Satellite Attitude Identification using Star Pattern Recognition

by

Philip Sigillito
Department of Computer Science
College of Information Science and Technology
University of Nebraska at Omaha
Omaha, NE 68182-0500
psigillito@unomaha.edu

Wednesday, December 7, 2022

Abstract

A satellite's attitude is its orientation, the direction it is facing, and relative rotation to another object. For operating satellites and other spacecraft, it is critical to control and know with precision the attitude of the craft. The most effective algorithms for determining the attitude of a craft use pattern recognition to look at a part of the night sky and identify a star based on its surrounding star pattern. Modern advancements have focused on increasing star identification accuracy, decreasing the amount of comparison data, decreasing time to process, and increasing tolerance to image noise. This project explores the implementation of a modern novel algorithm proposed in [Mehta and Chen 2017]. It uses the relative angles and distances of the stars surrounding the target star to determine its identity. Their paper illustrates this algorithm outperforming traditional algorithms in accuracy but is limited in testing how image noise impacts performance. Image noise can include fake stars such as other satellites, star brightness variations, and measurement inaccuracies from the sensor. It is unclear at what point this algorithm begins to fail due to noise and which types of noise are most detrimental. Their paper also states that spearman correlation was used because it is more effective than pearson correlation but does not quantify the performance difference. This project will quantifiably compare the difference between using spearman and pearson correlation.

Table of Contents

I. Introduction

I.1 Problem:

Attitude information is critical for space travel and unmanned satellites. Many scientific projects in space require high attitude accuracy to make valid measurements. Many satellites include gyroscopes, sun sensors, and other sensors that can be used to track attitude but these approaches are less effective and accurate than star pattern recognition. Star pattern recognition is identifying a star from the pattern of stars that surround it. Star pattern recognition systems generally consist of two parts: a simple camera or sensor that takes an image of a portion of the night sky and a star pattern database to compare against. The problem consists of designing an algorithm that is most accurate and fast in extracting features from the image and comparing them with star patterns in the database to correctly identify the star.

I.2 Motivations:

Although satellites have been able to use star patterns for attitude identification for decades, new approaches continue to be implemented and improved upon. Improvements have included reducing the size of the database, increasing accuracy in measurement, confidence in identification, and reduced compute cost and runtime.

I.3 Significance:

This paper explores the implementation of a modern novel algorithm proposed in [Mehta and Chen 2017]. Its algorithm outperformed previous algorithms in accuracy but was limited in its measurement of noise impacting performance. It is unclear at what threshold [Mehta and Chen 2017]'s algorithm begins to fail due to noise and which types of noise are most detrimental. This paper quantifiably compares the difference between using spearman and pearson correlation when selecting a matching pattern.

I.4 Challenges

Despite the variety of star pattern recognition algorithms, there are common challenges that all these algorithms face. One such issue is magnitude uncertainty (star brightness). Star brightness can fluctuate causing algorithms to miss identify stars based on their magnitude. This is especially apparent if an algorithm filters star data with a magnitude threshold [Mehta and Chen 2017]. Obstructions in the image captured or fake stars such as other satellites or planets can cause additional data points that are erroneous. Furthermore, as pointed out in [Liebe 1993], the sensor's accuracy can be slightly off causing the measured distances between stars to be slightly different than their actual distance.

Another common challenge in algorithms that use grids to divide the night sky is that the camera can only capture a small field of view. It is uncertain if the field of view will be inside a single grid or if it will overlap with multiple grids.[Ding et. al. 2019] Additionally the field of view could be rotated at an angle different from the grid pattern and that must be accounted for.

I.5 Objectives

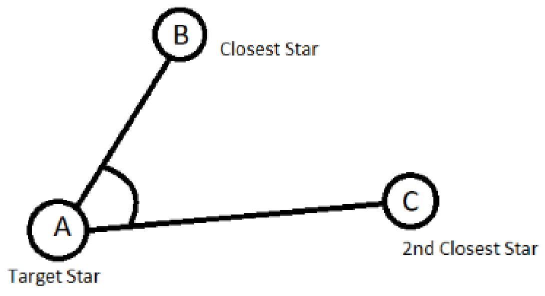
The objective of this project is to recreate the algorithm presented in [Mehta and Chen 2017]. That paper showed results of the algorithm performing significantly better than the older traditional star identification algorithms. It also stated it used the spearman correlation for scoring the similarity of the observed features against the SPD entries. Although this intuitively makes sense it did not provide a quantitative comparison with using a pearson correlation. This paper will extend the testing of image noise: fake stars, magnitude uncertainty, and positional displacement. This paper aims to identify the threshold at which image noise makes the algorithm unuseful.. This paper will also quantitatively compare the scoring between the two correlation coefficients.

II. Overview

II.1. History of the problem

Star pattern recognition has existed for decades as a solution to determining the attitude of satellites and spacecraft. Star pattern recognition algorithms can be divided into two general categories: geometric and pattern. Geometric algorithms generally use the areas formed by star clusters and angular distances between stars, and angles formed by star triplets. An example of this is the triangle approach described in [Liebe, 1992]. This algorithm extracts a triangle feature for each star that consists of the target star and the two closest stars and the angle between them.

All triplets that are within the captured image are then compared against the entries in the database to determine candidates. The algorithm then creates an expected constellation based on the distances of the stars relative to each other in the database and this constellation image is compared against the image constellation. Because measurements will not be exact, tolerances for discrepancy are included in the constellation comparison.



Liebe triangle feature created by three stars

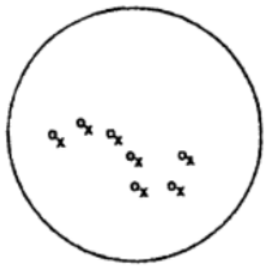


Figure from [Liebe 1993]
Shows generated constellation
compared with actual constellation

The planar algorithm is similar to Liebe’s algorithm except that it also includes the camera as a point. It measures the angle between three stars in the field of view as well as the angle to the camera. This approach provides greater accuracy but introduces more complexity. [Zhou and Ye 2015].The pyramid algorithm is a further advancement introduced in [Mortari et. al. 2004]. This algorithm uses four stars where three stars are connected and their angles measured to form the base of the pyramid. A fourth star is then considered the top point of the pyramid connecting to the three base starsand planar triangle.

The second family of algorithms are pattern algorithms. These algorithms generally involve dividing the celestial sphere into a grid of rectangles and then further dividing each rectangle into a grid where each grid is labeled either one or zero depending on whether it contains a star. One example of this [Jiang et. al. 2007] which utilizes the Knuth-Morris-Pratt pattern recognition algorithm to process the one dimensional vectors generated from images.

While all algorithms covered above perform well in ideal situations, their confidence in identifying stars decreases with the introduction of noise in the image. This noise can include magnitude uncertainty, distance uncertainty, position deviation, and false stars. Additionally, geometric algorithms face additional difficulty when stars are densely clustered together as angular distances are harder to calculate.

Some issues that would be expected to cause serious issues are insignificant or can consistently be adjusted for. Aberration as a result of the satellite's velocity, earth's velocity, and solar system's velocity can be consistently adjusted for as shown in [Liebe 1993]. [Liebe 1993] also states that the effect of parallax from the earth orbiting the sun is insignificant for calculating attitude.

Implementations also face a challenge in managing the size of the star pattern database and the speed of searching it for matches. Over the decades, the size of the databases used has been compressed [Ding et. al. 2019]. Additionally, pattern recognition implementations face an issue of handling images that overlap with multiple celestial sphere subsections [Jiang et. al. 2007].

II.2. State of the art

The state of the art algorithms are generally iterative improvements on the traditional algorithms covered above. For example, [Jiang et. al. 2007] uses the brightest star in its grid pattern as main guiding stars. The sphere is then divided into squares whose centers are the main guiding stars. When constructing the star pattern database, the brightest star for every step across the celestial sphere is selected as well as the second brightest star which is within a certain range of the primary star. The square is then rotated so that the line between the brightest and second brightest stars are in parallel with the bottom of the square. This solves the issue of the camera image being rotated differently than the pattern in the database. Upon capture, the image can be rotated similar to how the database image is rotated.

