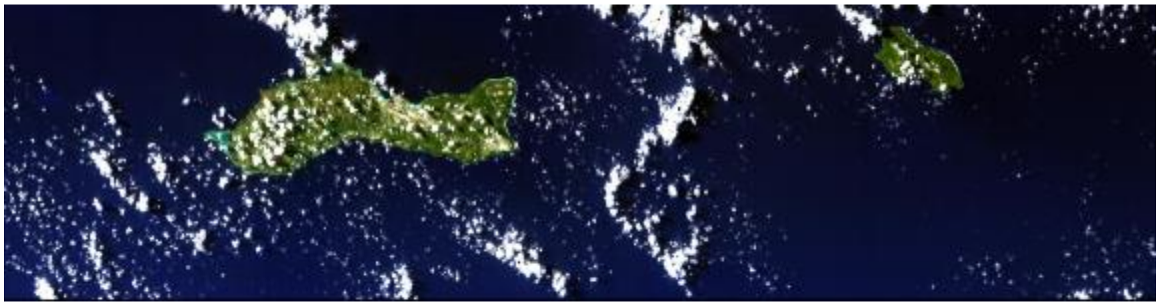

Figure 9. Guam Raw Image 2009.

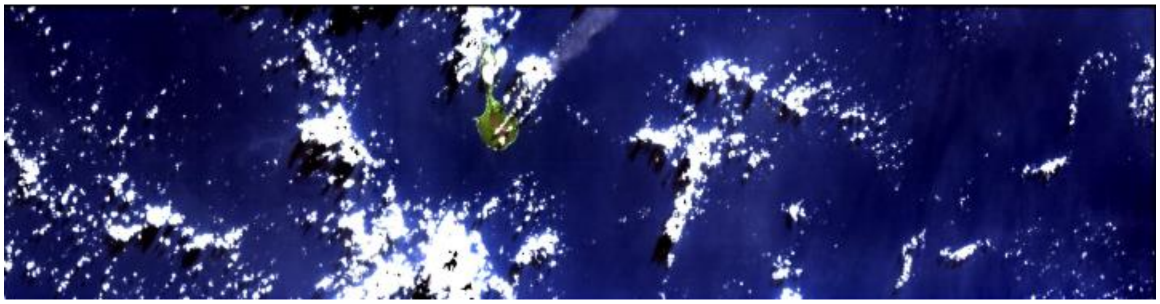

Figure 10. N. Mariana Pagan Raw Image 2009.

The algorithm flowchart is shown in Figure 11. This algorithm takes a true color image of any format and applies Guided Filter; .png file format was used. The enhanced image is processed for a categorization of land, water, and cloud pixels. Once cloud pixels are identified, cloud geometry is applied to find geometrically possible shadow pixels. These possible shadow pixels are reduced to the most likely shadow pixels through utilizing an edge neighbor minimum value threshold.

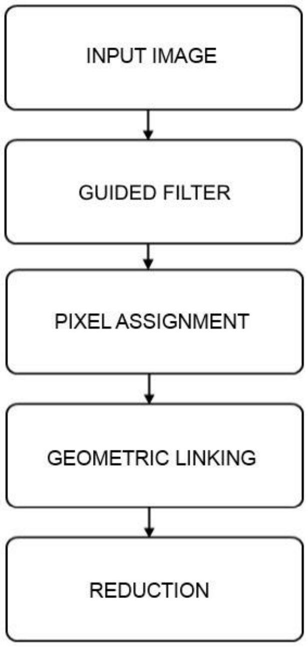

Figure 11. Stepwise Algorithm.

Step 1: Input Image: Image and Data Processing Tools

The data obtained from NRL for the purposes of this research includes a true color image in .png file format and a file of raw data in .hdf file format from different available locations and times. The .hdf file contains all the raw data obtained by HICO, which includes but is not limited to radiance values, navigational data, flags, and a header file. The information extracted and used in this research from the .hdf file is the navigational information: each pixel's latitude, longitude, sensor zenith and azimuth, and solar zenith and azimuth. Due to the file formats, Matlab was chosen to be the best environment to create the code for this research. Matlab is able to handle image reading, writing, and manipulation along with reading .hdf files easily. The navigational data was extracted from the .h5 file using the following code:

```
solar_zenith = h5read(input, '/navigation/solar_zenith');
solar_zenith = permute(solar_zenith,[2 1]);
rad_solar_zen = deg2rad(solar_zenith);
solar_azimuth = h5read(input, '/navigation/solar_azimuth');
solar_azimuth = permute(solar_azimuth,[2 1]);
rad_solar_az = deg2rad(solar_azimuth);
sensor_zenith = h5read(input, '/navigation/sensor_zenith');
sensor_zenith = permute(sensor_zenith,[2 1]);
rad_sensor_zen = deg2rad(sensor_zenith);
sensor_azimuth = h5read(input, '/navigation/sensor_azimuth');
sensor_azimuth = permute(sensor_azimuth,[2 1]);
rad_sensor_az = deg2rad(sensor_azimuth);
longitudes = h5read(input, '/navigation/longitudes');
longitudes = permute(longitudes,[2 1]);
latitudes = h5read(input, '/navigation/latitudes');
latitudes = permute(latitudes,[2 1]);
```

All of the data was converted into radians for consistency and Matlab function compatibility. Some of the data also had to be rearranged into the right direction for Matlab to correlate the data with the image direction.

The image was prepared by NRL through APS. It was sent in a .png file that was read into Matlab using the image toolbox commands:

```
img_info = imfinfo(['Images_Raw/' d(i).name]);
img_raw = imread(['Images_Raw/' d(i).name]);
width = img_info.Width;
height = img_info.Height;
```

The image was then stored as a matrix and used in that structure for the rest of the code. Matlab built in toolboxes were used for all instances of file/image handling. Execution times of each step, cumulative through that step, and a full run were displayed in the output for each image. This was done through the code:

```
ttotal = 0;
tstart = cputime;
timage = cputime;
tend = cputime;
telapsed = tend - tstart;
minutes = floor(telapsed/60);
hours = floor(minutes/60);
minutes = rem(minutes,60);
disp(['Reading time: ' num2str(hours) ' hours ' num2str(minutes) ' minutes '
num2str(rem(telapsed,60)) ' seconds']);
ttotal = ttotal + telapsed;
minutes = floor(ttotal/60);
hours = floor(minutes/60);
minutes = rem(minutes,60);
disp(['Total time: ' num2str(hours) ' hours ' num2str(minutes) ' minutes '
num2str(rem(ttotal,60)) ' seconds']);
timageend = cputime - timage;
minutes = floor(timageend/60);
hours = floor(minutes/60);
minutes = rem(minutes,60);
disp(['Image Total time: ' num2str(hours) ' hours ' num2str(minutes) ' minutes '
num2str(rem(timageend,60)) ' seconds']);
ttotal = ttotal + telapsed;
minutes = floor(ttotal/60);
hours = floor(minutes/60);
minutes = rem(minutes,60);
disp(['Total time: ' num2str(hours) ' hours ' num2str(minutes) ' minutes '
num2str(rem(ttotal,60)) ' seconds']);
```

Step 2: Guided Filter

The guided filter created by He et al. is capable of many applications: noise reduction, smoothing/enhancement, feathering and haze removal, to name a few. [2] It is proposed that applying this Guided Filter, or GF, to the HICO images before attempting to identify shadow will improve the final results.

The guided filter requires a guided image (*I*), an input image (*p*), and will produce an output image (*q*). Both the guided image and input image can be the same image. This flexibility is one of the reasons guided filter was chosen for this research. There is no guarantee that there will be more than one image for any location in HICO imagery; therefore, a filter with this flexibility was needed. The output for the guided filter at pixel (i, j) can be expressed as a weighted average with the equation of:

$$q_i = \sum_j W_{ij}(I)p_j$$

W_{ij} is the filter part of the equation. This filter is a function of *I*, independent of and linear with respect to *p*.

He states that the guided filter has the key assumption that there is a local linear model between *I* and q. By assuming that q is a linear transform of *I* in a window of w_k centered at the pixel k, where (a_k, b_k) are some linear coefficients assumed to be constant in w_k, the following equation is defined:

$$q_i = a_k I_i + b_k, \forall i \in w_k$$

The filter uses a square window of pixels of radius r. Because this is a linear model, q has an edge if and only if *I* has an edge, shown by $\nabla q = a\nabla I$. To define the coefficients, He et al. minimizes the following cost function within the window:

$$E(a_k, b_k) = \sum_{i \in w_k}((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2).$$

In this equation, ϵ is a regularization parameter used to keep a_k from being too large. After using linear regression to find a solution to the equation above, the linear model is applied to all local windows in the entire image. The filter output for all patches in the image becomes:

$$q_i = \bar{a}_i I_i + \bar{b}_i.$$

Because of the modification of the coefficients varying spatially, ∇q is no longer a scaling of ∇I. He et al. also states, that "since (a_k, b_k) are the output of an average filter, their gradients should be much smaller than that of *I* near strong edges." [2] This allows for abrupt intensity changes in *I* to be mostly maintained in *q*.

Although the guided filter is capable of many applications, only the enhancement option was used in this research. The guided filter is similar to the bilateral filter used in detail enhancement and HDR compression. The process can be simplified into the following: Given an input signal, its edge-preserving smoothed output is used as a base layer. The difference between the original and base layer becomes the detail layer. This layer is then magnified to boost the details. This enhanced signal is a combination of the boosted detail layer and the base layer. This implementation can be seen through the program call used in the example code of the guided filter and this research:

```
I_enhanced = (I - q) * 5 + q;
```

The guided filter, along with the input and guiding images, requires the input of two additional parameters. These parameters are the window size *r* which should be a square and ϵ which is the regularization parameter. This value should be either $.1^2$, $.2^2$, or $.4^2$. For this research, the sizes tested for the window were 2, 4, 8, and 16. Each window size was used with each regularization parameter to decide which enhanced image would be the most useful. It was found that the error values showed little to no difference in the output, so a value of $.4^2$ was used. The *r* sizes that yielded the best end results were 2 and 16.

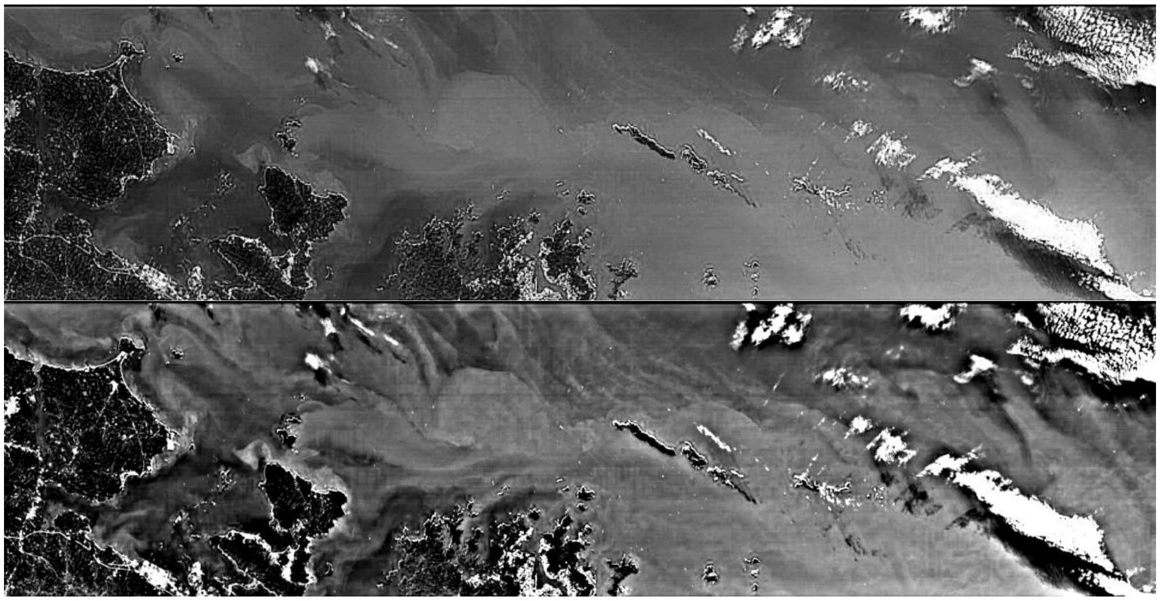

Figure 12. GAL produced Hong Kong Enhanced $r = 2$ $\epsilon = .4^2$. Bottom: Hong Kong Enhanced $r = 16$ $\epsilon = .4^2$.

It should be noted that although the finished algorithm applied GF to the blue channel of the enhanced image, the guided filter was manipulated and applied in different ways to ascertain in which way it would be most useful. The first idea was to apply GF to the radiance values in the HICO data. This would allow for enhancement at the very base level. This was done by using the RGB bands in HICO and running GF with those bands. It was also applied using an average of the bands that cover RGB spectrums. The final attempt was to alter GF itself to make calculations using all 87 accurate bands of HICO data. This theory did not yield any output that was useable. The radiance values in the level 1 data files do not have any atmospheric corrections made. This can be seen in Figure 13.

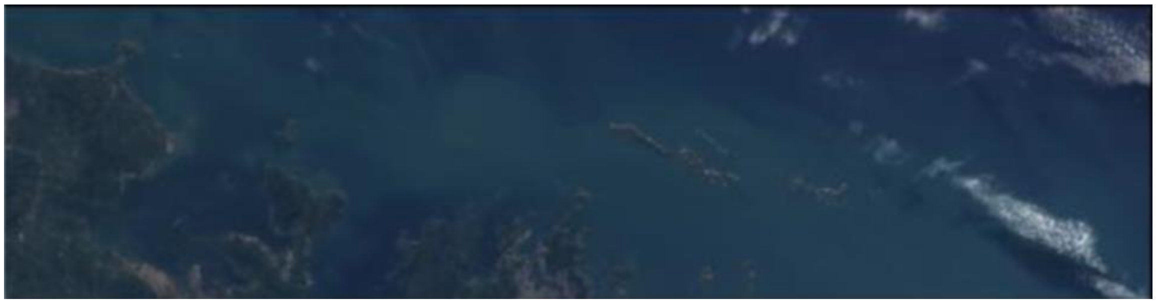

Figure 13. Hong Kong Image Created from Radiance Values.

There is a heavy blue tint over the entire image. This tinting caused skewing when GF is applied. Modifying and applying GF to all 87 bands did make a slight different in the enhancement of the image, but until the radiance values can be obtained with atmospheric corrections, this path was not further pursued.

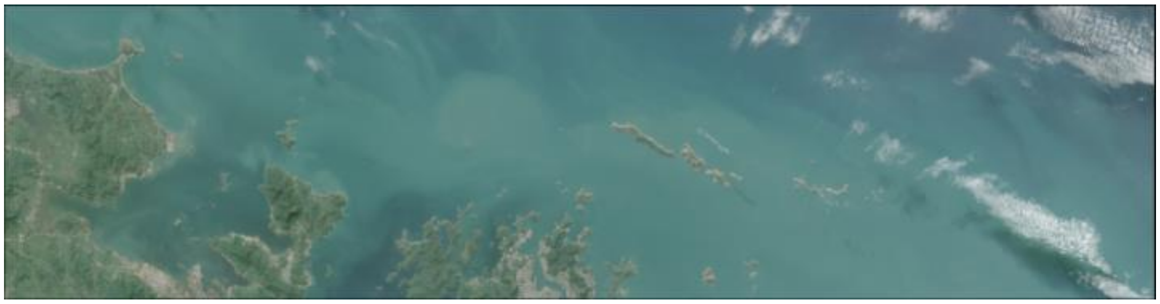

Figure 14. Hong Kong GF applied to 87 bands then processed.

At this point, the guided filter for colored images was used to enhance the provided raw image. Eventually, the guided filter for grayscale images was used on an image created using only the blue channel of the original RGB image. From previous research, the green and blue bands of water were used in the analysis of cloud shadow. During the beginning stages of this research, both green and blue bands were used as well. Better results were obtained when only the blue channel was used.

Step 3: Pixel Assignment

This research makes the assumption that all clouds, land, and water are identified before an image is processed. For the purposes of this research, land was identified by eye and masked using Paint .Net. The saved image was then loaded into Matlab to use for pixel assignment purposes.

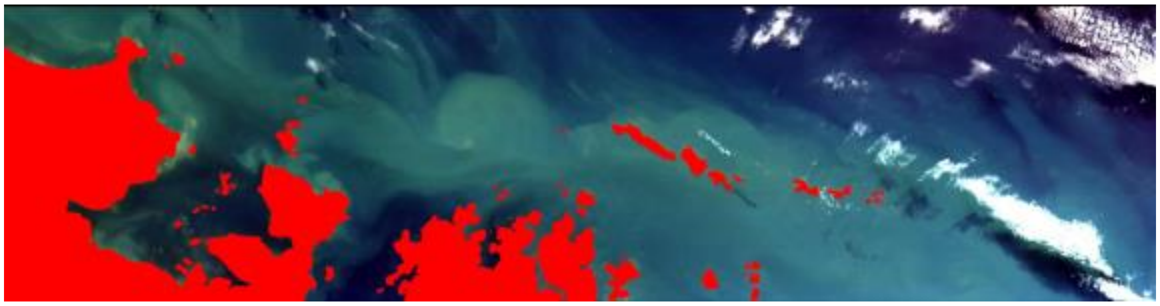

Figure 15. Hong Kong Land Masked.

Clouds were identified by masking any pixels in the enhanced image that had a value greater than 180. Anything not masked as cloud or land was assigned to water.

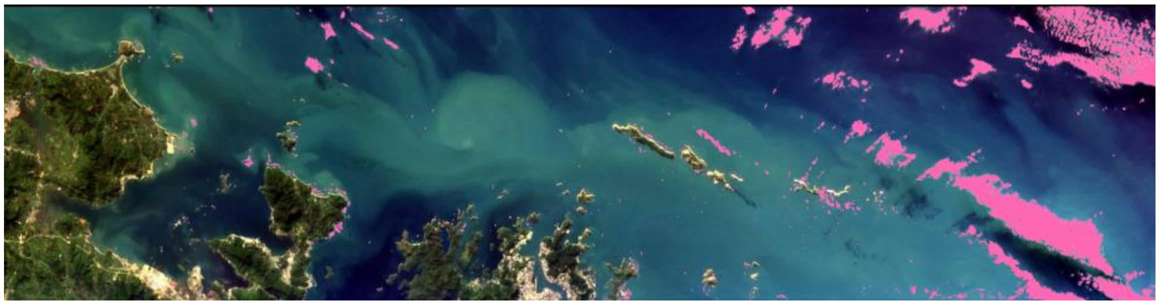

Figure 16. Hong Kong Identified Clouds Recolored in Pink.

A matrix is created to hold the pixel assignment values. As values are changed in the algorithm they are updated in this matrix:

```
pcm = cell(height, width, 1);
for row = 1:height
    for col = 1:width
        pcm{row, col} = 'W';
    end
end
```

```
%Map Cloud Pixels
for row = 1:height
   for col = 1:width
      pix_val = double(img_raw(row, col,3));
      if  pix_val > 180
         pcm{row, col} = 'C';
      end
   end
end

%Map Land Pixels
landimg = imread(['Images_Masked/' d(i).name]);
 for row = 1:height
   for col = 1:width
      if double(landimg(row, col, 1)) > 225 &&
            double(landimg(row, col, 3)) <100
            && double(landimg(row, col, 2)) < 100
         pcm{row, col} = 'L';
      end
   end
 end
```

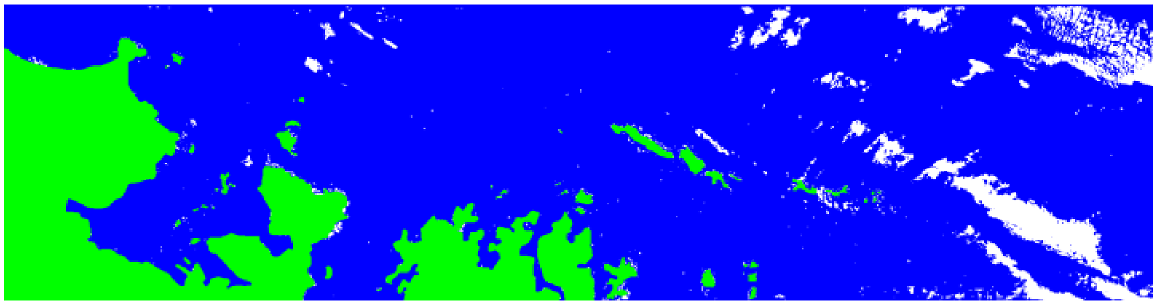

Figure 17. Hong Kong All Pixels Assigned.

Step 4: Geometric Linking

As discussed earlier, the navigational data was extracted from the .h5 file. All of these variables were imported, permuted in shape and converted from degrees to radians. The permutation of the data was necessary to match the direction of the image. Radians were needed as they are the default format for the Matlab geometric functions used in the cloud geometry. The area in an image can vary due to altitude and angle, defined by HICO specs, so a width and height measurement in meters was calculated for the image.

```
mwidth = (deg2km(distance(latitudes(1,1), longitudes(1,1), latitudes(1,width),
        longitudes(1,width))*1000))/width;
mlength = (deg2km(distance(latitudes(1,1), longitudes(1,1), latitudes(height,1),
        longitudes(height,1))*1000))/height;
```

To also prepare for the geometry section of the algorithm, the data was converted to be used in a UTM struct.

Universal Transverse Mercator (UTM) coordinates are useful when distances are needed on media such as maps or images of Earth. UTM is a 2- dimensional projected coordinate system. Some of the advantages of using UTM are no negative numbers, measurements are decimal based and in meters, and it allows for easier mathematics. A UTM grid system is applied to the earth. The image's location in the grid determines the projected coordinates. There are 60 zones in the grid of 6 degrees each. It should be noted that UTM is not used in the polar regions. Since HICO does not take images of polar regions, UTM was the only system used. The algorithm can be easily modified in the geometry to cover both regions. Each zone also includes bands of 8 degrees of latitude difference. Each of these bands is labeled C to X excluding I and O to reduce possible confusion and error. The measurements (Eastings and Northings) are constructed, so there are never negative numbers. Grid values increase left to right and bottom to top. Because this is the most accepted way of measuring distance for geographical means and because of the advantages listed above, UTM was utilized in preparing for the cloud shadow geometry.

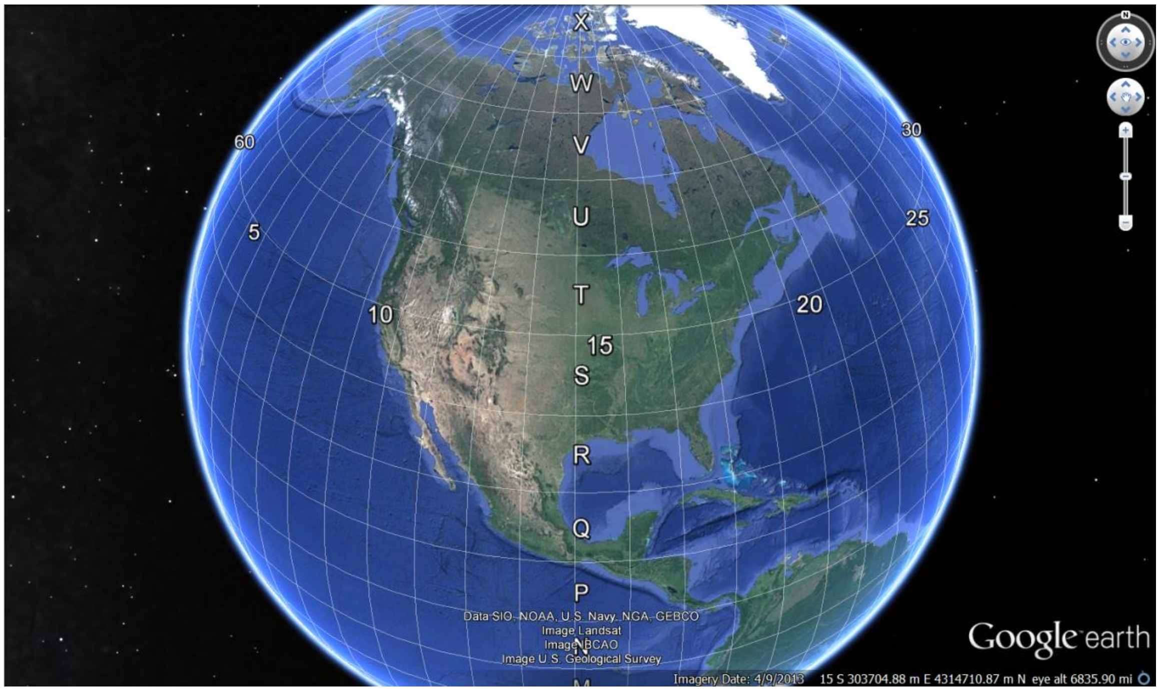
X
W
V
U
T
S
R
Q
P
60
5
10
15
20
25
30
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Image Landsat
Image IBCAO
Image U.S. Geological Survey
Google earth
Imagery Date: 4/9/2013 15 S 303704.88 m E 4314710.87 m N eye alt 6835.90 mi

Figure 18. Google Earth UTM Grid [17].

