Efficient and Autonomous Processing and Classification of Images on Small Spacecraft

Antony Gillette, Chris Wilson, Dr. Alan D. George
Department of Electrical and Computer Engineering
NSF CHREC Center, University of Pittsburgh
Pittsburgh, Pennsylvania
antony.gillette@chrec.org

***Abstract*— Small satellites and CubeSats are becoming an indispensable platform in space-industry development, however, these systems are severely resource-limited. Depending upon mission requirements and available communication bandwidth, it can take hours to days to downlink an image from a spacecraft to the ground. Improvements in sensors tend to generate increasingly larger data products. Since small spacecraft have limited storage space, it is crucial to filter and delete images that do not meet minimum science criteria. Depending on the mission, criteria may vary. Images can be prioritized based on having features such as high land percentage or a specific land color. Certain images are rarely useful, such as pitch-black images and cloud-filled images, and can be readily deleted. This research describes an autonomous image-classification framework to efficiently use downlink bandwidth by prioritizing image products with high science value for download while deleting others, as well as a training framework for classifier calibration.**

I. Introduction

Space is a challenging environment for engineers and designers to develop systems for due to severe constraints for size, weight, power, and cost. Additionally, designers must consider the hazardous nature of radiation effects on electronic components which can prematurely end mission operations [1].

Recent trends in the space industry show that many organizations are turning to small satellites and CubeSats over traditional large satellites [2]. CubeSats are small spacecraft shaped in the form of one or more $(10cm)^3$ cubes typically weighing less than 1-kg [3]. This development platform has risen in popularity lately to help designers more affordably meet next-generation mission requirements as a low-cost entry point for space-science missions and research [4].

Depending upon mission requirements, CubeSat electronics, available power, and communication bandwidth, it can take hours to days to downlink an image from the system. Additionally, the spacecraft platform may need to capture and store a substantial number of images for analysis. Along with improved sensor developments generating increasingly larger data products, this situation can lead to raw sensor images reaching cumbersome sizes which can quickly fill onboard storage. Small spacecraft have limited storage space, therefore filtering and deleting images that do not meet minimum science criteria can make efficient use of this space.

An effective way to correctly classify images is to use supervised classification, a well-known process in computer vision and image processing. With supervised classification, the user selects pixel values or patterns in advance that help distinguish key features or qualities. Identifying desirable patterns or traits in the imagery is called "training" the system, allowing the system to identify other images with similar characteristics. If the training is effective, classification will be more accurate, and the system will filter images based on the original training set [5].

Depending on the mission, science criteria may vary. Images can be prioritized based on having features such as high land percentage or a specific land color. When a CubeSat takes many images, unless there is an intelligent scheduler or advanced planner, it is likely the spacecraft will take images that have no scientific value. Certain types of images are rarely useful, such as pitch-black images and cloud-filled images, and may be deleted to reserve onboard storage, maximizing the possible number of quality stored images.

Classification can predict priority and determine the relevance of an image to a particular mission. However, for classifiers to be accurate, the system must undergo training. Without an efficient training procedure, the training process can be a time-consuming process of iteratively modifying settings. Different missions may have different cameras and lenses that will affect classifier settings. The same set of training data will not apply to every mission so training should be revisited for each mission. Images can fit into multiple classification categories which can make manually maintaining the categories burdensome.

Lastly, CubeSat image processing algorithms should not require too much power or memory overhead in order to increase their viability for the system. Algorithms should be simplified to reduce resource requirements and increase processing speed, especially if performing additional calculations does not significantly improve the accuracy of the classifier.

In this paper, we present an autonomous image-classification framework to efficiently use downlink bandwidth, by prioritizing image products with high science value for download while deleting others. Additionally, we describe a GUI as part of a training framework to quickly perform classifier calibration, which allows this framework and corresponding set of applications to be easily reused for other missions.

978-1-5386-3200-0/17/$31.00 ©2017 IEEE

II. BACKGROUND

This section provides a basic overview on general space technology targeted by this research. Additionally, this section describes the key mission that motivated primary challenges and considerations for the approach.

