

Celestia: A Graph Neural Network Approach for Constellation Identification and Learning

Daria Crăciun

Faculty of Computer Science
“Alexandru Ioan Cuza”
University,
Iași, Romania
daria.craciun411@gmail.com

Elena Opait

Faculty of Computer Science
“Alexandru Ioan Cuza”
University,
Iași, Romania
opaitelena@gmail.com

Adrian Iftene

Faculty of Computer Science
“Alexandru Ioan Cuza”
University,
Iași, Romania
adiftene@gmail.com

ABSTRACT

Identifying the 88 officially recognized constellations combines astronomical knowledge with pattern recognition skills that are not easily accessible to non-experts. Existing tools either rely on deterministic catalog-matching algorithms that are precise but slow and expert-oriented, or on convolutional classifiers that struggle with the irregular, non-Euclidean geometry formed by stars of the same constellation. This paper presents Celestia, a web application that pairs a Graph Isomorphism Network with Edge features (GINE) with an interactive real-time sky map. Stars detected in an uploaded image become a graph whose nodes encode rotation-invariant structural features and whose edges encode angular distances, classified into one of 88 classes. The model reaches 99.3% accuracy on a held-out test set, although both training and evaluation rely on synthetically generated imagery, so this figure characterises the model under the assumptions of the generative pipeline rather than its accuracy on real photographs, which remains to be established. A single upload triggers the learned classifier and a catalog-based service in parallel and shows both side by side, turning a latency asymmetry into an interpretable comparison. A usability study with ten participants indicates that the sky map and the identification feature were the most appreciated components.

AUTHOR KEYWORDS

Graph Neural Networks; constellation identification; astronomy education; human-computer interaction; usability evaluation.

DOI: 10.37789/icusi.2026.16

INTRODUCTION

Astronomers divide the entire sky into 88 officially recognized constellations, and identifying them is a fundamental step in understanding the structure of the night sky. For a non-expert observer, however, moving from a photograph of the sky to the name of what is in it remains difficult, and the tools that do this well are built for specialists. Identifying constellations in astronomical images has traditionally relied on algorithms that match observations against catalogs such as Hipparcos and Tycho [4, 9] or, more recently, Gaia [5]. These underpin a mature family of star identification techniques – grid, triangle and pyramid algorithms [16–18] and blind plate solvers such as Astrometry.net [13]. Such methods do not learn from data: they apply a deterministic sequence of steps based on geometry, photometry, and optical calibration, which makes them sensitive to image noise and slow on the data volumes of modern astronomy. Convolutional

networks, the usual learned alternative, assume a regular grid and a fixed number of neighbours, which does not reflect the irregular distribution of stars. Graph Neural Networks operate naturally on structures with a variable number of nodes, are invariant to permutations, rotations and scale, and learn geometric patterns directly [1], making them a better fit when data is incomplete, noisy or occluded. This paper presents **Celestia**, an application that analyses sky photographs, separates bright points from background noise, builds a graph of the spatial relationships between detected stars, and classifies them with a GNN. Alongside identification, it provides a real-time sky simulation based on the user’s GPS coordinates. Its contribution to user-system interaction lies less in the classifier than in the decisions that make it usable by non-experts: (i) a single upload triggering a fast learned prediction and a slow catalog-based one in parallel, displayed side by side; (ii) a representation based on three graph topologies and 20 rotation-invariant node features, chosen for robustness to the imperfect star detections produced by consumer photographs rather than calibrated instruments; (iii) a GPS-aware 3D sky map that doubles as the navigation entry point; and (iv) an empirical account of how these decisions were received.

EXISTING APPLICATIONS

In addition to applications presented in [3, 8], we tested three representative tools with images of Cancer, Gemini, and Orion at different levels of background noise, taken from Stellarium¹. **Nyckel**² is a no-code platform with a pretrained constellation classifier; it identified Orion (72.5%) and Gemini (67.5%) but misclassified Cancer as Virgo (62.5%) – accessible and interpretable, but not accurate enough. **Stellarium** simulates a planetarium with high astronomical accuracy and displays constellation lines intuitively, but its many configuration options (telescope, location, viewing angle, grids) make it hard for beginners, and it performs no identification at all: it renders the sky, leaving the matching step to the user. **Astrometry.net** [13] identifies celestial objects against star catalogs and returns very precise coordinates, pixel scale and orientation, but takes up to ten minutes per image (five on average for our three) and produces output that is hard to interpret without domain knowledge. Table 1 compares these systems with Celestia along the dimensions that matter for a non-expert: what the tool actually returns, whether that output is interpretable without domain knowledge, and whether the tool also supports learning. The design space is split: Stellarium is excellent for learning but performs no identification;

