RAPID RECONNAISSANCE AND RESPONSE (R^3)

THERMAL ALGORITHMS

by

Nathalie Vedie

Technical documentation to support the detailed design of:

R^3 Mission
Center for Space Systems
Georgia Institute of Technology

Spring 2010

Lead Designer ____Date __________

Project Systems Engineer ________________________________Date __________

Project Manager _______________________________________Date__________

REVISIONS

Revision	Description	Date	Approval
___	Initial Release	4/29/08	
___	PDR Update	7/12/09	
___	Detailed Design Update	12/17/09	
___	CDR Update	2/12/10	
	End of Spring semester	4/28/10	

GEORGIA TECH CENTER FOR SPACE SYSTEMS

ABSTRACT

Thermal Algorithms Subsystem Technical Design Document

by Nathalie Vedie

Principle Investigator: Professor David Spencer
Georgia Tech

An overview of the thermal algorithms utilized in the R^3 mission is presented. The R^3 satellite will have thermal algorithms to process images taken by the thermal imager. The algorithms will calculate the area of features based on whether the feature's size matches the intended features area and on whether the temperature of the feature falls within a certain range. The center of each matching feature can be calculated and validated onboard the satellite. The algorithms will also be able to detect edges to various ends.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The authors wish to express sincere appreciation to Professor David Spencer (Principle Investigator) and Annalisa Bracco (Associate Professor in Oceanography at the EAS department of the Georgia Institute of Technology) for their assistance in the preparation of this design and documentation. Special thanks also to the entire R^3 design team, whose tireless diligence and cooperative spirit was vital during the early programming phase of this undertaking.

NOMENCLATURE

Blobs – contiguous features found on the output of the blobber algorithm; used to identify features that might pass identification criteria (intensity and area)

Blobber Algorithm – algorithm used to find features of interest on the R3 mission

Edge Detection Algorithm – algorithm used to detect edges via gradients, supplement the blobber algorithm

Kernel – a matrix that is used to operate on a pixel and its surrounding pixels to determine various characteristics of the pixel in question

Laplace Operator – the basis for the edge detection algorithm that uses a second derivative approach to detect edges

Mask – see Kernel

Pass – a flight over a feature of interest by the R3 satellite

Pass-failure – when a pass fails to either image the feature due to some obstruction or fails to downlink to the ground station

Prewitt Operator – the basis for one of the two edge detection algorithms that utilize the first derivative approach to detect edges; see Sobel Operator

Script – a tool used to run algorithms multiple automatically in order to analyze the algorithm itself instead of worrying about code and to generate results automatically

Sobel Operator - the basis for one of the two edge detection algorithms that utilize the first derivative approach to detect edges; see Prewitt Operator

Thermal Imaging – the science of analyzing thermal images that are taken by a thermal imager

Thermal Imager – the tool used to generate, or take, thermal images

Thermal Variance – the difference in temperature that manifests as difference in pixel intensity in thermal image.

INTRODUCTION

The R^3 satellite project began with eight students attending Georgia Institute of Technology in January 2009. The goal of the R^3 project is to win a secondary payload launch opportunity through the Air Force Research Laboratory's sixth University Nanosat Program competition (UNP-6). The mission objectives are to demonstrate same-pass tasking and innovative on-board processing algorithms by detecting a thermal feature of known signature, calculating its coordinates, and downlinking them to the Georgia Tech Ground Station in the same pass; and to correlate the radiation environment in space with its effect on a non-radiation-hardened uncooled microbolometer. As a secondary payload, the satellite must be robust to successfully complete its mission in any orbit from 500-1000 km and at any inclination. It must also meet all UNP-6 program constraints, including staying within a size and mass envelope of 50cm x 50cm x 60cm and 50kg.

The R^3 satellite shall have a thermal algorithms subsystem in order to process captured images of interest and downlink information about these images to the ground station. The detail designed document will understand the logic behind the algorithms and determine what algorithm would yield the desired results in the most efficient manner.

CHAPTER 1: REQUIREMENTS FLOW-DOWN

The requirements for the Algorithms flow down directly from the mission statements of R^3. The requirements flow down presented in Table I detail the mission statement, the Mission Objectives, the Mission Success Criteria and the Subsystem Requirements.