Matlab has a built in UTM library. The data was prepared using the following code:

```
vlat = latitudes(:);
vlong = longitudes(:);
utmlatlong = [vlat vlong];
zone = utmzone(utmlatlong);
utmstruct = defaultm('utm');
utmstruct.zone = zone;
utmstruct.geoid = wgs84Ellipsoid('meters');
utmstruct = defaultm(utmstruct);
```

Matlab's UTM code allows for data to fall into more than one zone, though it automatically adjusts the data to fit in one zone. Once the data has been converted to UTM coordinates, the cloud shadow geometry can be applied.

A visual representation of basic shadow geometry can be seen in Figure 19.

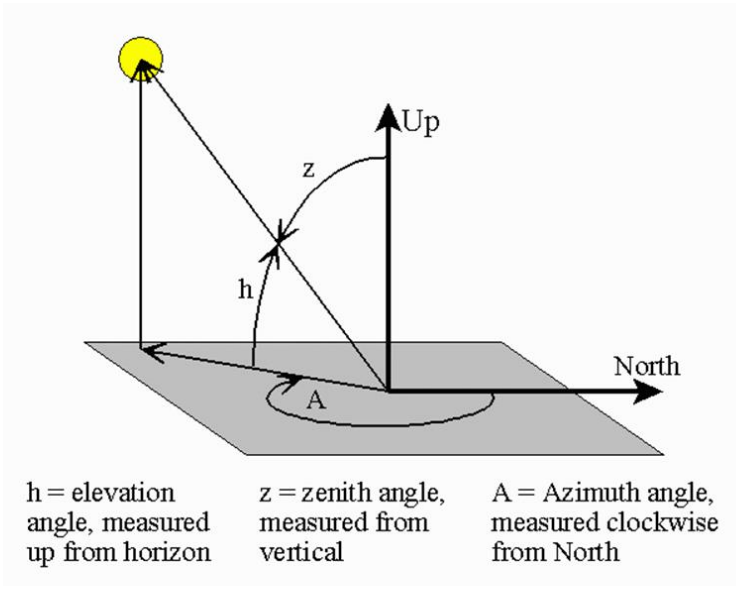
Up
z
h
North
A
h = elevation angle, measured up from horizon
z = zenith angle, measured from vertical
A = Azimuth angle, measured clockwise from North

Figure 19. Basic Shadow Geometry [18].

In this research, a slightly more involved shadow geometry is used. It can be visualized in Figure 20.

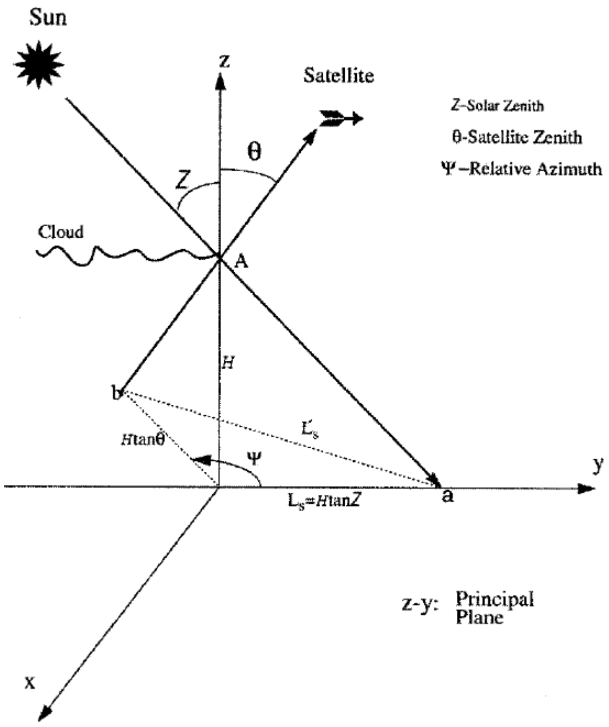

Figure 20. Simpson’s Sun Cloud Satellite Geometry [19].

Given the positions of the Sun, the satellite, and clouds, the cloud shadow locations can be explicitly determined [20]. In Luo's article, simplified equations for cloud shadow geometry are provided. These equations are used in this research. The basis for these equations and more detailed geometry can be found in Simpson's article [19].

Before the equations can be applied to the clouds in the image, the lack of cloud height values has to be addressed. HICO does not obtain any information on cloud height or information that can lead to approximating cloud height. This lead to an opportunity of creating a cloud shadow detection algorithm over water that is accurate using only an image and navigational data. Although the cloud height is unknown, it can be bound to a range based on atmospheric restrictions. Following previous research of Luo [20], cloud height can be limited to .5 to 16 km. The cloud ranges can be broken down into three categories; .5–8km (low latitudes 30º S to 30º N), .5–12km (mid latitudes 30–60º N or S), and .5–16km (high latitudes 60–90º N or S). The following code shows how this was accomplished:

```
%8 for 30S to 30N (-30 to 30)
%12 for 30 to 60N or S (-30 to -60 and 30 to 60)
%16 for 60 to 90N or S (-60 to -90 and 60 - 90)
%No need for Polar regions...HICO only goes 90 to 90
latmax = max(latitudes(:));
latmin = min(latitudes(:));
if abs(latmax) < abs(latmin)
        latmax = abs(latmin);
 else
        latmax = abs(latmax);
end
if latmax < 30
        hgtmax = 8000;
elseif latmax < 60
        hgtmax = 12000;
else
```

```
        hgtmax = 16000;
    end
```

An interval of .5 km was used in the iterative range of previous research. For this research, the interval was altered to fit the difference in resolution of HICO images. The following code was used to derive the interval:

```
    mean_zenith = mean(solar_zenith(:));
    iteration = tan(deg2rad(mean_zenith)) * mwidth;
```

The solar zenith is given for each pixel, but experimentation revealed that there is slight to no difference in geometry if a mean zenith is used. In fact, this reduces the number of computations necessary by calculating only one iteration value for all geometric calculations instead of recalculating for each cloud pixel.

Once the height range is determined, the geometry is straightforward. In this research, the simplified equations found in [20] and [21] are applied. The following equations derive the pixel coordinates for the cloud projection on the ground, given that (x_{img}, y_{img}) is a cloud pixel:

$$x_{nadir} = x_{img} + h_c \tan\theta_v \sin(\phi_v + \gamma)$$
$$y_{nadir} = y_{img} + h_c \tan\theta_v \cos(\phi_v + \gamma)$$

Those values can then be substituted into the following equations which then calculate the subsequent possible shadow projected onto the ground:

$$x_{shadow} = x_{nadir} - h_c \tan\theta_s \sin(\phi_s + \gamma)$$
$$y_{shadow} = y_{nadir} + h_c \tan\theta_s \cos(\phi_s + \gamma)$$

In these equations, h_c is the height of the cloud, θ_v represents the satellite zenith angle, θ_s represents the solar zenith angle, ϕ_v represents the satellite azimuth angles, and ϕ_s represents the solar azimuth angles. γ is the azimuth angle of true north from the y-axis, representing the shift in image to align with a projected xy plane. As stated earlier, the image has been converted to the UTM system. This allows for the γ portion of the equation to be dropped. The image is not rotated to fit a projected xy plane in the UTM system. The following code was created to satisfy these equations:

```
[x,y] = mfwdtran(utmstruct,latitudes(row,col), longitudes(row,col));
xn = x + h*tan(rad_sensor_zen(row, col))*sin(rad_sensor_az(row, col));
yn = y + h*tan(rad_sensor_zen(row, col))*cos(rad_sensor_az(row, col));
xs = xn - h*tan(rad_solar_zen(row, col))*sin(rad_solar_az(row, col));
ys = yn - h*tan(rad_solar_zen(row, col))*cos(rad_solar_az(row, col));
```

This code converts the navigational data of the cloud pixel into UTM coordinates. It then proceeds to find the cloud projected on the ground, then the cloud shadow projected onto the ground. This is not to be confused with the image pixel coordinates. This creates a projected space. At this point, geometry is used to take the projected space and find the closest coordinating pixel indices within the image. Figure 21 is a visual representation of the geometry applied to find the pixel indices from the projected shadow space.

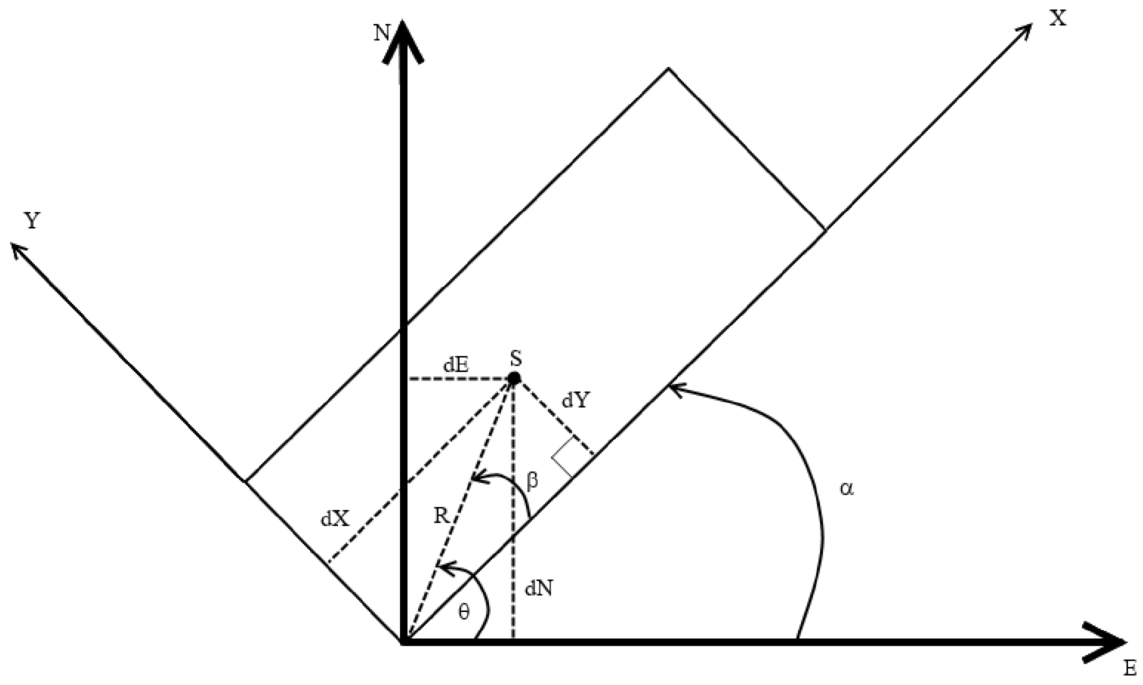

Figure 21. Geometry of Finding Pixel Indices from Projected Shadow Space.

From Figure 21 we can extrapolate the following triangle. The triangle in Figure 22 allows for the use of Directional Cosines to find the values dX and dY.

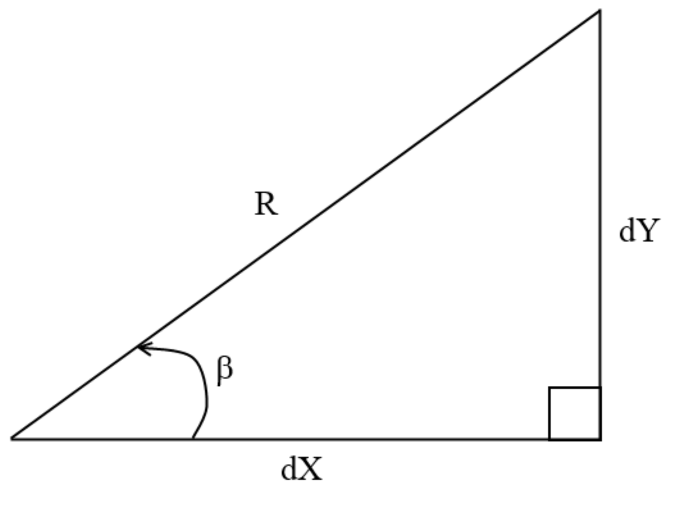

Figure 22. Directional Cosines Triangle.

These values were calculated in the following code:

```
r = sqrt(((xs - x)^2)+((ys - y)^2));
theta = atan((ys - y)/(xs - x));
```

```
if (xs - x) < 0
        theta = theta + acos(-1);
end
beta = theta - alpha;
drow = r*cos(beta);
dcol = r*sin(beta);
ys_adj = row + round((drow)/mlength);
xs_adj = col + round((dcol)/mwidth);
```

These values represent where the shadow would be in reference to the rows and columns of the image. These projected values allow for shadows that fall outside the realm of the image. Any values of this type are ignored. The pixels that are valid within the image space and assigned as water are marked as “P” in the assignment matrix. The lists of pixel indices that make a shadow path for a cloud pixel are also stored for later use.

When the cloud geometry section of the algorithm is finished, it yields all possible shadow pixels produced through cloud shadow geometry.

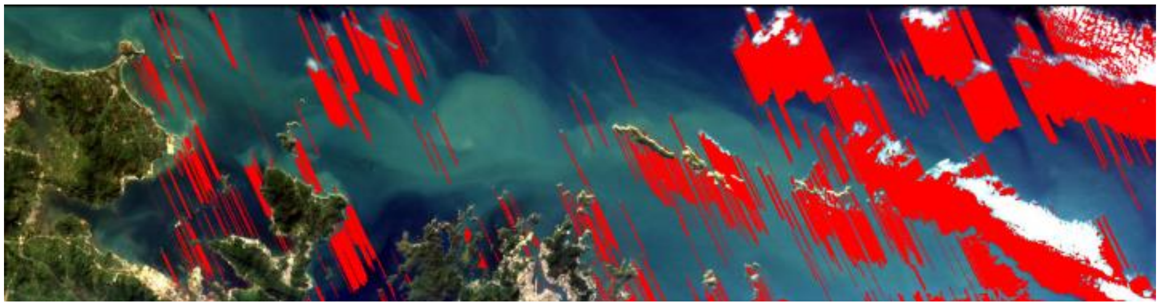

Figure 23. GAL produced Hong Kong Full Height Range Geometric Shadow Paths.

Figure 23 shows how important accurate cloud detection is. In the left side of the image there are possible shadow paths calculated for light colored water or land edges that where masked as cloud with the 180 threshold. This increases execution time and can lead to identifying false shadows.

Depending on how many clouds and how much of the image space the clouds cover, the possible shadow pixels may cover a substantial portion of the image, see

Figure 24. Because clouds truly only have one height, using a height range in geometry that covers all possible heights, the resulting shadow paths will always have pixels marked as shadow that are not shadow. Whether the image is greatly covered or whether extra heights are included, both situations lead to non-shadow marked as shadow. This would eliminate possible good values for end products. Given this knowledge, the next step in the algorithm is to reduce the possible shadow pixels down to a minimum.

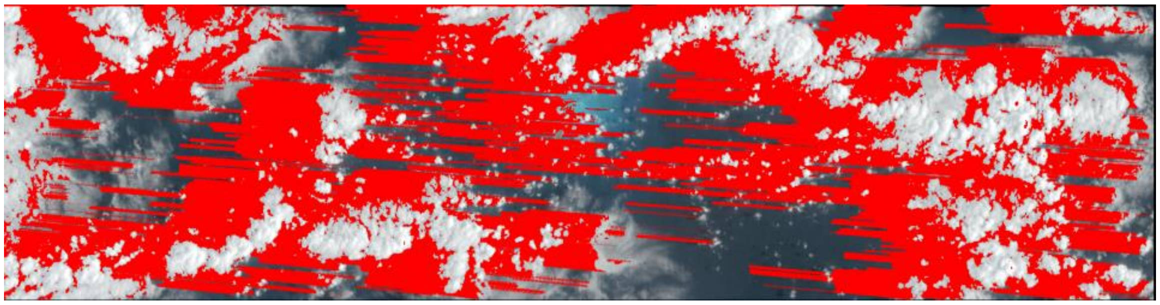

Figure 24. Lisianski Full Geometric Shadow Paths.

Step 5: Reduction

The reduction section starts with combining possible shadow pixels into shadow paths. This is completed by first making cloud blobs out of the cloud pixels. Using a connected components algorithm [22], the cloud blobs were connected if they resided within five pixels of another cloud pixel. The code is as follows:

```
cloudblobs = -1*ones(height, width);

%Map Cloud Pixels
for row = 1:height
    for col = 1:width
        if pcmnav{row,col} == 'C'
            cloudblobs(row, col) = 1;
        end
    end
end

pcmp = padarray(cloudblobs, [5 5], -1);
blobs = connect_components(pcmp);
```

```
[blobsc, blobcount] = count_blobs(blobs);
blobsc = reshape(blobsc, [width+10 height+10])';
cloudlabels = blobsc;
cloudlabels(:, 1:5) = [];
cloudlabels(:, end-4:end) = [];
cloudlabels(1:5, :) = [];
cloudlabels(end-4:end, :) = [];
```

A Boolean matrix of cloud pixels is created. That matrix is then padded by five rows and columns around the matrix. This padding allows for the testing of all indices without worrying about a border. A padding of 5x5 pixels is an effective means of reducing the amount of clouds to process. This padding can also be done with immediate neighbors or a large padding of pixels. It should be noted that padding assumes the cloud pixels linked together have a similar or equal height. With this assumption the padding of the cloud pixels should not be too large. A call to the function connect_components, sends the cloud pixel matrix into the function and reshapes it into a vector. This function creates a vector of parents for each pixel, initialized at -1. This vector value of -1 represents that, at this stage, every pixel is its own parent. This can be seen in the following code:

```
width = size(clouds,2);
cloudsv = reshape((clouds)', 1, []);
parents =  -1 * ones(size(cloudsv));

for y = 1:length(cloudsv)
        if (cloudsv(y) == 1)
        %left 4 pixels
                if (cloudsv(y-1) == 1)
                        parents = unionB(parents, y-1, y);
                end
                if (cloudsv(y-2) == 1)
                        parents = unionB(parents, y-2, y);
                end
                if (cloudsv(y-3) == 1)
                        parents = unionB(parents, y-3, y);
                end
```

```
if (cloudsv(y-4) == 1)
        parents = unionB(parents, y-4, y);
end

%upper 1 left 4 pixels
%upper pixel
if (cloudsv(y-width) == 1)
        parents = unionB(parents, y-width, y);
end
if (cloudsv(y-width-1) == 1)
        parents = unionB(parents, y-width-1, y);
end
if (cloudsv(y-width-2) == 1)
        parents = unionB(parents, y-width-2, y);
end
if (cloudsv(y-width-3) == 1)
        parents = unionB(parents, y-width-3, y);
end
if (cloudsv(y-width-4) == 1)
        parents = unionB(parents, y-width-4, y);
end

%upper 2 left 4 pixels
%upper pixel
if (cloudsv(y-(width*2)) == 1)
        parents = unionB(parents, y-(width*2), y);
end
if (cloudsv(y-(width*2)-1) == 1)
        parents = unionB(parents, y-(width*2)-1, y);
end
if (cloudsv(y-(width*2)-2) == 1)
        parents = unionB(parents, y-(width*2)-2, y);
end
if (cloudsv(y-(width*2)-3) == 1)
        parents = unionB(parents, y-(width*2)-3, y);
end
if (cloudsv(y-(width*2)-4) == 1)
        parents = unionB(parents, y-(width*2)-4, y);
end

%upper 3 left 4 pixels
%upper pixel
if (cloudsv(y-(width*3)) == 1)
        parents = unionB(parents, y-(width*3), y);
end
if (cloudsv(y-(width*3)-1) == 1)
```

```
                parents = unionB(parents, y-(width*3)-1, y);
            end
            if (cloudsv(y-(width*3)-2) == 1)
                parents = unionB(parents, y-(width*3)-2, y);
            end
            if (cloudsv(y-(width*3)-3) == 1)
                parents = unionB(parents, y-(width*3)-3, y);
            end
            if (cloudsv(y-(width*3)-4) == 1)
                parents = unionB(parents, y-(width*3)-4, y);
            end

            %upper 4 left 4 pixels
            %upper pixel
            if (cloudsv(y-(width*4)) == 1)
                parents = unionB(parents, y-(width*4), y);
            end
            if (cloudsv(y-(width*4)-1) == 1)
                parents = unionB(parents, y-(width*4)-1, y);
            end
            if (cloudsv(y-(width*4)-2) == 1)
                parents = unionB(parents, y-(width*4)-2, y);
            end
            if (cloudsv(y-(width*4)-3) == 1)
                parents = unionB(parents, y-width-3, y);
            end
            if (cloudsv(y-(width*4)-4) == 1)
                parents = unionB(parents, y-(width*4)-4, y);
            end
        end
    end
```

The center pixel, the vector of parents, and the neighbor pixel are then sent to the function unionB. This is completed for every neighboring pixel that is a cloud pixel in the 5x5 square around the center pixel, repeated for every cloud pixel seen in the code below:

```
    [parents, a] = find_set(parents, a);
    [parents, b] = find_set(parents, b);
    if (a ~= b)
        size = parents(a) + parents(b);
        if (parents(a) <= parents(b))
            parents(b) = a;
```

```
                parents(a) = size;
        else
                parents(a) = b;
                parents(b) = size;
        end
end
```

This function first calls find_set for both the center pixel and the neighbor pixel. The find_set function returns the root for each of the pixels. If they are not equal, the two trees become one, and the parent vector is updated. This can be seen here:

```
r = index;
while (parents(r) >= 0)
        r = parents(r);
end
i = index;
while (parents(i) >= 0)
        index = parents(i);
        parents(i)= r;
        i = index;
end
```

This is continued until all clouds blobs have been connected. Once the cloud blobs have been connected they are counted and then each cloud pixel is relabeled according to the cloud blob id number. This is done through the code:

```
count = 0;
blobsc = zeros(size(blobs));
%number roots
for y = 1:length(blobs)
        if blobs(y) < -1
                %see how many roots we have
                count=count+1;
        end
end
disp(['Number of clouds: ' num2str(count)]);
%loop through tree until root is found
for x = 1:length(blobs)
        if blobs(x) ~= -1
                [blobs,root] = find_set(blobs, x);
                blobsc(x) = root;
        end
    end
    count = 0;
    for z = 1:length(blobs)
        if blobsc(z) > 0
            root = blobsc(z);
            if root > count
                count = count + 1;
                blobsc(blobsc == root) = count;
            end
        end
    end
```

This vector is reshaped back into the image matrix shape and de-padded. The completion of this section allows for the combining of single pixel shadow paths to cloud blob shadow paths.

The shadow pixel paths for each cloud pixel in a now defined cloud blob are combined and unique values are stored using the following code:

```
    A = zeros(1,2);
        for row = 1:height
            for col = 1:width
                if cloudlabels(row, col) == clouds
                    A = [A; pcms{row, col}];
                end
            end
        end
    A = unique(A,'rows');
```

This code takes each vector path for each pixel within a cloud blob and combines them into one large vector. The Matlab function called unique, removes all the pixels that are redundant in the vector. This yields the shadow path for the entire cloud blob. The resulting image is the same as Figure 23.

Once a cloud shadow path can be determined, the pixels within that path need to be reduced as much as possible. The ocean water may be coastal or deep ocean in this imagery, which can yield either homogeneous or heterogeneous waters. It was thought that these two types of water would yield different results, and two thresholds were created to handle the issue.

Image variances were calculated and recorded. All images were run on both the average water threshold and the minimum value edge neighbor threshold. The average water threshold was created by calculating the mean value of all water pixels that were not already assigned to possible shadow paths. A comparison of the results revealed that the minimum value edge neighbor threshold worked the best on almost all images. At this point, the average water threshold was removed. It should be noted that the Hong Kong image, one of three images obtained as the test image sample, was one of two images from the final sample batch that performed better in areas with the average water threshold. As can be seen in Figure 25, the average water threshold identifies more shadow in some areas but seems to over identify in others. The edge threshold reduces better in some areas but under identifies in the rest. This is not the case for the majority of the test images.

