Algorithmic Details of Constellation Creator

February 1, 2026
One Choi, Minseung Kim
Contact: one@choione.com

Abstract

This document presents the design and implementation of *Constellation Creator*, an interactive web application that enables users to create custom star patterns and matches them against a database of 88 real constellations using rotation-invariant shape-matching algorithms. The system employs normalization techniques to eliminate translation and scale variations, combined with the Hungarian algorithm for optimal point correspondence under rotation. The application features an intuitive canvas-based interface, a comprehensive constellation gallery with traditional illustrations from Stellarium, and a lunar phase tracking module. This project demonstrates the application of computational geometry and combinatorial optimization techniques to astronomical pattern recognition in an accessible, educational format.

1 Introduction

1.1 Project Overview

Constellation Creator is a web-based interactive system designed to bridge human creativity with astronomical knowledge. Users draw arbitrary star patterns by clicking on a canvas, and the system identifies the closest matching constellation from the celestial database through sophisticated shape-matching algorithms. This approach transforms constellation recognition from passive observation into an active learning experience.

1.2 Motivation

Traditional constellation learning relies on memorization and guided observation. This project addresses three key limitations:

- **Engagement barrier**: Static star charts fail to capture learner attention
- **Pattern recognition difficulty**: Human-drawn patterns rarely match precise astronomical coordinates
- **Educational accessibility**: Complex astronomical concepts presented without interactive feedback

By allowing users to freely create patterns and providing instant algorithmic feedback on similarity to real constellations, the system creates a feedback loop that enhances pattern recognition skills while maintaining scientific accuracy.

1.3 Core Features

1. **Interactive Drawing Canvas**: Click-based star placement with real-time connection visualization (maximum 30 stars per pattern)
2. **Shape Matching Engine**: Rotation-invariant comparison against 88 IAU-recognized constellations using the Hungarian algorithm
3. **Constellation Gallery**: Browse all constellations with stereographic projections and traditional artwork derived from Stellarium (GPL 2.0)
4. **Lunar Phase Tracker**: Real-time moon phase visualization with illumination percentage and rise/set times

2 Shape Matching Algorithm

The core technical challenge is comparing user-drawn patterns against constellation database while being invariant to rotation, translation, and scale. This section presents the mathematical framework and algorithmic solution.

2.1 Problem Formulation

Input:

- User pattern: $U = \{(x_i, y_i)\}_{i=1}^{m}$ where m is the number of user-placed stars
- Database: For each constellation k, a point set $T_k = \{(x_j, y_j)\}_{j=1}^{n_k}$ in stereographic projection

Output:

- Best matching constellation k^*
- Optimal rotation angle θ^*
- Matching score E^*

2.2 Normalization: Removing Translation and Scale

To achieve translation and scale invariance, both user input and database patterns undergo identical normalization:

Step 1 - Centroid Removal (Translation):

$$\mu = (\bar{x}, \bar{y}) = \frac{1}{N} \sum_{i=1}^{N} (x_i, y_i) \tag{1}$$

$$P_0 = \{(x_i - \bar{x}, y_i - \bar{y})\} \tag{2}$$

Step 2 - RMS Normalization (Scale):

$$s = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \|(x_i, y_i) - \mu\|^2} \tag{3}$$

$$P_{\text{norm}} = \left\{ \frac{(x_i, y_i) - \mu}{s} \right\} \tag{4}$$

This produces unit-scale, zero-centered point sets where all coordinates are dimensionless. Database patterns are pre-normalized during preprocessing, ensuring consistent comparison.

2.3 Rotation Scanning Strategy

Since rotation angle is unknown, we perform a two-stage angular search:

Coarse Stage: Scan $\theta \in [0^\circ, 360^\circ)$ with step size $\Delta\theta_{\text{coarse}} = 3^\circ$ (120 samples)
Fine Stage: Around the best angle θ_0 from coarse search, refine with $\Delta\theta_{\text{fine}} = 0.3^\circ$ in window $[\theta_0 - 5^\circ, \theta_0 + 5^\circ]$ (34 samples)

For each angle θ, user points are rotated:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{5}$$

2.4 Optimal Point Correspondence via Hungarian Algorithm

Given rotated user points $S = \{s_i\}_{i=1}^{m}$ and database constellation $T = \{t_j\}_{j=1}^{n}$, we seek the optimal 1-to-1 point matching that minimizes total distance.