Table I Requirements Flow-Down

Index	Requirement	Source
	Mission Objectives	
M0-3	R3 shall acquire thermal images from low earth orbit, and utilize onboard image processing algorithms to detect and geolocate thermal features having specified signatures.	
	Mission Success Criteria	
MSC-4	The R3 mission shall utilize onboard image processing algorithms to detect features having specified signatures.	MO-3
MSC-5	The R3 mission shall geolocate the thermal features using onboard processes.	MO-3
	Thermal algorithms Requirements	
ALG-1	On-board image processing algorithms shall be capable of identifying thermal features within images	MSC-4
ALG-1.1	The image processing algorithm to be applied to a particular thermal image shall be identified via ground command.	ALG-1
ALG-1.1.1	The "blobber" algorithm shall be capable of identifying thermal features having a specified range of intensity, covering a specified range of contiguous pixel area.	ALG-1.1
ALG-1.1.2	The "edge detection" algorithm shall be capable of detecting areas within thermal images exhibiting thermal gradients within specified ranges.	ALG-1.1
ALG-1.2	The image processing algorithm shall downlink an error message if they could not be process the image correctly	
ALG-2	The image processing algorithms shall be capable of calculating the centroid pixel of identified thermal features.	MSC-5

CHAPTER 2: SUBSYSTEM ARCHITECTURE

TRADE STUDIES AND DESIGN DRIVERS

The thermal algorithms used on the R3 satellite can detect thermal variations in order to identify shapes and edges of objects of interest. The blobber algorithm can be used to identify the center of a feature. The edge detection algorithms can be used to various ends, such as detecting eddies to detecting changes in shape of a particular feature over multiple passes. We list below two possible design drivers: the blobber algorithm and the edge detection algorithm.

Blobber Algorithm

The blobber algorithm was created for the specific use of screening a grayscale image and finding certain features (blobs) based on intensity thresholds and area limitations. After all blobs that fit the intensity and area criteria are found, their centers can be calculated and downlinked to the ground station, or we can obtain the coordinates of each pixel belonging to a blob.

Edge Detection Algorithm

The edge detection algorithms are multifarious in their use. The algorithms can be used to detect eddies and currents in the water, such as the Gulf Stream, in addition to the blobber algorithm; or to detect river outflows. The data received from these algorithms can be used to either confirm that the blob we found is indeed the feature we were looking for or even track ocean currents and detect any shifts. We can also track changes along the years by comparing the results given by the edge detection algorithm applied to two different images taken at two different dates.

Trade Study: Edge Detection Algorithm

There are two approaches to edge detection: the first derivative approach and the second derivative approach. The first derivative method finds the gradient and looks for maxima in the gradient to detect the edges. Two examples employ the first derivative approach: the Sobel Operator and the Prewitt Operator. The second derivative method, however, finds the rate of change of the gradient and locates where the rate crosses zero, thereby detecting an edge. The algorithm utilizing the second derivative approach is the Laplace Operator.

Sobel Operator and Prewitt Operator

The Sobel operator and the Prewitt operator use the first derivative method to detect edges in the grayscale images. They both utilize two kernels (masks/matrices) to find the changes in the horizontal direction and the vertical direction. It then uses those values to calculate the direction of the change intensity in the original grayscale image and creates a new image representing the edges. The difference between the two operators is slight; the Sobel Operator is a bit brighter than the Prewitt operator, due to the different sets of kernels employed by each operator.

Laplace Operator

The Laplace operator (or Laplacian operator) calculates the second derivative of the given image in order to find the rate of change of the gradient. This allows the operator to determine whether the pixel is likely to be part of an edge or not. This method is often much faster than the Sobel and Prewitt operators mainly because the Laplace operator only uses one kernel to determine the information it needs to create the final image, rather than calculating the information after applying the two kernels to the image.

SUBSYSTEM DESCRIPTION AND COMPONENTS

Since this subsystem is dealing with algorithms, the results expected and their explanation is given in the Algorithms section.

CHAPTER 3: SUBSYSTEM ENGINEERING

ALGORITHMS

The blobber algorithm locates the blobs desired in a series of eight steps. A graphical representation is presented after the discussion.

1. Initialize all matrices to zero. All of these matrices are the same size as the input image.

2. Screen the original image to create an intensity screen. In this step, all pixels that meet the intensity threshold are given a value of 255 whilst all others are assigned 0. The result is a splattering of white on black.

3. Now it is time to build blobs. Looking at each individual pixel on the intensity screen, if the pixel has at least one neighbor (another pixel with the value of 255 is in one of the 8 squares surrounding the pixel) the pixel is kept as belonging to a blob. Otherwise the pixel is discarded and assigned the value of 0.

4. Now we must number the blobs in order to operate on them. Simply start labeling a pixel and if a pixel in the surroundings is already labeled with a number less than the current label, but different from zero, apply the lower number label.

5. We repeat step 4 in order to modify the pixels that do not have the correct label corresponding to their actual blob number yet.

6. Here, the area for each indidual blob is calculated.

7. Now it is time to eliminate the blobs that don't meet the area criteria specified.

8. Finally, out of the blobs that have successfully passed area screening, we can calculate their centers as possible matches for the features in question.

