Random Affine Transformation Feature Representation Learning for Fast Polygon Retrieval

Zhangyu Wang
University of California Santa Barbara
United States
zhangyuwang@ucsb.edu

Xingyi Du
Tencent
United States
xingyi.du@wustl.edu

Hao Li
Technical University of Munich
Germany
hao_bgd.li@tum.de

Martin Werner
Technical University of Munich
Germany
martin.werner@tum.de

ABSTRACT

Despite the advance of representation learning (RL) of image and text data, it is a challenging task to obtain a general-purpose representation of vector-based spatial data (e.g., point, polyline, and polygon) that fulfills domain-specific prerequisites in downstream spatial analysis. Towards filling this gap, we put focus on the first step of revisiting a classic computational geometry task, namely the polygon retrieval, with a modern RL approach. In this paper, we propose a novel representation learning method for polygon data, namely *Random Affine Transformation Features (RATFs)*. Inspired by Random Fourier Features, RATFs represent polygons on the plane into vector embeddings in the high-dimensional space using the basis of random anchor polygons, allowing us to transform complex, non-batch-wise computations between polygons such as Hausdorff distance into simple, batch-wise inner products. Experiments demonstrate that by using a naive 3-layer feed-forward network, our RATFs-based neural polygon embeddings are able to retrieve similar polygons from the corpus 100,000 times faster than computing Hausdorff distances at inferencing and achieve good retrieval quality. We aim to demonstrate the representational power of RATFs and leave the design and tuning of more advanced RATFs-based neural networks for future work.

CCS CONCEPTS

• **Information systems** → **Geographic information systems**; • **Computing methdologies** → **Artificial intelligence**.

KEYWORDS

GeoAI, Representation Learning, Implicit Neural Representation, Polygon Encoding, Polygon Retrieval

ACM Reference Format:
Zhangyu Wang, Xingyi Du, Hao Li, and Martin Werner. 2024. Random Affine Transformation Feature Representation Learning for Fast Polygon Retrieval. In *3rd ACM SIGSPATIAL International Workshop on Searching and Mining Large Collections of Geospatial Data (GeoSearch'24), October 29-November 1, 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3681769.3698582

GeoSearch'24, October 29-November 1, 2024, Atlanta, GA, USA

ACM ISBN 979-8-4007-1148-0/24/10.
https://doi.org/10.1145/3681769.3698582

1 INTRODUCTION

It is an ultimate goal for almost all learning methods and systems to represent massive raw data with a standard and unified format, with which one can largely facilitate the effectiveness and efficiency of downstream tasks. Researchers use different terminologies for this process, starting with feature engineering or feature extraction, now more often called representation learning (RL). Recently, RL has emerged [3], as a new paradigm of feature extraction and become the foundation for performance boosting in solving various AI tasks such as natural language processing (NLP), computer vision (CV), and speech recognition. In most cases of representation learning, text[1, 10], images [5], and audio [8] data are directly fed into dedicated neural network architecture modules to automatically learn low-level and high-level representations without the need for a manual feature engineering step.

However, few works develop such RL techniques for various types of vector-based spatial data (e.g., points, polylines, polygons, trajectories, triangulated irregular networks (TINs), 3D LiDAR point clouds, etc.) without rasterization. The main reasons are twofold: first, it can be a hotfix for non-spatial people to rely on rasterization, then treat everything as pixels so that everyone's life is easier. This is fully understandable as most people don't need the precision that comes with the raw vector spatial data then it does not matter; second, it is simply a challenging task to obtain a general-purpose nice representation of vector data that fulfills domain-specific prerequisite in sophisticated spatial analysis (e.g., geometry correctness, affine equivariance, and shape sensitivity). Different to text and image, spatial representation learning (SRL) demonstrates several unique challenges: 1) the geographic scale problem usually requires a representation learning model to continuously consider multiple scales ranging from global scale, continental scale, city scale, and neighborhood scale, etc; 2) though spatial data are usually stored in a discretized format, they represent continuous objects or geometries. This requires representation learning models to learn continuous vector data representations instead of treating them as a finite list of coordinates [13]. For example, a polygon is usually represented as a ring of vertices or several rings (polygons

with holes or multipolygons), but it actually refers to a continuous surface.

There are mainly three types of vector data, namely point, polyline, and polygon. For point data, classic point pattern analysis is widely used almost everywhere to obtain statistic insights of spatial data distribution. For the other two types (polyline and polygon), they are basically the same thing. The only difference is that a polyline does not need to be closed while a continuous surface is bounded by a closed ring. Therefore, it is key in SRL to find a robust and general-purpose representation of polyline and polygon vector data, from which all kinds of downstream statistic analysis can benefit. To narrow down the scope, we put our focus on polygons, especially the polygon retrieval task. In this context, our aim is to offer a modern solution to a classic computational geometry task (i.e., to calculate the similarity of two arbitrary polygons, similar to [7]) with a dedicated SRL approach. We see this as a first step toward a general-purpose representation of vector spatial data.

In this paper, we aim to fill this research gap by developing a novel random affine transformation feature representation learning approach for fast and efficient polygon vector data retrieval. The idea is to take the first step forward and show the potential benefit of a dedicated and native representation learning approach for vector-based spatial data. Specifically, we focus on learning a neural polygon embedding model which transforms variable-length vector polygon data (i.e., sequence of vertex coordinates) into fixed-length continuous representations. Moreover, we demonstrate that the simple inner product of such neural polygon embeddings preserves the Hausdorff distance between polygons, enabling fast and accurate polygon retrieval on large scales.

