

Kolamkala: A Computational Framework for Generation and Analysis of Traditional Kolam Patterns Using Rule-Based Design and Classical Image Processing

Dr. Sandeep Kulkarni¹, Palak Kolambe², Ashvitha Sree³, Kirti Rasal⁴

¹ Assistant Professor, Department of Computer Science, Pune, Maharashtra

² B.Tech Student, Department of Computer Science

Ajeenkya DY Patil University Lohegaon, Airport Rd, Charholi Budruk, Pune, Maharashtra

Abstract - Traditional Kolam (also known as rangoli or muggu) is a culturally significant geometric art form widely practiced in South India, characterized by symmetrical arrangements of dots, lines, and curves [1]. This paper presents Kolamkala, a web-based computational system designed for both the generation and analysis of Kolam patterns using a fully rule-based approach. The generation module employs coordinate-based logic on a user-defined dot grid to construct patterns that satisfy symmetry and continuity constraints. In parallel, the analysis module utilizes classical image processing techniques, including grayscale conversion, Gaussian smoothing, Canny edge detection, and contour extraction, to identify structural features such as symmetry, line patterns, and dot arrangements in uploaded Kolam images. The system is implemented using Python (FastAPI), NumPy, OpenCV, and visualization tools such as Matplotlib and SVG, along with a responsive frontend interface. By combining cultural heritage with computational methods, Kolamkala provides an interactive and explainable platform for exploring traditional designs while contributing to the digital preservation and educational understanding of Kolam patterns.

Key Words: Kolam patterns, pattern generation, image processing, OpenCV, rule-based system, symmetry detection, cultural heritage computing.

1. INTRODUCTION

Kolam is a traditional form of floor art practiced across South India, typically drawn using rice flour at the entrance of homes as a symbol of prosperity, harmony, and cultural identity [2]. These designs are composed of structured arrangements of dots and continuous lines, forming intricate geometric patterns that often exhibit strong symmetry [3]. Beyond their aesthetic value, Kolam patterns reflect underlying mathematical principles, including spatial organization, symmetry, and rule-based construction [2].

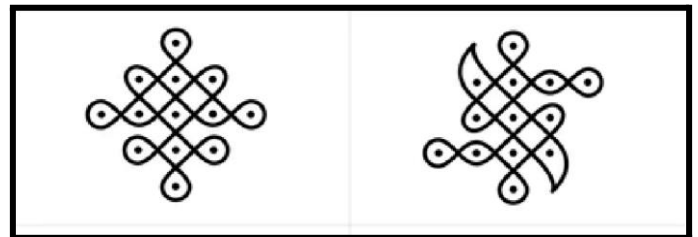


Fig -1: Example of a traditional Kolam pattern drawn using a grid of dots and continuous lines.

In traditional practice, Kolam designs are created using a grid of dots, around which lines are drawn in a continuous manner to form closed patterns. These designs frequently follow strict geometric constraints, such as enclosing all dots, maintaining continuity without breaks, and preserving symmetry across one or more axes. As a result, Kolam can be interpreted as a structured system governed by implicit mathematical rules [2]. Despite its cultural and mathematical richness, the computational representation of Kolam remains limited. Existing digital approaches primarily focus on either pattern recognition or isolated generation techniques, often relying on complex models that lack interpretability. Furthermore, there is a lack of accessible systems that allow users to both generate and analyze Kolam designs within a single framework. To address this gap, this paper introduces Kolamkala, a web-based system that integrates pattern generation and image analysis using rule-based and explainable methods. The system enables users to create Kolam patterns through customizable grid parameters and analyze existing designs using classical image processing techniques. By bridging traditional art and computational logic, Kolamkala offers a practical and educational platform for understanding and preserving Kolam design principles.

2. LITERATURE REVIEW

The study of Kolam patterns has attracted significant attention due to their unique combination of cultural relevance and mathematical structure. Kolam designs are traditionally created using grids of dots connected by