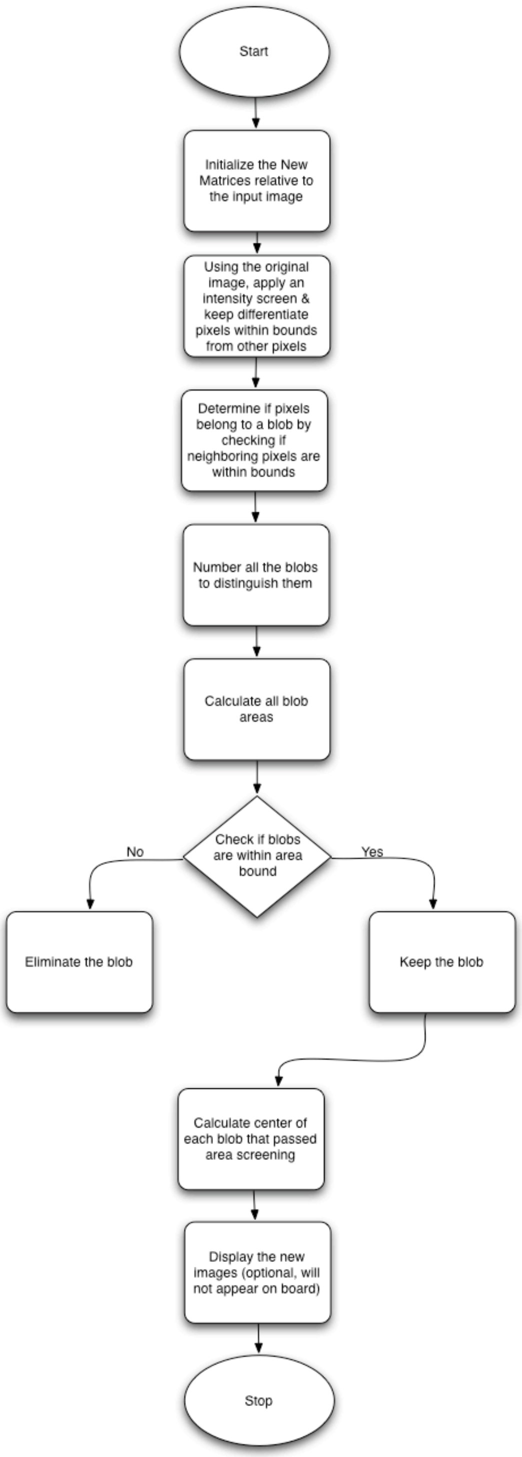

Start
Initialize the New Matrices relative to the input image
Using the original image, apply an intensity screen & keep differentiate pixels within bounds from other pixels
Determine if pixels belong to a blob by checking if neighboring pixels are within bounds
Number all the blobs to distinguish them
Calculate all blob areas
Check if blobs are within area bound
No
Yes
Eliminate the blob
Keep the blob
Calculate center of each blob that passed area screening
Display the new images (optional, will not appear on board)
Stop

Figure 1: Blobber Algorithm Flowchart

The two approaches in edge detection (first and second derivative) don't differ too much. The main difference is the size and amount of masks utilized.

FIRST DERIVATIVE APPROACH: SOBEL AND PREWITT OPERATORS

The steps for the first derivative are quite simple:

1. Initialize all matrices to zero. The matrices are again the same size as the original image.

2. Apply the horizontal and vertical masks to each pixel, which are then used to create horizontal and vertical images of the gradient. The mask sets for both the Prewitt and the Sobel operators are shown at the end of this listing.

3. Use the horizontal and vertical images to calculate the final edge detection image. Keep only the output pixels whose values are between thresholds. With this option, we can get rid of strong intensity jumps between neighboring pixels, and keep only smaller differences for example.

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Figure 2: Sobel Mask for vertical direction

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Figure 3: Sobel Mask for horizontal direction

$$\mathbf{G_y} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$$

Figure 4: Prewitt Mask for vertical direction

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}$$

Figure 5: Prewitt Mask for horizontal direction

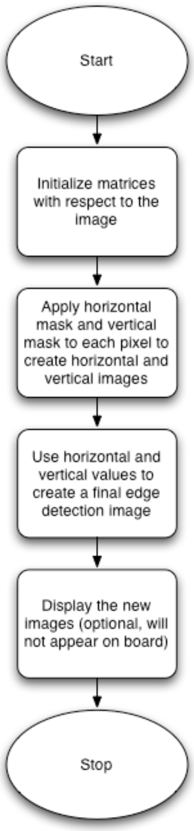

Start
Initialize matrices with respect to the image
Apply horizontal mask and vertical mask to each pixel to create horizontal and vertical images
Use horizontal and vertical values to create a final edge detection image
Display the new images (optional, will not appear on board)
Stop

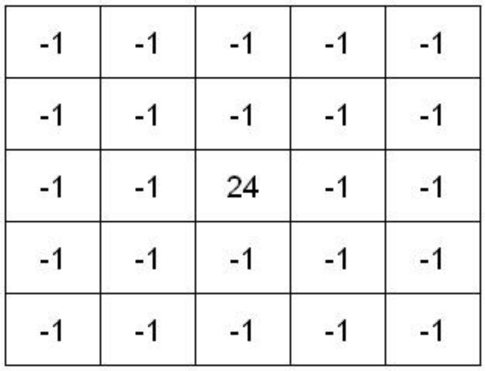
Figure 6: First derivative approach flowchart

SECOND DERIVATIVE APPROACH: LAPLACE OPERATOR

The methodology for the second derivative are simple and efficient, mainly because the Laplace operator only has one mask to apply and finds the value for the pixel in the edge detection image directly, thereby reducing calculations.

