Locality Sensitive Hashing for Efficient Similar Polygon Retrieval

Haim Kaplan
School of Computer Science, Tel Aviv University, Israel

Jay Tenenbaum
School of Computer Science, Tel Aviv University, Israel

Abstract

Locality Sensitive Hashing (LSH) is an effective method of indexing a set of items to support efficient nearest neighbors queries in high-dimensional spaces. The basic idea of LSH is that similar items should produce hash collisions with higher probability than dissimilar items.

We study LSH for (not necessarily convex) polygons, and use it to give efficient data structures for similar shape retrieval. Arkin et al. [2] represent polygons by their "turning function" - a function which follows the angle between the polygon's tangent and the x-axis while traversing the perimeter of the polygon. They define the distance between polygons to be variations of the L_p (for $p = 1, 2$) distance between their turning functions. This metric is invariant under translation, rotation and scaling (and the selection of the initial point on the perimeter) and therefore models well the intuitive notion of shape resemblance.

We develop and analyze LSH near neighbor data structures for several variations of the L_p distance for functions (for $p = 1, 2$). By applying our schemes to the turning functions of a collection of polygons we obtain efficient near neighbor LSH-based structures for polygons. To tune our structures to turning functions of polygons, we prove some new properties of these turning functions that may be of independent interest.

As part of our analysis, we address the following problem which is of independent interest. Find the vertical translation of a function f that is closest in L_1 distance to a function g. We prove tight bounds on the approximation guarantee obtained by the translation which is equal to the difference between the averages of g and f.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Theory of computation → Computational geometry; Information systems → Information retrieval

Keywords and phrases Locality sensitive hashing, polygons, turning function, L_p distance, nearest neighbors, similarity search

Digital Object Identifier 10.4230/LIPIcs.STACS.2021.46

Related Version *Full Version*: https://arxiv.org/abs/2101.04339

Funding This work was supported by ISF grant no. 1595/19, GIF grant no. 1367/2017 and the Blavatnik Foundation.

1 Introduction

This paper focuses on similarity search between polygons, where we aim to efficiently retrieve polygons with a shape resembling the query polygon.

Large image databases are used in many multimedia applications in fields such as computer vision, pattern matching, content-based image retrieval, medical diagnosis and geographical information systems. Retrieving images by their content in an efficient and effective manner has therefore become an important task, which is of rising interest in recent years.

When designing content-based image retrieval systems for large databases, the following properties are typically desired:

38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021).
Editors: Markus Bläser and Benjamin Monmege; Article No. 46; pp. 46:1–46:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- **Efficiency:** Since the database is very large, iterating over all objects is not feasible, so an efficient indexing data structure is necessary.
- **Human perception:** The retrieved objects should be perceptually similar to the query.
- **Invariance to transformations:** The retrieval probability of an object should be invariant to translating, scaling, and rotating the object. Moreover, since shapes are typically defined by a time signal describing their boundary, we desire invariance also to the initial point of the boundary parametrization.

There are two general methods to define how much two images are similar (or distant): intensity-based (color and texture) and geometry-based (shape). The latter method is arguably more intuitive [17] but more difficult since capturing the shape is a more complex task than representing color and texture features. Shape matching has been approached in several other ways, including tree pruning [18], the generalized Hough transform [5], geometric hashing [16] and Fourier descriptors [20]. For an extensive survey on shape matching metrics see Veltkamp and Hagedoorn [19].

A noteworthy distance function between shapes is that of Arkin et al. [2], which represents a curve using a cumulative angle function. Applied to polygons, the *turning function* (as used by Arkin et al. [2]) t_P of a polygon P returns the cumulative angle between the polygon's counterclockwise tangent at the point and the x-axis, as a function of the fraction x of the perimeter (scaled to be of length 1) that we have traversed in a counterclockwise fashion. The turning function is a step function that changes at the vertices of the polygon, and either increases with left turns, or decreases with right turns (see Figure 2). Clearly, this function is invariant under translation and scale of the polygon.

To find similar polygons based on their turning functions, we define the distance $L_p(P,Q)$ between polygons P and Q to be the L_p distance between their turning functions $t_P(x)$ and $t_Q(x)$. That is

$$L_p(P,Q) = \left(\int_0^1 |t_P(x) - t_Q(x)|^p\right)^{1/p}.$$

The turning function $t_P(x)$ depends on the rotation of P, and the (starting) point of P where we start accumulating the angle. If the polygon is rotated by an angle α, then the turning function $t_P(x)$ becomes $t_P(x) + \alpha$. Therefore, we define the (rotation invariant) distance $D_p^{\updownarrow}(P,Q)$ between polygons P and Q to be the $D_p^{\updownarrow}$ distance between their turning functions t_P and t_Q, which is defined as follows

$$D_p^{\updownarrow}(P,Q) \overset{def}{=} D_p^{\updownarrow}(t_P,t_Q) \overset{def}{=} \min_{\alpha\in\mathbb{R}} L_p(t_P+\alpha, t_Q) = \min_{\alpha\in\mathbb{R}} \sqrt[p]{\int_0^1 |t_P(x) + \alpha - t_Q(x)|^p \, dx}.$$

If the starting point of P is clockwise shifted along the boundary by t, the turning function $t_P(x)$ becomes $t_P(x+t)$. Thus, we define the distance $D_p(P,Q)$ between polygons P and Q to be the D_p distance between their turning functions t_P and t_Q which is defined as follows

$$D_p(P,Q) \overset{def}{=} D_p(t_P,t_Q) \overset{def}{=} \min_{\alpha\in\mathbb{R}, t\in[0,1]} \left(\int_0^1 |t_P(x+t) + \alpha - t_Q(x)|^p\right)^{1/p}.$$

The distance $D_p(f,g)$ between two functions f and g extends f to the domain $[0,2]$ by defining $t_P(x+1) = t_P(x) + 2\pi$. The distance metric D_p is invariant under translation, rotation, scaling and the selection of the starting point. A comprehensive presentation of these distances, as well as a proof that they indeed satisfy the metric axioms appears in [2].

We develop efficient nearest neighbor data structures for functions under these distances and then specialize them to functions which are turning functions of polygons.

Since a major application of polygon similarity is content-based image retrieval from large databases (see Arkin et al. [2]), the efficiency of the retrieval is a critical metric. Traditionally, efficient retrieval schemes used tree-based indexing mechanisms, which are known to work well for prevalent distances (such as the Euclidean distance) and in low dimensions. Unfortunately such methods do not scale well to higher dimensions and do not support more general and computationally intensive metrics. To cope with this phenomenon (known as the "curse of dimensionality"), Indyk and Motwani [15, 14] introduced Locality Sensitive Hashing (LSH), a framework based on hash functions for which the probability of hash collision is higher for near points than for far points.

Using such hash functions, one can determine near neighbors by hashing the query point and retrieving the data points stored in its bucket. Typically, we concatenate hash functions to reduce false positives, and use several hash functions to reduce false negatives. This gives rise to a data structure which satisfies the following property: for any query point q, if there exists a neighbor of distance at most r to q in the database, it retrieves (with constant probability) a neighbor of distance at most cr to q for some constant $c > 1$. This data structure is parameterized by the parameter $\rho = \frac{\log(p_1)}{\log(p_2)} < 1$, where p_1 is the minimal collision probability for any two points of distance at most r, and p_2 is the maximal collision probability for any two points of distance at least cr. The data structure can be built in time and space $O(n^{1+\rho})$, and its query time is $O(n^\rho \log_{1/p_2}(n))$ where n is the size of the data set.[1]

The trivial retrieval algorithm based on the turning function distance of Arkin et al. [2], is to directly compute the distance $D_2(P, Q)$ (or $D_1(P, Q)$) between the query Q and all the polygons P in the database. This solution is invariant to transformations but not efficient (i.e., linear in the size of the database).

