Wrocław University of Science and Technology

# Computer vision project - astrolocator

Michał Porębski 235181

Tutor: Msc Adrian Gałęziowski

Wrocław 2019

# Contents

## 1 Introduction

Once upon a time I was thinking what would happen if I would end up in place, under clear night sky, where is no GPS or any other signal and I wouldn't have any idea how did I get there. I thought then I could look at the sky and find the most recognizable constellations. When I would find or no, any familiar stars I could tell on which hemisphere I am. That's already something, but I would want something better than this. I realized that I have phone in my pocket, which has three things which I need: camera, much computing power and accurate clock. I would have found app on smartphone named: "Astrolocator". I would fired it up and took picture of sky. After 3 hours of patient waiting I would finally know, where I am (more or less, at least I would know that I'm in Europe).
I have made program which take photo of night sky with stars and time when it was taken. This application returns approximation of your localization: latitude and longitude in degrees.

## 2 Tasks

1. Process image to find localisation of stars
2. Find in database stars which are matching system on picture
3. Convert stars positions to corresponding place on earth.

## 3 Image processing

I needed to extract from picture only localization and magnitudes of stars. First thing I made was to take the gradient of image, to help separate only stars. Than I have converted image to black and white (binary) with adjustable threshold (adjusted automatically). Next step was to reduce noise on image. To do that I used two functions from OpenCV: erode and dilate. Those are very simple functions, erode is going through image with kernel and check if under kernel is any black pixel, than checked pixel is black, otherwise white. Dilate works as opposite to erode function, if under kernel is any white than checked pixel is white. After filtering noise I used find_contours function from OpenCV, which is just finding the continuous boundary of the same color. I'm checking if I found at least 15 stars and maximum of 500 and if the number of contours is not in this range I'm changing threshold of converting

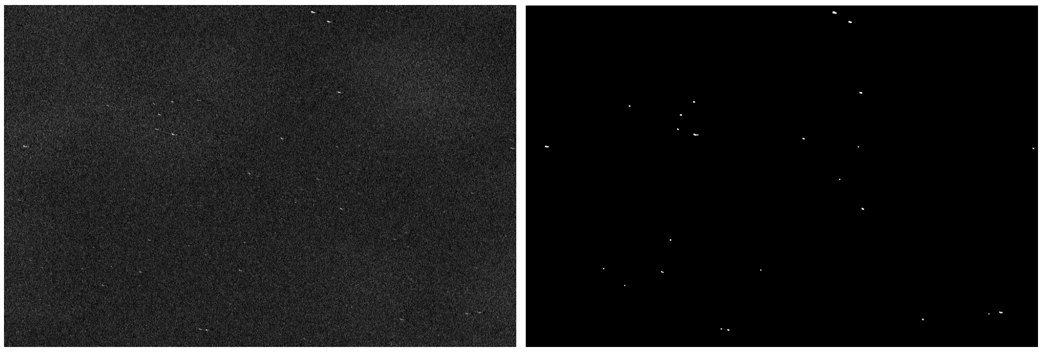
Figure 1: Gradient of image and filtered

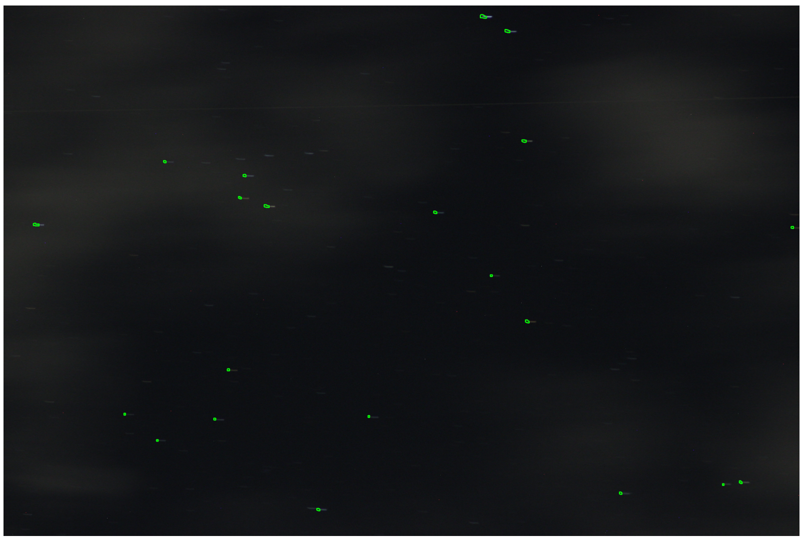
Figure 2: Found stars

to black and white. My function returns table of found stars, with position of center and magnitude of star - area of star.

## 4 Database processing

I downloaded HYG database [1] which contains almost 12000 stars. I don't need all data from database, I need only id of star, declination, right ascension and star's apparent visual magnitude. I have imported only this four columns. For calculations I needed declination and ascension in radians, but ascension is given in hours and declination in degrees from -90 to 90. After changing angles to radians I'm still have to calculate coordinate on visible sky which are just multiplying right ascension by cousins of declination[2]. I needed also magnitude in linear scale, because I wanted to check linear ratios of brightness. After converting I sorted whole database by distance from (0,0) point.

## 5 Finding stars in database

This part was the most difficult to design. I firstly imagined that I want to check the ratios of distances between three brightest stars on image and find the triangles which are similar in database. When I searched the web for similar problems I found algorithm "Lost-in-space pyramid"[3,4] which uses similar procedure. For finding more than three stars patterns I have used convex Hull and its angles.
Firstly I calculate two ratios in triangle created by three brightest stars in picture. The first ratio is the middle edge of triangle to longest one, second is the shortest to longest.
In order to find stars on bigger area I am progressively increasing number of checked stars by changing minimum magnitude of stars in database. I'm taking all elements of database at actual magnitude and shuffling them. I generate 100 neighbors of each star and get their pairs combinations. For each three stars I calculate lengths of edges in its triangle and ratios between them. Later I'm just calculating the similarity between distance ratio in stars from image and from database. If this similarity (which is just product of ratios of ratios of distances in triangle) is bigger than 99% then I'm checking the brightness similarities. For this purpose I'm just sorting the stars from database by its brightness and calculate proper ratios. If the brightness similarity is bigger than 80% then I'm looking for more stars which satisfy the scheme.
To find more then three stars I need to check something more than triangle. I decided for method using convex hull for checking the similarities of two polygons. I'm using function which make convex hull from numpy library. It is returning next vertices, which I'm changing to sorted angles with preserved order, but at the first index I put biggest angle.
I made recursive function which works until finding in my case at least 5 stars, for 6 it is working too long and I think it is overflowing something, because it can occasionally stuck. The function at first step make convex hull of four stars from image, later more in next call. Later it takes 10000 stars from raw database which are near first found star. Later it just check for each star the similarity of convex hulls, if it is greater than 90% than it is calling itself with one more star. If it finds at least minimal number of stars which we set it is returning array of its IDs.
I have tried to make function for plotting night sky, for checking of found IDs, but it didn't work well.

## 6 Localization from stars

I used method described on website [5]. I took date, counted number of days from 1 January 2000, because in this format the database was created. Calculated the ratio of number past days from 2000 minus 1.5 and days in one millenium. I have obtained the $\phi 0$ angle from equation: $\phi 0 = (100.4606 + 36000.7701 * ratio - of - days) modulo 360$. This equation takes into consideration the inregularities of Earth sphere. I have changed time to angles and changed angles from database to degrees. I have calculated the corrections of latitude and longitude and caltulated longitude and latitude from below equations:

$$longitude = rightAscencion + (phi0 - n * DeltaLongitude + (1.0027 * hours))$$

$$latitude = declination + 90 + n * DeltaLatitude$$

where n is number of years from 2000, hours are the time in degrees, DeltaLongitude and DeltaLatitude are the uncertainties.

## 7 Optimization

Convex Hull is pretty time consuming procedure, but finding only triangles works a lot faster. In future I could try implementing something similar to “Lost-in-space pyramid“ algorithm. For image recognition I could implement the finding by Gaussian function.

## 8 Summary

The project finds polygons made of stars in star database pretty good. It finds stars on image pretty well else, but finding in database take a lot of time. Unfortunately I didn’t have enough time for checking if the finding algorithm finds the proper stars. I tried plotting the fragment of night sky, but it didn’t work and the stellarium don’t have the database indexes of stars which I had, so it was very time consuming operation to check more than one star. I found that it for sure find the right polygons, because I checked two sets of found stars in Stellaris, unfortunately for pictures which I didn’t know the localization.
So at last the algorithm wasn’t very precise at location, but I observed that the latitude was in most cases accurate to even few degrees. I had problems with longitude, which very often was very weird. At least it recognized stars and find the schemes in star database pretty well.

## 9 Bibliography

[1] https://math.stackexchange.com/questions/52936/plotting-a-stars-position\-on-a-2d-map

[2] http://www.astronexus.com/hyg

[3] http://www.ntu.edu.sg/home/eechenss/Papers/conf-2012-A\%20Star\%20Pattern\%20Recognition\%20Algorithm\%20for\%20Satellite\%20Attitude\%20Determination.pdf

[4] http://www.astro.caltech.edu/~moncelsi/FTS_talk.pdf

[5] https://www.eaae-astronomy.org/WG3-SS/WorkShops/LongLatOneStar.html