¹<https://stellarium-web.org/>

²<https://www.nyckel.com/pretrained-classifiers/constellations/>

	Nyckel	Stellarium	Astrometry	Celestia
Identifies from photo	Yes	No	Yes	Yes
Primary output	Label	Rendered sky	Plate solution	Label + top-5
Interpretable by novices	Yes	Partly	No	Yes
Supports learning	No	Yes	No	Yes
Location-aware sky map	No	Yes	No	Yes
Free, no install	Yes	Yes	Yes	Yes

Table 1: Critical comparison of Celestia with the reviewed applications.

Astrometry.net is precise but slow and expert-oriented; Nyckel is fast but unreliable. Celestia is positioned deliberately in the gap – it keeps the fast response and low-friction upload of a learned classifier, adds the catalog-based result as a parallel second opinion rather than a replacement, and embeds both inside a learning environment.

PROPOSED SOLUTION

Dataset and Star Detection

The public *NeuML/constellations* dataset³ alone proved insufficient: a star detector applied to a real photograph finds every sufficiently bright star in the field of view, not only those of the target constellation, so a model trained on constellation stars alone performed poorly on realistic inputs. The final dataset combines its positions with the HYG v4.1 catalog⁴ (over 100,000 stars), from which all stars brighter than magnitude 6.5 within the 60° field of view are extracted as field stars. Synthetic images are produced by projecting these stars onto the plane tangent to the celestial sphere, rendering each with an intensity proportional to its magnitude, and adding Gaussian background noise. Star detection uses OpenCV and follows a standard sequence [6]. A 5×5 **Gaussian blur** suppresses sensor and sky noise so that isolated noisy pixels are not mistaken for faint stars, while preserving the compact peaks of genuine ones. **Local-maxima detection** keeps only pixels that are the maximum of their neighbourhood, turning the intensity surface into discrete candidates. **Adaptive thresholding** recomputes the cut-off per image from the mean and standard deviation of its intensity distribution, so bright-sky and dark-sky photographs receive different thresholds – this is what makes the detector transferable across shooting conditions. **Sub-pixel centroid refinement** recomputes each position as the intensity-weighted centre of mass over a 7×7 patch; because the downstream features are angular distances, a one-pixel error propagates directly into the graph. **Non-Maximum Suppression** accepts candidates greedily in decreasing order of brightness, keeping one only if no accepted candidate lies within 14 pixels, which removes the multiple detections a saturated star generates. Finally, **top- N selection** retains the N brightest candidates, bounding graph size. Pixel coordinates are re-projected onto the unit sphere via the inverse gnomonic projection [2] and intensities normalized to [0.50, 0.84].

Graph Representation and Node Features

Each constellation is an undirected graph in PyTorch Geometric. A node is initially described by four values – its 3D coordinates and brightness – but absolute coordinates carry the arbitrary orientation of the photograph, so this vector is expanded into 20 features that are invariant to rotation, derived only from brightness and from graph structure: the

normalized brightness (1 feature); *local connectivity* (3): degree normalized by the maximum degree, absolute degree scaled by 10, and the ratio to the mean degree; *local geometry* (5): mean, minimum, maximum and variance of the angular distances to adjacent stars, each normalized by π , plus the mean angle between incident edge pairs – computed from angles rather than coordinates, which is precisely what makes them orientation-independent; *graph-theoretic position* (3): clustering coefficient, eccentricity normalized by graph diameter, and closeness centrality, all from breadth-first search; and *global descriptors* (8): node and edge count, mean, standard deviation and maximum brightness, density, diameter and the leading adjacency eigenvalue, computed once per graph and replicated on every node so each node sees the global shape context during message passing. Edges carry a single attribute, the angular distance between the two stars.

Three topologies improve robustness to noisy detections: the official IAU edges, a full Delaunay triangulation, and a Minimum Spanning Tree built with Prim’s algorithm. Augmentation adds random 3D rotation (via QR decomposition), Gaussian brightness noise, and random star subset selection. The dataset is split 70/15/15, stratified per constellation.