In this paper, we rely on the turning function distance of Arkin et al. [2] for $p = 1, 2$, and create the first retrieval algorithm with respect to the turning function distance which is sub-linear in the size of the dataset. To do so, we design and analyze LSH retrieval structures for function distance, and feed the turning functions of the polygons to them. Our results give rise to a shape-based content retrieval (a near neighbor polygon) scheme which is efficient, invariant to transformations, and returns perceptually similar results.

Our contribution

We develop simple but powerful (r, cr)-LSH near neighbor data structures for efficient similar polygon retrieval, and give a theoretical analysis of their performance. We give the first structure (to the best of our knowledge) for approximate similar polygon retrieval which is provably invariant to shape rotation, translation and scale, and with a query time which is sub-linear in the number of data polygons. In contrast to many other structures for similar shape retrieval which often use heuristics, all our results are backed with theoretical proofs, using properties of the turning function distance and the theory of LSH.

[1] To ease on the reader, in this paper we suppress the term $1/p_1$ in the structure efficiency, and the time it takes to compute a hash and distances between two polygons/functions. For example for polygons with at most m vertices (which we call m-gons), all our hash computations take $O(m)$ time, and using Arkin et al. [2] we may compute distances in $O(m^2 \log(m))$ time.

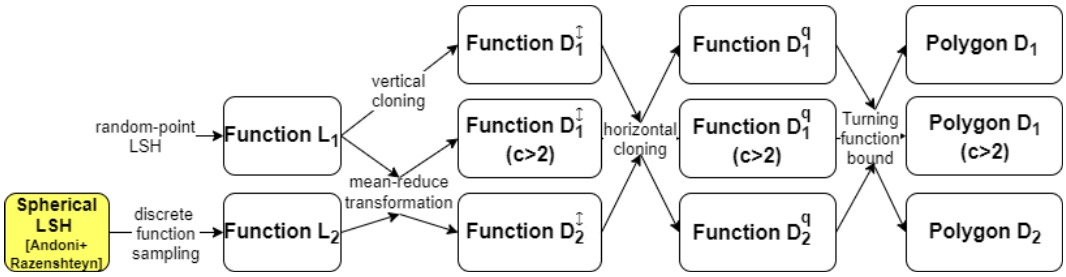

Figure 1 Our structures: each box is an (r, cr)-LSH near neighbor data structure, and the arrow $A \to B$ with label t signifies that we use the method t over the structure A to get a structure for B.

To give our (r, cr)-LSH near neighbor data structures for polygons, we build such structures for step functions with distances which are derived from the L_p distance for $p = 1, 2$, and apply them to turning functions of polygons.[2] Here $r > 0$ and $c > 1$ are the LSH parameters as defined above, and n is the number of objects in the data structure. The (r, cr)-LSH data structures which we present exist for any $r > 0$ and $c > 1$ (except when c is explicitly constrained). For an interval I, we say that a function $f : I \to \mathbb{R}$ is a k-step function, if I can be divided into k sub-intervals, such that over each sub-interval f is constant. All the following results for functions are for k-step functions with ranges bounded in $[a, b]$ for some $a < b$ where for simplicity of presentation, we fix $a = 0$ and $b = 1$.[3,4] The results we present below are slightly simplified versions than those that appear in the body of the paper. For an overview of our structures see Figure 1.

Near neighbors data structures for functions

1. For the L_1 distance over functions, we design a simple but powerful LSH hash family. This hash selects a uniform point p from the rectangle $[0, 1] \times [0, 1]$, and maps each function to 1, 0 or -1 based on its vertical relation (above, on or below) with p. This yields an (r, cr)-LSH structure for L_1 which requires sub-quadratic preprocessing time and space of $O(n^{1+\rho})$, and sub-linear query time of $O(n^{\rho} \log n)$, where $\rho = \log(1 - r)/\log(1 - cr) \le \frac{1}{c}$. For the L_2 distance over functions, we observe that sampling each function at evenly spaced points reduces the L_2 distance to Euclidean distance. We use the data structure of Andoni and Razenshteyn [1] for the Euclidean distance to give an (r, cr)-LSH for the L_2 distance, which requires sub-quadratic preprocessing time of $O(n^{1+\rho} + n_{r,c} \cdot n)$, sub-quadratic space of $O(n_{r,c} \cdot n^{1+\rho})$ and sub-linear query time of $O(n_{r,c} \cdot n^{\rho})$, where $\rho = \frac{1}{2c-1}$ and $n_{r,c} = \frac{2k}{(\sqrt{c}-1)r^2}$ is the dimension of the sampled vectors. We also give an alternative asymmetric LSH hash family for the L_2 distance inspired by our hash family for the L_1 distance, and create an LSH structure based on it.

2. For the $D_2^{\updownarrow}$ distance, we leverage a result of Arkin et al. [2], to show that the mean-reduce transformation, defined to be $\hat{\phi}(x) = \phi(x) - \int_0^1 \phi(s)ds$, reduces $D_2^{\updownarrow}$ distances to L_2 distances with no approximation loss. That is, for every f and g, $D_2^{\updownarrow}(f, g) = L_2(\hat{f}, \hat{g})$, so we get an

[2] Our structures for step functions can be extended to support also functions which are concatenations of at most $k \in \mathbb{N}$ functions which are M-Lipschitz for some $M > 0$. Also, we can give similar structures for variations of the function D_1 and D_2 distances where we extend the functions from the domain $[0, 1]$ to the domain $[0, 2]$, not by $f(x) = f(x - 1) + 2\pi$, but by $f(x) = f(x - 1) + q$ for any constant $q \in \mathbb{R}$.

[3] For general values of these parameters, the dependency of the data structure's run-time and memory is roughly linear or squared in $b - a$.

[4] Since $a = 0$ and $b = 1$, the distance between any two functions is at most 1, so we focus on $r < 1$.

(r, cr)-LSH structure for the $D_2^{\updownarrow}$ distance which uses our previous L_2 structure, and with identical performance. For the $D_1^{\updownarrow}$ distance, we approximately reduce $D_1^{\updownarrow}$ distances to L_1 distances using the same mean-reduction. We give a simple proof that this reduction gives a 2-approximation, and improve it to a tight approximation bound showing that for any two step functions $f, g : [0,1] \to [0,1]$, $L_1(\hat{f}, \hat{g}) \leq \left(2 - D_1^{\updownarrow}(f,g)\right) \cdot D_1^{\updownarrow}(f,g)$. This proof (see full version), which is of independent interest, characterizes the approximation ratio by considering the function $f - g$, dividing its domain into 3 parts and averaging over each part, thereby considering a single function with 3 step heights. This approximation scheme yields an (r, cr)-LSH structure for any $c > 2 - r$, which is substantially smaller than 2 (approaching 1) for large values of r.