A. CHREC Space Processor v1 (CSPv1)

The CSPv1 is a 1U CubeSat form-factor space computer developed by researchers at the National Science Foundation (NSF) Center for High-Performance Reconfigurable Computing (CHREC) as a flexible platform to meet a variety of mission needs. The CSPv1 is a high-performance and radiation-tolerant Single Board Computer (SBC) that features a hybrid-architecture design suitable for the hazardous environment of space. CSPv1 uses a combination of both commercial and radiation-hardened components as well as fault-tolerant computing concepts to mitigate the effects of space radiation while maintaining low SWaP-C (size, weight, power, cost), and it achieves high performance through a mix of fixed and reconfigurable logic processing [6]. The commercial variant of the design is featured in Fig. 1a.

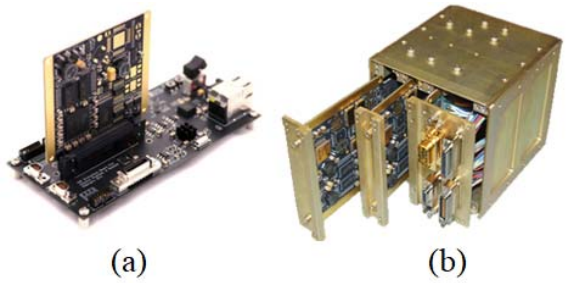

Fig. 1. CSPv1 plugged into Evaluation Board (left) and two CSPv1s in CubeSat form factor (right)

B. STP-H5/CSP

In order to rapidly develop space-system technologies for operational use, the Department of Defense established the Space Test Program (STP) as a means of testing systems in space preceding full-mission operations. STP provides spaceflight opportunities to the International Space Station (ISS) through its Houston office [7]. The STP-H5/CSP experiment was included as a sub-experiment to the ISS SpaceCube Experiment Mini (STP-H5/ISEM), as part of a collaboration with NASA Goddard.

The first mission for the CSPv1 was STP-H5/CSP, which was launched with several other experiments as part of STP-H5 in February 2017 via SpaceX's CRS-10. This experiment features twin CSPv1 processing cards, a power/interface card, and a backplane, all in a 1U chassis. The mission hardware is featured in Fig. 1b. STP-H5/CSP includes a high-resolution visual-spectrum imager which faces towards Earth to capture images for processing and Earth Observation. The STP-H5 payload was mounted on the ISS, and now STP-H5/CSP is processing, compressing, and downlinking terrestrial-scene images using the applications described in this paper. Examples of image products taken by STP-H5/CSP are featured in Fig. 2.