continuous lines and curves, often following strict geometric and symmetry-based rules. Early research has emphasized that Kolam patterns can be understood as structured systems governed by rule-based transformations, where complex designs emerge from simple geometric principles such as symmetry, repetition, and spatial organization. Several studies have explored algorithmic approaches for Kolam generation. Foundational work demonstrated that Kolam patterns can be represented using numerical sequences and grid-based constructions, highlighting their inherently procedural nature. Building on this, more recent methods have proposed rule-driven algorithms to generate one-stroke Kolam patterns by guiding line paths around predefined dot grids [1]. These approaches ensure 2 continuity and symmetry through deterministic rules, making them reliable and interpretable. However, such methods are often limited to specific Kolam styles and may lack flexibility when dealing with more complex or varied patterns. Additionally, many of these systems focus solely on generation without providing mechanisms for analyzing or validating the resulting designs. In parallel, research in computer vision has addressed the analysis of Kolam images. Classical image processing techniques, such as edge detection and contour extraction, have been used to identify structural elements including lines, loops, and enclosed regions. These approaches offer clear geometric interpretation, as each detected feature corresponds directly to a visual component of the Kolam. Some studies have extended this by combining handcrafted features, such as symmetry measures and shape descriptors, with machine learning models for pattern classification. While these hybrid systems can achieve good performance, they often prioritize classification accuracy over interpretability and do not explicitly capture the underlying construction logic of Kolam patterns. More recent work has applied deep learning techniques to Kolam recognition and classification tasks. Models such as convolutional neural networks have demonstrated the ability to learn complex visual features from Kolam images and handle variations in style and drawing quality. However, these approaches typically require large datasets for training and operate as black-box systems, offering limited insight into the geometric and rule-based nature of the patterns. As a result, they are less suitable for applications that require transparency, interpretability, or direct user interaction with the design process. A critical comparison of these approaches reveals a clear trade-off. Rule-based and algorithmic methods provide transparency and enforce geometric constraints effectively, but they may lack flexibility and adaptability. In contrast, data-driven approaches offer greater generalization but sacrifice interpretability and often fail to represent the explicit rules that define Kolam construction. Furthermore, most existing research treats Kolam generation and analysis as separate problems, resulting in fragmented solutions that do not address the full scope of user interaction with

Kolam designs. Therefore, there is a noticeable gap in the development of integrated systems that combine both generation and analysis within a single, explainable framework. Existing generation methods rarely include validation or feature extraction capabilities, while analysis systems do not support pattern creation. This separation limits both practical usability and educational value. The Kolamkala system builds upon these insights by introducing a unified approach that integrates rule-based pattern generation with classical image processing for analysis. By emphasizing simplicity, transparency, and user control, it provides a practical solution that aligns with the traditional principles of Kolam design while addressing the limitations of existing computational methods.

2.1 RESEARCH GAP

Although significant progress has been made in the computational study of Kolam patterns, several limitations remain in existing approaches. Many systems focus exclusively on either pattern generation or image classification, without providing an integrated solution that supports both functionalities within a single platform. As a result, users lack tools that allow simultaneous creation and analysis of Kolam designs. Another key limitation is the lack of explainability in modern methods. Machine learning-based approaches, while effective for classification tasks, often function as black-box systems that do not reveal the underlying design logic of Kolam patterns. This reduces their educational value and limits their ability to represent the traditional rule-based nature of the art form. Furthermore, existing algorithmic generation methods are often restricted to specific pattern types and do not provide flexible, user-driven customization. Similarly, many analysis techniques are sensitive to input quality and may not reliably extract meaningful features without controlled conditions. Therefore, there is a clear need for a system that integrates generation and analysis while maintaining transparency, flexibility, and usability. Kolamkala addresses this gap by providing a rule-based, explainable framework that allows users to both generate and analyze Kolam patterns through an interactive web-based interface.

3. System Design Overview

The Kolamkala framework was implemented as a web-based application using a clear client-server architecture. The front-end was developed using HTML, Tailwind CSS, and JavaScript, providing users with an interactive interface for generating and analyzing Kolam patterns. The back-end was built using Python with the FastAPI framework. This separation of concerns ensures that computationally intensive tasks—such as pattern generation and image processing—are handled on the server, while the front-end manages user interaction and

visualization. FastAPI was selected due to its support for RESTful API development and asynchronous execution, allowing efficient handling of multiple user requests.