[Hingston 2022] uses OpenCV(open source computer vision) to train a HAAR cascade classifier for each constellation that should be identified. This approach requires training each constellation classifier separately. Academic papers proposing using HAAR cascades for satellite attitude were not found. Although this is interesting, applying a HAAR cascade to identify stars has already been explored by [Hingston 2022] and new work would only involve additional classifiers. Additionally this process is very time consuming for a semester project.

The approach in [Mehta and Chen 2017] is a modern geometric approach that this paper aims to recreate. This algorithm is appealing because of its strong performance and elegant simplicity. This algorithm identifies the star at the center of the image. It then extracts a feature set by measuring the angle and distance of each star surrounding it.

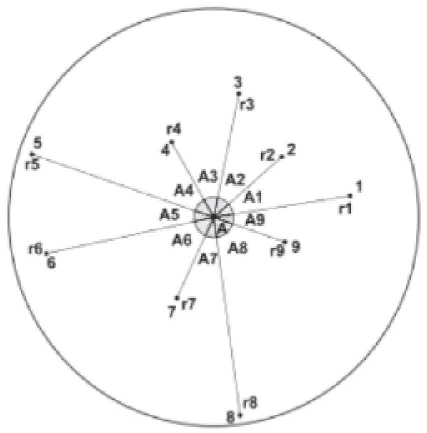


Figure from [Mehta and Chen 2017]
Central star A and the surrounding star distances and angle measurements

This feature set of distances and angles is further reduced into a single one dimensional array. The number of elements in the array correspond to how many sectors the 360 degree surrounding of the star is divided. For example, if 36 elements are used, each element would represent an arc range of 10 degrees. The value of each element corresponds to the distance value of the star in that angle sector. If multiple stars exist inside a range, the average of their distances is taken. For the star pattern database, each star has an entry and each entry consists of the same number of bins as those generated earlier. The spearman correlation coefficient is then used with the stored patterns and the observed pattern to identify the star. In [Mehta and Chen 2017], they found that 512 angle bins were sufficiently effective.

III. Techniques

III.1. Principles, Concepts, and Theoretical Foundations of the research prob

This paper is an extension of the mathematical concepts and principles found in [Mehta and Chen 2017]. See section III.2.1 below for more information. This paper will use the pearson correlation coefficient in addition to the spearman correlation coefficient. The equation for pearson correlation is the following formula where S_x and S_y are standard deviations:

$$r_s = \frac{\sum(X - mean_x)(Y - mean_y)}{(N-1)S_x S_y}$$

III.2. Techniques that have been used by other researchers for the research problem

III.2.1 [Mehta and Chen 2017]

The techniques used by [Mehta and Chen 2017] rely on the extraction of star angles and distances. These values can be calculated using the equations listed below:

$$r_n = \sqrt[2]{(x_n - x_c)^2 + (y_n - y_c)^2}$$

$$A_n = tan^{-1}\frac{(y_n - y_c)}{(x_n - x_c)}$$

In the above equations r_n is the distance between the central star and a neighboring star. x_c and y_c are coordinates of the center star and x_n and y_n are the coordinates of the neighboring star. A_n is the angle formed between the two. Determining the bin a star should belong to is trivial:

$bin_{SPD} = round(\frac{Angle}{180/N})$ where N = number of sample bins.

The spearman correlation coefficient is calculated by using the image and each record in the database. The image with the highest correlation is considered the matching star. The formula for the spearman correlation is:

$$r_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

In the above formula $d_i = R(x_i) - R(y_i)$

III.2.2 [Jiang et. al. 2016]

This paper's technique employs a bitstream of the image and candidate images from the database to identify the star. It does this by selecting a pair of the brightest and second brightest stars to rotate the image and compare against pre-rotated images in the database.

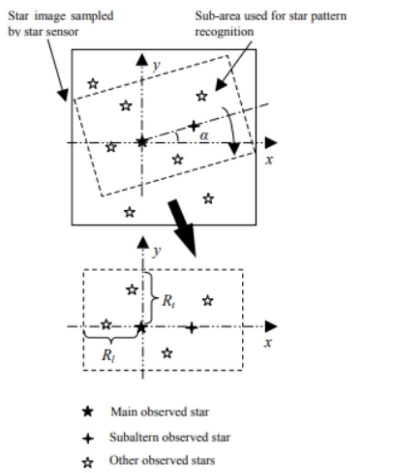

Star image sampled by star sensor
Sub-area used for star pattern recognition
y
x
α
R_t
R_l
Main observed star
Subaltern observed star
Other observed stars

Figure from [Jiang et. al. 2016]
Showing FOV rotation on two brightest star

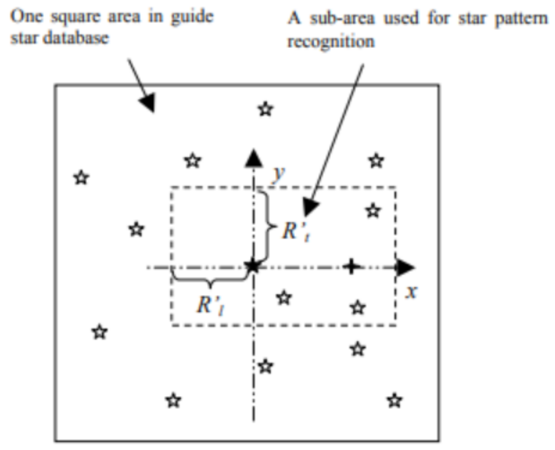

One square area in guide star database
A sub-area used for star pattern recognition
y
x
R'_t
R'_l

Figure from [Jiang et. al. 2016]
Showing comparison of rotated FOV with grid

This technique also includes cleaning the bit stream to make it more readable. For example it only allows one observed star per predefined area to reduce noise. The bit stream of the image and the database image are combined with a bitwise AND. The candidate with the highest number of ones after the bitwise AND is considered the match.

III.2.3 [Ding et. al. 2019]
This paper's technique is very similar to [Jiang et. al. 2016]. It divides the celestial sphere into grids, selects a star to profile, centers the FOV on the star and rotates the image so that the line drawn to the closest star is parallel to the grid's bottom. It then divides the image into grids to label each grid with a one or zero. It iteratively does this for each star so that each star has a one dimensional feature vector that can be compared against. Using this technique, they achieved a recognition accuracy of 87.64% without any noise.

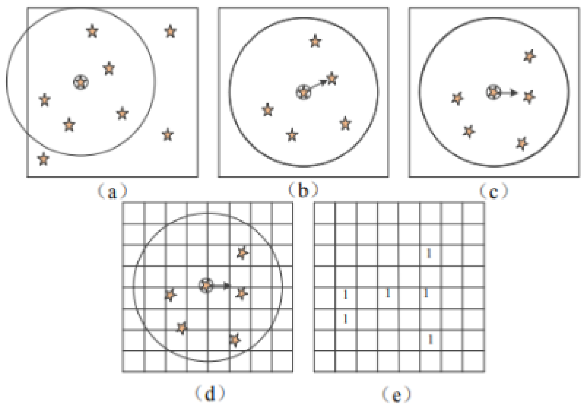


Figure from [Ding et. al. 2019]
shows star selection, centering on star,
rotating image, forming grid, and converting
into 1 values

III.2.4 [Liebe 1993]
The techniques used in [Liebe 1993] extract the angular distance and spherical angles between the target star and closest two stars as well as their magnitude. These measurements are stored as a star triplet explained further in section II.1. The star triplet is transformed into an array of three simplified integer values by dividing by the maximum deviation for the two distances by the maximum arc deviation of 0.1 degrees and the angle interval by an arbitrary 5.0 degrees. For example, in the paper the two distances of 5.596 and 8.191 are divided by 0.1 to become 55 and 81. The angle 164.204 degrees is divided by 5.0 and rounded to get 32. This feature transformation results in a much simpler feature value of [55, 81, 32].