We also give an alternative structure *step-shift-LSH* that supports any $c > 1$, but has a slightly diminished performance. This structure leans on the observation of Arkin et al. [2], that the optimal vertical shift aligns a step of f with a step of g. It therefore replaces each data step function by a set of vertical shifts of it, each aligning a different step value to $y = 0$, and constructs an L_1 data structure containing all these shifted functions. It then replaces a query with its set of shifts as above, and performs a query in the internal L_1 structure with each of these shifts.

3. For the D_1 and D_2 distances, we leverage another result of Arkin et al. [2], that the optimal horizontal shift horizontally aligns a discontinuity point of f with a discontinuity point of g. Similarly to *step-shift-LSH*, we give a structure for D_1 (or D_2) by keeping an internal structure for $D_1^{\updownarrow}$ (or $D_2^{\updownarrow}$) which holds a set of horizontal shifts of each data functions, each aligns a different discontinuity point in to $x = 0$. It then replaces a query with its set of shifts as above, and performs a query in the internal structure with each of these shifts.

Near neighbors data structures for polygons

We design LSH structures for the polygonal D_1 and D_2 distances, by applying the D_1 and D_2 structures to the turning functions of the polygons. We assume that all the data and query polygons have at most m vertices (are m-gons), where m is a constant known at preprocessing time. It is clear that the turning functions are $(m+1)$-step functions, but the range of the turning functions is not immediate (note that performance inversely relates to the range size).

First, we show that turning functions of m-gons are bounded in the interval $I = [-(\lfloor m/2 \rfloor - 1)\pi, (\lfloor m/2 \rfloor + 3)\pi]$ of size $\lambda_m := (2 \cdot \lfloor m/2 \rfloor + 2)\pi$. We show that this bound is tight in the sense that there are m-gons whose turning functions get arbitrarily close to these upper and lower bounds.

Second, we define the *span* of a function $\xi : [0,1] \to \mathbb{R}$ to be $span(\xi) = \max_{x \in [0,1]}(\xi(x)) - \min_{x \in [0,1]}(\xi(x))$, and show that for m-gons, the span is at most $\lambda_m/2 = (\lfloor m/2 \rfloor + 1)\pi$, and that this bound is tight - there are m-gons whose turning functions have arbitrarily close spans to $\lambda_m/2$. Since the D_1 and D_2 distances are invariant to vertical shifts, we perform an a priori vertical shift to each turning function such that its minimal value becomes 0, effectively morphing the range to $[0, \lambda_m/2]$, which is half the original range size. This yields the following structures:

For the D_1 distance, for any $c > 2$ we give an (r, cr)-LSH structure storing n polygons with at most m vertices which requires $O((nm)^{1+\rho})$ preprocessing time and space which are sub-quadratic in n, and $O(m^{1+\rho} n^{\rho} \log(nm))$ query time which is sub-linear in n, where ρ is roughly $2/c$. Also for D_1, for any $c > 1$ we get an (r, cr)-LSH structure which requires sub-quadratic preprocessing time and space of $O((nm^2)^{1+\rho})$, and sub-linear query time of $O(m^{2+2\rho} n^{\rho} \log(nm))$, where ρ is roughly $1/c$.

For the D_2 distance, we give an (r, cr)-LSH structure which requires sub-quadratic preprocessing time of $\tilde{O}(n^{1+\rho})$, sub-quadratic space of $\tilde{O}(n^{1+\rho})$, and sub-linear query time of $\tilde{O}(n^{\rho})$, where $\rho = \frac{1}{2\sqrt{c}-1}$.[5]

Other similar works

Babenko et al. [4] suggest a practical method for similar image retrieval, by embedding images to a Euclidean space using Convolutional Neural Networks (CNNs), and retrieving similar images to a given query based on their embedding's euclidean distance to the query embedding. This approach has been the most effective practical approach for similar image retrieval in recent years.

Gudmundsson and Pagh [13] consider a metric in which there is a constant grid of points, and shapes are represented by the subset of grid points which are contained in them. The distance between polygons is then defined to be the *Jaccard distance* between the corresponding subsets of grid points. Their solution lacks invariance to scale, translation and rotation, however our work is invariant to those, and enables retrieving polygons which have a similar shape, rather than only spatially similar ones.

Other metrics over shapes have been considered. Cakmakov et al. [7] defined a metric based on snake-like moving of the curves. Bartolini et al. [6] proposed a new distance function between shapes, which is based on the Discrete Fourier Transform and the Dynamic Time Warping distance. Chavez et al. [9] give an efficient polygon retrieval technique based on Fourier descriptors. Their distance works for exact matches, but is a weak proxy for visual similarity, since it relates to the distances between corresponding vertices of the polygons.

There has been a particular effort to develop efficient structures for the discrete Fréchet distance and the dynamic time warping distance for polygonal curves in $\mathbb{R}^d$. Such works include Driemel et al. [10] who gave LSH structures for these metrics via snapping the curve points to a grid, Ceccarello et al. [8] who gave a practical and efficient algorithm for the r-range search for the discrete Fréchet distance, Filtser et al. [11] who built a deterministic approximate near neighbor data structure for these metrics using a subsample of the data, and Astefanoaei et al. [3] who created a suite of efficient sketches for trajectory data. Grauman and Darrell [12] performed efficient contour-based shape retrieval (which is sensitive (not invariant) to translations, rotations and scaling) using an embedding of Earth Mover's Distance into L_1 space and LSH.

2 Preliminaries

We first formally define LSH, then discuss the turning function representation of Arkin et al. [2], and then define the distance functions between polygons and functions which rise from this representation.

2.1 Locality sensitive hashing

We use the following standard definition of a *Locality Sensitive Hash Family (LSH)* with respect to a given distance function $d : Z \times Z \to \mathbb{R}_{\geq 0}$.

[5] The $\tilde{O}$ notation hides multiplicative constants which are small powers (e.g., 5) of m, $\frac{1}{r}$ and $\frac{1}{\sqrt[4]{c}-1}$.

▶ **Definition 1** (Locality Sensitive Hashing (LSH)). *Let $r > 0$, $c > 1$ and $p_1 > p_2$. A family H of functions $h : Z \to \Gamma$ is an (r, cr, p_1, p_2)-LSH for a distance function $d : Z \times Z \to \mathbb{R}_{\geq 0}$ if for any $x, y \in Z$,*

1. *If $d(x, y) \leq r$ then $\Pr_{h \in H}[h(x) = h(y)] \geq p_1$, and*
2. *If $d(x, y) \geq cr$ then $\Pr_{h \in H}[h(x) = h(y)] \leq p_2$.*

Note that in the definition above, and in all the following definitions, the hash family H is always sampled uniformly.

We say that a hash family is an (r, cr)-*LSH* for a distance function d if there exist $p_1 > p_2$ such that it is an (r, cr, p_1, p_2)-LSH. A hash family is a *universal LSH* for a distance function d if for all $r > 0$ and $c > 1$ it is an (r, cr)-LSH.

From an (r, cr, p_1, p_2)-LSH family, we can derive, via the general theory developed in [15, 14], an (r, cr)-*LSH data structure*, for finding approximate near neighbors with respect to r. That is a data structure that finds (with constant probability) a neighbor of distance at most cr to a query q if there is a neighbor of distance at most r to q. This data structure uses $O(n^{1+\rho})$ space (in addition to the data points), and $O(n^\rho \log_{1/p_2}(n))$ hash computations per query, where $\rho = \frac{\log(1/p_1)}{\log(1/p_2)} = \frac{\log(p_1)}{\log(p_2)}$.