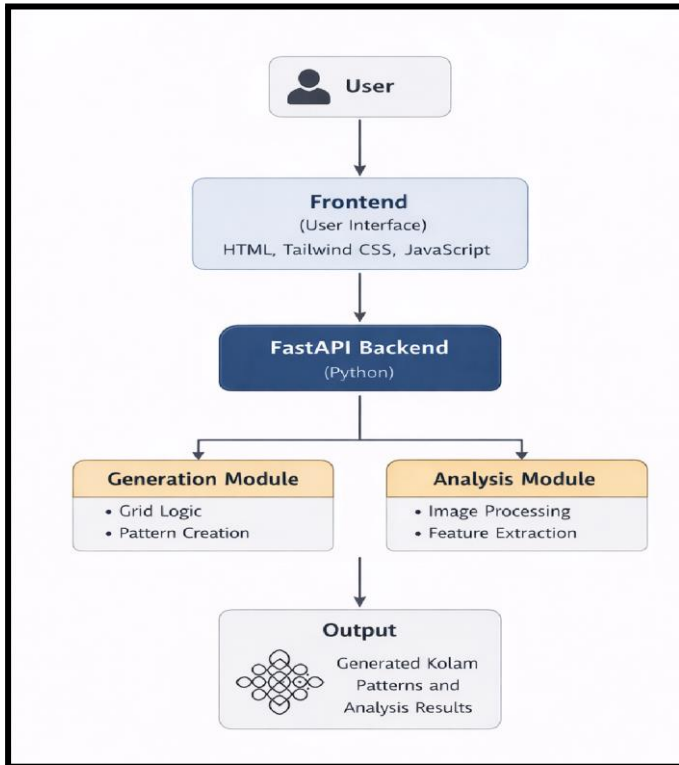


Fig -2: System architecture of the Kolumkala framework illustrating client-server interaction and module separation.

In this architecture, the front-end communicates with the back-end through HTTP requests. For example, when a user generates a Kolum pattern or uploads an image for analysis, the browser sends a request to a FastAPI endpoint. The server processes the request and returns either an image or structured data (such as JSON), which is then displayed on the interface. This design ensures modularity, scalability, and ease of maintenance. It also enables accessibility, as users can access the system through a web browser without installing additional software.

3.1 Kolum Pattern Generation Method

The Kolum pattern generator was designed using a grid-based mathematical approach inspired by traditional Pulli Kolum principles. The process begins by defining a lattice of anchor points (dots) on a Cartesian grid.

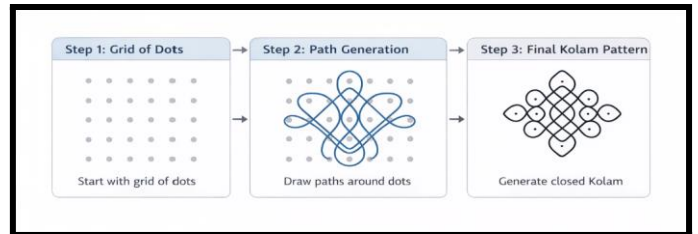


Fig -3: Step-by-step process of Kolum pattern generation from dot grid to final continuous design.

These dots serve as reference points around which the Kolum lines are constructed. Users can specify parameters such as grid size and symmetry type (e.g., bilateral or rotational symmetry). Based on these inputs, the system generates a structured arrangement of dots, typically symmetric about one or more axes. Once the grid is established, the Kolum pattern is generated by tracing a continuous path around the dots. The algorithm mimics traditional drawing techniques by creating geometric paths composed of straight lines and smooth curves. For example, diagonal connections at 45° angles are often used, followed by arcs or segments that wrap around adjacent dots.

The algorithm ensures that the generated path forms a closed loop, returning to its starting point. This is consistent with traditional Kolum designs, which are typically continuous and unbroken. Symmetry is enforced explicitly during pattern generation. After generating a base segment of the design, geometric transformations such as reflection and rotation are applied to replicate the pattern across the grid. NumPy is used to efficiently perform these transformations on coordinate arrays.

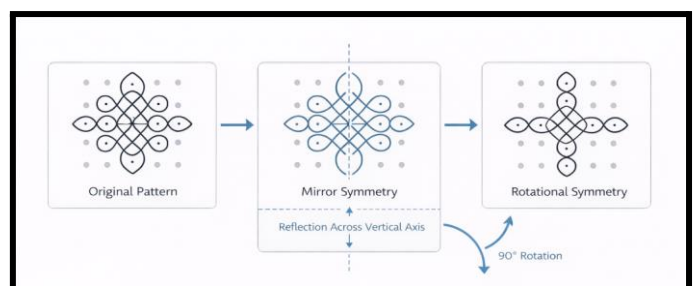


Fig -4: Illustration of symmetry transformations including mirror reflection and rotational symmetry applied in Kolum pattern generation.

