Indexed Polygon Matching under Similarities

Fernando Luque-Suarez[1], J. L. López-López[2], and Edgar Chavez[1]0000-0002-0148-695X

[1] CICESE Ensenada, Mexico
elchavez@cicese.mx
[2] UMSNH, México
jlopez@umich.mx

Abstract. Polygons appear as constructors in many applications and deciding if two polygons match under similarity transformations and noise is a fundamental problem. Solutions in the literature consider only matching pairs of polygons, implying a sequential comparison when we have a collection. In this paper, we present the first algorithm allowing indexed retrieval of polygons under similarities. We reduce the problem to searching points in the plane, exact searching in the absence of noise, and approximate searching for similar noisy polygons. The above gives a $O(n + \log(m))$ time algorithm to find the matching polygons under noise and $O(1)$ time for exact similar polygons. We tested our heuristic for indexed polygons in an extensive collection of convex, star-shaped, simple, and self-intersecting polygons. For small amounts of noise, we achieve perfect recall for all polygons. For large amounts of noise, the lowest recall is for convex polygons, while attaining the highest recall is for general (self-intersecting) polygons. The above is not a significant limitation. To recover convex polygons efficiently before indexing, we define a random permutation of the vertices, converting all input polygons to a general polygon and achieving the same successful recovery rates, which is a perfect recall for high noise levels.

Keywords: Polygon matching, Shape matching, Shape indexing

1 Introduction

Shapes appear in many applications fields like computer-aided design, computer-aided manufacturing, computer vision [18], medical imaging [12] and even archaeology [16]. Shape analysis deals with the concept of matching shapes. The definition of matching changes with the application field. It ranges from congruence transformations, where the shapes could be rotated, translated, or reflected without being scaled, to similarity transformations, which includes scaling to the previous set of transformations, affine transformations, projective transformations, to Riemannian isometries (for curved surfaces), and conformal mappings, or more general transformations. Matching could also include partial matching, where only a portion of the shape has a match. As a rule of thumb, the more general the transformation, it is more difficult to find a fast algorithm to find the best match. For an arbitrary transformation, the problem is NP-complete.

We fix our attention on the fundamental problem of complete polygon matching, as opposed to partial matching. The problem of partial matching, or dealing with insertions and deletions, can be handled by fragmenting the polygons being compared. Moreover, we are especially interested in the indexed version of the problem instead of just comparing two polygons for matching.

1.1 The problem: indexed polygon matching

We define the problem of *indexed matching*. A collection of polygons is preprocessed and stored, and a query is presented to the system. The outcome will be all the matching shapes in the collection.

We shall identify points (x, y) in the plane with corresponding complex numbers $z = x + y\sqrt{-1}$. A polygon in the plane will be an ordered set of points, or complex numbers, where the *order specifies consecutive vertices*. Self-intersection are allowed since the order is arbitrary. Therefore, the cyclic shifts $(z_2, z_3, \ldots, z_n, z_1)$, $(z_3, z_4, \ldots, z_1, z_2), \ldots, (z_n, z_1, \ldots, z_{n-2}, z_{n-1})$ and the reversed labeling $(z_n, z_{n-1}, \ldots, z_2, z_1)$ determine different labels for the same polygon $(z_1, z_2, \ldots, z_{n-1}, z_n)$. But a general permutation $\mathfrak{p} : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ could determine a different polygon $(z_{\mathfrak{p}(1)}, \ldots, z_{\mathfrak{p}(n)})$ because the consecutive vertices vary and therefore the edges are different.

An affine transformation $f : \mathbb{R}^2 \to \mathbb{R}^2$ can be (uniquely) written in terms of sums and products of complex numbers as

$$f(z) = \alpha z + \beta \bar{z} + \gamma,$$

where $\alpha, \beta, \gamma \in \mathbb{C}$ and $|\alpha|^2 - |\beta|^2 = \det f \neq 0$. Here $\bar{z}$ stands for the complex conjugated of z. When $\beta = 0$ the affine transformation is a similarity transformation.

Given polygons $Z = (z_1, z_2, \ldots, z_n) \in \mathbb{C}^n$ and $W = (w_1, w_2, \ldots, w_n) \in \mathbb{C}^n$, our problem consists of determining if there exists an affine transformation f such that $Z = f(W)$. Since affine transformations have three complex parameters, finding two corresponding triples of consecutive points in both polygons is enough. A naïve procedure will be to fix a triplet in Z and try all the cyclic shifts in W to find the correspondence, which takes $O(n)$ operations for one triplet. Since there are $O(n)$ consecutive triplets, the entire process takes $O(n^2)$ operations.

