Detection of Constellations in Astronomical Images

University of Malta

Faculty of Information & Communication Technology

Bachelor of Science in Information Technology (Honours) (Artificial Intelligence) 2nd Year

ARI2201 – Individual Assigned Practical Task

Dr Kristian Guillaumier

Francesca Maria Mizzi

118201L

francesca.mizzi.19@um.edu.mt

Table of Contents

I. Introduction

The main objective of this assigned practical task is to create a software which can recognize set constellations in images which are noisy and blurry. This will be done using computer vision and template matching.

Many different types of software have been created to do the task assigned above, some of them being “Star Walk 2”, “Star Tracker” and “SkyView”. These apps generally have very similar functionality: using the rear camera of a phone as a live feed and identifying constellations, stars and planets based on the location of the user as well as the video feed from the phone’s camera. While many apps have been created, not much academic research has been carried out regarding the best techniques involved in recognising constellations efficiently and with the highest accuracy.

Through this assigned task, the best techniques involved in distinguishing constellations in images will be explored and identified. The techniques which will be tested are template matching using computer vision as well as a more machine learning based approach through the testing of recognition with a convolutional neural network.

Another obstacle in creating tests is the lack of a database of constellations as well as images containing the constellations. Since there is no dataset with the information required to find the best techniques in constellation identification, the dataset will be created using the software “Stellarium”. This software allows the tracking and identification of constellations while also allowing the user to turn off any labels in the images which might influence any machine learning or template matching. The removal of labels is vital in the success of the task, seeing that in a real-life scenario, no labels will be available in the sky for our phone to match.

This assigned task will be split into two tasks: building the dataset and recognizing the constellations. At the end of this task, the software developed will be able to successfully identify seven different constellations with a certain degree of accuracy.

As mentioned, the dataset will be created using the software “Stellarium”. For each constellation identification attempt, two photos will be extracted from the software; a general empty photo with the constellation in the image as well as a “template” image which will be a photo containing the constellation only. Since two photos are not enough to build a database, each empty photo will have a random amount of blur and salt-and-pepper noise added to it to simulate a hypothetical amateur user’s photo. Each empty photo will generate five versions of the original photo with the added blur and noise to build the dataset.

Identifying the constellations in the image is perhaps the hardest part of this task. The two techniques which will be attempted and adopted are template matching using computer vision as well as convolutional neural networks.

II. Research and Literature Review

i. Constellation Detection

In their paper “Constellation Detection” [1], Suyao Ji et al. attempted to automatically detect all 88 recognized constellations from their dataset of images. Their project was made of three phases – image pre-processing, template machine learning and constellation pattern detection. Ultimately, the algorithm they created had an accuracy of 92.8% with their 15 test images.

Ji et al. purposed a constellation algorithm to detect constellations from a photo of the night sky based on the star patterns as well as the fixed patterns constellations have with neighbouring stars. They had three steps in their approach: image pre-processing, template machine learning and constellation matching. The first step involved filtering out all non-star objects and reducing noise. The second step involved generating a constellation database with information descriptors for the 88 constellation templates. The final step compared the pre-processed image with the templates to search for matching constellations.

Contrary to the research carried out in my project, Ji et al. detected multiple constellations from the same image. Thus, each test image could have multiple constellations. The flow for the implementation of their algorithm can be seen below in Fig. 1.

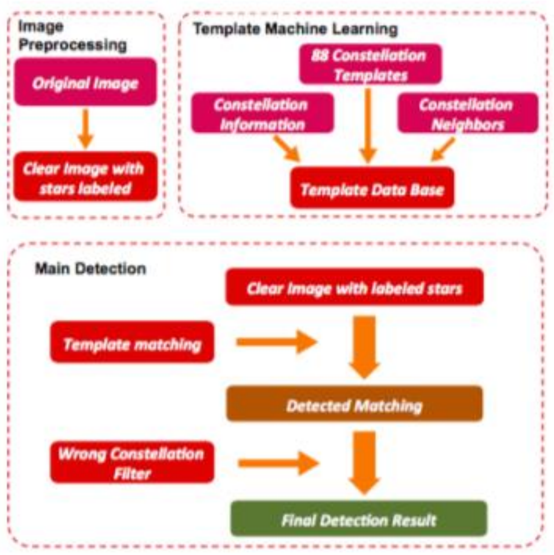


Image Preprocessing
Original Image
Clear Image with stars labeled
Template Machine Learning
88 Constellation Templates
Constellation Information
Constellation Neighbors
Template Data Base
Main Detection
Clear Image with labeled stars
Template matching
Detected Matching
Wrong Constellation Filter
Final Detection Result

Figure 1 - Implementation Flow for Ji et al. Extracted from [1]

In order to pre-process their images, the researcher aimed to get a binary image from the picture, containing only the stars. Thus, Ji et al. performed thresholding on a grayscale image to exclude trees and buildings while keeping the stars.

The templates used were based on a set of modified constellation charts which were extracted from the IAU standard constellation chart. Similar to the image pre-processing carried out earlier, the template images are binarized and different colours were assigned to stars and to connection lines. They then filtered out any unrelated features and recorded the stars and constellation lines from the template. They also recorded the two brightest stars close to the constellation’s area and extracted the distance between them.

To detect constellations from their processed image using the processed templates, Ji et al. performed the following steps:

1) Use a region labelling algorithm to rank the stars based off brightness.
2) Ascend the templates based on the number of stars existing in the constellation.
3) Pick the brightest two stars to determine the scale and rotation of the templates.
4) Check if there is a star in the image which matches the relative position of all the stars in the templates.
5) Repeat step 4 until the number of matched stars equals a constellation.
6) Repeat steps 3-6 to find other constellations.

Ji et al. tested 15 images with a total of 28 constellation within them. Constellations of 10 images are detected correctly with no missing or incorrect detected constellations. Of the 28 constellations, 3 of them were undetected and 2 were incorrectly detected. There were, however, some special cases where the detection was not perfect. These cases are when the constellation is cut off either by the edges of the image or by something blocking it such as trees, constellations that only have 2 or 3 stars such as “Canis Minor” and finally when the stars within the constellation were very dim.

ii. Pros and Cons

Some pros and cons can initially be seen when reading the description for the implementation technique chosen for our task.

A pro in our task is that should the implementation be successful, it would be very easy for a user to simply take a photo of the sky and quickly get a result for any constellations which may be present. Another pro is that the simplicity of the implementation chosen allows for results to be obtained quickly.

A con, however, is that since the templates were extracted from the software “Stellarium” certain real-world noise could not be added accurately. Certain aspects such as fog, the brightness of the stars as well as stars being missing from the constellation cannot be implemented when using the images extracted. Another con from the implementation is that only one constellation can be identified in a photo. Thus, any photos containing multiple constellations will result in only one being found.

III. Implementation

Following the research carried out in the literature review, it was decided that using a convolutional neural network was not the best approach to match constellations. It was thus decided that template matching will be used to identify the constellation in the images.

Using template matching to identify constellations involved two steps: 1) Dataset and templates acquisition and pre-processing and 2) Template matching, as described below.

i. Image Acquisition and Pre-Processing

To build the database to perform template matching, images and template images needed to be acquired from the software "Stellarium". For each constellation, a template image was acquired, outlining the constellation, as well as an empty image of the night sky which holds the image. Examples of these can be seen below in figures 2 and 3.

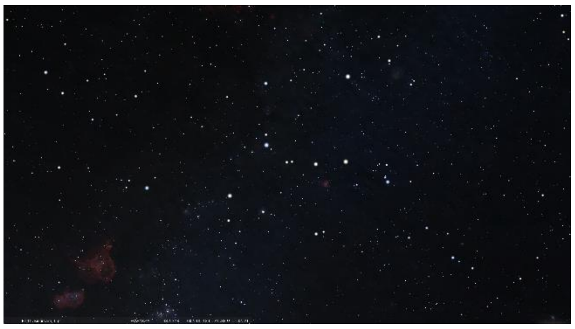


Figure 3 - Empty photo of the night sky containing the Cassiopeia constellation.

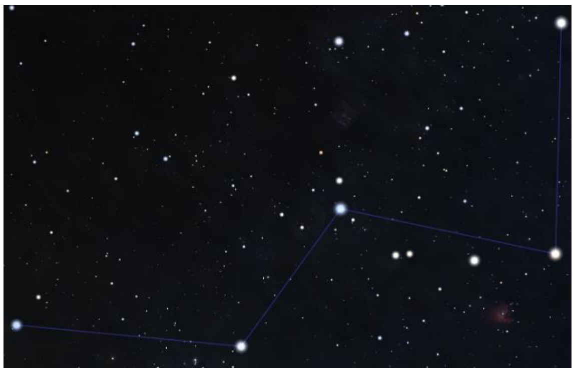


Figure 2 - Template image for the constellation Cassiopeia

To build the dataset, one empty photo of the sky per constellation was not enough. Therefore, modifications were made to the original empty image to acquire more photos. To do this, blur was added to the photo using the OpenCV [2] command *cv2.blur()*. The amount which the image was blurred was randomly determined. To imitate a real-life photo, salt-and-pepper noise was added to the image to simulate a grainy phone camera. Using the above steps, five images were acquired from each empty photo. The filters added to the photos can be demonstrated below in figures 4 and 5.

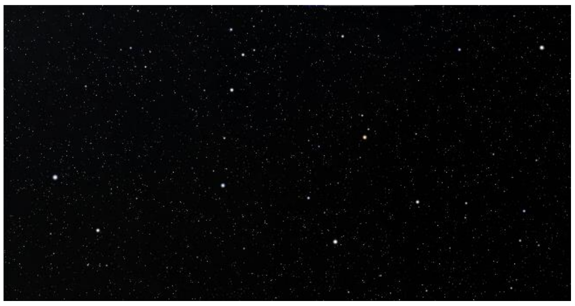


Figure 4 - Hercules constellation after noise and blur

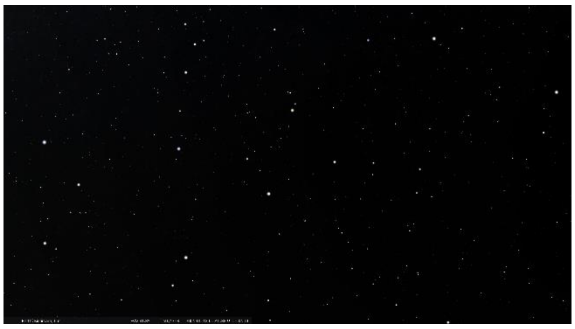


Figure 5 - Hercules constellation before noise and blur

The images must then be filtered to prepare for template matching. The pre-processing chosen to implement were median blur, to remove any extra noise in the photo, as well as edge detection to remove any stars which are not relevant to the constellation. An example of the images after pre-processing can be seen below in figure 6. It is to be noted that the pre-processing was also carried out on the template images before they were added to the dataset.

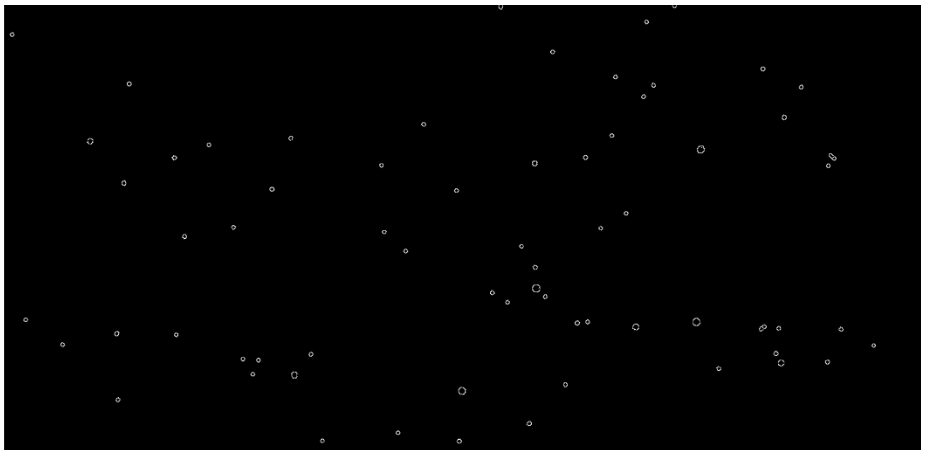


Figure 6 - Hercules constellation after pre-processing.

ii. Template matching

The second part of the task is to match the template to the respective image. To perform template matching on the two datasets, the OpenCV [2] function *cv2.matchTemplate()* was used. A check is then carried out based off the result of the template matching. If the result is above the defined threshold, the match was a success, and a true result is considered. If the result is below the defined threshold, a match was not found, and a false result is considered. Below is a figure of a visual description of what is considered a successful match.

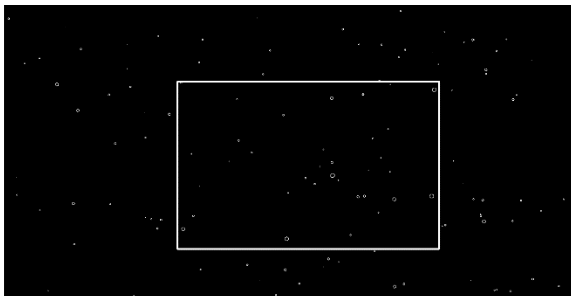


Figure 7 - A successful match of the Cassiopeia constellation

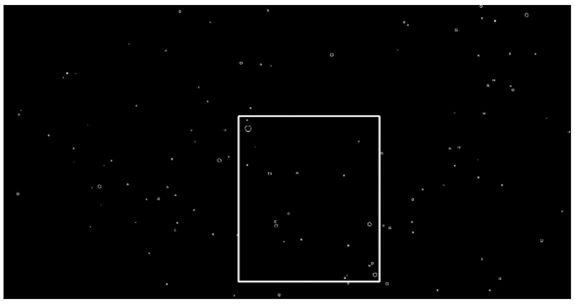


Figure 8 - A successful match of the Lyra constellation

For each image, the above steps are carried out on every template to check for every constellation. To measure how many of the matches were successful and how many were not, certain results are generated. To check if a template was successfully matched, a check is carried out on the values assigned to each image. Prior to this, when each image was added to the datasets, each image containing a constellation as well as each template was assigned a number based on the constellation it held. For example, all Cassiopeia constellations were assigned the number 1, all Hercules constellations were assigned the number 2, all Leo constellations were assigned the number 3, etc. To check if a match was successful, the values of the image and template which were matched are checked and if the numbers are the same and the flag returned was true, the match was a success. If the image and the template do have the same values but the flag was returned false, the match was not a success. If the image and the template do not have the same value but the flag returned was true, an incorrect match was made. Based off these three results, a general idea for the accuracy of the program can be acquired.

iii. Testing

To test the implementation, the template matching was carried out on seven different constellations as well as their corresponding images. In total, 245 different photos were attempted to be matched to 7 different templates.

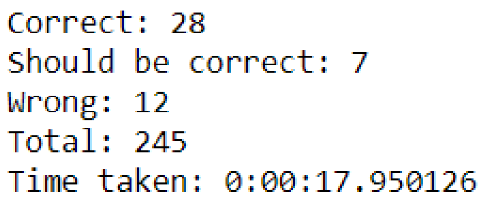
The main issue which was identified when testing the software was an issue concerning scale and rotation regarding the *cv2.matchTemplate()* function. The function only takes into consideration templates which are the same size and at the same rotation as that of the desired location in the image. Therefore, if the constellation is at a different angle or a different size than that of the template, a match is not found. Through research, some algorithms were identified which could possibly fix this problem such as the SIFT [3] module from the OpenCV library. Unfortunately, these modules would not install on the computer used and as such, the algorithms could not be tested.

A level of success was achieved when trying to fix the issue of rotation in the templates. However, the solution achieved was far too time consuming to run and a photo which would normally take milli-seconds to find a match took almost minutes. It was therefore decided that the scale and rotation variants would not be tackled in this task.

IV. Results, Evaluation and Critical Analysis

i. Results

After running tests on 7 different templates as well as 35 different photos, a total of 245 tests, the following results were achieved:



```
Correct: 28
Should be correct: 7
Wrong: 12
Total: 245
Time taken: 0:00:17.950126
```

Figure 9 - Results achieved from testing.

From the above figure 9, we can extract the following information: a total of 245 checks were carried out, 7 templates against 35 photos. Of those 245 checks, 28 resulted in a successful match. This means that 28 out of the 35 photos were successfully recognized. 7 out of the 35 were marked as "Should be correct". What this means is that the template was in fact present in the photo, but the function could not match it. Therefore, 28 photos were successfully matched and 7 were not: but what of the 12 photos incorrectly matched? After investigation, it was discovered that the 12 photos which were matched incorrectly were a result of certain constellations being found in pictures which were not originally detected. For example, the Ursa Minor constellation being found in the photo which also held the Leo constellation. This resulted in a match being found but the values of the images not matching, hence the "wrong" match. Overall, it took 18 seconds to extract the photos and templates, carry out image pre-processing and run 245 checks on the images.

ii. Strengths and Weaknesses

One of the biggest strengths of the overall project is the accuracy of the matches. 28 out of 35 photos were successfully matched to the correct constellation. This evaluates to an accuracy of around 80%. Another unplanned strength is the matches between constellations which were not initially seen. This implies that the template matching is working more accurately then initially expected.

One of the main weaknesses of the software is the lack of accommodation for templates at a different scale or rotation. If the template is not the same size or at the same rotation of the constellation in the original image, the function will not match a success.

iii. Future Improvements

One possible future improvement is the addition of the remaining 81 constellations. The addition of the remaining constellations would allow for users to be able to try locating constellations in most images of the sky.

Another future improvement would be creating a mobile app for the software. The implementation of a mobile app would allow users to use a live camera feed as the input device or perhaps it would allow users to upload photos to the app directly from their phone.

One of the biggest improvements which could occur is the implementation of some form of scale and rotation matching for the template. As mentioned, if the template is not the same size as the constellation in the image, a match would not occur. Some options to solving this problem were explored, including changing the scale of the image and keeping track which scale has the best overlap with the image and adjusting accordingly, while also randomly changing the rotation and keeping track of which overlaps best. However, it was discovered that both above implementations considered took far too long and were very inefficient.

Another future improvement would be the accommodation for multiple constellation matches within the same photo. As mentioned earlier, the template matcher matched some constellations in images which initially were not identified and thus, the addition of a system which can check for multiple matches would greatly improve the functionality overall.

V. Conclusion

Ultimately, by using computer vision techniques such as template matching and image pre-processing, 7 constellations were successfully identified when compared to empty sky photos containing the relevant constellation. Overall, an accuracy of around 80% was achieved when comparing 35 photos to their relevant templates.

The overall template matching was possible due to the pre-processing of photos to remove any noise and extra stars which would not be relevant to the constellation. The pre-processing carried out on the images involved adding a median blur to images and using edge detection to remove any excess noise in the photo which would result in the template not matching the constellation.

The template matching itself was a greater success then initially expected due to the function finding constellations in images which were not initially identified. However, due to the debugging and labelling system implemented, they were marked as incorrect matches due to the labels not being the same.

VI. References

[1] S. Ji, J. Wang, and X. Liu, "Constellation Detection,." [Online]. Available: https://web.stanford.edu/class/ee368/Project_Spring_1415/Reports/Ji_Liu_Wang.pdf.

[2] "OpenCV: OpenCV modules," *Opencv.org*, 2021. https://docs.opencv.org/master/

[3] "OpenCV: Introduction to SIFT (Scale-Invariant Feature Transform)," *Opencv.org*, 2020. https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html

[4] A. Rosebrock, "OpenCV Template Matching (cv2.matchTemplate) - PyImageSearch," *PyImageSearch*, Mar. 22, 2021. https://www.pyimagesearch.com/2021/03/22/opencv-template-matching-cv2-matchtemplate/

[5] "Stellarium Astronomy Software," *Stellarium.org*, 2021. https://stellarium.org/

[6] "Smoothing Images — OpenCV-Python Tutorials beta documentation," *Readthedocs.io*, 2016. https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html

i. Table of Figures