Design and Implementation of a Synthetic Polygon Generator

A Thesis

submitted to the designated

by the Assembly

of the Department of Computer Science and Engineering

Examination Committee

by

Vasileios Tsolis

in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER

SYSTEMS ENGINEERING

WITH SPECIALIZATION

IN DATA SCIENCE AND ENGINEERING

University of Ioannina

School of Engineering

Ioannina 2025

Examining Committee:

- **Nikolaos Mamoulis,** Professor at the Department of Computer Science and Engineering, University of Ioannina (Advisor)
- **Panos Vassiliadis,** Professor at the Department of Computer Science and Engineering, University of Ioannina
- **Apostolos Zarras,** Professor at the Department of Computer Science and Engineering, University of Ioannina

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Professor Nikolaos Mamoulis, for their continuous guidance, support, and valuable feedback throughout the course of this thesis. Their expertise and encouragement have been instrumental in shaping this work and in expanding my understanding of the field.

I am also deeply thankful to PhD Candidate's Thanasis Georgiadis, whose insightful discussions, technical advice, and generous assistance played a significant role during the development and implementation of the system presented in this thesis.

Finally, I would like to thank the University of Ioannina and the Department of Computer Science and Engineering for providing a stimulating academic environment and access to the necessary resources that made this research possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ABSTRACT

Vasileios Tsolis, M.Sc. in Data and Computer System Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, June 2025
Design and Implementation of a Synthetic Polygon Generator
Advisor: Nikolaos Mamoulis, Professor.

The need for synthetic spatial data has grown significantly in recent years, driven by the increasing demand for large, diverse, and statistically representative datasets in geospatial machine learning, benchmarking, and simulation tasks. While several solutions exist for generating synthetic point data or raster-based representations, tools for generating realistic and controllable polygonal geometries remain limited. This thesis presents the design and implementation of a novel web-based system for synthetic polygon generation that bridges the gap between algorithmic control and statistical realism. The system supports multiple generation methods, including procedural algorithms (irregular, Voronoi, elongated, and experimental shapes), a nonparametric empirical copula method for upload-based distribution matching, and a feature-based generator utilizing geometric descriptors such as area, convexity, aspect ratio, compactness, and spikiness. A key innovation of this work is the Distributional Geometry Alignment Score, a metric specifically developed to evaluate the similarity between synthetic and real polygon datasets in terms of both marginal distributions and inter-feature correlations. The generation platform is implemented using Open Layers and modern web technologies, offering real-time visualization, interactive configuration, and export in standard formats such as WKT, CSV, and GeoJSON. Extensive experimental evaluation demonstrates that the system can generate hundreds of thousands of polygons with high fidelity to reference data, maintaining scalability and diversity across various spatial distributions. The proposed framework provides a transparent, extensible, and statistically grounded solution for synthetic polygon generation, making it suitable for applications in data augmentation, simulation, and the development of machine learning models for spatial tasks.

CHAPTER 1

INTRODUCTION

1.1 Objectives

The generation of synthetic spatial data [1] has become increasingly important across a wide array of scientific and technological disciplines, including geospatial analysis, machine learning, remote sensing, and simulation-based planning. As access to high-quality, well-annotated real-world spatial datasets remain limited due to cost, privacy constraints, or geographical inaccessibility the ability to generate synthetic data that preserves key statistical and geometric properties has gained critical relevance. Particularly in the context of polygonal data, there is a clear need for methods that allow the controlled generation of shapes that are not only geometrically valid but also statistically meaningful. Polygonal geometries carry complex morphological and topological characteristics that make their generation more challenging than points or raster data, especially when the goal is to preserve shape diversity, internal structure, and statistical realism.

While several approaches [2] exist for synthetic point generation or image-based data simulation, tools specifically designed for polygon generation are scarce. Those that do exist are either overly simplified, focusing on primitive shapes or random noise, or tailored to highly specialized use cases such as urban footprint generation or terrain simulation. Furthermore, many systems operate as black boxes, offering limited transparency, low interactivity, and no

mechanism for evaluating the statistical fidelity of the generated data in comparison to a reference dataset. As a result, there is currently no open and interactive solution that supports explainable, reproducible, and statistically controlled polygon generation at scale.

The present thesis proposes the design and implementation of a novel web-based system for synthetic polygon generation that addresses these limitations. The proposed system combines algorithmic and data-driven methods [3] in a unified and extensible framework, allowing users to generate synthetic polygonal datasets that are morphologically diverse, statistically representative, and visually explorable. The system integrates three complementary approaches: procedural generation using configurable algorithms (e.g., irregular, Voronoi, and elongated shapes), empirical generation based on a nonparametric copula method that matches the joint distribution of a user-provided dataset, and feature-based generation using geometric descriptors extracted from real polygons. Together, these methods enable a wide range of use cases from the simulation of abstract geometric patterns to the replication of structural features found in real-world spatial datasets.

The user interface of the system, built using Open Layers and modern web technologies, offers real-time interaction for parameter tuning, visualization, bounding box placement, and file export in multiple formats such as CSV, WKT, and GeoJSON. A distinctive feature of our system is the implementation of a nonparametric empirical copula method that allows upload-based distribution matching. This allows users to guide the generation process using real data, ensuring that the synthetic output replicates not only the individual feature distributions (e.g., area, compactness) but also the correlations among them. Furthermore, a new evaluation metric the Distributional Geometry Alignment Score (DGAS) has been developed to quantify the alignment between synthetic and reference datasets, providing a rigorous basis for comparison.

This thesis aims to develop a fully functional, modular tool that supports explainable synthetic polygon generation under statistical control. It focuses on generating geometries that are structurally valid, statistically faithful to reference distributions, and suitable for visualization, testing, and augmentation tasks in spatial data science. The system also provides a mechanism for benchmarking generative models and evaluating trade-offs between shape complexity, distributional fidelity, and computational performance. The work culminates in a series of experiments that evaluate the system's performance across different generation scenarios, with extensive comparisons in terms of scalability, statistical similarity, and visual diversity.

1.2 Structure of the Thesis

The remainder of this thesis is structured into 6 chapters. Chapter 2 provides a comprehensive review of the literature on synthetic data generation, geometric modeling, and spatial evaluation methods, highlighting gaps in existing tools and motivating the system's design. Chapter 3 presents the architecture and implementation of the synthetic generation system, detailing its procedural, data-driven, and feature-based modules. Chapter 4 describes the interactive web interface, including visualization layers, parameter controls, and export functionalities. Chapter 5 focuses on the evaluation of the system in terms of performance, fidelity, and similarity to real datasets, introducing and applying the DGAS metric. Finally, Chapter 6 summarizes the contributions of the thesis, reflects current limitations, and outlines potential directions for future work.

CHAPTER 2

RELATED WORK

Synthetic polygon generation is a fundamental task in computational geometry, GIS, and spatial data science. It supports a variety of downstream applications, including simulation, machine learning, algorithm benchmarking, and spatial query evaluation. In recent years, the demand for diverse and controllable polygonal datasets has grown, especially in data-driven domains that require extensive training data for model generalization and robustness.

Several techniques have been proposed in the literature for the synthetic generation of polygonal geometries. One of the most common methods is based on sampling points around a central shape [4], typically a circle or an ellipse, and connecting them sequentially to form a closed polygon. These circle-based methods offer a straightforward way to control the number of vertices and the smoothness of the resulting shape but are generally limited to convex or mildly non-convex forms. Another widely used strategy involves triangulation. In this approach, a set of points is generated randomly or from a specified distribution, and then a triangulation method such as Delaunay triangulation is applied. From the triangulated mesh, subsets of adjacent triangles are merged to construct complex polygonal shapes. This enables the creation of both convex and non-convex geometries, though the output often requires cleaning steps to ensure topological validity, such as avoiding self-intersections or duplicate edges.

Alternative techniques include methods based on random walks, Voronoi diagrams [5], and Boolean operations over geometric primitives. Complex irregular polygons can be constructed by combining simple shapes through union and different operations or by perturbing grid-based patterns. Other approaches generate polygons from line string skeletons [6] or via noise-controlled deformation of basic shapes. Several methods rely on procedural noise, recursive subdivision, or rule-based grammar to generate synthetic forms with specific visual or

structural properties. There are also sampling-based approaches that attempt to reproduce the statistical distribution of vertices or angles found in real-world polygons.

Despite the variety of techniques, most existing implementations are either problem-specific or do not allow for high-level shape control [7]. Properties such as compactness, elongation, convexity, irregularity, presence of holes, or area-to-perimeter ratio are often not explicitly parametrized, making it hard to target specific polygon types or match distributions. Furthermore, existing approaches rarely allow a systematic exploration of how different polygon features affect downstream tasks, limiting their utility in the context of machine learning or algorithmic benchmarking.

One of the few tools designed for synthetic spatial dataset generation is SpiderWeb [8], which focuses on producing benchmark data for spatial query evaluation. Although it provides a GUI and supports multiple data types, it is limited to simple geometric structures such as rectangles and points. It lacks support for complex polygonal generation and does not offer shape-level customization or feature-based controls. In addition, visual inspection of the generated data and export functionality in standard formats such as WKT or GeoJSON is not fully supported.

Other tools and libraries found in GIS packages or geometric frameworks [1], [9], such as Shapely, GEOS, or CGAL, allow for manual polygon construction or manipulation but are not designed for scalable, user-friendly synthetic generation of diverse polygon datasets. Moreover, most open-source solutions do not offer flexible configuration interfaces, nor do they support high-level polygon descriptors as input parameters. There is also a general absence of tools that can combine generation with statistical evaluation and visual feedback in a single workflow.

The limited availability of open, flexible, and extensible polygon generators presents a significant challenge for the research community. A fully featured polygon generation system could address several critical needs. It would support the creation of large volumes of diverse training data for machine learning and deep learning models and enable robust benchmarking of spatial algorithms and geometric pipelines. Such a system would allow controlled experimentation on polygon properties, including the evaluation of algorithmic behavior on different shape types, such as convex, non-convex, irregular, or elongated forms.

Recent research on geometry-aware learning and shape interpretability emphasizes the importance of using synthetic data with meaningful and diverse shape descriptors [10]. In cases where real-world datasets are unavailable, private, expensive, or biased, synthetic data

becomes an essential alternative. A new system designed with the capacity to generate various polygon types, including those with holes or complex boundaries, and to provide parametric control over geometric features such as number of vertices, compactness, elongation, and spikiness, would be of great value.

Such a system would ideally offer visual preview and interactive manipulation of generated shapes, export options in common GIS formats like CSV, GeoJSON, or WKT, and seamless integration with statistical feature extraction modules and similarity metrics for data-driven generation. The proposed synthetic polygon generation framework aims to fulfill these objectives, offering both flexibility and usability. By combining procedural techniques, visual tools, and statistical modeling, it extends beyond the scope of existing tools like SpiderWeb and lays the foundation for systematic experimentation in spatial AI and geometry processing.

CHAPTER 3

METHODOLOGY

In this chapter, we present the methodological framework developed to support the generation of synthetic spatial data. The system incorporates two complementary approaches: **Synthetic Geometry Generation** and **Data-Driven Generation**, each addressing different use cases in spatial simulation and modeling.

Section 3.1 focuses on Synthetic Geometry Generation, detailing the design and implementation of various points and polygon generation algorithms. These include both traditional techniques such as uniform and Gaussian sampling and more complex, shape-oriented methods like irregular polygons and Voronoi tessellations. The objective is to offer flexible, controllable tools for creating diverse spatial structures within a user-defined bounding area.

Section 3.2 introduces the Data-Driven Generation paradigm, which allows users to upload real-world datasets and produce synthetic data that statistically mimics the uploaded samples. This is achieved through a nonparametric distribution-matching process that preserves both marginal distributions and inter-variable dependencies.

By integrating both geometry-oriented and data-driven techniques, the system empowers users to produce synthetic spatial datasets that are adaptable to a wide range of analytical tasks, offering a balance between structural variety and statistical fidelity.

3.1 Synthetic geometry generation

The system supports the generation of two types of primary spatial objects: points and polygons. All objects are generated within a global spatial domain specified by the user in the form of a bounding box. This bounding box defines the overall extent of the synthetic data space, not the bounds of each individual shape. Within this domain, spatial objects are generated according to user-defined parameters that influence object density, spatial distribution, average shape size, vertex complexity, and irregularity.

3.1.1 Shape Generation

The generation of synthetic shapes begins with the creation of spatial point distributions inside the user-specified generation area. Sampling strategies such as uniform, Gaussian, diagonal, or clustered placement are available to control how points are spatially distributed. These generated points then serve as the geometric foundation for constructing polygons. Depending on the selected method, different polygon generation algorithms are applied to produce diverse and valid geometries. The following algorithms are applied depending on the selected method.

To reduce overlaps, the system can space out centers during point generation while taking each shape's intended size into account. When the domain is large enough, this makes polygons non overlapping by construction. Voronoi cells are disjoint by definition.

A. Circle-Based Techniques

The first technique [11] focuses on generating irregular polygons around center points to simulate non-uniform geometric structures. The fundamental concept behind the method is to construct a closed polygon by sequentially generating vertices at varying angles and distances from a central point, thus introducing controlled randomness into both the shape's outline and vertex distribution.

This formulation is realistic because many real spatial patches grow outward from an interior core and exhibit locally uneven boundaries. Examples include small lakes and ponds, wetland patches, shrub or dune clusters, tree crowns, wildfire burn scars, flood extents, and

lava or landslide deposits. The irregularity control reflects heterogeneous vertex spacing and the spikiness control reflects boundary roughness observed in these data.

The algorithm operates in two main stages. Initially, a random number of vertices is selected within user-defined minimum and maximum limits. Following this, a set of angular steps between consecutive vertices is computed. These angular steps are not uniform but are randomized around a base angle, with the degree of variation governed by an "irregularity" parameter. A higher irregularity value leads to more irregular spacing between vertices.

For each angle, a corresponding radial distance from the center is generated. This distance is determined by adding a random perturbation, controlled by a "spikiness" parameter, to a base radius. Spikiness controls the variation in vertex distances, resulting in more jagged and complex shapes when higher values are used.

Algorithm 3.1 Generate Polygon around a Center Point

Require: $center$: (x, y), $\min\ \_radius$, $\max\ \_radius$, $irregularity$, $spikiness$, $min\_vertices$, $max\_vertices$

Ensure: $vertices\_list$: $List\ of\ points$

if $irregularity\ <\ 0\ or\ irregularity\ >\ 1$ **then**
    raise $ValueError(\text{"}Irregularity\ must\ be\ between\ 0\ and\ 1.\text{"})$
end if
if $spikiness\ <\ 0\ or\ spikiness\ >\ 1$ **then**
    raise $ValueError(\text{"}Spikiness\ must\ be\ between\ 0\ and\ 1.\text{"})$
end if
if $minRadius\ <=\ 0\ or\ maxRadius\ <=\ 0$ **then**
    raise $ValueError(\text{"}Radius\ values\ must\ be\ greater\ than\ 0.\text{"})$
end if
if $minRadius\ >\ maxRadius$ **then**
    raise $ValueError(\text{"}minRadius\ must\ be\ smaller\ or\ equal\ to\ maxRadius.\text{"})$
end if
sample $random\ integer\ numVertices\ between\ minVertices\ and\ maxVertices$
$irregularity \leftarrow irregularity \times (2\pi\ /\ numVertices)$
$angleSteps \leftarrow randomAngleSteps(numVertices, irregularity)$
sample $baseRadius\ uniformly\ in\ [minRadius, maxRadius]$
$maxSpike \leftarrow spikiness \times maxRadius$
initialize $empty\ list\ points$

sample $random\ angle\ in\ [0, 2\pi]$
for $i\ \boldsymbol{from}\ 1\ to\ numVertices$ **do**
  if $numVertices\ <=\ 5$ **then**
    if $i\ mod\ 2\ ==\ 0$ **then**
      sample $spike\ \boldsymbol{from}\ Uniform(0.6\ \times\ maxSpike,\ maxSpike)$
    else
      sample $spike\ \boldsymbol{from}\ Uniform(0.4\ \times\ maxSpike,\ 0.8\ \times\ maxSpike)$
    end if
  else
    sample $spike\ \boldsymbol{from}\ Uniform(-maxSpike,\ maxSpike)$
  end if
  sample $radius\ \boldsymbol{from}\ Gaussian(baseRadius\ +\ spike, 0.5\ \times\ maxSpike)$
  clip $radius\ to\ [0, baseRadius]$
  if $numVertices\ <=\ 5\ and\ i\ mod\ 3\ ==\ 0$ **then**
    $Multiply\ radius\ \boldsymbol{by}\ random\ number\ in\ [0.6, 0.8]$
  end If
  $x\ \leftarrow\ center[0]\ +\ radius\ \times\ cos(angle)$
  $y\ \leftarrow\ center[1]\ +\ radius\ \times\ sin(angle)$
  create $vertex\ (x, y)$
  append $vertex\ to\ points$
 $angle\ \leftarrow\ angle\ +\ angleSteps[i]$
end for
append $points[0]\ to\ points\ to\ close\ the\ polygon$
return $points$

The full procedure is detailed in **Algorithm 3.1**, which outlines input requirements (e.g., irregularity, spikiness, vertex range), validation steps, and the core loop responsible for computing and assembling the polygon's vertices. Specifically, the algorithm includes specialized logic for handling polygons with five or fewer vertices, introducing alternating positive and negative spikes to avoid degenerate or unrealistic shapes. Each vertex is then positioned using trigonometric transformations, and all vertices are connected in sequence to form a closed polygon.

Once constructed, the final polygon is appended with its initial vertex to ensure closure. This method provides fine-grained control over geometric properties such as shape complexity, edge irregularity, and compactness, making it highly suitable for synthetic spatial data generation in applications that require diverse polygonal features. **Figure 3.1** shows typical outputs of the circle-based method over the unit square and illustrates how vertex spacing and outline roughness vary with the chosen settings.

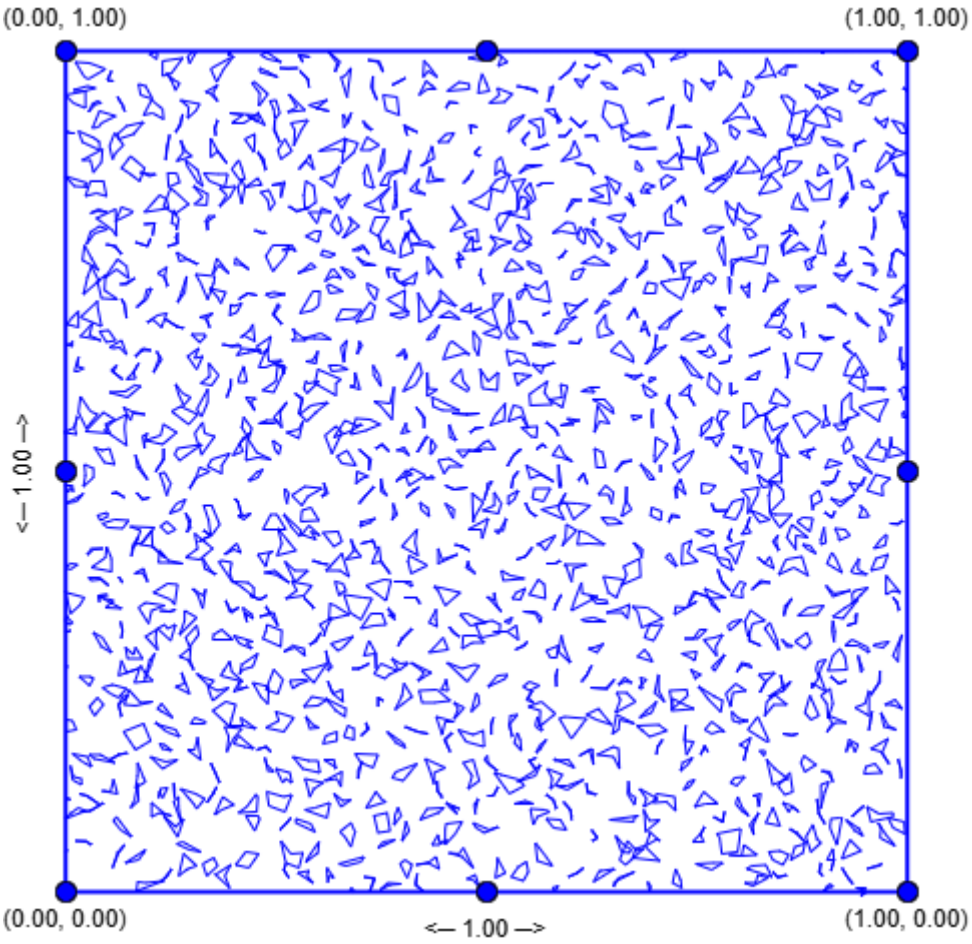

Figure 3.1. Circle based irregular polygons generated across the unit square $[0,1] \times [0,1]$. Centers come from the selected point distribution, and each polygon is built by randomizing angular steps and radii around a base value. The plot highlights the diversity produced by different vertexes count together with irregularity and spikiness settings.

B. Voronoi Polygons

The second method employs Voronoi tessellation [11] to generate spatial polygons based on a set of input seed points. A Voronoi polygon is defined as the region of space closer to a given point than to any other, resulting in a natural partitioning of the plane into non-overlapping, contiguous cells. This approach is particularly useful in applications that require polygons sharing edges, such as urban simulations, land division, and spatial proximity analysis.

The **Algorithm 3.2** operates as follows. First, the set of seed points is collected and transformed into a NumPy array for efficient processing. A Voronoi diagram is then constructed using the Scipy library, based on the Delaunay triangulation of the points. For each region in the Voronoi diagram, if the region is finite (i.e., it does not extend to infinity), the corresponding vertices are extracted, and a Shapely polygon is formed.