Now assume we have a given collection of polygons $Z_1, Z_2, \ldots, Z_m$ and a query polygon W, and we want to know which of the Z_ℓ are images of W under a similarity. Using a sequential approach and the naïve procedure above, the solution can be found in $O(mn^2)$ operations. In general, without an index, the complexity will depend linearly on the number of polygons in the collection, multiplied by the complexity of an individual match. We will show how to improve this complexity using the defined invariants and a two-dimensional index for querying.

1.2 Summary of results

For polygons $Z \in \mathbb{C}^n$, we construct complex scalar functions $\varphi_j : \mathbb{C}^n \to \mathbb{C}, j = 1, \ldots, \lfloor (n-1)/2 \rfloor$ with the following properties

1. $\varphi_j^n(Z) = \varphi_j^n(f(Z))$ with $f : \mathbb{C}^n \to \mathbb{C}^n$ an arbitrary similarity function, including mirroring and cyclic shifts.
2. φ_j is analytic, that is for $\Delta Z = (\Delta z_1, \ldots, \Delta z_n)$, an unknown *bounded* additive noise, we have $|\varphi_j(Z)^n - \varphi_j(Z + \Delta(Z))^n| \leq r$.
3. If $Z, W \in \mathbb{C}^n$ and Z and W are not similar, then $\varphi_j(Z) \neq \varphi_j(W)$ almost surely.
4. For a collection of polygons $Z_1, \ldots, Z_m$ and a query polygon W, we show how to use the previous properties to preprocess $Z_1, \ldots, Z_m$ to quickly find all Z_i such that $Z_i = f(W)$. This is done by using a two-dimensional spatial index to store $\varphi_j(Z_1)^n, \ldots, \varphi_j(Z_m)^n$ at preprocessing time, and finding the nearest neighbor of $\varphi_j(W)^n$, as $NN(\varphi_j(W)^n$ at query time.
5. The above procedure has high recall only for general, auto-intersecting polygons when the amount of noise Δ is above a certain threshold. We show that if we permute the polygons Z_i before indexing, using an arbitrary but fixed permutation Π, we can obtain the same high recall results even for convex polygons, which had the lowest recall rates without permutations.

This paper is an experimental report of a previous theoretical paper [7]. We reproduce here the mains results to make this contribution self-contained. The experimental parts, not reported before, corresponds to numerals 4 and 5 above. In particular, using the nearest neighbor search, or k-nearest neighbor search when we expect multiple matches for the query polygon is new. In the previous paper, we derived precise bounds, also discussed for a complete presentation, where there was the need for a precise maximum radius. This radius depends on the polygon as well as the amount of noise. Each polygon has associated a maximum allowed noise, posing difficulties for indexing, as convex polygons are more sensitive to noise. We show experimentally in this paper that we can achieve maximum tolerated noise for most polygons by randomly permuting the polygon vertex with a fixed permutation.

1.3 Related work

Before matching polygons in a natural scene, it is necessary to detect them. In general, reconstructing an arbitrary polygon from partial readings is NP-hard [5]; although some instances can be solvable in practice, such as detecting regular polygons [13, 4]. However, shapes and contours can be obtained from other sources, and sometimes the problem consists in measuring if a set of points can be put in correspondence with a nominal polygon; this has applications in manufacturing inspection and city planning [10]. Two arbitrary simple polygons can be compared using several notions of distance [2], being the more general the Fréchet distance as described in [6], which can be computed in polynomial time.

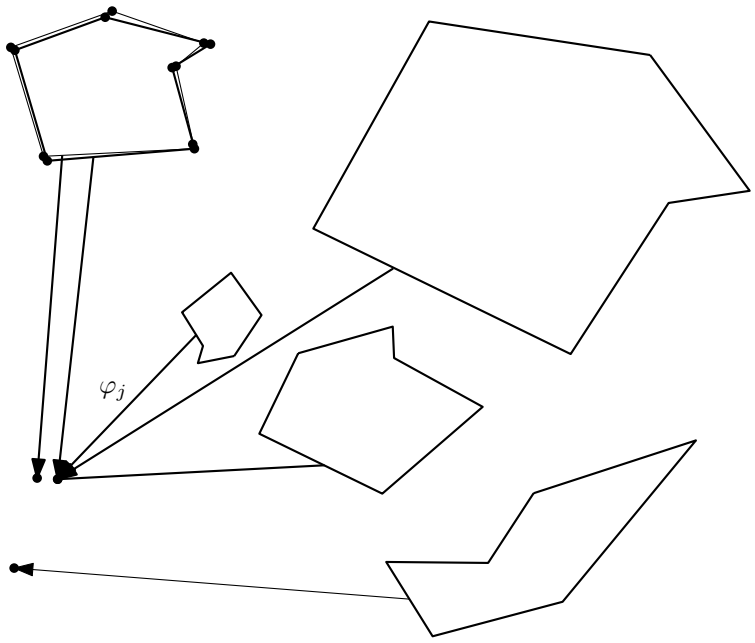

Fig. 1. Application of the invariant φ_j to similar polygons gives the same complex number. For similar polygons plus noise, it gives complex numbers that are close under euclidean distance. For not similar polygons it gives different complex numbers.