2 PRELIMINARIES

2.1 Hausdorff Distance

The Hausdorff distance is a common measure used to quantify the similarity between two geometric shapes. Given a metric space (M, d), the Hausdorff distance between non-empty subsets $X, Y \subset M$ is defined as:

$$d_H(X,Y) = \max\left\{\sup_{x\in X} d(x,Y), \sup_{y\in Y} d(y,X)\right\}$$

where $d(a,B) = \inf_{b\in B} d(a,b)$ denotes the distance from a point a to a subset $B \subset M$.

In practice, the Hausdorff distance between two shapes is often approximated by sampling discrete points from each set. Let $P = \{p_i\}_{i=1}^{N_p} \subset X$ and $Q = \{q_j\}_{j=1}^{N_q} \subset Y$ be point samples from X and Y, respectively. The approximate Hausdorff distance between the two shapes is then computed as:

$$d_H(X,Y) \approx d_H(P,Q) = \max\left\{\max_i \min_j d(p_i,q_j), \max_j \min_i d(q_j,p_i)\right\}$$

This discrete approximation of the Hausdorff distance is widely used in computational geometry for shape comparison and analysis.

2.2 Shape Registration

Shape registration seeks to find an appropriate transformation that best aligns one shape with a reference shape. This process is crucial for bringing two shapes into the same reference frame or for quantifying the differences between the same shape observed under different conditions. Shape registration has broad applications in areas such as computer vision, computer graphics, and medical imaging.

For geometric shapes in Euclidean spaces, the transformations considered are often rigid or affine. A rigid transformation, which includes rotation and translation, preserves Euclidean distances and angles. An affine transformation, being more general, further includes scaling and shearing. It preserves lines and parallelism but not necessarily distances and angles.

In an n-dimensional Euclidean space, an affine transformation can be represented by a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^n$. Given a point set $P = \{p_i\}_{i=1}^{N_p} \subset \mathbb{R}^n$ and a reference point set $Q = \{q_j\}_{j=1}^{N_q} \subset \mathbb{R}^n$, the goal of affine registration is to find the matrix $\boldsymbol{A}$ and vector $\boldsymbol{t}$ that best align P with Q. This is typically formulated as the following optimization problem:

$$\min_{\boldsymbol{A},\boldsymbol{t},\{c_i\}} \sum_{i=1}^{N_p} \left\|\boldsymbol{A}p_i + \boldsymbol{t} - q_{c_i}\right\|_2^2,$$

where c_i denotes the index of the corresponding point in Q for each point p_i in P. For a comprehensive review of the state-of-the-art techniques in point set registration, we refer the reader to a survey [14].

In general, shape registration in Euclidean space is solved by first sampling points on the surfaces of the shapes, then tackling the point set registration problem to compute the optimal transformation.

2.3 Iterative Closest Point (ICP) method

An important and widely used algorithm for point set registration is the Iterative Closest Point (ICP) method [4]. ICP solves the rigid registration problem by iteratively refining the transformation between two point sets. The method alternates between two main steps:

(1) **Correspondence Step:** For each point p_i in the point set P, find the closest point q_{c_i} in the reference point set Q after applying the current estimate of the transformation. This is formulated as:

$$c_i = \arg\min_j \left\|\boldsymbol{R}_k p_i + \boldsymbol{t}_k - q_j\right\|_2,$$

where $\boldsymbol{R}_k$ and $\boldsymbol{t}_k$ are the rotation matrix and translation vector at iteration k.

(2) **Transformation Update Step:** Update the rotation matrix and translation vector by minimizing the sum of squared distances between the transformed points and their correspondences:

$$\boldsymbol{R}_{k+1}, \boldsymbol{t}_{k+1} = \arg\min_{\boldsymbol{R},\boldsymbol{t}} \sum_{i=1}^{N_p} \left\|\boldsymbol{R}p_i + \boldsymbol{t} - q_{c_i}\right\|_2^2.$$

This iterative process continues until convergence, resulting in the optimal rigid transformation aligning P with Q. The ICP method can be extended to handle affine transformations [6, 17], allowing for scaling and shearing in addition to rotation and translation.

3 METHOD

The goal of this research is to efficiently evaluate the Hausdorff distance in the embedding space. Specifically, we wish to find an encoder *Enc* which embeds a given polygon X into a d-dimensional real-valued embedding $e_X = Enc(X) = (e_1, e_2, \cdots e_d)$, and a metric d_e in the embedding space such that

$$d_H(X, Y) \approx d_e(e_X, e_Y)$$

We set d_e to be the cosine distance, i.e.,

$$d_e = 1 - \frac{\langle e_X, e_Y \rangle}{|e_X||e_Y|}$$

because cosine distance is one of the most commonly used metrics of embeddings and there are many industry-level optimization for computing cosine distance in extremely large scales.