Liebes' technique allows for intervals of error to be stored in the database. For example, although the measured value for the star is [55,81,32] it will also have entries for values within the error interval such as [54,81,32], [56,81,32], etc. Under this technique, the typical star has 180 entries. After the triplets in the field of view have been generated, it tries to generate the expected constellation of the candidate stars based on maximizing the least distance fit between stars. The best matching constellation to the generated constellation is selected.

Liebe also includes a technique for measuring how accurate the calculated attitude can be. The following equation was used for this measurement:

$$accuracy\ of\ calculated\ axis\ = \frac{position\ accuracy\ of\ single\ star\ in\ image}{\sqrt{number\ of\ stars}}$$

Liebe found that accuracy increased as the number of stars in the field of view increased as shown in the figure below:

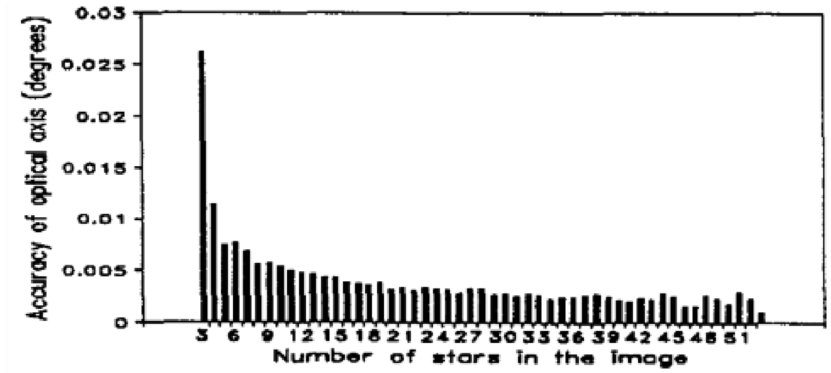


Figure from [Liebe 1993]
Shows accuracy of axis measurement as number of stars increase

III.3. Relevant technologies that would be useful to this research

It is anticipated that all logic for this project can be written with python scripts and libraries. The libraries will be used for generating images and extracting feature sets from images. The dataset that will be used for this project will be the SKY2000 Guide Star Catalog maintained by NASA. This dataset includes 300,000 of the brightest stars and tools to access the database.

IV. Approach(s)

IV.1. Methodologies I applied in this research

My methodology focused on implementing the geometric algorithm described in [Mehta and Chen 2017]. My methodology consists of implementing their feature extraction, database construction, and feature comparison using python scripts. I used the SKY2000 Guide star catalog as the dataset to build and run experiments.

Python scripts were used to generate a two dimensional image from an entry in the database. The image is then rotated correctly. Depending on the test, stars are either added, removed, or the entire image is rotated a set number of degrees incorrectly. The image then has its features extracted and processed for a match against the star entries in the database. My methodology includes listing the two runner-up star candidates for matching and their coefficient scores to illustrate the magnitude of differentiation between true and false matches.

This process of generating an image, altering it, extracting its one dimensional vector and trying to match it against the database is automatically scripted using a bash script to call the relevant python scripts. In my testing, I used the first 100 stars in the SKY2000 database per scenario and recorded the corresponding stars with the highest Pearson and Spearman coefficients as the selected matches. Due to processing slow-down from using a csv file as my star database, I loaded the first 50,000 stars in the catalog.

IV.2. Techniques I used to solve the problem

The novel techniques of this project consist of generating images, adding noise, and extracting a new feature from that result. Additionally I compared the Spearman and Pearson coefficient vectors to quantify the impact of each type of noise: fake stars, missing stars, and incorrect rotation.

IV.3. Processes I engaged in this research

My process consisted of python and bash scripting to manipulate and edit the sky2000 data set. Additional scripts were used to perform feature extraction from the image, extract features for each star in the database, and perform the comparison. Additionally scripts were used to generate the star image and extract features from the star image. My personal computer was able to handle a reasonably large dataset of 50,000 stars in the database representation.

Experiments involved generating and identifying the first 100 images in the SKY2000 dataset. These stars were compared against the 50,000 star entries in our database representation.The following test conditions were tested:

- Ideal conditions (no image noise)
- False stars added to the images at random positions:
  - Adding 1, 2, 4, and 8 stars were tested.
- Removing random stars (excluding the target and brightest star)
  - Removing 1, 2, 4, and 8 stars were tested.
- Rotating the image incorrectly by 1, 2, 5, 8, and 10 degrees.

The Pearson and Spearman coefficient values were determined for each image in comparison to each entry in the database. The entry with the highest correlation coefficient was selected as the matching star. I recorded the top three matching entries for each image tested. These test scenarios extend the highest listed values for noise testing in [Mehta and Chen 2017] which were:

- One pixel of star position deviation
- 0.4 magnitude uncertainty
- Five false stars

IV.4. How the results (outcomes) of the research are demonstrated?

Each star has a unique id that was known during testing. This makes it trivial to confirm the accuracy of the algorithm. Additionally, the runner up candidates and their coefficient values were measured to see how differentiated the correct selection is from the most similar wrong selections or if the correct selection is the second or third candidate.

IV.5. How the results (outcomes) of the research were evaluated, analyzed, and compared with previous research?

[Mehta and Chen 2017] include results that show how the algorithm performs as the noise from positional deviation, magnitude uncertainty, and false stars increases. Previous research stops at relatively simplistic noise so this project increases the noise to see how it impacts identification accuracy. The results of this project extend the results already reported in [Mehta and Chen 2017] and depict the points at which noise causes selection using the Pearson and Spearman correlation coefficients to degrade.

IV.6. Facilities and supplies used for this research

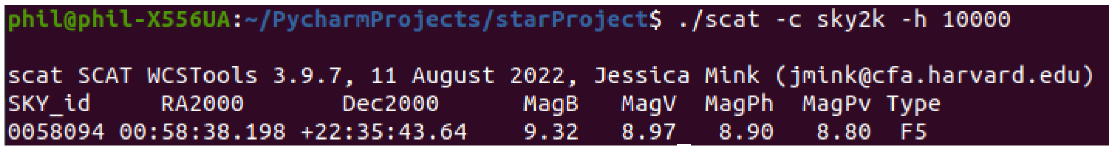
No special facilities or supplies were needed. My personal laptop was able to run a test scenario for the first 100 stars in approximately 30 minutes. This included getting the star coordinates using scat, generating an image, altering the image if noise required, extracting the feature set from the image and determining the correlation coefficients for each entry in the database. 50,000 entries were included in the database. If additional resources were available and the database was represented in memory: parallelized comparison to the database could have improved performance. Although this would be a noticeable improvement, using a csv file was acceptable for the scope of this project.

V. Work Conducted

V.1. Tasks perform in this research

Task 1

The first task consisted of researching how to use the sky2000 catalog and its associated query tools. The sky2000 catalog was originally maintained by NASA and consists of approximately 300,000 of the brightest stars and their positions. This catalog is readily available through a number of sources. I decided to use the WCSTools package for querying against the catalog. The WCSTools package is used in the papers I researched and appears to be the of the most common tools for astronomical research. I primarily used the scat command line tool which can be used to query a binary file representation of the catalog. Scat allows various types of queries. The first query I used was to return a star's location and magnitude based on its id. This query is depicted below:


```
phil@phil-X556UA:~/PycharmProjects/starProject$ ./scat -c sky2k -h 10000

scat SCAT WCSTools 3.9.7, 11 August 2022, Jessica Mink (jmink@cfa.harvard.edu)
SKY_id      RA2000        Dec2000       MagB    MagV   MagPh   MagPv Type
0058094 00:58:38.198 +22:35:43.64     9.32    8.97    8.90    8.80  F5
```

The second query I used was to search for all stars within a selected distance around a position. Understanding how the tools work and deciding on the queries to use took approximately one week. I spent the next week reading about astronomy terms and how astronomical positioning works. This included learning some key terms necessary for properly understanding query results. Some of the key terms necessary to understand this project are outlined below:

- **Celestial Sphere**: Stars are located on the celestial sphere which has an infinite radius and earth at its center. Stars are located in three dimensional space but are so extremely far away that we can think of them as positioned on a sphere surrounding the earth. This simplifies describing a star's location. A star's precise location on the sphere can be given using its right ascension and declination.