2.4.1 Cost Matrix Construction

Let $K = \max(m, n)$. Construct a $K \times K$ cost matrix C where:

$$C[i][j] = \begin{cases} \|s_i - t_j\|_2 & \text{if } i < m \text{ and } j < n \text{ (real-to-real)} \\ \lambda & \text{if } i < m \text{ and } j \geq n \text{ (user outlier)} \\ \lambda & \text{if } i \geq m \text{ and } j < n \text{ (database missing)} \\ 0 & \text{if } i \geq m \text{ and } j \geq n \text{ (dummy-to-dummy)} \end{cases} \tag{6}$$

The penalty parameter λ represents the cost of leaving a point unmatched. This is computed adaptively per constellation:

$$\lambda = \lambda_{\text{factor}} \cdot d_{\text{med}} \tag{7}$$

where d_{med} is the median nearest-neighbor distance in the normalized constellation, and $\lambda_{\text{factor}} = 2.5$ (empirically determined). This ensures λ reflects the typical inter-star spacing: if a match would require distance exceeding $2.5 \times d_{\text{med}}$, it's better to treat the point as unmatched.

2.4.2 Hungarian Algorithm for Minimum Assignment

The Hungarian algorithm (also known as the Kuhn-Munkres algorithm) solves the linear assignment problem in $O(K^3)$ time. It finds the permutation $\pi : \{1, \ldots, K\} \to \{1, \ldots, K\}$ that minimizes:

$$\text{cost} = \sum_{i=1}^{K} C[i][\pi(i)] \tag{8}$$

The algorithm maintains dual variables (potentials) u_i and v_j and iteratively augments the matching until all rows are assigned. The key property is that it guarantees optimality: no other assignment can achieve lower total cost.

2.4.3 Error Metric

To compare scores across different constellations (with varying K), we normalize:

$$E(\theta) = \frac{\text{cost}}{K} \tag{9}$$

For each constellation k and each rotation angle θ, we compute $E_k(\theta)$. The global minimum determines the best match:

$$(k^*, \theta^*) = \arg\min_{k,\theta} E_k(\theta) \tag{10}$$

2.5 Algorithm Summary

Input: User points U_{raw}, Database $\{T_k\}$ with precomputed $\{d_{\text{med},k}\}$
Output: Best constellation k^*, rotation θ^*, error E^*

$U_{\text{norm}} \leftarrow \text{Normalize}(U_{\text{raw}})$;
best $\leftarrow \{\text{error} : +\infty\}$;
for *each constellation* k **do**
  $T \leftarrow T_{k,\text{norm}}$;
  $\lambda \leftarrow \text{clamp}(2.5 \cdot d_{\text{med},k}, 0.05, 0.5)$;
  `// Coarse scan`
  $(\theta_0, E_0) \leftarrow \text{ScanAngles}(U_{\text{norm}}, T, \lambda, 3^\circ, [0^\circ, 360^\circ))$;
  `// Fine refinement`
  $(\theta_{\text{best}}, E_{\text{best}}) \leftarrow \text{ScanAngles}(U_{\text{norm}}, T, \lambda, 0.3^\circ, [\theta_0 - 5^\circ, \theta_0 + 5^\circ])$;
  if $E_{best} < best.error$ **then**
    best $\leftarrow \{k : k, \theta : \theta_{\text{best}}, \text{error} : E_{\text{best}}\}$;
  end
end
return *best*;

Function *ScanAngles*$(S, T, \lambda, \Delta\theta, range)$**:**
  $\theta_{\text{opt}} \leftarrow \text{null},\ E_{\text{min}} \leftarrow +\infty$;
  for θ *in range with step* $\Delta\theta$ **do**
    $S' \leftarrow \text{Rotate}(S, \theta)$;
    $C \leftarrow \text{BuildCostMatrix}(S', T, \lambda)$;
    $(\text{assignment}, \text{cost}) \leftarrow \text{Hungarian}(C)$;
    $E \leftarrow \text{cost}/K$;
    if $E < E_{min}$ **then**
      $E_{\text{min}} \leftarrow E,\ \theta_{\text{opt}} \leftarrow \theta$;
    end
  end
  return (θ_{opt}, E_{min});

Algorithm 1: Constellation Shape Matching

3 Database Structure

3.1 Data Source and Format

The constellation database is derived from Stellarium's open-source astronomical catalog (GPL 2.0 licensed). Each constellation is represented in stereographic projection coordinates, which preserve angular relationships while mapping the celestial sphere onto a 2D plane.

3.2 Schema Design

Table 1: Constellation Metadata

Field	Type	Description
constellation_id	string	IAU 3-letter code (e.g., "And", "Aql")
name	string	Full name (e.g., "Andromeda")
center_ra	float	Right ascension of centroid (degrees)
center_dec	float	Declination of centroid (degrees)
n_stars	int	Number of stars in pattern
median_nn_dist	float	Precomputed for penalty λ

Table 2: Constellation Stars

Field	Type	Description
constellation_id	string	Foreign key to constellation
hip	int	Hipparcos catalog number
ra	float	Right ascension (degrees)
dec	float	Declination (degrees)
x_stereo	float	Stereographic x (normalized)
y_stereo	float	Stereographic y (normalized)

3.3 Preprocessing Pipeline

Input: Raw CSV with columns: `constellation, name, HIP, ra, dec, x_stereo, y_stereo`
Steps:

1. Group stars by `constellation_id`
2. Apply normalization (Section 2.2) to each constellation's $(x_{\text{stereo}}, y_{\text{stereo}})$ coordinates
3. Compute median nearest-neighbor distance d_{med} for adaptive penalty
4. Store normalized coordinates and statistics in runtime-optimized format (JSON)

This preprocessing ensures $O(1)$ lookup during matching and eliminates redundant normalization at runtime.

