Star Matching Algorithm

Aseel Ahmad

May 2025

1 Introduction

We aim to efficiently match stars between two astronomical images:

- **Source image**: Contains hundreds of stars.
- **Target image**: Contains approximately 10-20 stars.

The challenge is accurately matching stars between these two images using a clear and efficient algorithm.

2 Algorithm Steps

The following algorithm provides a straightforward and effective approach:

2.1 Step 1: Star Detection (Hough Transform)

Detect stars in both images using the Hough Circle Transform, representing each star as a tuple:

$$(x, y, r, b)$$

where x, y are coordinates, r is radius, and b is brightness.

2.2 Step 2: Filtering Brightest Stars

Select the top 50 brightest stars from each image based on brightness b, significantly reducing the computational complexity.

2.3 Step 3: RANSAC Line Fitting

Apply the RANSAC algorithm to identify dominant linear patterns among the stars, isolating key star groups for reliable matching.

2.4 Step 4: Triangle Geometric Matching

Randomly select sets of three stars from each image, creating triangles. For each pair of triangles (one from each image):

1. Calculate the sorted side-length ratios.
2. Compare triangles based on these side-length ratios.
3. If triangles match closely (threshold < 0.25), consider these as initial match candidates.

2.5 Step 5: Affine Transformation

Compute an affine transformation matrix from matching triangle pairs using the OpenCV function `cv2.getAffineTransform`, allowing star coordinates to be mapped from the source to the target image.

2.6 Step 6: Counting Inliers

Transform each star from the source image onto the target image and count as an inlier if the transformed star closely aligns (distance $\leq$ 20 pixels) with a real star in the target image. The best match has the maximum number of inliers.

2.7 Step 7: Matching Ratio

Evaluate the quality of the match using:

$$\text{Matching Ratio} = \frac{\text{Number of Inliers}}{\min(\text{stars in source}, \text{stars in target})}$$

A high ratio indicates a robust match.

3 Algorithm Advantages

The proposed method is:

- **Simple**: Clearly defined geometric and transformation steps.
- **Efficient**: Reduces complexity via brightness filtering and geometric constraints.
- **Robust**: Uses affine transformations and validates through inlier counting.

Overview

As part of this project, I aimed to evaluate how well my star-matching algorithm performs across a diverse set of astronomical images. To do this, I conducted two main experiments:

Part 2: Detecting and Matching Stars Between Image Pairs

I began by detecting stars in four different images: `fr1.jpg`, `fr2.jpg`, `ST_db1.png`, and `ST_db2.png`.

These images vary significantly in visual characteristics—some include hundreds of stars, while others contain just a few dozen.

Using my custom star detection pipeline, I extracted features such as coordinates, radius, and brightness for each detected star and saved the results into the following text files:

- `stars_fr1.txt`
- `stars_fr2.txt`
- `stars_ST_db1.txt`
- `stars_ST_db2.txt`

I then applied my star-matching algorithm on six different image pairings. For each pair, the algorithm followed these steps:

1. Detected dominant lines of stars using RANSAC
2. Sampled triangles from these lines to compute an affine transformation
3. Mapped stars from the source image onto the destination image
4. Counted the number of valid inlier matches based on distance and brightness thresholds

Each experiment generated a file (e.g., `matches_fr1_to_fr2.txt`) that contains the list of matched star coordinates and the matching ratio.

Part 3: Large-Scale Matching Across Multiple Combinations

After validating the algorithm on basic pairs, I extended the experiment to a broader set of image combinations. This included both self-matches (like `fr1` to `fr1`) and cross-image matches (like `fr2` to `ST_db2`).

I created a loop that:

- Reads the relevant star detection files
- Applies the matching algorithm
- Saves the match results and visualization images

All results were stored in:

- `star_matching_results/` — text files showing match coordinates and ratios
- `imgs_detected/` — overlay images showing detected stars
- Visual comparison images with lines connecting matched stars

- **Self-matches** (e.g., `fr1` to `fr1`) performed nearly perfectly, confirming the algorithm's correctness.
- **Similar-image matches** (e.g., `fr1` to `fr2`) yielded strong results, with minor mismatches due to brightness variation or detection noise.
- **Cross-library matches** (e.g., `fr1` to `ST_db2`) were more difficult, with lower match ratios caused by differences in orientation, scale, and star layout.
- **Visualizations** helped verify the results and provided clear insight into where mismatches occurred.