- **Right Ascension**: This is the longitude position of a star in the night sky. Its value is expressed in hours:minutes:seconds. This format is based on the rotation speed of the earth. For example, one hour equals approximately 15 degrees because the earth rotates approximately 15 degrees per hour. To simplify calculations, I converted this format to degrees using the following equation:
$$Degrees = (Hours + (Minutes/60) + (Seconds/3600)) * 15$$

- **Declination:** This is the latitude position of a star. This value can be negative or positive depending on if it is above or below the equator. It is expressed in degrees:minutes:seconds. This format is slightly different from right ascension and I used the following equation to convert to degrees:
$$Degrees = Degrees + ((Minutes/60) + (Seconds/3600)) * 15$$

- **Zenith**: Point on the celestial sphere directly overhead the viewer.

- **Meridian:** Imaginary arc passing through celestial poles and through the zenith.

- **J2000:** This is the dataset the sky2000 catalog is based on. J2000 corresponds to the star positions taken on January 1st 2000.

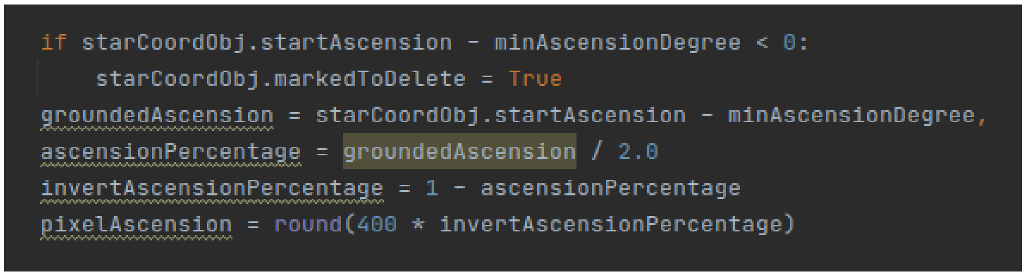
Task 2:

This task required writing a python script that would generate a star image using positional data queried from scat. I used opencv to draw the image. This task took approximately two weeks and required the following steps:

1. Query a star's position given its id using scat.
2. Search that star's surrounding area using scat.
3. Parse scat's return values to extract each surrounding star's right ascension, declination, and brightness.

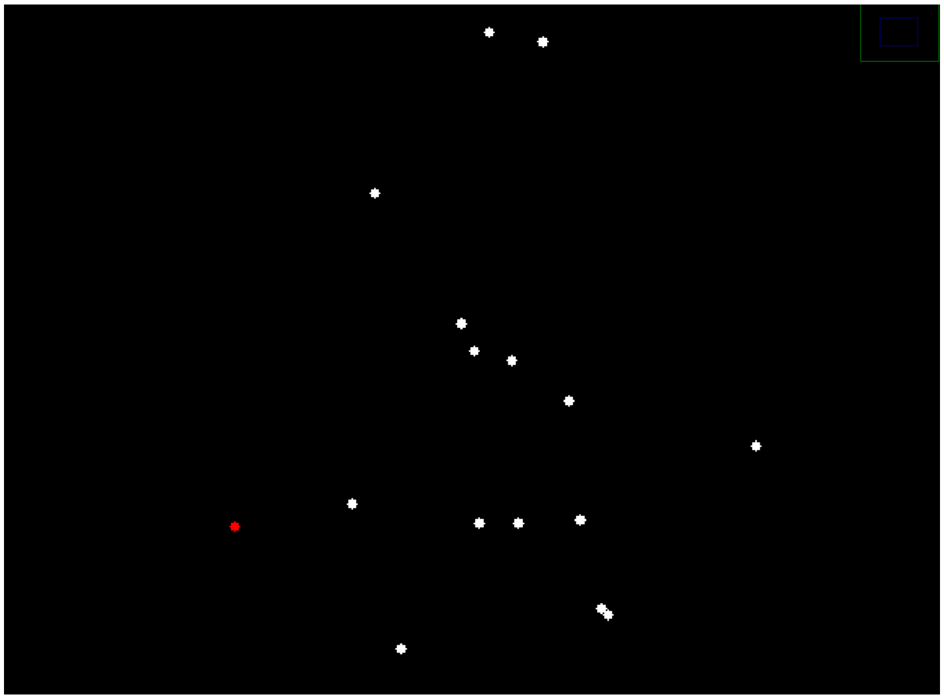
4. Convert each star's celestial sphere position to a position in our 900x1200 pixel image.
5. Identify the brightest star and rotate the image so that the brightest star is horizontally aligned to the right of the target star.
6. Draw the image using opencv to draw a white circle (red for brightest star) for each star in the image and save the image to disk.

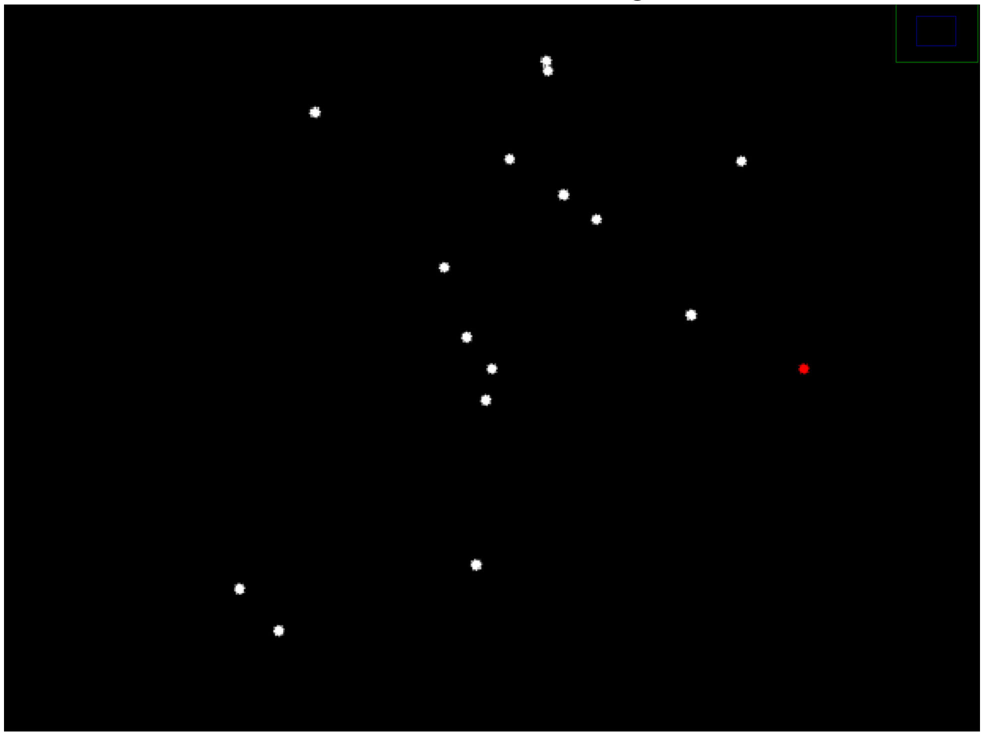
The most difficult aspect of this step was converting each position's ascension and declination to a degree value and then converting that into a pixel position. I achieved this by converting the star's position to a percentage value between the search area's minimum and maximum search area. For example, if the search range is between 9 and 11 and the star's x-axis location is 10, then the x-axis position of the star is 50 percent. This generally works but I had to add additional logic for the edge cases where the search area crosses the 'threshold' between 0 degrees and 360 degrees. For example, if we are searching 1 degree in each direction around degree 0 the max value is 1 and the min value is 359. This required identifying when we were crossing that threshold and adding 360 to all values so that higher degree values were always numerically greater. Below is a segment of the script that converts the ascension value into degrees.



```
if starCoordObj.startAscension - minAscensionDegree < 0:
    starCoordObj.markedToDelete = True
groundedAscension = starCoordObj.startAscension - minAscensionDegree,
ascensionPercentage = groundedAscension / 2.0
invertAscensionPercentage = 1 - ascensionPercentage
pixelAscension = round(400 * invertAscensionPercentage)
```

I also found that searching around some stars resulted in stars being returned that were outside the degree search radius. I recreated this several times using the scat tool manually and comparing my results in another application called Stellarium to confirm I was not making a mistake. To fix this, I simply added a distance check to cull stars returned outside the search radius. Pictured below is the initial pattern with the brightest star colored red and the same stars rotated so that the brightest star and center target star are horizontally aligned.