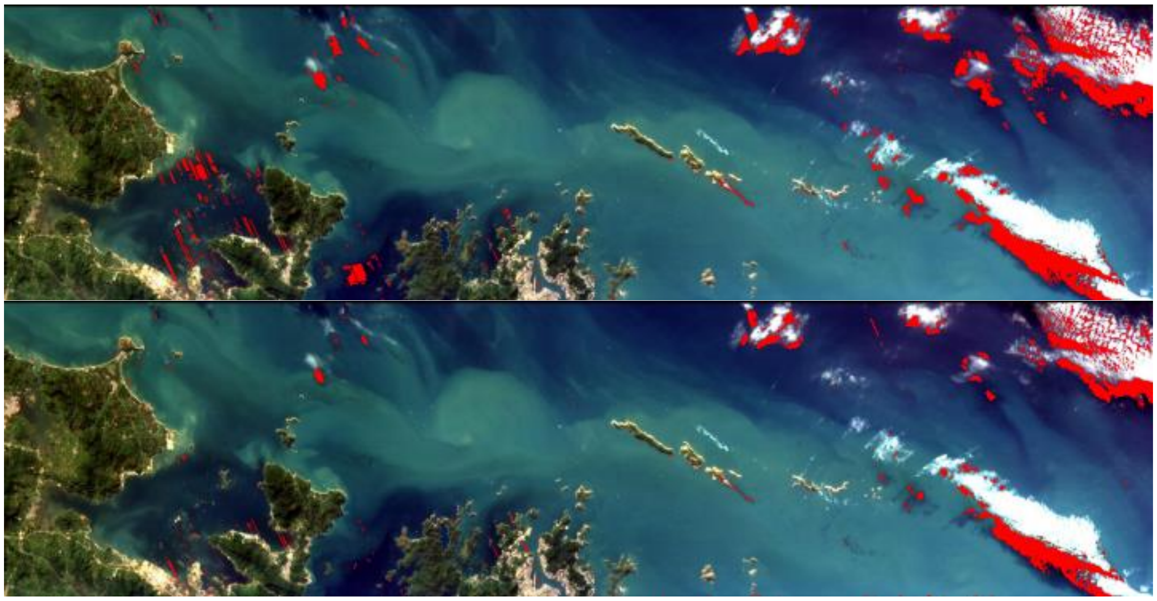

Figure 25. Hong Kong Reduction Top: Average Water Threshold Bottom: Minimum Value Edge Neighbor Threshold.

For the final reduction section of the algorithm, a minimum value edge neighbor threshold was used for all images. In this threshold, the edge function in Matlab was used. The edge function requires an input of an intensity or black/white matrix; the grayscale enhanced image works fine. When extracting the values from the shadow path, it is imperative that the path retain its shape so that edges are undisturbed. The shape of the path is most likely not along rows and columns, which means the path cannot be converted into an edge input matrix without distorting the edges.   Figure 26 illustrates that the paths are not in a rectangular shape, so they cannot be directly entered into the function.

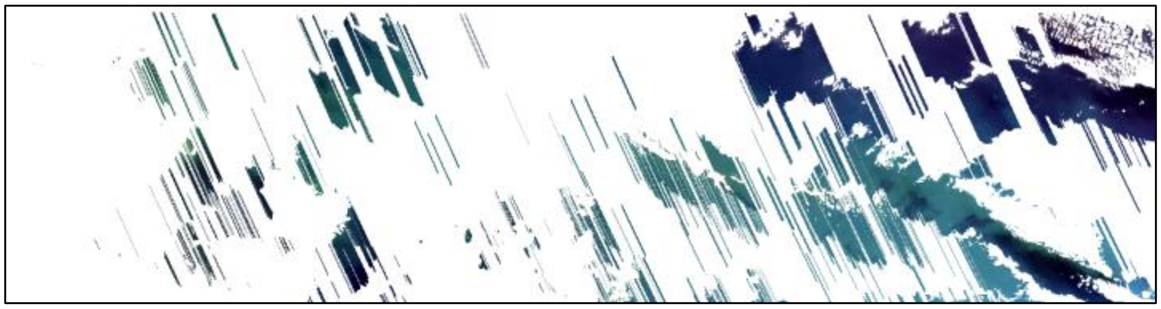

Figure 26. Hong Kong Shadow Paths Isolated.

The path was inserted into a matrix, and empty matrix values were filled with an unrelated value. Using these values did not work because edge detected only the edge of the path and no edges within the path. This limitation led to the use of the actual image values by creating a rectangle around the shadow path. This rectangle is obtained by identifying the minimum and maximum row and column values, shown by the code below where A is the shadow path vector:

```
min_row = min(A(:,1));
min_col = min(A(:,2));
max_row = max(A(:,1));
max_col = max(A(:,2));
```

These values are then used to extract the rectangle of pixels, which include the shadow path. The rectangle of pixels extracted was taken from the intensity image I_enhanced. As stated earlier, this image is the enhanced blue channel of the raw image. The edge function allows for choice of method for edge detection. Through trial and error, the canny method yielded the best results. Matlab states that the Canny method “finds edges by looking for local maxima of the gradient of I. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges.” [23] Although Canny had the best results, most of the weak edges it identified were not shadow but color variance in the water. This issue was resolved by modifying the threshold to which canny uses to identify weak edges. There was no option to choose only strong edges, so the weak edge threshold was calculated to be as close to the strong threshold as possible within the canny parameter restrictions.

```
test = I_enhanced(min_row:max_row, min_col:max_col);
[th, tw] = size(test);
[e1, thresh] = edge(test, 'canny');
if thresh(1, 2) < 1
        dthresh = thresh(1,2) - thresh(1,1);
        dthresh = dthresh/100;
        thresh = [thresh(1,2) - dthresh, thresh(1,2)];
        e1 = edge(test, 'canny', thresh);
end
```

The edge function returns a black and white matrix the size of the input where the values are 0 if not an edge, 1 if an edge.

At this point in the algorithm, the edge detection in the path has identified both shadow edges and water variance edges. For each pixel identified as an edge, the pixels around it were examined. This is done using the code:

```
[e1_height, e1_width] = size(e1);
min_e1 = 255;
for erow = 1:e1_height
        for ecol = 1:e1_width
                if e1(erow ,ecol) == 1
                        if pcmnav{erow + min_row - 1, ecol + min_col - 1} == 'P'
                        %tl
                                if pcmnav{erow + min_row - 2, ecol + min_col - 2}
                                == 'P'
                                        e_tl = I_enhanced(erow + min_row - 2, ecol
                                        + min_col - 2);
                                        if e_tl < min_e1
                                                min_e1 = e_tl;
                                        end
                                end
                        %tc
                                if pcmnav{erow + min_row - 2, ecol + min_col - 1}
                                == 'P'
                                        e_tc = I_enhanced(erow + min_row - 2, ecol
                                        + min_col - 1);
                                        if e_tc < min_e1
                                                min_e1 = e_tc;
                                        end
                                end
                        %tr
                                if pcmnav{erow + min_row - 2, ecol + min_col} ==
                                'P'
                                        e_tr = I_enhanced(erow + min_row - 2, ecol
                                        + min_col);
                                        if e_tr < min_e1
                                                min_e1 = e_tr;
                                        end
                                end
                        %l
                                if pcmnav{erow + min_row - 1, ecol + min_col - 2}
                                == 'P'
                                        e_l = I_enhanced(erow + min_row - 1, ecol
                                        + min_col - 2);
                                        if e_l < min_e1
                                                min_e1 = e_l;
                                        end
                                end
                        %r
                                if pcmnav{erow + min_row - 1, ecol + min_col} ==
                                'P'
                                        e_r = I_enhanced(erow + min_row - 1, ecol
                                        + min_col);
                                        if e_r < min_e1
                                                min_e1 = e_r;
                                        end
                                end
                        %bl
                                if pcmnav{erow + min_row, ecol + min_col - 2} ==
                                'P'
                                        e_bl = I_enhanced(erow + min_row, ecol +
                                        min_col - 2);
                                        if e_bl < min_e1
                                                min_e1 = e_bl;
                                        end
                                end
                        %bc
                                if pcmnav{erow + min_row, ecol + min_col - 1} ==
                                            'P'
                                                e_bc = I_enhanced(erow + min_row, ecol +
                                                min_col - 1);
                                                if e_bc < min_e1
                                                    min_e1 = e_bc;
                                                end
                                            end
                                        %br
                                            if pcmnav{erow + min_row, ecol + min_col} == 'P'
                                                e_br = I_enhanced(erow + min_row, ecol +
                                                min_col);
                                                if e_br < min_e1
                                                    min_e1 = e_br;
                                                end
                                            end
                                        %center
                                            if I_enhanced(erow + min_row - 1, ecol + min_col -
                                            1) < min_e1
                                                min_e1 = I_enhanced(erow + min_row - 1,
                                                ecol + min_col - 1);
                                            end
                                        end
                                    end
                                end
```

If the pixel was in the shadow path and not an edge itself, the value was stored. The minimum value of all of these pixels was calculated. This value was used as the threshold for shadow detection. The assumption is that if the edge is a shadow edge there will be a dark pixel next to it. Using the darkest near edge pixel will identify most of the shadow. With the deep enhancement that is obtained through GF it allows edge to better detect the shadow edges. This seems to break down only when the images are extremely light in color intensity and/or contrast, such as in Figure 27.

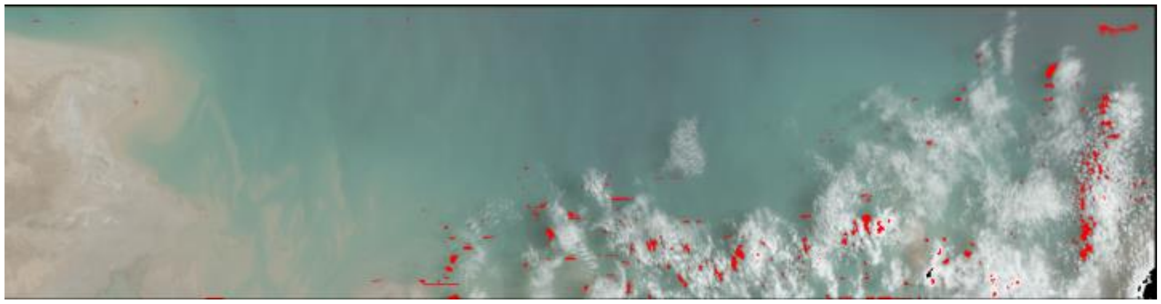

Figure 27. Persian Gulf Final Shadow Detection.

CHAPTER III

CONCLUSION

Results

Three images were used during the development phase of this research. The finalized model was applied to a total sample size of 31 images. Of these 31 images, two were removed for possible data inaccuracies. This leaves a complete sample of 29 images. These images range in dates from 2009 to 2014, with the majority from 2014. All but four images were of unique location, scattered around the globe. Although these four images are of the same area, none of them cover the exact same latitudes and longitudes. All 31 raw images, with their process through GAL, are included in the Appendix.

The results show the model works very well for images of great contrast with dark well-defined cloud shadows. The type of water, homogeneous or heterogeneous, does not affect the outcome of the model. The images that have cloud shadows that have a large gradient result in partially identified shadows. The images that have the worst results are those that have extremely light values and very little water to shadow contrast.

When examining the results, it can be difficult to judge what is shadow and what is not shadow. This leads to all final results being visually interpreted. The method used to measure GAL's final results was identified by two tasks. The first task was to identify obvious cloud shadows. The second task was to work as well or better than the previous research. As stated above, GAL identifies most obvious cloud shadows with only a few exceptions. When examining previous research, Figure 29 was created by layering the two images from Figure 28 using a 40–60% opacity on the shadow detected image. This image visually shows how well the shadow pixels are detected in Amin's et al. results.

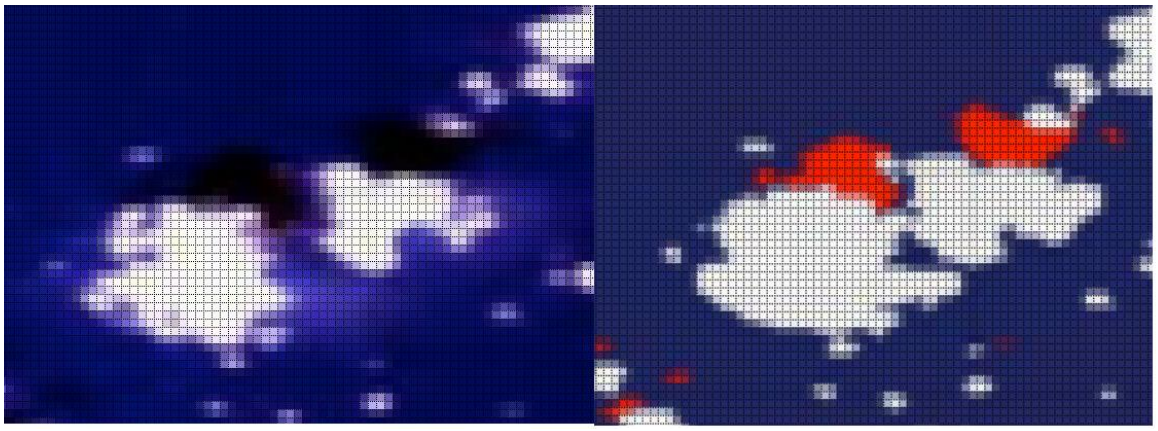

Figure 28. Amin's et al. cropped Virgin Islands results blown up.

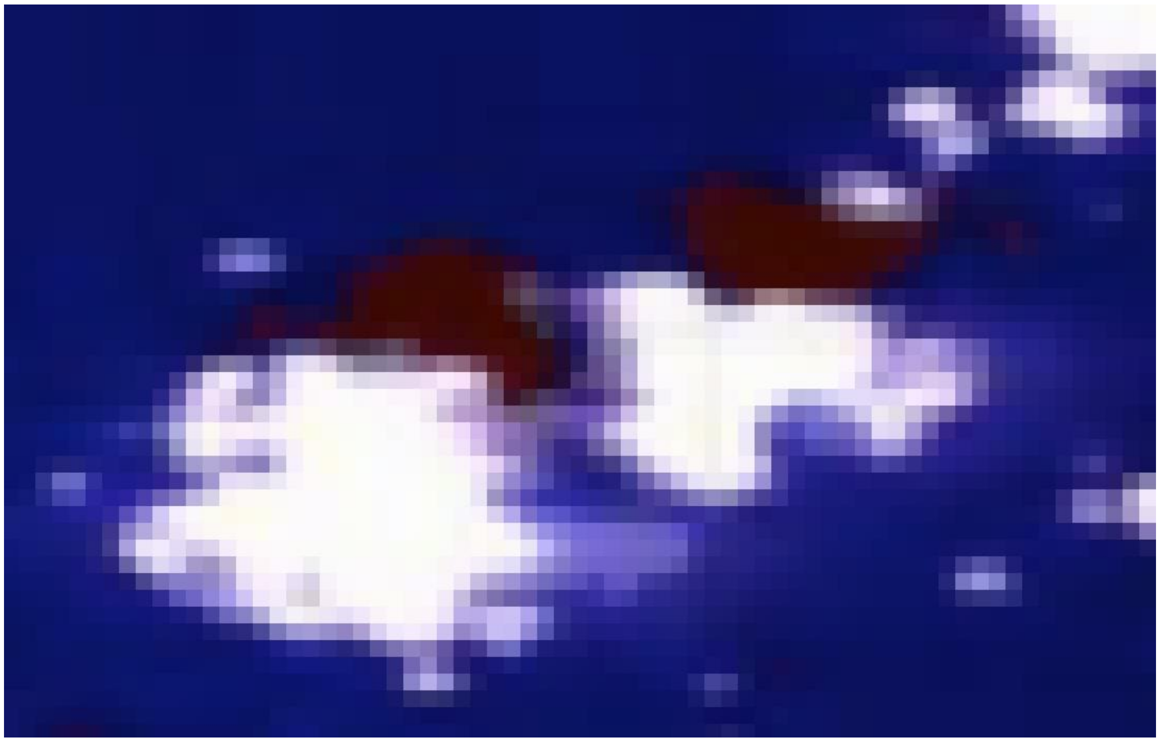

Figure 29. Amin's et al. results layered together with 40 – 60% opacity in the shadow identified image.

The pixels around the shadow detected image visually appear to be part of the shadow edge. This would support Amin's statement of having some issues with shadow edge detection. The same image combining technique was applied to GAL results of the same area. This can be seen in Figure 30 and Figure 31.

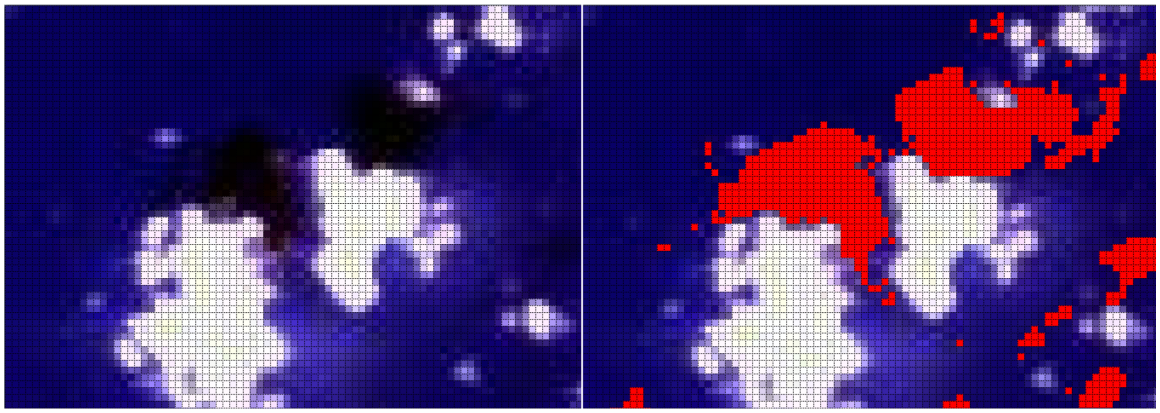

Figure 30. GAL results using the same area of Virgin Islands as in Amin's et al. research.

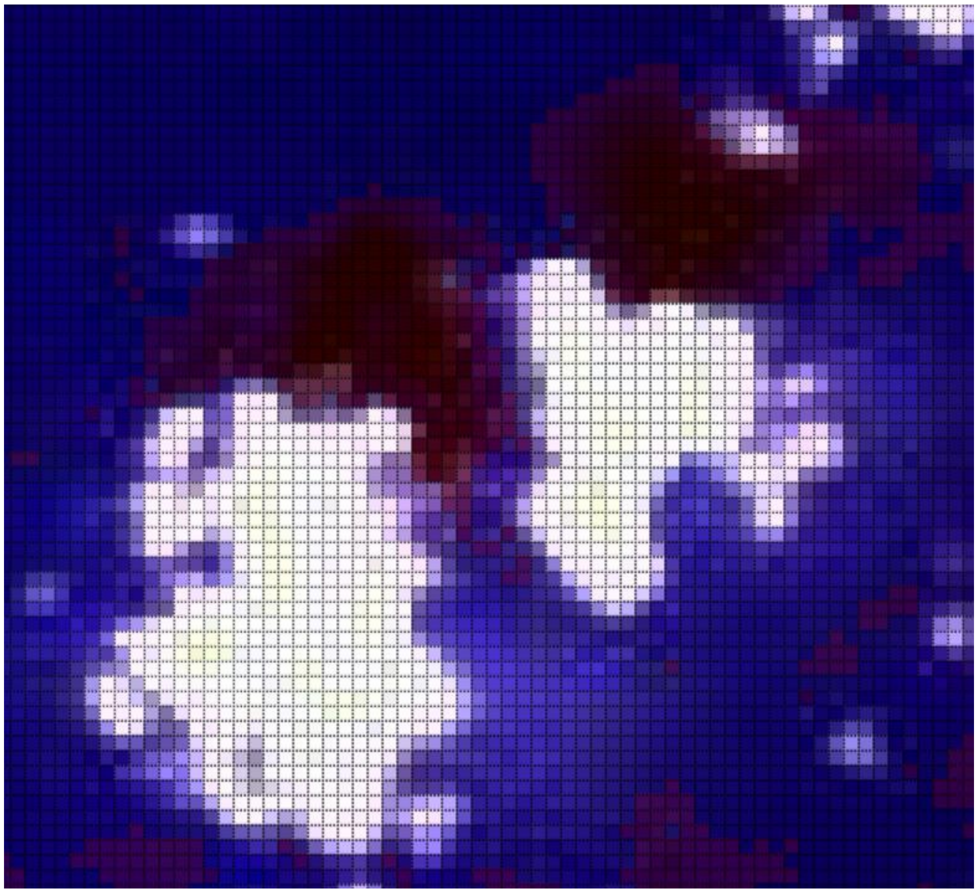

Figure 31. GAL Virgin Islands results layered using 40–60% opacity in the shadow detected image.

It can be seen that GAL identified the cloud shadow pixels just as Amin's algorithm did plus some. It should be noted that there is some distortion from the original image from Amin's publication of his image results, as well as the pixels identified as cloud are obviously different from GAL's cloud detection. Some of the shadow pixels identified by GAL are identified as cloud in Amin's results. The images also show that the shadow edges are well detected when using GAL. Although these edges are detected, it should be noted that GAL appears to over-identify the cloud shadow in some areas. This could be a loss of good data. It could be argued that due to resolution and computing the accuracy by eye, some of these outer edge pixels could be contaminated with shadow. In the case of those outer pixels, it would be better to over-identify than not to identify them as shadow at all. More images of GAL's results with the layered technique are below.

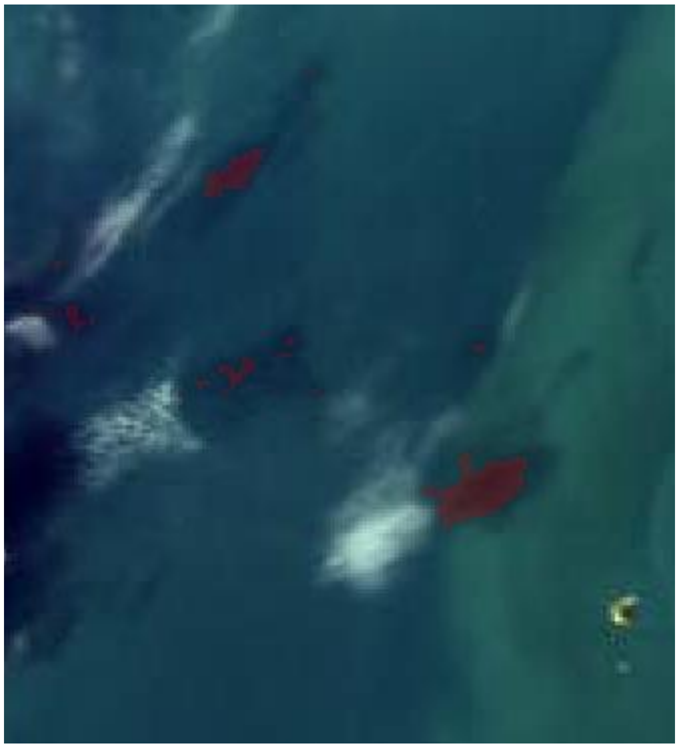

Figure 32. Hong Kong layered image. Shows cloud shadow not detected due to gradient shadow and cloud detection.

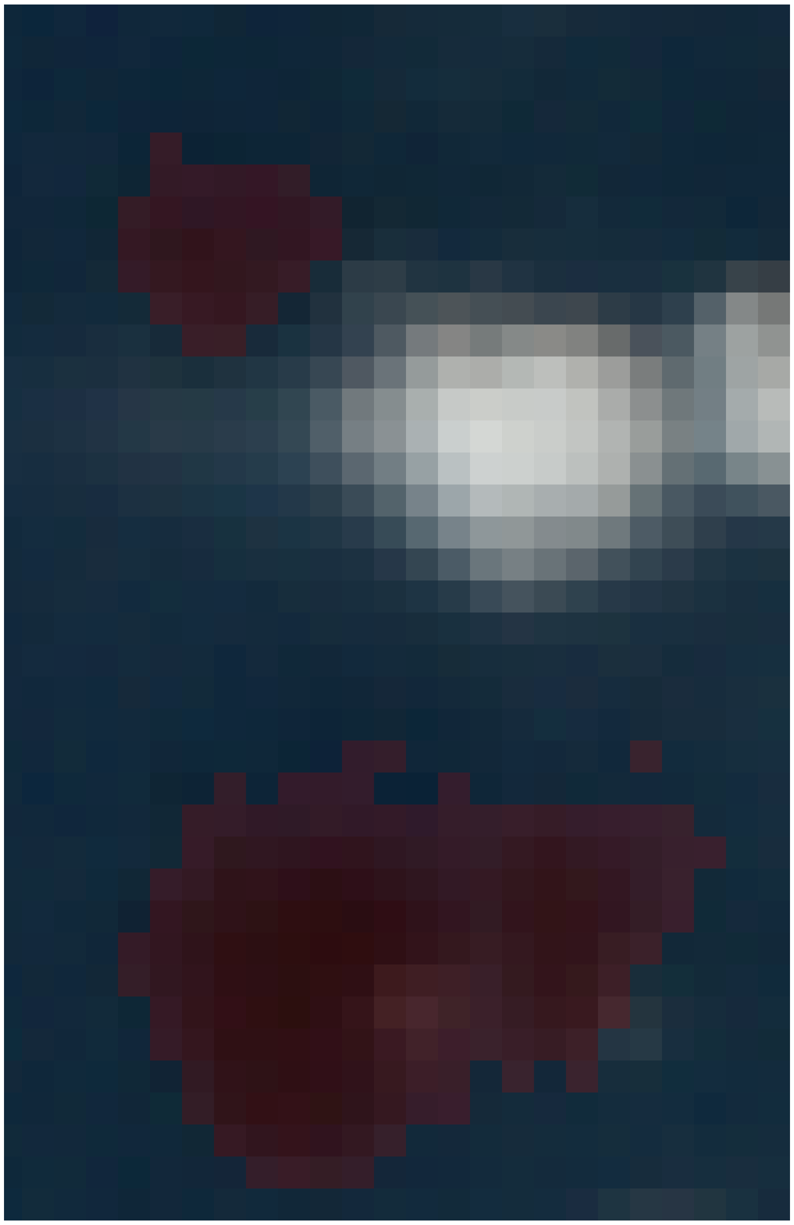

Figure 33. Key Largo with layered image. Shows well defined shadow detection with possible false positives.