Algorithm 3.2 Generate Voronoi Polygon with shrink factor

Require: $A\ set\ of\ seed\ points\ points, bounding\ box\ bbox\ = (minX, minY, maxX, maxY)$, $shrink\ factor\ shrink\ factor\ \in [0,1]$

Ensure: $A\ list\ of\ validated, clipped\ Voronoi\ polygons$

If $points\ <\ 2$ **then**
  $exit$
end if
$points\_np\ \leftarrow\ Convert\ `points`\ to\ NumPy\ array$
$vor\ \leftarrow\ Voronoi(points\_np)$
$bounding\_box\ \leftarrow\ box(minX, minY, maxX, maxY)$
$polygons\ \leftarrow\ [\,]$
for $each\ region\ in\ vor.regions$ **do**
  if $region\ is\ empty\ OR\ contains\ -1$ **then**
    $continue$
  end if
  $vertices\ \leftarrow\ Extract\ vertices\ from\ vor.vertices$
  $polygon\ \leftarrow\ Polygon(vertices)$
$centroid\ \leftarrow\ polygon.centroid$
$modified\_polygon\ \leftarrow\ []$
for $each\ vertex\ (x, y)\ in\ vertices$ **do**
  $new\_x\ \leftarrow\ x\ +\ shrink\_factor\ \times\ (centroid.x\ -\ x)$
  $new\_y\ \leftarrow\ y\ +\ shrink\_factor\ \times\ (centroid.y\ -\ y)$
  append $(new\_x, new\_y)\ to\ modified\_polygon$
end for
$shrink\_polygon\ \leftarrow\ Polygon(modified\_polygon)$
$clipped\_polygon\ \leftarrow\ shrunk\_polygon\ \cap\ bounding\_box$
$validated\_polygon\ \leftarrow\ correct\_invalid(clipped\_polygon)$
If $validated\_polygon\ is\ valid\ \boldsymbol{AND}\ not\ empty$ **then**

25: $Write\ validated\_polygon\ to\ sink$

26: **end if**

27: **end for**

To improve the geometric consistency of the generated Voronoi polygons, a shrink transformation is applied, as illustrated in **Figure 3.1**. Specifically, each vertex is moved toward the centroid of its corresponding polygon according to a user-defined shrink factor. This operation reduces the spread of vertices, resulting in more compact and visually coherent shapes. Mathematically, the new coordinates $(x',\ y')$ of each vertex are computed by interpolating between the original vertex (x, y) and the centroid $(c_{x,}\ c_y)$ based on the shrink factor f, following the formulas:

$$x' = x + f(c_x - x)$$
$$y' = y + f(c_y - y)$$

where $f = 0$ leaves the polygon unchanged and $f = 1$ collapses it entirely to its centroid. After applying the shrink operation, each polygon is clipped against the predefined bounding box to ensure that all resulting geometries remain within the spatial boundaries and are valid for further processing.

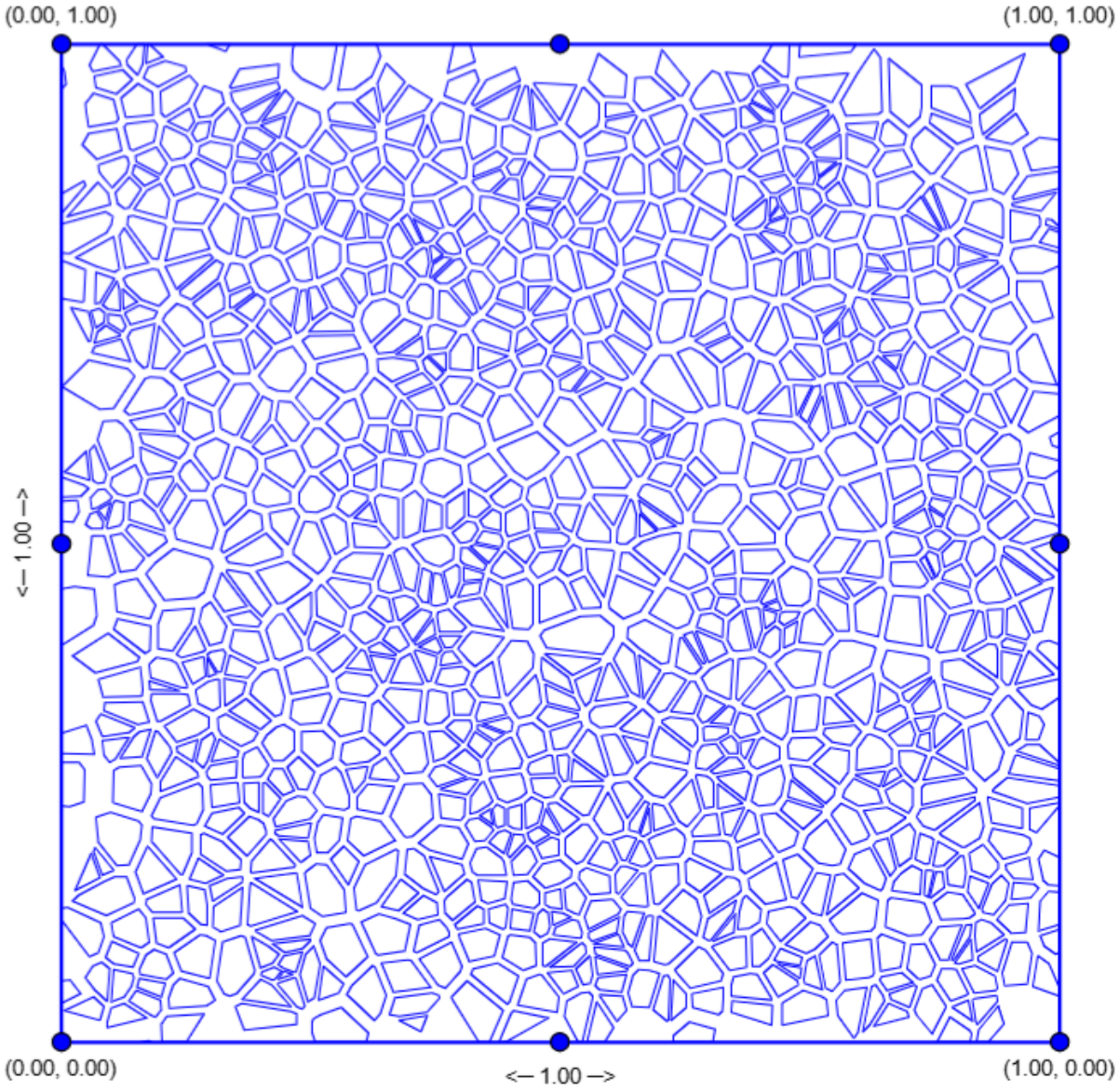

Figure 3.2. Shrink-transformed Voronoi polygons within the bounding box. The control points (blue dots) define the rectangular boundary, and each Voronoi cell has been post-processed using a shrink factor to interpolate its vertices toward the centroid. This operation increases compactness and visual coherence while ensuring all geometries remain spatially valid.

The entire process is formalized in Algorithm 3.2, which outlines the steps for validating and adjusting each region. During validation, polygons are checked for common geometric issues such as self-intersections and invalid topologies. Minor artifacts such as sliver polygons, nearly colinear edges, or small holes, are corrected through simplification, snapping, or removal based on threshold criteria.

Compared to circle-based polygon generation techniques, Voronoi-based generation ensures complete spatial coverage and shared edges, properties that are difficult to achieve with purely stochastic methods. However, this approach offers limited control over detailed polygon characteristics like spikiness or irregularity. The final structure and size of the polygons are primarily influenced by the initial distribution of the seed points.

Nevertheless, Voronoi tessellation provides a powerful and structured method for generating realistic, contiguous spatial datasets suitable for simulations, mapping, and analytic tasks that benefit from spatial coherence.

C. Elongated Polygons

The third polygon generation method focuses on simulating elongated [12], river-like geometries, which are commonly encountered in natural landscapes such as water networks, valleys, or flow-dominated terrains. This technique is designed to produce narrow, flow-aligned polygons based on user-defined boundary points and a given flow direction vector.

The generation process is encapsulated in a specialized class and proceeds through a multi-step algorithm, which systematically constructs an elongated polygon by leveraging a Triangulated Irregular Network (TIN) and geometric filtering operations.

The overall workflow is outlined below:

1. **TIN Construction**: The algorithm first constructs a Triangulated Irregular Network [13] from the provided river boundary points using Delaunay triangulation. This representation captures the local topology and connectivity of the boundary geometry.
2. **Perpendicular Edge Selection**: From the generated TIN, edges that are approximately perpendicular to the specified flow direction are identified. These edges are presumed to represent cross-sectional slices of the river or flow path.
3. **Mainstream Filtering**: To ensure that only relevant structures contribute to the final shape, edges are filtered to retain only those that intersect with the main river polygon. This step eliminates noise from peripheral or disconnected areas.
4. **Centerline Generation**: A river centerline is constructed by computing the midpoints of the selected TIN edges. This centerline serves as the backbone of the elongated polygon.
5. **Polygon Construction**: The centerline is then expanded laterally by applying parallel offsets to the left and right sides, simulating river width. These offset lines are merged to form a closed polygon, with safeguards in place to correct invalid geometries through buffering and coordinate rounding.

6. **Validation and Scaling**: The resulting polygon is validated using Shapely's topological checks and optionally scaled down for visualization consistency. Any internal rings (holes) are removed to retain a clean outer boundary.

Algorithm 3.3 Elongated Polygon Generation via TIN and Offset Construction

Require:

$river\ boundaries$: $list\ of\ coordinates$

$flow\ direction$: $vector$

$offset\ distance$: $width\ parameter$

$sink$: $polygon\ output\ interface$

Ensure: $A\ valid\ elongated\ polygon\ representing\ the\ river\ shape$

function $RiverPolygon(river\ boundaries, flow\ direction)$

  if $length(river\ boundaries) < 3$ **then**

    $raise\ Error$ "$At\ least\ 3\ boundary\ points\ required$"

  end if

  $TIN \leftarrow GenerateTIN(river\ boundaries)$

  if $TIN\ is\ empty$ **then**

    $raise\ Error$ "$TIN\ generation\ failed$"

  end if

  $perpendicular \leftarrow SelectEdgesPerpendicularToFlow(TIN, flow\ direction)$

  if $perpendicular\ edges\ is\ empty$ **then**

    $raise\ Error$ "$No\ suitable\ edges\ found$"

  end if

  $filter \leftarrow FilterEdgesWithinPolygon(perpendicular\ edges, river\ boundaries)$

  if $filter\ is\ empty$ **then**

    $raise\ Error$ "$Filtered\ edge\ set\ is\ empty$"

  end if

  $centerline \leftarrow ComputeMidpoints(filtered\ edges)$

  if $length(centerline) < 2$ **then**

    $raise\ Error$ "$Centerline\ must\ have\ 2\ points$"

  end if

  $left\ offset \leftarrow OffsetLine(centerline, 'left', offset\ distance)$

  $right\ offset \leftarrow OffsetLine(centerline, 'right', offset\ distance)$

23: $coords \leftarrow Concatenate(left\ offset, Reverse(right\ offset))$

24: $raw\ polygon \leftarrow Polygon(coords)$

25: **if** $not\ IsValid(raw\ polygon)$ **then**

26: $raw\ polygon \leftarrow BufferFix(raw\ polygon)$

27: **end if**

28: $cleaned\ polygon \leftarrow KeepExteriorOnly(raw\ polygon)$

29: $scaled\ polygon \leftarrow ScalePolygon(cleaned\ polygon, 0.1)$

30: $sink.writePolygon(scaled\ polygon)$

31: $sink.flush$

32: **end function**

The full procedure integrates the above steps and writes the final polygon to the data sink. Each subroutine (such as generate_tin, select_perpendicular_edges, generate_centerline, and construct_river_polygon) are modular, enabling reuse and easy extension. **Figure 3.3** shows typical outputs of the elongated method over the unit square and illustrates narrow flow-aligned corridors produced by the TIN-based centerline and lateral offsets.

This method is particularly effective in simulating hydrologically inspired structures or linear geographic features, where elongation and directionality are essential. Compared to other polygon generation techniques, it offers fine-grained geometric control aligned with real world flow phenomena.

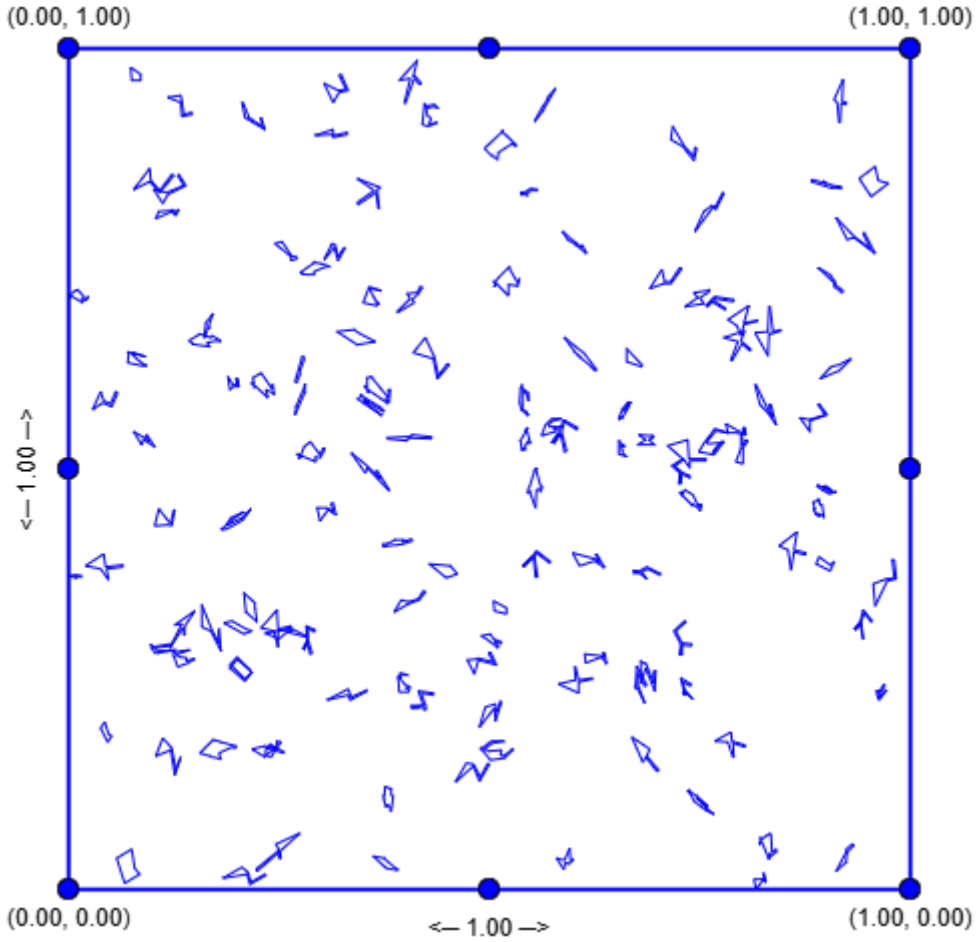

Figure 3.3. Elongated polygons generated by the flow-aligned method. A TIN is built from boundary points, edges perpendicular to the flow are selected, a centerline is extracted from edge midpoints, and lateral offsets form.

D. Mixed-Type Polygon

To further enhance the diversity of the synthetic spatial dataset, the system includes a Mixed-Type Polygon Generation mode. The goal is to emulate heterogeneous environments where different geometric primitives co-exist in space, as in urban tiles that contain rectilinear building footprints together with irregular parks, water bodies, or land-use patches. This mode allows the interleaved creation of multiple geometric types, such as bounding boxes and random polygons, within a single dataset. The primary objective is to simulate heterogeneous environments, where different geometric primitives coexist mirroring real-world scenarios such as urban landscapes, land parcels, and infrastructure planning.

The generation logic is controlled by the user-specified geometry type "mixed" and iterates over a desired number of geometries. For each instance, a random decision is made

between generating a box or a polygon, based on a uniform selection from a list of supported types.

The process is as follows:

1. **Random Point Sampling**: A random coordinate is generated within the spatial domain using the configured distribution (e.g., uniform, clustered). This coordinate serves as the anchor or center point for geometry.
2. **Geometry Type Selection**: A geometry type is randomly selected from the set ("box", "polygon"). Each type triggers a distinct generation logic:
    - **Box Generation**: The system constructs a bounding box by sampling a random size within a predefined range (typically 5% to 20% of the bounding box extent). The size is applied symmetrically along each dimension to compute the minimum and maximum corner coordinates. The result is an axis-aligned rectangle centered at the anchor point.
    - **Polygon Generation**: A polygon is generated using the irregular shape algorithm described previously (see Section 3.1.A). For each polygon:
        - A pair of random radii determines the size range.
        - Random values for irregularity and spikiness introduce controlled randomness.
        - A random number of vertices (typically between 3 and 15) defines the shape complexity.
        - The anchor point serves as the polygon's center, and the shape is written using a specialized sink (e.g., PointToPolygonSink), which applies the generation logic described in **Algorithm 3.4**.

Algorithm 3.4 Mix-Type polygon generator

Require:

$numGeometries$: $number\ of\ geometries\ to\ generate$

$generator$: $spatial\ point\ generator$

$scaled\ maxsize$: $scaling\ factor\ based\ on\ bounding\ box$

$sink$: $geometry\ output\ interface$

Ensure: $A\ set\ of\ randomly\ generated\ boxes\ and\ polygons$

1: **if** $geometryType = \text{“}mixed\text{”}$ **then**

2: $possible\ geometries \leftarrow [\text{“}box\text{”}, \text{“}polygon\text{”}]$

3: **for** $i = 1\ to\ numGeometries$ **do**

```
        coordinates ← generator.generate point
        selected geometry ← random choice from possible geometries
        if selected geometry = "box" then
            minCoordinates ← [ ]
            maxCoordinates ← [ ]
            dimensions ← length of coordinates
            for d = 1 to dimensions do
                minsize ← 0.05 × scaled maxsize
                maxsize ← 0.2 × scaled maxsize
                size ← Uniform(minsize, maxsize) / 2
                minCoordinates.append(coordinates[d] − size)
                maxCoordinates.append(coordinates[d] + size)
            end for
            sink.writeBox(minCoordinates, maxCoordinates)
        else if selected geometry = "polygon" then
            r1, r2 ← Uniform(0.01 × scaled maxsize, scaled maxsize)
            min radius ← min(r1, r2), max radius ← max(r1, r2)
            irregularity ← Uniform(0.1, 1.0)
            spikiness ← Uniform(0.1, 1.0)
            min vertices, max vertices ← random integers in [3, 15]
            polygon sink ← PointToPolygonSink(sink, min radius, max radius,
                irregularity, spikiness, min vertices, max vertices)
            polygon sink.writePoint(coordinates)
        end if
    end for
end if
```

As illustrated in **Figure 3.4**, the mixed-type generation produces a spatial composition of simple rectangular boxes and irregular polygons distributed across the same area. The resulting pattern effectively demonstrates the coexistence of multiple geometric forms within a shared spatial domain, successfully replicating the heterogeneity typical of real-world spatial structures such as urban blocks and open spaces.

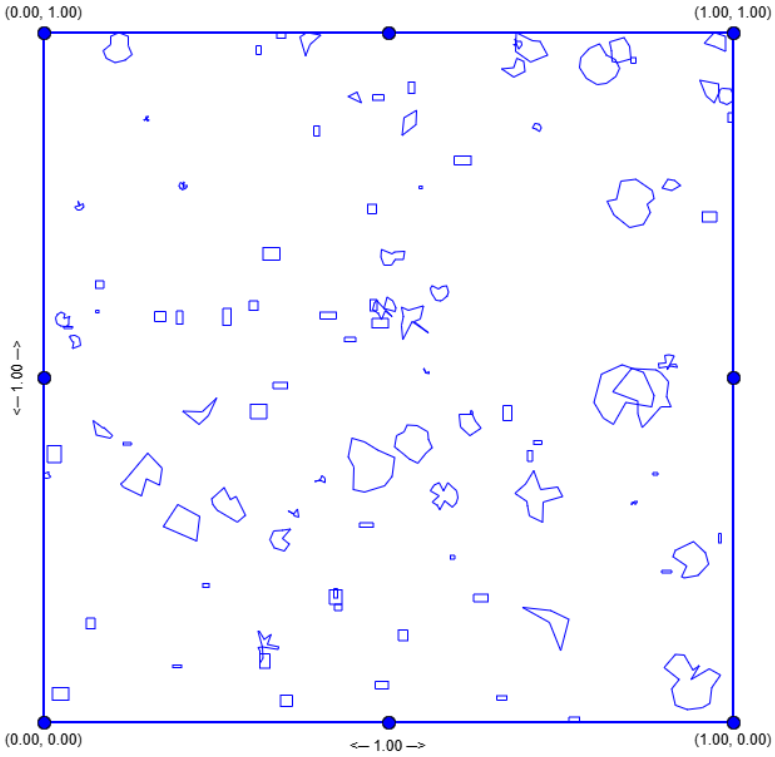

Figure 3.4. Example of Mixed-Type Polygon Generation. The dataset includes a random combination of axis-aligned boxes and irregular polygons distributed within a common bounding area, illustrating the coexistence of heterogeneous geometric primitives.

E. Other Polygon Generation Techniques

In addition to the core generation methods described above, the system also incorporates a set of experimental polygon generation techniques. These methods were explored in a research context to assess their potential for further enriching the geometric diversity of synthetic spatial datasets. Although not integrated into the main pipeline, they demonstrate additional possibilities for shape manipulation and dynamic geometry simulation.

The techniques include:

- **Sliding Algorithm Sink**: This method [14] simulates sliding or moving geometries by applying controlled transformations to pre-existing polygons. It is useful for modeling temporal dynamics, such as the progression of a moving front, deformation under physical forces, or simulation of motion-driven spatial processes.
- **Minkowski Difference Front**: This technique uses basic morphological geometry. Given a base polygon and a small "structuring element" (for example a disk, a square, a diamond, or a star-shaped kernel), we slide the kernel inside the polygon and keep the set of kernel centers where the kernel still fits entirely inside the polygon. The

result is the polygon eroded by that kernel, also known as the Minkowski difference. With a disk kernel this is simply an inward buffer. With non-circular kernels the boundary develops directional facets and fine detail, which increases shape complexity in a controlled way.

- **Random Polygon with Vertex Constraints**: A variation of the random polygon generator, this method [15] introduces explicit constraints on vertex spacing to ensure minimum distances between consecutive points. It allows the generation of more evenly distributed and realistic polygonal shapes, especially useful in scenarios where sharp angles or collapsed edges must be avoided.

These methods were implemented and tested in a preliminary, research-oriented setting, and while they are not part of the core generation framework, they illustrate the extensibility of the system. Future work may integrate these techniques more systematically or explore their application in domain-specific simulations.

Overall, the synthetic geometry generation framework presented in this section enables the creation of diverse, controllable, and application-specific spatial structures. By combining multiple generation strategies ranging from irregular and Voronoi-based polygons to elongated and mixed-type geometries the system provides flexible tools for simulating real-world spatial phenomena. To ensure geometric validity, a post-processing phase is applied, correcting topological defects such as self-intersections, decomposing invalid shapes, and trimming the final output to match the desired number of polygons. These steps guarantee that the resulting datasets are robust and suitable for downstream tasks such as machine learning, spatial querying, or geospatial benchmarking.

Importantly, many of the polygonal generation techniques described above depend on an underlying point generation process, either as seed points (e.g., Voronoi, irregular polygons) or as geometric anchors (e.g., for boxes and movement-based methods). The distribution of these initial points plays a critical role in determining the structure and diversity of the resulting spatial data.

We now turn to the synthetic data distribution models used in this system, which define how points are generated across the spatial domain. These models serve either as standalone spatial datasets or as the foundation for shape construction, supporting a range of real-world simulation needs.

3.1.2 Data Distribution

To support the generation of diverse and application-specific spatial patterns, this study incorporates four distinct spatial point distribution models. Each model reflects different types of real-world spatial phenomena and serves as a basis for constructing either standalone point datasets or seed-based polygonal geometries.

A. Uniform

The uniform distribution model generates points randomly across the entire extent of the spatial bounding box $[x_{min}, y_{min}, x_{max}, y_{max}]$, ensuring that each location within this domain has equal probability of selection. For each point, the x and y coordinates are independently sampled from a uniform distribution:

$$x \sim U(x_{min}, x_{max}), \qquad y \sim U(y_{min}, y_{max})$$

This method produces a homogeneous spatial distribution with no inherent clustering or directional bias. The resulting pattern is spatially isotropic and often used to simulate evenly spaced phenomena such as sensor grids or baseline datasets for benchmarking.

B. Gaussian

The Gaussian (normal) distribution model samples each coordinate from a normal distribution centered at a mean location $(\mu_{x,}\mu_y)$, with specified standard deviations (σ_x, σ_y). Each point is generated using:

$$x \sim N(\mu_x, \sigma_x^2), \qquad y \sim N(\mu_y, \sigma_y^2)$$

In implementation, the Box–Muller transform is used to obtain normally distributed samples from uniform random values:

$$z = \mu + \sigma\sqrt{-2lnU_1} \cdot sin(2\pi U_2), \qquad where\ U_1, U_2 \sim U(0,1)$$

This model can capture clustered spatial behaviors observed in urbanization, vegetation density gradients, or other naturally occurring aggregations.

C. Diagonal

The diagonal distribution is designed to generate spatially aligned data along the main diagonal of a bounding box $[x_{min}, y_{min}, x_{max}, y_{max}]$, simulating directional spatial patterns such as linear infrastructures. Each point follows one of two paths:

1. Deterministic alignment (with probability p)
2. Perturbed offset using Gaussian noise (with probability $1 - p$)

This behavior is controlled by a Bernoulli random variable:

$$B \sim Bernoulli(p)$$

In the noisy case (B=0), vertical perturbation is added:

$$\epsilon \sim N(0, \sigma^2), \qquad \sigma = buffer \cdot \frac{min(w, h)}{\sqrt{2}}$$

$$y = y + \epsilon$$

Where $w = x_{max} - x_{min}]$ and $w = y_{max} - y_{min}]$ are the width and height of the bounding box.

The buffer parameter controls the intensity of noise around the diagonal, with higher values yielding broader dispersion. The final coordinates are clamped to ensure they remain within the bounding box:

$$x = min(max(x, x_{min}),\ x_{max}), \qquad y = min(max(y, y_{min}),\ y_{max})$$

D. Clustered

The clustered distribution model simulates localized groupings of points around randomly generated centroids $(c_i^x,\ c_i^y)\ for\ i = 1, \dots, K_i$, where K is the number of clusters. Each cluster's points are generated using polar coordinates with randomized angles and radius:

$$\theta \sim U(0,2\pi), \qquad r \sim U(0, R)$$

$$x = c_i^x + r\ \cos\theta\,, \qquad y = c_i^y + r \sin\theta$$

Where R is the maximum intra-cluster radius. This formulation enables modeling of both tightly packed and widely dispersed clusters, effectively capturing spatial heterogeneity.

These distribution models not only serve as foundations for polygon generation but can also be used independently for tasks requiring synthetic spatial point clouds or geostatistical simulations.

3.2 Data-Driven Generation

In this section, we introduce a nonparametric method for synthetic data generation based on user-uploaded spatial datasets. The proposed approach extends the system's capabilities beyond predefined parametric models by leveraging empirical statistics to reproduce both marginal distributions and inter-variable dependencies. The method is formalized as a data-driven generation pipeline rooted in empirical copula theory and frequency-based sampling.

Following the description of the generation algorithm, we present a complementary feature extraction framework that captures geometric characteristics such as size, compactness,

irregularity, and vertex complexity from input polygons. These features enable a morphologically aware generation process, capable of producing realistic and diverse shapes.

Finally, to quantitatively evaluate the alignment between original and synthetic datasets, we introduce a similarity assessment metric, the Distributional Geometry Alignment Score (DGAS). This metric assesses both distributional and structural consistency, providing an interpretable and robust evaluation of the generative fidelity.

3.2.1 Empirical Copula Method

In this method, the joint distribution of the uploaded data is approximated without relying on any predefined parametric copula models like Gaussian. Instead, the algorithm constructs empirical cumulative distribution functions (ECDFs) [16] for each variable to map the data into a [0,1] range, thereby normalizing the marginals. It then uses frequency tables to discretize these marginals into intervals with associated probability. To generate new data, a row from the ECDF-transformed dataset is randomly selected, and for each variable, a new value is sampled uniformly within the corresponding interval from the frequency table. This process preserves both marginal distributions and the dependency structure (copula-like behavior) across variables.

Importantly, this approach reproduces the multivariate statistical behavior of the original data without making assumptions about the underlying parametric form and aligns with the core copula principle: separating marginals from dependency structure. Although it doesn't use copulas in a strict mathematical sense, it emulates their function by reconstructing the dependence structure empirically. As demonstrated in the paper, this method is effective for generating synthetic data that retains both the univariate and multivariate properties of the original dataset, making it valuable for data augmentation, privacy-preserving analytics, and generative simulations.

The main goal of **Algorithm 3.5** is to allow users to upload their own spatial datasets typically consisting of 2D points and generate new synthetic data that preserve both the marginal distributions and the underlying dependency structure of the original data. The method begins by estimating the empirical cumulative distribution function (CDF) [17] for each variable, thereby mapping the original values to the unit interval [0,1], while maintaining uniform marginals. Then, for each variable, a frequency table is constructed through histogram binning, capturing the empirical distribution without assuming any parametric form. To synthesize new samples, the algorithm randomly selects rows from the transformed CDF-matrix and, for each variable, determines the corresponding histogram interval. It then samples a value uniformly

within this interval, thereby mimicking the local distribution of the original data. This approach effectively models joint dependencies using the empirical copula structure, even though no explicit copula function is used. Finally, if a new spatial bounding box is provided, the synthetic data are scaled accordingly using affine transformation. This nonparametric framework ensures that the generated data shares the same statistical properties and spatial behavior as the input dataset, making it suitable for downstream tasks such as geospatial simulation, visualization, or augmentation.

Algorithm 3.5 Upload-Based Distribution Matching

Require:

X: $Dataset\ with\ columns\ x\ and\ y$,

$bins$: $Integer\ >\ 0$,

N: $Number\ of\ synthetic\ samples\ to\ generate\ (N\ >\ 0)$,

$bounding\_box$: $Optional\ list\ [min\_x, min\_y, max\_x, max\_y]$

Ensure: $X\_generated$: $New\ dataset\ of\ N\ points\ with\ the\ same\ structure\ and\ empirical\ distribution\ as\ X$

1: **if** $X\ does\ not\ contain\ 'x'\ and\ 'y'\ columns$ **then**
2: **raise** $ValueError("Input\ data\ must\ include\ 'x'\ and\ 'y'\ columns.")$
3: **end if**
4: **if** $bins\ \leq\ 0\ or\ N\ \leq\ 0$ **then**
5: **raise** $ValueError("bins\ and\ N\ must\ be\ greater\ than\ 0.")$
6: **end if**
7: $old\_bbox\ \leftarrow\ (X['x'].min(), X['y'].min(), X['x'].max(), X['y'].max())$
8: **for** $each\ column\ i\ in\ ['x', 'y']$ **do**
9: $x\_sorted, F\ \leftarrow\ empirical\_cdf(X[i])$
10: **for** $each\ value\ z\ in\ X[i]$ **do**
11: **replace** $z\ with\ corresponding\ F[z]$
12: **end for**
13: **end for**
14: **for** $each\ column\ i\ in\ ['x', 'y']$ **do**
15: $construct\ frequency\ table\ with\ bins\ \rightarrow\ Freq\_abs, Freq\_rel, Freq\_acum$
16: **end for**
17: **initialize** $empty\ dataset\ X\_generated$
18: $generate\ N\ random\ indices\ from\ [0, len(X))\ \rightarrow\ list\_N$

for $each\ sub\_n\ in\ list\_N$ **do**
    $initialize\ empty\ list\ sample\_point$
    for each column i in ['x', 'y'] do
        $h \leftarrow transformed\ value\ from\ matrix\_F[sub\_n, i]$
        $find\ interval\ such\ that\ Freq\_acum \geq h$
        if $no\ such\ interval$ **then** $interval \leftarrow -1$
        $[lim\_inf, lim\_sup] \leftarrow bounds\ of\ interval$
        $sample\ value\ v \sim Uniform(lim\_inf, lim\_sup)$
        $append\ v\ to\ sample\_point$
    end for
    $append\ sample\_point\ to\ X\_generated$
end for
if $bounding\_box\ is\ provided$ **then**
    $scale\ X\_generated\ from\ old\_bbox\ to\ bounding\_box$
end if
return $X\_generated$

Based on this algorithm framework, the implemented system supports multiple modes of synthetic data generation. Specifically, the upload-based distribution matching algorithm has been successfully applied in the following contexts:

- **Point-to-Point Generation**: Given a set of input 2D points, the system generates new synthetic point sets that statistically replicate the spatial distribution of the original dataset. This is particularly useful for augmenting sparse spatial data or simulating variations within a known spatial extent. The implementation steps for this distribution-based generation process are described in Algorithm 3.6.

Algorithm 3.6 Point-to-Point Generation

Require: $Point\ dataset\ X\ with\ columns\ 'x', 'y';\ number\ of\ bins;\ N\ samples$
$optional\ bounding\ box$
Ensure: $Xsynthetic$: $synthetic\ points$
$Validate\ input\ columns$
$Xsynthetic \leftarrow Apply\ Empirical\ Distribution\ Matching\ on\ X$
if $bounding\ box\ provided$ **then**
    $Rescale\ Xsynthetic\ to\ bounding\ box$

5: **end if**

6: **return** $Xsynthetic$

- **Box-to-Box Generation**: For uploaded datasets consisting of rectangular (box-shaped) geometries, the centroids are likewise extracted and passed through the distribution matching process. The synthetic centroids are then used to generate new boxes with dimensions derived from the statistical properties (e.g., average size, aspect ratio) of the original set. This allows the system to replicate grid-like or structured layouts commonly observed in applications such as field plots, urban blocks, or sensor grids. The step-by-step procedure for this generation method is described in Algorithm 3.7.

Algorithm 3.7 Polygon-to-Polygon Generation

Require: $Set\ of\ polygons\ P, bins, N, bounding\ box$

Ensure: $Synthetic\ polygons\ Psynthetic$

1: $Extract\ box\ centers\ C\ \leftarrow\ \{c1, \ldots, cn\}$

2: $Csynthetic\ \leftarrow\ Apply\ Empirical\ Distribution\ Matching\ on\ C$

3: $Estimate\ average\ box\ size\ s\ from\ B$

4: **for** $all\ cj\ \in\ Csynthetic$ **do**

5: $\quad Generate\ axis-aligned\ rectangle\ centered\ at\ cj\ using\ size\ s$

6: **end for**

7: **return** $Bsynthetic$

- **Point-to-Voronoi Generation**: The synthetic points produced by the algorithm are used as seed points for constructing Voronoi polygons. These tessellations preserve the same spatial density and layout as the uploaded data while ensuring spatial contiguity and edge-sharing, making them well-suited for simulations involving spatial partitioning or proximity-based analysis. The generation pipeline for this method is detailed in Algorithm 3.8.

Algorithm 3.8 Point-to-Voronoi Generation

Require: $Point\ dataset\ X, bins, N, bounding\ box, shrink\ factor$

Ensure: $Voronoi\ polygons\ Pvoronoi$

1: $Xsynthetic\ \leftarrow\ Apply\ Empirical\ Distribution\ Matching\ on\ X$

2: $Pvoronoi\ \leftarrow\ GenerateVoronoi(Xsynthetic, bounding\ box, shrink\ factor)$

3: **return** $Pvoronoi$

- **Polygon-to-Polygon Generation**: When the uploaded file contains polygonal geometries, the system extracts representative centroids and applies the same empirical generation method to produce new centroid positions with similar statistical properties. These new points are then used to generate polygons either as rectangles or as irregular shapes depending on the structural characteristics of the original dataset (e.g., squareness, irregularity). The resulting synthetic polygons inherit the size, spatial dispersion, and complexity of the input, while incorporating controlled generative variation through sampled geometric parameters. The detailed steps of this approach are presented in Algorithm 3.9.

Algorithm 3.9 Polygon-to-Polygon Generation

Require: $Set\ of\ polygons\ P, bins, N, bounding\ box$

Ensure: $Synthetic\ polygons\ Psynthetic$

$Extract\ centroids\ C\ \leftarrow\ \{c1, \ldots, cn\}\ from\ P$

$Csynthetic\ \leftarrow\ Apply\ Empirical\ Distribution\ Matching\ on\ C$

for $all\ cj\ \in\ Csynthetic$ **do**

  if $original\ polygons\ are\ rectangular$ **then**

    $Generate\ rectangle\ centered\ at\ cj$

  else

  $Sample\ parameters: size, irregularity, spikiness, vertices$

  $Generate\ irregular\ polygon\ at\ cj$

  end if

end for

return $Psynthetic$

This flexible application of the empirical copula-based generation pipeline capable of transforming both point- and shape-based inputs into statistically consistent outputs demonstrates the robustness and generality of the method across a wide range of geospatial simulation tasks.

In the Voronoi generation workflow, once a set of synthetic points is produced using the empirical method, these serve directly as inputs to the Voronoi tessellation algorithm developed earlier in this study. The algorithm then constructs a partition of the space where each cell represents the area closest to a given synthetic seed. As a result, the generated Voronoi

polygons preserve not only the global spatial distribution but also the local neighborhood structure observed in the original data.

Similarly, in the polygon-to-polygon and box-to-box generation pipelines, the workflow begins by extracting centroids from the uploaded geometries. These centroids are processed using the same upload-based distribution matching algorithm to generate a new set of statistically consistent points. The resulting points are subsequently used to construct new geometries by invoking the shape generation algorithms previously developed. For irregular polygons, the algorithm synthesizes shapes with controlled spikiness, irregularity, and vertex complexity; for rectangular shapes, it generates axis-aligned bounding boxes with appropriate scaling.

This two-stage generative process distribution matching followed by geometry constructions that the synthetic outputs reflect not only the statistical profile of the original data but also its spatial structure and geometric diversity. In all cases, the previously introduced modules (e.g., Voronoi tessellation and polygon/box generation) operate as downstream components that transform statistically coherent synthetic points into high-fidelity spatial geometrie**s**.

3.2.2 Feature Extraction

Beyond the use of nonparametric distribution matching techniques for point synthesis, the proposed framework incorporates a feature-driven generative mechanism that enables the creation of synthetic polygons based on the morphological characteristics of existing geometries. This approach recognizes that polygonal shapes encode rich geometric and topological information, and that replicating this structure is essential for high-fidelity simulation and data augmentation.

The process begins by applying a detailed feature extraction pipeline to a set of polygon geometries in Algorithm 3.10. For each polygon, a suite of descriptive metrics is computed, including size (as the maximum radial distance from the centroid), number of vertices, aspect ratio, perimeter, area, compactness, and irregularity, which quantifies angular and radial variance relative to a regular polygon. Additional binary descriptors, such as convexity and equilaterality, are derived using geometric rules and tolerance-based comparisons. Collectively, these features form a high-dimensional representation of shape, enabling a comprehensive analysis of spatial and structural diversity within the dataset.

Algorithm 3.10 Feature Extraction from Polygons

Require: $Set\ of\ polygons\ P$

Ensure: $Feature\ table\ F$

for $each\ Pi\ \in\ P$ **do**

  $Compute\ area, perimeter$

  $Compute\ centroid\ coordinates\ (xi, yi)$

  $Count\ vertices\ vi$

  $Compute\ aspect\ ratio\ from\ bounding\ box$

  $Compute\ compactness\ C_i = \frac{4\pi \cdot area}{perimeter^2}$

  $Determine\ convexity\ using\ cross - product\ checks$

  $Determine\ equilateral\ property\ via\ edge\ uniformity$

  $Compute\ irregularity\ using\ angular\ and\ radial\ variance$

  $Add\ feature\ vector\ to\ F$

end for

Once features are extracted in Algorithm 3.11, the system constructs empirical statistical models to capture their distributions and interdependencies. For continuous attributes such as size, compactness, and aspect ratio, normalized histograms are used to model their marginal distributions. Additionally, conditional dependencies such as the relationship between polygon size and number of vertices are quantified through binned aggregations and summary statistics. To capture the spatial variation in shape classes, a kernel density estimation (KDE) [18] process is applied separately to convex and non-convex polygons, using centroid coordinates as input. This KDE-based analysis provides a probabilistic estimate of "spikiness" at each location, which informs the irregularity and convexity of the generated shapes.

Algorithm 3.11 Estimate Spikiness via KDE of Centroids

Require: $Convex\ and\ non - convex\ centroids$

Ensure: $KDE - based\ spikiness\ estimator$

$Fit\ KDEconvex\ on\ convex\ centroids$

$Fit\ KDEnonconvex\ on\ non - convex\ centroids$

for $each\ centroid\ (x, y)$ **do**

  $Compute\ pc\ =\ KDEconvex(x, y)$

  $Compute\ pn\ =\ KDEnonconvex(x, y)$

  $Compute\ spikiness\ s\ =\ 1\ -\ \frac{p_c}{p_c + p_n}$

end for

Based on this statistical modeling in Algorithm 3.12, a new set of shape descriptors is synthesized by sampling from the learned distributions. The size of each new polygon is drawn from the empirical histogram, while the number of vertices is sampled conditionally on size. The convexity of each polygon is estimated to be using the KDE density maps, allowing spikiness to be assigned in a data-driven manner. If the original dataset predominantly consists of rectangular shapes identified using side-length uniformity and angular checks the system switches to an axis-aligned rectangle generator to preserve the geometric consistency. Otherwise, an irregular polygon generator is invoked, using the sampled feature set (size, spikiness, irregularity, vertices) to control the shape synthesis process.

Algorithm 3.12 Statistical Analysis of Features

Require: $Feature\ table\ F$

Ensure: $Histogram\ and\ conditional\ mappings$

1: **for** $each\ feature\ f\ \in\ F$ **do**
2: $\quad Compute\ histogram\ of\ f$
3: **end for**
4: $Compute\ correlation\ matrix\ of\ numeric\ features$
5: $Group\ by\ convexity\ and\ compute\ group\ means$
6: $Discretize\ size\ into\ bins\ and\ compute\ mean, std\ of\ vertex\ count$

This feature-aware in Algorithm 3.13 pipeline enables the generation of synthetic polygons that retain the global and local characteristics of the original dataset. Unlike purely spatial generation methods, this approach reproduces both the morphological diversity and the statistical structure of input geometries, making it highly suitable for geospatial simulation, synthetic data augmentation, and machine learning pretraining tasks. By coupling geometric analysis with nonparametric modeling, the framework strikes a balance between fidelity and generative flexibility.

Algorithm 3.13 Feature-Aware Synthetic Polygon Generation

Require: $Histograms, conditional\ tables, KDE\ models, N\ synthetic\ samples$

Ensure: $Synthetic\ polygons\ P'$

1: **for** $i\ =\ 1\ to\ N$ **do**
2: $\quad Sample\ size\ si\ from\ histogram$
3: $\quad Sample\ number\ of\ vertices\ vi\ conditioned\ on\ si$

4: $Sample\ centroid\ (xi\ , yi)\ from\ copula\ distribution$

5: $Estimate\ spikiness\ spi\ via\ KDE$

6: **if** $polygons\ are\ mostly\ square$ **then**

7: $Generate\ rectangle\ at\ (xi, yi)\ of\ size\ si$

8: **else**

9: $Generate\ irregular\ polygon\ using\ (si, ui, spi)$

10: **end if**

11: $Add\ polygon\ to\ P'$

12: **end for**

3.2.3 Similarity Assessment

To ensure that the generated synthetic polygonal data faithfully reflects the statistical and geometric structure of the original input, we introduce a robust similarity assessment metric: the **Distributional Geometry Alignment Score**. This metric is specifically designed to quantify the alignment between two datasets original and synthetic in terms of their feature distributions and structural interdependencies, rather than relying solely on raw spatial proximity.

The DGAS is computed by evaluating two complementary components:

A. **Feature Distribution Similarity:**

This term measures the alignment of marginal distributions across selected geometric features such as polygon size, compactness, number of vertices, or aspect ratio. Each feature is normalized using Min-Max scaling to account for scale discrepancies. The similarity is quantified using the inverse of the 1D Wasserstein distance (Earth Mover's Distance) [19], which provides a principled way to compare continuous distributions. Importantly, each feature's contribution is weighted based on its empirical variance, prioritizing more informative variables in the global score.

B. **Feature Structure Similarity:**

Beyond marginal alignment, this component captures how feature interrelationships are preserved between datasets. For this, we compute the pairwise correlation matrices of the selected features in both original and generated datasets. The Frobenius norm [20] of the difference between these matrices serves as a proxy for structural divergence. A normalized score is then derived to express the degree of alignment, where values close to 1 indicate strong preservation of internal dependencies.

The final similarity score is calculated as a convex combination of the two components:

$$DGAS = \alpha \cdot S_{dist} + \beta \cdot S_{struct}, \qquad with\ default\ weights\ \alpha = 0.6,\ \beta = 0.4$$

This formulation balances the importance of preserving both the distributional shape and internal geometric logic of the data.

Unlike classic similarity metrics that rely on spatial proximity (e.g., pointwise Euclidean distance, Hausdorff distance), DGAS intentionally **does not** incorporate absolute positional information. This design decision is grounded in the core principle of non-parametric, distribution-based generation: synthetic geometries are not meant to reproduce the spatial layout of the input, but to statistically match its distributional properties.

When generating new polygons or points within a potentially different bounding box or with randomized spatial allocation, enforcing positional similarity becomes both meaningless and restrictive. Instead, our focus is on maintaining the statistical essence of the input its shape complexity, feature relationships, and geometric behavior rather than its specific location in space.

While this section introduces the similarity assessment mechanism, a more detailed analysis including case studies, quantitative evaluations, and performance benchmarking is presented in the Evaluation section of this thesis.

CHAPTER 4

WEB APPLICATION AND VISUALIZATION

The generator tool includes a web-based user interface that allows users to configure, visualize, and generate synthetic spatial datasets interactively. The top section of the interface provides multiple input fields enabling users to specify generation parameters according to their needs. Users can choose the desired point distribution (Uniform, Gaussian, Diagonal, or Clustered), define the cardinality (number of geometries), select the type of generated geometry (point, polygon, convex, non-convex, Voronoi-based, etc.), and adjust various generation parameters such as average radius, irregularity, and spikiness for polygons. Additionally, users can configure bounding box limits, fix the polygon centers across generations, and apply optional transformations. The interface also supports setting specific parameters depending on the distribution type, such as buffer size for diagonal distributions or cluster radius for clustered distributions.

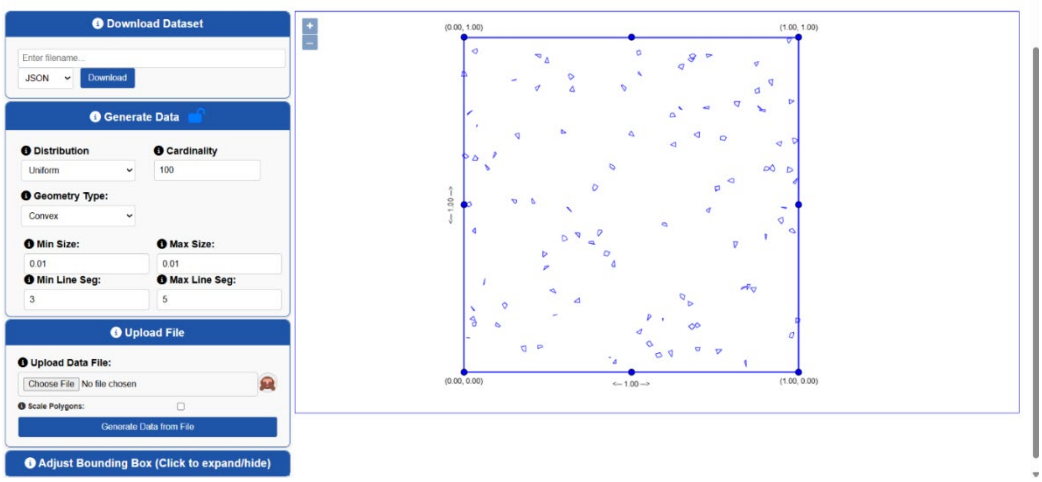

Figure 4.1. Web interface of the synthetic spatial data generation platform. The left panel provides interactive controls for selecting distribution type, geometry configuration, cardinality, and polygon complexity parameters. Users can also upload external datasets and adjust bounding box extents. The right panel displays the generated geometries in real-time within the editable bounding box domain.

A key feature of the web interface is the real-time visualization of the generated data. Once the user defines the configuration parameters and triggers the generation request, a sample of the dataset is visualized on the page using the Open Layers library. Open Layers provides a flexible and dynamic environment for rendering vector geometries, allowing multiple layers to be added, compared, and manipulated. Users can interact with the displayed geometries by zooming, panning, or overlapping different layers for comparative purposes. Furthermore, the visualization updates immediately upon changes in the configuration, ensuring that users can preview the effects of their adjustments without the need for reloading or manual interventions.

To maintain responsiveness, visualization is limited to a manageable number of geometries (e.g., 1000 samples) even if the final dataset is much larger. This approach ensures fast rendering and provides a representative preview without overloading the browser or server. Users also can download the full dataset in formats such as JSON, WKT, or GeoJSON for external analysis.

4.1 Architecture Overview

The architecture of the developed system follows a modular and layered design, tailored to support interactive generation, visualization, and evaluation of synthetic spatial data. It integrates client-side rendering, server-side computation, and data exchange mechanisms in a

cohesive workflow that ensures usability, scalability, and reproducibility. The system has been designed with the dual goal of supporting real-time experimentation and maintaining scientific rigor in the analysis of generated geometries.

At the client level, the web interface is implemented using standard web technologies, including HTML5, CSS, and JavaScript, with OpenLayers [21] serving as the main visualization library. OpenLayers enables efficient rendering and manipulation of vector geometries, allowing users to explore generated datasets interactively. The interface provides a rich set of input controls through which users can define generation parameters such as the type of spatial distribution (e.g., uniform, Gaussian, diagonal, clustered), geometric configuration (e.g., convex polygons, Voronoi diagrams), and structural constraints (e.g., spikiness, irregularity, number of vertices). Input fields dynamically adjust based on the selected options, enabling intuitive and context-aware configuration. A central element of the UI is the bounding box editor, which supports direct manipulation of spatial extents through corner and edge dragging, coupled with real-time geometric regeneration.

On the server side, a Python-based backend is responsible for the actual data generation and similarity assessment. Upon receiving a request, the backend parses the configuration parameters, generates a synthetic dataset according to the selected distribution and geometry settings, and optionally performs transformations such as shrinkage or scaling. When a reference dataset is uploaded, the server also performs similarity evaluation using the Distributional Geometry Alignment Score (DGAS) metric. DGAS quantifies how closely the generated dataset matches the reference in terms of both distributional and structural similarity, providing users with immediate feedback. Additional outputs include statistical feature summaries and visual plots of marginal distributions.

The overall architecture is illustrated in Figure 4.2, which shows the interaction between the client interface and the backend services. The diagram highlights the flow of user input, the geometry generation pipeline, similarity evaluation, and data export mechanisms that enable reproducible experimentation and efficient dataset delivery.

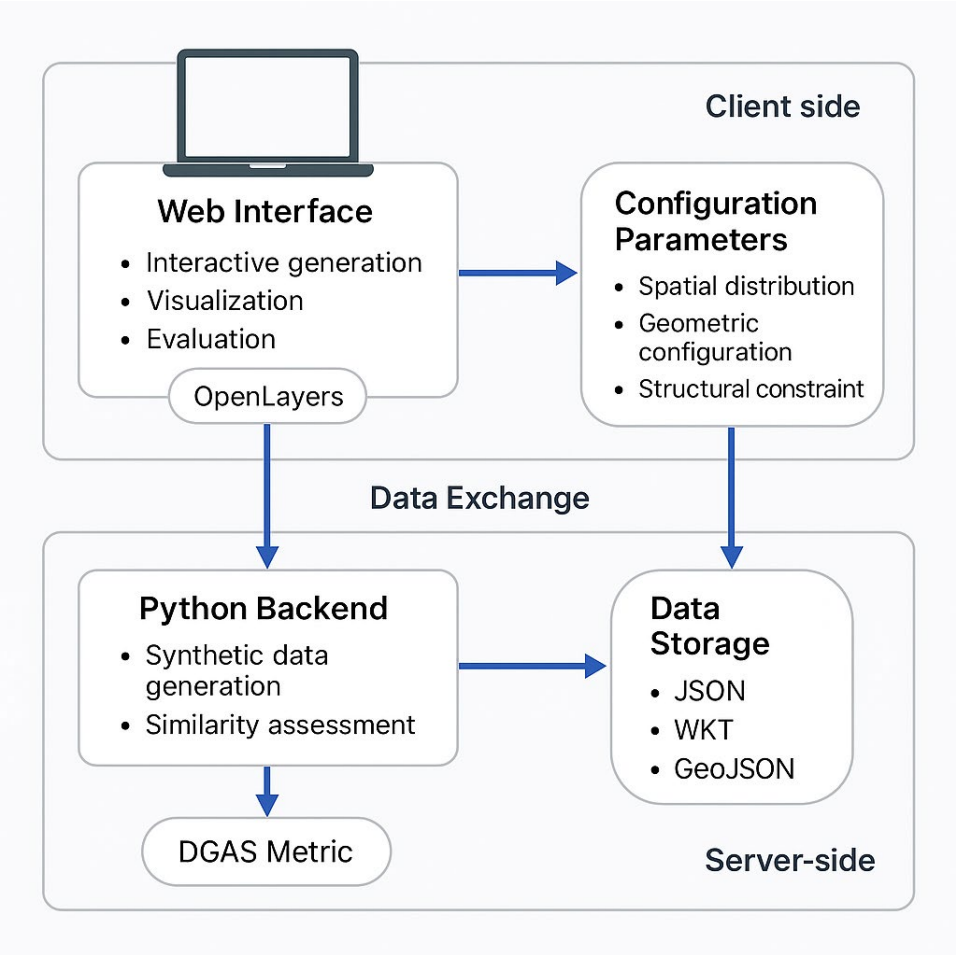

Figure 4.2. System architecture diagram of the synthetic spatial data generation platform. The design integrates a web-based client interface (OpenLayers-based) for interactive parameter input and geometry visualization, a Python backend for geometry generation and similarity evaluation, and mechanisms for reproducibility and data export.

The system further supports reproducibility and traceability through automated permalink generation. All session parameters are encoded in a URL that can be stored or shared, ensuring that the exact dataset can be regenerated in the future or by collaborators. Data exports are available in standard formats such as JSON, WKT, and GeoJSON, facilitating further analysis or integration with external GIS tools.

To maintain responsiveness, the interface renders only a limited number of geometries (typically 1000) for preview purposes, even when the full dataset is significantly larger. This decoupling of visualization from generation allows the interface to remain fast and interactive, while preserving the complete dataset for downstream use.

In summary, the architecture combines scientific flexibility with interactive usability, making the system suitable for both researchers aiming to model spatial phenomena and practitioners requiring customized datasets for simulation, benchmarking, or model validation.

4.2 Web Interface

The developed web interface serves as the central access point for configuring, generating, and visualizing synthetic spatial datasets. It is built using HTML5, JavaScript, and the Open Layers library, providing a modular, responsive, and highly interactive environment. Its design philosophy prioritizes user experience through clarity, parameter visibility, real-time visual feedback, and minimal user friction during iterative testing.

The system consists of four core panels: (1) data generation configuration, (2) file upload and distribution matching, (3) bounding box manipulation, and (4) geometry visualization. Together, they enable seamless experimentation with point- and shape-based data generation strategies.

4.2.1 Generation Interface

The generation interface (**Figure 4.1, left panel**) allows users to define synthetic data properties via an intuitive form. Users can select the distribution type (e.g., uniform, diagonal, Gaussian, cluster), geometry type (e.g., point, box, convex, Voronoi, mixed), and cardinality (number of records). For shape-based geometries, advanced controls such as minimum and maximum radius, line segment counts, and shape complexity parameters (irregularity, spikiness, shrink factor) are dynamically enabled based on the chosen method.

The interface includes a "lock" toggle feature that allows users to regenerate geometries with identical centers but different shapes, enabling controlled variation a useful capability for benchmarking, visual analysis, or sensitivity testing. The generation process is instant and updates the map view in real-time.

4.2.2 Interactive Data Space

The interactive bounding box is a core component of the interface, defining the spatial domain within which all geometries are generated. It is visualized on the map as a blue rectangle with labeled corner coordinates and edge dimensions (Figure 4.1, right). Users can manually reshape the bounding box by dragging its four corner points or adjusting its edges via midpoints. These interactions automatically update the underlying extent ([xmin, ymin, xmax, ymax]), which in turn triggers a regeneration of the synthetic dataset to fit the new spatial domain.

The bounding box serves both as a spatial constraint for generations and as a visual cue for scale and layout control. Internally, any change to its shape updates the generation engine

with a new extent through asynchronous calls. This guarantees coherence between user interactions and generated data, enabling real-time iteration and adjustment without requiring form resubmission.

An alternative to interactive dragging is the numeric bounding box editor shown in **Figure 4.3**, where users can enter precise values for the spatial domain coordinates. This enables exact control in scenarios requiring reproducibility or alignment with known geospatial extents.

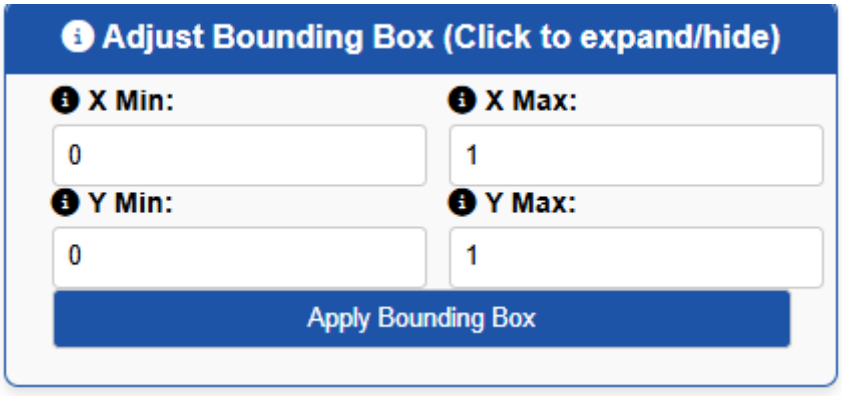

Figure 4.3. Bounding box configuration panel. Users can manually define the spatial extent of data generation by specifying minimum and maximum values for the x and y axes. Upon pressing "Apply Bounding Box", the map and generation domain are immediately updated, ensuring precision control over the spatial boundaries.

Upon submitting new values via the "Apply Bounding Box" button, the map is automatically updated to reflect the new bounding region. The updated bounding box is immediately reflected in subsequent data generation processes, ensuring consistency between user-defined input and rendered geometries. This dual-mode interaction (visual + numeric) supports both exploratory and precision-driven workflows.

4.2.3 Interactive Visualization

The map panel serves as a dynamic visualization environment where all generated geometries are rendered in real-time. The system supports a wide range of geometry types including points, boxes, and polygons with efficient vector rendering, ensuring smooth performance even under high cardinalities. Polygons are controlled by user-defined constraints such as minimum and maximum size, as well as minimum and maximum number of vertices, enabling fine-tuned control over their structural complexity and spatial footprint.

A critical feature enhancing interactivity is the lock toggle (displayed as a padlock icon). When enabled, this feature preserves the centroid locations of all existing geometries during regeneration. While spatial positions remain fixed, users are still allowed to modify key shape parameters including polygon size and number of vertices. This results in updated shapes that retain their original spatial anchoring, making it particularly valuable for controlled experiments where positional stability is essential. The lock ensures that generated polygons are not lost or repositioned, providing users with consistent visual references and repeatable outputs.

When the lock is disabled, both position and shape attributes are randomized, allowing for a full reshuffling of the synthetic dataset. The lock's state dynamically influences several form components, disabling randomization controls and triggering corresponding updates in the visualization layer. This mechanism supports both exploratory and repeatable generation workflows, making the interface suitable for simulation studies, perturbation analyses, and synthetic benchmarking tasks.

4.2.4 Upload Dataset

In addition to forward generation, the interface includes a file upload feature that supports standard formats such as CSV, WKT, and GeoJSON. This functionality allows users to import real-world datasets either point-based or polygonal and visualize them alongside generated data within the same map canvas.

Upon upload, the system automatically detects the geometric type and displays the associated bounding box, which becomes editable through the standard interactive tools. The user may then choose to scale the imported polygons to fit the current generation domain or to preserve their original proportions and coordinates. This feature provides flexibility in aligning heterogeneous datasets with synthetic generation space.

A central capability of the upload module is that it allows users to view and interact with the original dataset directly on the map. This includes full spatial rendering of the imported geometries and the ability to overlay synthetic outputs for comparative analysis. Once the data is visualized, users can select additional generation operations for example, they may choose to compute a Voronoi tessellation based on the uploaded point dataset, enabling structured polygonal partitioning of the space. The resulting Voronoi cells can be optionally scaled via a shrink factor, controlling the tightness or dispersion around each seed point.

Through this upload-and-generate mechanism, the interface bridges the gap between empirical data and synthetic modeling. It enables workflows where users can both replicate and generalize spatial distributions and structural patterns using the same interactive framework. Combined with similarity scoring and visual comparison tools, this functionality supports informed evaluation and reproducible synthetic data experimentation.

CHAPTER 5

EVALUATION

This chapter presents a comprehensive evaluation of the developed system for synthetic spatial data generation. The evaluation is twofold. **Section 5.1** focuses on the performance and scalability of the geometry generation algorithms, assessing their suitability for various simulation tasks. **Section 5.2** evaluates the statistical fidelity of the data-driven generation approach, which aims to reproduce spatial distributions using empirical matching techniques.

All experiments were conducted on a machine equipped with an Intel Core i5-8300H CPU (4 physical cores / 8 threads, base frequency 2.30 GHz), 32 GB RAM, and Windows 11 Pro 64-bit. The implementation was executed in Python 3.10 in single-thread (serial) mode, without explicit parallelization. This ensures that execution time reflects the raw efficiency of each algorithm without influence from parallel optimizations.

Each generation method was tested under increasing cardinality, ranging from **10,000 to 700,000 polygons**, with a maximum of **50 vertices per polygon**. Identical spatial bounds and parameter settings were applied throughout the evaluation.

For the **data-driven** evaluation, two stages were conducted. In the first stage, the system was assessed using synthetically generated points to verify whether it could correctly infer and reproduce the underlying spatial distribution. In the second stage, real spatial data were used to test the model's capability to match authentic spatial patterns and statistical properties. The same two-step procedure was also applied to polygon generation, first validating the method with generated reference shapes and then with real geometric datasets.

5.1 Random Generation Evaluation

The performance of four core polygon generation algorithms Circle-Based, Convex Hull-Based, Random Vertex, and Sliding Generator was evaluated across varying dataset sizes. The results are presented in Table 5.1 and Figure 5.1.

Table 5.1 shows the execution time in seconds for generating different numbers of polygons. The Circle-Based Generator demonstrated superior scalability, completing the generation of 700,000 polygons in under 30 seconds. In contrast, both the Convex and Sliding Generators failed to scale beyond 200,000 due to computational bottlenecks. The Random Vertex Generator showed moderate scalability but suffered from increasing overhead due to collision detection and shape validation steps. **Figure 5.1** visualizes the execution time growth across the four methods. The Circle-Based Generator exhibits a near-linear scaling trend, making it the most suitable choice for large-scale applications.

Table 5.1 Execution time (in seconds) for generating with ranges from 10,000 to 700,000. A dash (–) indicates that the method does not scale beyond that size.

	Cardinality								
	10.000	50.000	100.000	200.000	300.000	400.000	500.000	600.000	700.000
Circle-based	0.59	2.18	5.24	9.75	12.77	17.14	20.84	25.41	**29.22**
Convex	16.79	72.44	159.32	336.07	-	-	-	-	-
Random	2.50	12.24	27.08	45.29	69.30	93.87	118.61	137.78	160.62
Sliding	16.34	86.16	181.61	293.37	-	-	-	-	-

These results clearly indicate that the Circle-Based Generator provides the best trade-off between geometric complexity and execution efficiency. It was therefore selected as the default method for scalable polygon generation in the final system.

In contrast, the Convex Hull and Sliding Generators, while offering precise geometric structures, are better suited for smaller datasets or specialized use cases where geometric rigor outweighs performance constraints.

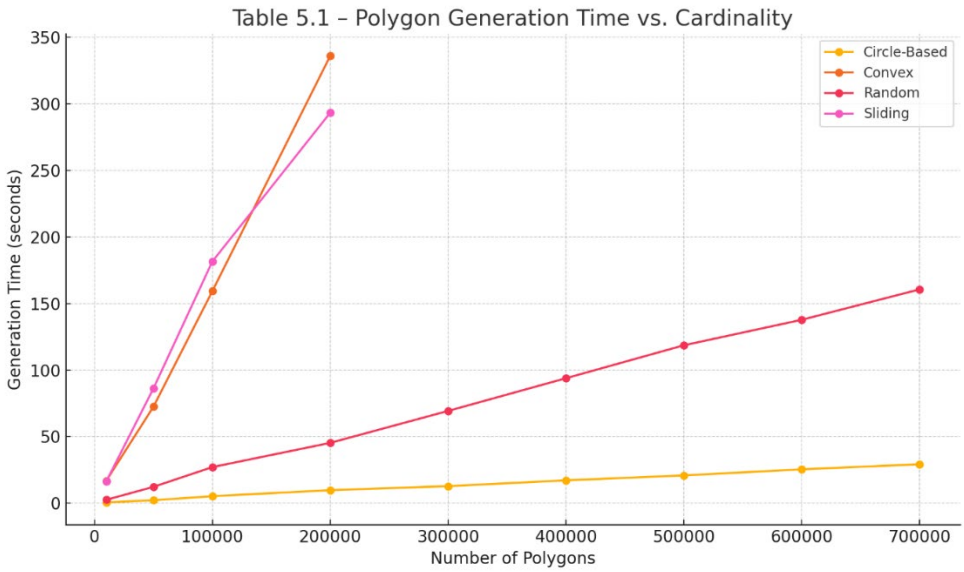

Figure 5.1. Generation time comparison for core polygon generators across increasing cardinality. Circle-Based remains highly efficient, while Convex and Sliding show steep growth due to computational complexity

To support diverse spatial simulation needs, the system also integrates additional methods: Points, Boxes, Voronoi, Elongated, and Mixed-Type generators. Table 5.2 and Figure 5.2 summarize their performance.

Table 5.2 highlights the speed of point and box generators, which remained below 2.5 seconds even for 700,000 instances. These methods are optimal for applications requiring simple geometric entities. The Mixed-Type Generator also showed excellent scalability, effectively combining box and irregular shapes with minimal performance degradation.

Table 5.2 Final Selection – Generation Time (in seconds) for N Polygons Using Different Algorithms N ranges from 10,000 to 700,000 polygons. A dash (–) indicates that the method did not scale beyond that size.

	Cardinality								
Methods	10.000	50.000	100.000	200.000	300.000	400.000	500.000	600.000	700.000
Points	0.01	0.13	0.34	0.76	0.80	0.89	1.23	0.78	0.93
Box	0.02	0.09	0.25	0.60	0.89	1.42	1.52	2.13	2.06
Circle-based	0.59	2.18	5.24	9.75	12.77	17.14	20.84	25.41	29.22
Voronoi	8.44	38.83	56.66	124.21	168.01	236.59	267.43	344.43	429.49
Elongated	21.58	91.43	217.91	-	-	-	-	-	-
Mix-Type	0.11	0.67	1.66	3.07	4.10	5.64	7.59	8.70	10.26

In contrast, the Voronoi Generator exhibited significantly higher execution times, exceeding 400 seconds for 700,000 polygons. Similarly, the Elongated Generator, which produces flow-aligned or infrastructure-like geometries using TIN-based logic, struggled beyond 100,000 polygons due to its high computational complexity.

Figure 5.2 illustrates these findings, confirming that while point, box, and circle-based methods are computationally efficient, Voronoi and elongated generation methods are more suited for domain-specific tasks where structural realism outweighs performance considerations.

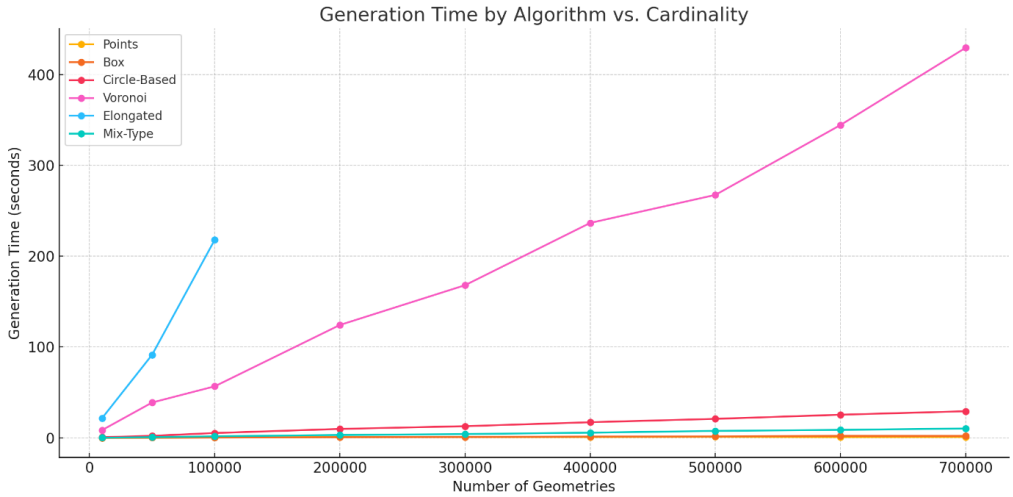

Figure 5.2. Extended performance comparison including additional methods. Circle-Based remains the most scalable for polygons, while Voronoi and Elongated are suitable for high-fidelity or domain-specific tasks despite higher runtime.

The Circle-Based Generator emerged as the most balanced solution, combining fast execution, shape complexity control (via irregularity and spikiness parameters), and robustness. Its lightweight implementation and parameter-rich design enable controlled generation of realistic yet computationally efficient polygon datasets.

In conclusion, the performance evaluation confirms the modular strength of the system: users can select from a spectrum of generators based on task requirements, balancing between scalability, shape fidelity, and structural complexity.

5.2 Data-Driven Generation Evaluation

This section presents a detailed evaluation of the system's data-driven generation capabilities, emphasizing its ability to replicate the statistical and morphological properties of real-world spatial datasets. Unlike parametric techniques, the implemented empirical framework does

not assume a predefined model structure; instead, it aims to preserve both the marginal distributions and the joint dependencies among spatial variables using a nonparametric approach. The evaluation proceeds in three stages: (i) analysis of the distributional similarity between real and synthetic point sets; (ii) assessment of feature extraction fidelity across polygon types; and (iii) quantitative evaluation of structural similarity using the Distributional Geometry Alignment Score (DGAS). These analyses jointly validate the generator's performance in terms of both spatial fidelity and generalizability.

5.2.1 Distribution Evaluation

The first component of the evaluation investigates whether the generated point sets accurately replicate the spatial distribution of original data. Visual comparisons were conducted for five distinct distribution types of circles, spiral, petals, clusters, and moons. In each case, synthetic points were generated from empirical copula-based transformations and compared to the original datasets.

As shown in **Figure 5.3**, the circular dataset, characterized by a double-ring structure with a central void, is successfully approximated by the generator. The synthetic data (blue) follows the same radial density as the original (red), both at the inner and outer rings, without attempting to match specific positions. This confirms that the model captures the overall spatial distribution, despite minor diffusion near the central gap an artifact consistent with probabilistic sampling.

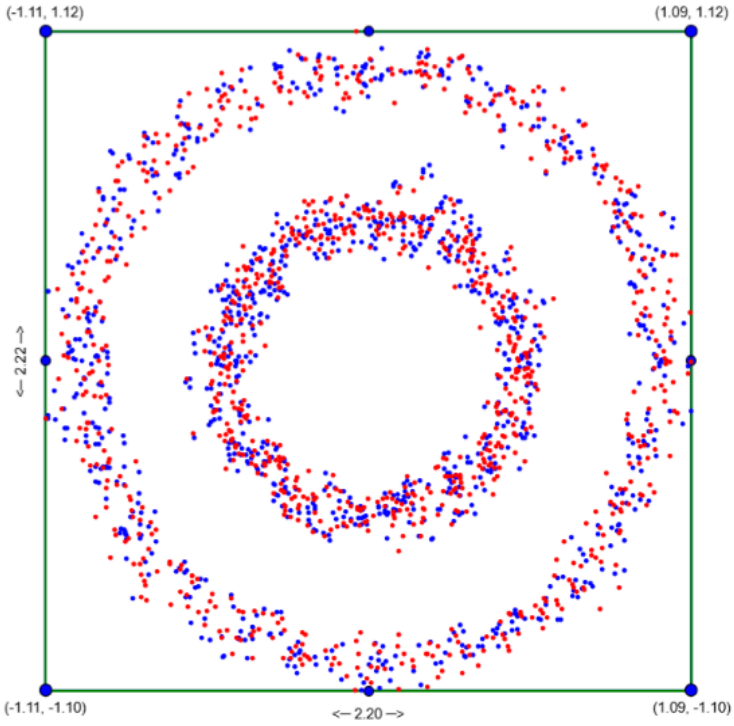

Figure 5.3. Visual comparison between original (red) and synthetic (blue) data for the circular distribution. The generator accurately replicates the radial density and double-ring structure. Minor central diffusion arises from the non-deterministic nature of sampling, without affecting the overall distributional fidelity.

In **Figure 5.4**, the spiral distribution is also well reproduced. The synthetic data accurately follows the curvature and radial growth of the original spiral, including the dense inner coils and sparser outer arms. The alignment between distributions confirms the generator's ability to model structured polar patterns using nonparametric, distribution-based techniques.

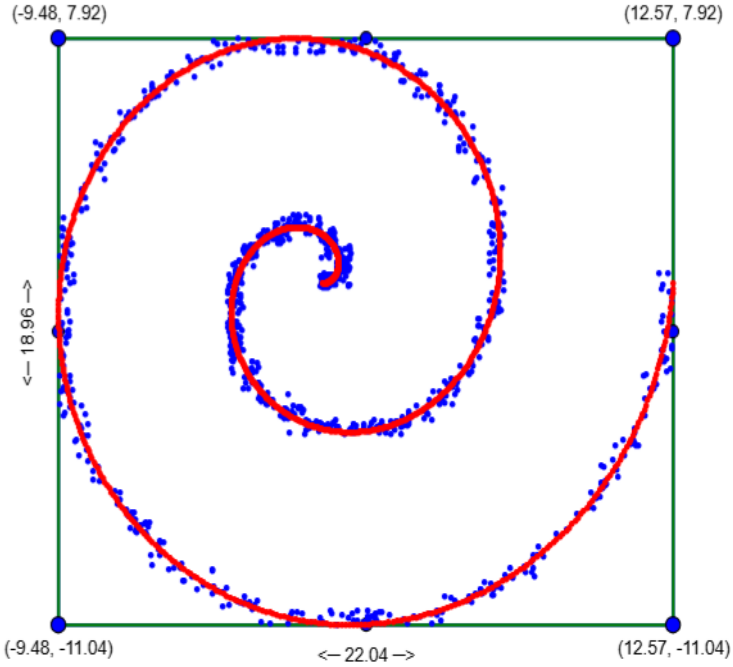

Figure 5.4. Original (red) and synthetic (blue) spiral datasets. The generator successfully captures the winding curvature and radial expansion, maintaining spatial continuity and density progression along the spiral arms.

The petal-shaped dataset shown in **Figure 5.5** further demonstrates the fidelity of the generator. The synthetic points preserve the sinusoidal radial structure and inter-lobe spacing, maintaining angular symmetry and density variation consistent with the original dataset. Slight fluctuations near the origin do not compromise the distributional validity of the result.

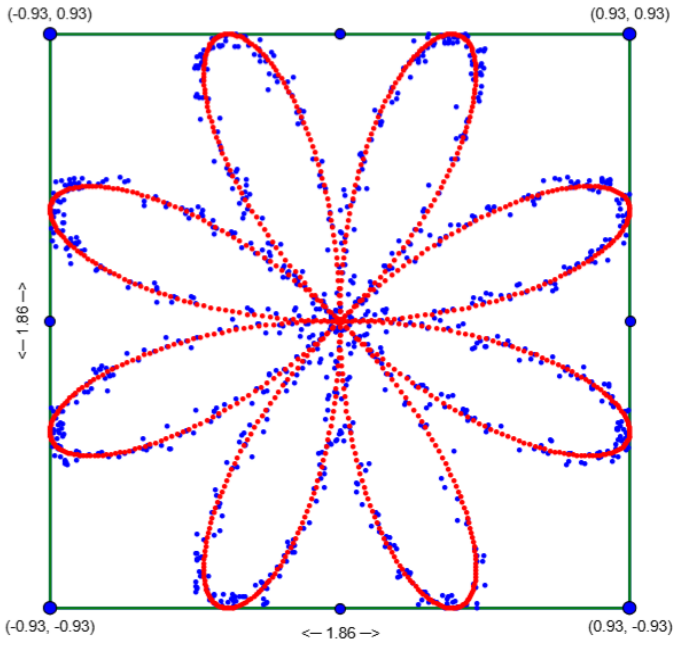

Figure 5.5. Petal-shaped distribution generated from a sinusoidal radial function. The synthetic data reproduces the petal structure, preserving radial symmetry and inter-lobe spacing, with minimal distortion near the center.

In **Figure 5.6**, the clustered distribution is effectively reproduced. The synthetic data maintains the location, density, and spatial extent of the original clusters. While additional micro-clusters may appear due to stochastic effects, the dominant distributional characteristics remain intact. This illustrates the robustness of the generation method when applied to multi-modal spatial data.

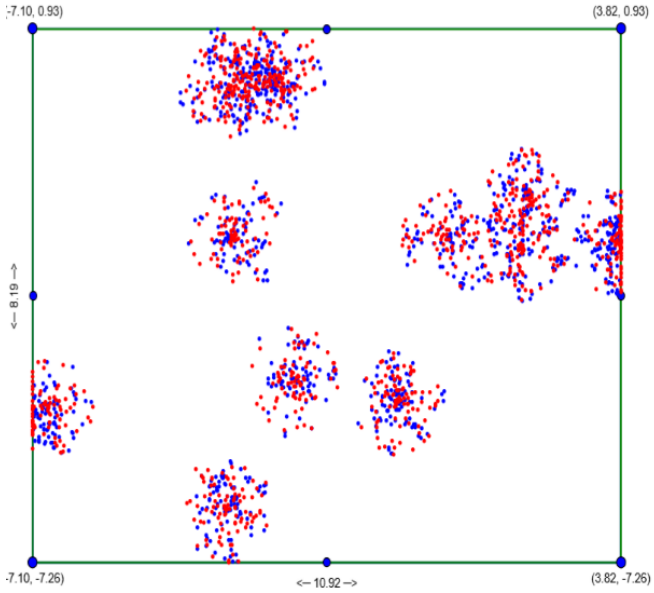

Figure 5.6. Clustered distributions showing real (red) and synthetic (blue) points. The main clusters are faithfully reproduced in terms of location and density, despite some emergence of micro-clusters due to bin-based sampling.

Finally, **Figure 5.7** shows the classic moons dataset. Despite the well-known complexity of this non-linear shape, the generator reproduces the twin arc structure and preserves the overall spatial balance between the two classes. The gap between classes is maintained, and curvature is respected to a high degree, confirming the method's ability to replicate complex geometry-aware distributions.

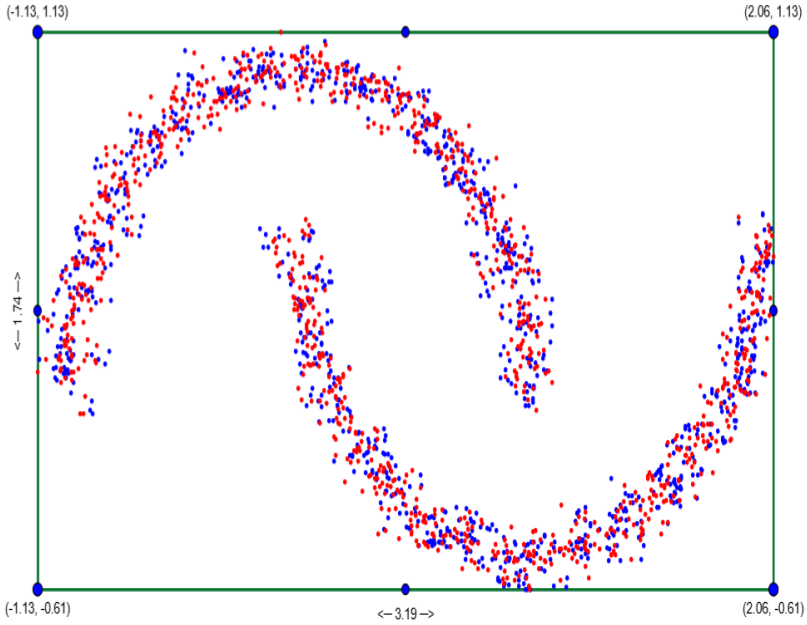

Figure 5.7. Moons dataset comparison. The generator preserves the twin arc structure and class separation, successfully capturing the underlying non-linear geometry of the distribution.

In addition to synthetic reference datasets, the data-driven generator was also tested using ***real spatial data*** to assess its capacity to reproduce authentic geographic distributions. Specifically, a dataset of building centroid points from the city of **Ioannina, Greece** was employed as the empirical reference (Figure 5.7). The red points correspond to the real building locations, while the blue points represent the data-driven synthetic generation derived through the empirical copula transformation. As shown in the figure, the generator successfully captures the elongated and clustered spatial pattern characteristic of the urban structure of Ioannina, maintaining both local density variations and the overall spatial extent of the city. Minor deviations at peripheral regions are attributed to sampling variability and do not affect the overall fidelity of the generated distribution.

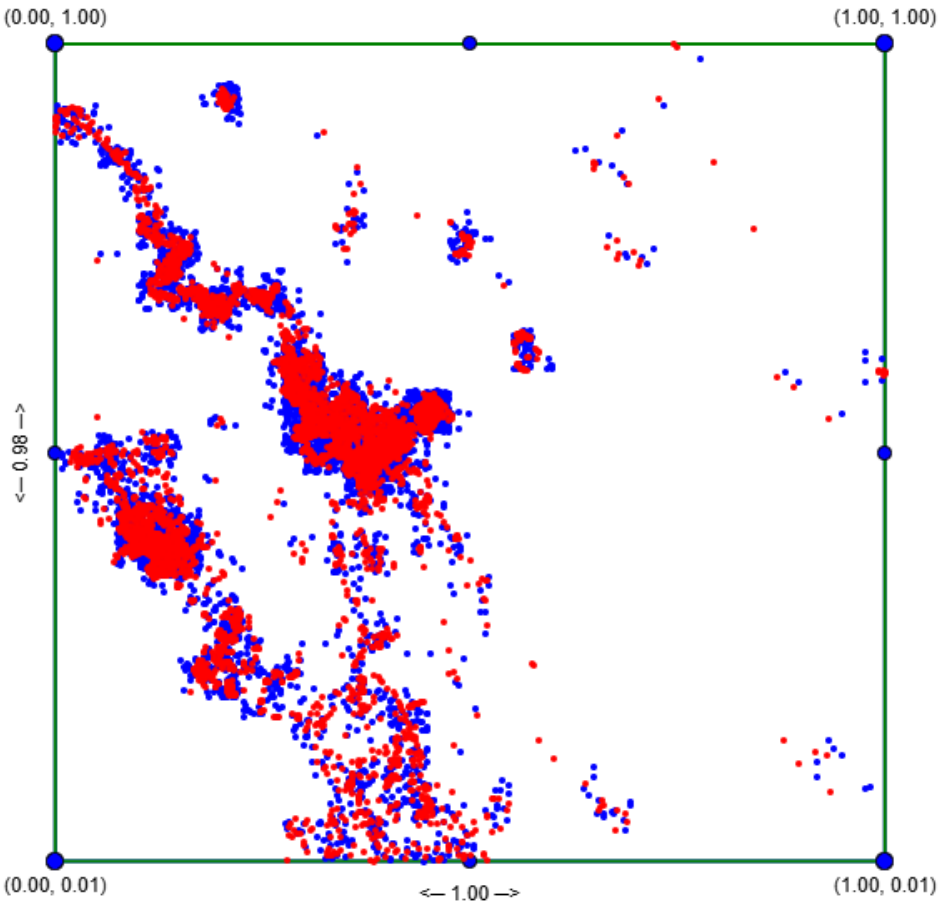

Figure 5.8. Comparison between real and generated spatial distributions for the city of Ioannina. The red points represent real building centroids obtained from OpenStreetMap data, while the blue points denote synthetic points generated through the data-driven empirical copula method. The generator successfully reproduces the clustered and elongated urban pattern observed in the real dataset.

In summary, across all six cases, the empirical generation approach succeeds in replicating the underlying statistical distribution of the original datasets. The generated points do not attempt to match absolute positions but instead preserve the probability density, symmetry, and spatial tendencies of the original patterns. This confirms that the generator fulfills its design objective: to produce new, randomly sampled data points that follow the same distributional behavior as the reference dataset.

To further assess the algorithm's performance, we evaluate the marginal distributions of the x and y coordinates using one-dimensional Wasserstein distance (also known as Earth Mover's Distance). **Figure 5.9** presents stacked histograms comparing the real (blue) and synthetic (red) distributions along each axis.

These visualizations offer valuable insights into the degree of fidelity achieved by the generator. In cases where the spatial structure is symmetric or approximately uniform, the

synthetic data tends to follow the original distribution closely. For instance, the peak alignment in both histograms suggests that the empirical sampling preserves the most frequent value ranges and overall distributional mass.