Like location encoding [12, 15], we adopt a two-stage encoder architecture: in the first stage, the polygon X is encoded by a non-parametric algorithm PE into a **polygon encoding**, analogous to the feature engineering stage in computer vision; in the second stage, a simple neural network NN is used to map the polygon encoding non-linearly into a **neural polygon embedding**. Formally,

$$e_X = Enc(X) = NN(PE(X))$$

The usefulness of *Enc* relies heavily on *PE*. In this paper, we propose a novel polygon encoding algorithm based on *Random Affine Transformation Features* (RATF). The learned embedding $NN(PE(X))$ is a neural implicit representation of polygons [16].

3.1 Random Affine Transformation Features (RATFs) for Polygon Encoding

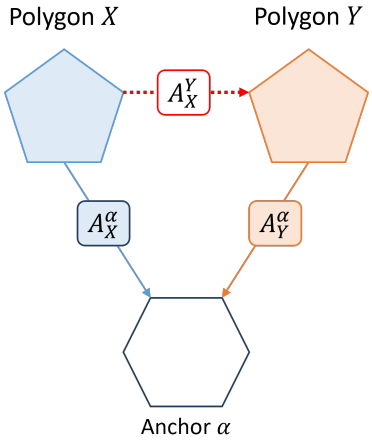

Figure 1: Illustration of polygon anchoring by affine transformation. Suppose X, Y, α are pairwise affine transformable, and the affine transformation matrix from polygon Z to polygon Z' is $A_Z^{Z'}$. Then A_X^Y is identity if and only if $A_X^\alpha = A_Y^\alpha$, i.e, the equivalence relation between polygons is preserved by this anchoring process.

The motivation behind Random Affine Transformation Features is intuitive. Given two polygons X and Y, we are interested in computing their similarity. Instead of direct comparison, we can compare their relative differences to an anchor polygon α – i.e., compare whether the two polygons are identically similar to α.

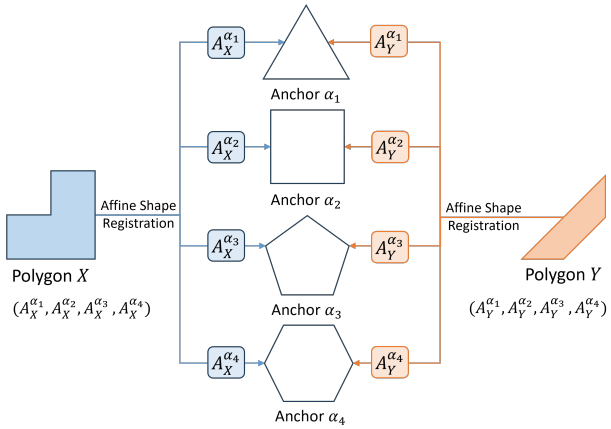

Figure 2: Illustration of Random Affine Transformation Features (RATFs).

Specifically, let the affine transformation from X to Y be $\boldsymbol{A}_X^Y$, from X to α be $\boldsymbol{A}_X^\alpha$ and from Y to α be $\boldsymbol{A}_Y^\alpha$. Without loss of generality, suppose all polygons are centered at the origin so that we can omit the translation term $\mathbf{t}$ for now. Then by simple linear algebra, $\boldsymbol{A}_X^\alpha = \boldsymbol{A}_Y^\alpha \boldsymbol{A}_X^Y$. Since affine transformations are full-rank, $\boldsymbol{A}_X^Y = \boldsymbol{I}$ if and only if $\boldsymbol{A}_X^\alpha = \boldsymbol{A}_Y^\alpha$, that is, X and Y are the same in every sense (e.g., zero Hausdorff distance) if their relative differences $\boldsymbol{A}_X^\alpha$, $\boldsymbol{A}_Y^\alpha$ to the anchor α are the same. See Figure 1.

The problem with this comparison is that it is binary, i.e., we can only tell that two polygons are identical if $\boldsymbol{A}_X^\alpha = \boldsymbol{A}_Y^\alpha$, but we can not quantify how different they are, in terms of Hausdorff distance for this paper, if $\boldsymbol{A}_X^\alpha \neq \boldsymbol{A}_Y^\alpha$. For example, it is easy to construct two polygons Z, Z' which have identical positive Hausdorff distances to the anchor α, but their corresponding affine transformations $\boldsymbol{A}_Z^\alpha, \boldsymbol{A}_{Z'}^\alpha$ are drastically different.

Our solution is to compare X and Y to a basis of random anchor polygons $\{\alpha_1, \cdots \alpha_d\}$. The total difference between each pair of affine transformations $\boldsymbol{A}_X^{\alpha_i}, \boldsymbol{A}_Y^{\alpha_i}$ will be a good probabilistic approximation of the similarity between X and Y: the more pairs of affine transformations are similar, the more likely the Hausdorff distance between X and Y is small. Formally, we hypothesize that there exists a mapping g such that

$$d_H(X, Y) \approx g(\boldsymbol{A}_X^{\alpha_1}, \cdots, \boldsymbol{A}_X^{\alpha_d}; \boldsymbol{A}_Y^{\alpha_1}, \cdots, \boldsymbol{A}_Y^{\alpha_d}) \tag{1}$$

See Figure 2 for an illustration.