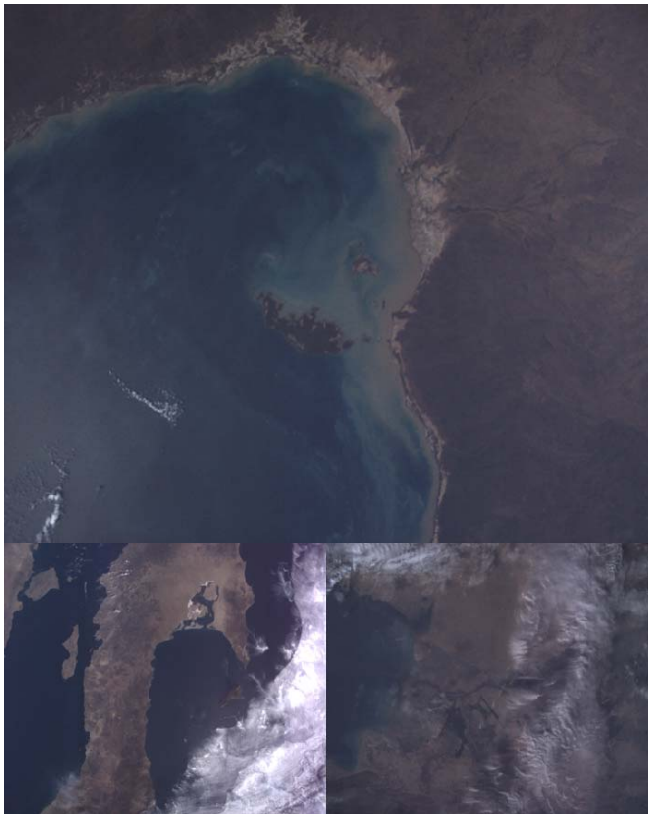

Fig. 2. Example images taken by STP-H5/CSP

C. STP-H4/ISE2.0

STP-H5/CSP mission ideas and design were influenced by previous experiments from close collaborators at NASA Goddard's SpaceCube Team. The primary mission example was the ISS SpaceCube Experiment 2.0 (ISE2.0). The purpose of STP-H4/ISE2.0 was to demonstrate flight operation of NASA Goddard's SpaceCube 2.0 flight computer. The experiment included a power unit, SpaceCube v2.0 instrumentation to detect and measure terrestrial gamma-ray flashes from lightning, and a set of Earth-viewing high-resolution cameras. The ISE 2.0 experiment was the follow-up experiment to the SpaceCube team's previous experiment on MISSE-7 (Materials International Space Station Experiment-7) [9]. Fig. 3 shows example images provided by STP-H4/ISE2.0. Fig. 4 displays the location of STP-H4/ISE2.0 in relation to the location of STP-H5/CSP. The image-processing research and application development for STP-H5/CSP was based on images captured by STP-H4/ISE 2.0 as an example test suite, since STP-H5 would feature images with similar orientation and image size.

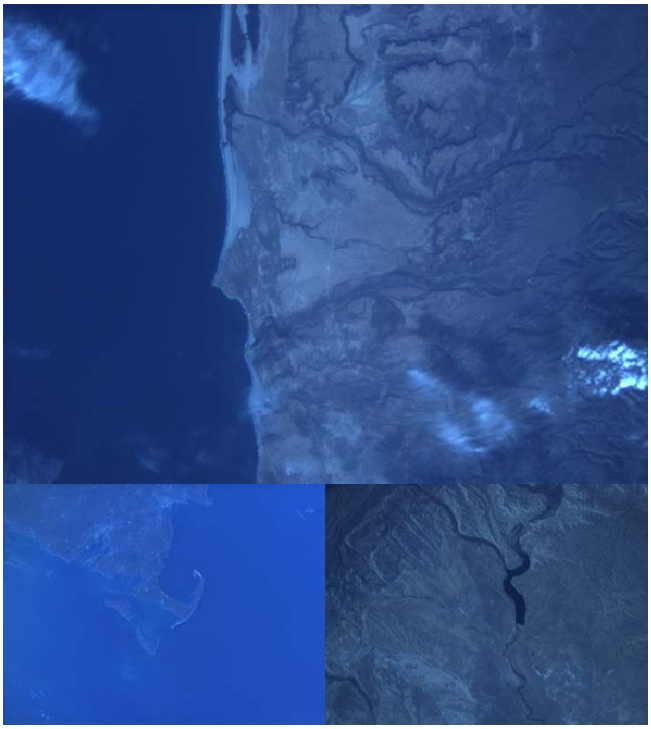

Fig. 3. Example images taken by STP-H4/ISE2.0[1]

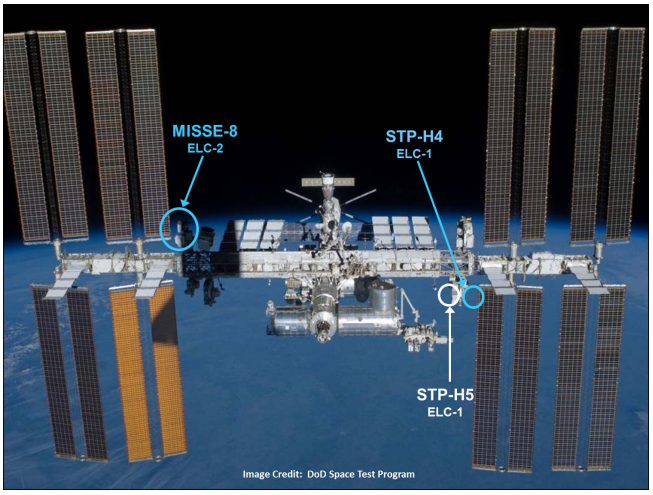

Fig. 4. Locations of STP payloads[2]