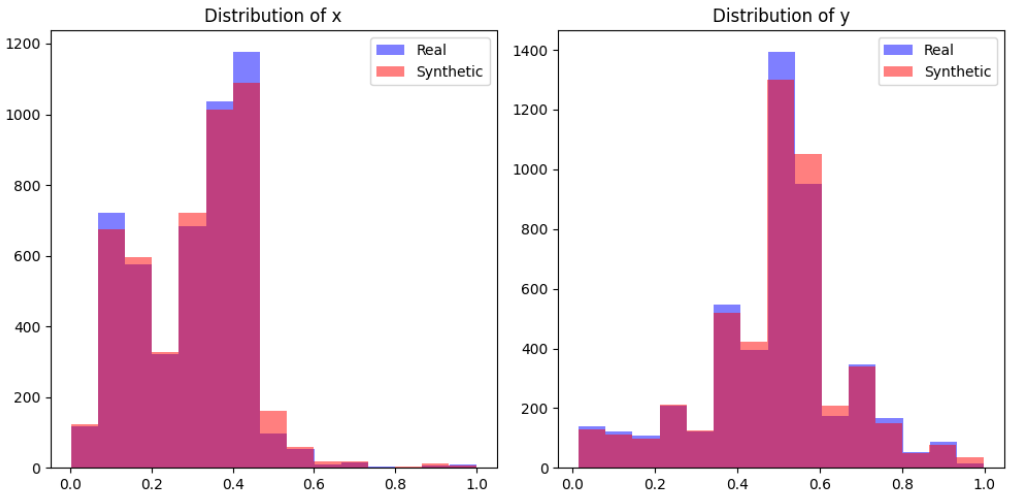

Figure 5.9. Marginal histograms of the x (left) and y (right) coordinates for real (blue) and synthetic (red) data. High overlap across bins confirms strong marginal distributional alignment between real and generated data.

However, this performance deteriorates in more asymmetric or structured input datasets, such as the spiral or crescent-shaped moons. In those cases, the ECDF transformation and histogram binning induce smoothing effects that result in central clustering or edge diffusion. In practical terms, this means that the algorithm oversamples mid-range intervals while underrepresenting rare or boundary values particularly in non-linear or multi-modal distributions.

This discrepancy highlights a key limitation: while the algorithm preserves marginal uniformity in an average sense, it lacks awareness of joint spatial dependencies that govern the structure of more complex patterns. As a result, the generated dataset may appear statistically similar in isolation but deviate significantly in spatial behavior when visualized or analyzed holistically.

To mitigate this, future versions of the system could incorporate density-adaptive binning or copula-enhanced joint modeling, ensuring that both marginals and dependencies are faithfully represented.

In addition**, Figure 5.10** illustrates the Voronoi tessellation derived from synthetic points generated to follow a clustered distribution. Each Voronoi polygon represents a spatial region that is closer to a particular seed point than to any other. As a result, this transformation

provides a structured polygonal view of the synthetic point cloud and serves as a valuable tool for examining the spatial footprint and neighborhood relationships of the generated data.

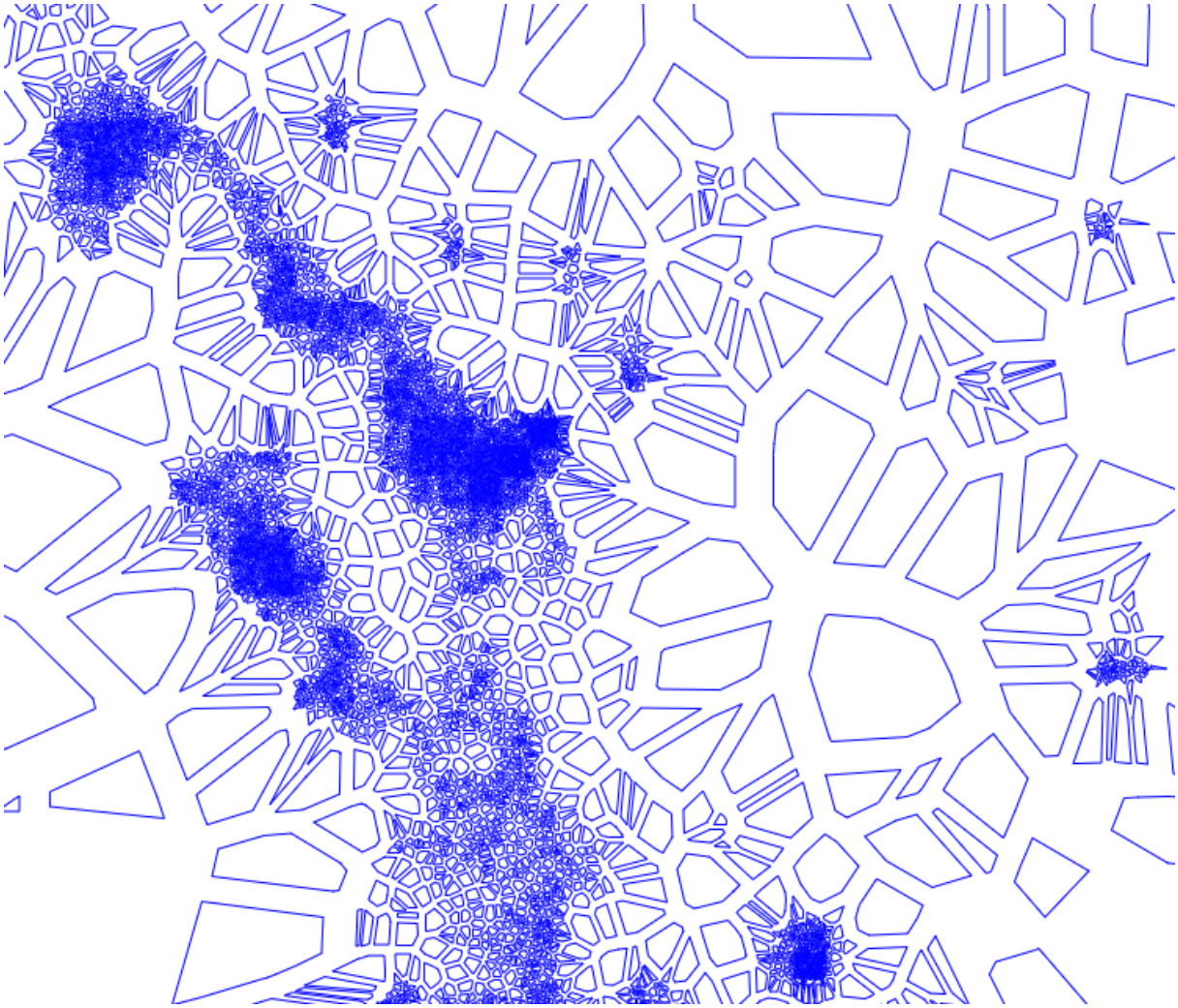

Figure 5.10. Voronoi tessellation is constructed from real spatial distributions for the city of Ioannina. Each polygon defines the region of influence of a single point, illustrating the spatial footprint and neighborhood relationships induced by the generated distribution.

5.2.2 Features Extraction Evaluation

The second component of the evaluation focuses on the system's ability to replicate polygonal shape characteristics through feature-based generation. Experiments were conducted across three representative categories: axis-aligned rectangles, convex polygons, and non-convex irregular polygons. For each shape type, geometric descriptors were extracted from both original and synthetic datasets and compared.

As shown in **Figure 5.11**, the axis-aligned rectangles generated by the system demonstrate a high level of structural fidelity. The synthetic shapes (blue) preserve edge parallelism, right-angle geometry, and size uniformity relative to the originals (red). The results confirm the

model's capability to enforce strict angular and linear constraints, making it suitable for structured spatial layouts such as land parcels, agricultural plots, or grid-based urban modeling.

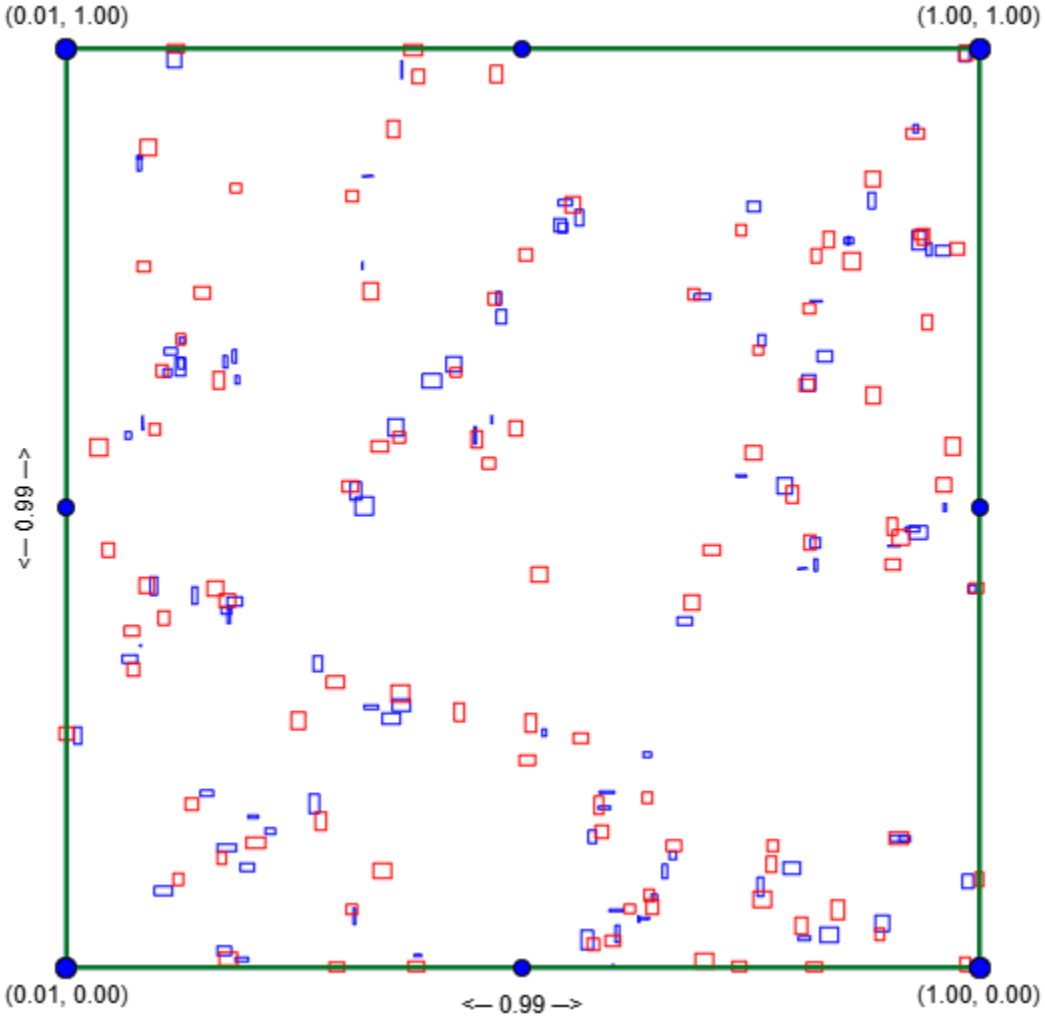

Figure 5.11. Axis-aligned rectangular shapes: comparison of original (red) and synthetic (blue) geometries. The generator maintains edge alignment, angular consistency, and proportional aspect ratios across the dataset.

For convex polygons, illustrated in **Figure 5.12**, the system introduces controlled irregularity while maintaining convexity. Vertex counts, compactness, and shape regularity metrics remain within acceptable bounds. The generated shapes follow the overall geometry of the originals without collapsing symmetry or structural consistency. This highlights the model's capacity to emulate naturally occurring convex forms, often found in geospatial footprints and environmental mapping.

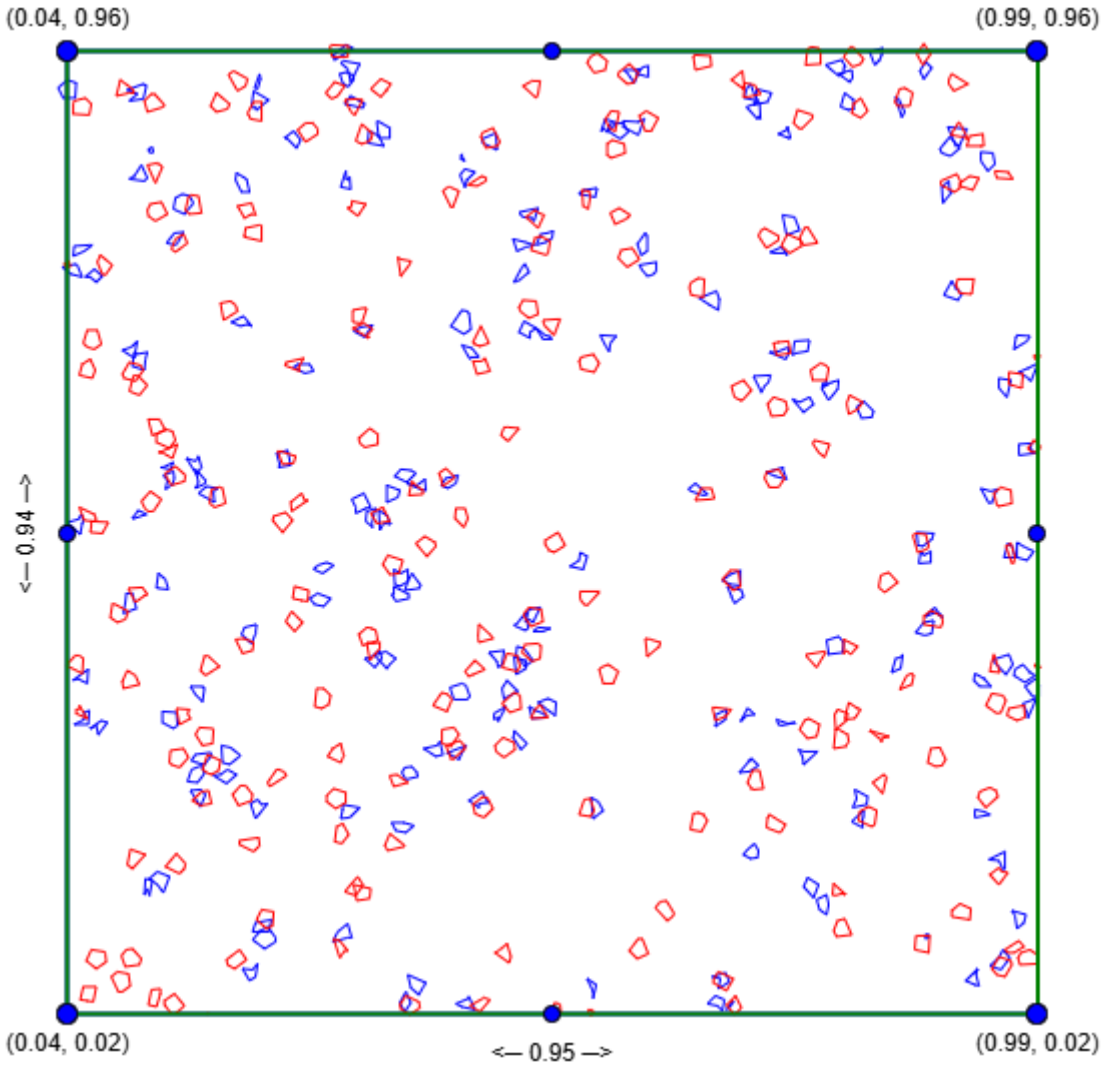

Figure 5.12. Convex polygons: visual comparison between original (red) and synthetic (blue) shapes. The synthetic output preserves vertex count, compactness, and convexity without collapsing the global structure.

Non-convex shapes, as depicted in **Figure 5.13**, present a more complex generative task. Nevertheless, the model captures spikiness, indentation, and irregular contours by leveraging learned shape descriptors. The output includes synthetic polygons with varying vertex count, radial asymmetry, and local deformations, aligned with the statistical structure of the input set. This indicates the generator's flexibility in replicating non-trivial morphological features with high variance.

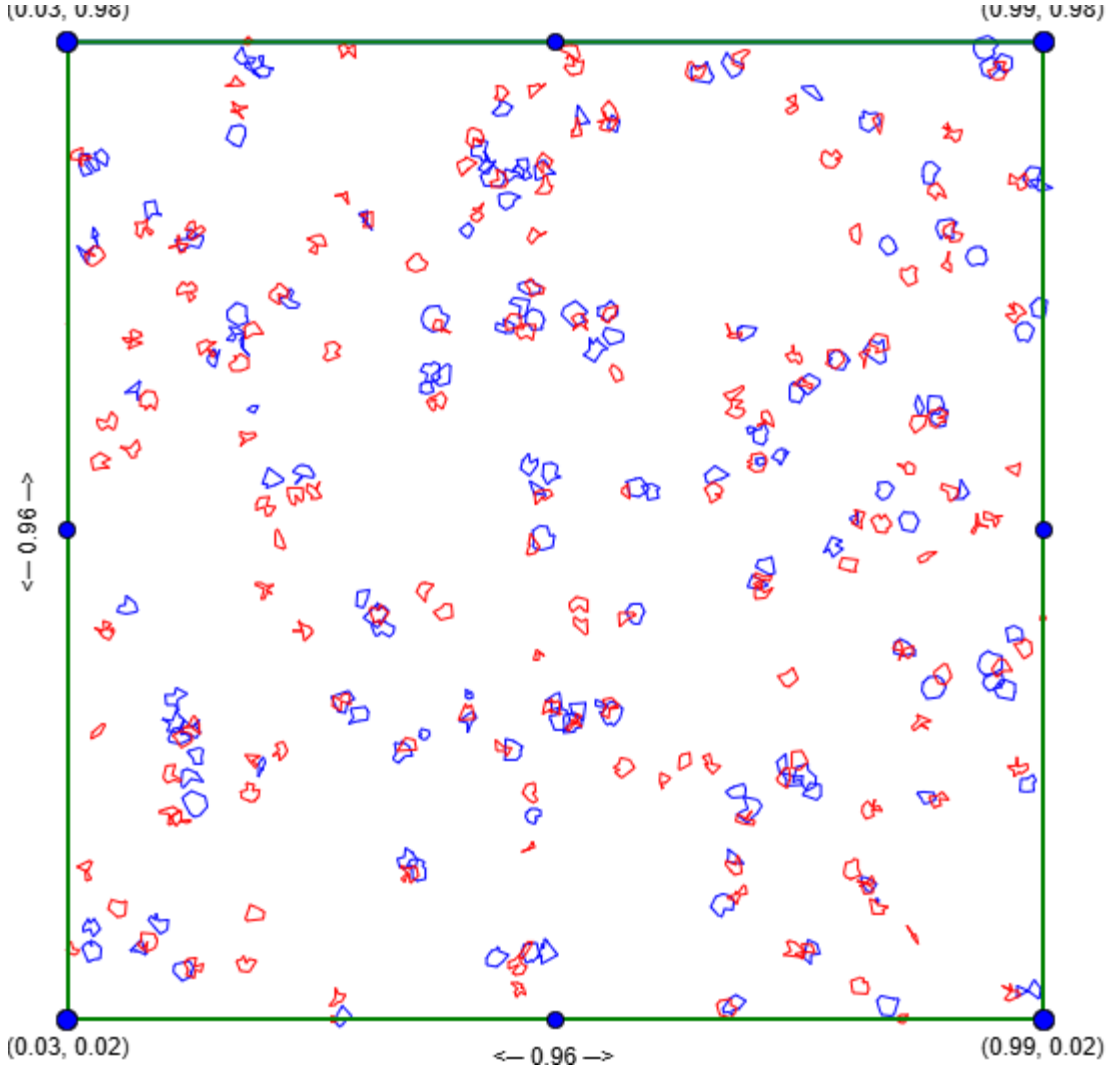

Figure 5.13. Non-convex irregular polygons: original (red) and generated (blue) shapes exhibit comparable spikiness, asymmetry, and variation in vertex distribution, capturing morphological complexity.

In addition to synthetic datasets, the feature-based generator was also evaluated using **real polygonal data**, specifically **building footprints from the city of Ioannina, Greece** (Figure 5.14). The red outlines correspond to real building geometries obtained from OpenStreetMap,

while the blue outlines depict the synthetic polygons generated by the system based on extracted geometric descriptors such as size, compactness, and convexity.

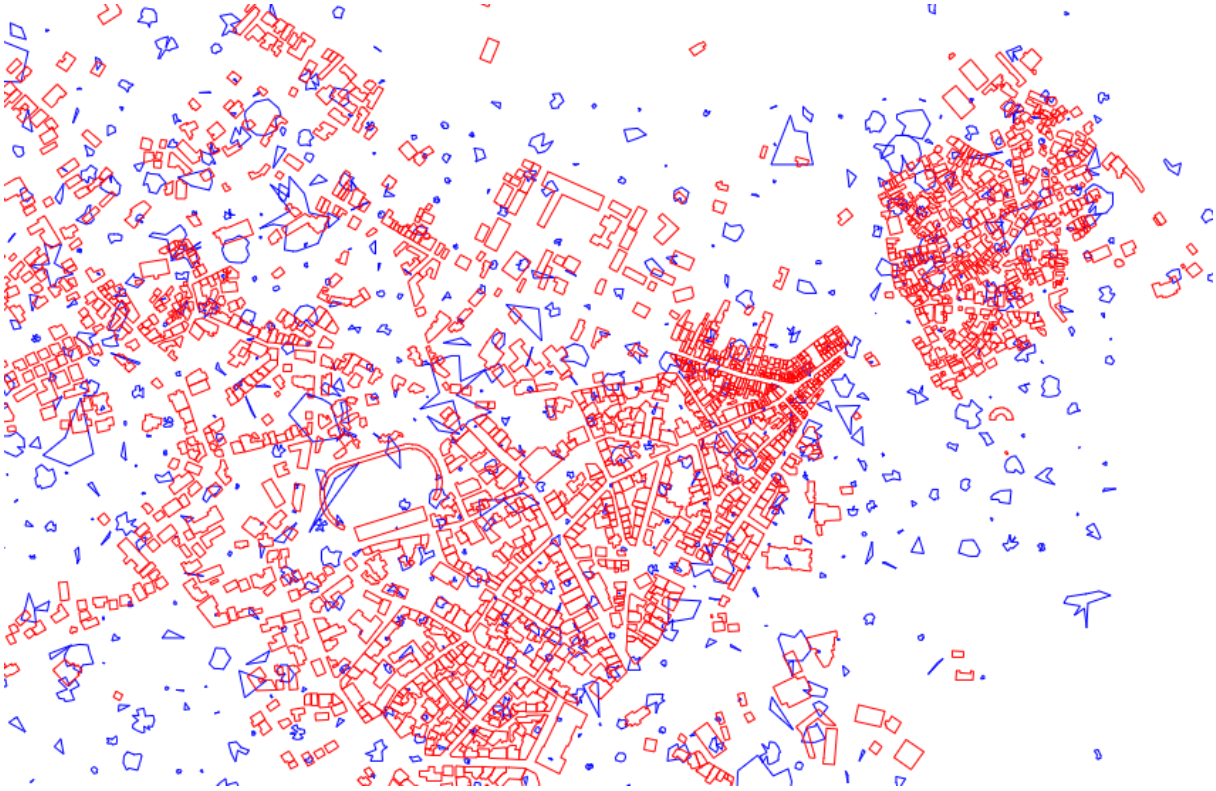

Figure 5.14. Comparison between real and generated polygonal data for the city of Ioannina. Red outlines represent real building footprints from OpenStreetMap, while blue outlines denote synthetic polygons generated from feature-based distribution matching. The generator captures the spatial density, orientation, and irregularity characteristic of the city's urban morphology.

