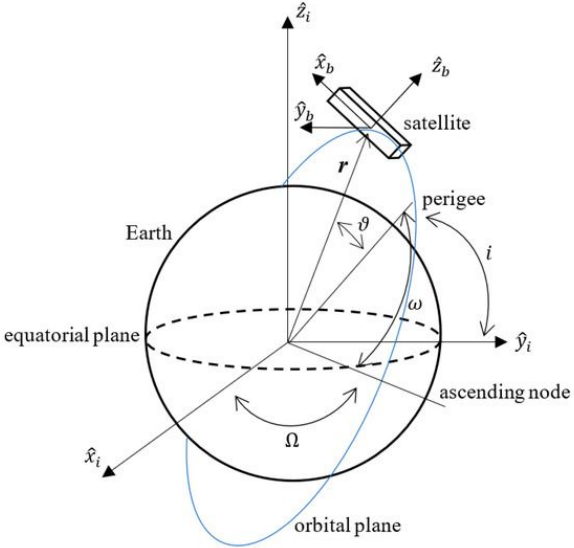
Visual Attitude Estimation

Team Mike: Will Pope, Victor Xia, Jinhe Xu, Alex Zhen
Advisor: Professor Mehran Mesbahi

Overview

- ➔ Background and Theory
- ➔ Prompt, Purpose, and Requirements
- ➔ System Design
  - ◆ Hardware
  - ◆ Image Processing
  - ◆ Star Identification Algorithm
  - ◆ Confidence Level
- ➔ Test and Validation
- ➔ Conclusions and Left-to-Do



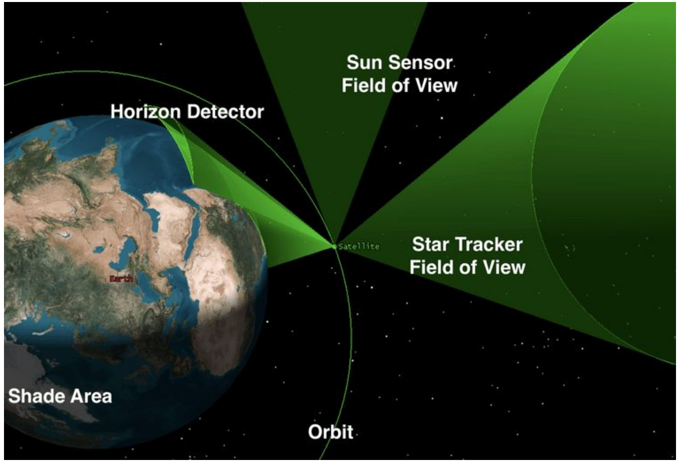
Background and Theory

➔ Spacecraft star tracker

- Very accurate (and very expensive) optical sensor
- Calculates attitude based on position of stars in field of view (FOV) of sensor
- Used to generate “true” attitude to update attitude propagation on-orbit

➔ Operating theory

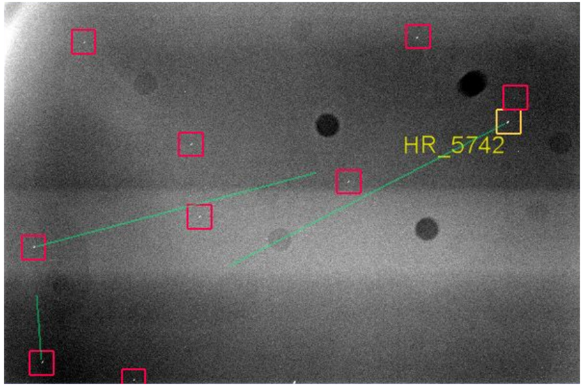
- Star positions are fixed inertially (approx.)
- Sensor is loaded with star catalog
- Geometric algorithm identifies stars in FOV
- Computes camera frame to inertial frame



Prompt, Purpose, and Requirements

- ➔ **Prompt**: using a camera and a pattern on a wall, determine the orientation of a tilting camera platform
- ➔ **Purpose**: create a cheap setup to explore star tracking algorithms
- ➔ **Requirements**:

#	Requirement	Subsystem
1	Capture and store image at 99.9% success	Hardware
2	Locate stars on 2-D plane within 3.5% error	Image Processing
3	Calculate attitude within 2.0° of true attitude	Star ID Algorithm
4	Produce confidence level from star map metrics	Confidence Level