2.2 Representation of polygons

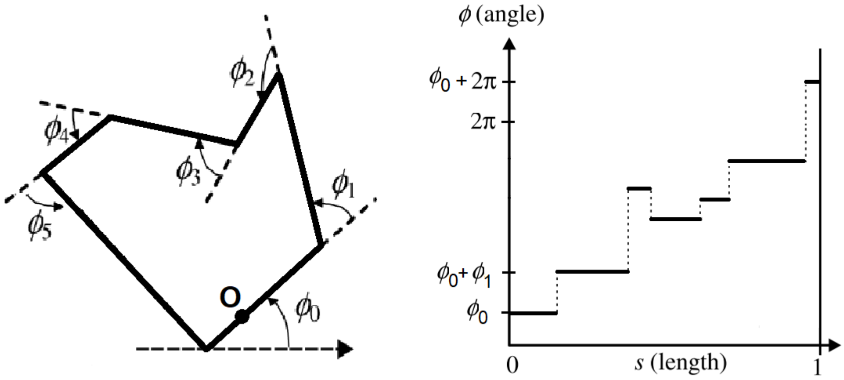

Figure 2 Left: a polygon P with 6 vertices. Right: the turning function t_P of P, with 7 steps.

Let P be a simple polygon scaled such that its perimeter is one. Following the work of Arkin et al. [2], we represent P via a *turning function* $t_P(s) : [0, 1] \to \mathbb{R}$, that specifies the angle of the counterclockwise tangent to P with the x-axis, for each point q on the boundary of P. A point q on the boundary of P is identified by its counterclockwise distance (along the boundary which is of length 1 by our scaling) from some fixed reference point O. It follows that $t_P(0)$ is the angle α that the tangent at O creates with the x-axis, and $t_P(s)$ follows the cumulative turning, and increases with left turns and decreases with right turns. Although t_P may become large or small, since P is a simple closed polygon we must have that $t_P(1) = t_P(0) + 2\pi$ if O is not a vertex of P, and $t_P(1) - t_P(0) \in [\pi, 3\pi]$ otherwise. Figure 2 illustrates the polygon turning function.

Note that since the angle of an edge with the x-axis is constant and angles change at the vertices of P, then the function is constant over the edges of P and has discontinuity points over the vertices. Thus, the turning function is in fact a step function.

In this paper, we often use the term m-gon – a polygon with **at most** m vertices.

2.3 Distance functions

Consider two polygons P and Q, and their associated turning functions $t_P(s)$ and $t_Q(s)$ accordingly. Define the *aligned* L_p *distance* (often abbreviated to L_p *distance*) between P and Q denoted by $L_p(P,Q)$, to be the L_p distance between $t_P(s)$ and $t_Q(s)$ in $[0,1]$: $L_p(P,Q) = \sqrt[p]{\int_0^1 |t_P(x) - t_Q(x)|^p \, dx}$.

Note that even though the L_p distance between polygons is invariant under scale and translation of the polygon, it depends on the rotation of the polygon and the choice of the reference points on the boundaries of P and Q.

Since rotation of the polygon results in a vertical shift of the function t_P, we define the *vertical shift-invariant* L_p distance between two functions f and g to be $D_p^{\updownarrow}(f,g) = \min_{\alpha\in\mathbb{R}} L_p(f+\alpha, g) = \min_{\alpha\in\mathbb{R}} \sqrt[p]{\int_0^1 |f(x) + \alpha - g(x)|^p \, dx}$. Accordingly, we define the *rotation-invariant* L_p distance between two polygons P and Q to be the vertical shift-invariant L_p distance between the turning functions t_P and t_Q of P and Q respectively: $D_p^{\updownarrow}(P,Q) = D_p^{\updownarrow}(t_P, t_Q) = \min_{\alpha\in\mathbb{R}} \sqrt[p]{\int_0^1 |t_P(x) + \alpha - t_Q(x)|^p \, dx}$.

To tweak the distance $D_p^{\updownarrow}$ such that it will be invariant to changes of the reference points, we need the following definition. We define the 2π*-extension* $f^{2\pi} : [0,2] \to \mathbb{R}$ of a function $f : [0,1] \to \mathbb{R}$ to the domain $[0,2]$, to be $f^{2\pi} = \begin{cases} f(x), & \text{for } x \in [0,1] \\ f(x-1) + 2\pi, & \text{for } x \in (1,2] \end{cases}$.

A turning function t_P is naturally 2π-extended to the domain $[0,2]$ by circling around P one more time. We define the u-slide of a function $g : [0,2] \to \mathbb{R}$, $slide_u^{\leftrightarrow}(g) : [0,1] \to \mathbb{R}$, for a value $u \in [0,1]$ to be $(slide_u^{\leftrightarrow}(g))(x) = g(x+u)$. These definitions are illustrated in Figure 3. Note that shifting the reference point by a counterclockwise distance of u around the perimeter of a polygon P changes the turning function from t_P to $slide_u^{\leftrightarrow}(t_P^{2\pi})$.

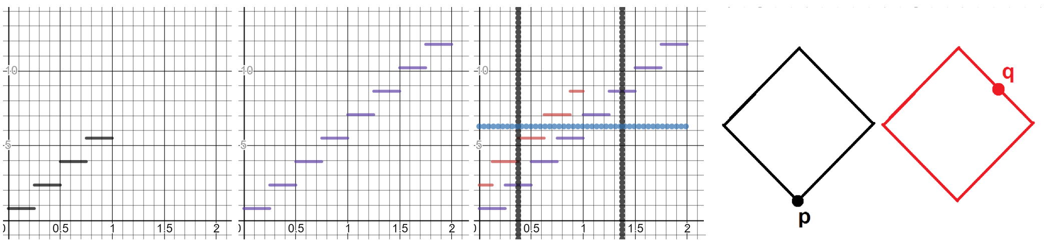

Figure 3 Left: The turning function t_P of the square with reference point p. Center: the 2π-extension $t_P^{2\pi}$ of t_P. Right: The turning function of the square with the reference point q in red (this is in fact the function $t_P^{2\pi}$ cropped to between the black vertical lines, i.e., to $[0.375, 1.375]$).

We therefore define the (vertical and horizontal) *shift-invariant* L_p *distance* between two functions $f, g : [0,1] \to \mathbb{R}$ to be: $D_p(f,g) = \min_{u\in[0,1]} D_p^{\updownarrow}(slide_u^{\leftrightarrow}(f^{2\pi}), g) = \min_{\alpha\in\mathbb{R},\ u\in[0,1]} \sqrt[p]{\int_0^1 |f^{2\pi}(x+u) + \alpha - g(x)|^p \, dx}$, and define the (rotation and reference point invariant) L_p distance between two polygons P and Q to be $D_p(P,Q) = D_p(t_P, t_Q)$. Arkin et al. [2] proved that $D_p(f,g)$ is a metric for any $p > 0$.

3 L_1-based distances

In this section, we give LSH structures for the L_1 distance, the $D_1^{\updownarrow}$ distance and then the D_1 distance. Note that the D_1 distance reduces to the $D_1^{\updownarrow}$ distance, which by using the *mean-reduction* transformation presented in Section 3.2, reduces to the L_1 distance.