III. Related Work

There are a wide number of CubeSat missions that feature Earth imaging, however, there are only a few examples of missions with readily accessible imagery to the public, and even fewer missions that describe image classification in general. The most significant example mission featuring image classification is the Intelligent Payload EXperiment (IPEX) CubeSat mission based at Cal Poly San Luis Obispo in collaboration with California Institute of Technology's Jet Propulsion Laboratory [10]. For its image analysis, IPEX used the TextureCam image-processing suite, which uses a random forest classifier made up of decision trees which uses branching and a sequence of threshold tests to assign a classification probability upon reaching a terminal leaf node. A decision forest runs trees in parallel while analyzing spacial neighborhoods for incorporating texture. Hand-labeled training images were used to teach the classifier to categorize an image into four categories: clear surface (land or water), planetary limb (edge of the earth), clouds, and outer space. Compared to the classification techniques in this project, TextureCam has the benefit of machine learning and is reported to have high accuracy. However, as discussed in [10], their approach does not distinguish between land and water and can take more than half a minute to compute. They also noted problems updating the classifier, due to the training executable not being onboard, and because retrained forests were too large to uplink [11].

Another approach previously taken for small satellite image classification was the development of a Content Based Image Retrieval (CBIR) system. The CBIR approach was developed to be able to retrieve from a satellite the image most similar to an input image used as a query [12]. This system uses a combination of color, shape, and texture features from the image in order to calculate similarity. For color, a histogram approach was used to keep track of the number of pixels of each value. This approach does not account for ratios between color-components. For texture, the features extracted were contrast, correlation, energy, and homogeneity. For shape, the features extracted were perimeter, area, and roundness. These features are similar to the edge classifier features described in this paper in that single values are extracted from the entire image to represent an aspect of the image. Although this CBIR approach is not suitable to autonomously prioritize or delete images robustly and requires the user to have examples of high priority images to use for retrieval, the advantage is that no classifier training is necessary and arbitrary images that are temporarily high priority can be retrieved at any time.

IV. Approach

The main motivation for this work was to have a method that autonomously analyzes images before choosing to downlink a full image which would need to be performed by extracting image statistics to predict priority and relevance to the mission. In preparation for STP-H5/CSP, a suite of image-processing applications was developed for CSPv1 with the goal of being lightweight and portable, and to facilitate acceleration from parallelization. These applications include 2D convolution, various image classifiers, and various support applications to help with image-data manipulation. The typical image sizes STP-H5/CSP manipulates are displayed in Table 1.

A GUI-based classifier training framework was also designed, which allows for real-time, user-controlled image labeling and classifier calibration. This training framework needed to be flexible and easy-to-use to allow retraining for new missions.

[1] https://www.flickr.com/photos/nasa_goddard/albums/72157638323032963/with/11192970873/

[2] Image courtesy of DoD Space Test Program and NASA GSFC

TABLE I. IMAGE SIZES OF STP-H5/CSP FORMATS

Image Type	Storage Size	Resolution
RAW Image	25.09 MB	40-bit 2448×2050
PPM	15.05 MB	24-bit 2448×2050
J2K Thumbnail	0-350 KB	24-bit 489×410

V. IMAGE CLASSIFIERS

To identify high-priority, terrestrial-scene images, a set of image classifiers was built to detect various aspects of desirable images. All the algorithms used either process each image pixel individually or also take into account neighboring pixels. Each image pixel contains 24 bits of color data, which corresponds to a value of 0-255 for the red, green, and blue component values. By analyzing these values with varying methods, autonomous conclusions can be made about the image contents. After developing various classifiers to handle identifying the various categories of images, the classifiers naturally fell into two categories: general classification and specific classification.

A. General Classification

The main goal of the general classifiers was to autonomously classify the terrestrial-scene image composition with four categories: cloud, land, water, and dark/space. The image pixel data is first loaded into a large array where it can then be analyzed and processed by the classifiers. As shown in Fig. 5, the array containing the processed image data can be converted back to an image to show classification results visually.

1) Color Classifier: The color classifier sets color thresholds, where certain pixel color ranges are set to correspond to the four categories. These ranges are scaled dynamically based on the average brightness of the image. For visualization purposes, the categories cloud, land, water, and dark/space are represented with red, green, blue, and black respectively. The color classifier was fairly accurate in categorizing the cloud and space categories, but it was difficult to distinguish land from water in darker images.

2) Texture Classifier: The texture classifier calculates the standard deviation around each pixel to calculate a value for roughness. Water generally has a low roughness level in terrestrial-scene images, which allows it to be easily distinguished even in darker images. Based on the roughness level calculated for each pixel and the average brightness of the image, the pixel is either classified as water or not water (blue and green respectively). The result of the texture classifier is only used to supplement the color classifier, by overriding water pixels with land pixels.

