Star Mapping Algorithm

Darick W. LaSelle
Stanford University Department of Electrical Engineering
dwlaselle@gmail.com

***Abstract*—This paper proposes a project investigating how to map a set of points (presumed to be stars) onto a star map. It will implement an algorithm that does star mapping and explore the robustness of the algorithm.**

Keywords— digital image; point-matching star map;

I. Introduction

This is an ambitious project. Getting, understanding, and parsing a star map itself is a non-trivial task. Next, while Star Mapping is well understood, algorithms to implement star matching are not highly available in the public domain as anything other than rough descriptions, so implementing one requires some effort. Finally, trying to use real star imagery and parse that image for mostly valid stars will be a final effort if time permits. I'd like to try applying this to a real-time video of a starry sky as a further stretch goal.

With these tasks in mind, this paper proposes the following breakdown of the project:

1) Obtaining a star map and parsing it into code or lookup database.

There are lots of star catalogues out there:
http://www.astronexus.com/node/34
http://answers.google.com/answers/threadview/id/282438.html
http://www.projectrho.com/public_html/starmaps/catalogues.php

There needed to be research for which will be best for this project, based on: coordinates used, number of stars (only care about visible stars and should filter out non-visible ones either prior to using the data, or dynamically), and how easily it will integrate with the code.

2) Researching, choosing, and implementing a star matching algorithm.

There are a few papers that talk about matching a star chart to visible stars. This kind of star mapping is used today in a few satellites to determine orientation of the satellite.

http://www3.sympatico.ca/vpaquin/tutorial/tutorial1.htm
http://ntrs.nasa.gov/search.jsp?R=19960035749
http://pdf.aminer.org/000/994/725/an_application_of_point_pattern_matching_in_astronautics.pdf

The algorithm is an application of point matching. It'll have two sets of points … a database and a query set. Searching the database for the "best" match including transforms for this set of points is an application that should be highly scalable. This will be the meat of the project. Initially, it will use reference points that are manually taken from the database and transformed by hand to have some searches that the answers are known in advance. I can then experiment with removing less luminous stars from our query set of points and applying more aggressive transforms to determine if the algorithm performs well..

3) Processing real star imagery for points.

I can take photographs of night sky as well as get images off the internet. Processing them will be the more difficult part, but should be much easier with Digital Image Processing. This will be a test of whether I can map "real" imagery to the star database. This is assumed to be a stretch goal.

II. Previous Work

Point matching algorithms have existed, and been well vetted, for many years. As a specific example, the Institute for Aerospace and Astronautics of theTechnical University of Berlin uses point matching to measure position and orientation on their satellites. [1] Additional work on this application was published in 1994 in Academic Press Limited. [2] This early work identifies that the concept is feasible, even with limited computing power. However, I was not able to locate any detailed descriptions of the algorithms. The majority of the code developed for the star matching system and the data used to test the system has been organically developed.

III. Programming Implementation

A. Basic Algorithm

The basic algorithm for point matching is:

- Identify a point
- Find an adjacent point (neighbor), and set it as a reference neighbor
- Find additional neighbors
- Match the parameters to associated truth files
- Score the comparisons
- Select a best match (if one exists)

Identifying the first point is an arbitrary selection. For most of the pieces of the program design, a point in space is selected irrespective of whether or not it is a star. Some limited testing was done by selecting a star, but that inevitably creates increased specificity which is undesirable for the current tests, though may be useful in creating a mapping of the entire sky.

Assign a distance from a point that is an acceptable region (this can be a region close to a star, or an entire constellation, depending on the place in the program/testing). This value is

actually a degree reference. The declination and right ascension angles are assumed to correlate to a Cartesian coordinate system (mathematically acceptable due to the small-angle approximation). [3]

For each point (star) in the selected region, perform the following calculations:

A second point is selected (order is irrelevant as will be explained further on) and the line segment from the first point to this second point will be used as a reference angle. Every point identified past that point that is within the acceptable distance is referenced from that first point using the assumption that the angle from the central star to this second star will be defined as 0º. This step saves a processor operation later, because the testing will need to identify reference angles anyway. It is worth noting that the industry standard for referencing stars is J2000, which means it is where the stars were at noon on January 1, 2000. That is important to note that here, because except at rare times, the rotations of the test images are expected to be different from that of the truth data. The additional points inside the region are then captured and referenced to the center point. Their angles are coupled with their relative distance (unitless) to create a description of this point's relative position to all the other points from this region.