3.1 Structure for L_1

In this section we present *random-point-LSH*, a simple hash family for functions $f : [0,1] \to [a,b]$ with respect to the L_1 distance. *Random-point-LSH* is the hash family $H_1(a,b) = \{h_{(x,y)} \mid (x,y) \in [0,1] \times [a,b]\}$, where the points (x,y) are uniformly selected from the rectangle $[0,1] \times [a,b]$. Each $h_{(x,y)}$ receives a function $f : [0,1] \to [a,b]$, and returns 1 if f is vertically above the point (x,y), returns -1 if f is vertically below (x,y), and 0 otherwise.

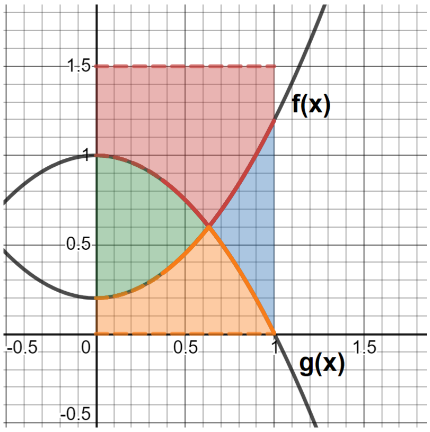

Figure 4 Illustration of the hash of two functions f and g w.r.t. $h_{(x,y)}$ for $a = 0$ and $b = 1.5$. For (x,y) in the green area $h_{(x,y)}(f) = -1 \neq 1 = h_{(x,y)}(g)$, in the blue area $h_{(x,y)}(f) = 1 \neq -1 = h_{(x,y)}(g)$, in the red area $h_{(x,y)}(f) = h_{(x,y)}(g) = -1$, and in the orange area $h_{(x,y)}(f) = h_{(x,y)}(g) = 1$.

The intuition behind *random-point-LSH* is that any two functions $f, g : [0,1] \to [a,b]$ collide precisely over hash functions $h_{(x,y)}$ for which the point (x,y) is outside the area bounded between the graphs of f and g. This fact is illustrated in the following Figure 4. Thus, this hash incurs a collision probability of $1 - \frac{L_1(f,g)}{b-a} = 1 - \frac{L_1(f,g)}{b-a}$, which is a decreasing function with respect to $L_1(f,g)$. This intuition leads to the following results.

▶ **Theorem 2.** *For any two functions $f, g : [0,1] \to [a,b]$, we have that $P_{h \sim H_1(a,b)}(h(f) = h(g)) = 1 - \frac{L_1(f,g)}{b-a}$.*

Proof. Fix $x \in [0,1]$, and denote by $U(S)$ the uniform distribution over a set S. We have that

$$P_{y \sim U([a,b])}(h_{(x,y)}(f) = h_{(x,y)}(g)) = 1 - P_{y \sim U([a,b])}(h_{(x,y)}(f) \neq h_{(x,y)}(g))$$
$$= 1 - \frac{|f(x) - g(x)|}{b-a},$$

where the last equality follows since $h_{(x,y)}(f) \neq h_{(x,y)}(g)$ precisely for the y values between $f(x)$ and $g(x)$. Therefore, by the law of total probability,

$$P_{h \sim H_1(a,b)}(h(f) = h(g)) = P_{(x,y) \sim U([0,1] \times [a,b])}(h_{(x,y)}(f) = h_{(x,y)}(g))$$
$$= \int_0^1 P_{y \sim U([a,b])}(h_{(x,y)}(f) = h_{(x,y)}(g)) dx$$
$$= \int_0^1 \left(1 - \frac{|f(x) - g(x)|}{b-a}\right) dx = 1 - \frac{L_1(f,g)}{b-a}.$$ ◀

▶ **Corollary 3.** *For any $r > 0$ and $c > 1$, one can construct an $(r, cr)-$LSH structure for the L_1 distance for n functions with ranges bounded in $[a, b]$. This structure requires $O(n^{1+\rho})$ space and preprocessing time, and has $O(n^\rho \log(n))$ query time, where $\rho = \frac{\log\left(1-\frac{r}{b-a}\right)}{\log\left(1-\frac{cr}{b-a}\right)} \approx \frac{1}{c}$ for $r \ll b - a$.*

Proof. Fix $r > 0$ and $c > 1$. By the general result of Indyk and Motwani [15], it suffices to show that $H_1(a, b)$ is an $(r, cr, 1 - \frac{r}{b-a}, 1 - \frac{cr}{b-a})$-LSH for the L_1 distance.

Indeed, by Theorem 2, $P_{h\sim H_1(a,b)}(h(f) = h(g)) = 1 - \frac{L_1(f,g)}{b-a}$, so we get that

- If $L_1(f, g) \leq r$, then $P_{h\sim H_1(a,b)}(h(f) = h(g)) = 1 - \frac{L_1(f,g)}{b-a} \geq 1 - \frac{r}{b-a}$.
- If $L_1(f, g) \geq cr$, then $P_{h\sim H_1(a,b)}(h(f) = h(g)) = 1 - \frac{L_1(f,g)}{b-a} \leq 1 - \frac{cr}{b-a}$. ◀

3.2 Structure for $D_1^\updownarrow$

In this section we present *mean-reduce-LSH*, an LSH family for the vertical translation-invariant L_1 distance, $D_1^\updownarrow$. Observe that finding an LSH family for $D_1^\updownarrow$ is inherently more difficult than for L_1, since even evaluating $D_1^\updownarrow(f, g)$ for a query function g and an input function f requires minimizing $L_1(f + \alpha, g)$ over the variable α, and the optimal value of α depends on both f and g.

Our structure requires the following definitions. We define $\bar{\phi} = \int_0^1 \phi(x)dx$ to be the mean of a function ϕ over the domain $[0, 1]$, and define the *mean-reduction* of ϕ, denoted by $\hat{\phi} : [0, 1] \to [a - b, b - a]$, to be the vertical shift of ϕ with zero integral over $[0, 1]$, i.e., $\hat{\phi}(x) = \phi(x) - \bar{\phi}(x)$. These definitions are illustrated in Figure 5. Our solution relies on the crucial observation that for the pair of functions $f, g : [0, 1] \to [a, b]$, the value of α which minimizes $L_1(f + \alpha, g)$ is "well approximated" by $\bar{g} - \bar{f}$. That is the distance $L_1(f + (\bar{g} - \bar{f}), g) = L_1(f - \bar{f}, g - \bar{g}) = L_1(\hat{f}, \hat{g})$ approximates $D_1^\updownarrow(f, g)$. This suggests that if we replace any data or query function f with $\hat{f}$, then the $D_1^\updownarrow$ distances are approximately the L_1 distances of the shifted versions $\hat{f}$, for which we can use the hash H_1 from Section 3.1.

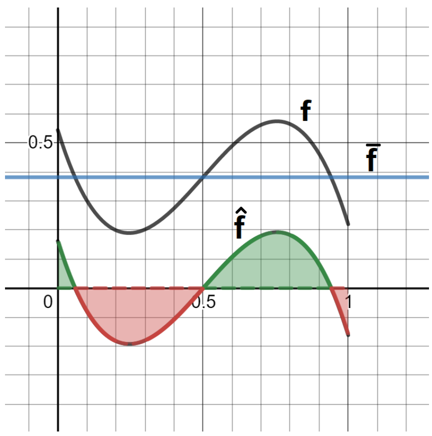

Figure 5 A function f (black), its mean $\bar{f}$(blue), and its mean-reduction $\hat{f}$ (below). Notice that the red and green areas are equal.