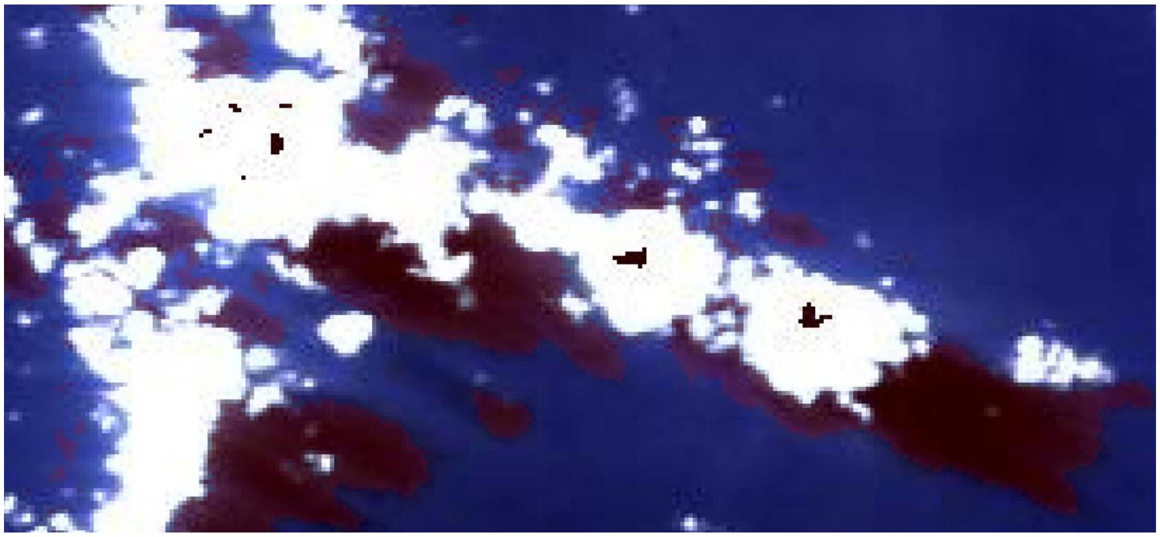

Figure 34. N. Mariana Pagan layered image. Shows best case shadow detection.

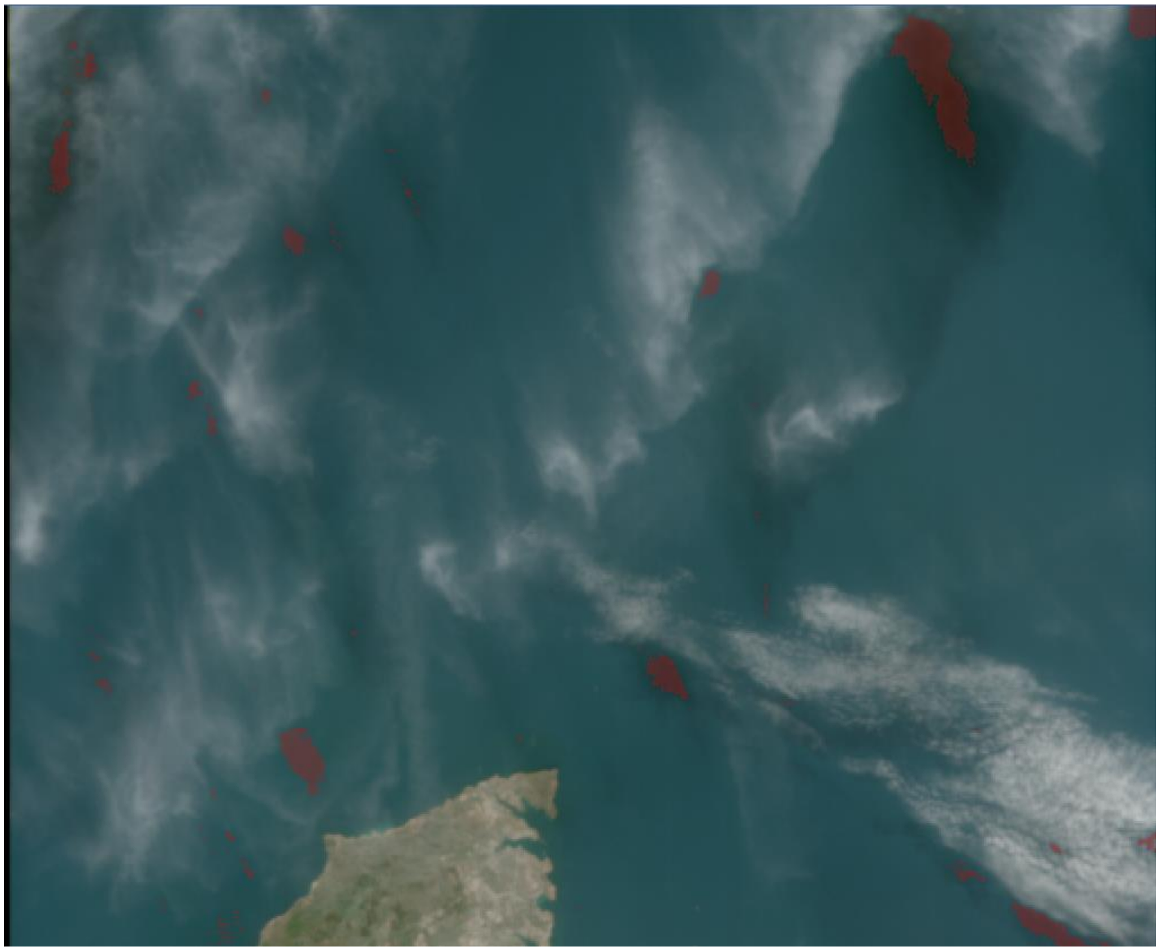

Figure 35. Sevastopol Crimea layered image. Shows worst case, many shadow not detected.

The results shown above and of the entire sample in the appendix show that GAL works well. The area of image processing is a very hot area of research. With the increasing amount of data stored and obtained daily through new and old satellite images, cloud shadow detection is an important need. GAL was created to fill this need. GAL takes previous research along with new techniques to create a novel, fully automated, stepwise cloud shadow detection algorithm for over water. This research combines the past techniques of cloud shadow geometry, edge detection, and thresholding with the new techniques of guided image filtering in such a way as never has been done before. Not only is it a model that can be applied to HICO data immediately, it has the potential of being applied to all satellite imagery worldwide.

Future Work

One of the main drawbacks to this research is the execution time of the cloud shadow geometry. The geometry was responsible for an average of 88% of the execution time. This large chunk of execution time is directly correlated to the number of times the image completes the shadow geometry for loop. This for loop varies in count due to the iteration value, cloud pixel count, and the height range maximum. This correlation is shown in Table 1. This table was created to calculate the average execution time attributed to the cloud shadow geometry and to find any other useful correlations about execution times.

Table 1

Image Statistics

Location	Clouds	Cloud Pixels	Iteration	Height Max	For Loop Count	Processing Time	Geometry Time	Time Diff	Geo %
MVCO	110	12367	281.9331	12000	504448	11	9	2	82%
Key Largo	234	21186	169.0136	8000	940131	20	16	4	80%
Port au Prince	95	21526	153.4425	8000	1052153	19	17	2	89%
Sevastopol Crimea	63	15809	114.8757	12000	1582611	29	27	2	93%
Puerto Rico	530	15777	63.9107	8000	1851451	40	30	10	75%
Noumea New Caledonia2	787	80967	256.8175	8000	2364529	53	39	14	74%
Hong Kong	395	52285	149.7162	8000	2619206	51	43	8	84%
Arabian Sea	477	60628	133.6847	8000	3401362	66	58	8	88%
Guam	1142	78761	152.2339	8000	3880263	86	64	22	74%
Tasmania 2	111	47198	121.3578	12000	4472535	76	73	3	96%
Palau3	770	123794	204.1777	8000	4547289	90	73	17	81%
Lucinda Jetty	764	120138	178.4225	8000	5050008	98	83	15	85%
Virgin Islands	888	99955	147.6736	8000	5076483	102	84	18	82%
Red Sea Straits	524	51136	69.9546	8000	5482413	98	88	10	90%
MOBY	439	98829	127.8203	8000	5798903	105	95	10	90%
Tasmania	202	81726	159.2323	12000	5902377	102	97	5	95%
MOBY2	309	47493	56.5743	8000	6296101	115	108	7	94%
Swan River Estuary	379	69740	113.6334	12000	7057872	125	117	8	94%
North Mariana Pagan	874	137525	135.9478	8000	7587011	140	122	18	87%
Lake Idhu	464	70657	103.4120	12000	7857459	141	130	11	92%
Noumea New Caledonia	496	153707	143.3769	8000	8040364	145	132	13	91%
BLZ	482	279338	259.1212	8000	8085155	156	137	19	88%
Iquique Chile	190	187736	146.3344	8000	9621934	170	160	10	94%
Lake Victoria	218	217412	147.8880	8000	11025844	196	179	17	91%
Wake Atoll	379	265519	170.0772	8000	11708756	218	195	23	89%
Persian Gulf	199	129216	102.1504	12000	14547021	253	233	20	92%
Lisianski	638	358475	178.4791	8000	15063739	267	244	23	91%
Noumea New Caledonia3	341	219569	103.6039	8000	15894841	265	252	13	95%
West Samoa	752	401428	93.6994	8000	32131582	559	514	45	92%
							Average Geo Time		88%

Note. Times are in unit of minutes. Height and iteration are in meters.

Another correlation this table identifies is that the large time differences seem to be related to the number of clouds processed after geometry. This affects the amount of times the edge detection and threshold code is applied. These statistics support the claim stated in previous publications that the shadow geometry is time consuming. Although it is time consuming, the shadow geometry is the only finite way to identify the path of cloud shadow. Since it is so important, it cannot be removed, but perhaps the shadow geometry could be parallelized. Because the cloud pixels are independent, it provides a good case for splitting up the input. It is also possible to use shadow geometry after the cloud pixels are blobbed. This allows for shadow geometry to be applied to cloud edges, and anything in between would also be flagged as possible shadow. Either option allows for a dramatic reduction in geometry execution time. Execution time of shadow geometry aside, the rest of the execution takes very little time.

If shadow geometry execution time was reduced, the next issue to be addressed is to improve the results. There are two major concerns when looking at the current results. First, the results do not work well in light or low contrast images, such as the image of the Persian Gulf seen in Figure 27. This could be fixed by exploring more into the GF. If the GF was able to increase the contrast between water and shadow more, the edge detection algorithm would recognize the shadow edges better. This could be done through either a more extensive experimentation of the parameters entered or possibly by extending the original algorithm for better enhancement. The second concern is when the shadows are under-identified. Some of these images have shadow with high gradients within the shadow like in the Hong Kong image in Figure 4 that caused the problem. The edge detection algorithm recognizes the full shadow edge when using the default threshold values with the true shadow edge as a weak edge. When the thresholds are altered to retain only strong edges, the weaker edges are no longer detected. A better lower threshold may yield better results. It may be useful to explore the values recognized as weak edges to yield a better threshold for these images. Another problem that occurs, leaving some shadow pixels unidentified, is that stronger edges are found in the rectangle around the shadow path. This happens often when clouds and cloud shadows are close to each other and may be in the rectangle around the path or in the path itself. One possible way to combat this is to revisit the filling of the matrix empty values around the path with an arbitrary number. Possibly identifying a water pixel around the path that is light, giving the assumption it is water, might be a better fit for the matrix filler. This would then eliminate all edges outside the path but should make the shadow path edge closer to the actual values, so the edge function does not identify it as an edge.

As well as the drawbacks, there are always additions that could improve performance. For GAL, one addition, in particular, is already apparent. All images have borders. Shadows in the border areas of the image where clouds would be outside of the imaged area are a major problem for shadow detection. Without a cloud to associate with the shadow, there is no guarantee that it is a shadow. Quite often though, these questionable areas give every indication of being cloud shadow by their shape, color, and/or closeness to other clouds and their shadows. In these situations it is the same as other cloud shadows; the pixels will skew end products. To avoid including possibly contaminated data, these pixels should be masked. A border shadow detection should be included for better shadow detection. A possible area to explore is to use the values of cloud shadows within close proximity of that questionable border area to identify possible cloud shadows.

Although not an addition, future work would also include applying GAL to different satellite data in hopes of proving that GAL is not limited to HICO. If GAL was proven to work with any satellite data, given the parameters of an image and navigational data, the contribution to the field would increase greatly. At the moment most satellites have shadow detection based on what their satellite offers. GAL would be universal and would also apply to images that may no longer have full satellite data available.

APPENDIX

Complete Sample Results

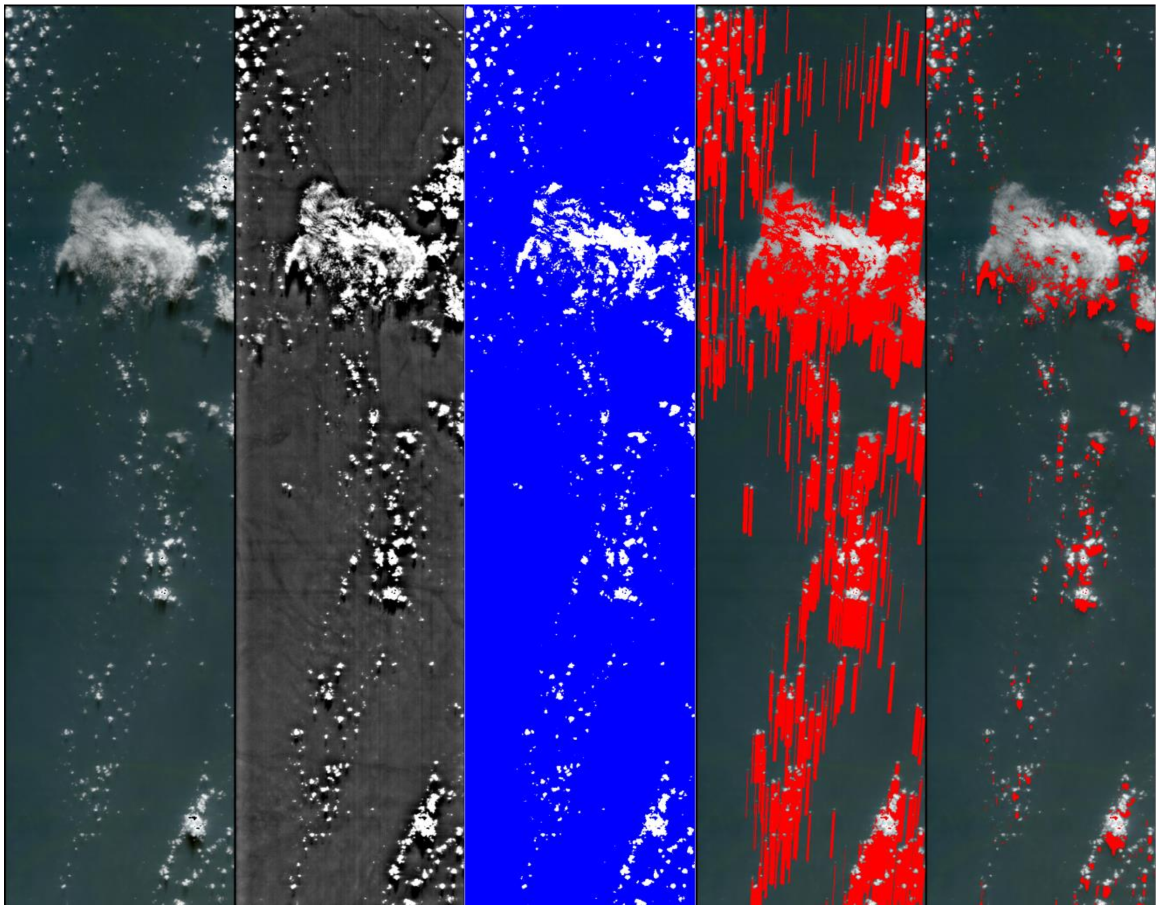

Figure. Arabian Sea 2014 GAL step progression.

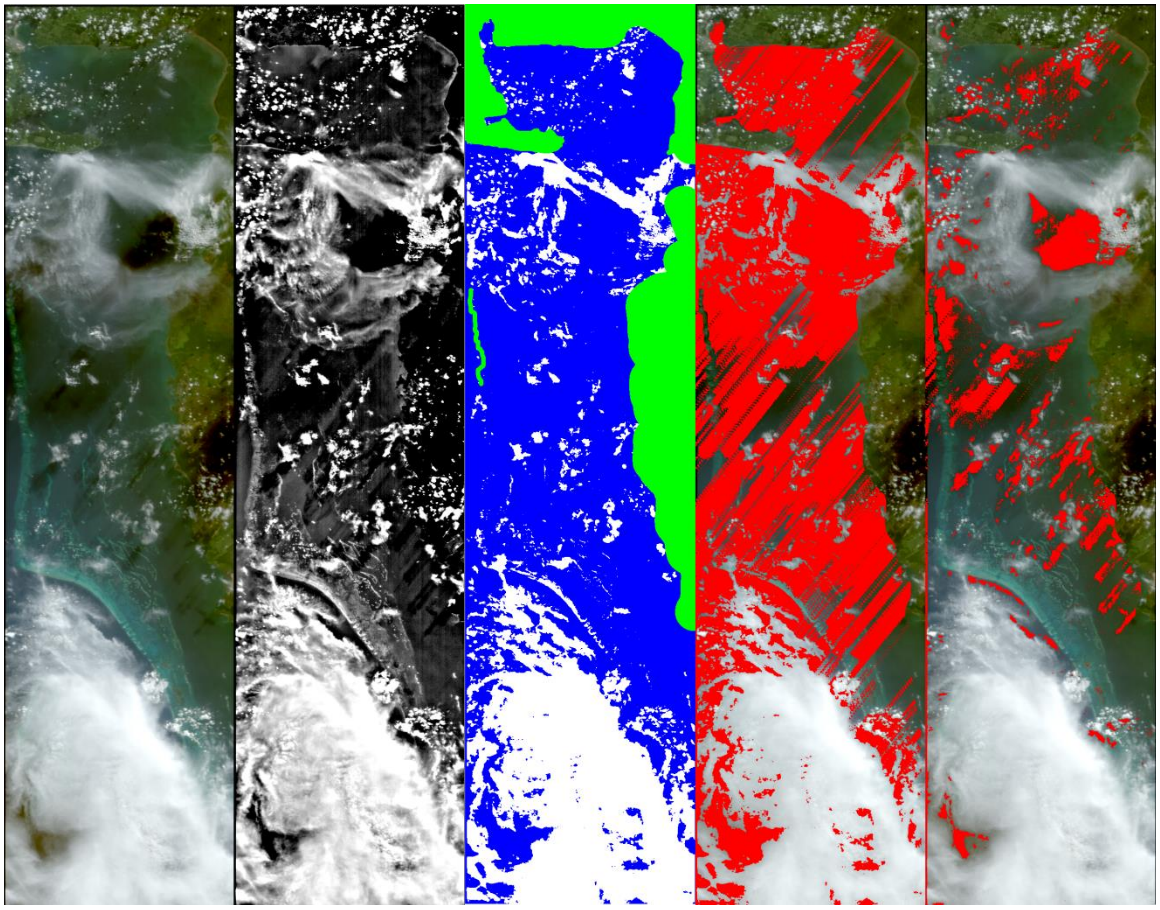

Figure. BLZ 2013 GAL step progression.

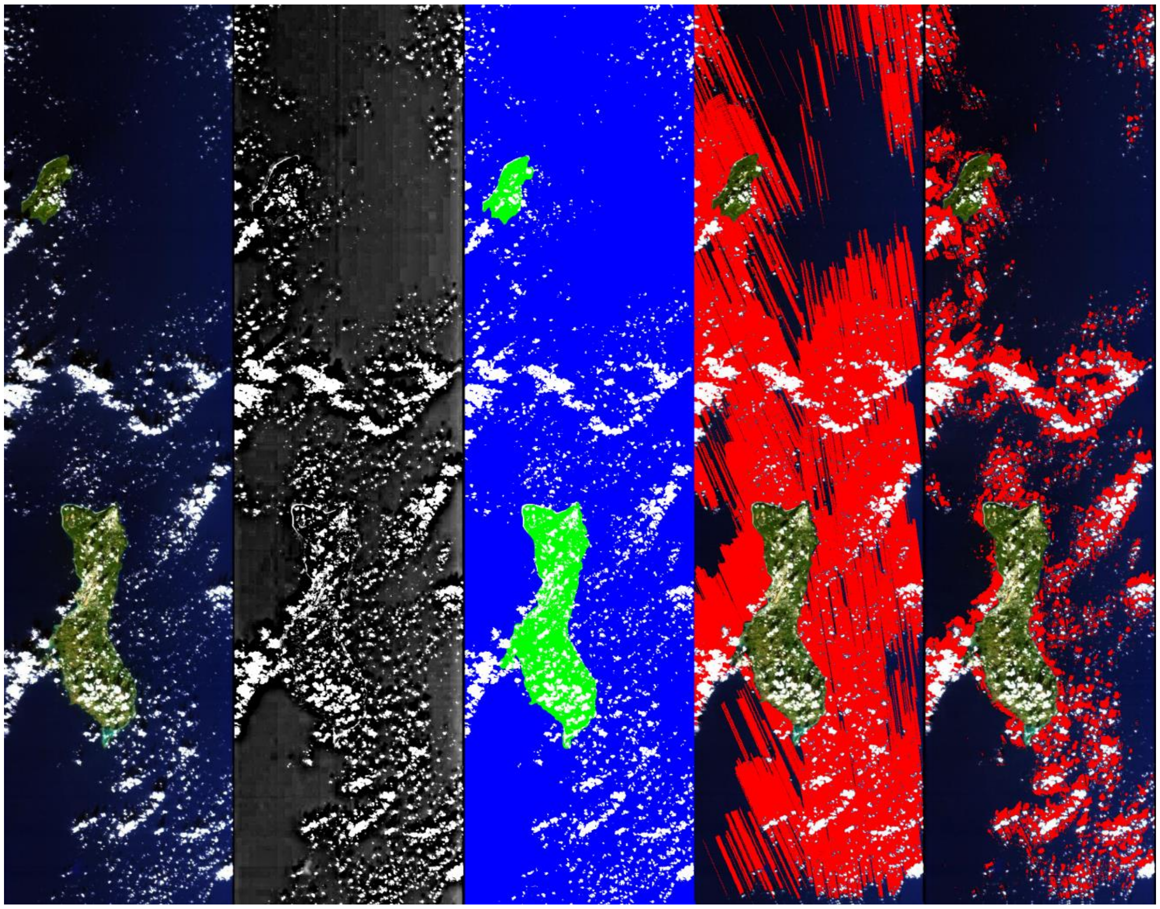

Figure. Guam 2010 GAL step progression.

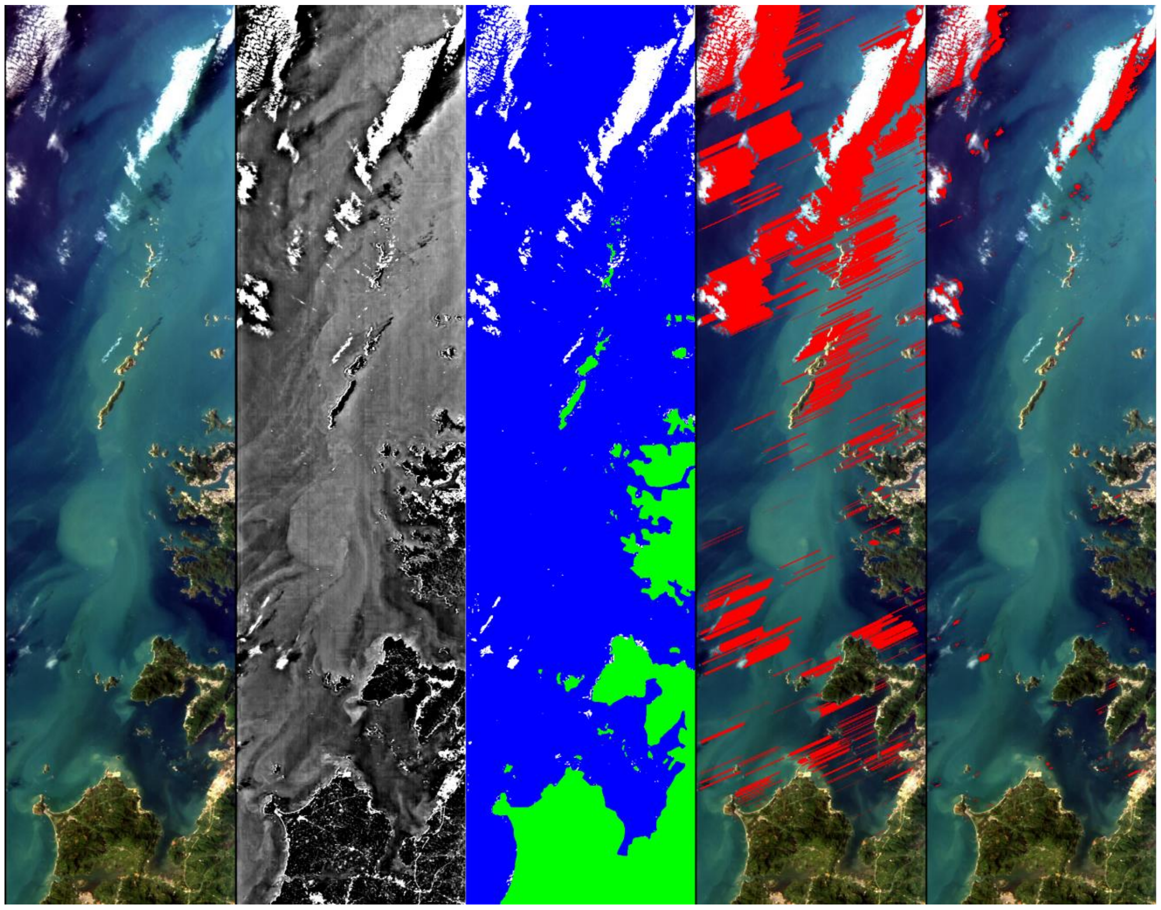

Figure. Hong Kong 2009 GAL step progression.

