CubeSat Payload Software Overview

# Introduction

The primary objective of the LORIS Payload team is to research, then develop a camera system utilizing color and near infra-red (NIR) cameras that can operate in a near orbit environment. The target location of these photographs is Nova Scotia and its surrounding oceans. The system would have to accommodate for challenges that arise in this environment, notably communication delay and limited download opportunities. To account for these problems, the payload team created software with multiple optimizations and countermeasures to prevent these problems, while still excelling at the core objective of taking photos. While designing the software, the team decided to focus on three areas. These areas are functionality, speed and adaptability.

# Functionality

Functionality was a core attribute that the Payload team heavily focused on. It was of highest importance that the Payload Software achieved all of the objectives that was demanded of the OS. These objectives include support for color and NIR cameras to allow for highly detailed imagery while keeping file size low. Having a low digital footprint is important since the photographs have limited opportunities to be extracted from the satellite. To this end, we utilized the highly efficient Open Computer Vision (OpenCV) library, an open source library of various computer vision and compression algorithms. As a result, the software has seamless support for dual cameras and twenty-two different compression formats (*see appendix for list)* while opening the door for future improvements and features in the future via software patches. The strength and breadth of this library allows for a highly efficient software design and easy implementation of industry standard compression techniques. As a result, the software is able to take photographs from both cameras while keeping file sizes under **600KB**. This allows the software to maintain a low digital footprint and allowing for ease of transporting photographs from the satellite to earth.

# Speed

Optimization and speed were considered during every step of the design and development process. The need for speed came as a result of potential communication delay from the moment we issue a command to take a photo to when the satellite operating system processes and activates the camera acquisition software. This can result in potentially inaccurate photos since the cameras could miss the target location of the photographs. To account for this problem, we prioritized initializing both cameras and grabbing their frame data as soon as possible. This is made possible due to OpenCV allowing for frame data to be saved and held before being compressed and written to the disk. As a result, during preliminary testing of the software we were able to achieve picture taking speeds under **900ms** for each test camera. The low latency of the software allows for accurate and precise picture taking of intended locations.

## Sample Runs

Hardware:

* MacOS Mojave 10.14
* 2.7Ghz Intel i5
* 8GB DDR3 Ram
* Camera 0 resolution: 800 x 600
  + Average Image Time: ~900ms
* Camera 1 resolution: 1280 x 720 (accurate to CubeSat cameras)
  + Average Image Time ~850ms

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Image Capture Time (ms) | Compression Time | Total Time |
| 1 | 1775 | 37 | 1812 |
| 2 | 1646 | 40 | 1686 |
| 3 | 1713 | 37 | 1750 |
| 4 | 1579 | 43 | 1622 |
| Average | 1678.25 | 39.25 | 1717.5 |

Total File Size: 525KB with >1 KB variation between tests using JPEG compression

# Adaptability

Lastly, adaptability was stressed due to the long duration of the mission and the unpredictability of a near orbit environment. The adaptability methods we implemented aim to reduce the amount of maintenance that the software will require, conserving upload time for other parts of the operating system. These methods include many error checking and prevention methods throughout the runtime of the software. The methods ensure that the various hardware and software components of the system work seamlessly together while accounting for the hardware errors that near orbit environments can produce. A notable example of these methods includes a suite of command line arguments which allow the OS to customize the behavior of the payload software. These arguments include error/behavior logging, compression techniques and targeting specific cameras among other arguments. This creates a highly flexible and reliable software as well as introducing metrics to further analyze the performance of the software.

# Conclusion

Overall, the software that the Payload team created was specifically designed for image acquisition of Nova Scotia and its surrounding ocean in near orbit. By focusing on important problems that arise in near orbit environments, we have created software that will reliably and rapidly achieve the objectives given to the team while giving opportunities for many future improvements and optimizations in the future. The immense power and flexibility of the OpenCV library allows for an efficient operation and easy implementation of industry standard compression and computer vision algorithms. Further, by focusing on speed as one of the core tenants of the software, the software is able to account for communication delay produced by being in an near orbit environment increasing the accuracy of the satellite. As such, we believe that our software will provide an excellent platform for the LORIS CubeSat to excel at its mission while providing options for further functionality in the future.

# Appendix – Available Software Compression Formats

* Windows Bitmaps
  + .bmp
  + .dib
* JPEG files
  + .jpeg (default)
  + .jpg
  + .jpe
* JPEG 2000 files
  + .jp2
* Portable Network Graphics
  + .png
* WebP
  + .webp
* Portable Image Format
  + .pbm
  + .pgm
  + .ppm
  + .pxm
  + .pnm
* Sun Rasters
  + .sr
  + .ras
* TIFF files
  + .tiff
  + .tif
* OpenEXR Image files
  + .exr
* Radiance HDR
  + .hdr
  + .pic
* Raster and Vector geospatial data supported by GDAL\*

Reference:

OpenCV Documentation – cv::imread https://docs.opencv.org/4.0.0/d4/da8/group\_\_imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56