Indeed, we use the hash family H_1 from Section 3.1, and define *mean-reduce-LSH* for functions with images contained in $[a, b]$ to be the family $H_1^\updownarrow(a, b) = \{f \to h \circ \hat{f} \mid h \in H_1(a - b, b - a)\}$. Each hash of $H_1^\updownarrow(a, b)$ is defined by a function $h \in H_1(a - b, b - a)$, and given a function f, it applies h on its mean-reduction $\hat{f}$.

The following theorem gives a tight bound for the L_1 distance between mean-reduced functions in terms of their original vertical translation-invariant L_1 distance $D_1^{\updownarrow}$. The proof of this tight bound as well as a simpler 2-approximation appear in the full version of the paper. Our elegant but more complicated proof of the tight bound characterizes and bounds the approximation ratio using properties of $f-g$, and demonstrates its tightness by giving the pair of step functions f,g which meet the bound.

We conclude this result in the following theorem.

▶ **Theorem 4.** *Let $f,g:[0,1]\to[a,b]$ be step functions and let $r\in(0,b-a]$ be their vertical shift-invariant L_1 distance $r=D_1^{\updownarrow}(f,g)$. Then $r\le L_1(\hat{f},\hat{g})\le\left(2-\frac{r}{b-a}\right)\cdot r$. This bound is tight, i.e, there exist two functions f_0,g_0 as above for which $L_1(\hat{f_0},\hat{g_0})=\left(2-\frac{r}{b-a}\right)\cdot r$.*

We use Theorem 4 to prove that *mean-reduce-LSH* is an LSH family (Theorem 5). We then use Theorem 5 and the general result of Indyk and Motwani [15] to get Corollary 6.

▶ **Theorem 5.** *For any $r\in(0,b-a)$ and $c>2-\frac{r}{b-a}$, $H_1^{\updownarrow}(a,b)$ is an $\left(r,cr,1-\left(2-\frac{r}{b-a}\right)\cdot\frac{r}{2(b-a)},1-c\cdot\frac{r}{2(b-a)}\right)$-LSH family for the $D_1^{\updownarrow}$ distance.*

▶ **Corollary 6.** *For any $r>0$ and $c>2-\frac{r}{b-a}$, one can construct an $(r,cr)-$LSH structure for the $D_1^{\updownarrow}$ distance for n functions with ranges bounded in $[a,b]$. This structure requires $O(n^{1+\rho})$ extra space and preprocessing time, and $O(n^{\rho}\log(n))$ query time, where $\tilde{r}=r/(2(b-a))$ and $\rho=\log\left(1-(2-2\tilde{r})\cdot\tilde{r}\right)/\log\left(1-c\tilde{r}\right)$ for small $\tilde{r}$.*

Step-shift-LSH

We present *step-shift-LSH*, a structure for the $D_1^{\updownarrow}$ distance which works for any $c>1$ (unlike *mean-reduce-LSH*), but has a slightly worse performance, which depends on an upper bound k on the number of steps in of the data and query functions. This structure uses an internal structure for the L_1 distance, and leverages the observation of Arkin et al. [2] that the optimal vertical shift α to align two step functions f and g, is such that $f+\alpha$ has a step which partially overlaps a step of g, i.e., there is some segment $S\subseteq[0,1]$ over which $f+\alpha=g$.

Therefore, we overcome the uncertainty of the optimal α by a priori cloning each function by the number of steps it has, and vertically shifting each clone differently to align each step to be at $y=0$.[6] For a query function g, we clone it similarly to align each step to $y=0$, and use each clone as a separate query for the L_1 structure. This process effectively gives a chance to align each step of the query g with each step of each data step function f.

▶ **Corollary 7.** *For any $a<b$, $r>0$ and $c>1$, there exists an (r,cr)-LSH structure for the $D_1^{\updownarrow}$ distance for n functions, each of which is a k-step function with range bounded in $[a,b]$. This structure requires $O((nk)^{1+\rho})$ extra space and preprocessing time, and $O(k^{1+\rho}n^{\rho}\log(nk))$ query time, where $\rho=\log\left(1-\frac{r}{2(b-a)}\right)/\log\left(1-\frac{cr}{2(b-a)}\right)\approx\frac{1}{c}$ for $r\ll b-a$.*

3.3 Structure for D_1

In this section, we present *slide-clone-LSH*, a data structure for the distance function D_1 defined over step functions $f:[0,1]\to[a,b]$. To do so, we use an $(r',c'r')$-LSH data structure (for appropriate values of r' and c') for the distance function $D_1^{\updownarrow}$ which will hold slided functions with ranges contained in $[a,b+2\pi]$.

[6] This idea of cloning appears once again (but in a horizontal version), and in more detail, in Section 3.3 for the D_1 distance.

Recall that the D_1 distance between a data function f and a query function g is defined to be the minimal $D_1^{\updownarrow}$ distance between a function in the set $\left\{slide_u^{\leftrightarrow}(f^{2\pi}) \mid u \in [0,1]\right\}$ and the function g, and we obviously do not know u a priori and cannot build a structure for each possible $u \in [0,1]$. Fortunately, in the proof of Theorem 6 from Arkin et al. [2], they show that for any pair of step functions f and g, the optimal slide u is such that a discontinuity of f is aligned with a discontinuity of g. They show that this is true also for the D_2 distance.

Therefore, we can overcome the uncertainty of the optimal u by a priori cloning each function by the number of discontinuity points it has, and sliding each clone differently to align its discontinuity point to be at $x = 0$. For a query function g, we clone it similarly to align each discontinuity point to $x = 0$, use each clone as a separate query. The above process effectively gives a chance to align each discontinuity point of the query function g with each discontinuity point of each data step function f.

Slide-clone-LSH works as follows.

Preprocessing phase

We are given the parameters $r > 0$, $c > 1$, $a < b$ and a set of step functions F, where each function is defined over the domain $[0,1]$ and has a range bounded in $[a,b]$. Additionally, we are given an upper bound k on the number of steps a data or query step function may have. First, we replace each function $f \in F$ with the set of (at most $k+1$) u slides of it's 2π-extension for each discontinuity point u, i.e., $slide_u^{\leftrightarrow}(f^{2\pi})$ for each discontinuity point $u \in [0,1]$. For each such clone we remember its original unslided function. Next, we store the at most $(k+1) \cdot |F|$ resulted functions in an $(r', c'r')$-LSH data structure for the $D_1^{\updownarrow}$ distance for functions with ranges bounded in $[a, b+2\pi]$, tuned with the parameters $r' = r$ and $c' = c$.

Query phase

Let g be a query function. We query the $D_1^{\updownarrow}$ structure constructed in the preprocessing phase with each of the slided queries $slide_u^{\leftrightarrow}(g^{2\pi})$ for each discontinuity point $u \in [0,1]$. If one of the queries returns a data function f, we return its original unslided function, and otherwise return nothing.

In Theorem 8, we prove that *slide-clone-LSH* is an (r, cr)-data structure for D_1.