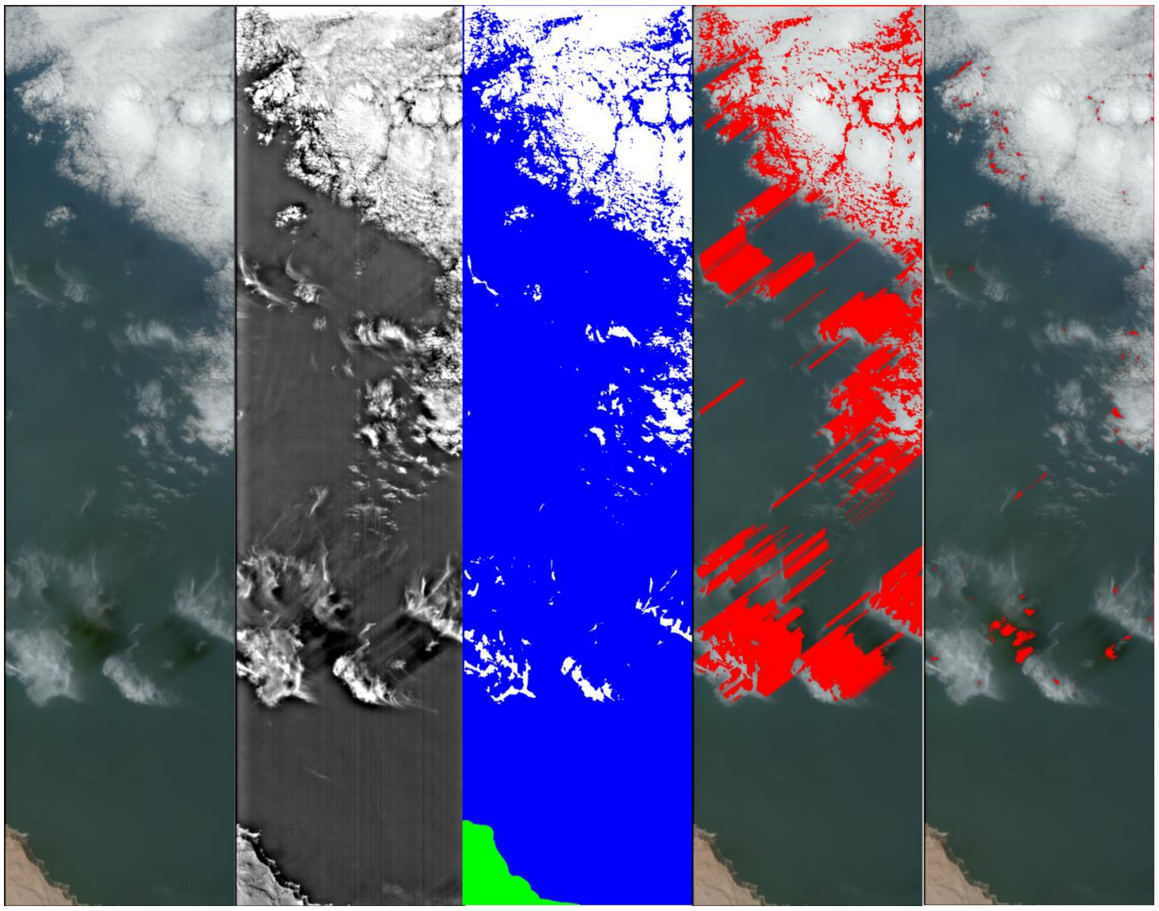

Figure. Iquique Chile 2014 GAL step progression.

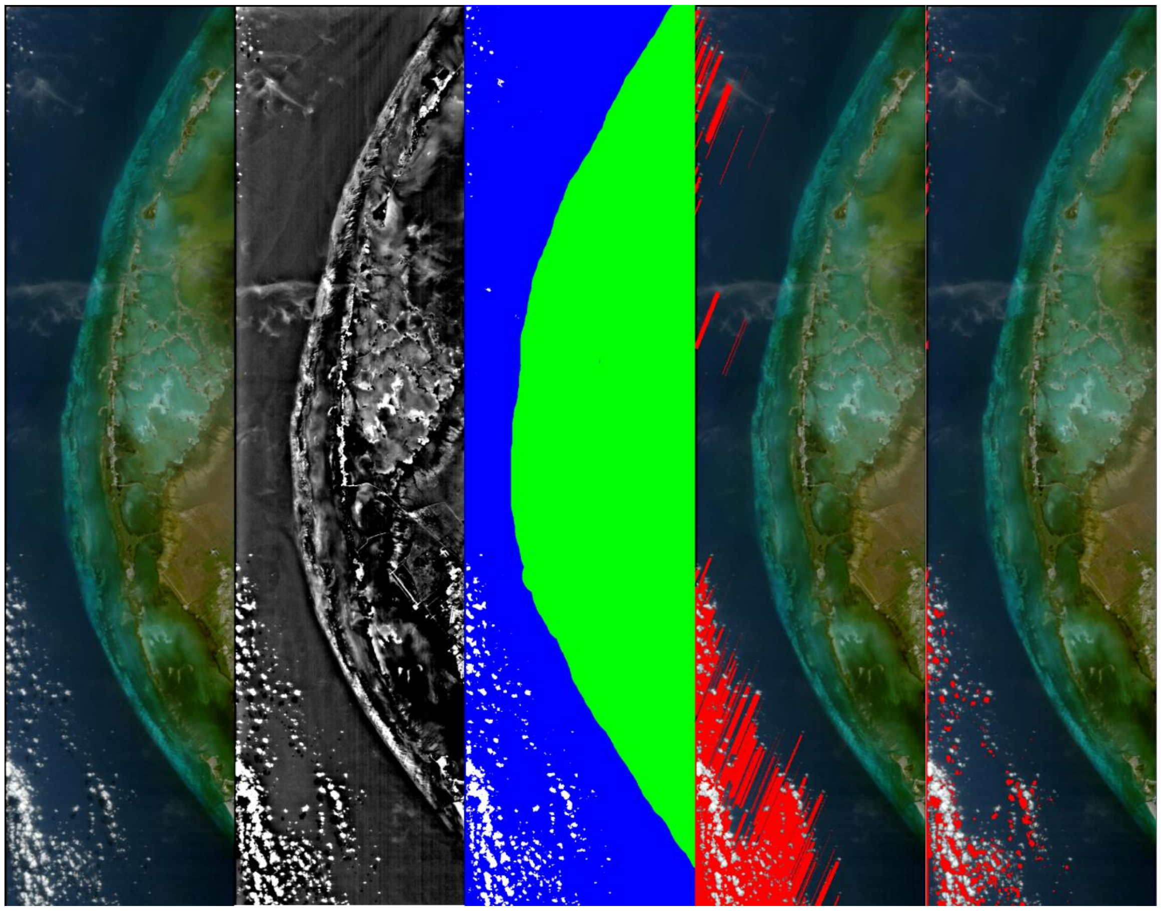

Figure. Key Largo 2013 GAL step progression.

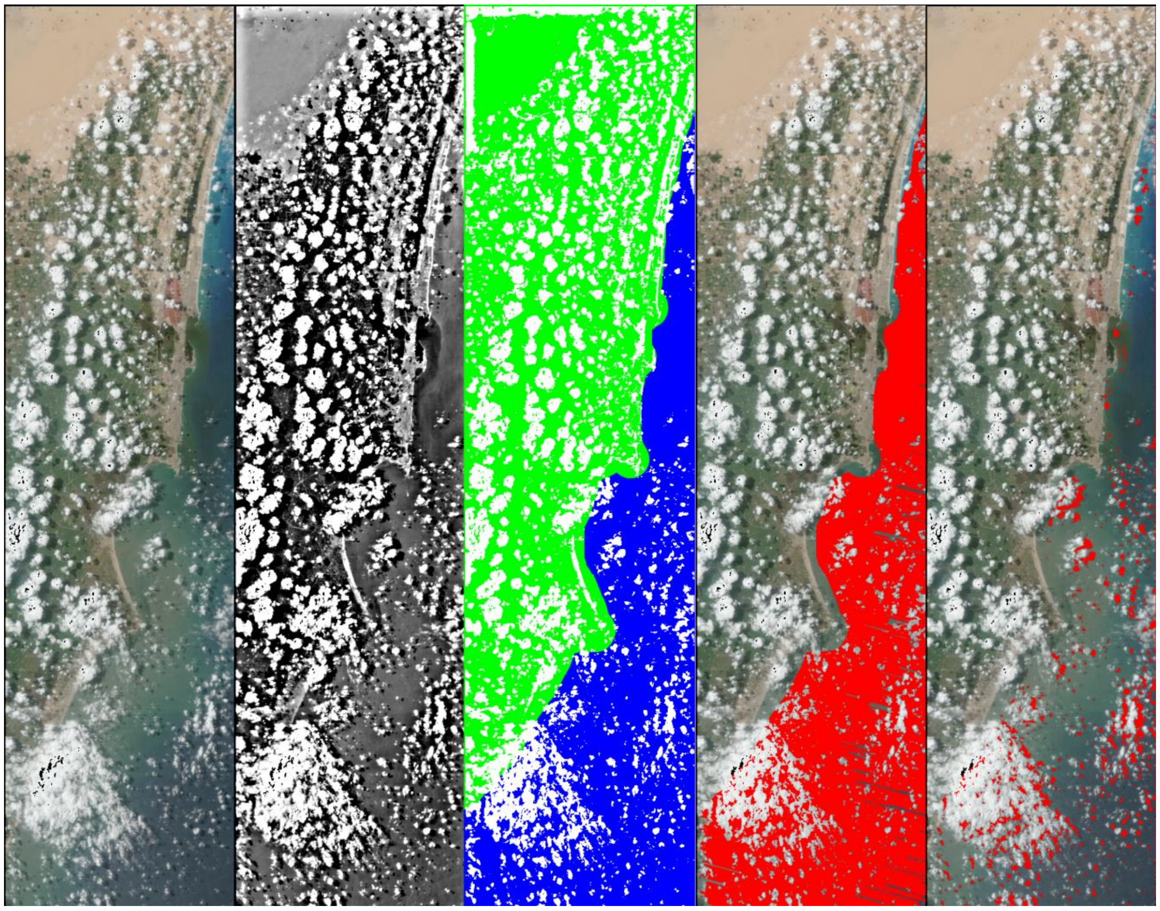

Figure. Lake Idku 2014 GAL step progression.

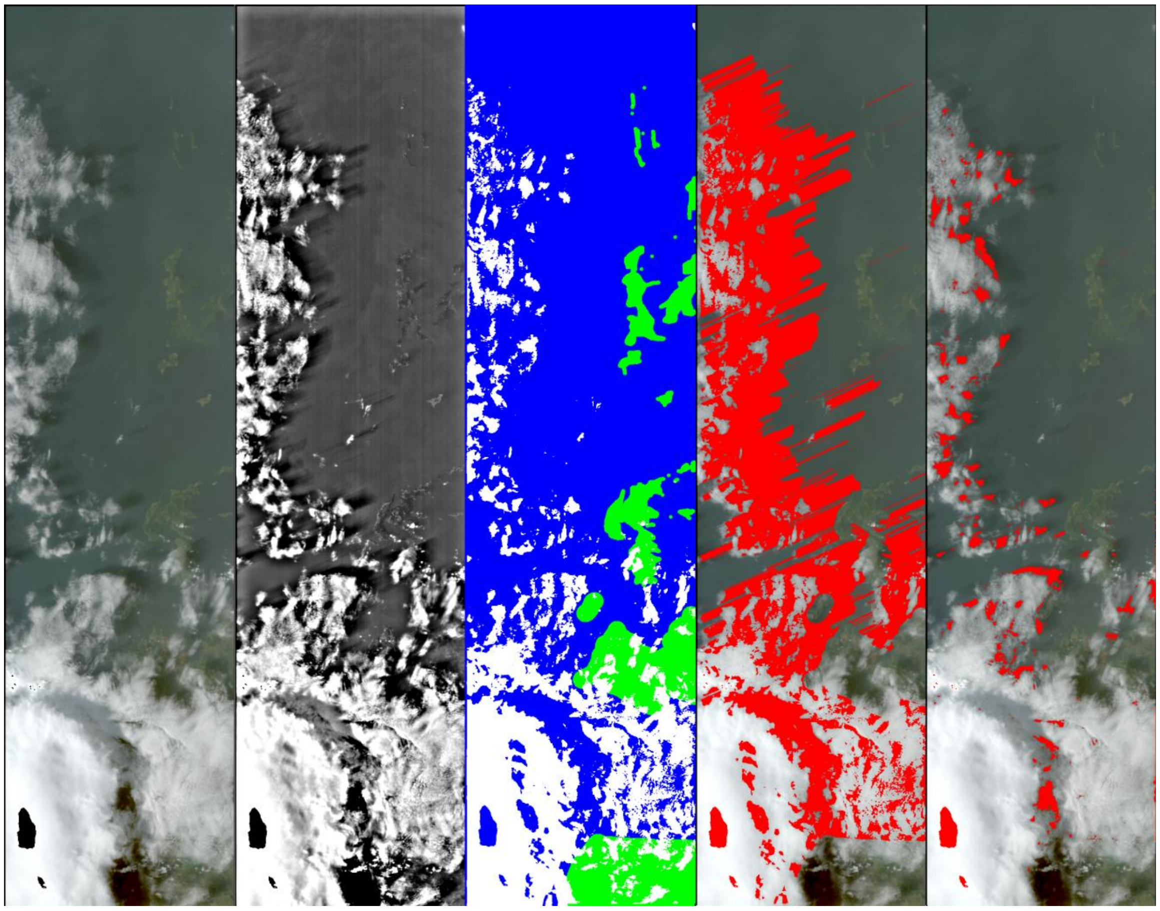

Figure. Lake Victoria 2014 GAL step progression.

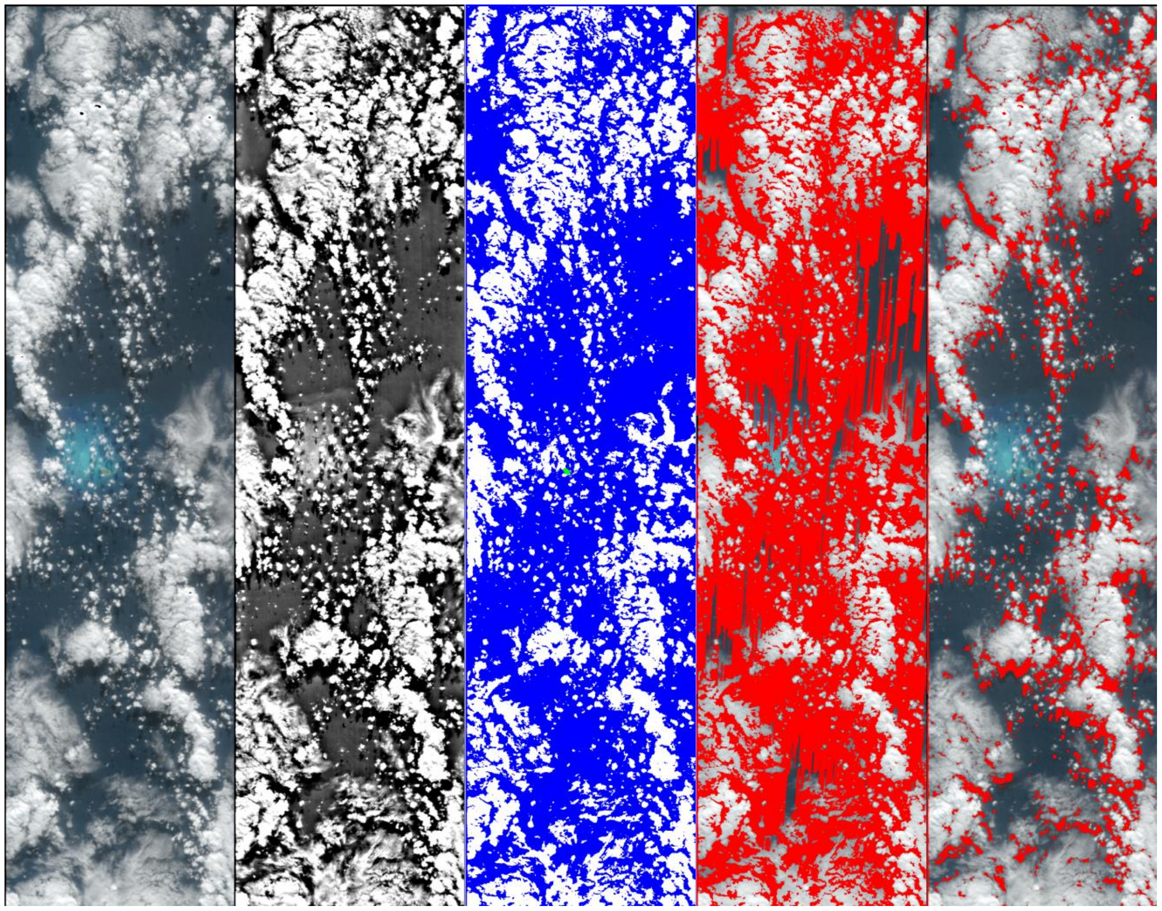

Figure. Lisianski 2014 GAL step progression.

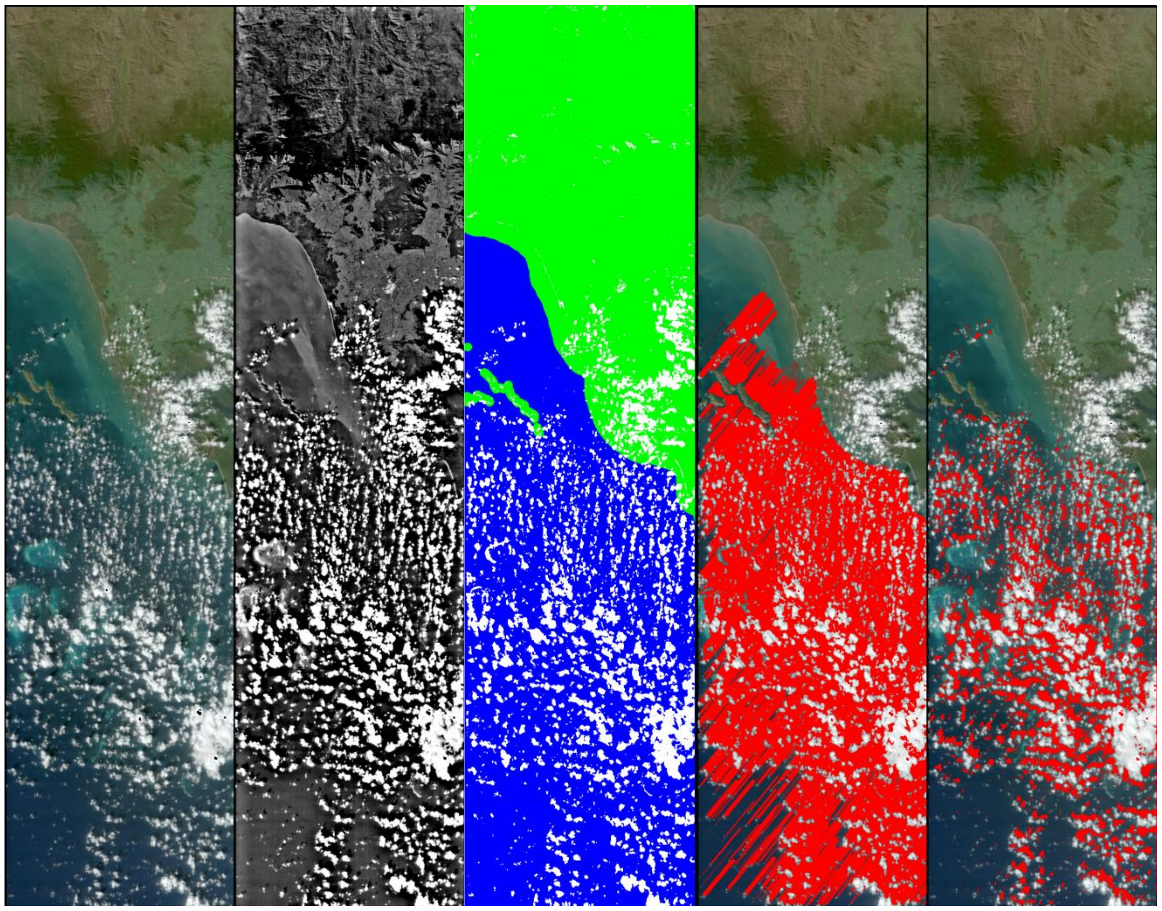

Figure. Lucinda Jetty 2014 GAL step progression.

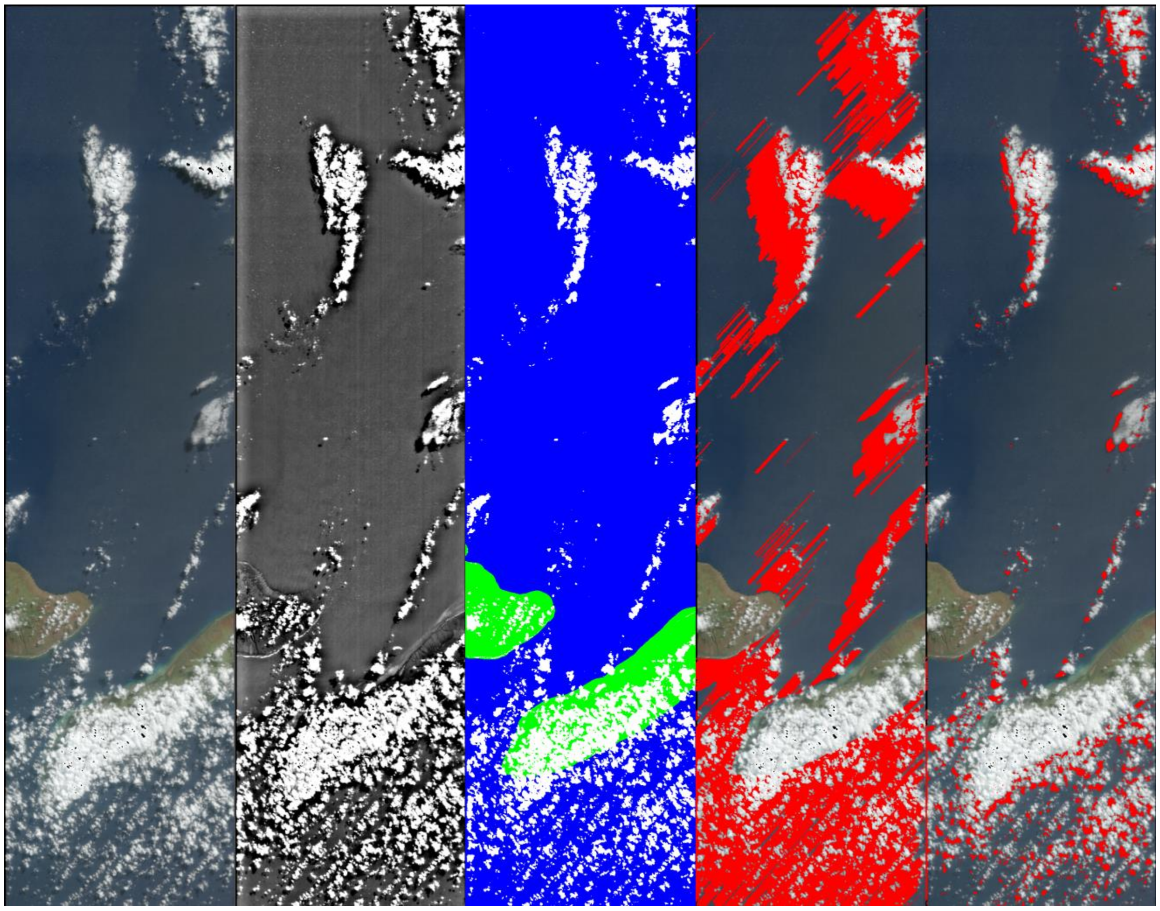

Figure. MOBY 2014 GAL step progression.