3) Image Combiner: The image combiner merges the desired aspects of the two classifiers. Because the texture classifier was developed solely to supplement the accuracy of water/land classification, the image combiner uses the color-classified image as the base, allowing the texture classifier image to overlap only the blue regions. After the image combiner stage, the percentages of cloud/land/water/space are recorded in a log file. An example of these applications working together can be seen in Fig. 5.

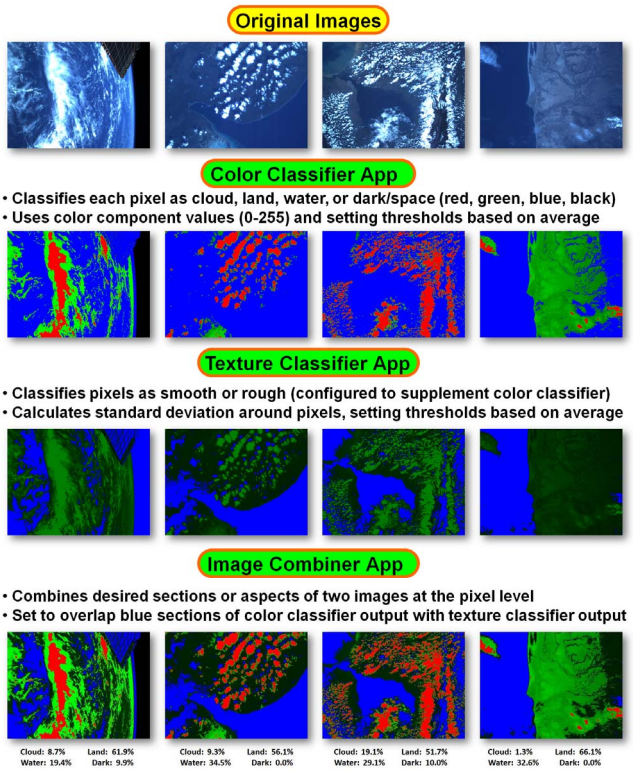

Fig. 5. General classification of terrestrial-scene images

B. Specific Classification

The main goal of the specific classifiers was to be able to search an image for specific features or aspects that could be isolated and quantified. Fig. 6 shows example outputs of these classifiers.

1) Edge Classifier: The edge classifier analyzes the color and quantity of pixels in an image after applying a blur filter followed by an edge filter using 2D convolution. By looking at the sample terrestrial-scene images, there were three features of interest: mountains, snowcapped mountains, and sunglint images showing sunlight reflection. After the image was edge filtered, mountains were detectable by a high amount of blue and green colored pixels, snowcapped mountains were detectable by a high amount of blue, green, and red pixels, and sunglint images were detectable by a high amount of green and red colored pixels. The edge classifier records the average brightness, number of pixels above a certain brightness threshold, and color-component ratios in a log file.

2) Color-Search: The color-search application finds all pixels in an image matching an input color and range. The red, green, and blue component values of each pixel are compared with the input color-component values and if the percentage difference is under the input range, it is classified as a match. In terrestrial-scene images, the color of the land can vary depending on the region, with some examples shown in Fig. 6. Aside from being able to detect certain land types, it can also be used to detect city lights for images taken at night, and

specific colors representing unique cases, such as lens flare and camera glitches.