▶ **Theorem 8.** *Slide-clone-LSH is an (r, cr)-LSH structure for the D_1 distance.*

▶ **Corollary 9.** *For any $a < b$, $r > 0$, $\omega = b + 2\pi - a$ and $c > 2 - \frac{r}{\omega}$, there exists an (r, cr)-LSH structure for the D_1 distance for n functions, each of which is a k-step function with range bounded in $[a,b]$. This structure requires $O((nk)^{1+\rho})$ extra space and preprocessing time, and $O(k^{1+\rho} n^{\rho} \log(nk))$ query time, where $\tilde{r} = r/(2\omega)$ and $\rho = \log\left(1 - (2 - 2\tilde{r}) \cdot \tilde{r}\right) / \log\left(1 - c\tilde{r}\right) \approx \frac{2}{c}$ for small $\tilde{r}$.*[7]

▶ **Corollary 10.** *For any $a < b$, $r > 0$ and $c > 1$, there exists an (r, cr)-LSH structure for the D_1 distance for n functions, each of which is a k-step function with range bounded in $[a,b]$. This structure requires $O((nk^2)^{1+\rho})$ extra space and preprocessing time, and $O(k^{2+2\rho} n^{\rho} \log(nk))$ query time, where $\rho = \log\left(1 - \frac{r}{2(b+2\pi-a)}\right) / \log\left(1 - \frac{cr}{2(b+2\pi-a)}\right) \approx \frac{1}{c}$ for $r \ll 2(b + 2\pi - a)$.*

[7] Given a bound s on the span of the functions, we can a priori vertically shift all the functions such that their minimum is 0, effectively making the range size smaller (within $[0,s]$) and improving the performance of the structure (see the full version).

4 L_2-based distances

This section, which appears in detail in the full version of the paper, gives LSH structures for the L_2 distance, the $D_2^{\updownarrow}$ distance and then the D_2 distance.

First, we present *discrete-sample-LSH*, a simple LSH structure for functions $f : [0,1] \to [a,b]$ with respect to the L_2 distance. The intuition behind *discrete-sample-LSH* is that the L_2 distance between the step functions $f, g : [0,1] \to [a,b]$ can be approximated via a sample of f and g at the evenly spaced set of points $\{i/n\}_{i=0}^{n}$. Specifically, by replacing each function f by the vector $vec_n(f) = \left(\frac{1}{\sqrt{n}} f\left(\frac{0}{n}\right), \frac{1}{\sqrt{n}} f\left(\frac{1}{n}\right), \ldots, \frac{1}{\sqrt{n}} f\left(\frac{n-1}{n}\right)\right)$, one can show that for a large enough value of $n \in \mathbb{N}$, $L_2(f,g)$ can be approximated by $L_2\left(vec_n(f) - vec_n(g)\right)$. We prove that for any two k-step functions $f, g : [0,1] \to [a,b]$, and for any $r > 0$ and $c > 1$: **(1)** if $L_2(f,g) \le r$ then $L_2\left(vec_{n_{r,c}}(f), vec_{n_{r,c}}(g)\right) \le c^{1/4} r$, and **(2)** if $L_2(f,g) > cr$ then $L_2\left(vec_{n_{r,c}}(f), vec_{n_{r,c}}(g)\right) > c^{3/4} r$ for a sufficiently large $n_{r,c}$ (see full version for the exact value). Note that the bounds $A = c^{1/4} r$ and $B = c^{3/4} r$ are selected for simplicity, and other trade-offs are possible. The proof of this claim relies on the observation that $(f-g)^2$ is also a step function, and that $L_2\left(vec_{n_{r,c}}(f), vec_{n_{r,c}}(g)\right)^2$ is actually the left Riemann sum of $(f-g)^2$, so as $n \to \infty$, it must approach $\int_0^1 (f(x) - g(x))^2 dx = (L_2(f,g))^2$. *Discrete-sample-LSH* replaces data and query functions f with the vector samples $vec_{n_{r,c}}(f)$, and holds an $(c^{1/4} r, c^{3/4} r)$-LSH structure for the $n_{r,c}$-dimensional Euclidean distance (e.g., the *Spherical-LSH* based structure of Andoni and Razenshteyn [1]). The resulting structure has the parameter $\rho = \frac{1}{2c-1}$.

In the full version of the paper, we present an alternative structure tailored for the L_2 distance for general (not necessarily k-step) integrable functions $f : [0,1] \to [a,b]$, based on a simple and efficiently computable asymmetric hash family which uses *random-point-LSH* as a building block. We note that this structure's ρ values are larger than those of *discrete-sample-LSH* for small values of r.

Next, we give *vertical-alignment-LSH*– a structure for $D_2^{\updownarrow}$. Recall that the mean-reduction (Section 3.2) of a function f is defined to be $\hat{f}(x) = f(x) - \int_0^1 f(t)dt$. We show that the *mean-reduction* has no approximation loss when used for reducing $D_2^{\updownarrow}$ distances to L_2 distances, i.e., it holds that $D_2^{\updownarrow}(f,g) = L_2\left(\hat{f}, \hat{g}\right)$ for any f, g. Thus, to give an (r, cr)-LSH structure for $D_2^{\updownarrow}$, *vertical-alignment-LSH* simply holds a (r, cr)-LSH structure for L_2, and translates data and query functions f for $D_2^{\updownarrow}$ to data and query functions $\hat{f}$ for L_2.

Finally, we employ the same cloning and sliding method as in Section 3.3, to obtain an (r, cr)-LSH structure for D_2 using a structure for $D_2^{\updownarrow}$.

5 Polygon distance

In this section (which appears in detail in the full version of the paper) we consider polygons, and give efficient structures to find similar polygons to an input polygon. All the results of this section depend on a fixed value $m \in \mathbb{N}$, which is an upper bound on the number of vertices in all the polygons which the structure supports (both data and query polygons). Recall that the distance functions between two polygons P and Q which we consider, are defined to be variations of the L_p distance between the turning functions t_P and t_Q of the polygons, for $p = 1, 2$. To construct efficient structures for similar polygon retrieval, we apply the structures from previous sections to the turning functions of the polygons.

To apply these structures and analyze their performance, it is necessary to bound the range of the turning functions, and represent them as k-step functions. Since the turning functions are $(m+1)$-step functions, it therefore remains to compute bounds for the range of the turning function t_P.

A coarse bound of $[-(m+1)\pi, (m+3)\pi]$ can be derived by noticing that the initial value of the turning function is in $[0, 2\pi]$, that any two consecutive steps in the turning function differ by an angle less than π, and that the turning function has at most $m+1$ steps.
We give an improved and tight bound for the range of the turning function, which relies on the fact that turning functions may wind up and accumulate large angles, but they must almost completely unwind towards the end of the polygon traversal, such that $t_P(1) \in [t_P(0)+\pi, t_P(0)+3\pi]$. Our result is as follows.

▶ **Theorem 11** (Simplified). *Let P be a polygon with m vertices. Then for the turning function t_P, $\forall x \in [0,1], -(\lfloor m/2 \rfloor - 1)\pi \le t_P(x) \le (\lfloor m/2 \rfloor + 3)\pi$, and this bound is tight.*

We denote the lower and upper bounds on the range by $a_m = -(\lfloor m/2 \rfloor - 1)\pi$ and $b_m = (\lfloor m/2 \rfloor + 3)\pi$ respectively, and define λ_m to be the size of this range, $\lambda_m = (2 \cdot \lfloor m/2 \rfloor + 2)\pi$. Having the results above, we get LSH structures for the different corresponding polygonal distances which support polygons with at most m vertices, by simply replacing each data and query polygon by its turning function.