1. Initialize the edge detection matrix. There is only one matrix necessary for this algorithm, as the values are computed and stored directly, rather than through any intermediary matrices.

2. Apply Laplacian mask to all pixels and store results to the final image. One again, the output value is kept only if it belongs to a certain range of intensity jumps. Below is a flowchart for the second derivative approach and the Laplace Operator.



-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	24	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

Figure 7: Laplace Operator's Mask

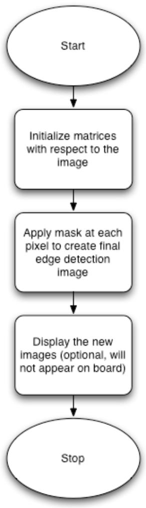

Start
Initialize matrices with respect to the image
Apply mask at each pixel to create final edge detection image
Display the new images (optional, will not appear on board)
Stop

Figure 8: Second derivative approach flowchart

RESULTS

A lot of examples have been realized to check the results of the blobber and edge detection algorithm.

The edge detection algorithms are meant to detect sharp changes in intensity indicative of an edge. The main difference in the algorithms is the intensity of edges. The second derivative method yields larger thermal jumps, as shown below when all three algorithms are applied to an image of the Charleston coast in South Carolina.

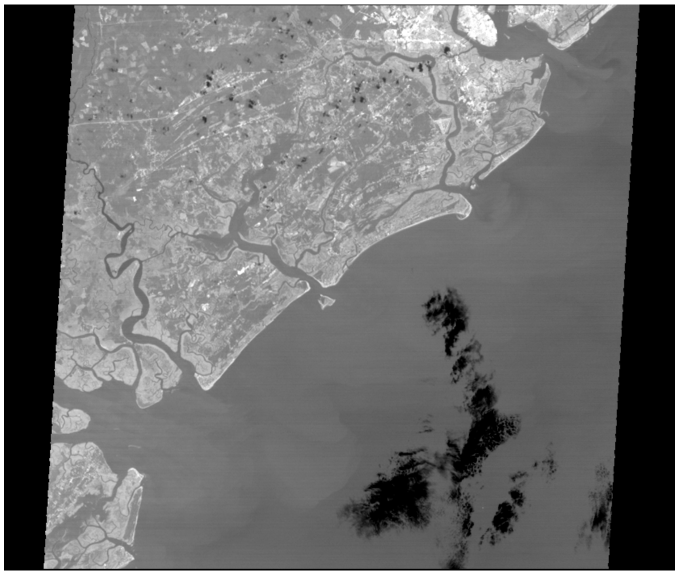


Figure 9: Charleston, SC original image

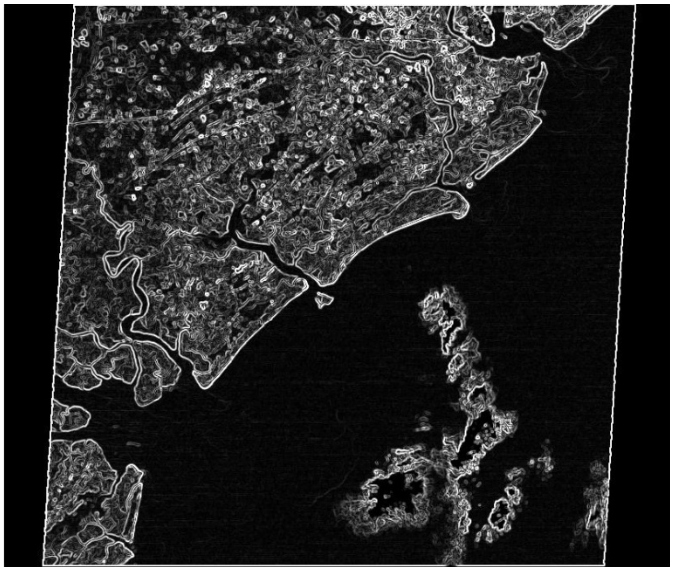


Figure 10: Sobel operator

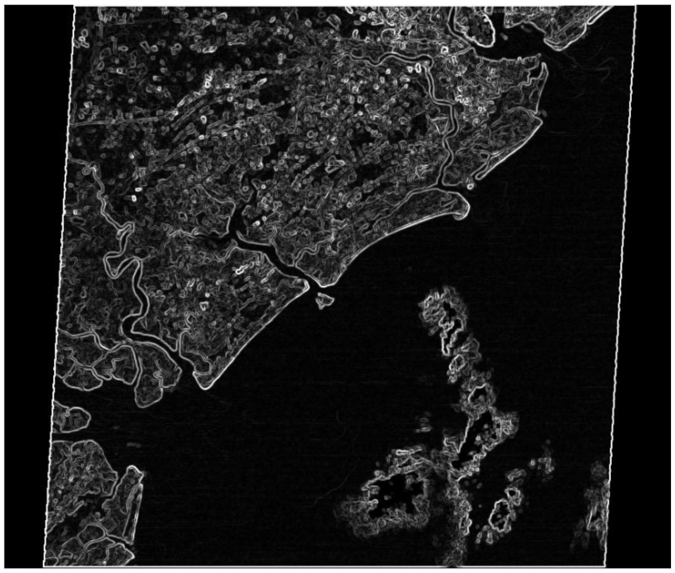


Figure 11: Prewitt operator

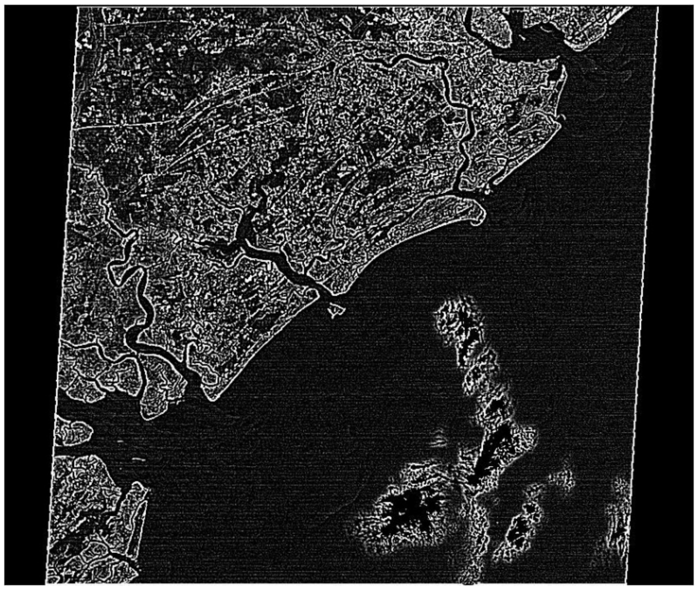


Figure 12: Laplace operator

The blobber algorithm gives us the possibility to detect contiguous features whose area and pixels' intensity fall within some ranges. This capability has been applied to detecting the Gulf Stream and its limits close to the coast. Below are some examples of tests run on images representing the Gulf Stream. This algorithm can also be applied to other gyres all around the world to know their boundary in real time.