Additionally, the generation process follows a rule-based traversal concept similar to a gating mechanism. Each dot is treated as a node with possible entry and exit paths. The algorithm ensures that exactly two valid paths are selected at each dot, allowing the line to pass smoothly without intersections or breaks. This guarantees that the final pattern consists of a single continuous loop. For

visualization, the generated coordinates are rendered using Matplotlib for raster images and SVG for vector output. This allows high-quality display of Kolam patterns within the web interface. NumPy is used throughout to handle coordinate calculations and transformations efficiently.

3.2 Image Analysis Method

The image analysis module uses a classical image-processing pipeline to extract meaningful and interpretable features from uploaded Kolam images.

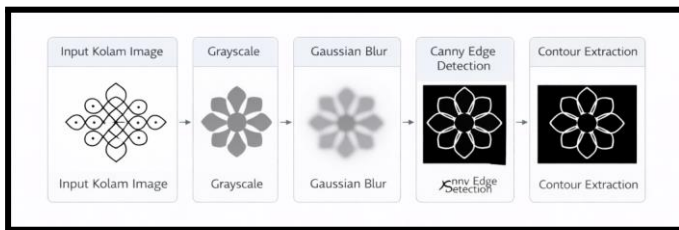


Fig -5: Image processing pipeline for Kolam analysis including grayscale conversion, Gaussian smoothing, edge detection, and contour extraction.

1. First, the input image is converted to grayscale to simplify processing and remove unnecessary color information. A Gaussian blur is then applied to reduce noise and smooth the image, enhancing the visibility of structural elements such as lines and curves.
2. Next, Canny edge detection is used to identify significant edges in the image. This step produces a binary edge map that highlights the main line structures of the Kolam.
3. Contour detection is then performed using OpenCV to identify continuous shapes within the edge map. Small contours caused by noise are filtered out, and only significant contours representing Kolam structures are retained.
4. From these contours, several quantitative features are extracted. The number of dots is estimated using techniques such as circle detection, while line-based features such as total contour length and number of loops are computed to characterize the pattern.
5. To evaluate symmetry, the system applies transformations such as horizontal and vertical flipping and compares the resulting images. Additional statistical measures, such as central moments, are used to assess symmetry more precisely.

6. The final output of the analysis includes interpretable features such as dot count, line density, number of loops, and symmetry scores. These features directly correspond to the structural properties of Kolam designs and provide meaningful insights to the user.

The entire pipeline is implemented using OpenCV and NumPy, relying on deterministic image-processing techniques rather than machine learning. This ensures transparency, efficiency, and explainability of the results.

3.3 Data Flow and Processing Pipeline

The system followed a structured data flow from user input to output generation. For pattern generation, users provided input parameters through the frontend interface. These parameters were sent as API requests to the backend, where the generation algorithm processed them and returned a rendered image. The frontend then displayed the generated pattern to the user.

For image analysis, users uploaded Kolam images through the interface. The backend processed these images using the defined pipeline and returned extracted features in a structured format. These results were presented in the frontend as visual and numerical outputs.

FastAPI endpoints handled all communication, ensuring efficient request processing and clear separation between system components.

3.4 System Implementation

The frontend consisted of dedicated pages for pattern generation and image analysis. Users interacted with input forms to specify parameters or upload images, and results were displayed dynamically using JavaScript. Tailwind CSS was used to maintain a clean and responsive interface. The backend defined API endpoints for generation and analysis tasks. Each endpoint invoked corresponding modules implemented in Python. The generation module handled grid construction and pattern rendering, while the analysis module performed image processing and feature extraction. These modules were integrated within the FastAPI application, ensuring smooth communication between frontend and backend components.

3.5 Explain ability Approach

A key aspect of the system was its emphasis on explainability. All processes were based on deterministic rules rather than learned models, allowing each output to be traced back to specific computational steps. In pattern generation, the placement of lines and symmetry transformations followed clearly defined geometric rules. In image analysis, each extracted feature corresponded

directly to observable elements in the image, such as contours or detected edges.

This transparency ensured that users could understand how patterns were generated and how features were derived, making the system suitable for both educational and analytical purposes.

3.6 Design Justification

The system adopted a rule-based approach to align with the inherent structure of Kolam patterns, which are traditionally governed by explicit geometric rules. This approach ensured consistency, interpretability, and ease of implementation without requiring training data.

Classical image processing techniques were selected for analysis due to their reliability in handling high-contrast line drawings and their ability to produce interpretable results. Unlike data-driven methods, these techniques provided direct control over processing steps and eliminated the need for complex training procedures.