Stars drawn as retrieved from scat. Brightest star is red.


Stars after rotating brightest star to be directly right of target star

Task 3:

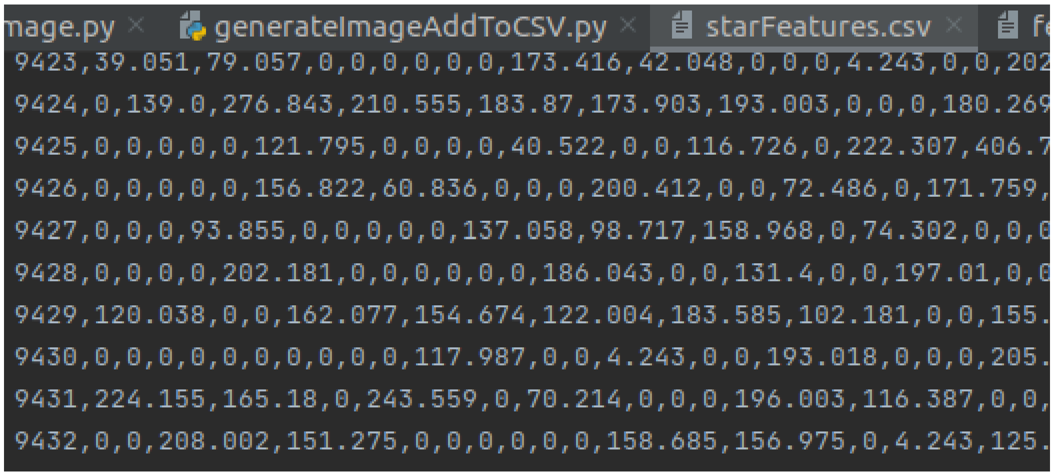
This step required writing a script to take a star image, extract the star locations from the image, determine the distance and angle of each star around it, and assign those distance values into bins depending on their angle relative to the target star. For now, I set the number of bins to

36 (10 degrees per bin). Determining each star's angle only required finding the angle formed by the star, the central target star, and an imaginary point directly to the right of the center star. If multiple stars are stored in the same bin, their average is saved.

I tried three different approaches for identifying stars in the image. The first approach used Harris Corner Detection to identify star locations. This resulted in multiple corners being identified for each star. As a result, some stars were being counted correctly but some were being counted multiple times. I also tried the Hough Circle Transform using opencv. This approach only identified the stars if they were large enough to be considered a circle. If the stars were enlarged enough to be detected, they would often overlap and result in stars not being counted. I eventually found that using contour detection was the most effective. Contour detecting finds the borders between objects. This was very effective for white stars on a black background. This is not a robust solution as stars that overlap are counted as one but this is tolerable as it impacts both our test images and the images that were used to generate the csv database.

Task 4:

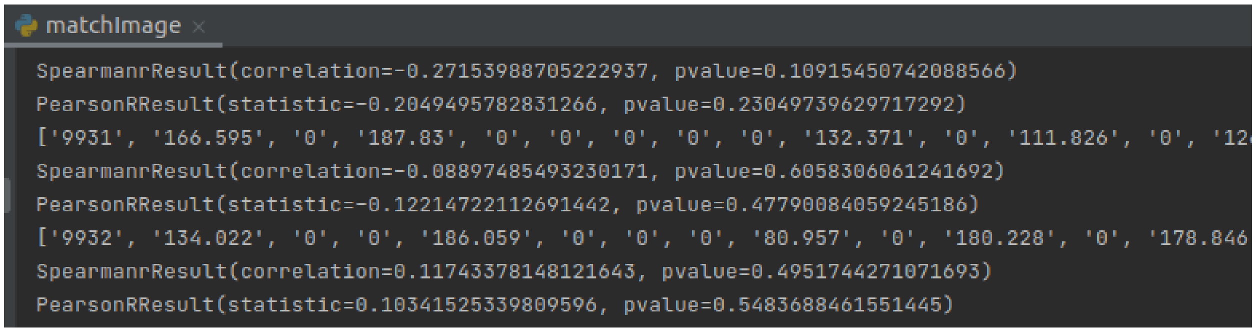
This task consists of writing a bash script which calculates the features for each star and stores them in a csv file. The script loops through a range of star ids, calculates their feature set, and saves that feature set as a line in the csv file. This file will serve as a simple database to compare against. Each row consists of the star id followed by its bin values. Processing each star took approximately 1 second. Before final testing I will run this script and generate 100,000 entries. Pictured below are csv file entries for star ids 9423 to 9432.



```
9423,39.051,79.057,0,0,0,0,0,0,173.416,42.048,0,0,0,4.243,0,0,202
9424,0,139.0,276.843,210.555,183.87,173.903,193.003,0,0,0,180.269
9425,0,0,0,0,0,121.795,0,0,0,0,40.522,0,0,116.726,0,222.307,406.7
9426,0,0,0,0,0,156.822,60.836,0,0,0,200.412,0,0,72.486,0,171.759,
9427,0,0,0,93.855,0,0,0,0,0,137.058,98.717,158.968,0,74.302,0,0,0
9428,0,0,0,0,202.181,0,0,0,0,0,0,186.043,0,0,131.4,0,0,197.01,0,0
9429,120.038,0,0,162.077,154.674,122.004,183.585,102.181,0,0,155.
9430,0,0,0,0,0,0,0,0,0,0,117.987,0,0,4.243,0,0,193.018,0,0,0,205.
9431,224.155,165.18,0,243.559,0,70.214,0,0,0,196.003,116.387,0,0,
9432,0,0,208.002,151.275,0,0,0,0,0,0,158.685,156.975,0,4.243,125.
```

Task 5:

After generating the database, I wanted to be able to run every step of the actual testing. This required writing another python script that takes an image as input, decomposes it into its feature list and compares the feature list to the entries in the database. It uses the spearman and pearson coefficient values to select the most similar value. Initial testing showed the coefficients to select the correct star. Querying against the database took approximately two seconds with most of the time spent printing each comparison. I intend to speed this up but since my testing is not focused on execution speed, it is not a primary point of concern. Pictured below is the console output for our image being compared against stars 9931 and 9932 and listing their spearman and pearson coefficients.



```
SpearmanrResult(correlation=-0.27153988705222937, pvalue=0.10915450742088566)
PearsonRResult(statistic=-0.2049495782831266, pvalue=0.23049739629717292)
['9931', '166.595', '0', '187.83', '0', '0', '0', '0', '0', '132.371', '0', '111.826', '0', '12
SpearmanrResult(correlation=-0.08897485493230171, pvalue=0.6058306061241692)
PearsonRResult(statistic=-0.12214722112691442, pvalue=0.47790084059245186)
['9932', '134.022', '0', '0', '186.059', '0', '0', '0', '80.957', '0', '180.228', '0', '178.846
SpearmanrResult(correlation=0.11743378148121643, pvalue=0.4951744271071693)
PearsonRResult(statistic=0.10341525339809596, pvalue=0.5483688461551445)
```

Task 6:

This step consisted of writing a bash script that does the entire end-to-end process: querying scat, image generation and rotation, noise addition, feature extraction, and database comparison for the first 100 stars in the catalog. During this task I also ran the tests for the ideal case of no noise in the images.

Task 7:

This step consisted of updating my scripts to introduce noise. Noise includes adding stars, removing stars, and altering the image rotation. Adding invalid stars involved selecting a random pixel location in the image and drawing the star. Removing a star required updating the script to randomly select a star and remove it from the list before rendering the image. Importantly, removing stars would not select the target star or brightest star. Removing the brightest star would also cause the image to be rotated incorrectly and does not reflect the type of noise to be tested. Rotating the image required only calling opencv's rotate image function again with the degree change we desired. After completing this work, I ran the test scenarios for image noise.