Model and Architecture

The classifier is a **GINE** network [10], chosen because sum aggregation preserves more structural information than the mean aggregation of standard GCNs [12], because GIN-based models are among the most expressive GNNs [19], and because GINE incorporates edge attributes into message passing – essential here, since angular distances are informative. It has 4 layers with hidden dimension 256, 20 node features and 1 edge feature. Sum, max, and mean pooling after each layer are concatenated across all 4 layers (Jumping Knowledge [20]) into a 3072-dimensional embedding, passed to an MLP classifier (dropout 0.15) over the 88 classes. Training used Focal Loss [14] and Adam [11] (learning rate 10^{-3} , weight decay 5×10^{-4} , 10-epoch warm-up), up to 400 epochs, batch size 64, early stopping after 60 epochs without improvement.

Celestia is deployed as three independent layers communicating over REST (Figure 1): a **React** frontend, a **Spring Boot** (Java 21) backend with **MongoDB** storage, and a **Flask** microservice hosting the GNN, kept separate from the JVM backend because of its Python dependencies (PyTorch, PyTorch Geometric, OpenCV). The Flask service exposes `GET /health` and `POST /predict/all`, the latter invoking the model and Astrometry.net in parallel. Model weights are loaded once at start-up, so loading does not contribute to per-request latency.

User Interaction

The frontend is a single-page application built around five scenarios: registration, identification from a photo, sky map exploration, constellation detail pages, and analysis history. On the **identification page**, the interaction is reduced to one action – uploading a photo from the camera or the file system. Behind it, two requests run in parallel: the GNN service returns the predicted constellation, its confidence, and the top-5 predictions, while Astrometry.net returns catalog-matched objects. Rather than forcing a choice of method upfront, both results stream back and are displayed side by side as they arrive, turning a technical limitation into a transparent part of

³<https://huggingface.co/datasets/NeuML/constellations>

⁴<https://www.astronexus.com/projects/hyg>

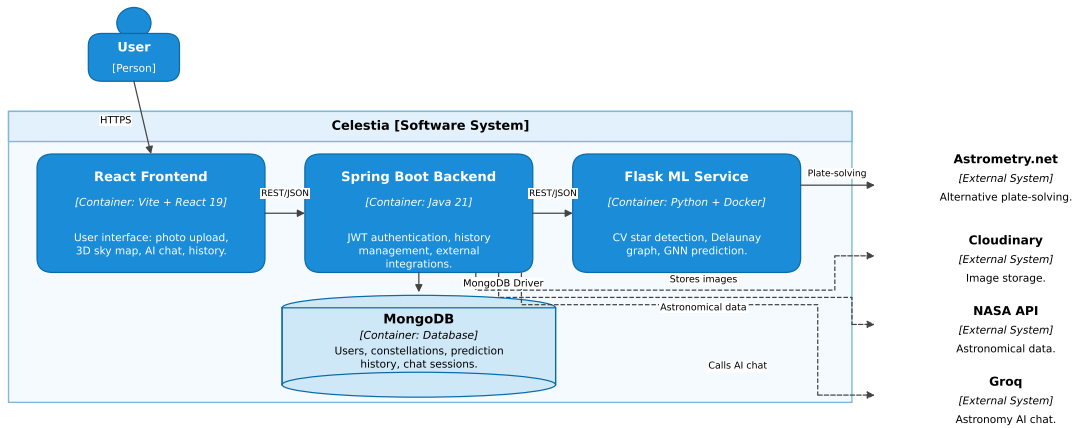


Figure 1: Container-level (C2) architecture of Celestia: frontend, backend, database and deep learning service communicate over REST/JSON.

the experience instead of a hidden delay. The most interaction-heavy component is the GPS-aware **3D sky map**, rendered with Three.js. It requests the user’s coordinates and, if unavailable, silently falls back to Bucharest rather than blocking with an error. Users drag to rotate and hover over stars for name, magnitude, and spectral class; clicking a constellation label opens its **detail page**, so the map is simultaneously a visualization and the navigation entry point. The detail page shows a diagram and NASA imagery beside structured information (astronomical data, mythology, nearby objects); rather than exposing raw API output, this content passes through LLaMA 3.3 70B [15] served via Groq [7], and a floating *Ask AI* button allows follow-up questions without leaving the page. Authentication is optional and exists only to enable the history view.