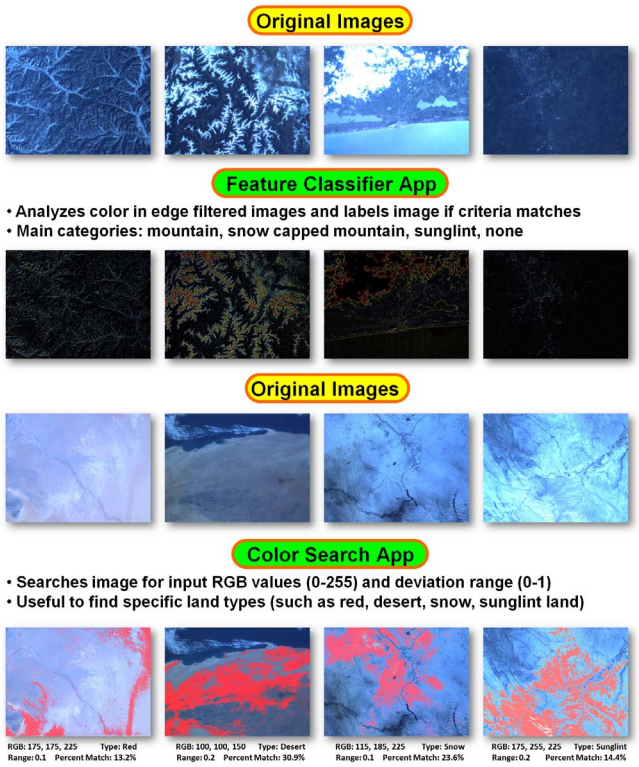

Fig. 6. Specific classification of terrestrial-scene images

VI. Classifier Training

The accuracy of the developed classifiers depends on the calibration settings to be set properly. A classifier training framework was built to obtain the current optimal classifier settings and to easily adapt the classifier settings for other missions with different cameras and lenses.

A. Image-Labeling GUI

OpenCV, which is an open-source computer-vision library, includes a high-level GUI (HighGUI) for quickly building a user interface without strict requirements. A classifier training and image-labeling GUI was built with OpenCV's HighGUI module to train all of the classifiers concurrently while being able to see the classifier outputs in real time, as well as, to be able to easily navigate between images and keep track of the calibration outputs. A labeled image of the image-labeling GUI can be seen in Fig. 7.

On the left side of the GUI, the color and texture classifier outputs can be seen in real time and can be adjusted using two sliders each. The two sliders both adjust the same threshold value and the goal is to find the minimum and maximum threshold values that result in an acceptable output. The purpose of this feature is to later find a best fit threshold that satisfies the most number of images.

On the right side of the GUI, the original image can be seen as well as the category checkboxes and the image navigator. The category options are those that can be determined using the edge classifier and color-search application. The original image can be clicked to apply the color-search application functionality using the selected pixel as an input. This feature can be used to identify colors of interest to add to the autonomous, image-prioritization logic.

After adjusting the sliders for the color and texture classifiers, selecting all applicable checkboxes, and optionally clicking the original image to apply the color-search application functionality, the "save+next" button can be pressed to write the filename, calibration data, and image statistics such as average RGB component averages and other extracted features to a log file. This process can be repeated to rapidly label a large number of images.

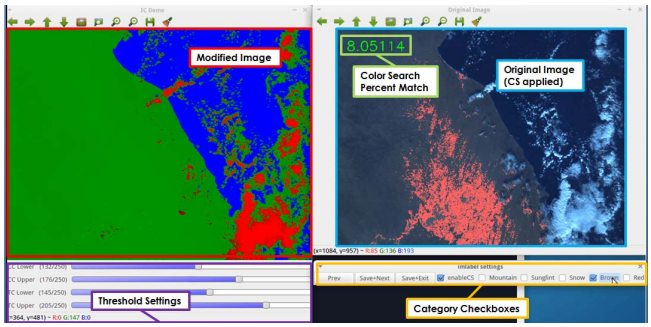

Fig. 7. Image-Labeling GUI built with OpenCV's HighGUI

B. Data Parsing and Gnuplot

After labeling the set of training images, the output log file containing the labels is processed to obtain information to help with classifier calibration. For the color and texture classifiers, a threshold is selected that fits the most ranges set during labeling. For the category checkboxes, correlations are found between the extracted image features and the matched categories. For the colors of interest using the color-search application, similar colors can be combined and the rest can be ranked and added to the image-priority calculation logic.

To help with range fitting and category correlation, the data can be plotted with Gnuplot, a command line graphing utility commonly available in most Linux distributions. For range matching with the color and texture classifiers, the data for the image labels is sorted based on the range medians before being plotted as seen in Fig. 8. In the figure, it is visually represented that a threshold value around 160 fits most of the ranges in the upper left plot. This outcome means that, with a classifier setting of 160, these test images would be satisfactorily classified. A strictly increasing or decreasing trend in one of the other plots signifies that the threshold could be modified by one of the other values to result in a more uniform set of ranges. Having ranges more uniform allows for the chosen classifier threshold to be closer to the center of more threshold ranges overall which means a more accurate classification result overall. This plot shows the result after tuning thresholds based on the color-component averages; therefore, there is no major upward or downward trend corresponding with the increasing threshold ranges. Nonetheless, there is still correlation between images with narrow threshold ranges and low color-component averages. These also correspond with

images with a high average brightness after applying 2D convolution (plot in the bottom right). These correspondent features can be used to detect images with very specific ranges. For category correlation, a similar plot can be generated by sorting the category match (0 or 1). The left side of all the plots will correspond with images not matching the category while the right side will match with images matching the category, which allows for the user to see correlations between the extracted image features and the category match.

