Picosatellite Relative Pose and Position via CUDA Accelerated Computer Vision

ArGus Your Fire Nation
Eric Swenson, Gustavo Cornejo, Andrew Tran, Alex Thielk, Angel Ortega
AEROSPACE
Ron Scrofano, Aaron Brown

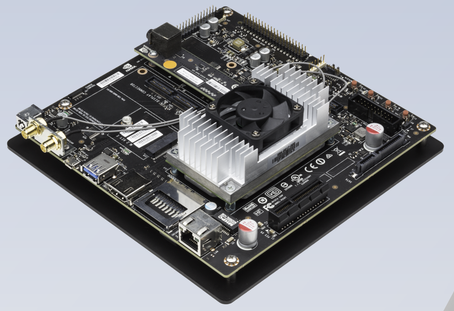
Picosatellites & Our AeroCube Analog

Satellites are expensive — for a full-size satellite, a single deployment mission can cost up to **400 million dollars**.

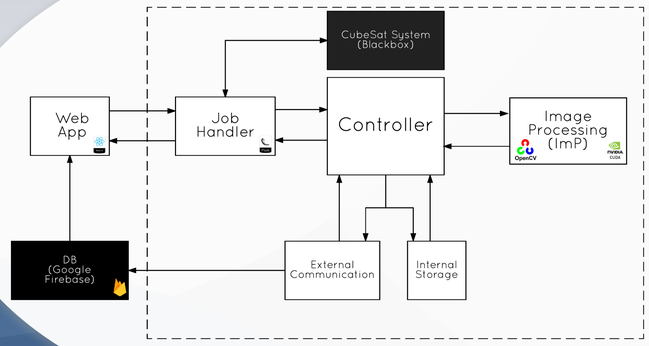
There are now a number of smaller satellite sizes, all the way down to microsatellites between 1kg and 10kg or picosatellites **under 1kg**. One standard for picosatellites is a CubeSat called the **AeroCube**.

For our project, Aerospace provided a NVIDIA **Jetson TX1** running a Cortex A57 and a Maxwell architecture GPU with 256 CUDA cores capable of **over 1 TeraFLOPs** to act as our AeroCube analog.

Meet the Jetsons



Component Structure



The Problem

The primary strengths of picosatellites are dependent on **precise formation flying** while completing a distributed task, such as simulated "large-aperture" interferometry to analyze the chemical signatures of planets thousands of light years away.

Existing solutions for understanding pose and position have **strict limitations** such as the presence of a known magnetic field, line of sight to the sun, or precision insufficient for close-range (under 1m) maneuvers.

The Constraints

As though developing for an actual picosatellite, our system is constrained by **limited energy resources** and **limited computation power**.

In addition, the lack of an actual picosatellite system required the encapsulation of modules so that simple mixins could be applied to utilize the code in another system altogether.

OpenCV

Open-source computer vision library with Python bindings (except for CUDA modules)

CUDA C/C++

GPU-accelerated technology, with usage in OpenCV's CUDA modules

Python

Python compiler to C/C++ code that allowed Python bindings to CUDA-accelerated C/C++ functions

Fiducial Markers

Fiducial Markers are objects used to estimate scale in an image because they are a known size and shape (e.g., a ruler). OpenCV's **ArUco** library produces such markers with unique IDs encoded as bits, as seen to the right. Each marker can then be used to identify a side of an AeroCube in addition to conveying information about the position and pose of the AeroCube.

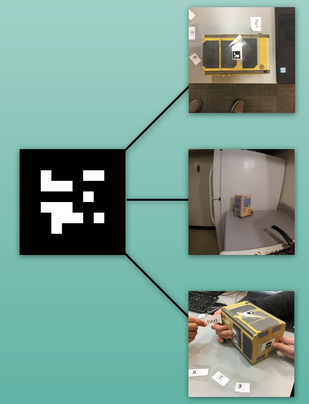


Image Processing to Models

How do we map marker information to AeroCube information?

By placing unique markers on each side of multiple AeroCubes, we can use the marker ID to determine the AeroCube's ID and face. Because we can identify the face, we just need to apply transformation matrices (rotation and translation) to determine the AeroCube center's position and relative pose from the markers — if we have multiple markers for an AeroCube, we can simply average their calcalated centers.

Why represent pose as a quaternion?

Quaternions are 4-element vectors that are commonly used in aerospace to represent pose due to a couple of nice properties. They compactly represent **rotation in 3D space** (as opposed to a 3x3 rotation matrix). Additionally, multiplying two quaternions together to combine their rotations is faster than multiplying two rotation matrices together. Quaternions also avoid certain issues faced with other pose representations — e.g., they avoid the issue of gimbal lock when using Euler Angles.

Lifecycle of an Event