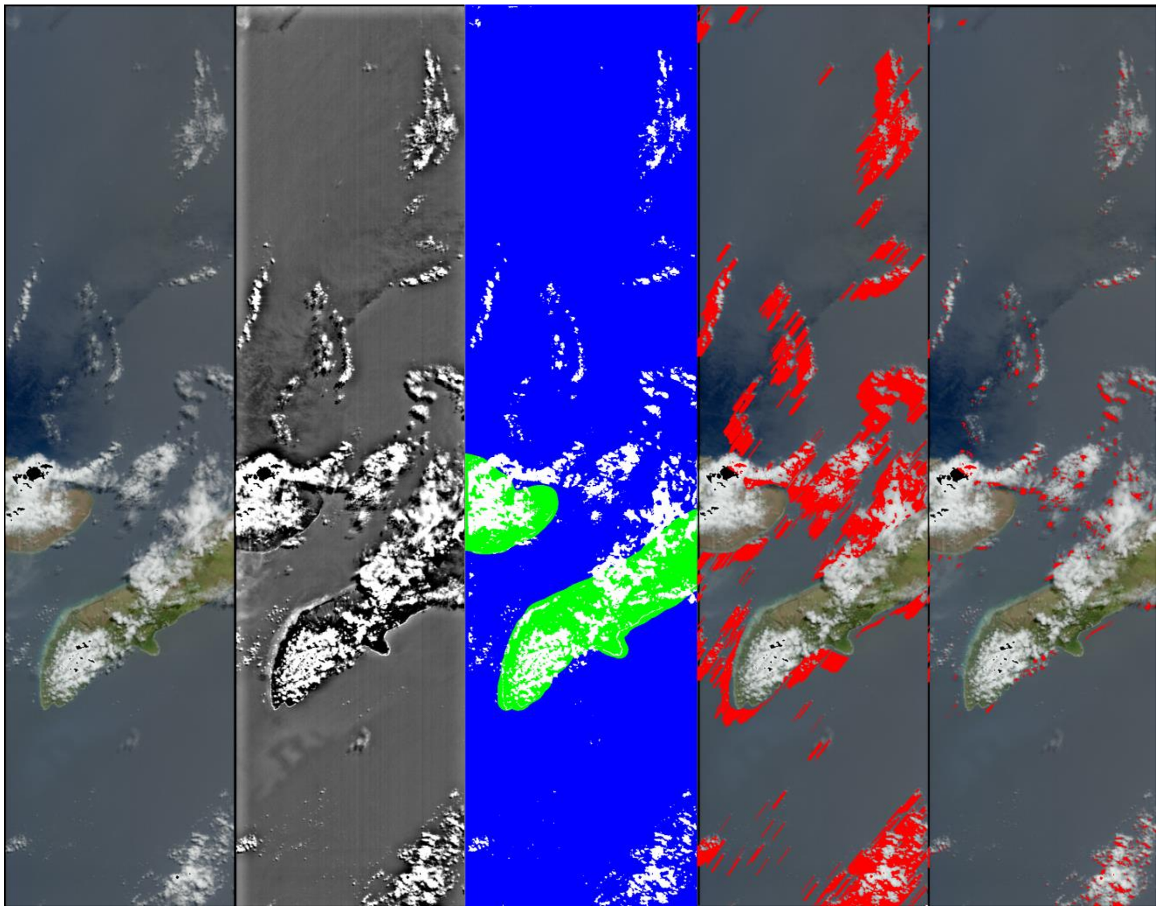

Figure. MOBY-2 2014 GAL step progression.

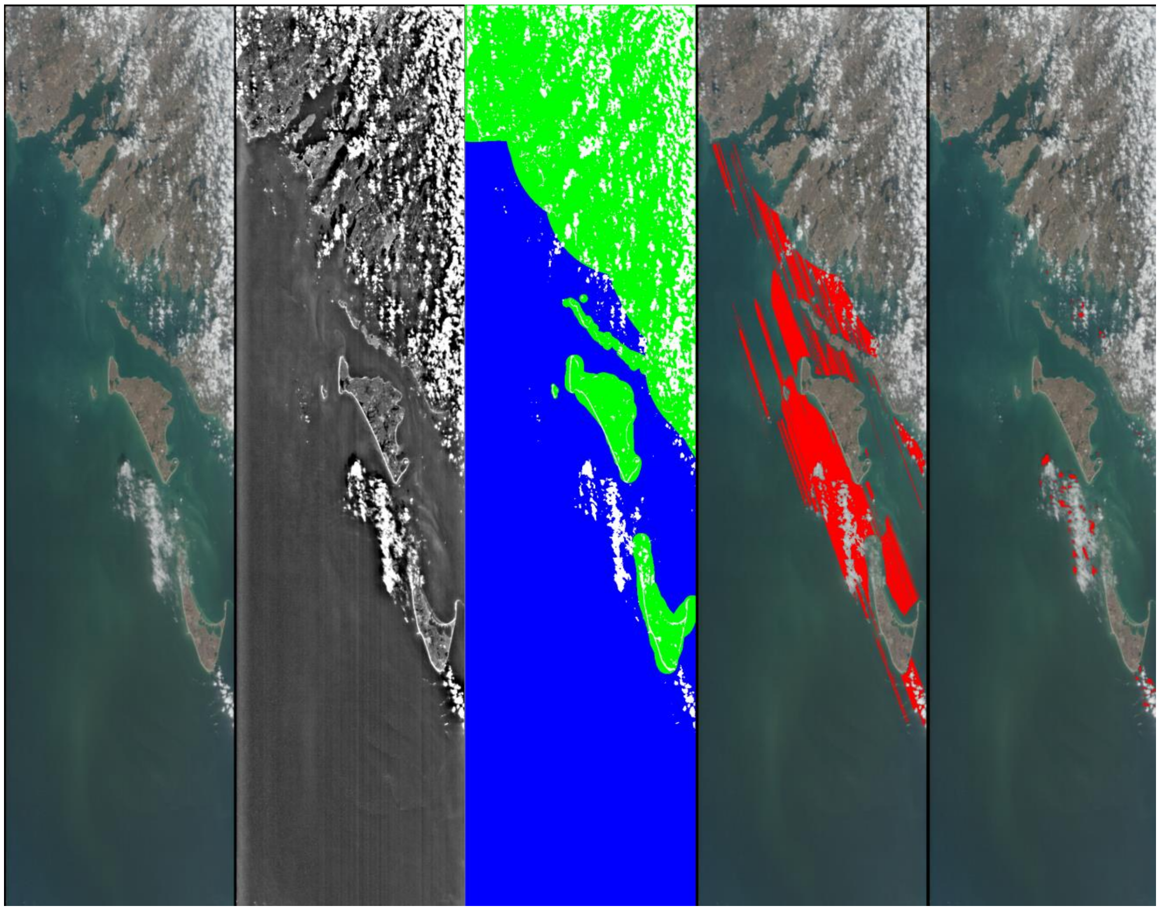

Figure. MVCO 2014 GAL step progression.

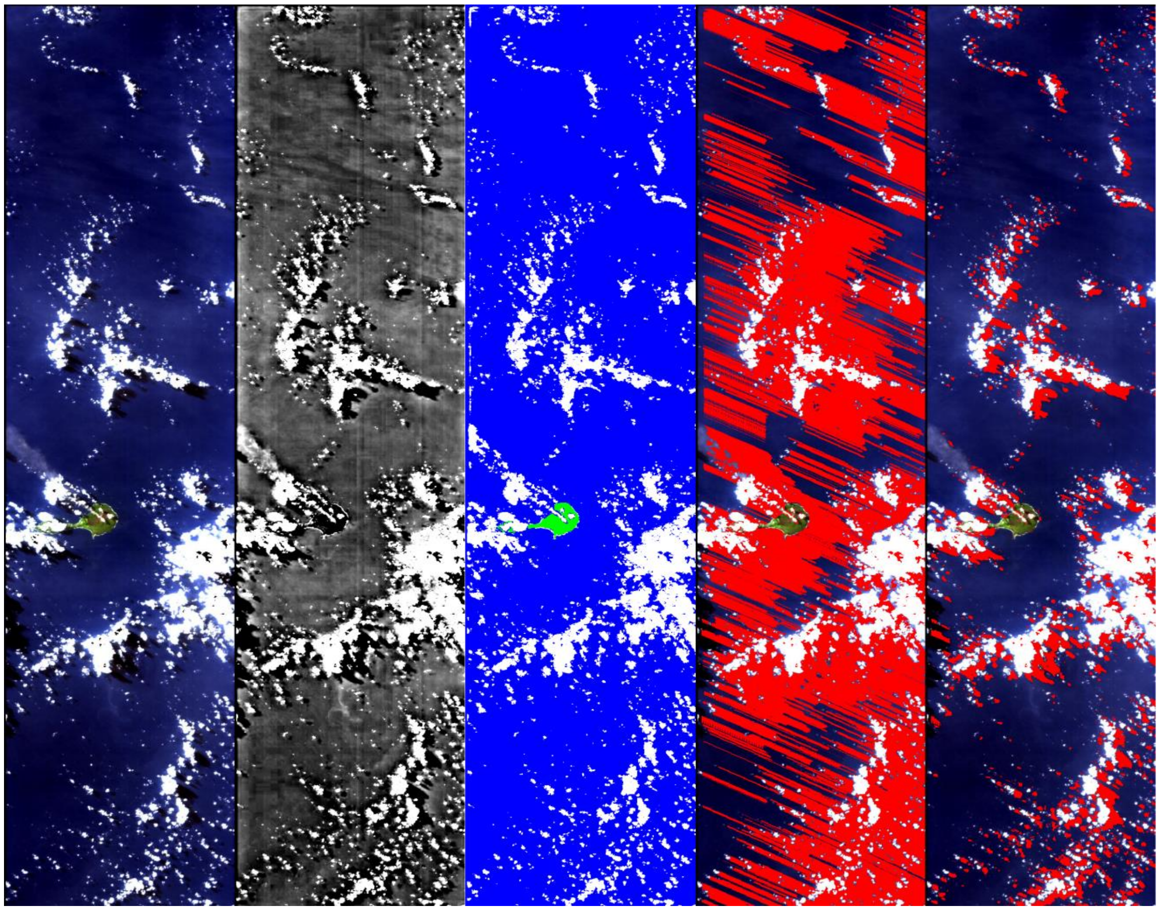

Figure. N. Mariana Pagan 2010 GAL step progression.

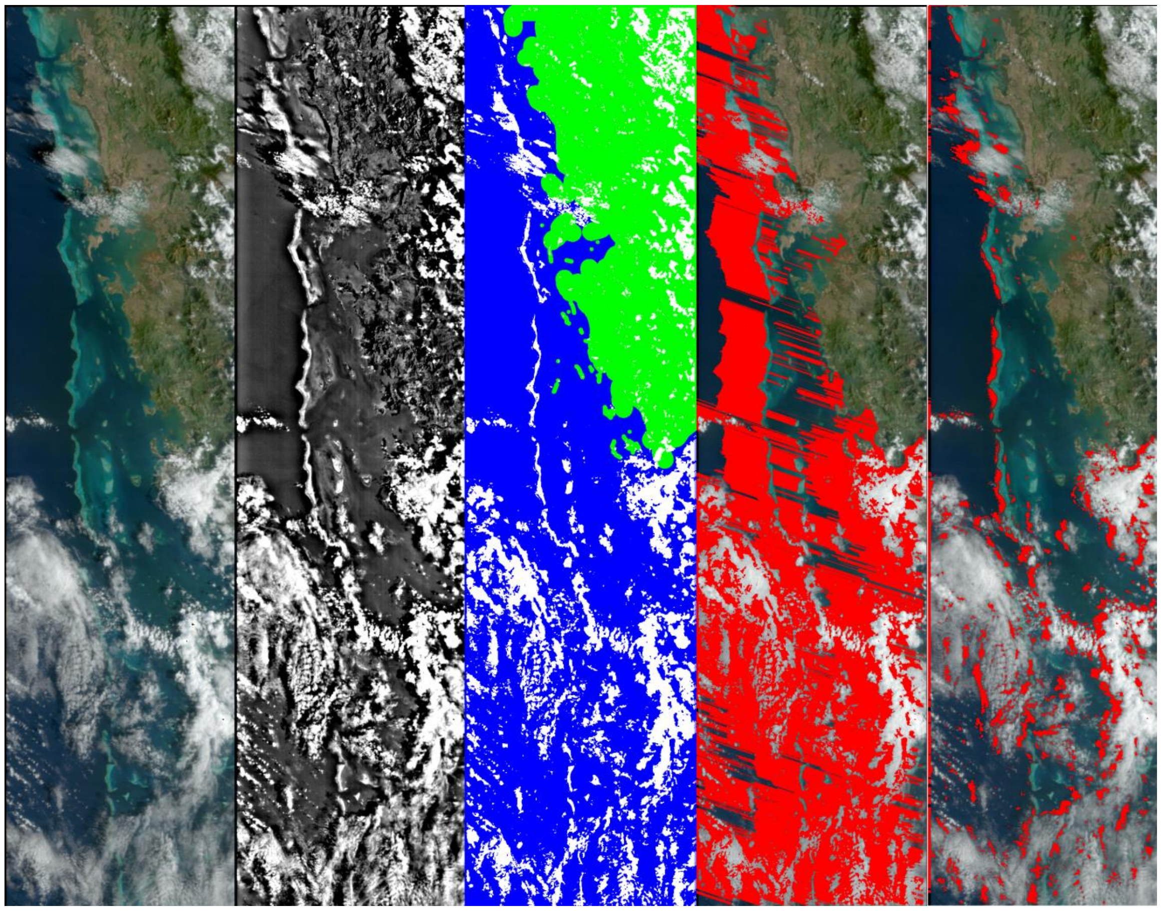

Figure. Noumea New Caledonia 2014 GAL step progression.

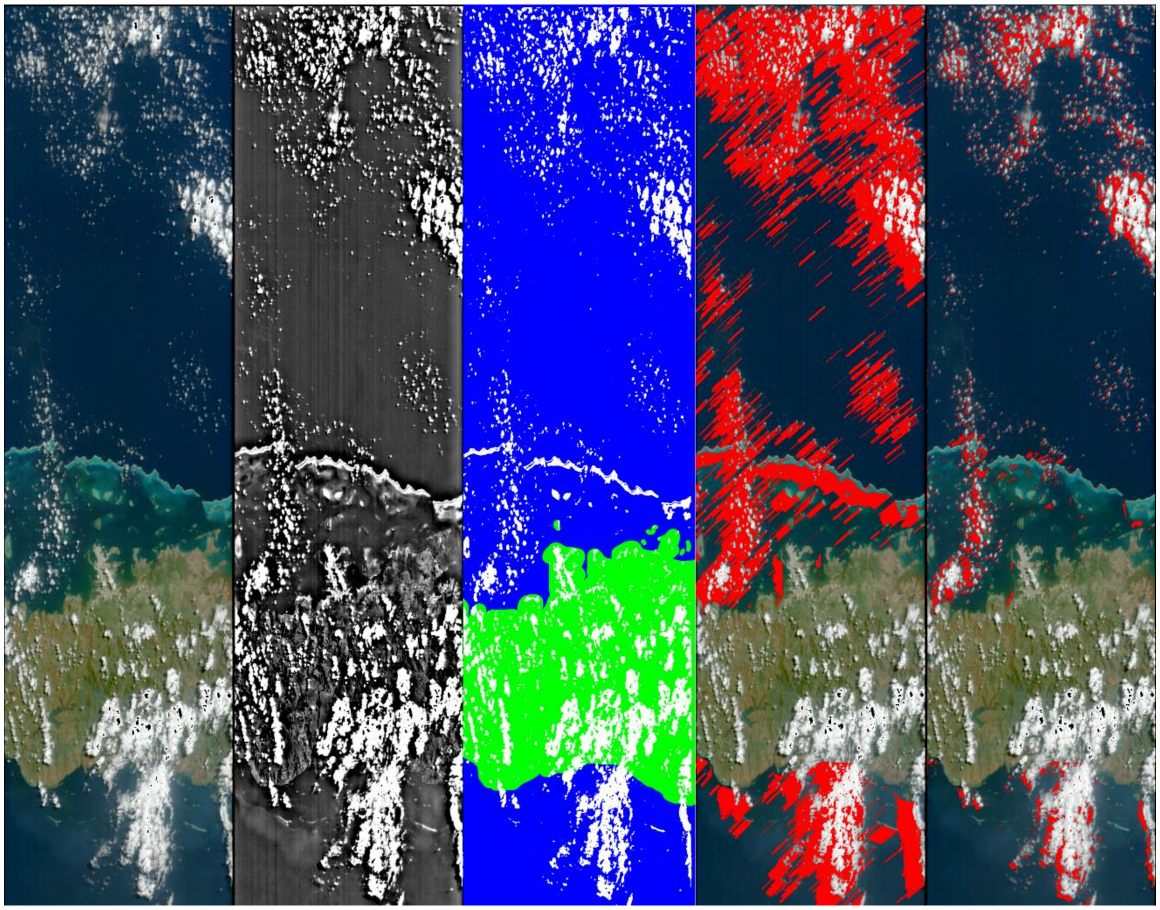

Figure. Noumea New Caledonia-2 2014 GAL step progression.

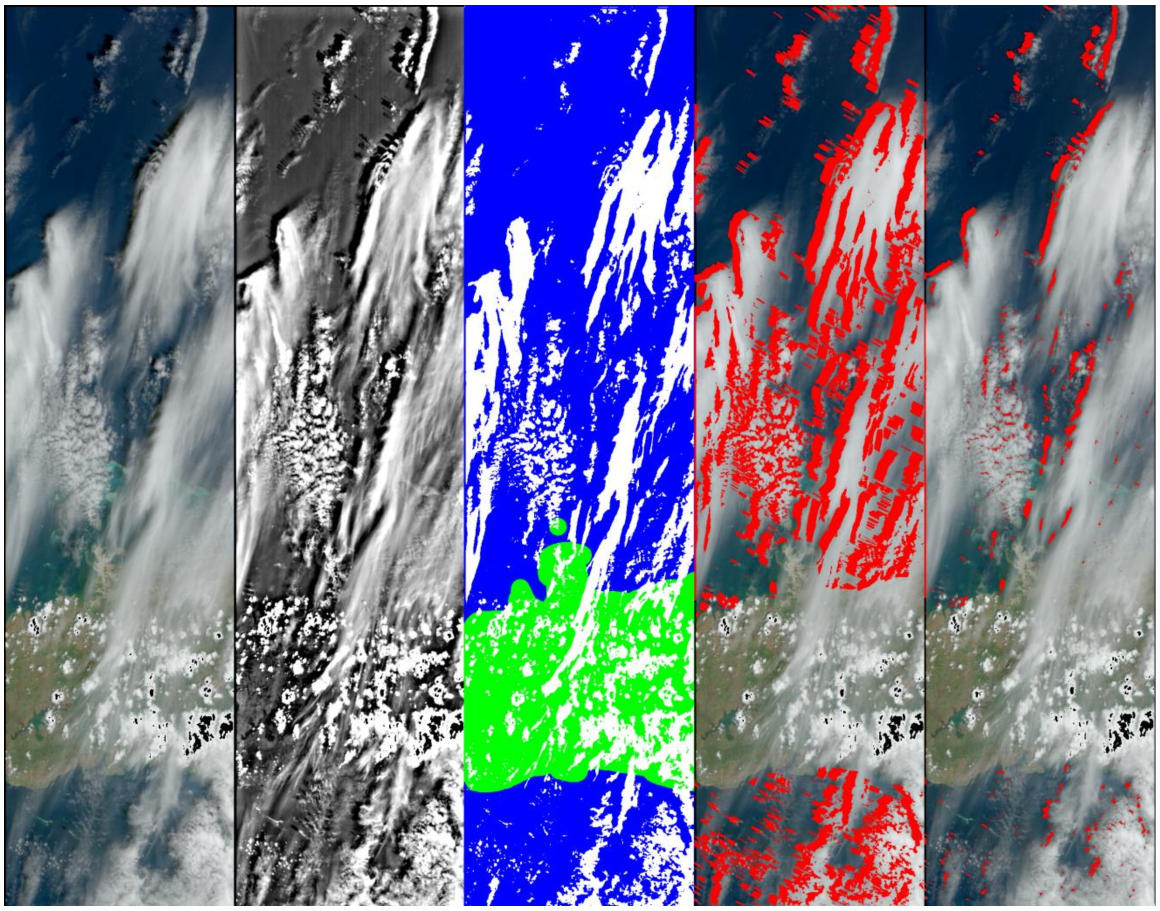

Figure. Noumea New Caledonia-3 2014 GAL step progression.

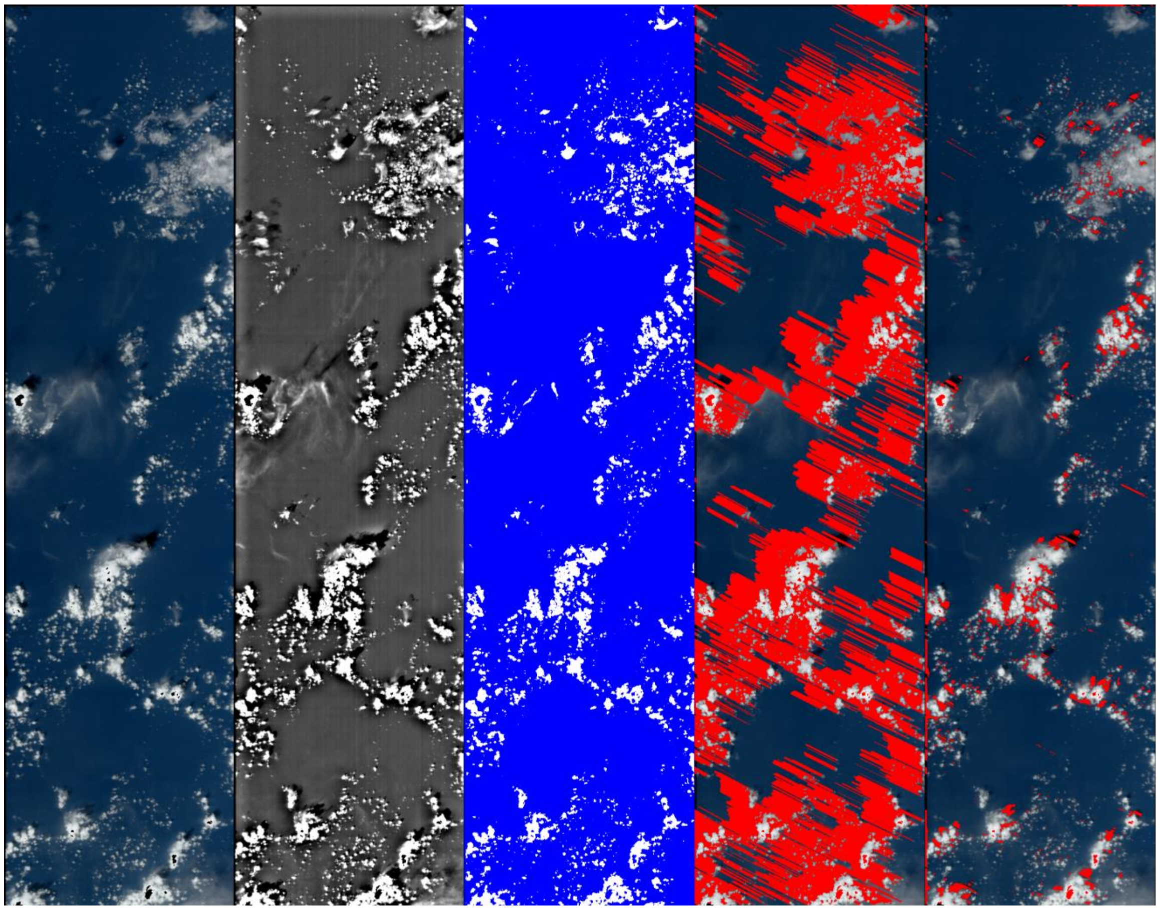

Figure. OC Pac 2014 GAL step progression.