While the mapping does not have a closed-form expression, we can learn g from data using neural networks. It is conceptually related to deep metric learning [9] and implicit neural representations [16] if we consider $(\boldsymbol{A}_X^{\alpha_1}, \cdots, \boldsymbol{A}_X^{\alpha_d})$ and $(\boldsymbol{A}_Y^{\alpha_1}, \cdots, \boldsymbol{A}_Y^{\alpha_d})$ as the coordinates of X and Y in the high-dimensional affine transformation space $\mathbb{R}^{n \times n \times d}$ with constraint

$$\forall 1 \leq i \leq d, A_X^{\alpha_i}, A_Y^{\alpha_i} \text{ are affine matrices;}$$

$$g(\boldsymbol{A}_X^{\alpha_1}, \cdots, \boldsymbol{A}_X^{\alpha_d}; \boldsymbol{A}_X^{\alpha_1}, \cdots, \boldsymbol{A}_X^{\alpha_d}) = 0$$

In practice, $\boldsymbol{A}_X^\alpha$ alone is not sufficient to describe the transformation from X to α because polygons are not always centered and there are residuals after affine transformation, i.e., some shape differences are not affine transformable. Let $(\boldsymbol{A}_X^\alpha, \boldsymbol{t}_X^\alpha) := \min_{\boldsymbol{A},\boldsymbol{t}} d_H(\boldsymbol{A}X + \boldsymbol{t}, \alpha)$, i.e. the affine transformation on X that results in a polygon with the least Hausdorff distance to α. The actual transformation from X to α is

$$\alpha = (\boldsymbol{A}_X^\alpha X + \boldsymbol{t}_X^\alpha) + \epsilon_X^\alpha$$

where $\epsilon_X^\alpha \subset M$ is the non-affine residual. As is discussed in Section 2.2, X and α can be approximated by sampling reference points on the polygons. Let $\boldsymbol{P}_X$ and $\boldsymbol{P}_\alpha$ be $2 \times N$ matrices that represent the N reference points on X and α, respectively, $\boldsymbol{I}$ be a $1 \times N$ vector of all ones, and $\boldsymbol{\epsilon}_X^\alpha$ be a $2 \times N$ matrix that represents the non-affine residual translations between paired points in $\boldsymbol{P}_X$ and $\boldsymbol{P}_\alpha$. Then

$$\boldsymbol{P}_\alpha = \boldsymbol{A}_X^\alpha \boldsymbol{P}_X + \boldsymbol{t}_X^\alpha \boldsymbol{I} + \boldsymbol{\epsilon}_X^\alpha$$

We then extend our hypothesis in Equation 1 to

$$d_H(X, Y) \approx g(\{\boldsymbol{A}_X^{\alpha_i}, \boldsymbol{t}_X^{\alpha_i}, \boldsymbol{\epsilon}_X^{\alpha_i}\}_{i=1}^d; \{\boldsymbol{A}_Y^{\alpha_i}, \boldsymbol{t}_Y^{\alpha_i}, \boldsymbol{\epsilon}_Y^{\alpha_i}\}_{i=1}^d) \tag{2}$$

$\boldsymbol{A}_X^{\alpha_i}, \boldsymbol{A}_Y^{\alpha_i}$ and $\boldsymbol{t}_X^{\alpha_i}, \boldsymbol{t}_Y^{\alpha_i}$ can be obtained by shape registration, but there is no easy way to compute $\boldsymbol{\epsilon}_X^{\alpha_i}, \boldsymbol{\epsilon}_Y^{\alpha_i}$. We hypothesize that the residual Hausdorff distances $d_\epsilon(X, \alpha_i) := d_H(\boldsymbol{A}_X^{\alpha_i} \boldsymbol{P}_X + \boldsymbol{t}_X^{\alpha_i}, \boldsymbol{P}_{\alpha_i})$, $d_\epsilon(Y, \alpha_i) := d_H(\boldsymbol{A}_Y^{\alpha_i} \boldsymbol{P}_Y + \boldsymbol{t}_Y^{\alpha_i}, \boldsymbol{P}_{\alpha_i})$ are good proxies for measuring the effect of $\boldsymbol{\epsilon}_X^{\alpha_i}, \boldsymbol{\epsilon}_Y^{\alpha_i}$ on $d_H(X, Y)$, i.e.,

$$d_H(X, Y) \approx g(\{\boldsymbol{A}_X^{\alpha_i}, \boldsymbol{t}_X^{\alpha_i}, d_\epsilon(X, \alpha_i)\}_{i=1}^d; \{\boldsymbol{A}_Y^{\alpha_i}, \boldsymbol{t}_Y^{\alpha_i}, d_\epsilon(Y, \alpha_i)\}_{i=1}^d) \tag{3}$$

The set $\boldsymbol{r}_X := \{\boldsymbol{A}_X^{\alpha_i}, \boldsymbol{t}_X^{\alpha_i}, d_\epsilon(X, \alpha_i)\}_{i=1}^d\}$ is defined as the RATF of polygon X given random anchor polygons $\{\alpha_i\}_{i=1}^d$. We will see in Section 3.3 that in application there are several vectorized implementations of $\boldsymbol{r}_X$, while being mathematically equivalent, showing very different performance on polygon retrieval tasks.

3.2 Polygon Affine Registration

In this research, we focus on the affine registration of simple polygons with a single boundary. Inspired by the ICP method, we develop a technique to solve the polygon affine registration problem by reducing it to point set registration. This is achieved by sampling points uniformly along the boundary of the polygons.