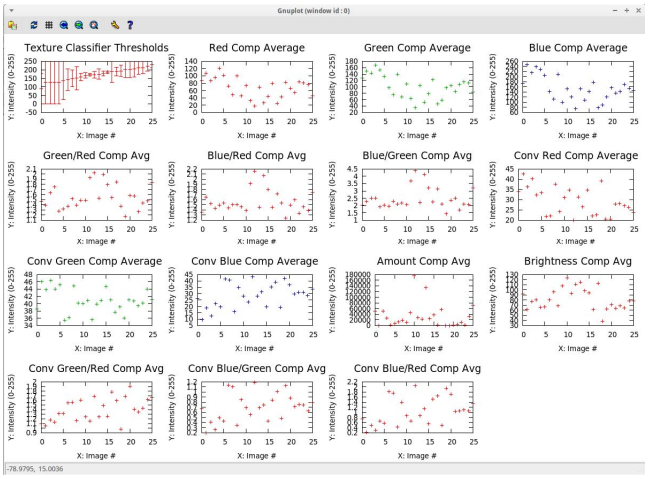

Fig. 8. Image-Labeling GUI output plotted with Gnuplot

VII. Efficient Processing

Space-computing missions feature embedded devices with limited resources. Therefore, minimizing computational complexity and maximizing computational efficiency is important to obtain results quickly and minimize power consumption.

The image-classification and processing algorithms were all written in 'C' with no external dependencies which allowed the algorithms to be extremely efficient and portable. The most computationally-intensive application used for the classifier framework was 2D convolution, so this application was the main target for acceleration.

Several ARM processors, such as the Cortex-A9 processor featured in the Zynq on CSPv1, includes the NEON Media Processing Engine which provides a Single-Instruction Multiple-Data (SIMD) instruction set for computing on large sections of data in parallel. SIMD functions by loading NEON registers with multiple values for calculation and applying mathematical operations to these registers, effectively applying them to all of the values in parallel. This decreases the amount of time required for the calculation. This method can be used by manually packing NEON registers in assembly code or by using NEON intrinsic functions (function calls that the compiler replaces with the appropriate NEON instruction, or sequence of instructions) in 'C'.

For this project, SIMD using NEON intrinsics was applied to 2D convolution using 3×3 kernels. Table 2 shows execution times for 2D convolution on a 2448×2050 image, which is the dimension of images taken by STP-H5/CSP. The testing platform used was the Zedboard, which uses the same System-on-Chip (SoC) that the CSPv1 uses (Xilinx Zynq 7020). The results showed 2x speedup achieved when using NEON *q* registers (128-bit wide registers made up of two 64-bit regular registers), which although require additional time for register load, increases the number of elements processed per operation from 4 to 8.

The times in Table 2 did not consider the time required for image read and write, which took approximately 0.05s for read and 0.16s for write. Additional time is also needed for data deinterleaving if the camera outputs interleaved RGB. Deinterleaving and re-interleaving the data took approximately 0.20s and 0.43s respectively.

These experiments were also conducted for grayscale images. The output of grayscale 2D convolution for edge detection can be just as useful as its color counterpart depending on the application. For grayscale images, image read and write took approximately 0.017s and 0.036s respectively. These times and the times in Table 2 are not limited to 3× more efficient (as would be expected) and approach 4×. This outcome is attributed to processing one array of data directly rather than an array of structs containing red, green, and blue components. In order to load the NEON registers, the grayscale data still had to be converted to 16-bit data which took approximately 0.07s, and then 0.10s to convert back to 8-bit data. Under these conditions, it is only more efficient to use NEON when filtering an image more than once and the effective speedup approaches the figures in Table 2 and Table 3 as the number of iterations increases.