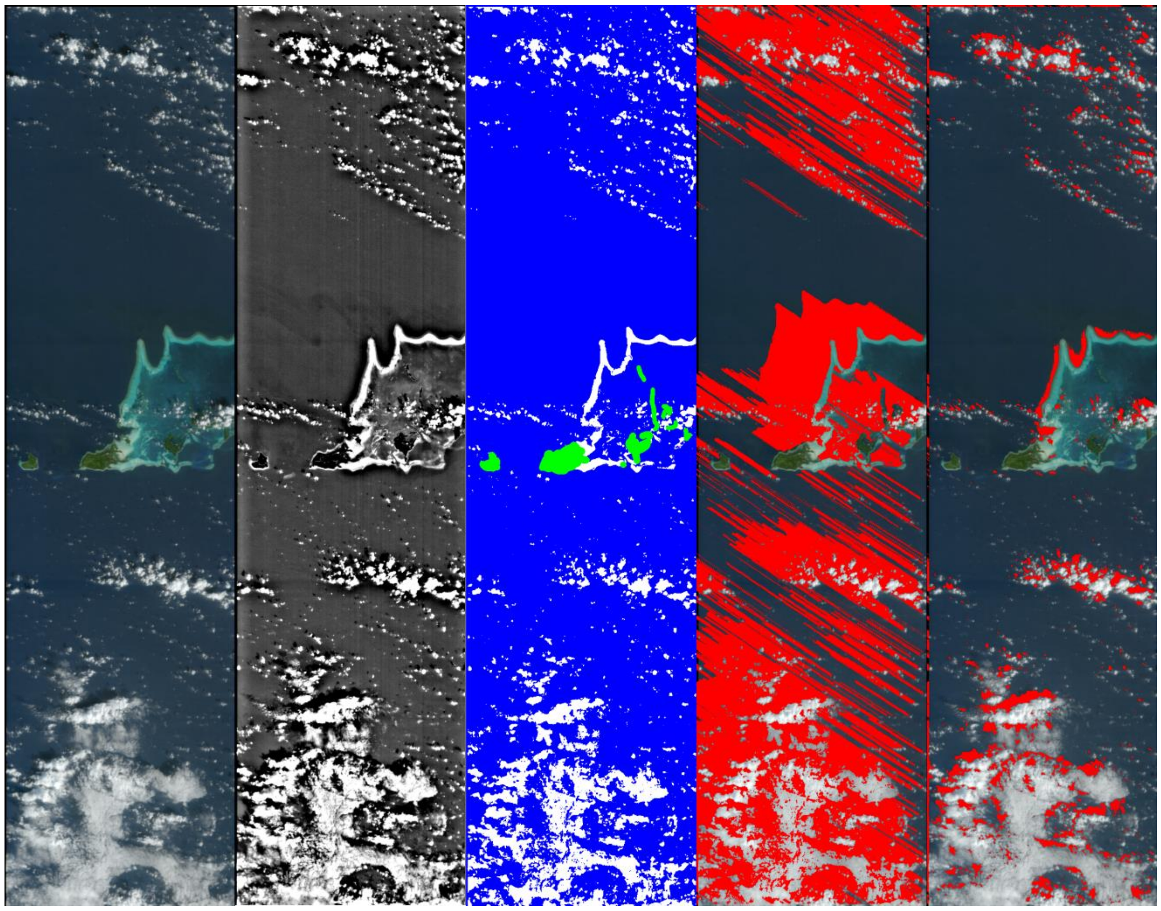

Figure. Palau 2014 GAL step progression.

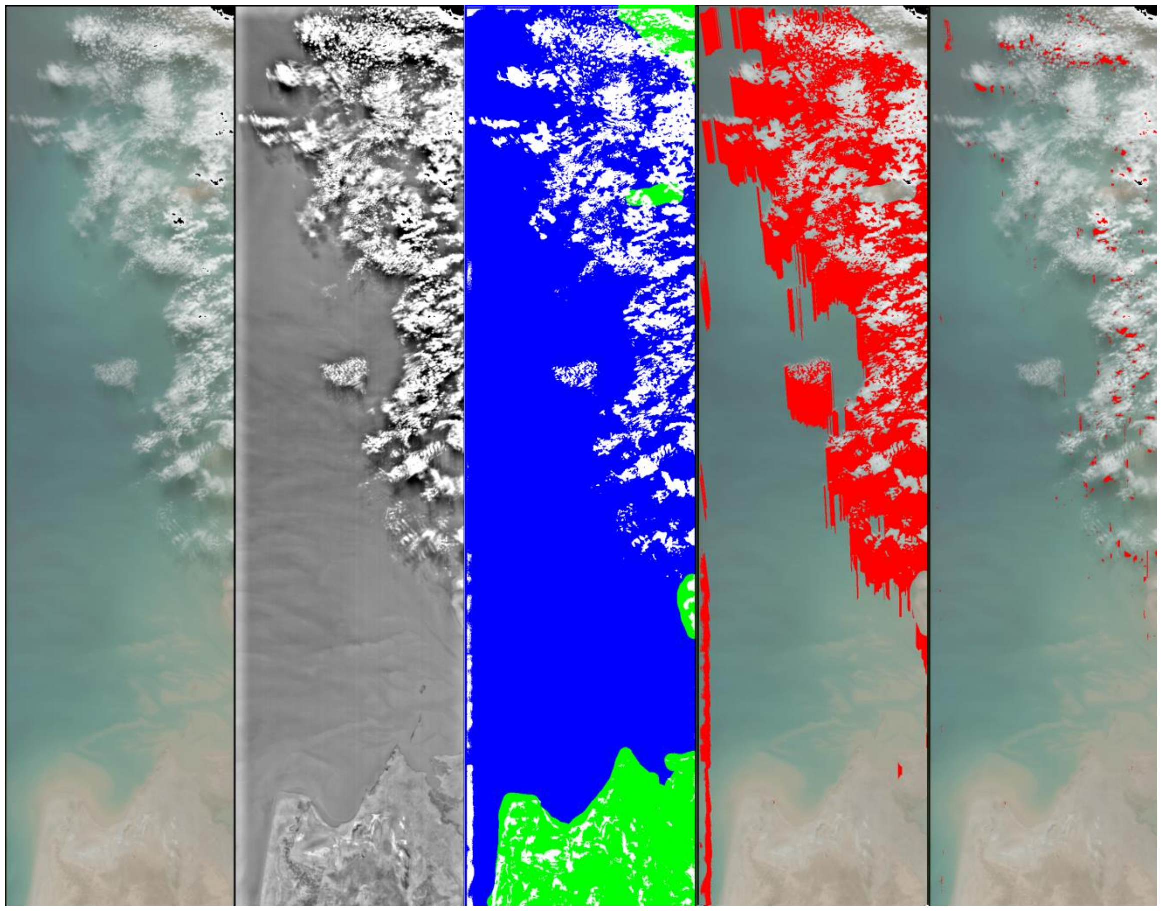

Figure. Persian Gulf 2014 GAL step progression.

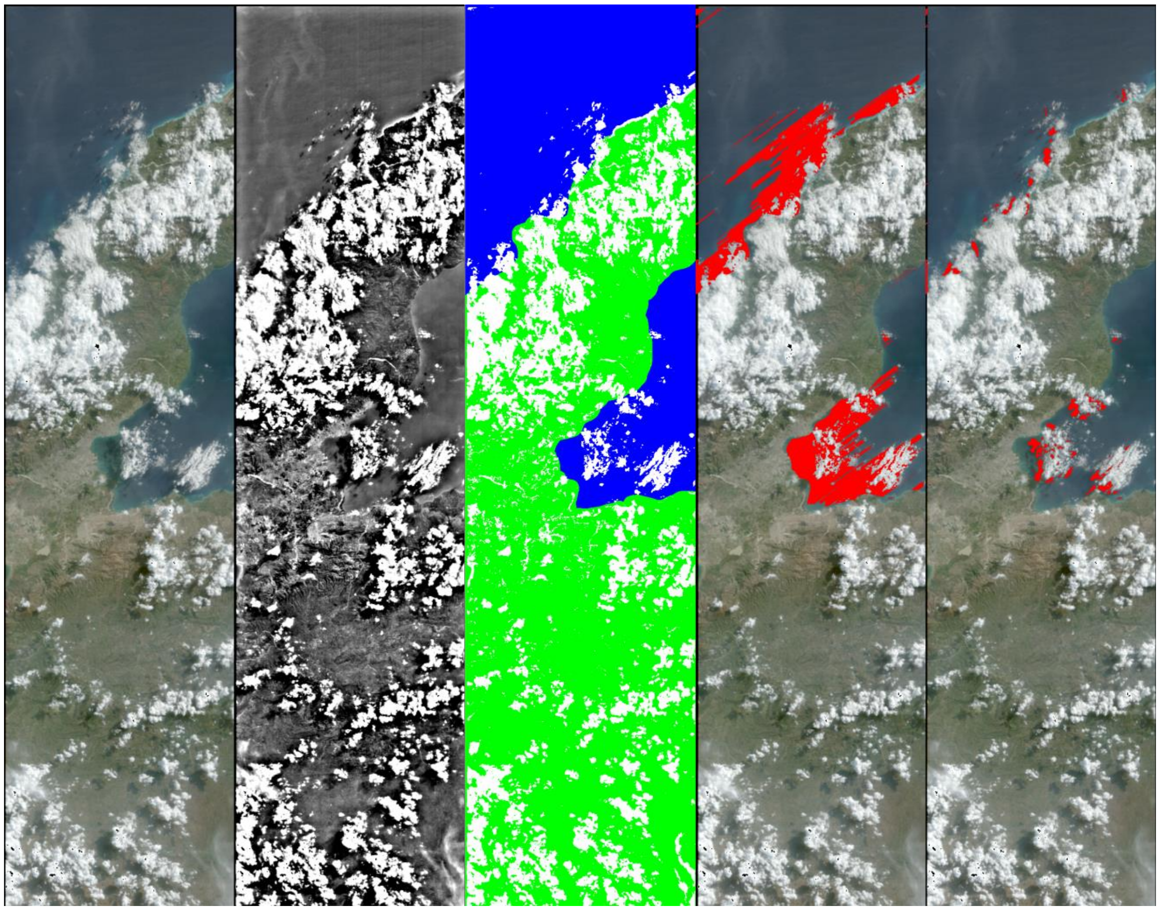

Figure. Port au Prince 2014 GAL step progression.

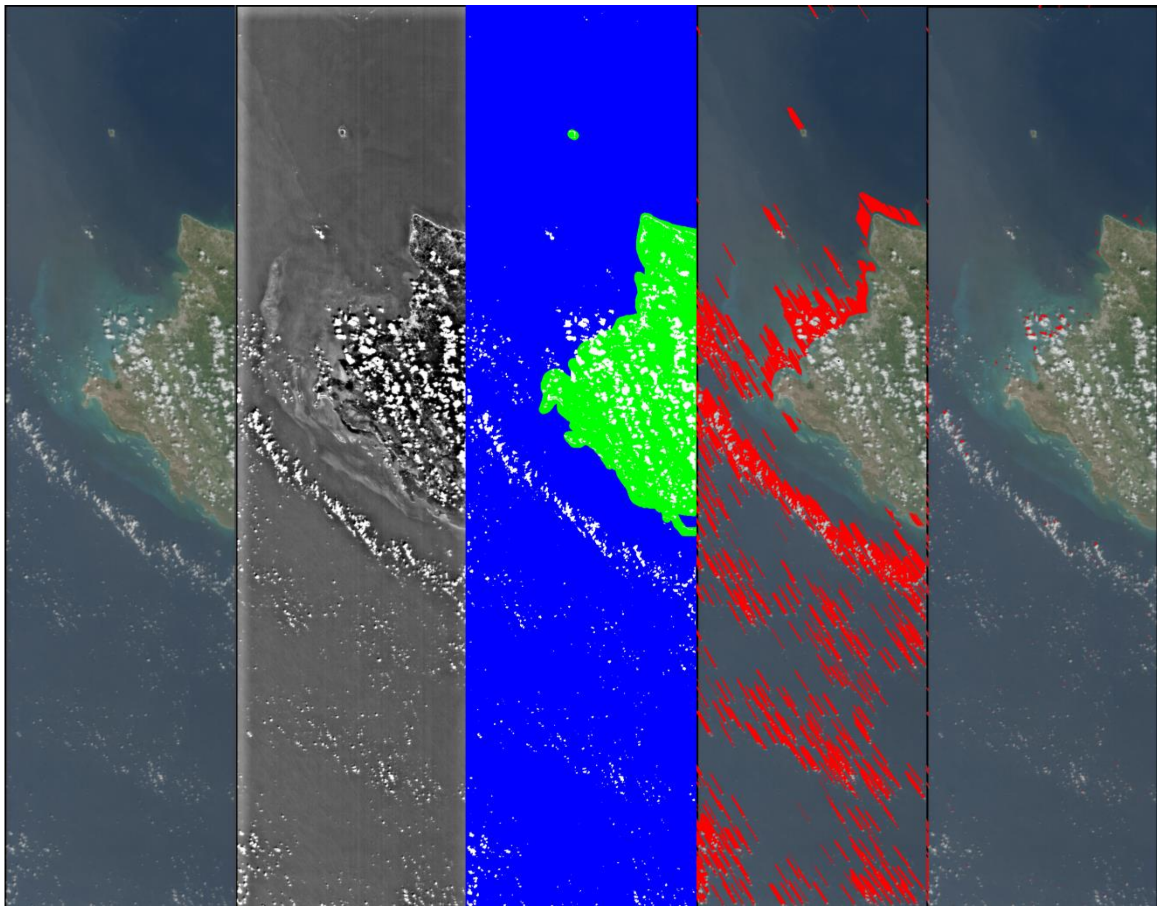

Figure. Puerto Rico 2014 GAL step progression.

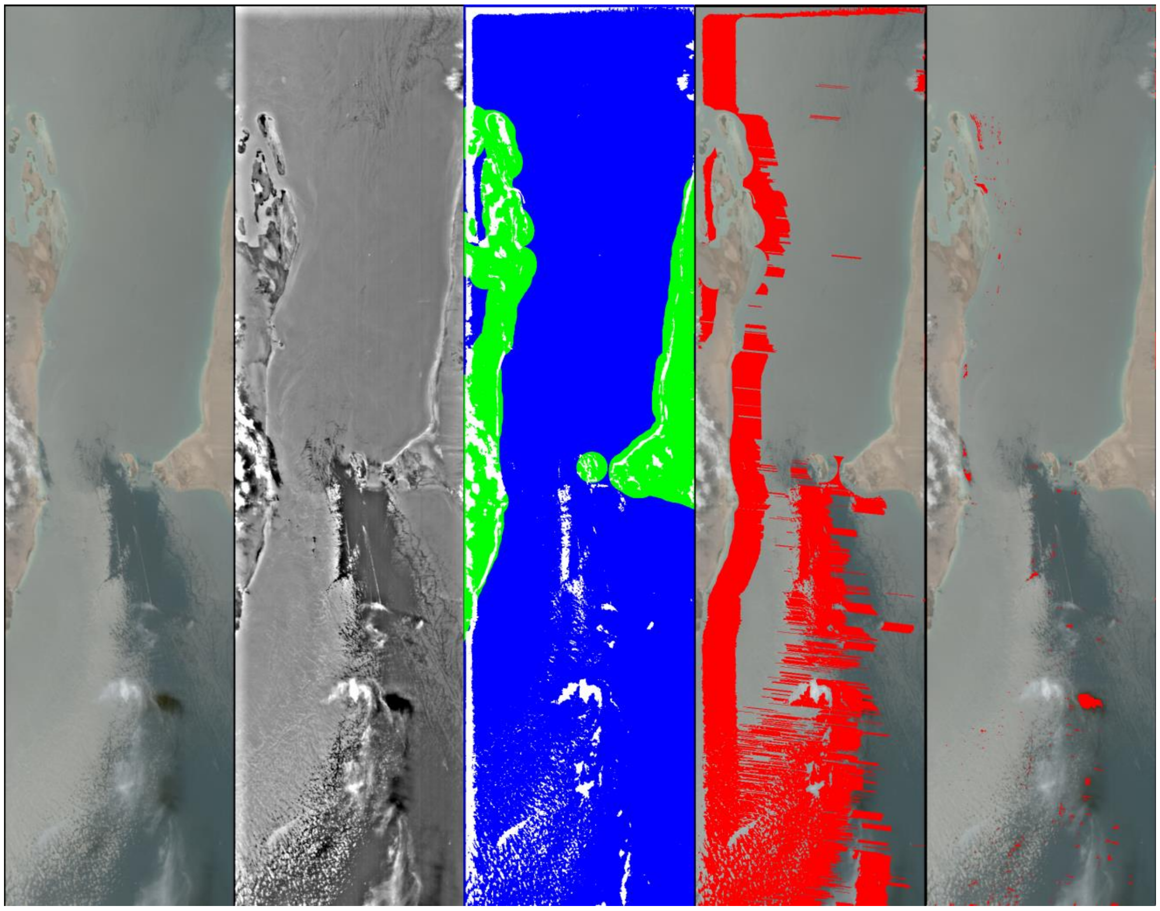

Figure. Red Sea Straits 2014 GAL step progression.

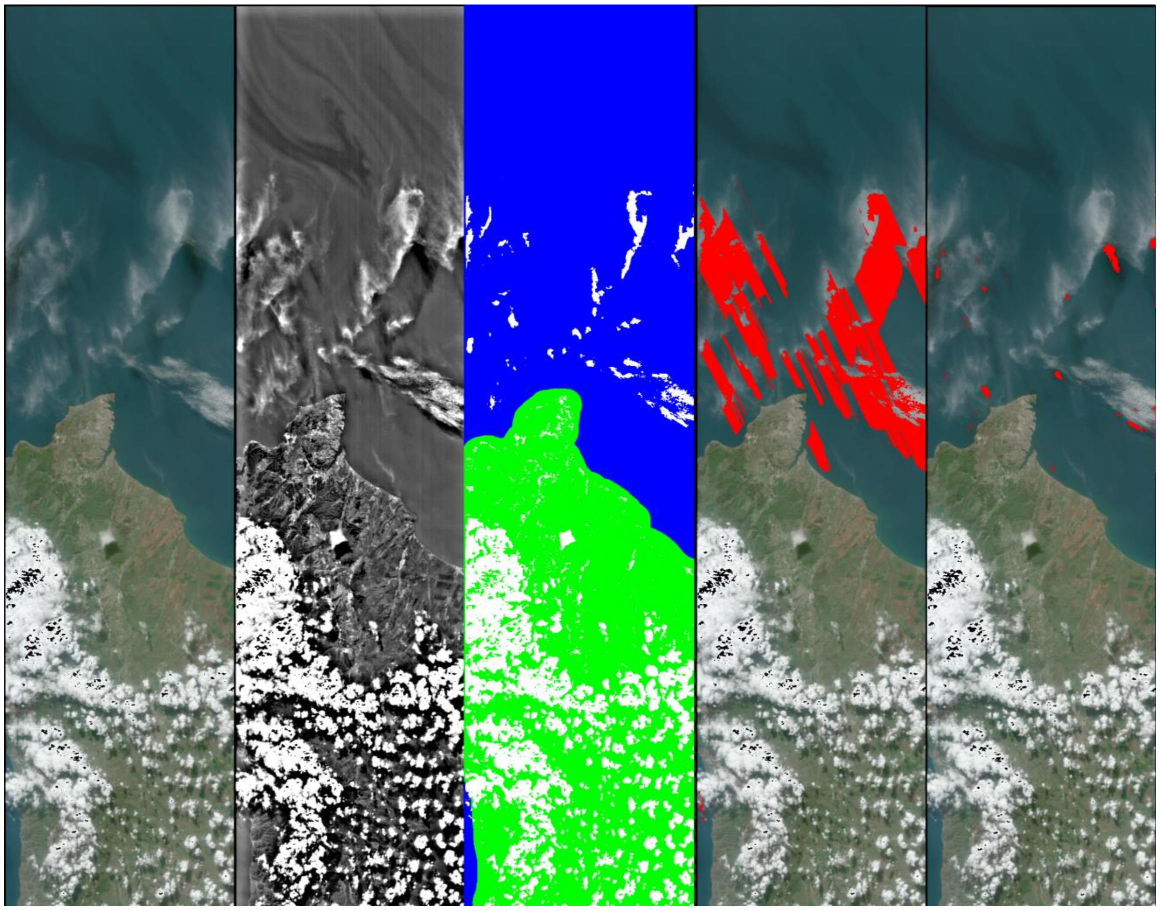

Figure. Sevastopol Crimea 2014 GAL step progression.

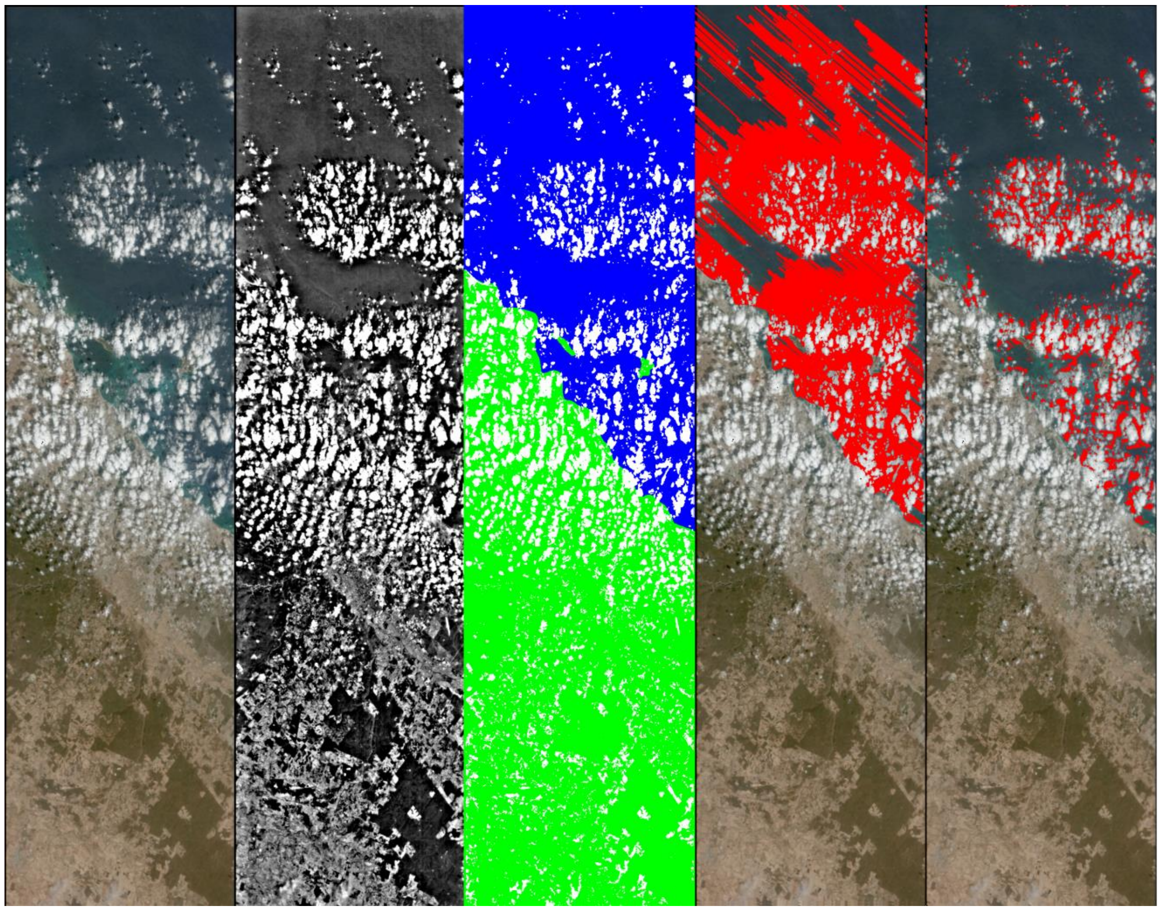

Figure. Swan River Estuary 2014 GAL step progression.