To sample points uniformly, we distribute the total number of sample points among the edges of the polygon proportionally to their lengths. On each edge, points are sampled equidistantly along the line segment.

Given two point sets $P = \{p_i\}_{i=1}^{N_p}$ and $Q = \{q_j\}_{j=1}^{N_q}$ sampled from the polygons, we simplify the registration problem by centering both point sets at the origin. This is done by subtracting the centroid of each point set:

$$p_i' = p_i - \frac{1}{N_p} \sum_{i=1}^{N_p} p_i, \quad q_j' = q_j - \frac{1}{N_q} \sum_{j=1}^{N_q} q_j.$$

For point sets centered at the origin, the translation vector becomes zero, and the affine registration reduces to finding the optimal linear transformation $\boldsymbol{A}$. The registration problem can be formulated as:

$$\min_{\boldsymbol{A}, \{c_i\}} \sum_{i=1}^{N_p} \left\| \boldsymbol{A} p_i' - q_{c_i}' \right\|_2^2.$$

We adopt an iterative approach similar to ICP to solve for $\boldsymbol{A}$:

(1) **Correspondence Step:** For each point p_i' in P, find the nearest point in Q under the current transformation estimate $\boldsymbol{A}_k$:

$$c_i = \arg\min_j \left\| \boldsymbol{A}_k p_i' - q_j' \right\|_2.$$

(2) **Transformation Update Step:** Update the linear transformation by minimizing the sum of squared distances between the transformed points and their correspondences:

$$\boldsymbol{A}_{k+1} = \arg\min_{\boldsymbol{A}} \sum_{i=1}^{N_p} \left\| \boldsymbol{A} p_i' - q_{c_i}' \right\|_2^2.$$

The transformation update is a least squares problem, which can be efficiently solved using standard linear algebra methods.

An initial transformation is required to start the iterative process. We assume the initial transformation has the form $\boldsymbol{A}_0 = s\boldsymbol{R}$, where s is a scale factor and $\boldsymbol{R}$ is a rotation matrix.

To compute the scale factor s, we calculate the diameter of the point sets:

$$\text{diameter}(P) = \max_i \left\| p_i' \right\|_2, \quad \text{diameter}(Q) = \max_j \left\| q_j' \right\|_2,$$

and set:

$$s = \frac{\text{diameter}(Q)}{\text{diameter}(P)}.$$

For the rotation matrix $\boldsymbol{R}$, we perform Principal Component Analysis (PCA) on the point sets to find their principal axes and align them accordingly.

The complete polygon affine registration algorithm is summarized in Algorithm 1.

3.3 RATFs Polygon Encoding Algorithm

Based on Equation 3 and Algorithm 1, we formally define the algorithm to generate Random Affine Transformation Features of a polygon in Algorithm 2.

Instead of simply extracting the elements of the affine transformation matrix $\boldsymbol{A}_X^{\alpha_i}$, we can alternatively introduce more geometrical inductive bias. By Single Value Decomposition (SVD), one can rewrite $\boldsymbol{A}_X^{\alpha_i}$ uniquely as $\boldsymbol{U}^i \Sigma^i (\boldsymbol{V}^i)^\top$, where the matrices $\boldsymbol{U}^i, \boldsymbol{V}^i$ represent rotations and the diagonal elements λ_1^i, λ_2^i of matrix Σ^i represent scaling factors (i.e. eigenvalues). We can further derive the rotation angles θ_U^i and θ_V^i from $\boldsymbol{U}^i$ and $\boldsymbol{V}^i$, respectively. From an information theoretic perspective, knowing the SVD matrices or the rotation angles and the eigenvalues is equivalent to knowing the affine transformation matrix, but more geometrically interpretable. To investigate whether representing polygons with more explicit geometrical quantities helps with the polygon retrieval task, we construct three variants of RATFs:

Algorithm 1: Polygon Affine Registration Algorithm

Data: Polygons X, Y; number of sampling points N_p, N_q; maximum iterations N; convergence threshold ϵ
Result: Affine transformation matrix $\boldsymbol{A}$, translation vector $\boldsymbol{t}$
$\boldsymbol{P} \leftarrow$ Sample N_p points on X;
$\boldsymbol{Q} \leftarrow$ Sample N_q points on Y;
$p_{\text{center}} \leftarrow \frac{1}{N_p}\sum_{i=1}^{N_p} p_i$, $q_{\text{center}} \leftarrow \frac{1}{N_q}\sum_{j=1}^{N_q} q_j$;
$p'_i \leftarrow p_i - p_{\text{center}}$, $q'_j \leftarrow q_j - q_{\text{center}}$;
$\text{diam}_P \leftarrow \max_i \left\|p'_i\right\|_2$, $\text{diam}_Q \leftarrow \max_j \left\|q'_j\right\|_2$;
scale factor $s \leftarrow \text{diam}_Q / \text{diam}_P$;
Compute initial rotation $\boldsymbol{R}$ using PCA to align P and Q;
Initialize transformation: $\boldsymbol{A}_0 \leftarrow s\boldsymbol{R}$;
$k \leftarrow 0$;
while $k < N$ **do**
  for $i \leftarrow 1$ **to** N_p **do**
    Find correspondence: $c_i \leftarrow \arg\min_j \left\|\boldsymbol{A}_k p'_i - q'_j\right\|_2$;
  end
  Update transformation:
  $\boldsymbol{A}_{k+1} \leftarrow \arg\min_{\boldsymbol{A}} \sum_{i=1}^{N_p} \left\|\boldsymbol{A}p'_i - q'_{c_i}\right\|_2^2$;
  if $\|\boldsymbol{A}_{k+1} - \boldsymbol{A}_k\|_F < \epsilon$ **then**
    $k \leftarrow k + 1$;
    break;
  end
  $k \leftarrow k + 1$;
end
Compute translation: $\boldsymbol{t} \leftarrow q_{\text{center}} - \boldsymbol{A}_k p_{\text{center}}$;
return $\boldsymbol{A}_k, \boldsymbol{t}$;