Task 8:

This step consisted of collating and analyzing the results, making charts, and writing the final report.

V.2. Schedule, timeline, and milestones

Order	**Dates**	**Task/Activity**	**Prerequisites**	**Results Obtained**
1	Week of Sept. 12th and 19th	Learn sky2000 catalog and astronomy concepts/terminology.	Project proposal	Understand how to consume and convert values returned from the scat tool.

2	Week of Sept. 19	Write a script that uses scat tool to get star locations, determine pixel positions, apply rotation transformation, and generate a png image.	Understand how to use query tools and results.	I can now generate an image for testing.
3	Week of Sept. 26	Write a script that extracts stars from an image, calculates distances and angles of each star from center, and generates a feature set.	Ability to generate test images.	Now that we can generate feature sets, We can populate the database.
4	Week of Oct. 3	Write a script that generates csv database representation for each star.	Scripts from previous tasks to generate image, decompose into feature set and store in file.	I can now build a star id database.
5	Week of Oct. 10 to Oct. 24th	Script that expands feature extraction from step 3 to also compare feature set against the values in csv file database using Spearman and Pearson correlation coefficient.	Database to compare against and test images generated by previous tasks to use in testing.	I can now execute all steps of a simple star identification using our generated csv file database.
6	Weeks of Oct. 31, Nov. 7	Write bash script to run end-to-end image generation, feature extraction, and database comparison for the first 100 stars. Also update scripts to output comparison results to text file. I determined the 50,000 star database is a reasonable size.	All scripts required for each step of the end-to-end process.	We now have a process to run each test scenario
7	Week of Nov 14 and Nov 21	Update scripts to introduce noise(add/remove stars and rotate image incorrectly). Ran noise tests.	End-to-end testing pipeline.	Noise test results complete.
8	Week of NOv 28th and Dec 5	Collated results from tests into tables and graphs. Wrote final paper sections.	Results from testing.	Project complete.

VI. Results and Analyses

VI.1. Results

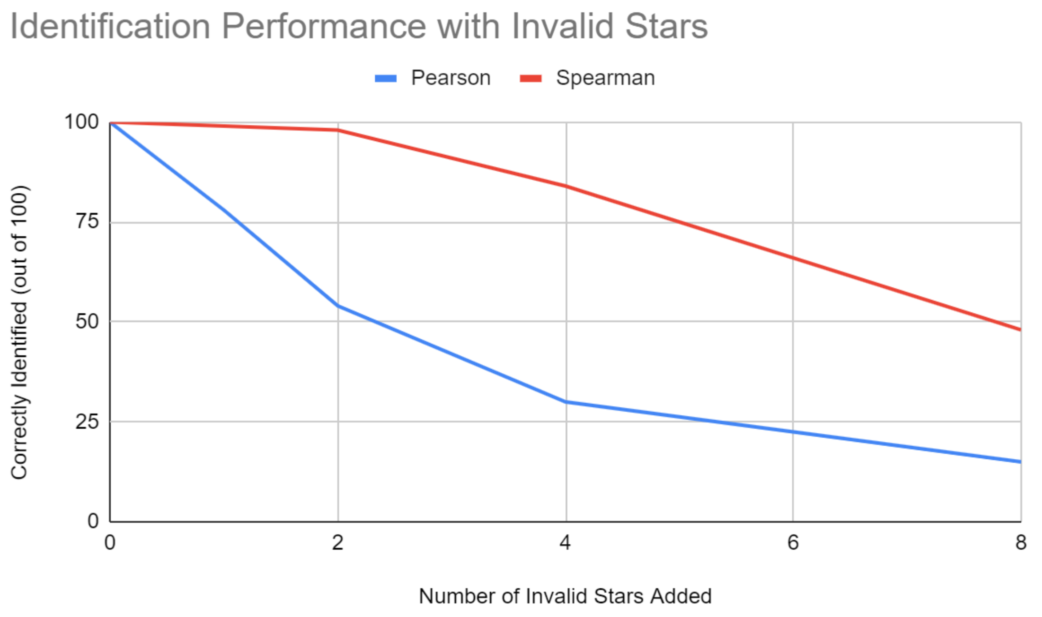
Tests for each scenario used the first 100 stars in the sky2000 database. The highest three matching database entries by Spearman and Pearson correlation coefficients were recorded into a text file. Each testing batch used the same 100 stars; therefore, confirming whether the algorithm identified the star correctly was trivial. The database contained the first 50,000 stars in the sky2000 catalog.

All tests used 36 bins (10 degrees per bin). Results were output to a text file and collated in Google Sheets. Spearman and Pearson coefficient correlation selection was performed on the same stars between all tests; therefore, there is no probability of one test using easier stars to classify than the other. The four test scenarios are outlined below:

- False stars added to the images at random positions:
    - Adding 1, 2, 4, and 8 stars.
- Removing random stars (excluding the target and brightest star)
    - Removing 1, 2, and 4. Some images started with less than eight stars; therefore, testing stopped at 4.
- Rotating the image incorrectly by 1, 2, 5, 8, and 10 degrees.

The case of no noise is not displayed as both coefficients displayed perfect identification. This is expected and makes sense as the stars are simply transformed into an image and extracted out again so there is no distortion and minimal loss of data.

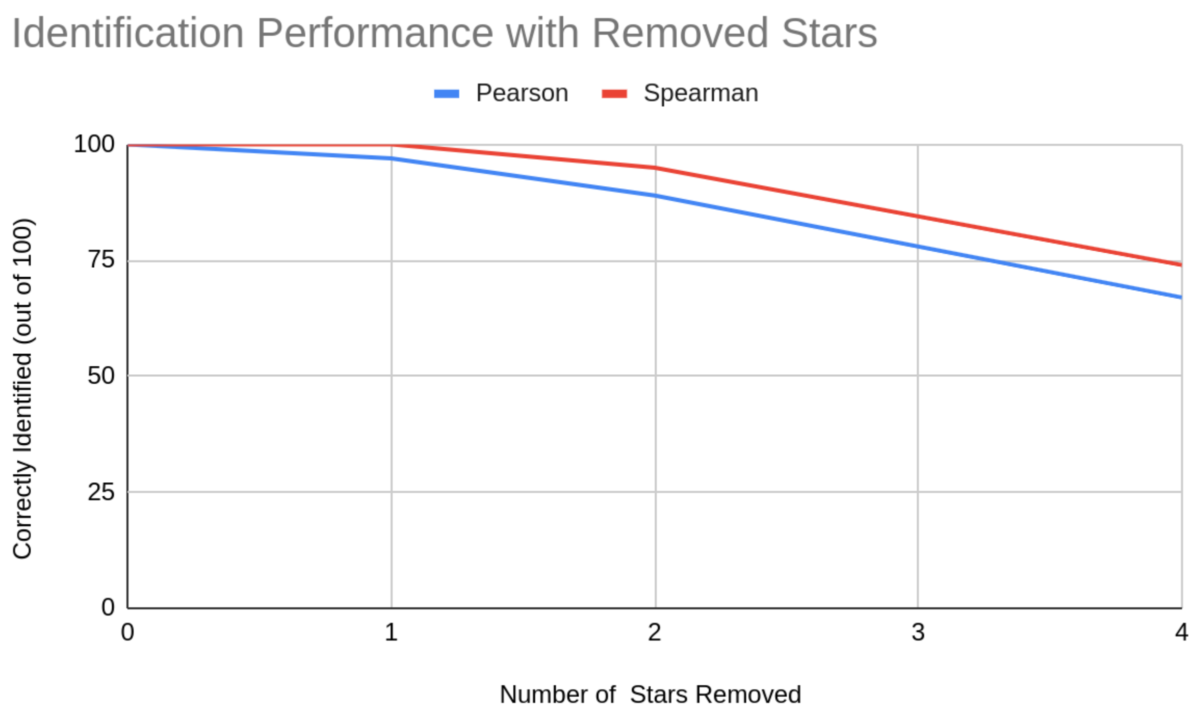
Adding False Stars:



Stars Identified Correctly		
Invalid Star Count	**Pearson**	**Spearman**
0	100	100
1	78	99
2	54	98
4	30	84
8	15	48

It is interesting to see that adding false stars noticeably decreases the accuracy of using both coefficients. As expected, the Spearman coefficient outperforms the Pearson coefficient. The difference between the two coefficients is notable as adding a single star to the image causes Spearman to only misidentify one star while Pearson misidentifies 22 stars. Notably, Spearman still has approximately 50 percent accuracy when eight stars are added to the scene. Although possible, it seems unlikely eight foreign objects would be detected in an image captured by a satellite.