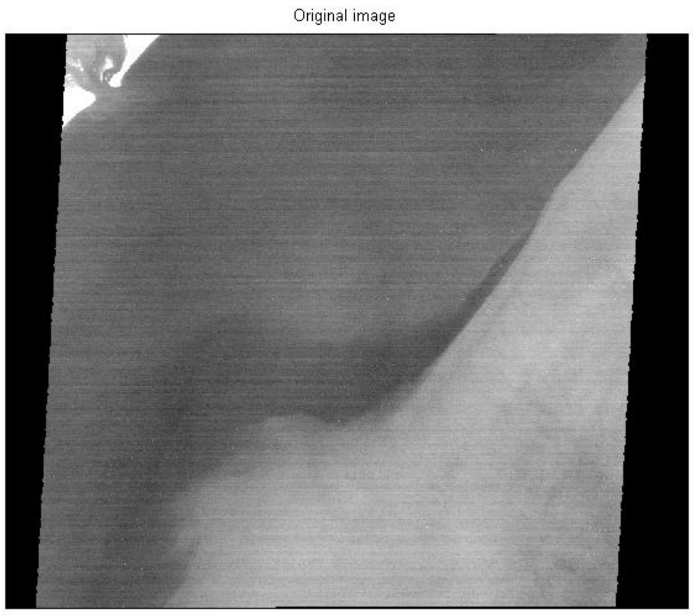


Figure 13: Ocracoke, NC original image

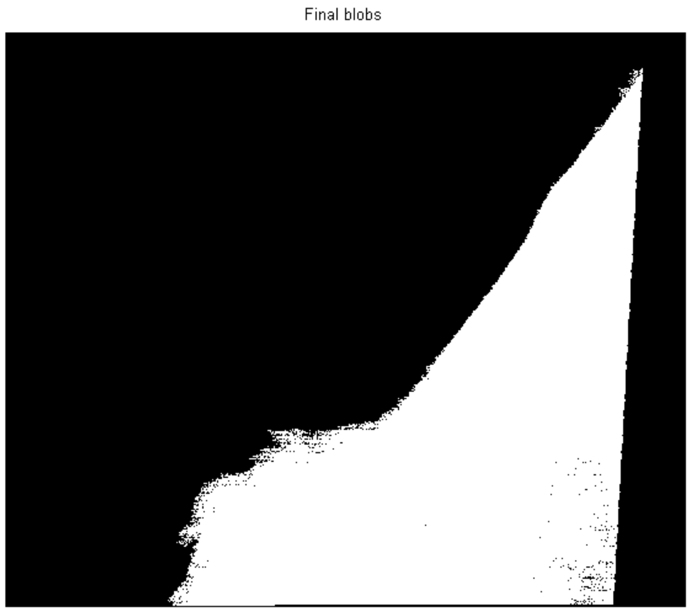


Figure 14: Blobber algorithm results

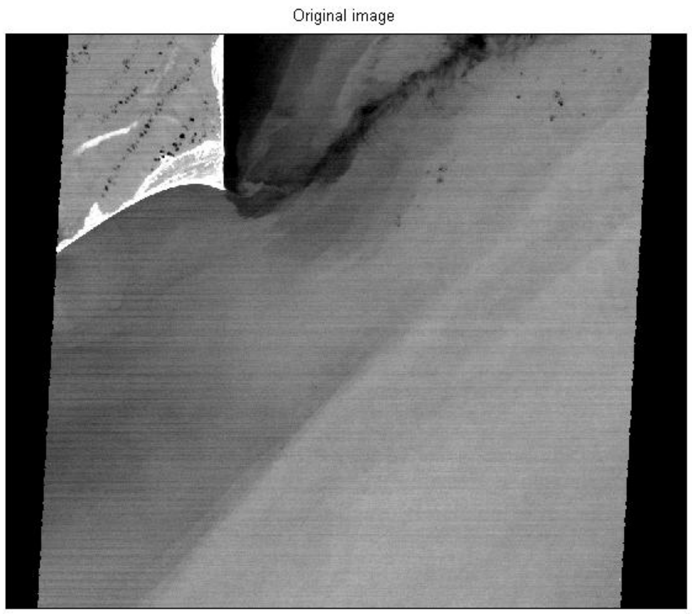


Figure 15: Cape Hatteras, NC original image

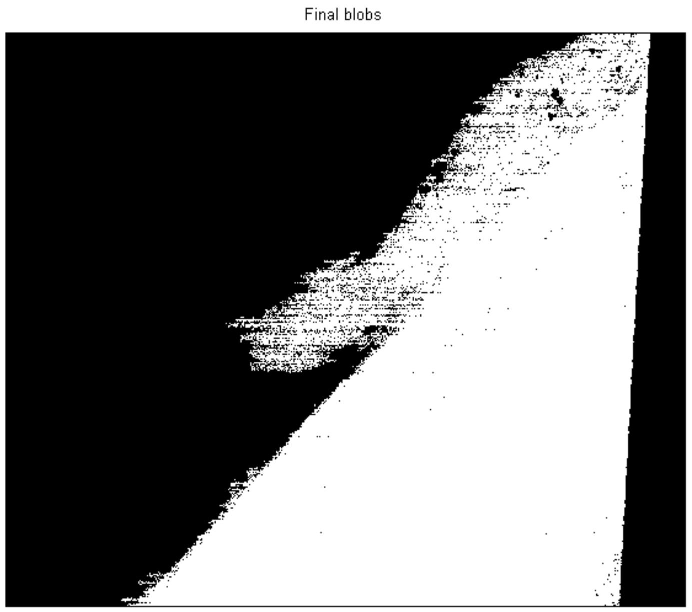


Figure 16: Blobber algorithm results

After running the blobber algorithm, we can also apply the edge detection algorithm and obtain the exact limit:

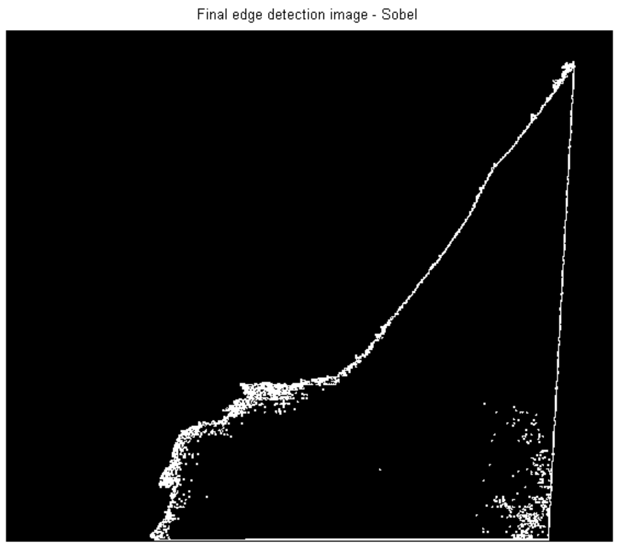


Figure 17: Blobber algorithm + Sobel edge detection results

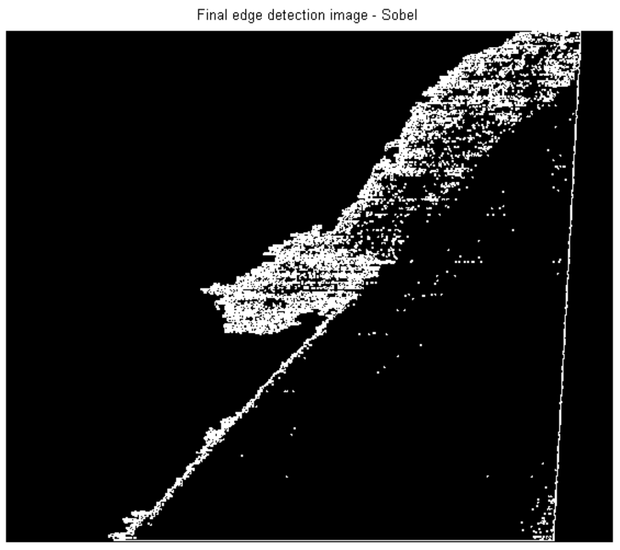


Figure 18: Blobber algorithm + Sobel edge detection results

CHAPTER 4: OPERABILITY

COMMANDS

The blobber algorithm needs a range of intensities and a range of areas in order to function correctly. The intensity range is related to the temperature range of the feature we want to detect. Once we have the average and the standard deviation of the picture (in Kelvin, and ignoring the pixels where the temperature is 0 K), we can convert this minimum and maximum temperature to the corresponding intensities (with values between 0 and 255 (8 bits)), using the function:

□−□□□□□□□□□□□□□□□□□+*127*

We run tests to determine the minimum and maximum temperature. The algorithm will select feature whose minimum temperature is equal to the average temperature T_{av} in the image, and the maximum is $T_{av} + 3\sigma$. Since the images we select will mainly have ocean, we want to select the warmer part of the range of temperature displayed in the image to select the Gulf Stream. That is how our selection of the range of temperatures was chosen. (See results in the algorithm section)

The algorithm also needs a set of area ranges to screen the blobs and find a feature that meets the desired criteria. The calculation of the possible range of areas for the blob is based on the percentage of land and clouds present in the picture, the number of pixels that don't represent ground parcels (i.e. the two black bands that appear on both sides of the image), and a given precision. In the examples displayed in the algorithms section, the parameters that were chosen were

- Number of pixels: 90,230
- Precision: 5%

The calculation of the area range is then simple. The minimum size of the feature to be called a blob must be the maximum number of pixels that the land and the clouds can occupy (to make sure we select features in the water). This is calculated by multiplying the estimation of percentage of the image occupied by land and clouds, adding the precision percentage and then multiplying it by the total number of pixels. This gives us the minimum number of pixels that the blob must contain. The maximum number is calculated by subtracting to the total number of pixels in the image the number of pixels that don't represent any parcel on the ground (the side bands) and the minimum number of pixels occupied by the land and the clouds (calculated by multiplying the estimation of percentage of the image occupied by land and clouds, substracting the precision percentage and then multiplying it by the total number of pixels). This explains why we need to have images that contain more than half the image "covered" in water. The results are showed in the algorithms section.

Note that we have a minimum range in area that is bigger than the land coverage to make sure that the blobs selected represent data from the ocean. The temperature on the ground and in the Gulf Stream are pretty similar, and we need to make sure that we don't select the land by requiring a blob bigger than the land area on the image.

The edge detection algorithms don't really need any parameters, except for the range of intensity jumps wanted. They simply need to read in an image and will return the edges detected. When used in conjunction with the blobber algorithm, shapes and features can be distinguished.

TELEMETRY

The algorithms need the land percentage estimated from the ground thanks to a tool that will be developed if the Georgia Tech wins the competition (cf. section 8). This will be an input from the ground, whereas the cloud cover percentage, the standard deviation and average temperature of the image have to be calculated onboard. These last three are also inputs for the algorithms. The cloud cover percentage is not a required parameter though. This is an estimation of how much cloud will obstruct real data from the ground, and help us calculate the area range for the blobs. However, this time, pixels showing clouds cannot be mistaken for water, since the clouds are much colder. It is important though to be able to discard an image if the cloud cover is too important, and to not run the algorithms, since the results could be completely off. I recommend discarding the image if its cloud cover is greater than 40%.

The outputs that need to be send to the ground consist of the coordinates of the blob or its center of brightness, depending on the type of feature we want to detect. If the algorithms could not run properly an error message must be sent to the ground in order to know that some data might not be accurate. The algorithms can still be processed on the ground with the downlinked image.