Some heuristics have been defined for other simpler realizations of the distance between polygons. One approach is to consider the polygon as a (circular) string with either the edges [17] or the vertices as symbols [11]; this allows efficient comparison of a shape against a nominal polygon allowing insertions and deletions of vertices. Another efficient metric is discussed in [1]. In [15] are discussed algorithms for the specific case of polygon matching upon congruency, including the case of partial matches. A more comprehensive discussion of the problem of shape matching and several efficient approaches are discussed in the survey [18].

To the best of our knowledge, there is no prior attempt to solve the indexed polygon matching discussed in this paper. The metrics mentioned above are designed to compare pairs of polygons and do not contemplate the problem of indexed matching. Moreover, any function φ_j holds more information than a metric because φ_j endows the space of polygons with a two-dimensional coordinate system (complex numbers are two-dimensional). In contrast, a metric can only be considered a one-dimensional coordinate because each polygon is associated with a real number, which is the distance to a fixed polygon.

1.4 Acknowledgments

We want to thank David Mount for carefully reading an early version of this manuscript and providing precious suggestions. We are grateful to Tomas Auer and Martin Held[3] who maintain a repository for polygon generation. We used their software to generate polygons of various types for our experiments.

2 Invariants

2.1 Similarity invariants for polygons

In what follows, we shall fix an integer $n \geq 3$.

Definition 1. *For any integer $j = 1, \ldots, \lfloor (n-1)/2 \rfloor$ we consider the function $\varphi_j : \mathbb{C}^n \to \mathbb{C} \cup \{\infty\}$ given by*

$$\varphi_j(z_1, \ldots, z_n) = \frac{\sum_{k=1}^{n} \lambda^{jk} z_k}{\sum_{k=1}^{n} \lambda^{-jk} z_k},$$

where $\lambda = e^{2\pi\sqrt{-1}/n}$ is a nth root of unit.

Proposition 1. *φ_j is invariant under the action of orientation-preserving similarity transformations on polygons with n vertices; that is, if $\alpha, \gamma \in \mathbb{C}$ with $\alpha \neq 0$, then*

$$\varphi_j(\alpha z_1 + \gamma, \alpha z_2 + \gamma, \ldots, \alpha z_n + \gamma) = \varphi_j(z_1, z_2, \ldots, z_n).$$

Proof.

$$\varphi_j(\alpha z_1+\gamma,\alpha z_2+\gamma,\ldots,\alpha z_n+\gamma)=\frac{\sum_{k=1}^n\lambda^{jk}(\alpha z_k+\gamma)}{\sum_{k=1}^n\lambda^{-jk}(\alpha z_k+\gamma)}=$$

$$\frac{\alpha\sum_{k=1}^n\lambda^{jk}z_k+\gamma\sum_{k=1}^n\lambda^{jk}}{\alpha\sum_{k=1}^n\lambda^{-jk}z_k+\gamma\sum_{k=1}^n\lambda^{-jk}}=\frac{\alpha\sum_{k=1}^n\lambda^{jk}z_k+\gamma\sum_{k=0}^{n-1}\lambda^{jk}}{\alpha\sum_{k=1}^n\lambda^{-jk}z_k+\gamma\sum_{k=0}^{n-1}\lambda^{-jk}}$$

$$\frac{\alpha\sum_{k=1}^n\lambda^{jk}z_k+\gamma(\lambda^{jn}-1)/(\lambda^j-1)}{\alpha\sum_{k=1}^n\lambda^{-jk}z_k+\gamma(\lambda^{-jn}-1)/(\lambda^{-j}-1)}=$$

$$\frac{\alpha\sum_{k=1}^n\lambda^{jk}z_k}{\alpha\sum_{k=1}^n\lambda^{-jk}z_k}=\frac{\sum_{k=1}^n\lambda^{jk}z_k}{\sum_{k=1}^n\lambda^{-jk}z_k}=\varphi_j(z_1,z_2,\ldots,z_n).$$

Remarks

Remark 1. The numerator and denominator involved in the definition of φ_j are the coefficients appearing when $Z=(z_1,\ldots,z_n)$ is expressed in certain basis of $\mathbb{C}^n$, namely the basis of star-shaped polygons

$$E_k=((\lambda^k)^1,(\lambda^k)^2,\ldots,(\lambda^k)^{n-1},(\lambda^k)^n),\quad k=1,2,\ldots,n$$