EVALUATION

Classification Accuracy

On the held-out test set (7,432 graphs) the model reached 99.3% accuracy at epoch 370, without triggering early stopping and with no visible overfitting. This is an upper bound under the generative assumptions of the synthetic pipeline, not a field accuracy. On synthetic images the model identified Orion (98.1%), Cygnus (98.6%) and Ursa Major (95.6%) correctly, but misclassified Cassiopeia as Aries (23.5%), illustrating its sensitivity to constellations with few stars and rotation-sensitive shapes (e.g. Centaurus, Telescopium, Microscopium). Placing this number in context requires care, in two distinct ways. First, the published results for classical star identification are not measured on the same task: grid and pyramid algorithms [16, 17] report identification rates above 99%, but on calibrated star-tracker imagery with a known field of view, solving a catalog-matching rather than an 88-class shape classification problem, and Astrometry.net likewise solves for a plate solution rather than a constellation label [13]. The directly comparable baseline is a learned classifier on the same inputs: on three Stellarium images, the pretrained Nyckel classifier was correct on 2 of 3, with confidences between 62.5% and 72.5%.

Validity of the Evaluation

Second, and more importantly, our 99.3% is measured on the same synthetic pipeline that produced the training data. Both

sets are generated by projecting catalog stars onto a tangent plane with a fixed 60° field of view, rendering intensities as a function of magnitude, and adding Gaussian background noise. The test split is therefore held out in the statistical sense but not in the distributional one, and the figure should be read as an upper bound characterising the model under the assumptions of the generator – not as the accuracy a user would experience. We have not yet established what that accuracy is. Real photographs differ from the generator along at least three axes that the model is known to be sensitive to: the field of view is rarely exactly 60° , which rescales every angular distance; light pollution and exposure change the brightness distribution that feeds the normalization step; and lens distortion perturbs the geometry that the inverse gnomonic projection assumes. Since permutation feature importance identifies brightness (0.784), *min_dist* (0.774) and *density* (0.464) as the three dominant features – all three directly affected by those differences – a substantial drop under real conditions should be expected rather than treated as unlikely. Quantifying it on an annotated set of real photographs is the necessary next step, and the single most important one for the system.

Which Design Decisions Matter

Permutation feature importance also indicates which parts of the representation carry the prediction. Beyond the three features above, *mean_dist* (0.410) and *mean_angle* (0.379) contribute meaningfully, while *eccentricity*, *diameter* and *bright_max* are close to negligible and could be removed. This mirrors how a human observer recognizes a constellation – from the brightest stars and their closest spacing rather than from global path-length statistics – and suggests the feature set can be reduced without loss. We report this analysis in place of a full ablation over the three graph topologies and the Jumping Knowledge pooling, which would require retraining the model once per configuration and which we have not yet carried out.

Usability Study

We distributed a Google Forms questionnaire to **10 participants** aged 21–57, with varying astronomical expertise (50% beginners, 20% advanced) and limited prior experience with similar applications (20%). Participants completed the five

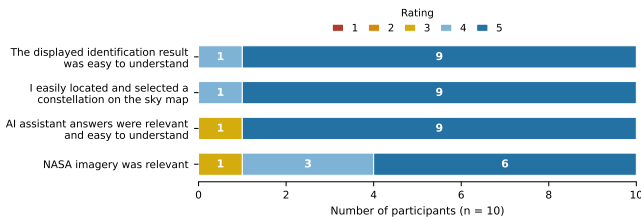


Figure 2: Usability ratings for the four items where the responses were not unanimous (1 = poor, 5 = excellent, $n = 10$).

task-based scenarios above, each rated 1 (poor) to 5 (excellent), followed by general-impression questions. Every item received a 5 from every participant except the four shown in Figure 2. Two show minor friction: one participant rated the clarity of the GNN result and the process of locating a constellation on the map with a 4. The two larger discrepancies concern content rather than interaction: one participant rated the AI assistant’s answers with a 3, and the NASA-sourced images were rated 3 by one participant and 4 by three more (mean 4.5), indicating that retrieved images are sometimes poorly aligned with the constellation. The 3D sky map was the most preferred feature, followed by the AI assistant and the prediction model; participants asked for more map detail and stricter image filtering.