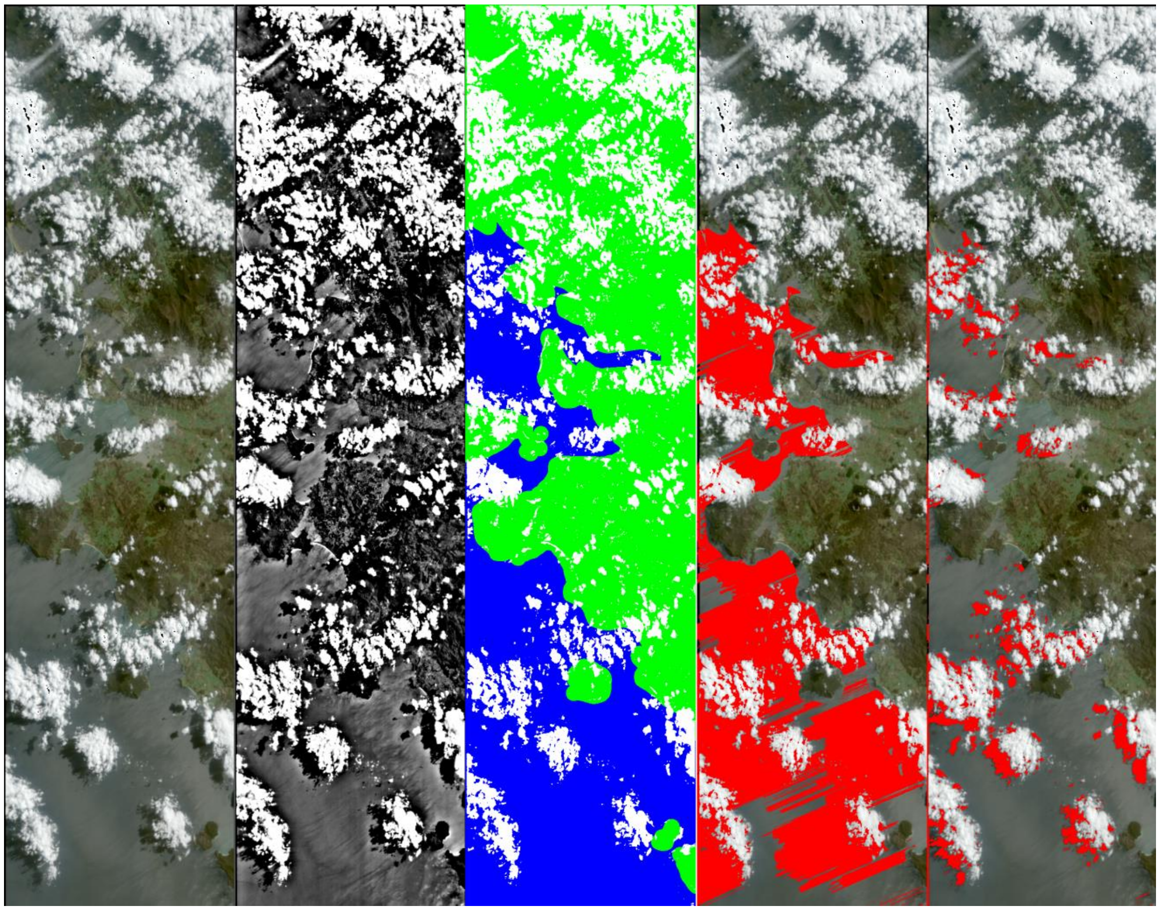

Figure. Tasmania 2013 GAL step progression.

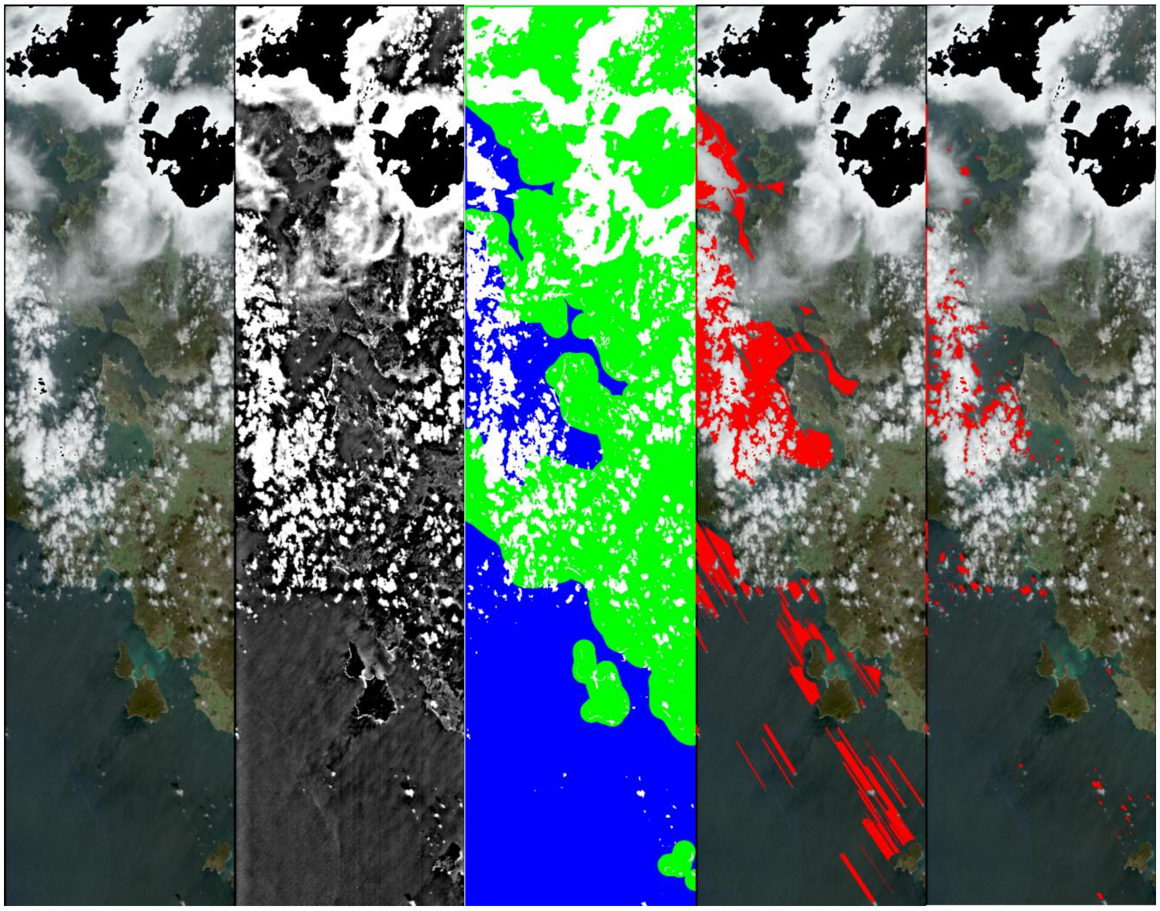

Figure. Tasmania-2 2013 GAL step progression.

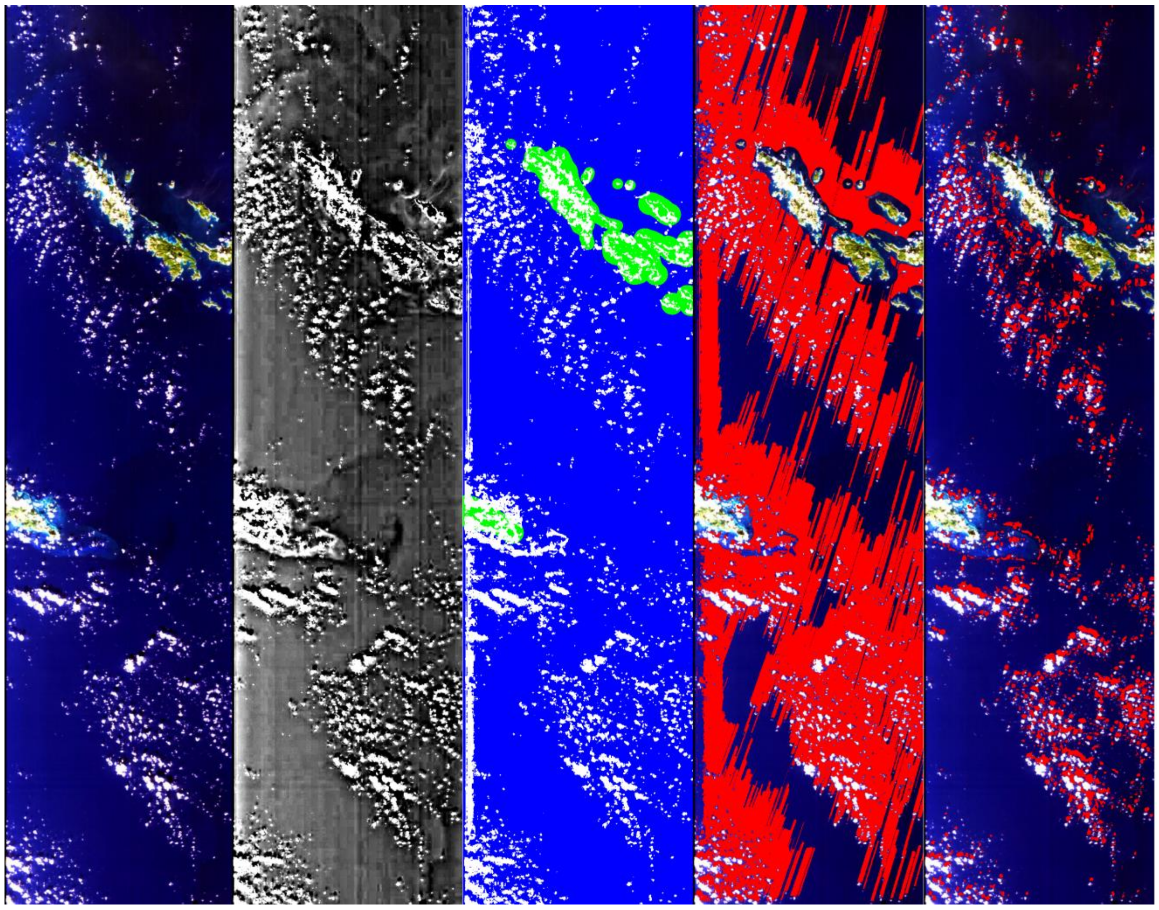

Figure. Virgin Islands 2009 GAL step progression.

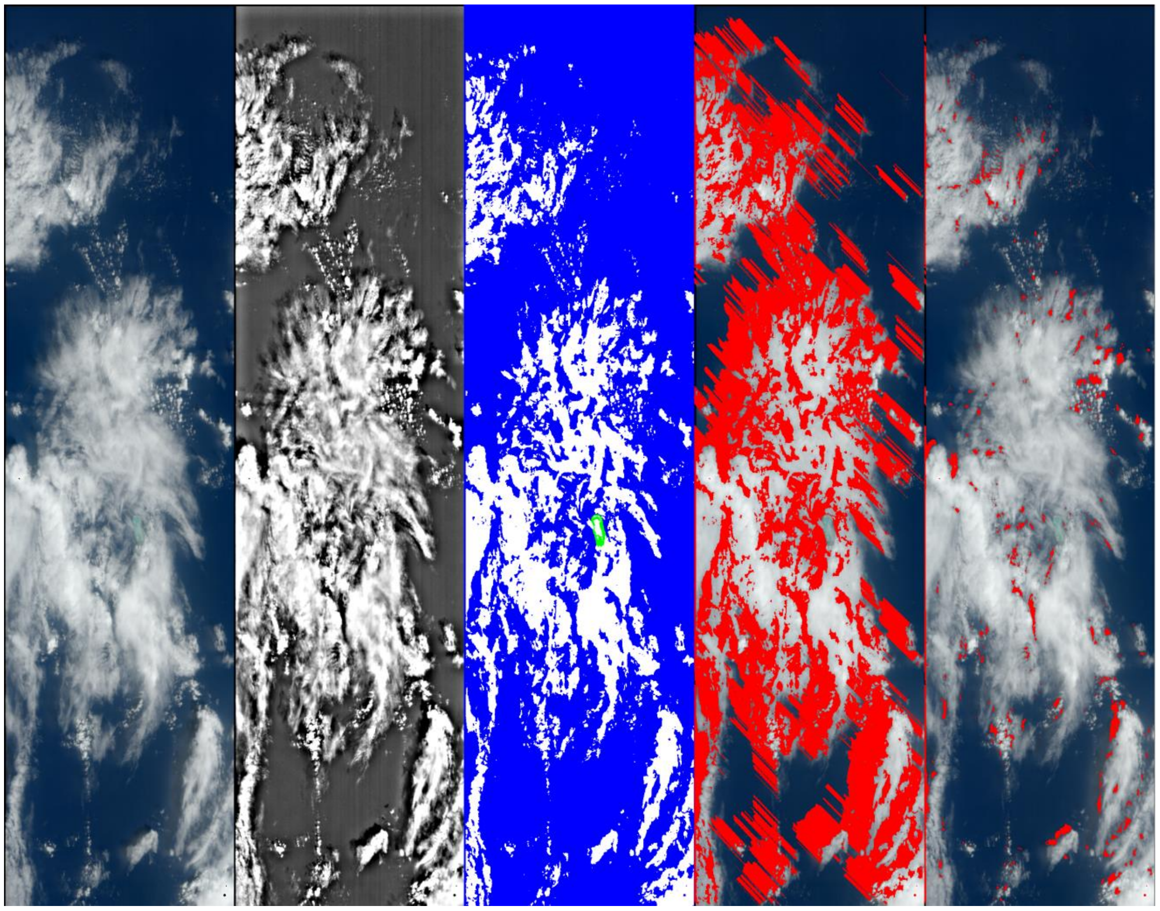

Figure. Wake Atoll 2014 GAL step progression.

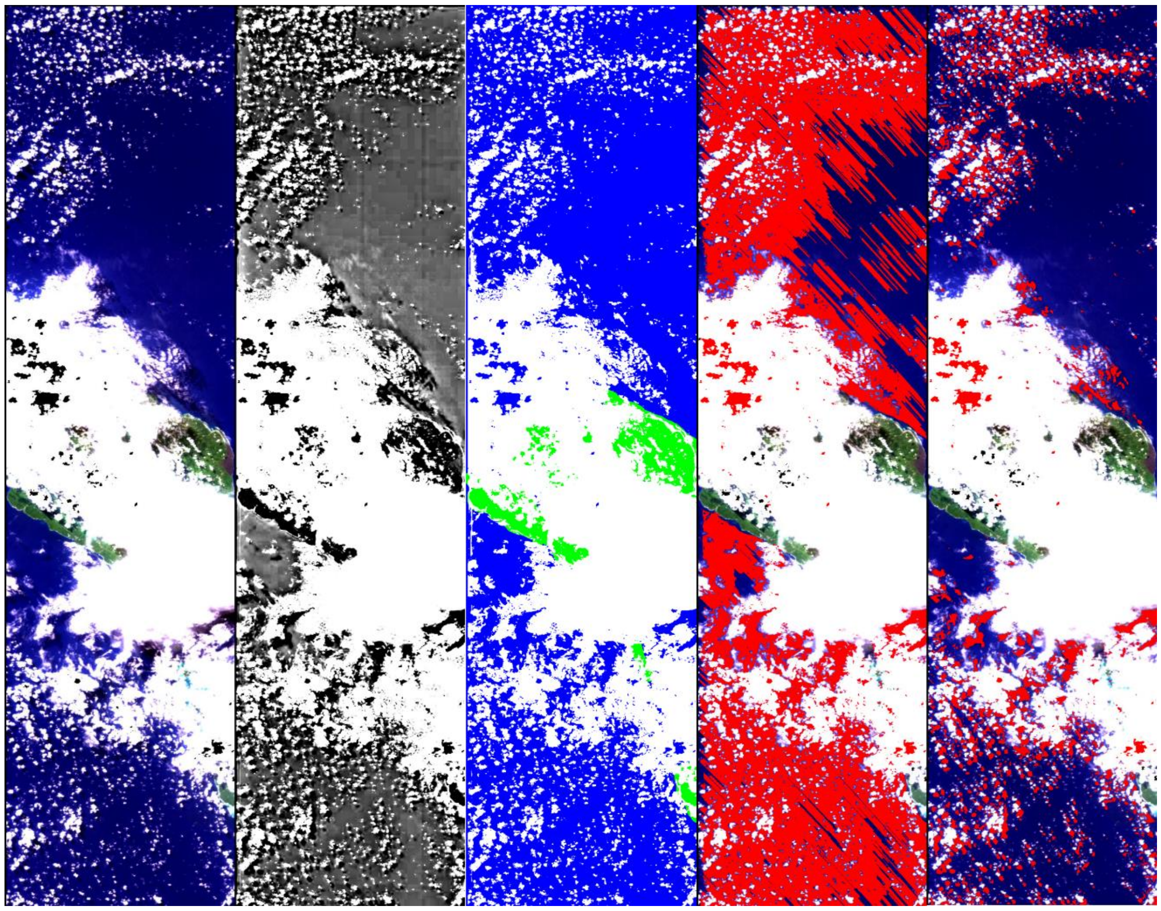

Figure. W. Samoa 2010 GAL step progression.

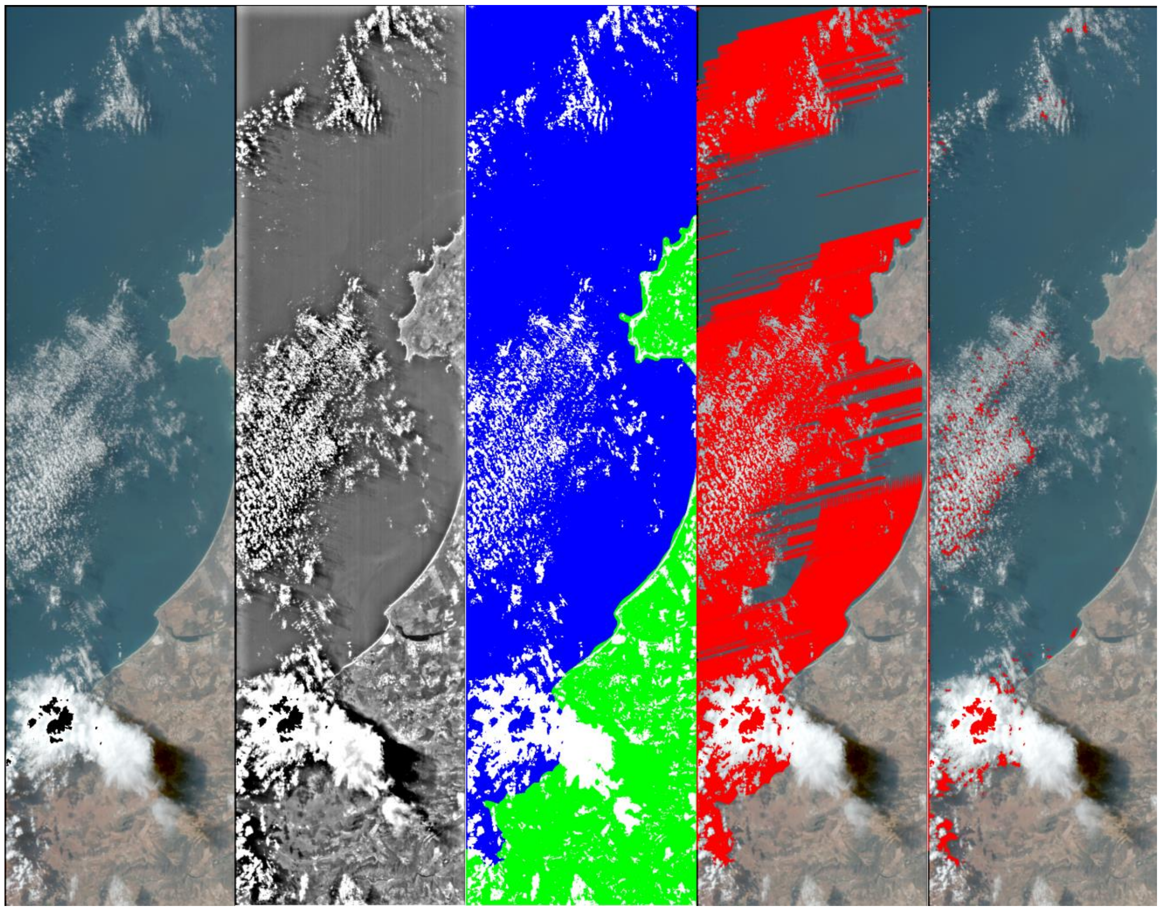

Figure. ZAF St. Helena Bay 2014 GAL step progression

REFERENCES

[1] R. Amin, R. Gould, W. Hou, R. Arnone and Z. Lee, “Optical Algorithm for Cloud Shadow Detection Over Water,” *IEEE Transactions on Geoscience and Remote Sensing,* vol. 51, no. 2, pp. 732-741, 2013.

[2] K. He, J. Sun and X. Tang, “Guided Image Filtering,” in *Computer Vision - ECCV 2010,* Heraklion, Crete, 2010.

[3] J. Nahomiak, “HICO - Hyperspectral Imager for the Coastal Ocean,” Oregon State University, 2009. [Online]. Available: http://hico.coas.oregonstate.edu/. [Accessed 2013].

[4] “International Space Station Fact Sheet,” NASA, [Online]. Available: http://www.nasa.gov/mission_pages/station/research/experiments/689.html. [Accessed 2013].

[5] “http://modis.gsfc.nasa.gov/about/,” [Online].

[6] NOAA, “http://noaasis.noaa.gov/NOAASIS/ml/avhrr.html,” [Online]. Available: http://noaasis.noaa.gov/NOAASIS/ml/avhrr.html.

[7] J. J. Simpson and J. R. Stitt, “A procedure for the detection and removal of cloud shadow from AVHRR data over land,” *IEEE Transactions of Geoscience & Remote Sensing,* vol. 36, no. 3, pp. 880-897, 1998.

[8] R. Amin, D. Lewis, R. W. Gould, W. Hou, A. Lawson, M. Ondrusek and R. Arnone, “Assessing the Application of Cloud-Shadow Atmospheric Correction Algorithm on HICO,” *IEEE Transactions on Geoscience and Remote Sensing,* vol. 52, no. 5, pp. 2646-2653, 2014.

[9] M. D. Lewis, R. W. Gould, R. A. Arnone, P. E. Lyon, P. Maritnolich, R. Vaughan, A. Lawson, T. Scardino, W. Hou, W. Snyder, R. Lucke, M. Corson, M. Montes and C. Davis, “The hyperspectral imager for the coastal ocean (HICO): sensor and data processing overview,” in *IEEE*, Biloxi, 2009.

[10] NRL, “U.S. Naval Research Laboratory (NRL),” [Online]. Available: http://www.nrl.navy.mil. [Accessed 2013].

[11] NRL, “United States Naval Research Laboratory - Bio-Optical/Physical Processes and Remote Sensing Section,” [Online]. Available: http://www7333.nrlssc.navy.mil. [Accessed 2013].

[12] M. R. Corson, D. R. Korwan, R. L. Lucke, W. A. Snyder and C. O. Davis, “The hyperspectral imager for the coastal ocean (HICO) on the international space station,” in *IEEE International Geoscience and Remote Sensing Symposium*, Boston, 2008.

[13] ESR, “Earth & Space Research,” [Online]. Available: http://www.esr.org/outreach/glossary/albedo.html. [Accessed 2013].

[14] P. M. Dare, “Shadow Analysis in High-Resolution Satellite Imagery o Urban Areas,” *Photogrammetric Engineering & Remote Sensing*, vol. 71, no. 2, pp. 169-177, 2005.

[15] K. V. Khlopenkov and A. P. Trishchenko, “SPARC: New Cloud, Snow and Cloud Shadow Detection Scheme for Historical 1-km AVHHR Data over Canada,” *Journal of Atmospheric and Oceanic Technology*, vol. 24, pp. 322-343, 2006.

[16] L. Jiang and M. Wang, “Identification of pixels with stay light and cloud shadow

contaminations in the satellite ocean color data processing," *Applied Optics*, vol. 52, no. 27, pp. 6757 - 6770, 2013.

[17] Google Inc., *Google Earth 7.1.2.2041*, 2013.

[18] NOAA, "NOAA Earth System Research Laboratory," [Online]. Available: http://www.esrl.noaa.gov/gmd/grad/solcalc/azelzen.gif. [Accessed 2012].

[19] J. J. Simpson and Z. Jin, "Cloud Shadow Detection Under Arbitrary Viewing and Illumination Conditions," *IEEE Transactions on Geoscience & Remote Sensing,* vol. 38, no. 2, pp. 972-976, 2000.

[20] Y. Luo, A. P. Trishchenko and K. V. Khlopenkov, "Developing clear-sky, cloud and cloud shadow mask for producing clear-sky composites at 250-meter spatial resolution for the seven MODIS land bands over Canada and North America," *Remote Sensing of Environment*, vol. 112, pp. 4167-4185, 2008.

[21] R. Zhang, D. Sun, S. Li and Y. Yu, "A stepwise cloud shadow detection approach combining geometry determination and SVM classification for MODIS data," *International Journal of Remote Sensing*, vol. 34, no. 1, pp. 211-226, 2013.

[22] R. Seyfarth, *USM Lecture Notes: Disjoint Set Union/Find,* Hattiesburg, Ms, 2006.

[23] The MathWorks, Inc., "Matlab Documentation," [Online]. Available: http://www.mathworks.com/help/index.html.

[24] B.-C. Gao, M. J. Montes, Z. Ahmad and C. O. Davis, "Atmospheric correction algorithm for hyperspectral remote sensing of ocean color from space," *Applied Optics,* vol. 39, no. 6, pp. 887-896, 2000.

[25] X. He, D. Pan, Y. Bai, Q. Zhu and F. Gong, "Evaluation fo the aerosol models for

SeaWiFS and MODIS by AERONET data over open oceans," *Applied Optics,* vol. 50, no. 22, pp. 4353-4364, 2011.

[26] P. N. Reinersman, K. L. Carder and F. I. Chen, "Satellite-sensor calibration verification with the cloud-shadow method," *Applied Optics,* vol. 37, no. 24, pp. 5541-5549, 1998.

[27] V. Shettigara and G. Sumerling, "Height Determination of Extended Objects Using Shadows in SPOT Images," *Photgrammetric Engineering & Remote Sensing*, vol. 64, no. 1, pp. 35-44, 1998.