TABLE II. Color 2D convolution execution times

Type	Execution Time (s)	Speedup
Serial (without NEON)	1.21	1
NEON 64-bit registers	0.68	1.78
NEON 128-bit registers	0.51	2.37

TABLE III. Grayscale 2D Convolution Execution Times

Type	Execution Time (s)	Speedup
Serial (without NEON)	0.33	1
NEON 64-bit registers	0.22	1.50
NEON 128-bit registers	0.16	2.06

VIII. Conclusions

Due to severe resource restrictions on CubeSats and other small spacecraft, it is critical that applications strive to be efficient, lightweight, and portable. This paper presents an application library that is designed with these key attributes under consideration. By focusing on necessary functionality (e.g., simple classifier algorithms and static convolution filter sizes), application acceleration and portability considerations are reduced in complexity.

This paper also presents a classifier training framework to complement the image applications. A flexible framework is essential in facilitating the use of these applications when

transitioning to future missions and other cameras. By employing these applications, images on CSPv1 on the International Space Station can be analyzed and subsequently deleted or prioritized for download autonomously, thereby minimizing onboard non-volatile memory usage and increasing the quality of images received. This feature is desirable and essential for sustained mission operation because of the inherent complexity in sending commands to the CSPv1 operating through the ISS operation center.

ACKNOWLEDGMENTS

This research was funded by industry and government members of the NSF CHREC center, and the National Science Foundation I/UCRC Program under Grant No. IIP-1161022. The authors would like to thank Tom Flatley, branch head of NASA Goddard's Science Data Processing Branch, for providing the large suite of reference images from STP-H4/ISE2.0 to use as a training and example dataset.

REFERENCES

[1] K. LaBel, "Radiation Effects on Electronics 101: Simple Concepts and New Challenges," NASA Electronics Parts and Packaging Program, Apr. 21, 2004.

[2] National Academies of Sciences, Engineering, and Medicine, "Achieving Science with CubeSats: Thinking Inside the Box," The National Academies Press, Washington, DC, 2016.

[3] S. Waydo, D. Henry, and M. Campbell, "CubeSat Design for LEO-Based Earth Science Missions," IEEE Aerospace Conference, Big Sky, MT, Mar. 9-16 2002.

[4] M. A. Swartwout, "CubeSats and Mission Success: A Look at the Numbers", 2016 CubeSat Developers Workshop, San Luis Obispo, April 2016.

[5] T. Parece, J. Campbell, and J. McGee, Remote Sensing Analysis in an ArcMap Environment, Blacksburg, VA: VAView, 2015.

[6] D. Rudolph, C. Wilson, J. Stewart, P. Gauvin, G. Crum, A. D. George, M. Wirthlin, and H. Lam. 2014. CSP: A multifaceted hybrid system for space computing. In Proceedings of the 28th Annual AIAA/USU Conference on Small Satellites, Logan, UT, Aug. 2-7, 2014.

[7] E. Sims, "The Department of Defense Space Test Program: Come Fly with Us," IEEE Aerospace Conference, March 2009.

[8] C. Wilson, A. George, et al., "CSP Hybrid Space Computing for STP-H5/ISEM on ISS", Proc. of AIAA/USU Conference on Small Satellites (SmallSat), Logan, UT, Aug. 8-13, 2015.

[9] A. Schmidt, M. French, and T. Flatley, "Radiation hardening by software techniques on FPGAs: Flight experiment evaluation and results," IEEE Aerospace Conference, Big Sky, MT, Mar. 4-11, 2017.

[10] S. Chien, J. Doubleday, D. Thompson, K. Wagstaff, J. Bellardo, C. Francis, E. Baumgarten, A. Williams, Edmund Yee, D. Fluitt, E. Stanton, J. Piug-Suari, Onboard Autonomy on the Intelligent Payload EXperiment (IPEX) Cubesat Mission: A pathfinder for the proposed HyspIRI Mission Intelligent Payload Module, Proc 12th International Symposium in Artificial Intelligence, Robotics and Automation in Space, Montreal, Canada.

[11] J. Doubleday, S. Chien, C. Norton, K. Wagstaff, D. Thompson, J. Bellardo, C. Francis and E. Baumgarten, "Autonomy for remote sensing - Experiences from the IPEX CubeSat", 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2015.

[12] J.M. Gashayija and A. Bierman, "Development of Nanosatellite based Image Retrieval System", *International Journal of Computer Applications*, vol. 99-No.11, pp. 25-31, Aug. 2014.