CONCLUSION

This paper presented Celestia, an application for identifying constellations from astronomical images with a GINE graph neural network, integrated into a learning platform. The approach reached 99.3% accuracy on a held-out synthetic test set. Beyond the classifier, its contribution to user-system interaction lies in how the model is surfaced: a single upload producing two independent, side-by-side identifications with very different latency and reliability profiles, and a sky map acting simultaneously as visualization and navigation layer – the feature participants appreciated most.

The main limitation is that the system has been evaluated only on synthetic imagery, so its accuracy under real photographic conditions is currently unknown; the work should be read as a prototype demonstrating that a graph-based representation is learnable for this task, not as a validated identification tool. Three further limitations are visible already within the synthetic regime: the model is sensitive to constellations with few stars or rotation-sensitive shapes, of which Cassiopeia’s W is the clearest case; it depends on the 60° field of view used in training; and errors propagate from the computer-vision stage, since a constellation whose principal stars are

not detected cannot be recovered by the classifier. Assembling an annotated set of real photographs, measuring the resulting accuracy, and reporting the run-time cost of the identification service are the immediate priorities.

References

- [1] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.
- [2] Mark R. Calabretta and Eric W. Greisen. 2002. Representations of Celestial Coordinates in FITS. *Astronomy & Astrophysics* 395 (2002), 1077–1122.
- [3] Irina Cercel and Adrian Ifte. 2021. Planetarium - An Augmented Reality Application. In *Proceedings of the Conference on Mathematical Foundations of Informatics (MFOI2020)*. Kyiv, Ukraine, 62–77.
- [4] ESA. 1997. *The Hipparcos and Tycho Catalogues*. European Space Agency, Noordwijk, The Netherlands.
- [5] Gaia Collaboration, Timo Prusti, Jos H. J. de Bruijne, and Anthony G. A. Brown. 2016. The Gaia Mission. *Astronomy & Astrophysics* 595 (2016), A1.
- [6] Rafael C. Gonzalez and Richard E. Woods. 2018. *Digital Image Processing* (4th ed.). Pearson, USA.
- [7] Groq Inc. 2025. GroqCloud: Low-Latency Inference for Open Large Language Models. <https://groq.com/>. Accessed: 2026-08-01.
- [8] Diana-Elena Gușă, Adrian Ifte, and Daniela Gifu. 2019. Solar System Explorer. In *Proceedings of the 5th Conference on Mathematical Foundations of Informatics*. Iași, Romania, 295–304.
- [9] Erik Høg, Claus Fabricius, Valeri V. Makarov, Sean Urban, Thomas Corbin, Gary Wycoff, Ulrich Bastian, Peter Schwemkendiek, and Andreas Wicenec. 2000. The Tycho-2 Catalogue of the 2.5 Million Brightest Stars. *Astronomy & Astrophysics* 355 (2000), L27–L30.
- [10] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations (ICLR)*. ICLR, New Orleans, Louisiana, US, 1–22.
- [11] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*. ICLR, San Diego, US, 1–13.
- [12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.
- [13] Dustin Lang, David W. Hogg, Keir Mierle, Michael Blanton, and Sam Roweis. 2010. Astrometry.net: Blind Astrometric Calibration of Arbitrary Astronomical Images. *The Astronomical Journal* 139, 5 (2010), 1782–1800.
- [14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2980–2988.
- [15] Llama Team, AI @ Meta. 2024. The Llama 3 Herd of Models. arXiv:2407.21783.
- [16] Daniele Mortari, Malak A. Samaan, Christian Brucoleri, and John L. Junkins. 2004. The Pyramid Star Identification Technique. *Navigation* 51, 3 (2004), 171–183.
- [17] Curtis Padgett and Kenneth Kreutz-Delgado. 1997. A Grid Algorithm for Autonomous Star Identification. *IEEE Trans. Aerospace Electron. Systems* 33, 1 (1997), 202–213.
- [18] Benjamin B. Spratling and Daniele Mortari. 2009. A Survey on Star Identification Algorithms. *Algorithms* 2, 1 (2009), 93–107.
- [19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations (ICLR)*.
- [20] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *International Conference on Machine Learning (ICML)*.