Overall, the design prioritized simplicity, transparency, and practical usability while effectively addressing the objectives of Kolam pattern generation and analysis.

4. Results

The Kolamkala system successfully generated structured Kolam patterns based on user-defined grid sizes and symmetry constraints. For example, when a 5×5 grid with fourfold rotational symmetry was selected, the system produced a continuous one-stroke design in which a single line looped around all anchor points and returned to its starting position, forming a closed and symmetric pattern. Similarly, smaller grids such as 3×3 with mirror symmetry resulted in designs that clearly exhibited bilateral symmetry across both axes.

Across all tested configurations, the generated patterns maintained geometric correctness. The line segments accurately followed the defined grid coordinates, and the rendered outputs were visually clean and free from distortions. The system consistently ensured that all dots were enclosed within continuous loops, without improper overlaps or breaks. As the grid size increased, the complexity of the patterns also increased, producing more intricate designs while preserving structural consistency. For instance, patterns generated on larger grids such as 8×8 demonstrated multiple interlocking loops while still maintaining continuity and symmetry.

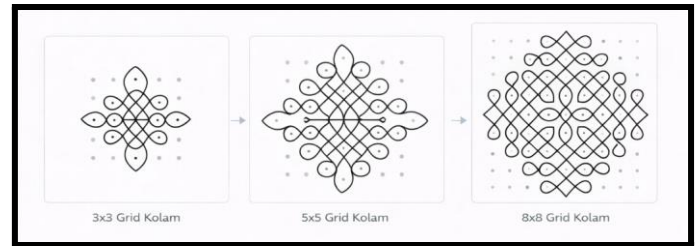


Fig -6: Generated Kolam patterns for different grid sizes (3×3, 5×5, and 8×8) demonstrating increasing complexity and structural consistency.

On the analysis side, the OpenCV-based pipeline performed reliably for structured inputs. After applying grayscale conversion and Gaussian smoothing, Canny edge detection produced clear outlines of the Kolam strokes [2]. Contour detection successfully identified the main structural elements of the designs. In simpler patterns, a single dominant contour represented the primary loop, while smaller contours corresponded to individual dots. In more complex patterns, multiple contours were detected, but the major ones aligned correctly with the main segments of the design.

The system also demonstrated accurate feature extraction. Dot structures were identified using contour-based filtering, and the detected number of dots matched the expected grid configurations. Similarly, loop detection corresponded well with the known structure of the generated patterns. For example, a 4×4 grid design resulted in 16 detected dots and a single continuous loop, consistent with the input parameters.

Symmetry detection further validated the correctness of the system. By comparing the original image with its mirrored or rotated versions, the system was able to assess symmetry accurately. Patterns that were intentionally symmetric showed high similarity scores, while asymmetric or distorted inputs resulted in lower values. This confirmed that the extracted features such as dot count, loop structure, and symmetry closely reflected the actual geometry of the Kolam patterns.

Overall, the results indicate that the system produces visually consistent designs and accurately extracts structural features from input images. The outputs remain interpretable and reproducible, as they directly follow the defined rule-based logic.

5. Discussion

The findings demonstrate that Kolamkala effectively captures the fundamental principles of traditional Kolam design. The successful generation of continuous,

symmetric patterns confirms that grid-based rules and geometric constraints are sufficient to reproduce authentic Kolam structures. This aligns with existing studies that describe Kolam patterns as structured arrangements of dots connected by continuous lines under symmetry constraints.

The image analysis results further support this observation. The ability of classical image processing techniques to detect edges, contours, and structural features indicates that Kolam patterns can be reliably analyzed without the need for complex learning-based models. In controlled conditions, the system's outputs closely matched expected structural properties, reinforcing the reliability of the approach.

Compared to existing methods, Kolamkala offers a distinct advantage through its emphasis on explainability. While many prior approaches rely on machine learning for classification or recognition, they often operate as black-box systems with limited interpretability. In contrast, the rule-based design of Kolamkala ensures that every generated pattern and extracted feature can be traced back to explicit computational steps. This makes the system more transparent and suitable for educational and analytical use.

Another strength of the system lies in its consistency. Given the same input parameters, the generator always produces the same output, ensuring reproducibility. The use of numerical computation through NumPy also contributes to precise and visually accurate designs. Similarly, the analysis pipeline produces stable results for structured and high-quality inputs, making it reliable for typical Kolam patterns.

However, the system also exhibits certain limitations. Since it is based on grid-defined rules, it performs best with structured Kolam designs and may not handle freehand or irregular patterns effectively. In such cases, the analysis pipeline may produce incomplete or fragmented results. Additionally, the performance of edge detection and contour extraction depends on image quality. Variations in lighting, noise, or drawing clarity can affect the accuracy of feature detection.

There is also a trade-off between simplicity and flexibility. While the rule-based approach ensures transparency and reliability, it limits the system's ability to adapt to highly diverse or unconventional patterns. In contrast, data-driven methods can generalize better across variations but lack interpretability. Kolamkala prioritizes clarity and control over adaptability, which is appropriate for its intended purpose but restricts its scope in more complex scenarios.

6. Limitations and Future Work

One key limitation of the system is its dependence on structured, grid-based inputs. Future improvements could focus on extending the generation module to support a wider variety of Kolam styles, including more complex or free-form designs.

On the analysis side, incorporating advanced preprocessing techniques such as adaptive thresholding or morphological operations could improve robustness against noise and variations in image quality. Enhancing dot detection and contour refinement would further increase accuracy.

Another potential direction is to introduce limited adaptive methods to improve flexibility while maintaining explainability. Additionally, improving the user interface to allow interactive editing or customization of patterns could enhance usability.

7. CONCLUSIONS

This study presented Kolamkala, a web-based computational framework for the generation and analysis of traditional Kolam patterns using a rule-based approach. The primary objective was to translate the geometric logic of Kolam design into an interpretable and implementable system that allows both pattern creation and structural analysis.

The results demonstrated that the system was capable of generating consistent and visually accurate Kolam designs based on user-defined grid sizes and symmetry constraints. The generated patterns preserved key characteristics of traditional Kolams, including continuous loops, enclosed dots, and symmetrical structures. In addition, the image analysis module effectively extracted important features such as edges, contours, dot arrangements, and symmetry, confirming that classical image processing techniques are sufficient for analyzing structured Kolam designs.

The significance of this work lies in its emphasis on simplicity and explainability. Unlike data-driven approaches, the system relies on deterministic rules that make every output transparent and reproducible. This makes Kolamkala particularly useful for educational purposes, as users can understand how patterns are formed and how their structural properties are evaluated. Furthermore, the system contributes to the digital preservation of traditional Kolam art by providing a computational representation of its underlying design principles.

However, certain limitations remain. The system is primarily designed for grid-based Kolam patterns and

performs best with clean, well-defined inputs. It may not handle freehand or irregular designs effectively, and the accuracy of the analysis module depends on image quality. These limitations highlight the trade-off between simplicity and flexibility in rule-based systems.

Future work can focus on expanding the range of supported Kolam styles, improving the robustness of image processing under varying conditions, and enhancing user interaction through more flexible design tools. Incorporating adaptive techniques while maintaining explainability could further improve the system's applicability.

In conclusion, Kolamkala demonstrates that traditional Kolam patterns can be effectively modeled using rule-based computational methods. By combining generation and analysis within a single framework, the system provides a practical and interpretable approach to studying and preserving this culturally significant art form.

REFERENCES

- [1] Sahapedia, "Significance of Kolam in Tamil Culture." [Online]. Available: <http://www.sahapedia.org/significance-of-kolam-tamil-culture>
- [2] Tran et al., "KolamNetV2: Deep Learning for Kolam Pattern Recognition," Nature Heritage Science, 2024. [Online]. Available: <https://www.nature.com/articles/s40494-024-01167>
- [3] P. M. C. Article, "Kolam Drawings and Mathematical Patterns." [Online]. Available: <https://pmc.ncbi.nlm.nih.gov/articles/PMC10427318>
- [4] Deccan Herald, "The Mathematics Behind Rangoli," 2026. [Online]. Available: <https://www.deccanherald.com/dhie/social-studies/2026/02/17/the-mathematics-behind-rangoli>
- [5] Economic Times, "Kolam Patterns as Mathematical Codes." [Online]. Available: <https://economictimes.indiatimes.com/news/india/how-kolam-patterns-in-south-india-are-secretly-mathematical-codes>
- [6] Algorithmic Pattern Conference, "Kolam Generation Methods," 2025. [Online]. Available: <https://2025.algorithmicpattern.org/proceedings/S9 NXAC/paper.html>