Although the synthetic shapes do not replicate the exact building geometries, they successfully reproduce the statistical distribution of sizes, orientations, and spatial density observed in the real data. The generator effectively captures the heterogeneous and irregular urban morphology of Ioannina, demonstrating statistical and morphological fidelity rather than geometric duplication. Minor deviations in smaller or highly fragmented buildings result from descriptor aggregation and random sampling but do not compromise the overall representational accuracy.

To further quantify the quality of the shapes generated, we compared the empirical distributions of key geometric features between the real and synthetic datasets. **Figure 5.14**

shows histograms of size and number of vertices. The generated shapes approximate the distribution of size and structural complexity, achieving strong overlap in global descriptors.

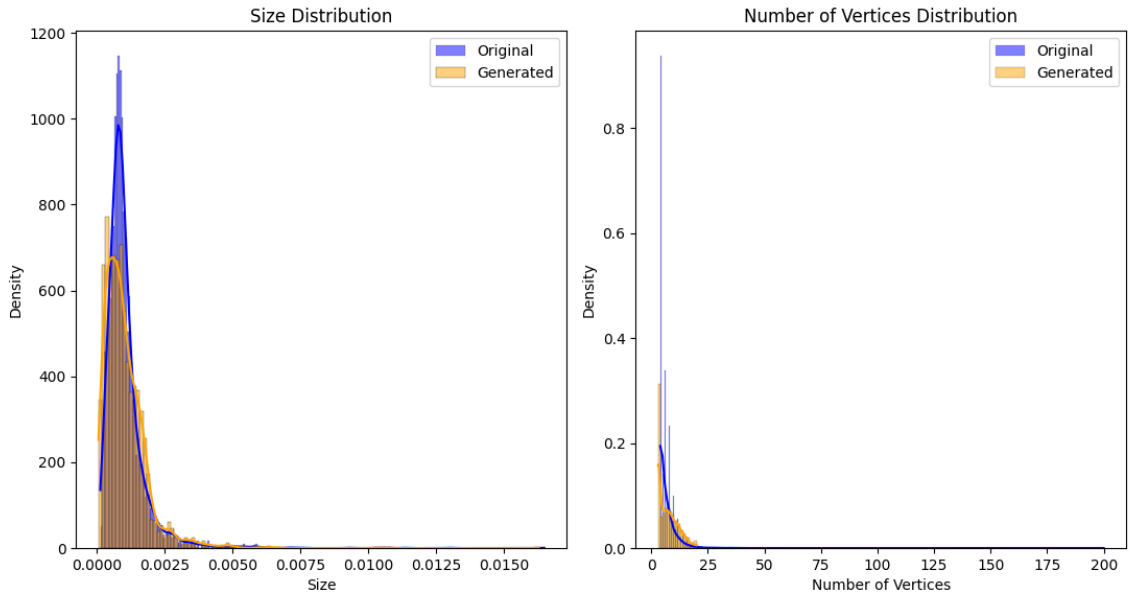

Figure 5.15. Feature distribution comparison between real and generated polygons. Left: distribution of polygon size. Right: distribution of number of vertices. Close alignment demonstrates fidelity of feature-based generative modeling.

As illustrated in **Figure 5.16**, the synthetic polygons align closely with the overall spatial pattern of the real buildings, confirming the system's robustness even in complex urban settings. Furthermore, when the dataset is restricted to specific and more isolated building categories, such as *schools, universities, hospitals,* and *parking areas*, the generator achieves an even closer geometric approximation. This improvement occurs because such structures are typically larger, more spatially distinct, and exhibit consistent geometric characteristics, allowing the feature-based synthesis process to capture their shape with higher precision.

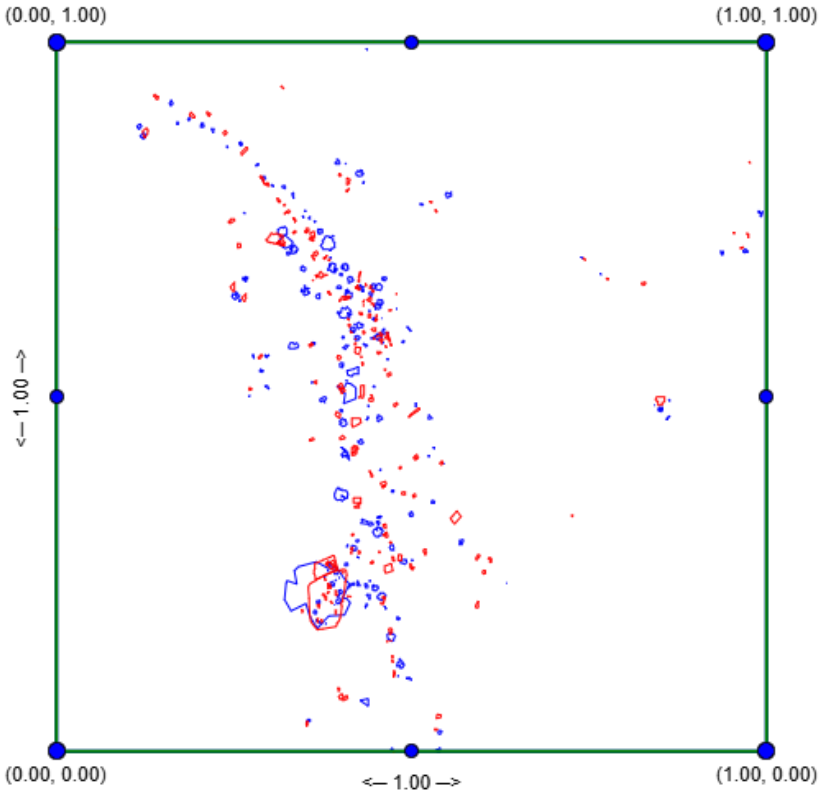

Figure 5.16. Comparison between real (red) and generated (blue) building polygons for the city of Ioannina. The system successfully reproduces the spatial density and geometric diversity of the real dataset. When focusing on isolated categories such as schools, universities, hospitals, and parking areas, the approximation improves further due to their higher regularity and reduced spatial clustering.

This evaluation demonstrates that the system learns and reproduces structural patterns from empirical data without relying on hardcoded rules. By modeling the underlying feature distributions, the generator adapts to different shape types while maintaining internal coherence and realism. These properties make the system appropriate for diverse downstream applications, including training data generation for computer vision models, simulation of built or natural environments, and synthetic benchmarking in spatial domains.

5.2.3 Similarity Evaluation

Across all synthetic polygon categories (rectangular, convex, non-convex), DGAS scores consistently exceeded 0.7, with distributional alignment scores averaging around 0.85 and structural alignment ranging from 0.6- 0.75, depending on shape complexity.

Unlike traditional similarity metrics based on spatial proximity (e.g., Hausdorff distance or Chamfer distance), DGAS intentionally omits absolute positional alignment. This decision reflects the core philosophy of the framework: the goal is not to reproduce exact spatial configurations, but rather to statistically mirror the geometric structure of the original data. This makes the method robust under translation, scaling, or changes in spatial extent conditions commonly encountered in practical scenarios involving generalization, data synthesis, or domain transfer. The DGAS thus serves not only as a robust evaluation tool but also as a guiding principle for the design of future generative models in spatial data science.

CHAPTER 6

CONCLUSION

This thesis introduced a comprehensive framework for the design, implementation, and evaluation of a web-based system for synthetic spatial data generation, specifically tailored to the controlled creation of polygonal geometries. The work was motivated by the increasing demand for reproducible, scalable, and statistically representative synthetic datasets in fields such as spatial analysis, machine learning, and simulation, where access to real-world data may be restricted or biased. At the core of the system lies a modular generation engine supporting both algorithmic and data-driven methods. Procedural generators were implemented to produce a diverse range of geometric patterns, including circle-based irregular polygons, Voronoi tessellations, elongated (flow-aligned) shapes, and more experimental forms based on Minkowski sums and sliding transformations. These generators allow users to configure parameters interactively, enabling the production of synthetic datasets with customized morphological characteristics.

In parallel, a nonparametric empirical distribution-matching module was developed to enable the synthesis of spatial data that aligns closely with the statistical properties of a user-provided reference dataset. Based on empirical cumulative distribution functions and copula theory, this method preserves both the marginal distributions and the dependency structure of input variables, supporting realistic and explainable data-driven synthesis. Feature-based generation was also introduced as a complementary approach, leveraging extracted descriptors such as size, irregularity, spikiness, compactness, convexity ratio, and aspect ratio to sample new geometries from kernel density estimations, thereby preserving the feature space complexity of the original data.

A key contribution of the system is the integration of these components within an interactive, browser-based interface built using OpenLayers and modular JavaScript. The platform supports real-time bounding box manipulation, dataset uploading, parameter tuning, and dynamic visualization of generated geometries in multiple map layers. Download options for CSV, WKT, and GeoJSON formats further enhance its applicability in downstream workflows. Additionally, a robust evaluation framework was developed to assess the fidelity and

scalability of the system. Quantitative assessments showed that circle-based irregular generators could efficiently scale up to 700,000 polygons, and that the empirical copula approach consistently achieved high resemblance to original datasets across synthetic distributions such as spirals, petals, moons, clustered patterns and real dataset. These results were validated using a newly introduced metric, the Distributional Geometry Alignment Score (DGAS), which measures distributional similarity based on both feature histograms and correlation matrices. The evaluation demonstrated that the system successfully achieves its central goal: generating statistically similar, but spatially distinct, synthetic datasets that retain the essential geometric and relational patterns of the originals.

In the case of real-world datasets with highly dense and contiguous polygonal structures such as central urban areas with tightly packed buildings, the Voronoi-based generation method demonstrated superior performance. This is because the Voronoi tessellation inherently adapts to local density variations, effectively partitioning space into distinct yet continuous regions. In contrast, the feature-based polygon generation approach tends to focus on capturing geometric characteristics (e.g., size, compactness, or convexity) rather than spatial adjacency, which can lead to overlapping or misaligned geometries in densely built environments.

Conversely, the data-driven empirical method performed particularly well when applied to more spatially separated and morphologically consistent building categories, where inter-feature correlations are easier to identify and preserve. In those cases, the reduced spatial density allowed the generator to better capture and reproduce the statistical dependencies among shape descriptors, resulting in higher fidelity of the synthesized polygons.

While the results are promising, several limitations remain. Procedural polygon generators in the current stack are reliable only up to 50 vertices, beyond that threshold we frequently observe self-intersections or geometric artifacts that invalidate the shape. On the data-driven side, the method underperforms footprints with very high vertex counts, where intricate boundaries are not well approximated by the learned feature distributions. More broadly, the empirical generation can introduce centralization bias when the reference distribution is sparse or strongly non-uniform, and some experimental generators are not yet exposed in the web interface due to runtime complexity. The system is also limited to 2D, static geometries, with no support for temporal evolution or volumetric (3D) structures.

In summary, this thesis delivers a complete, extensible solution for synthetic polygon generation, combining algorithmic versatility, data-driven accuracy, and user-oriented design in a

single platform. By offering explainable synthetic generation with statistical control and interactive visualization, the system opens new possibilities for testing, simulating, and augmenting spatial datasets in a transparent and reproducible manner. It provides a robust foundation for future developments in synthetic geometry generation and contributes a valuable tool to the geospatial and data science communities.

BIBLIOGRAPHY

[1] T. Vu, S. Migliorini, A. Eldawy, and A. Belussi, “Spatial Data Generators,” in *Spatial Gems, Volume 1*, 1st ed., New York, NY, USA: Association for Computing Machinery, 2022, pp. 13–24. [Online]. Available: https://doi.org/10.1145/3548732.3548736

[2] M. Elhefnawy, A. Ragab, and M. S. Ouali, “Polygon generation and video-to-video translation for time-series prediction,” *J Intell Manuf*, vol. 34, no. 1, pp. 261–279, Jan. 2023, doi: 10.1007/s10845-022-02003-1.

[3] Y. Liang, B. Nobakht, and G. Lindsay, “The application of synthetic data generation and data-driven modelling in the development of a fraud detection system for fuel bunkering,” in *Measurement: Sensors*, Elsevier Ltd, Dec. 2021. doi: 10.1016/j.measen.2021.100225.

[4] J. Liu *et al.*, “PolyFormer: Referring Image Segmentation as Sequential Polygon Generation.” [Online]. Available: https://polyformer.github.io/

[5] B. Wang, X. Song, C. Weng, X. Yan, and Z. Zhang, “A Hybrid Method Combining Voronoi Diagrams and the Random Walk Algorithm for Generating the Mesostructure of Concrete,” *Materials*, vol. 17, no. 18, p. 4440, Sep. 2024, doi: 10.3390/ma17184440.

[6] G. Eder, M. Held, and P. Palfrader, “Implementing straight skeletons with exact arithmetic: Challenges and experiences,” *Comput Geom*, vol. 96, Jun. 2021, doi: 10.1016/j.comgeo.2021.101760.

[7] G. H. Hughes, “The Edge Geometry of Regular N-gons (Part I for N ≤ 25).”

[8] P. Katiyar, T. Vu, A. Eldawy, S. Migliorini, and A. Belussi, “SpiderWeb: A Spatial Data Generator on the Web,” in *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, Association for Computing Machinery, Nov. 2020, pp. 465–468. doi: 10.1145/3397536.3422351.

[9] P. Gorry and P. Mooney, “A software tool for generating synthetic spatial data for GIS-classroom usage,” *AGILE: GIScience Series*, vol. 5, pp. 1–7, May 2024, doi: 10.5194/agile-giss-5-26-2024.

[10] Y. Kang and E. J. Kubatko, “An automatic mesh generator for coupled 1D-2D hydrodynamic models,” *Geosci Model Dev*, vol. 17, no. 4, pp. 1603–1625, Feb. 2024, doi: 10.5194/gmd-17-1603-2024.

[11] D. Ankur, D. Mrunalini Thamankar, N. D. Rai, R. G. Veeresh, and P. Auradkar, “Synthetic Generation of Spatial Polygons On Cloud,” in *2024 3rd International Conference for Innovation in Technology, INOCON 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/INOCON60754.2024.10511892.

[12] E. Lewandowicz and P. Flisek, “A method for generating the centerline of an elongated polygon on the example of a watercourse,” *ISPRS Int J Geoinf*, vol. 9, no. 5, May 2020, doi: 10.3390/ijgi9050304.

[13] K. Hermes and M. Poulsen, “A review of current methods to generate synthetic spatial microdata using reweighting and future directions,” *Comput Environ Urban Syst*, vol. 36, no. 4, pp. 281–290, 2012, doi: https://doi.org/10.1016/j.compenvurbsys.2012.03.005.

[14] Q. Luo and Y. Rao, “Improved Sliding Algorithm for Generating No-Fit Polygon in the 2D Irregular Packing Problem,” *Mathematics*, vol. 10, no. 16, Aug. 2022, doi: 10.3390/math10162941.

[15] *2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*. IEEE, 2020.

[16] J. P. Restrepo, J. C. Rivera, H. Laniado, P. Osorio, and O. A. Becerra, “Nonparametric Generation of Synthetic Data Using Copulas,” *Electronics (Switzerland)*, vol. 12, no. 7, Apr. 2023, doi: 10.3390/electronics12071601.

[17] F. Benali, D. Bodénès, N. Labroche, and C. de Runz, “Synthetic Complex Data Generation Using Copula. 23rd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP),” 2021. [Online]. Available: https://hal.science/hal-03188317v1

[18] A. De Araujo, J. M. Do Valle, and N. Cacho, “Geographic feature engineering with points-of-interest from openstreetmap,” in *IC3K 2020 - Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, SciTePress, 2020, pp. 116–123. doi: 10.5220/0010155101160123.

[19] F. Li, “Feature Selection Based on Wasserstein Distance,” Nov. 2024, [Online]. Available: http://arxiv.org/abs/2411.07217

[20] Z. Zhang, J. Ge, Z. Wei, C. Zhou, and Y. Wang, “Feature Selection Based on Orthogonal Constraints and Polygon Area.”

[21] G. Farkas, “Possibilities of using raster data in client-side web maps,” *Transactions in GIS*, vol. 24, no. 1, pp. 72–84, Feb. 2020, doi: 10.1111/tgis.12588.

APPENDIX A

IMPLEMENTATION SAMPLES

A. 1 Overview

This appendix provides implementation screenshots that complement the synthetic polygon generation framework described in Chapters 3–5. It serves to illustrate the core components of the system, validate its output, and demonstrate its flexibility across different generation modes and evaluation stages.

A.2 Generation UI - Screenshot

Below are sample screenshots from the web-based generation interface Below are sample screenshots from the web-based generation interface

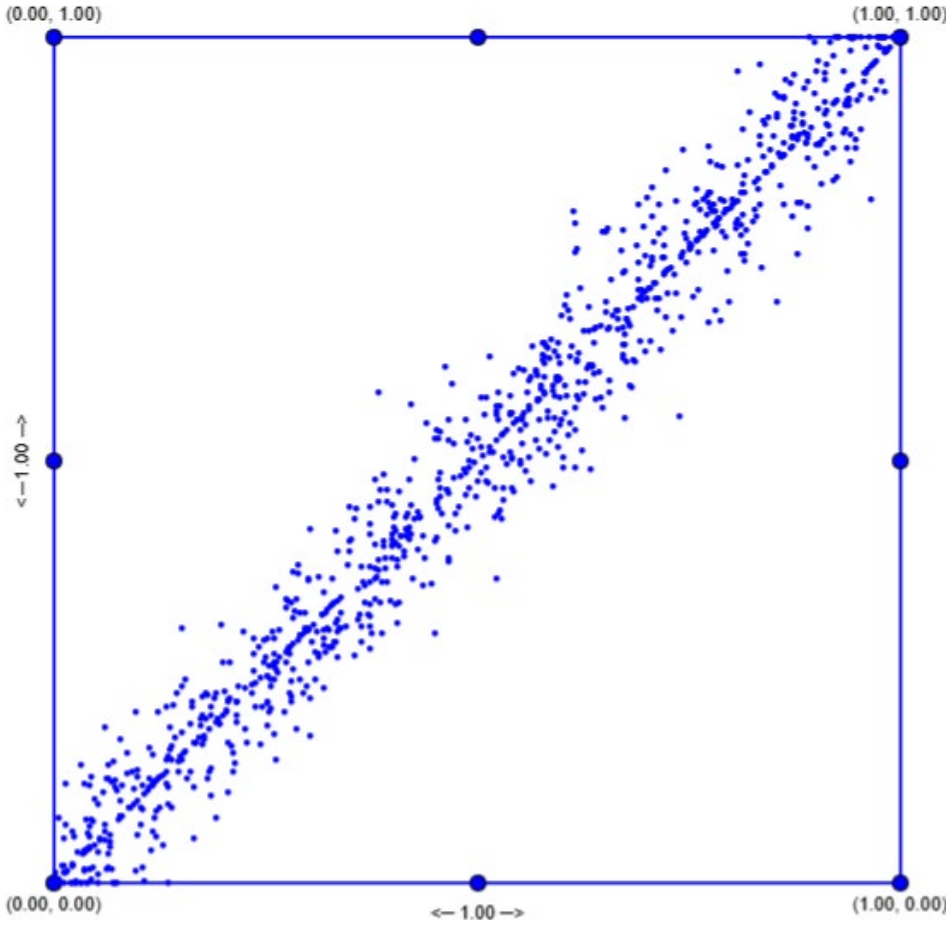

Figure A.1: Synthetic point generation using diagonal distribution.