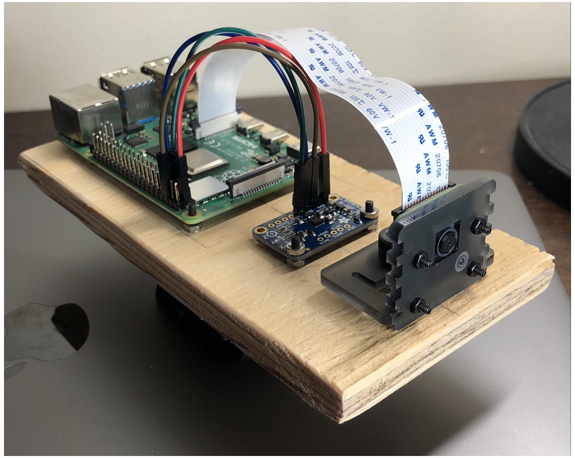


System Design:

Hardware

➔ **Raspberry Pi-based setup:**
- Raspberry Pi - controls peripherals, runs algorithm on-board in Python
- Camera - takes 8.0 megapixel photos of star map
- Accelerometer (IMU) - outputs true angle of camera platform
- Tripod - provides 135° of roll about +z-axis

➔ **Star map:** black dots on a white poster board



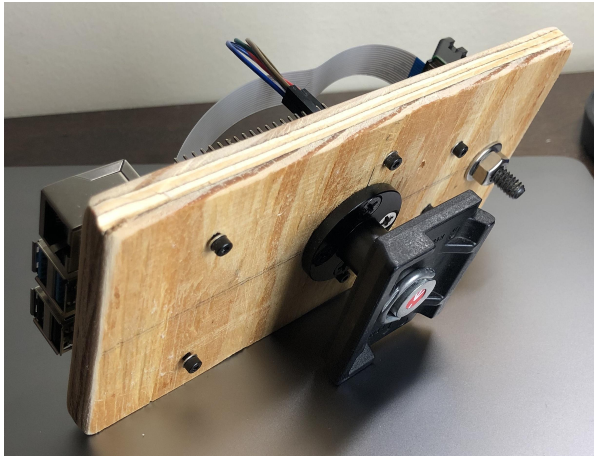
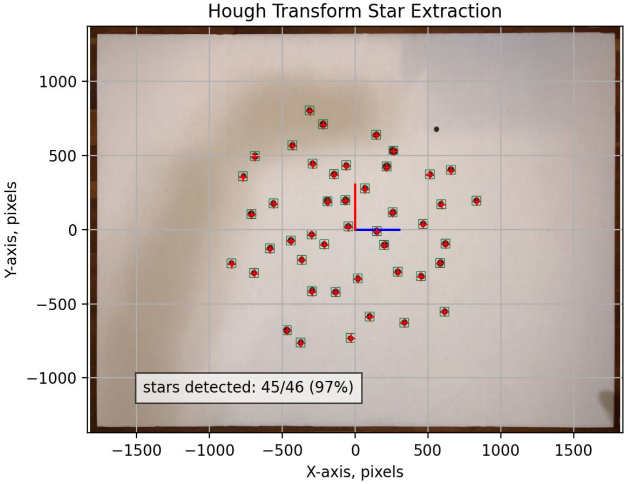
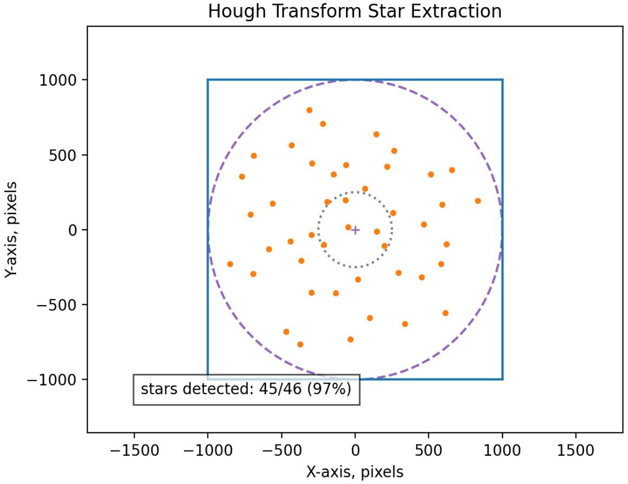


Image Processing

- ➔ **Function:** produce usable coordinates from visual input
- ➔ **Method:** Circle Hough Transform
  - ◆ Feature extraction technique from Python OpenCV library
  - ◆ Detects center of circular curvatures (left), produces array of 2-D coordinates (right)





System Design:

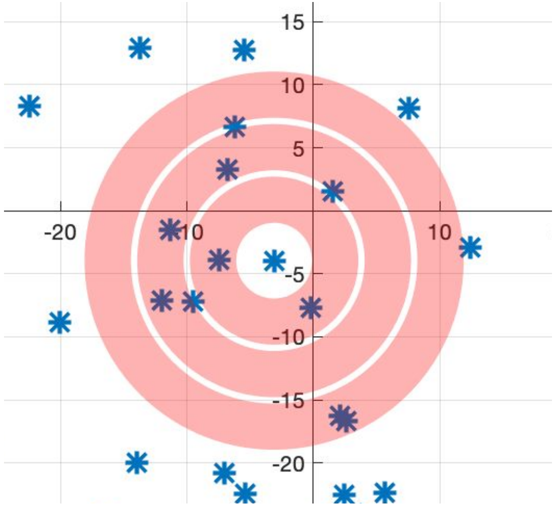
Star Identification Algorithm

➔ **Two modes:**

- Catalog - builds star catalog from given star map (done by a telescope on Earth)
- Flight - analyzes FOV, references stars to given catalog (done by satellite on-orbit)

➔ **Ring Method:** core of algorithm, creates unique profile for each star

- A numerical label/name
- X-Y coordinates in the star map
- Numeric “fingerprint” based off 3 rings around star
  - Number of stars in each ring
  - Position of the stars in each ring

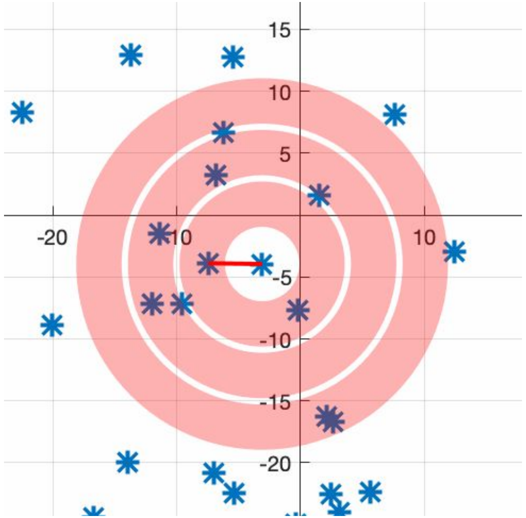


Star Identification Algorithm (cont.)

- **Star fingerprint:**
  - 3 numbers calculated with the following equation:

$$\alpha_j = \sum_{i=1}^{n} ||v_1 \times v_i||_2$$

    - j=1,2,3 corresponds to the three rings
    - i=1,2,3,n corresponds to the stars within each ring



Star Identification Algorithm (cont.)

- **Operation:**
  - Catalog Mode:
    - Takes reference photo at 0 deg roll angle
    - Runs Ring Method on every star in the FOV, stores fingerprints
  - Flight Mode:
    - Takes target photo at unknown roll angle
    - Runs Ring Method on several stars at the center of the FOV
    - Finds target stars in catalog
    - Calculates roll angle to rotate target coordinates to match catalog coordinates
  - Three outcomes:
    - Success - calculates roll angle within error margin of true roll angle
    - Failure - calculates roll angle outside error margin of true roll angle
    - Indeterminate - algorithm is unable to calculate a roll angle from the given information

System Design:

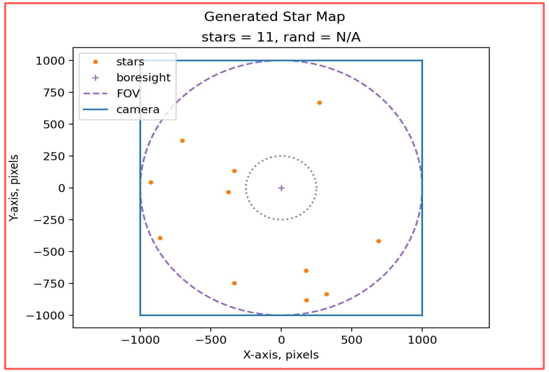
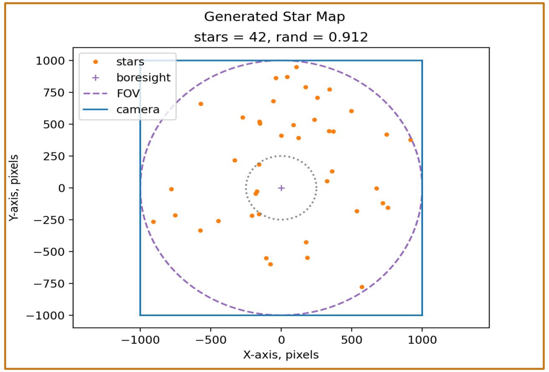
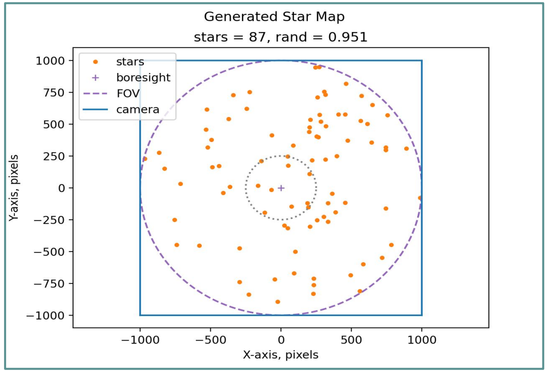
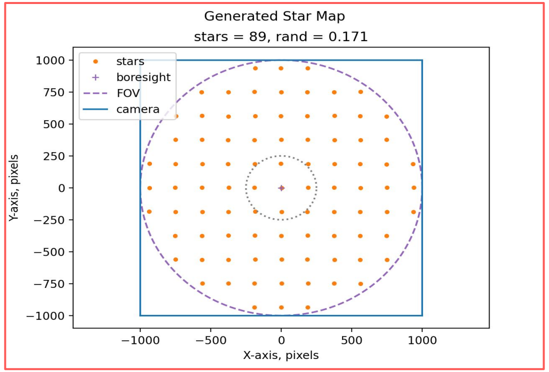
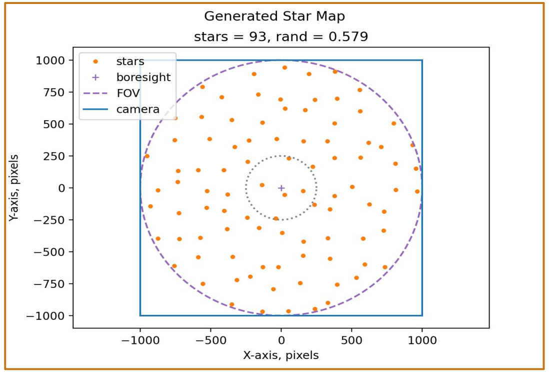
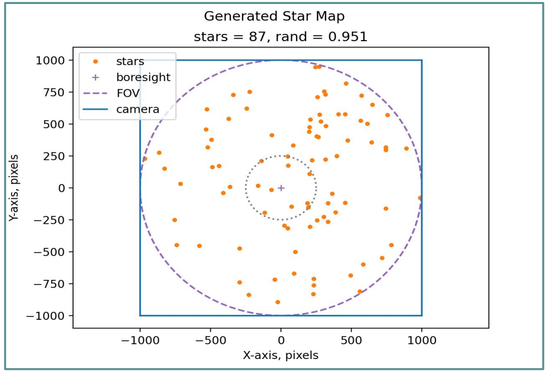
Confidence Level

- ➔ **Likeliness that calculated output is correct** (scale: 0-100%)
  - Quantifies performance envelope of the algorithm
  - Used in higher level sensor fusion algorithms
- ➔ **Determined empirically through iterative testing**
  - Randomly generate simulated star maps
  - Measure star map for two independent stellar metrics:
    - Stars in FOV (scale: 0 - 200)
    - Randomness of spatial distribution (scale: 0.00 - 1.00)
  - Run identification algorithm repeatedly per map, per metric step
    - Records accuracy and failure point for range of each metric
- ➔ **Flight mode:** calculate metrics of target photo, gives historical success rate

System Design:

Confidence Level (cont.)

➔ **Metrics:**

	Bad	Okay	Good
Stars in FOV			
Randomness of Spatial Distribution			

Test and Validation

Algorithm

- ➔ Tested through confidence level iterative testing
  - **Valid if:** Averages +/-2.0° accuracy in realistic stellar metric range

Imaging

- ➔ Tested through real world tests on poster board star field
  - **Valid if:** Imaging system introduces no more than +/-2.0° error to algorithm results

Conclusions and Left-to-Do

- **Conclusions:**
  - Dangers of scope creep
  - Value of working meetings

- **Left-to-Do:**
  - Mesh algorithm with imaging and simulation code
  - Carry out iterative algorithm testing
  - Carry out imaging testing

Acknowledgements

- Thank you to:
  - Prof. Mehran Mesbahi
  - Prof. James Hermanson
  - Prof. Eric Hurlen
  - Andrew Jensen
  - The A&A Department

Questions?