EE367 Project Proposal

Debris Detection from Stacked Star Tracker Images

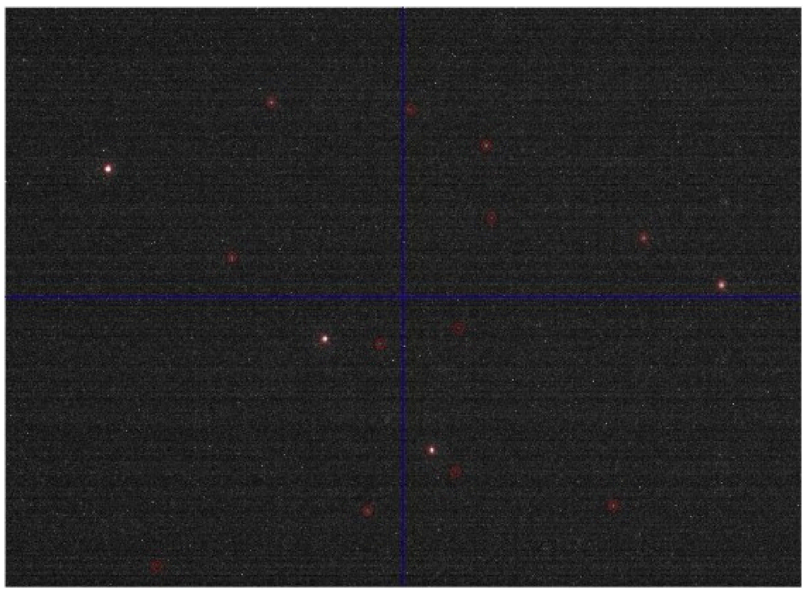
Tim Vrakas

Overview and Motivation

Modern LEO satellite constellations consist of thousands of space vehicles, and are fast approaching tens of thousands. Constellations of this size are subject to significant collision risk. Reducing this risk involves cateloging orbital objects, and performing minor orbital adjustments to reduce the risk of collision. For a conjunction (a close approach) between two active vehicles operated by the same entity, planing avoidance manuvers is relatively straightforward. However, for dead vehicles, non-cooperative vehicles, and other debris, operators must currently rely on ground radar systems to measure, index, and track objects. Organizations like Space-Track and LeoLabs provide tracking services to help prevent collisions. The bigest shortcoming of these systems is their revisit frequency. Because they rely on a small number of radar systems across the world, they cannot provide frequent updates on the position of all objects.

Many satellites are equipt with Star Tracking imagers, which take images of the sky at known angles, identify bright stars, and use the position of those stars to compute vehicle orientation. These imagers are generally tuned for high gain, to reliably image dim stars. I will explore improving the image quality from these sensors, performing multi-image stacking to improve SNR, and detecting debris moving across the frame.

Here's a few examples of these Star Tracker Images:



Related work

The CMOS imaging sensors used in these constellations are sensitive to the high flux of proton radiaition present in earth orbit. This is explored in [4], where the noise levels of these CMOS sensors are used to map the radiation environment in real time. Exposure to radiation causes bright spots on the imager, when particles deposit energy, as well as development of fixed patern noise on the sensor over time, due to Total Ionizing Dose (TID) damage to the pixels and readout electronics. The rate and intensity of these bright spots depends heavily on the radiaion flux, which varies over the Earth's surface, and over time. This may help inform an acurate prior of the noise we expect to see.

In standard star tracker application, single still images are captured, and a thresholding and centroiding algorithim is used to identify the positions of stars in the image. In [1], an improved thresholding algorithim is used to improve peformance in a CMOS imaging system that has been degraded by radiation.

An image processing pipeline is described in [2], where images downlinked from an experimental imager are first passed through an image artifact correction step, then a registration step, to project the image into an inertial frame, so that multiple images can be compared. Then, elements common to multiple images are removed, leaving only elements that are moving across the background. Then, tracking is performed across multiple images to extract a position and velocity vector, which can be used to compute object orbit.

Improved classification is the focus of [3], where "RSONet", a CNN consisting of 3 convolutional layers, followed by a fully connected layer. This network is trained on simulated Point Spread Functions, enabling the CNN to clasify elements as stars, noise, or space objects.

Goals

There are several goals to explore:

0) Generate synthetic training data representitive of Star Tracker images

I don't have access to enough image/image sequences to properly train a CNN, and even if I did, I would need to label the objects in the image in order to use them for training. To sidestep this issue, I'm going to try to generate some realistic training data that similar features and a similar noise distribution to real star trackers images. This may not generalize well to true images, but if I had access to a large set of data, and time to label it, I could add that later.
Timeline: 3/1

1) Filter out fixed-pattern noise caused by radiation damage

I can explore approaches to filtering single images, and evaluate which approaches are effective at eliminating fixed-pattern noise but still maintain sharpness of stars in the image. The optimal filtering aproach may depend on the radiation environment, and so the parameters (or prior) of the filter may need to change with the environment.
Timeline: 3/3

2) Perform image stacking on Star Tracker images

It may be possible to significantly improve SNR for celestial objects, by shifting and stacking images. This should reduce the effect of temporary bright-spots, which do not persist between readouts. However, this approach will likely obscure near-earth objects moving across the frame.
Timeline: 3/8

3) Train a CNN on multi-image sequences,

This may be useful to detect differences between frames, and better seperate background stars, moving objects, and radiation spots. I'd like to train a neural network to detect and seperate features depending on persitance and motion in frame. This will most likely involve generating a set of synthetic training data, because I don't have access to a large dataset of labeled images.
Timeline: 3/10

References

[1] Abigail Lee, "Star Tracker Algorithm Improvements to Restore Performance After Radiation Exposure", SSC23-XII-02, https://digitalcommons.usu.edu/smallsat/2023/all2023/2/

[2] Elozor Plotke, Peter C. Lai, "Dual Use Star Tracker and Space Domain Awareness Sensor In-Space Test", AMOS 2021, https://amostech.com/TechnicalPapers/2021/Poster/Plotke.pdf

[3] Dave S, Clark R, Lee RSK. "RSOnet: An Image-Processing Framework for a Dual-Purpose Star Tracker as an Opportunistic Space Surveillance Sensor" *Sensors (Basel)*, doi: 10.3390/s22155688, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9370977/pdf/sensors-22-05688.pdf

[4] Hamil Shah, Russell Van Cleave, "Global Ionizing Radiation Environment Mapping Using Starlink Satellite Data" *IEEE Transactions on Nuclear Science* 10.1109/TNS.2023.3339132. https://www.researchgate.net/publication/376464384_Global_Ionizing_Radiation_Environment_Mapping_Using_Starlink_Satellite_Data

In [ ]: