Computer Vision Course Project Report
Topic: Star Matching

Wei-Chao Chen

Department of Computer Science
The University of North Carolina at Chapel Hill

Abstract

In this report, I describe the method I have used for the Computer Vision course project for Spring 2000. Given an arbitrary image of the night sky, the goal of this project is to recognize the name, constellation and other relevant informations of the stars detected on the images. The various parameters such as camera intrinsic/extrinsic parameters and location of the camera is assumed to be unknown. The algorithm first detect the location and intensity of the stars on the image, and these information are matched with other images where the names and constellation of the stars are known.

Figure 1: A snapshot of the night sky.

1 Introduction and Motivation

I am motivated by the fact that I cannot recognize the names and constellation of most of the stars on the sky. The images of the night sky is intriguing and beautiful, and it would be wonderful if there is an automatic way of recognizing their names and associate them with the databases that are already established by the astrophysical research.

In this project, the assumptions is that except for the images taken from the night sky using regular cameras, no other information are given. It is also assumed that the FOV of the camera is not very large and the sky can be treated locally as a two dimensional image. This problem thus reduce to finding a 2D affine transformation among images so that the error after the transformation is minimized. A brute-force minimization process can be done by finding the magnitude of correlation by doing a 6-dimensional search (translation, rotation and scaling). This is very computationally expensive and it is very unstable. I implemented a more efficient method based on [3] and the details are presented in the next section.

2 Methods and Steps

The algorithm can be divided into two stages, namely star detection and star matching.

2.1 Star Detection

After the images are gathered from the sky, star detection is necessary. Several methods of image processing techniques can be found at [1]. I have implemented a method that is suitable for my specific purposes of detecting bright small objects on the background.

First, a foreground and background separation is necessary. A simple thresholding on the input image generally works pretty well. For cases with ambient light disturbances, we can find the coarse fitting of the background color by finding a "local background color". This is done by finding the minimum value of a 16-by-16 block on the image and use this value as the background color of this block. This value is then used as a scaling factor on the global threshold.

After the foreground/background subtraction, the rest of the portion is treated as effective star images. A calibration on the transfer function of luminance is necessary, though not implemented. We do not wish the user to calibrate their camera, and therefore the next stage of the algorithm need to be able to handle the uncalibrated luminance condition.

Stars are assumed to have a maximum size of wxw pixels and the value of w is adjustable. At the first pass, each scanline on the image is processed and the centroid of a star is calculated with the nearby w pixels added to this centroid. Then, these centroids are combined in the y direction if they are closer than w in y axis. The position of the combined centroids is the interpolation of the original centroids weighted by their intensity. The higher the intensity, the closer the combined centroid is to the original centroid. A detection result using figure 1 is presented in figure 2.

2.2 Star Matching

The star detected from the original images are listed in a file in the format of $(x, y, i)_k$ tuples, where x, y is the centroid of the k-th detected star and i is its intensity (sum of the pixel values of the star) in logarithmic scale. Since the absolute intensity is not accurate due to the fact that no calibration on the illumination is not known, we can only rely on the order of intensity within one picture.

A traditional way of matching stars by astronomers are looking at the aspect ratio of the triangles formed by brighter stars. It is naturally reasonable to use a similar method for automatic recognition of stars. In [3], an algorithm is proposed based on the above observation and my implementation is based on this algorithm. The stars on the image are sorted according to their intensity. The brightest N stars

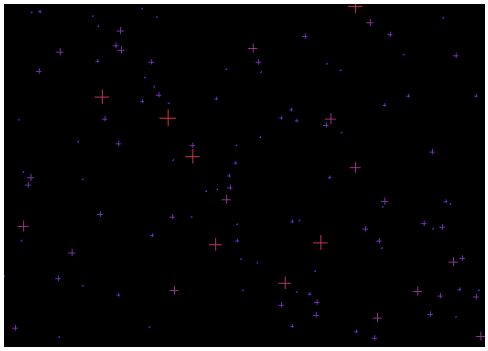

Figure 2: Stars detected from figure 1. Brighter stars are marked with larger cross-hairs.

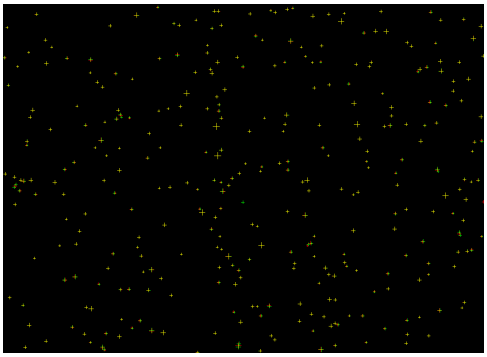

Figure 3: Matching result with pure rigid transformation. Matched star are shown in yellow color.

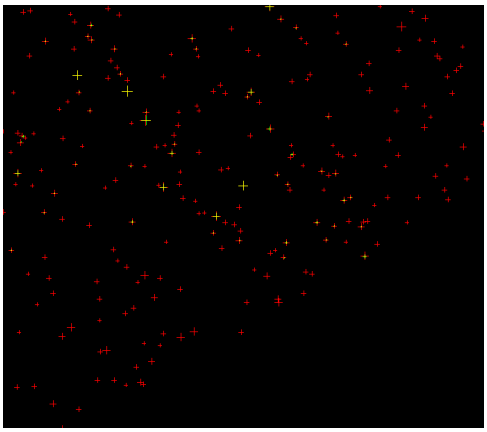

Figure 4: Matching result with rigid transformation and image clipping. Matched star are shown in yellow color.

on an image are selected, and a triangle can be formed by choosing any three of the N stars. There are thus $O(N^3)$ triangles formed by N stars. The triangle can be defined by the length of its three sides (a, b, c) where $a < b < c$. Normalizing the triangle with respect to its longest edge, each triangle can be represented in the normalized tuple space of $(a/c, b/c)$. Sorting the tuple according to their Euclidean distance to $(0, 0)$, we can get an ordered list of tuple space for each image.

After obtaining the ordered tuple space, we decide whether a triangle is matched to another triangle by calculating the Euclidean distances between two tuples. Since spurious matches are possible, the correctness of matching is decided by the number of matches found. If the distance between two tuple is less than a threshold, we increase the number of matches on the three vertices (stars) of the triangles. Ideally, each star can have up to $O(N^2)$ matches. A brute-force implementation will require $O(N^6)$ computation, but since the tuples are sorted, we do not have to search for the whole space and thus the real computation time is close to $O(N^3)$. A further optimization can be made by dividing the tuple space into regular grids and check only the tuples located on the same grid and its neighboring grids. Assume even distribution of the tuple, the complexity is reduced to $O(N^2)$.

After the matched star are found

3 Results and Conclusion

The input star images from a popular star viewing program for PC instead of real pictures, because the pictures I took from the sky do not have a lot of visible stars (wrong shutter time setting:P). The algorithms are implemented on a Pentium II PC running Linux Operating System. After star extraction, I choose between 20 to 60 brightest stars from both images for the input of the star matching algorithm.

First I tested the algorithm and generate the second source image by taking the first input image and applying rotation, scaling and translation. Figure 3 the result of matching these two images. The stars from one picture is shown in red, another in green, and matched results are in yellow. Notice that the algorithm works fairly well for this case.

Figure 4 shows another case wheter the second input image is obtained by rotating, translating, scaling and clipping of the first input image. It is clear that even though almost half of the stars do not appear in the second image, the algorithm generates correct translation and rotation for these two input images. In figure 5, one of the input images from figure 3 is added with some noise. The algorithm generates correct transformation between the two images.

For the cases with wider field-of-view lenses, we can not assume that the image can be matched by pure affine transformation. Therefore we need to test the radial distortion. I applied re-projection of the first source image using a wide FOV camera (approximately 100 degrees) and tried to match these two images. The algorithm failed to handle this case and the results are shown in figure 6.

Obviously, for the above method to be applicable for wide FOV camera lenses, we need to post-warp the image into another coordinate system. Since the stars are very far away from the earth, there are no motion parallex and thus the world coordinate system for stars can be represented in 2D. We can choose a coordinate system such as [2] and extract the stars into this coordinate system, if we know some parameters of the camera. Here, only the field-of-view of the camera is required, and thus the calibration is very easy since we can use the focal length of the camera and size of the film (or CCD) to calculate FOV, and these parameters are generally known by the user. A small amount of distortion is tolerable for the user, and thus it is not strictly required that the estimates of these two parameters are very accurate.

For a more complete implementation of the system, we

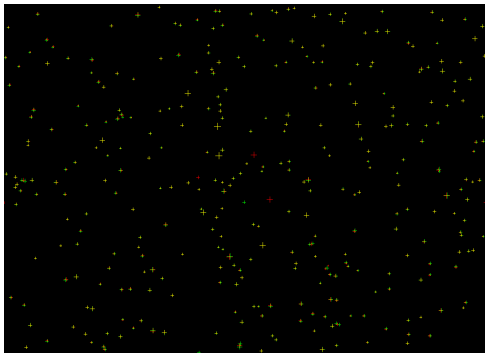

Figure 5: Matching result with some noise applied to one of the input image.

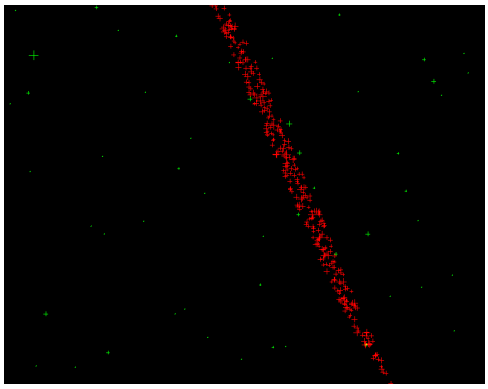

Figure 6: Matching result for image taken with wide field-of-view cameras.

need to incorporate the above post-warping of the image and the matching target should be a star catalog. Given the size of the catalog, the algorithm might run at very slow speed if the whole catalog is matched against the images taken by the user. It is feasible to divide the sky into overlapped cells of the size of the camera FOV before matching, and refine the matching further by choosing the best match among these cells and re-iterate matching.

References

[1] Rafael C. Gonzales and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1993.

[2] Eric W. Griesen and Mark Calabretta. Representations of world coordinates in fits. *Astronomy and Astrophysics*, Nov 1999.

[3] Francisco G. Valdes, Luis E. Campusano, Juan D. Velasquez, and Peter B. Stetson. Focas automatic catalog matching algorithms. *Publications of the Astronomical Society of the Pacific*, Nov 1995.