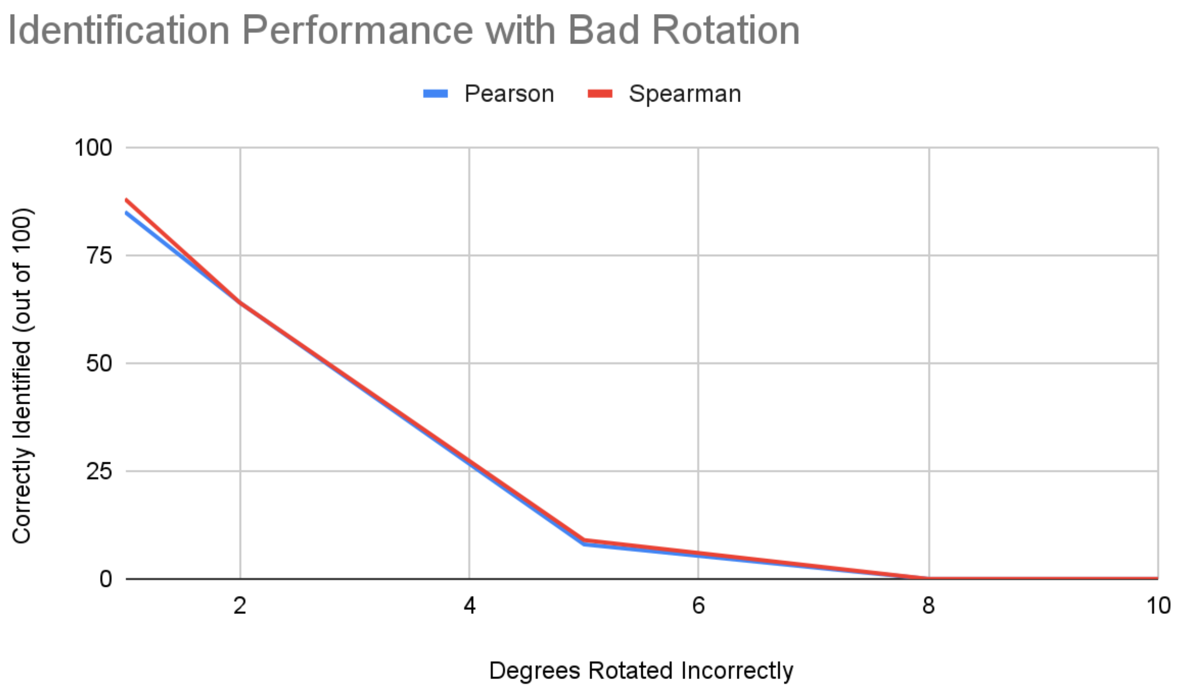
Removing Stars:



Stars Identified Correctly		
Removed Stars	Pearson	Spearman
0	100	100
1	97	100
2	89	95
4	67	74

Removing stars from the images also degrades performance but appears to degrade slower than adding the same number of stars. Removing one star caused the Spearman selection to only mislabel three stars as opposed to 22 when a star was added. Notably, testing of removing stars stopped at four stars because removing more stars resulted in some images being blank besides the target star and brightest star.

Image Rotation:

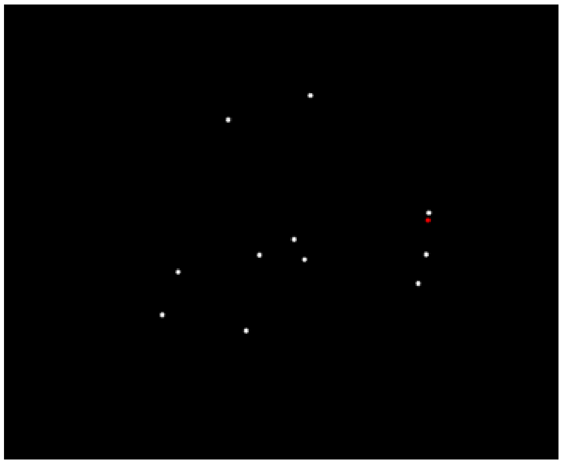
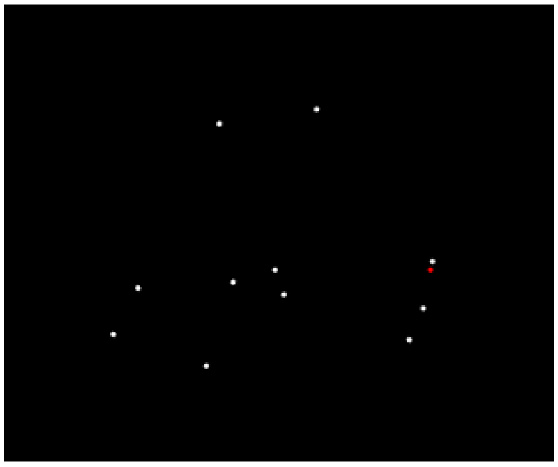


Stars Identified Correctly		
Additional Rotation (Degrees)	Pearson	Spearman
1	64	88
2	64	64
5	8	9
8	0	0
10	0	0

Image rotation proved to be the most detrimental to star identification. This is somewhat surprising as the bins are 10 degrees each. This high drop off was not addressed in [Mehta and Chen 2017] and suggests a potentially significant cause of concern for using this approach. If a follow-up project was performed, the effect of image rotation would be explored further. In some respects this change makes sense as a change in degree rotation impacts all stars in the image. It would also be informative to see how image rotation impacts representations with a different bin count. With 36 bins, I would have expected the impact to be minimal as the stars would generally stay within their same bins and the vector representation would not change.

VI.2. Analysis of the results

The impact of adding and removing stars shows a reasonable dropoff in accuracy. Additionally, the performance of Spearman is consistently better or comparable to using the Pearson coefficient. This is expected as [Mehta and Chen 2017] also stated that Spearman performed better. Adding two stars causes Pearson's accuracy to drop to approximately 50 percent while adding 8 stars results in Spearman dropping to the same approximate accuracy. Removing stars showed a similar trend with both techniques degrading in accuracy but at a slower rate. Interestingly, image rotation caused the greatest dropoff in accuracy. Just rotating two degrees caused both coefficients to dop to 64 percent correctly identified. This is unexpected and should be followed up on in future research. I was suspicious my rotation implementation was not correct but could not find any issues when debugging the code. Results appear correct as depicted below. The left image depicts an unrotated image. The right image is the same as the left rotated eight degrees.

One important consideration is that our experiments removed a set number of stars as opposed to a percentage of the stars in the image. If tests were to be rerun, I would run them adding/removing a percentage of stars relative to the total count of stars in the image.

The average number of stars in the images tested is 19. To further analyze results, the average number of stars in incorrectly identified star images is depicted in the tables below. As shown in the tables, the images with a smaller number of stars are the first to be incorrectly identified when adding and removing stars. This makes sense as the change is more impactful to a smaller sample set. In contrast, there does not appear to be a trend with regard to the number of stars and rotating the image. The number of stars quickly reaches an average of 19 as all stars are misidentified when rotating 8 or more degrees.