Algorithm 2: Random Affine Transformation Features (RATFs) Algorithm

Data: Polygon $X \subset M$, number of sampling points N, random anchor polygons $\{\alpha_i\}_{i=1}^{d}$
Result: Random affine transformation features $\boldsymbol{r}_X$ of X
$\boldsymbol{r}_X \leftarrow \emptyset$;
$\boldsymbol{P}_X \leftarrow$ Sample N points on X;
for $i \leftarrow 1$ **to** d **do**
  $\boldsymbol{P}_{\alpha_i} \leftarrow$ Sample N points on α_i;
  $(\boldsymbol{A}_X^{\alpha_i}, \boldsymbol{t}_X^{\alpha_i}) \leftarrow$ Shape registration for (X, α_i);
  $(A^i_{1,1}, A^i_{1,2}, A^i_{2,1}, A^i_{2,2}) \leftarrow$ Extract the elements of $\boldsymbol{A}_X^{\alpha_i}$;
  $(t^i_1, t^i_2) \leftarrow$ Extract the elements of $\boldsymbol{t}_X^{\alpha_i}$;
  $d^i_\epsilon \leftarrow d_H(\boldsymbol{A}_X^{\alpha_i}\boldsymbol{P}_X + \boldsymbol{t}_X^{\alpha_i}, \boldsymbol{P}_{\alpha_i})$ for (X, α_i);
  $\boldsymbol{r}_X \leftarrow \boldsymbol{r}_X \cup \{A^i_{1,1}, A^i_{1,2}, A^i_{2,1}, A^i_{2,2}, t^i_1, t^i_2, d^i_\epsilon\}$;
end
return $\boldsymbol{r}_X$

Basic Variant The same as described in Algorithm 2.

SVD Variant $\boldsymbol{r}_X := \bigcup_{i=1}^{d}\{\boldsymbol{U}^i_{1,1}, \boldsymbol{U}^i_{2,1}, \boldsymbol{V}^i_{1,1}, \boldsymbol{V}^i_{2,1}, \lambda^i_1, \lambda^i_2, t^i_1, t^i_2, d^i_\epsilon\}$

Rotation-Scale Variant $\boldsymbol{r}_X := \bigcup_{i=1}^{d}\{\theta^i_U, \theta^i_V, \lambda_{i,1}, \lambda_{i,2}, t^i_1, t^i_2, d^i_\epsilon\}$

The method for obtaining the affine transformation between two polygons is detailed in Section 3.2.

Experiments show that, slightly conter-intuitively, the basic variant consistently outperforms the others. It may indicate that explicit rotation angles and scaling factors are not the proper inductive bias for the task of preserving Hausdorff distance.

3.4 Neural Polygon Embedding

Having encoded rich information in RATFs, we only need a simple neural network to learn the polygon embedding. The model used in this paper is a 3-layer FFN (feed-forward-network), following the theory that a model with a hidden layer is able to approximate highly complex non-linear mapping. Specifically, the neural polygon embedding is

$$e_X = \boldsymbol{W}_3(ReLU(\boldsymbol{W}_2(ReLU(\boldsymbol{W}_1\boldsymbol{r}_X^\top + \boldsymbol{b}_1)) + \boldsymbol{b}_2)) + \boldsymbol{b}_3$$

Let the dimension of polygon encoding be d. The dimensions of $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3$ are $64 \times d$, 32×64 and 8×32, respectively. This naive design of neural network is only to demonstrate the effectiveness of RATFs. There is nothing that stops researchers to build more advanced neural networks to learn the embeddings. We leave this to future work.

Given a pair of polygons X, Y in the training data and the ground-truth Hausdorff distance $d_H(X, Y)$, the loss is the mean squared error between the cosine distance of polygon embeddings and $d_H(X, Y)$

$$\mathcal{L}(X, Y) = \left|1 - \frac{\langle e_X, e_Y\rangle}{|e_X||e_Y|} - d_H(X, Y)\right|^2$$

By minimizing this loss, the learned neural polygon embedding e_X will have the desirable property that smaller cosine distance implies smaller Hausdorff distance. It is extremely useful for fast polygon retrieval – we can use cosine similarity to rank and retrieve similar polygons from the database.

4 EXPERIMENTS

In this section, we aim to demonstrate the representation capability of RATFs on preserving Hausdorff distance between polygons. For quality control, we focus on synthetic data where polygons are randomly generated and ensured to be simple polygons (i.e., no holes). We train three variants of neural polygon embedding models – Basic, SVD and Rotate-Scale – corresponding to the three variants of RATFs. Considering polygon retrieval as a ranking task, we evaluate the test performance of learned neural polygon embeddings with commonly used metrics such as Mean Squared Error (MSE), Hit@k and Mean Reciprocal Rank (MRR).