4 User Interface and Experience

4.1 Interactive Canvas

The primary interface is an HTML5 canvas where users create star patterns through mouse clicks. Key design decisions:

- **Star limit:** 30 stars maximum to balance expressiveness with computational efficiency (typical constellations range from 5-15 stars)
- **Visual feedback:** Stars are connected sequentially with lines, showing the pattern formation in real-time
- **Real-time counter:** Displays current star count prominently
- **Responsive design:** Canvas scales to viewport while maintaining aspect ratio

4.2 Matching Results Display

Upon submission, the system displays:

- Loading animation with progress indicator
- Top 3-5 candidate constellations ranked by matching score
- For each candidate: name, similarity percentage, and traditional illustration overlay
- Interactive selection to view detailed constellation information

4.3 Constellation Gallery

The gallery module provides educational context with:

- Searchable sidebar listing all 88 constellations
- Canvas visualization showing constellation stars in stereographic projection
- Toggleable illustration overlay from Stellarium artwork
- Metadata display: abbreviation, hemisphere, best viewing season

4.4 Lunar Phase Tracker

An auxiliary feature using the SunCalc library to compute:

- Current moon phase with high-fidelity shadow rendering
- Moon age (days since new moon) and illumination percentage
- Moonrise and moonset times based on user's geolocation
- Monthly calendar view showing phase progression

5 Implementation Notes

5.1 Technology Stack

- **Frontend:** Vanilla JavaScript (ES6+), HTML5 Canvas API, CSS3
- **Algorithm:** Custom Hungarian implementation (JavaScript), approximately 200 lines
- **Data Format:** JSON for constellation database, CSV for preprocessing
- **Server:** Node.js with Express for local development (static hosting for production)

5.2 Performance Considerations

- **Preprocessing:** All normalizations computed offline; runtime loads pre-normalized data
- **Angular resolution:** Two-stage scanning reduces computation from 1200 angles (0.3° steps) to 154 angles per constellation
- **Total complexity:** For $N = 88$ constellations, $A = 154$ angles, $K \approx 20$ average constellation size: $O(NAK^3) \approx 88 \times 154 \times 8000 = 108M$ operations, completing in < 2 seconds on modern browsers

6 Conclusion and Future Enhancements

Constellation Creator successfully demonstrates that sophisticated pattern recognition algorithms can be made accessible through intuitive web interfaces. The Hungarian algorithm provides mathematically optimal matchings, while careful normalization ensures rotation, translation, and scale invariance.

6.1 Key Achievements

- Robust shape matching handling arbitrary user input patterns
- Educational tool bridging computational geometry and astronomy
- Real-time performance despite combinatorial optimization complexity

6.2 Future Directions

1. **3D Celestial Sphere:** Extend to full 3D coordinates using quaternions for rotation-invariant matching on the sphere itself
2. **Machine Learning Enhancement:** Train a neural network on user patterns to predict constellation likelihood before exhaustive search
3. **Collaborative Features:** Allow users to share custom constellations and build a community database
4. **Augmented Reality:** Integrate smartphone camera to overlay identified constellations onto the night sky in real-time
5. **Accessibility:** Add screen reader support and alternative input methods for visually impaired users

7 References

[1] Stellarium Development Team. *Stellarium: Open Source Planetarium*. GNU General Public License v2.0. https://github.com/Stellarium/stellarium

[2] Kuhn, H. W. (1955). "The Hungarian method for the assignment problem." *Naval Research Logistics Quarterly*, 2(1-2), 83-97.

[3] Munkres, J. (1957). "Algorithms for the assignment and transportation problems." *Journal of the Society for Industrial and Applied Mathematics*, 5(1), 32-38.

[4] Umeyama, S. (1991). "Least-squares estimation of transformation parameters between two point patterns." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4), 376-380.

[5] International Astronomical Union. *IAU Constellation Boundaries*. https://www.iau.org/public/themes/constellations/

[6] SunCalc Library. "JavaScript library for calculating sun/moon positions and phases." https://github.com/mourner/suncalc

---

Project Source: https://github.com/choione07/choione-constellation
Webpage Link: https://constellation88.com