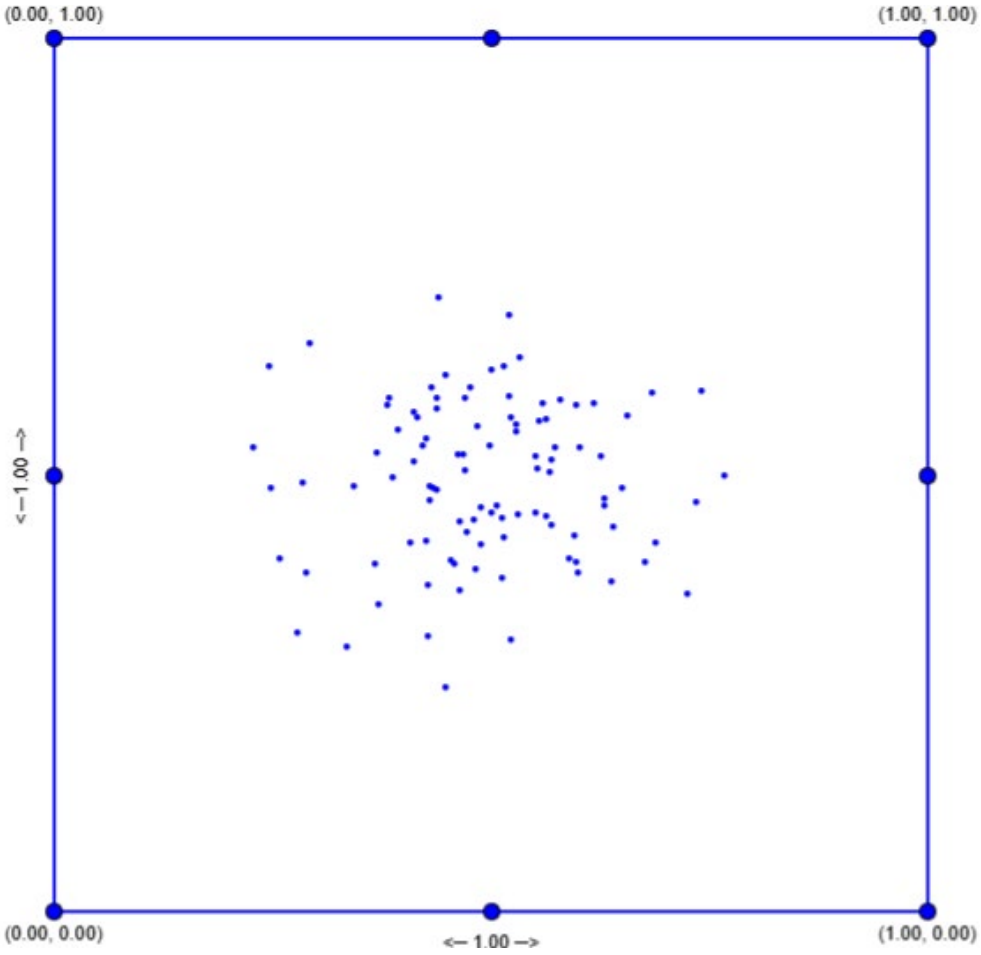

Figure A.2: Synthetic point generation using Gaussian distribution.

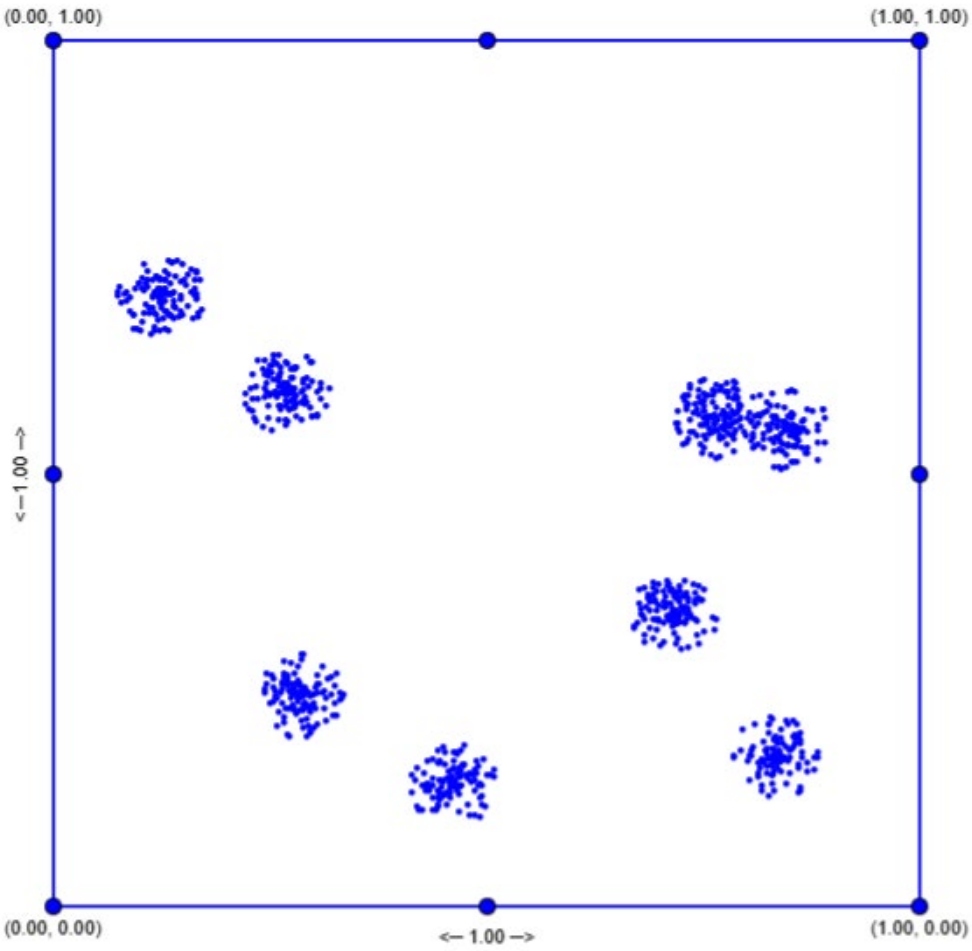

Figure A.3: Synthetic point generation using diagonal distribution.

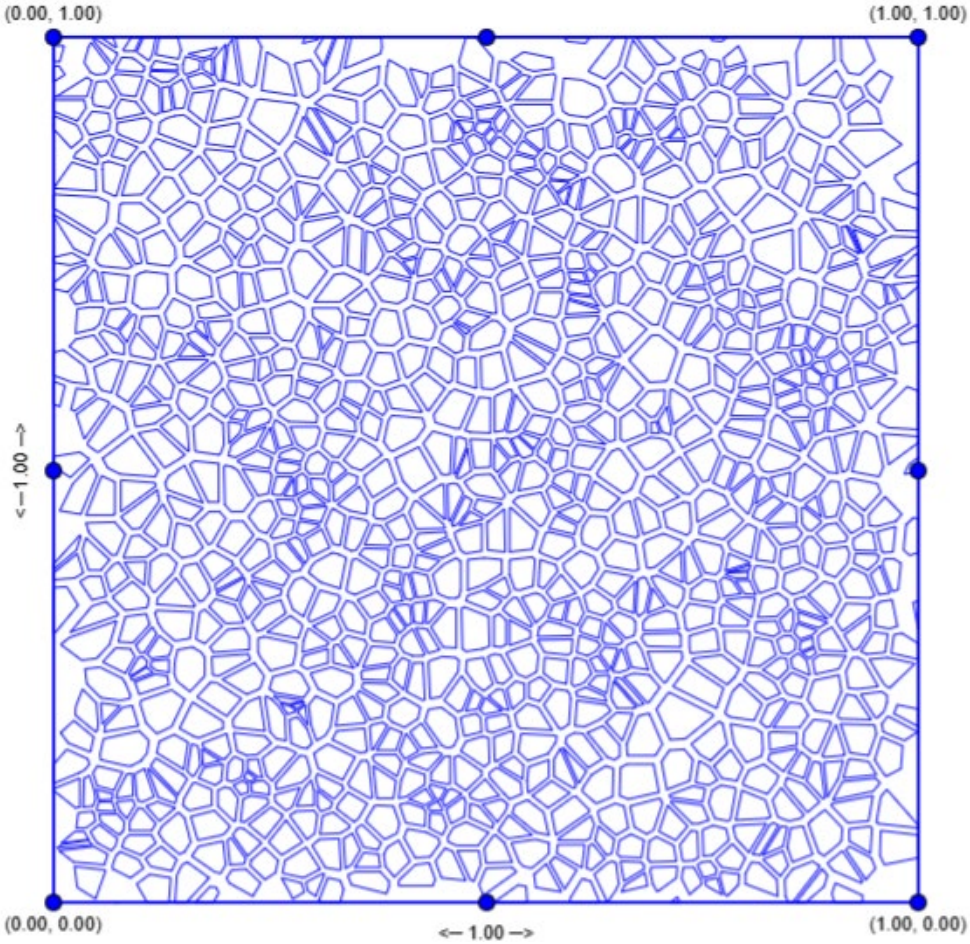

Figure A.4: Generate polygons with Voronoi shapes.

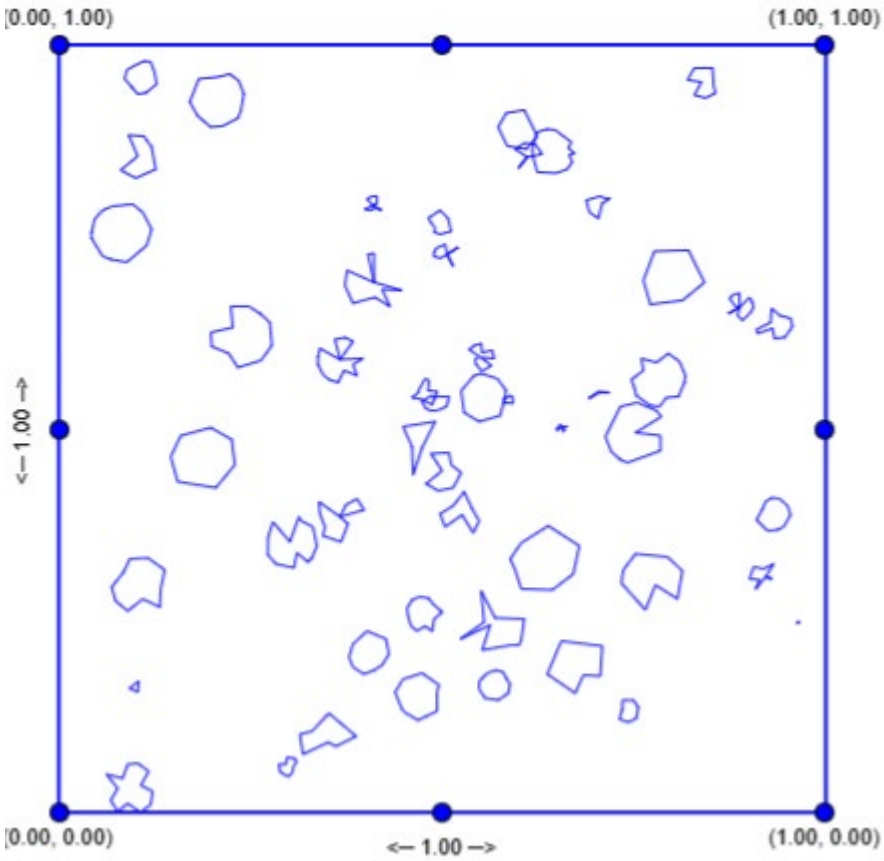

Figure A.5: Generate polygons with Mix-Type mode.

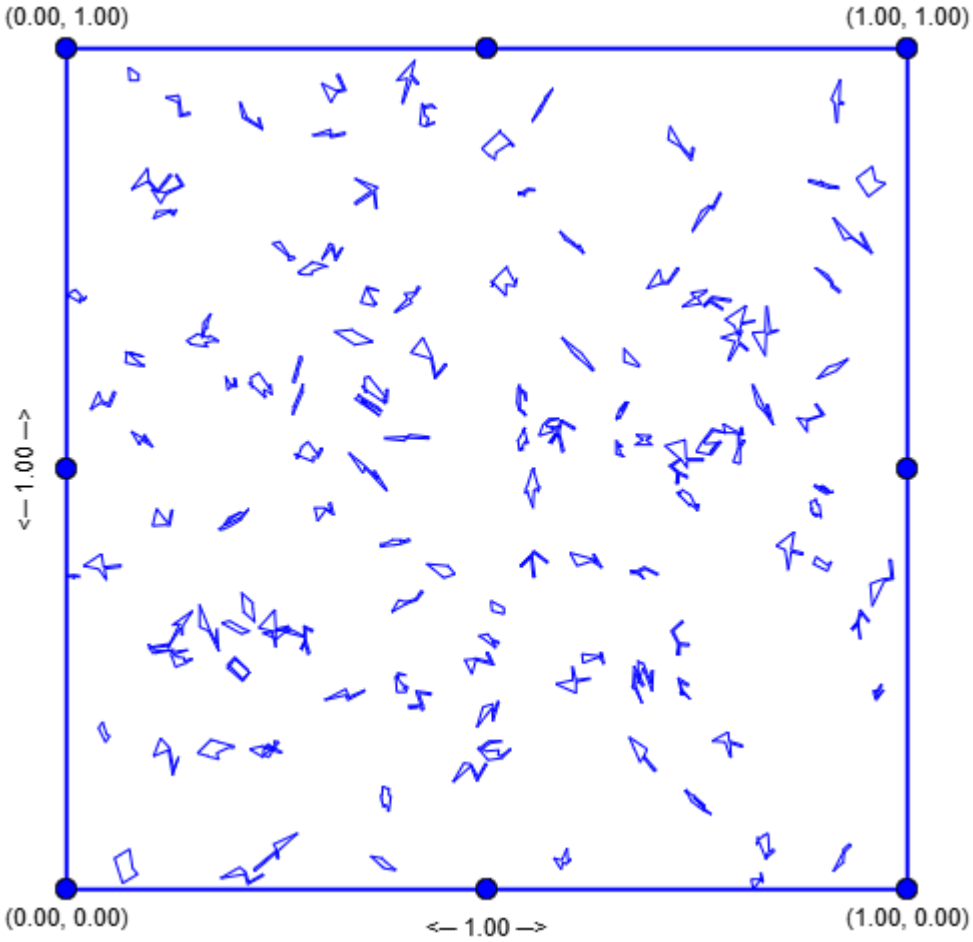

Figure A.6: Generate polygons with Elongated shapes.

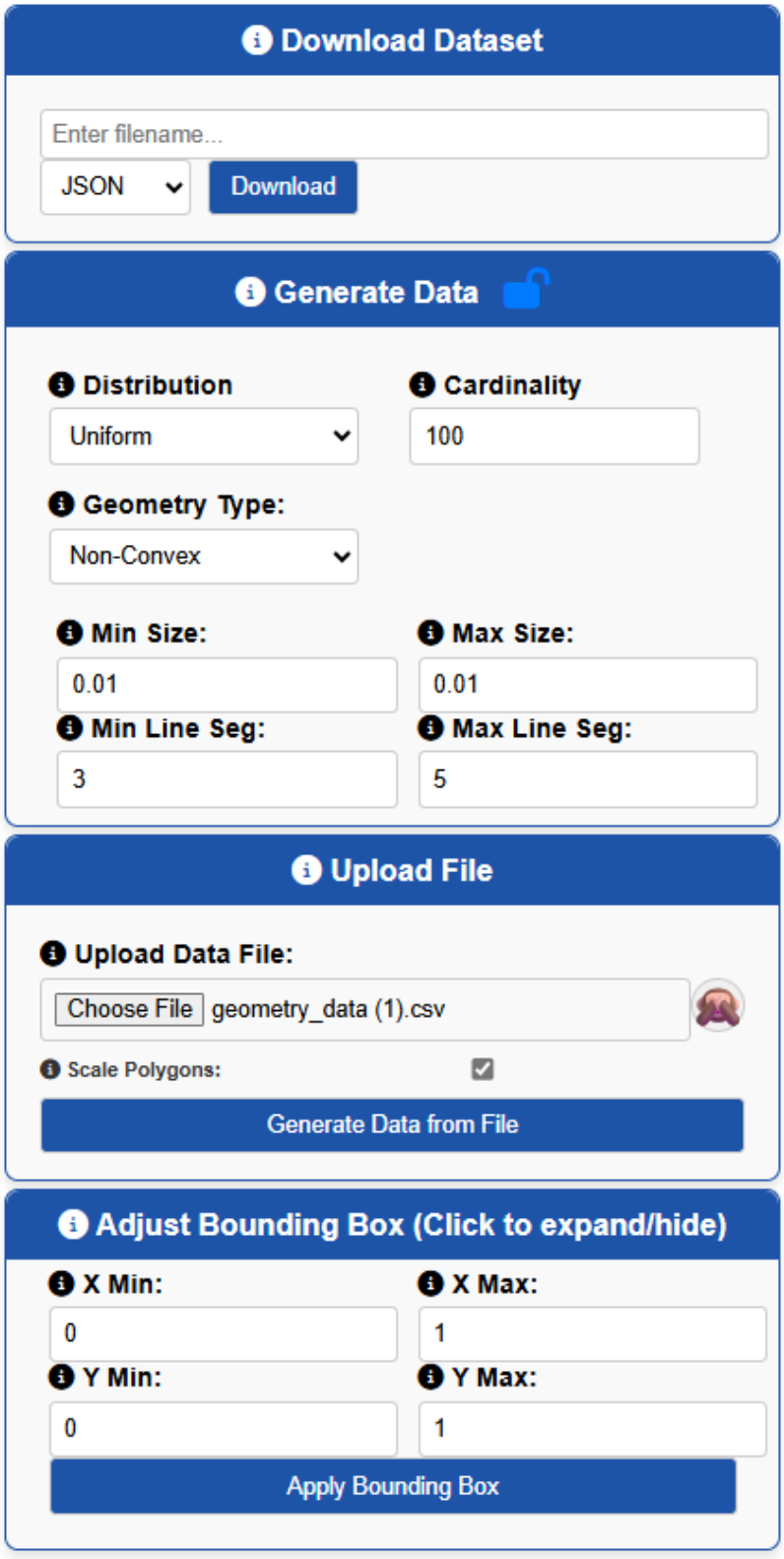

Figure A.7: User Interface for Synthetic Geometry Generation and Dataset Management.

A.3 Example of Feature Extraction Output

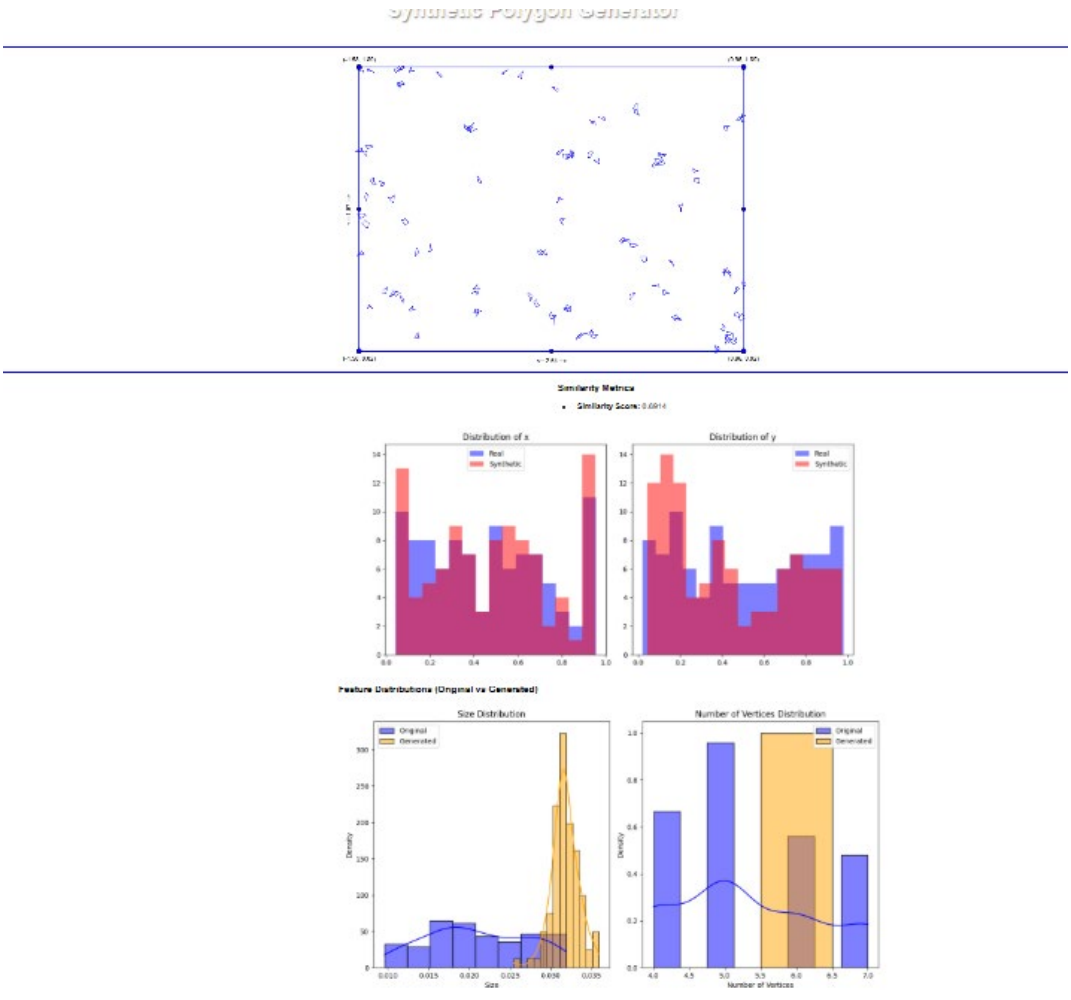

Figure A.1: Evaluation of Synthetic Geometry Generation Output.

Averages

area	0.001550
aspect_ratio	1.014492
centroid_x	0.483469
centroid_y	0.435406
compactness	0.746771
convexity	0.490000
equilateral	0
irregularity	0.425817
num_vertices	6
perimeter	0.160726
size	0.031748

Figure A.2 Average Feature Values of the Generated Polygons.

SHORT BIOGRAPHY

Vasileios Tsolis was born in Ioannina in 1996. He received his undergraduate degree in Informatics from the Ionian University, where he completed his thesis on the recognition of diseases through data extracted from social media platforms.

He is currently pursuing a master's degree in data science and engineering at the University of Ioannina. His academic interests include data management and machine learning. As part of his graduate studies, he has focused on developing tools and methodologies for generating and evaluating polygonal geometries in data-driven environments.

He is proficient in Python and has experience with scientific computing libraries such as NumPy, Pandas, Matplotlib, Scikit-learn, PyTorch and TensorFlow. He has works on Java and Kotlin for software development, as well as SQL for data querying and relational database management. During his academic and personal projects, he has also worked with Javascript, HTML5, Node and he is familiar with Git for version control and collaborative development workflows.

He is passionate about bridging theory and application in the field of data science, and aims to contribute to innovative, reproducible, and impactful research in the broader domain of intelligent systems and applied machine learning.