Automated Star Identification System Using Computer Vision and Astrometry.net

[Your Name] Your Institution¿Your Institution
Your Email¿Your Email

June 20, 2025

Abstract

This paper presents an automated star identification system that detects stars in astronomical images and matches them with catalog entries. The system uses OpenCV for star detection through contour analysis, Astrometry.net API for automatic plate solving, and the Yale Bright Star Catalog (YBC5) for star identification. When automatic plate solving succeeds, the system can identify bright stars in the field. The implementation includes a fallback mechanism using filename pattern matching and a detection-only mode when coordinate determination fails. The system is implemented as both a standalone Python application and a Flask web service.

1 Introduction

Star identification is the process of matching detected stars in an astronomical image with their corresponding entries in stellar catalogs. This enables astronomers to:

- Determine which stars are visible in an image
- Obtain stellar properties like magnitude and spectral type
- Verify telescope pointing
- Assist in astronomical observations and research

This work presents a practical implementation of an automated star identification system that combines computer vision techniques with online astrometric services.

2 System Overview

The star identification system consists of three main components:

1. **Star Detection**: Uses OpenCV to find bright spots in images
2. **Astrometric Solving**: Uses Astrometry.net to determine sky coordinates
3. **Catalog Matching**: Matches detected stars with the Yale Bright Star Catalog

3 Implementation Details

3.1 Star Detection

The star detection algorithm uses basic image processing techniques:

The threshold value and area limits can be adjusted based on image characteristics.

Algorithm 1 Star Detection Process

```
1: Convert image to grayscale
2: Apply Gaussian blur (kernel size = 3)
3: Apply binary threshold (default = 70)
4: Find contours in the binary image
5: Filter contours by area (1 ¡ area ¡ 800 pixels)
6: Extract centroid and brightness for each valid contour
7: return List of star positions (x, y, radius, brightness)
```

3.2 Coordinate Determination

The system attempts to determine image coordinates using three methods in order:

1. **Astrometry.net API**: Uploads the image to Astrometry.net for automatic plate solving
2. **Filename Pattern Matching**: If automatic solving fails, checks if the filename contains known object names (e.g., ”orion”, ”m31”)
3. **Detection-Only Mode**: If both methods fail, returns only detected stars without identification

3.3 Coordinate Transformation

Once the image center coordinates are known, pixel positions are converted to celestial coordinates using:

$$\text{pixel\_scale} = \frac{2 \times \text{radius}}{\min(\text{width}, \text{height})} \tag{1}$$

$$\Delta x = (x_{\text{pixel}} - \text{width}/2) \times \text{pixel\_scale} \tag{2}$$

$$\Delta y = -(y_{\text{pixel}} - \text{height}/2) \times \text{pixel\_scale} \tag{3}$$

$$\text{RA} = \text{RA}_{\text{center}} + \frac{\Delta x}{\cos(\text{Dec}_{\text{center}})} \tag{4}$$

$$\text{Dec} = \text{Dec}_{\text{center}} + \Delta y \tag{5}$$

3.4 Star Matching

The matching algorithm compares detected stars with catalog stars:

1. Query YBC5 catalog for stars within 2× the field radius
2. For each catalog star, find the nearest detected star
3. Accept matches with angular separation ¡ 0.8°
4. Each star can only be matched once

4 Yale Bright Star Catalog Integration

The system uses the Yale Bright Star Catalog (YBC5) which contains approximately 9,000 stars brighter than magnitude 6.5. The catalog is stored in an SQLite database with the following fields:

- HR number (catalog identifier)

- Star name
- Right Ascension (RA) in degrees
- Declination (Dec) in degrees
- Visual magnitude
- Spectral type

5 Web Service Implementation

The system includes a Flask web service that provides:

- Web interface for uploading images
- Adjustable detection threshold
- Three API endpoints:
  - `/identify` - Returns annotated image with JSON metadata
  - `/identify_fast` - Returns results without matplotlib visualization
  - `/identify_raw` - Returns annotated image file directly

6 System Behavior

The system operates in three modes depending on coordinate determination success:

6.1 Full Identification Mode

When Astrometry.net successfully solves the image:

- Detected stars are shown as red circles
- Identified stars have green circles with labels
- Star names, HR numbers, and magnitudes are displayed

6.2 Fallback Mode

When Astrometry.net fails but filename matches a known pattern:

- Uses approximate coordinates for the region
- Attempts identification with reduced accuracy
- Indicates ”filename pattern” was used

6.3 Detection-Only Mode

When coordinate determination completely fails:

- Shows only detected stars as red circles
- No identification attempted
- Displays star count only

7 Limitations

The current implementation has several limitations:

1. **Catalog Limitations**: YBC5 contains only bright stars (magnitude ¡ 6.5), limiting identification to the brightest objects in most images
2. **Internet Dependency**: Requires internet connection for Astrometry.net
3. **Simple Coordinate Model**: Linear transformation assumes small field of view
4. **No Distortion Correction**: Does not account for optical distortions
5. **Single Catalog**: Uses only YBC5, missing fainter stars

8 Technical Requirements

The system requires the following Python packages:

- OpenCV (cv2) - Image processing and star detection
- NumPy - Numerical computations
- Pandas - Data manipulation
- Matplotlib - Visualization
- Astroquery - Astrometry.net API interface
- Flask - Web service (optional)
- SQLite3 - Database management

9 Usage Examples

9.1 Standalone Script

```
# Basic usage
python star_identifier.py image.jpg

# Process multiple images
for img in *.jpg; do
    python star_identifier.py "$img"
done
```

9.2 Web Service

```
# Start the server
python flask_server.py

# Upload image via web interface
# Navigate to http://localhost:5000

# Or use API
curl -X POST -F "image=@star_image.jpg" \
     -F "threshold=120" \
     http://localhost:5000/identify_fast
```

10 Future Improvements

Potential enhancements include:

1. Integration with deeper catalogs (e.g., Gaia)
2. Local plate solving to remove internet dependency
3. Proper WCS transformations for better accuracy
4. Support for FITS files with existing WCS headers
5. Improved star detection for crowded fields
6. Pattern matching algorithms for offline solving

11 Conclusion

This paper presented a practical star identification system that combines computer vision with online astrometric services. While the system has limitations due to catalog coverage and coordinate transformation simplicity, it provides a functional solution for identifying bright stars in astronomical images. The modular design allows for future improvements and extensions.

The complete source code demonstrates how modern tools and services can be combined to create useful astronomical software accessible to both amateur and professional astronomers.

Acknowledgments

We acknowledge the Astrometry.net team for providing their plate solving API, and the Yale Bright Star Catalog maintainers for this essential astronomical resource.

References

[1] Lang, D., Hogg, D.W., Mierle, K., Blanton, M., & Roweis, S. 2010, *The Astronomical Journal*, 139, 1782. "Astrometry.net: Blind astrometric calibration of arbitrary astronomical images"

[2] Hoffleit, D. & Jaschek, C. 1991, *The Bright Star Catalogue*, 5th ed., Yale University Observatory.

[3] Bradski, G. 2000, *Dr. Dobb's Journal of Software Tools*, 25, 120. "The OpenCV Library"

[4] Ginsburg, A. et al. 2019, *The Astronomical Journal*, 157, 98. "astroquery: An Astronomical Web-querying Package in Python"