Regarding the distances $D_1^{\updownarrow}$ and D_1, we can improve the bound above using the crucial observation that even though the range of the turning function may be of size near $m\pi$, its span can actually only be of size approximately $\frac{m}{2} \cdot \pi$ (Theorem 12), where we define the span of a function ϕ over the domain $[0,1]$, to be $span(\phi) = \max_{x\in[0,1]}(\phi(x)) - \min_{x\in[0,1]}(\phi(x))$.

A simplified version of this result is as follows.

▶ **Theorem 12** (Simplified). *Let Q be a polygon with m vertices. Then for the turning function t_Q, it holds that $span(t_Q) \le (\lfloor m/2 \rfloor + 1)\pi = \lambda_m/2$. Moreover, for any $\varepsilon > 0$ there exists such a polygon with span at least $(\lfloor m/2 \rfloor + 1)\pi - \varepsilon$.*

Since the $D_1^{\updownarrow}$ distance is invariant to vertical shifts, we can improve the overall performance of our $D_1^{\updownarrow}$ LSH structure by simply mapping each data and query polygon $P \in S$ to its vertically shifted turning function $x \to t_P(x) - \min_{z\in[0,1]} t_P(z)$ (such that its minimal value becomes 0). This shift morphs the ranges of the set of functions F to be contained in $[0, \max_{f\in F}(span(f))]$. By Theorem 12, we can therefore use the adjusted bounds of $a = 0$ and $b = \lambda_m/2$ (each function $f \in S_0$ is obviously non-negative, but also bounded above by $\lambda_m/2$ by Theorem 12), and effectively halve the size of the range from $\lambda_m = b_m - a_m$ to $\lambda_m/2$.

To summarize our results for polygons, we use the $\tilde{O}$ notation to hide multiplicative constants which are small powers (e.g., 5) of m, $\frac{1}{r}$, and $\frac{1}{\sqrt{c}-1}$:
For the D_1 distance, for any $c > 2$ we give an (r, cr)-LSH structure which for $r \ll \frac{2\lambda_m}{c}$ roughly requires $\tilde{O}(n^{1+\rho})$ preprocessing time and space, and $\tilde{O}(n^{1+\rho}\log n)$ query time, where ρ is roughly $\frac{2}{c}$. Also for D_1, for any $c > 1$ we get an (r, cr)-LSH structure which for $r \ll \lambda_m$ roughly requires $O((nm^2)^{1+\rho})$ preprocessing time and space, and $O(m^{2+2\rho}n^{\rho}\log(nm))$ query time, where ρ is roughly $1/c$.

For the D_2 distance, we give an (r, cr)-LSH structure which requires $\tilde{O}(n^{1+\rho})$ preprocessing time, $\tilde{O}(n^{1+\rho})$ space, and $\tilde{O}(n^{\rho})$ query time, where $\rho = \frac{1}{2\sqrt{c}-1}$.

6 Conclusions and directions for future work

We present several novel LSH structures for searching nearest neighbors of functions with respect to the L_1 and the L_2 distances, and variations of these distances which are invariant to horizontal and vertical shifts. This enables us to devise efficient similar polygon retrieval structures, by applying our nearest neighbor data structures for functions, to the turning functions of the polygons. For efficiently doing this, we establish interesting bounds on the range and span of the turning functions of m-gons.

As part of our analysis, we proved that for any two functions $f, g : [0,1] \to [a,b]$ such that $D_1^{\updownarrow}(f,g) = r$, it holds that $L_1(\hat{f},\hat{g}) \le \left(2 - \frac{r}{b-a}\right) \cdot r$. This tight approximation guarantee may be of independent interest. An interesting line for further research is to find near neighbor structures with tighter guarantees for simple and frequently occurring families of polygons such as rectangles, etc.

All the reductions we describe have some performance loss, which is reflected in the required space, preprocessing and query time. Finding optimal reduction parameters (e.g., an optimal value of ξ in Section 3.3 for polygons) and finding more efficient reductions is another interesting line for further research. Finding an approximation scheme for the horizontal distance (similarly to the $\left(2 - \frac{r}{b-a}\right)$-approximation for the $D_1^{\updownarrow}$ distance which appears in Section 3.2) is another intriguing open question.

References

1. Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *STOC*, pages 793–801. ACM, 2015.
2. Esther M Arkin, L Paul Chew, Daniel P Huttenlocher, Klara Kedem, and Joseph S Mitchell. An efficiently computable metric for comparing polygonal shapes. Technical report, Cornell University, 1991.
3. Maria Astefanoaei, Paul Cesaretti, Panagiota Katsikouli, Mayank Goswami, and Rik Sarkar. Multi-resolution sketches and locality sensitive hashing for fast trajectory processing. In *SIGSPATIAL*, pages 279–288. ACM, 2018.
4. Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *ECCV*, pages 584–599. Springer, 2014.
5. Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.
6. Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. Using the time warping distance for fourier-based shape retrieval. Technical report, IEIIT-BO-03-02, 2002.
7. Dusan Cakmakov and Emilija Celakoska. Estimation of curve similarity using turning functions. *International Journal of Applied Mathematics*, 15:403–416, 2004.
8. Matteo Ceccarello, Anne Driemel, and Francesco Silvestri. Fresh: Fréchet similarity with hashing. In *WADS*, pages 254–268. Springer, 2019.
9. Edgar Chávez, Ana C Chávez Cáliz, and Jorge L López-López. Affine invariants of generalized polygons and matching under affine transformations. *Computational Geometry*, 58:60–69, 2016.
10. Anne Driemel and Francesco Silvestri. Locality-sensitive hashing of curves. In *SOCG*, pages 37:1–37:16, 2017.
11. Arnold Filtser, Omrit Filtser, and Matthew J Katz. Approximate nearest neighbor for curves—simple, efficient, and deterministic. *arXiv preprint*, 2019. `arXiv:1902.07562`.
12. Kristen Grauman and Trevor Darrell. Fast contour matching using approximate earth mover's distance. In *CVPR*, pages I–220. IEEE, 2004.
13. Joachim Gudmundsson and Rasmus Pagh. Range-efficient consistent sampling and locality-sensitive hashing for polygons. In *ISAAC*, 2017.
14. Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of computing*, 8(1):321–350, 2012.
15. Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, pages 604–613. ACM, 1998.
16. Yehezkel Lamdan and Haim J Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *ICCV*, page 238–249, 1988.
17. Lambert Schomaker, Edward de Leau, and Louis Vuurpijl. Using pen-based outlines for object-based annotation and image-based queries. In *AVI*, pages 585–592. Springer, 1999.

18 Shinji Umeyama. Parameterized point pattern matching and its application to recognition of object families. *TPAMI*, 15(2):136–144, 1993.

19 Remco C Veltkamp and Michiel Hagedoorn. State of the art in shape matching. In *Principles of visual information retrieval*, pages 87–119. Springer, 2001.

20 Charles T Zahn and Ralph Z Roskies. Fourier descriptors for plane closed curves. *TOC*, 100(3):269–281, 1972.