4.1 Dataset Preparation and Evaluation Metrics

There are four factors that control the diversity of synthetic polygons: the number of edges N_{data}, the radius r, the spikeness s, and the irregularity irr. We briefly explain their meanings. The number of edges, naturally, means how many edges the polygon has, e.g. a hexagon has $N_{\text{data}} = 6$. N_{data} can be constant for a dataset or a random number, i.e., the dataset consists of polygons of different edge numbers. The radius means the average distance from each

Table 1: Experiments with pentagons. The anchor polygons are also pentagons.

PE Variant	Basic				SVD				Rotate-Scale			
#Anchor	4	8	16	32	4	8	16	32	4	8	16	32
MSE (↓)	0.1029	0.1021	0.1015	**0.1013**	0.1113	0.1089	0.1074	0.1059	0.1243	0.1203	0.1204	0.1184
Hit@100 (↑)	0.5094	0.5042	0.5241	**0.5491**	0.4319	0.4783	0.4853	0.5253	0.3730	0.3941	0.3991	0.4159
MRR (↑)	0.7541	**0.7756**	0.7665	0.7736	0.6829	0.6935	0.7042	0.7244	0.6225	0.6285	0.6182	0.6251

Table 2: Experiments with hexagons. The anchor polygons are also hexagons.

PE Variant	Basic				SVD				Rotate-Scale			
#Anchor	4	8	16	32	4	8	16	32	4	8	16	32
MSE (↓)	0.0911	0.0978	0.0968	**0.0959**	0.1111	0.1027	0.1014	0.1010	0.1207	0.1158	0.1140	0.1120
Hit@100 (↑)	0.4208	0.4626	0.4644	**0.4808**	0.3751	0.4033	0.4432	0.4497	0.3115	0.3340	0.3508	0.3677
MRR (↑)	0.7191	0.7232	0.7306	**0.7355**	0.6092	0.6731	0.6864	0.6947	0.5542	0.5674	0.5701	0.5898

Table 3: Experiments with a random mixture of pentagons and hexagons. The anchor polygons are also a random mixture of pentagons and hexagons.

PE Variant	Basic	SVD	Rotate-Scale
#Anchor	32	32	32
MSE (↓)	**0.0938**	0.0979	0.1098
Hit@100 (↑)	**0.4750**	0.4507	0.3706
MRR (↑)	**0.7288**	0.6665	0.5784

vertex to the center of the polygon, affecting the size of the polygon. Without loss of generality, we set the radius of all datasets to be 1, since we care about the shape similarity, which is invariant under uniform scaling, instead of size differences. The spikeness controls the angle of the sharpest spike of the polygon, and the irregularity controls the evenness of the polygon, i.e., whether the angles by connecting two neighboring vertices and the center have similar radians. We randomly pick spikeness s and irregularity irr for each polygon we generate.

Each train set contains 10,000 random polygons and their pairwise (i.e. 100,000,000 in total) ground-truth Hausdorff distances. Each test set contains 1,000 random polygons and their pairwise ground-truth Hausdorff distances. Let the polygons in the test set be $\{T_i\}_{i=1}^{n}$. For each query polygon T_i in the test set, we define the polygons that have top 10 smallest Hausdorff distances (excluding the query polygon itself) to form the set of ground-truth positives Pos_i, and the rest 989 polygons to be the set of ground-truth negatives Neg_i. Suppose the ascending ranking of polygons according to the neural polygon embedding model is $T_{i_1}, T_{i_2}, \cdots T_{i_{n-1}}$ (excluding T_i itself). Given this setting, we formally define the following evaluation metrics:

Mean Squared Error (MSE)

$$MSE = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbb{1}(i \neq j) \mathcal{L}(T_i, T_j)$$

Hit@k

$$Hit@k = \frac{1}{n} \sum_{i=1}^{n} \frac{|\text{Pos}_i \cap \{T_{i_1} \cdots T_{i_k}\}|}{10}$$

Mean Reciprocal Rank (MRR)

$$MRR = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\min\{k \mid T_{i_k} \in \text{Pos}_i\}}$$

MAE measures the raw difference between ground-truth Hausdorff distances and the predictions made by the neural polygon embedding model. If we do not care about ranking – i.e., we do not need the retrieved polygons to be the "most similar" but just "similar enough", MSE is the good metric to look at. Hit@k measures the proportion of ground-truth positives that fall into the top k model predictions, i.e. how likely the top k model predictions contain the ground-truths. MRR measures how top the model predictions rank the ground-truth positives, i.e., the higher MRR, the more likely one can find ground-truth positives early in the ranking. Three measures focus on different aspects of the ranking quality, combined to form a comprehensive evaluation framework for testing the representational power of RATFs.

4.2 Experiment Results

We report the MSE, Hit@100 and MRR values for polygon retrieval on pentagons and hexagons. We also construct a dataset of polygons with mixed numbers of edges, which resembles the real-world scenarios better. The results are present in Table 1, Table 2 and Table 3. Our neural polygon embedding model, though only trained on a very small amount of data, achieves fairly good retrieval quality on the test set. It is worth noting that, whereas increasing the number of anchor polygons in general has a positive impact on the model performance in all situations, using 4 random anchor polygons

alone yields satisfactory results and the increment by doubling the anchor size diminishes. It is evidence that the representational power of RATFs comes from the careful design of features instead of brute-force dimension increase.