CHAPTER 5: INTEGRATION AND TESTING

REQUIREMENT VERIFICATION

On-board image processing algorithms shall be capable of identifying thermal features within images

We run the algorithms several times on various images to check that the Gulf Stream was the feature detected, as expected.

To have the description of this test, please refer to the Algorithms section explaining how it works.

The image processing algorithm to be applied to a particular thermal image shall be identified via ground command.

The mission planning process selects the time when images will be taken. This helps up figure out parameters such as the land cover. The operator will select which features will be detected and chooses which algorithm he will apply. It can be the blobber, the edge detection, or a combination of both.

The "blobber" algorithm shall be capable of identifying thermal features having a specified range of intensity, covering a specified range of contiguous pixel area.

We have run the algorithms on several images representing the Gulf Stream, in order to check that the algorithm was working properly.

The "edge detection" algorithm shall be capable of detecting areas within thermal images exhibiting thermal gradients within specified ranges.

We have run the algorithms on several images representing different thermal gradients, in order to check that the algorithm was working properly.

The image processing algorithm shall downlink an error message if they could not be process the image correctly

We have run the algorithms on several images in order to check that the algorithm was diplaying an error message if it was not working properly (like not finding a blob for example).

The image processing algorithms shall be capable of calculating the centroid pixel of identified thermal features.

Calculating the centroid of a blob is an option that can be selected once the blobs have been found. Some images have been tested to check if that this option was giving us the results expected. Below are examples of blobs and their centroid

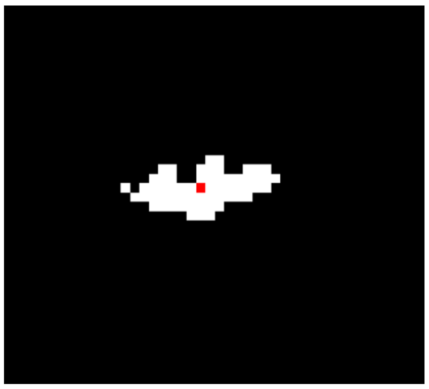


Figure 19: Stone Mountain, GA and its centroid (ASTER image, 90m/pixel resolution)

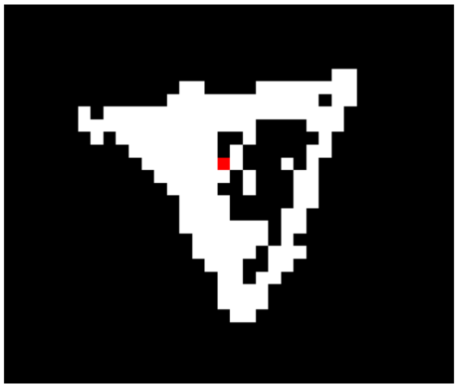


Figure 20: Charleston, SC airport and its centroid (ASTER image, 90m/pixel resolution)

INTRAGRATED TESTS

Blobber algorithm test

The blobber algorithm needs a lot of inputs from another subsystem, such as Instruments, to find the correct blobs. This means that an end-to-end test needs to be realized when all the code has been put together.

CHAPTER 6: DEVELOPMENT SCHEDULE

DETAILED DESIGN

By may 7th, 2010, the algorithms should be finished and documented properly.

INTEGRATION & TEST

Every test has been realized so far, it is now the Software team that must check the results obtained when coding from Matlab to C.

CHAPTER 7: FLIGHT RULES

CLOUD COVER

The algorithms should not been run if the cloud cover in the image exceeds 40%.

ERROR MESSAGE

The algorithms should downlink an error message to the ground if the image was not processed correctly.

CHAPTER 8: RISK AREAS AND OPEN ITEMS

RISK AREAS

The image could not be acquired properly, and the thermal gradients could be too small to detect the Gulf Stream.

OPEN ITEMS

The second mission of the algorithms is to detect river outflows, and this is one of the open items left so far. The results so far have not been conclusive, but some work is still possible.

BIBLIOGRAPHY

"Edge detection." Wikipedia, the free encyclopedia. 07 Dec. 2009 <http://en.wikipedia.org/wiki/Edge_detection>.

"Edge Detection Tutorial." Drexel University - Unix Web Service. 07 Dec. 2009 <http://www.pages.drexel.edu/~weg22/edge.html>.

"Laplace operator." Wikipedia, the free encyclopedia. 07 Dec. 2009 <http://en.wikipedia.org/wiki/Laplacian>.

"Prewitt." Wikipedia, the free encyclopedia. 07 Dec. 2009 <http://en.wikipedia.org/wiki/Prewitt>.

"Sobel operator." Wikipedia, the free encyclopedia. 07 Dec. 2009 <http://en.wikipedia.org/wiki/Sobel_operator>.