Average Number of Stars in Incorrectly Identified Images		
Added Stars	**Pearson**	**Spearman**
1	13.2	6.0
2	16.3	8.5
4	17.7	12.4
8	17.2	16.3

Average Number of Stars in Incorrectly Identified Images		
Removed Stars	**Pearson**	**Spearman**
1	8.0	None
2	9.9	6.4
4	13.3	12.6

Average Number of Stars in Incorrectly Identified Images		
Additional Rotation (Degrees)	**Pearson**	**Spearman**
1	20.3	20.7
2	21.9	21.5
5	18.8	19.0
8	19.0	19.0
10	19.0	19.0

One improvement learned from this testing is that images with more stars are classified better. If possible a satellite should capture several images and use the image that has the most stars in it. Additionally, we found that selection by the Spearman coefficient is consistently better than using the Pearson coefficient. This is expected as [Mehta and Chen 2017] also stated Spearman performed better. In all cases the Pearson coefficient not only performed worse than Spearman but its rate of degradation was significantly faster. We also saw that adding stars caused faster degradation in performance than removing stars.

VI.3. Discussions

VI.3.1. Problems and issues arisen during the research

During testing, no major problems impeded collecting results. The number of entries in the database representation decreased from sky2000's 300k entries to 50k. This is because the database was represented as a CSV file where each entry was listed on a different line. This is an inefficient database representation as each entry must be iterated over using a file stream object. Despite these inefficiencies, using a CSV file was acceptable for the scope of this project. If we were testing runtime performance or for real-world implementation then a more efficient database representation would be utilized.

The other challenge was finding a simple method for identifying stars in the image. I tried three different approaches for identifying stars in the image. The first approach used Harris Corner Detection to identify star locations. This resulted in multiple corners being identified for each star. As a result, some stars were being counted correctly but some were being counted multiple times. I also tried the Hough Circle Transform using opencv. This approach only identified the stars if they were large enough to be considered a circle. If the stars were enlarged enough to be detected, they would often overlap and result in stars not being counted. I eventually found that using contour detection was the most effective. Contour detecting finds the borders between objects. This was very effective for white stars on a black background. This is not a robust solution as stars that overlap are counted as one but this is tolerable as it impacts both our test images and the images that were used to generate the CSV database.

VI.3.2. Limitations and constraints of the research (and results)

It is important to note that this experiment compared the results of identifying white dots on a black background using generated images. This shows that identifying stars using the Spearman correlation coefficient is very accurate in this narrow test setting. This does not necessarily prove that this approach would be effective for satellites in real-life. For example, images in real life are not as clean as our images. Real life images include noise and occlusions. Additionally, this implementation depends on a good algorithm for identifying stars in the image.

VII. Summary

Attitude information is critical for space travel and unmanned satellites. Many scientific projects in space require high attitude accuracy to make valid measurements. Star pattern recognition is an effective technique for attitude determination that works by identifying a star from the pattern of stars that surround it. This system consists of two main parts: a simple camera that takes an image of a section of the night sky and a database to compare against that image. The problem consists of designing an algorithm that is both accurate and fast in extracting features from the image and comparing them with the star patterns in a database for identification.

New approaches continue to be implemented and improved upon. Improvements have included reducing the size of the database, increasing accuracy in measurement, confidence in identification, and reduced compute cost. This paper explores the implementation of a modern novel algorithm proposed in [Mehta and Chen 2017]. This algorithm outperforms previous algorithms in accuracy but was limited in its empirical measurement of how image noise impacts performance. It is unclear at what threshold [Mehta and Chen 2017]'s algorithm begins to fail due to noise and which types of noise are most detrimental. The paper also stated that the Spearman correlation coefficient is more accurate for selecting a match than the Pearson coefficient but did not quantify the difference. This paper quantifiably compares the difference between using the two coefficients when selecting a matching pattern.

The approach in [Mehta and Chen 2017] is a modern geometric approach that this paper recreates and tests. The algorithm operates by generating a one dimensional feature set based on the distances and angles of stars surrounding the target star within a specified distance. The number of elements in the array corresponds to how many sectors the 360 degree surrounding of the star is divided. For example, if 36 elements are used, each element would represent an arc range of 10 degrees. The value of each element corresponds to the distance value of the star in that angle sector. If multiple stars exist inside a range, the average of their distances is taken. For the star pattern database, each star has an entry and each entry consists of the same number of bins as those generated earlier. The spearman correlation coefficient is then used with the stored patterns and the observed pattern to identify the star.

The novel techniques of this project consist of generating images, adding noise, and extracting a new feature from that result. Additionally, I compared the Spearman and Pearson coefficient vectors to quantify the impact of each type of noise: fake stars, missing stars, and incorrect rotation.

My process consisted of python and bash scripting to manipulate and edit star data extracted from the sky2000 star catalog. These scripts encapsulated all functionality required for testing. This functionality included querying against the sky2000 catalog, generating an image from the star data, adding noise to the images, extracting star locations from each image and generating a feature vector, and comparing that feature vector against feature vectors in the database to calculate their Spearman and Pearson coefficients.

Experiments involved generating and identifying the first 100 images in the sky2000 dataset. These stars were compared against the 50,000 star entries in our database representation. For each star, the Spearman and Pearson coefficients were used for selecting a match and each result was recorded. The following scenarios were tested:

- Ideal conditions (no image noise)
- False stars added to the images at random positions:
  - Adding 1, 2, 4, and 8 stars were tested.
- Removing random stars (excluding the target and brightest star)
  - Removing 1, 2, and 4 stars were tested.
- Rotating the image incorrectly by 1, 2, 5, 8, and 10 degrees.

I found that Spearman consistently outperformed or was comparable to Pearson in all scenarios. The impact of adding and removing stars shows a reasonable dropoff in accuracy. Adding two stars caused Pearson's accuracy to drop to approximately 50 percent while adding 8 stars resulted in Spearman dropping to the same approximate accuracy. Removing stars showed a similar trend with both techniques degrading in accuracy but at a slower rate. Interestingly, image rotation caused the greatest dropoff in accuracy. Just rotating two degrees caused both coefficients to drop to 64 percent correctly identified.

These experiments removed a set number of stars as opposed to a percentage of the stars in the image. If tests were to be rerun, I would run them adding and removing a percentage of stars relative to the total count of stars in the image. The average number of stars in the images tested is 19. Profiling the images that failed in the different scenarios, it is clear that images with fewer stars in them failed more often than images with many stars. This makes sense as adding or removing a star from a smaller set of stars is proportionally bigger in impact than a larger set of stars. Future research should test results of accuracy based on the number of stars in the candidate images to see how noise impacts images depending on their star count.

References

C. C. Liebe, "Pattern recognition of star constellations for spacecraft applications," *IEEE Aerospace and Electronic Systems Magazine*, Volume 8, Issue 1, 1993, pp. 31-39.

D. Hingston, "Star Recognition Using Computer Vision (OpenCV), "https://www.instructables.com/Star-Recognition-Using-Computer-Vision-OpenCV/" (as of September 3, 2022).

D. Mortari, M. Samaan, C. Bruccoleri, & J. Junkins, "The Pyramid Star Identification Technique," https://cupdf.com/document/the-pyramid-star-identification-technique.html (as of September 3, 2022)

D. S. Mehta and S. Chen, "A Spearman Correlation Based Star Pattern Recognition," *2017 IEEE International Conference on Image Processing*, Beijing, China, 2017, pp. 4372-4376.

F. Zhou, T. Ye, "Lost-in-Space Star Identification Using Planar Triangle Principal Component Analysis Algorithm," https://www.hindawi.com/journals/mpe/2015/982420/ (as of September 3, 2022).

L. Sun, J. Jiang, G. Zhang and X. Wei, "A Discrete HMM-Based Feature Sequence Model Approach for Star Identification," *IEEE Sensors Journal*, Volume 16, Issue 4, 2016, pp. 931-940.

M. Jiang, Y. -Z. Ye, M. -Y. Yu, J. -X. Wang and B. -H. Li, "A Novel Star Pattern Recognition Algorithm for Star Sensor," *2007 International Conference on Machine Learning and Cybernetics*, Unknown, Unknown, 2007, pp. 2673-2677.

W. Ding, X. Li, H. Yang and Z. Zhang, "A Star Map Recognition Algorithm based on Pattern Recognition," *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference,* Chengdu, China, 2019, pp. 1292-1297.