We can also notice that the basic variant of the neural polygon embedding model consistently outperforms the other two variants. It is an interesting finding because the raw elements of affine transformation matrices seem to preserve Hausdorff distances better than the more geometrical interpretation of rotation and scaling. It is worth future investigation.

Compared against raw Hausdorff distance computation, the speedup of using neural polygon embeddings is significant and massive. On the same machine without multi-processing, using SciPy's implementation as the baseline[1], computing the Hausdorff distances of 1 million pairs of polygons sequentially takes 63 seconds, while computing the cosine distances in batches of 1000 pairs only takes around 0.5 milliseconds, 126,000 times speeding-up. The larger the dataset and the larger the batch size, the larger the speedup, since sequential Hausdorff distance computation is $O(n)$ time complexity while the batched cosine distance computation, as long as the memory holds, is $O(1)$.

5 CONCLUSION AND FUTURE WORK

In conclusion, this paper takes a solid step forward towards general neural polygon representations. While there are existing work that encodes polygons as sequences of vertices or closed polylines, we propose a novel approach to represent polygons as collections of anchor polygons, just like how Fourier transformation views functions as collections of functions. This approach naturally circumvents many long-standing problems in vertex-based and polyline-based polygon representations such as variable length, non-convexity and cycle invariance, while remains extremely simplistic. We use a very naive 3-layer neural network to achieve good polygon retrieval quality and magnitude faster computation speed, demonstrating the immense potential of our RATFs method in geospatial searching applications (e.g., in [2] and [11]).

We expect to see future work in two major directions: one direction is to develop more advanced neural networks (of course better than a 3-layer FFN) to boost the polygon retrieval performance to industry-applicable level and find real-world applications in large polygon databases; the other direction is to dig deeper into the science behind RATFs, for example, to what extent the hypothesis we make in Equation 3 holds and whether we can find better proxies to deal with non-affine residuals.

REFERENCES

[1] Josh Achiam et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Dirk Ahlers. 2013. Towards a development process for geospatial information retrieval and search. In *Proceedings of the 22nd International Conference on World Wide Web*. 143–144.

[3] Yoshua Bengio et al. 2013. Representation learning: A review and new perspectives. *TPAMI* 35, 8 (2013), 1798–1828.

[4] Paul J Besl and Neil D McKay. 1992. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, Vol. 1611. Spie, 586–606.

[5] Alexey Dosovitskiy et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR 2020*.

[6] Shaoyi Du, Nanning Zheng, Shihui Ying, and Jianyi Liu. 2010. Affine iterative closest point algorithm for point set registration. *Pattern Recognition Letters* 31, 9 (2010), 791–799. https://doi.org/10.1016/j.patrec.2010.01.020

[7] Hongchao Fan, Alexander Zipf, Qing Fu, and Pascal Neis. 2014. Quality assessment for building footprints data on OpenStreetMap. *International Journal of Geographical Information Science* 28, 4 (2014), 700–719.

[8] Yuan Gong et al. 2022. Ssast: Self-supervised audio spectrogram transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 10699–10709.

[9] Mahmut Kaya and Hasan Şakir Bilge. 2019. Deep metric learning: A survey. *Symmetry* 11, 9 (2019), 1066.

[10] Kenton et al. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.

[11] Hao Li, Jiapan Wang, Johann Maximilian Zollner, Gengchen Mai, Ni Lao, and Martin Werner. 2023. Rethink geographical generalizability with unsupervised self-attention model ensemble: A case study of openstreetmap missing building detection in africa. In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*. 1–9.

[12] Gengchen Mai et al. 2023. Sphere2Vec: A general-purpose location representation learning over a spherical surface for large-scale geospatial predictions. *ISPRS Journal of Photogrammetry and Remote Sensing* 202 (2023), 439–462.

[13] Gengchen Mai, Chiyu Jiang, Weiwei Sun, Rui Zhu, Yao Xuan, Ling Cai, Krzysztof Janowicz, Stefano Ermon, and Ni Lao. 2023. Towards general-purpose representation learning of polygonal geometries. *GeoInformatica* 27, 2 (2023), 289–340.

[14] Baraka Maiseli, Yanfeng Gu, and Huijun Gao. 2017. Recent developments and trends in point set registration methods. *Journal of Visual Communication and Image Representation* 46 (2017), 95–106. https://doi.org/10.1016/j.jvcir.2017.03.012

[15] Marc Rußwurm, Konstantin Klemmer, Esther Rolf, Robin Zbinden, and Devis Tuia. 2024. Geographic Location Encoding with Spherical Harmonics and Sinusoidal Representation Networks. arXiv:2310.06743 [cs.LG] https://arxiv.org/abs/2310.06743

[16] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. 2020. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems* 33 (2020), 7462–7473.

[17] Dmitrii Tihonkih, Artyom Makovetskii, and Vladislav Kuznetsov. 2016. The Iterative Closest Points Algorithm and Affine Transformations.. In *AIST (Supplement)*. 349–356.

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.directed_hausdorff.html