The data thus far catalogued from this point is whatise reference as a fingerprint file. An example fingerprint array is shown below in Table 1. If we were to only look within 1.5 degrees of the star Alnilam in our filtered sky map, this is what we would get.

The goal of this kind of file is to create a "fingerprint" that can be reliably compared regardless of scale or rotation of the comparison point. We developed this approach from a real time shape matching approach.

RA:	Dec:						
84.1	-1.2						
Star 1		Star 2		Star 3		Star 4	
Angle	Dist	Angle	Dist	Angle	Dist	Angle	Dist
0	0.96	332	0.67	2.7	0.6	160	1.16

Table 1: Fingerprint array for Alnilam

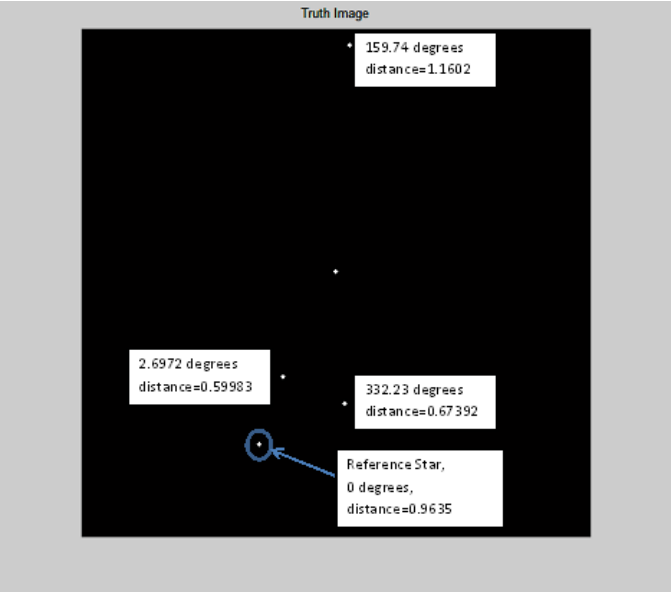
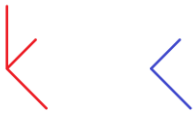


Figure 1: Graphical Fingerprint for Alnilam

Every star in a truth fingerprint file will have a fingerprint array associated with it. The point matching algorithm then compares every star in the test file to every star in the associated region. The score to a given point will be representative of the best match of any one test fingerprint file to a reference fingerprint file.

B. Comparing Fingerprint Files

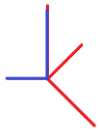
For an example of how two fingerprint files are compared, let's look at the following example:



The image on the left with three rays will represent our reference fingerprint file for a star, while the figure on the right will represent a test fingerprint file. You can visually see that the blue fingerprint exactly matches portions of the red fingerprint, but let's follow through how a computer might look at this.

First, we must pick two rays to compare. Simply picking the first ray of each fingerprint seems obvious:

It is equally obvious that these two images do not match. In the comparison, note that while the first angle matches (by declaration), the second does not. We can also quickly see that the next possible rotation of the test image (setting the next ray as the reference) does not produce a match.



The next step is to rotate the truth image. To, by declaration, say that the *second* angle in the truth image is now the reference angle.

This obviously creates a perfect match of the angles, and the difference of scale of each ray will be similarly equal (within a margin). Now we can say that the fingerprint matches the reference image. This example should also show why it is required to try all rotations of *both* the test image and the reference image. If just the wrong angle is occluded, there would be no matching results for a real match.

In this manner, test every rotation of the test fingerprint (declaring each angle in turn to be the reference angle) to every rotation of the reference fingerprint, and produce a score. The score itself is not meant to be a measure of absolute "truth", but a confidence value. The score we used says that we give up to 1.0 points for every angle matched within a threshold. Each match then loses points if the scale of the two radii do not match the scale of the first two reference angles. Finally, we divide by the maximum number of angles in the test fingerprint or truth fingerprint to find our total confidence.

C. *Specific Programming Implementation – Framework*

To begin with, not knowing which database I'd be using or how the stars would be described, I started with stars as if they were points in an image, and used relative X, Y coordinates. For a first "test" image, I created a file similar to the red reference fingerprint shown above (with a few more and less regular angles). I used that to compare to itself to verify I could match an image correctly.

From there, I stepped up to a points-representation of the Orion constellation. Knowing that there was something that mapped 1 to 1 images (but verifying it), I then scaled and rotated that image randomly a few times to produce test data. The beauty of this fingerprint implementation is that it should produce results regardless of scale or rotation of the image. Needless to say, this was the biggest debugging stage (and a stage that produced log files on the order of 900MB for all the processing that was required).

To do the comparison, you need to compare:

- The test star –to–
- Each star in the truth array –with–
- Each rotation of the test star –against–
- Each rotation of the truth star

So, four loops were needed to get through all the data. It is easy to determine how many loop iterations are needed, then. In the data set that was tested presenting, there were 80 stars in the truth array and used those same stars as part of the test array. Each fingerprint file, then, had one star with 79 angles (we can limit to a nearest neighbor in the test cases for speed, see Future Work). In this most straightforward of comparisons, then, simply to compare one test file had 250E9 loop iterations to process through. Most of the work in implementing this algorithm came from keeping track of the four layers of indices for both the reference and test data.

D. *Main database generation*

The finding of a usable star database ended up being more time consuming than originally expected. While there are plenty of applications and applets available that will return a star given its coordinates, and vice versa, none of those provided access to the raw data. Additionally, some databases had an embedded SQL database, but getting at the raw data was again troublesome. I ended up finding two databases that were in some respect usable. One from the Saguaro Astronomy Club, and a second from David Nash, called the HYG Database.

1) *Saguaro Astronomy Club Database*

This is a conglomerate of amateur astronomers, loosely based out of Arizona. The database they provide has been in the works for 20 plus years. [4] The descriptions and setup of the stars appears to be very accurate, and is easy to work with. The drawback from this database was that virtually all stars visible to the naked eye are assumed to be a maximum magnitude. In a point matching algorithm, that is not particularly important, but being able to visually identify constellations and stars is virtually impossible.

2) *The HYG Database*

While not quite as easy to use initially, the HYG Database generated by David worked much better for our application. HYG stands for Hipparcos, Yale, and Gilese, and are the three star catalogs that this database is a compilation of. [5] It has a much more accurate magnitude reporting system, which has made visual testing more plausible, and the database easier to filter to the desired size. From this database, through Matlab, you can produce images from a center point and visually verify that we are looking at the same point.

(a)Matlab representation

(b)Sky Imagery

Figure 2: Comparison of Matlab imagery using the HYG database (a) to a real image of the sky (b)

It is clear from looking at the two images in Figure 2 that both are looking at the same set of stars.

The HYG database allowed me to parse out the information used in this database. The values are declination (dec), right ascension (RA) and magnitude (mag). The magnitude given is the stars apparent visual magnitude. For the sake of generating the imagery out of Matlab, I converted the logarithmic scale given by the \magnitude to a linear intensity (ranged 0 to 1). I then used the fingerprint function described in section 3.A to create a master fingerprint file.

E. Test file generation

In order to identify good benchmarks for "positive" identification, a series of test files were created. For ease of visual clarity, the section of the sky typically defined by the constellation Orion was used. From a mathematical perspective, the section of sky is described by Table 2.

Constellation	Orion
Center RA	5.6 degrees
Center Dec	-1.2 degrees
RA Range	+- 10 degrees
Dec Range	+- 10 degrees

Table 2: Test file numerical description

As part of the testing, a truth file was produced to match the database exactly. Figure 3 shows the Orion truth image.

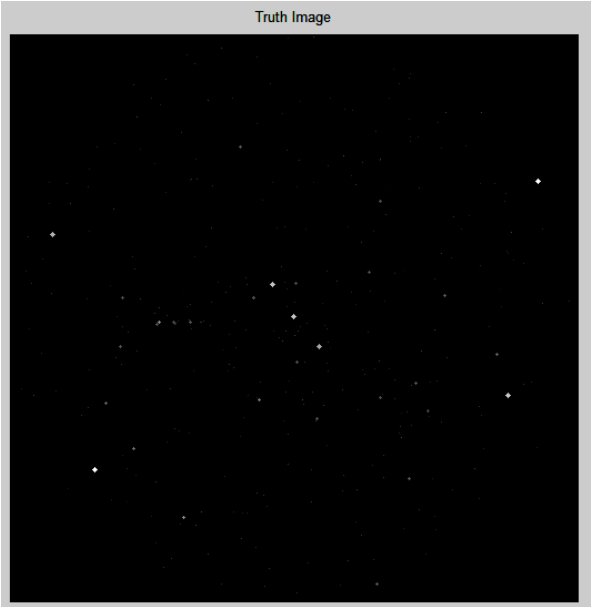


Figure 3: Orion truth image

In order to be a useful star matching algorithm, the consideration that no image will exactly match the test image has to be taken. To test the image, images that closely match the test image should be tested, and the matching scores benchmarked.

Figure 4: Orion with 50% occlusion

For each test file, one or more modification was applied from the following list:

- Occlusion
- Addition
- Skewed Position

1) Occlusion

A number of factors can make a test image have fewer stars than the truth file. Clouds, trees, birds, the moon, city lights are just a couple of examples. The test files generated ranged from no occlusion to 30% occlusion. That is, some portion (up to 30%) of the stars from the test image was removed. Figure 4 is an exaggerated case, where 50% of the stars were removed. The easiest way to image this is looking up at the sky in the middle of a city versus out in the desert. The database is setup closer to the visible stars in the desert, meaning that the pictures taken close to a city will have significantly less stars when compared to the truth image. This should have a low impact on the algorithm because angles will still be perfect matches with the reference stars.

2) Addition

The other side of the spectrum is when things show up in the pictures that aren't actually stars. Satellites are the most common stray point, and asteroids are another possibility. Even airplanes might show up in an image of the night sky. Figure 5 shows Orion with 50% addition. This will be the most difficult for the algorithm to sort out, as many new angles will be (falsely) introduced into the test fingerprint. This is particularly easy to see in comparing the occlusion test file verses the addition test file. The addition file would be very difficult to pick out as Orion even for a human at 50% addition, whereas the occlusion file is still quite obviously Orion.

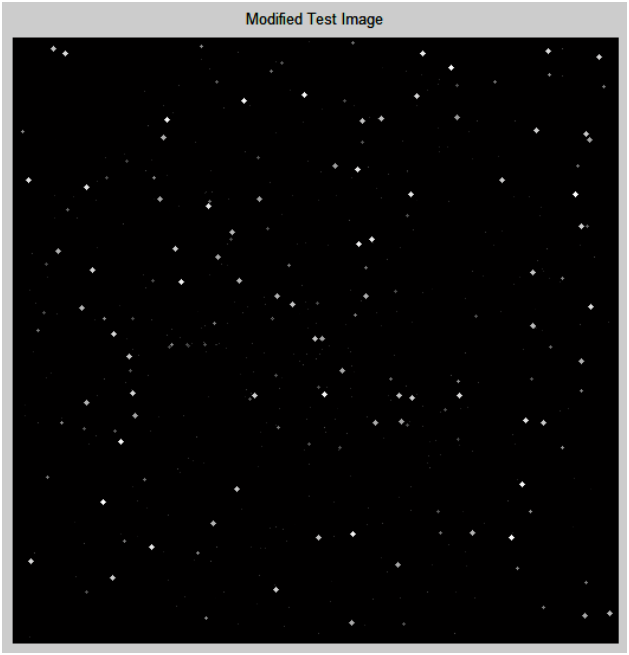


Figure 5: Orion with 50% addition

3) Skewed Position

Whether due to the atmospheric effect (twinkle), or motion of the camera, it is possible for a star to be slightly skewed on the image. The test images included different skewing amounts up to 0.1 degrees. This skewing is shown in Figure 6 (exaggerated to 1 degree). The skewing is visually obvious if you look at the left hand star on Orion's belt.

Figure 6: Orion with up to 1 degree skewing on 50% of the stars

4) Combination

It is unlikely that one effect will be present on the image, while the other two are not. The image shown in figure 7 is what we refer to as the hail storm. It includes 50% occlusion, 50% addition, 1% skewing on 50% of the stars. This image is not intended to be matched, but rather to visually exemplify the range of the test.

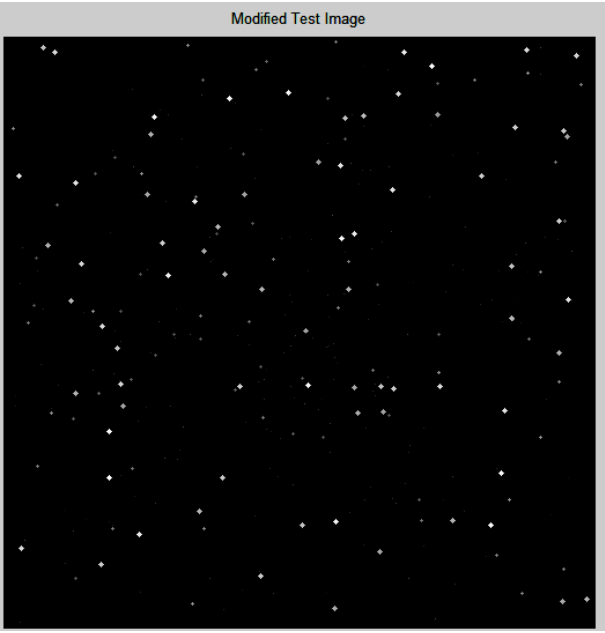


Figure 7: Orion with maximum modifications

In total, 2662 separate images were generated to benchmark the algorithm and decide how to score correct matches. We used a narrowed down version of the HYG database to include only those stars that have a magnitude brighter than 6, which is generally assumed to be visible to the naked eye. This database includes 5,007, and makes the truth file for Orion include 80 stars.

IV. Experiments

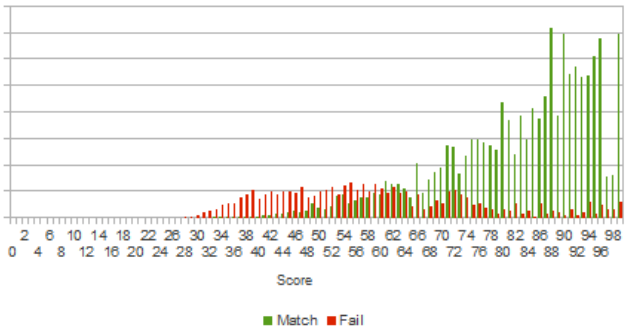
After the initial test experiments that showed I could get an exact truth match between identical images and that scale and rotation did not affect the answers, I moved on to a full scale simulation.

I took the Orion constellation region (80 stars) and made a reference map of all fingerprints in that region. I then took that map and put it through the following permutations:

- Adding stars, 0 to 30% in increments of 3%
- Occluding stars, 0 to 30% in increments of 3%
- Wiggling stars, 0 to 0.1º in increments of 0.01º

.

Next, I looked at what the answers were telling us. To measure the confidence score, since I attempt to generate a score for every "best" match available, I can see how many false positives we generate vs correct answers very easily. I kept the reference to the "real" star that was in the chart, even if it was moved or occluded. This let me compare real data to real data.

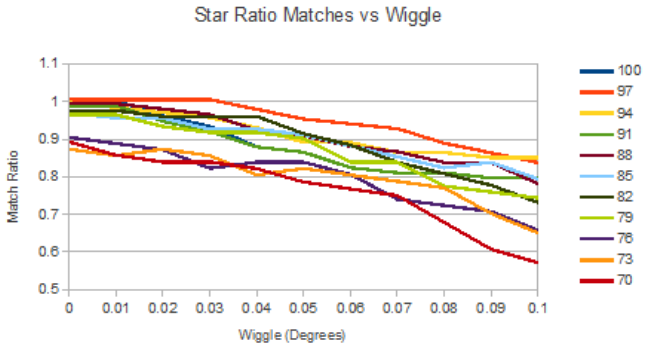


This chart shows the number of matches that were correct vs incorrect in each bin from 0 to 100 of the generated scores. I could quickly pick a threshold of about 65 or 70 if we only wanted to assure that multiple readings would generate a majority (or in this case, a 2/3 majority) of correct positives. If you then take the median of the direction of all the reference stars matched, you would have the approximate center of the image correctly identified.

Notice also that the false positives in the lower scores, while they exist, are much smaller than the positive matches of the higher scores. Higher scores, then, are that much more certain, and while there are not any that scored "perfect" (generated no false positives), if we have multiple stars in any picture that generate a "good" score above our threshold, we would be able to say with some certainty that we could identify the orientation of that image.

Lastly, I wanted to look at the effect of "wiggle" on our data. In the real world, stars appear to shift slightly due to atmospheric conditions. We allowed for up to 0.1° of wiggle in all of our stars. This is a huge amount that would not be seen in the real world. As an example, take two stars in Orion's belt. They lie only 0.7 total degrees apart in the night sky. One seventh of that distance is a huge amount of movement and seemed appropriate as a maximum test.

To do this, I measured the number of "real" stars in any test image as a ratio. This could go from 100% down to 70% as we added stars. Occlusions would not impact this ratio, though might have an impact on data where large amounts of added stars existed (forming perhaps a majority of false angles for some fingerprints). Plotted each of these ratios vs the percentage of stars correctly matched and the amount of wiggle.



We see that many of the ratios stay well above 80% regardless of wiggle, but do decrease over larger percentages of movement. We also see that the three worst starting ratios are those that would have the most stars occluded and the most stars added, as expected. Still, to have even those start at over 90% success ratio and end higher than 50% in all cases was better than we expected.

This chart shows only the true positives, so a threshold in this case wouldn't change the output we see here. In other words, this is a "best case", assuming you can know that a match is correct or not. It shows that star movement can play a large role in correct detection, as can occlusions and additions, but to have a truly large effect, both problems must be highly pronounced, which seems generally unlikely. Keeping in mind that a star shift by even 0.05° would be unlikely (and even more unlikely that *every* star in the image would be shifted randomly by that amount), as well as the unliklihood of finding an *additional* 30% stars in the night sky, it seems reasonable to generally assume a match ratio of well over 90% using this algorithm.

All in all, the algorithm performed very well, and should be very scalable

V. Digital Image Processing

With a high successful set of test runs, another major piece of this project was to develop a way to compare date from real imagery to the truth file. While this seemed like a very daunting task, the use of DIP techniques learned in EE368 at Stanford made it almost trivial. There were three steps involved in taking a digital image, Thresholding, Region Identification, and Point Mapping.

A. Thresholding

Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images[6]. With most star images, almost any threshold method will work, as there is such a high contrast between the background and forground when looking at the night sky. For this particular algorithm, I scaled the threshold slightly in order to manipulate the number of stars pulled out of an image.

B. Region Identification

Region identification is also sometimes called blob detection. It is a systematic way to identify disparate regions of the foreground image. It is especially effective on binary images. The built in Matlab code "regionprops" contains a data set for centroids, which provides exactly the information I was looking for. Images 8-10 show the image processing steps on the Bootes constellation.

Figure 8: Original Image

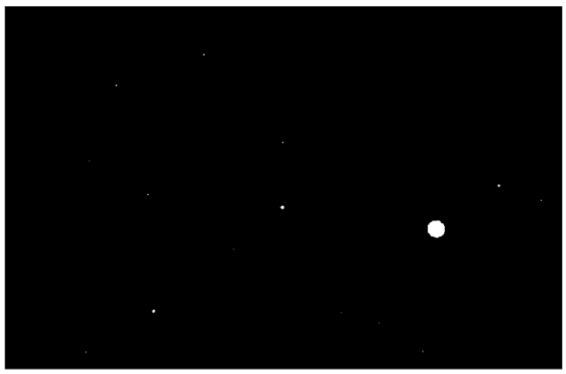


Figure 9: After Thresholding

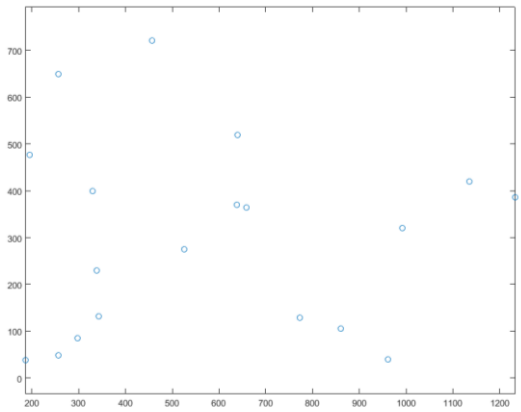


Figure 10:Identified regions

C. *Point Mapping*

Point mapping is not universally held as the same thing, especially when it comes to star mapping. For my purposes, I identified the point (presumed star) closest to the middle of the image, and made that the reference point, noting that it is likley the target of the image if it is in the middle. From there, ever point is mapped by its distance and angle. This data is no available to use to test against the truth data.

D. *Imagery Results*

The results on the actual images were not as encouraging as the test data. While the confidence scores seem tomakes sense (most in the 50% range), there is not as large of a distinction between the "correct" star and the next closest "incorrect" star. This also led to more false positives

VI. Future Work

There are a lot of possibilities for future work in this project.

- This algorithm can (and likely will) be modified to fit into an OpenCL/GPU environment.
- The real imagery process should be verified again, as there may be a small scaling issue or quadrant trigonometry issue that can be fixed
- Lastly, improving how the confidence score is calculated and implementing a threshold could result in fewer false positives.

VII. Conlusion

This is, I believe, a novel approach to point matching, and with the advent of "embarrassingly parallel" computation, may be a very effective way to match shapes.. It may not beat current approaches (we're not sure since they weren't available to study), but still produce excellent results and could possibly be scaled to real-time.

I also found that our scoring metric for "confidence" was fairly robust along a large range of parameters. After comparing several fingerprints, we were able to say with some confidence (by inspecting the median of the directions) what the orientation of the image was.

There is a lot of possible future work that could be done, but this was a wholly successful start.

VIII. Acknowledgements

The author would like to gratefully acknowledge Andrew McLauthlin at the University of Colorado for his help in vetting algorithms

IX. About The Authors

Darick LaSelle is a Controls Engineer for a pump and compressor manufacturer in Arvada, CO, and is a graduate student in the Electrical Engineering Department at Stanford University.

[1] V. Paquin, "*Point Pattern Matching*," [Online]. Available: http://www3.sympatico.ca/vpaquin/tutorial/

[2] G. Weber, L. Knipping, H. Alt, "*An application of point pattern matching in astonautics"* Aerospace and Astronautics of theTechnical University of Berlin, J. Symbolic Computation (1994) 11

[3] (13 March, 2013), *"Small-angle approximation",* [Online]. Available*:* http://en.wikipedia.org/wiki/Small-angle_approximation

[4] S. Coe, "*Saguaro Astronomy Club Database version 8.1*," [Online]. Available: http://www.saguaroastro.org/content/downloads.htm

[5] D. Nash, "The HYG Database," [Online]. Available: http://www.astronexus.com/node/34

[6] (Shapiro, et al. 2001:83).