([8], [14, proof of Proposition 3]). More precisely, if $Z=\sum_{k=1}^n x_kE_k$, then

$$\varphi_1=\frac{x_{n-1}}{x_1},\quad \varphi_2=\frac{x_{n-2}}{x_2},\ldots,\quad \varphi_{(n-1)/2}=\frac{x_{(n+1)/2}}{x_{(n-1)/2}}\quad \text{if } n \text{ is odd,}$$

and

$$\varphi_1=\frac{x_{n-1}}{x_1},\quad \varphi_2=\frac{x_{n-2}}{x_2},\ldots,\quad \varphi_{(n-2)/2}=\frac{x_{(n+2)/2}}{x_{(n-2)/2}}\quad \text{if } n \text{ is even.}$$

All the quotients x_i/x_j satisfy Proposition 1, but only those of the form x_{n-j}/x_j satisfy a more general theorem involving affine transformations as described in [7].

Remark 2. For $Z=(z_1,\ldots,z_n)\in\mathbb{C}^n$ the precise form of the coefficients of the linear combination $Z=\sum_{k=1}^n x_kE_k$ is

$$x_k=\frac{1}{n}\sum_{l=1}^n\lambda^{-kl}z_l$$

They are precisely the Fourier descriptors or coefficients of the discrete Fourier transform of Z. This is very handy in our experimental construction.

Remark 3. The function φ_j is well-defined except on

$$\mathcal{N}_j=\left\{(z_1,\ldots,z_n)\in\mathbb{C}^n\ :\ \sum_{k=1}^n\lambda^{jk}z_k=0=\sum_{k=1}^n\lambda^{-jk}z_k\right\}.$$

$\mathcal{N}_j$ is a $(n-2)$-dimensional complex linear subspace with measure zero in $\mathbb{C}^n$. According to Remark 1, $\mathcal{N}_j$ is spanned by $\{E_k\}_{k\neq j,n-j}$.

Remark 4. The level sets $\varphi_j^{-1}(c) = \{(z_1, \ldots, z_n) \in \mathbb{C}^n : \varphi_j(z_1, \ldots, z_n) = c\}$ are $(n-1)$-dimensional complex submanifolds with measure zero in $\mathbb{C}^n$ because every point in $\mathbb{C} \cup \{\infty\}$ is a regular value of φ_j, for any j. This follows from a straightforward calculation which shows that $\dfrac{\partial \varphi_j}{\partial z_k} = 0$ implies $\varphi_j = \lambda^{2jk}$. In this sense, the probability that two randomly chosen polygons Z and W satisfy $\varphi_j(Z) = \varphi_j(W)$ is equal to zero.

2.2 Cyclic shifts and reversed labeling

Proposition 2. *The behavior of φ_j under cyclic shift and reversed labeling is given by the formulas*

$$\varphi_j(z_2, z_3, \ldots, z_n, z_1) = \lambda^{-2j}\varphi_j(z_1, z_2, \ldots, z_n),$$
$$\varphi_j(z_n, z_{n-1}, \ldots, z_2, z_1) = \frac{\lambda^{2j}}{\varphi_j(z_1, z_2, \ldots, z_n)},$$

for all $j = 1, \ldots, \lfloor (n-1)/2 \rfloor$. Hence, if Z and W are relabeling of the same polygon, we have by raising to the nth power the equalities

$$\begin{array}{l} \varphi_j(Z)^n = \varphi_j(W)^n \text{ if the labels have the same orientation, and} \\ \varphi_j(Z)^n = \varphi_j(W)^{-n} \text{ if the labels have the opposite orientation.} \end{array} \tag{1}$$

Proof.

$$\varphi_j(z_2, z_3, \ldots, z_n, z_1) = \frac{\sum_{k=1}^n \lambda^{jk} z_{k+1}}{\sum_{k=1}^n \lambda^{-jk} z_{k+1}} =$$
$$\frac{\lambda^{-j} \sum_{k=1}^n \lambda^{j(k+1)} z_{k+1}}{\lambda^{j} \sum_{k=1}^n \lambda^{-j(k+1)} z_{k+1}} = \lambda^{-2j}\varphi_j(z_1, z_2, \ldots, z_n),$$

where subscript $n+1$ should be taken as 1. Likewise

$$\varphi_j(z_n, z_{n-1}, \ldots, z_2, z_1) = \frac{\sum_{k=1}^n \lambda^{jk} z_{n+1-k}}{\sum_{k=1}^n \lambda^{-jk} z_{n+1-k}} =$$
$$\frac{\lambda^{j(n+1)} \sum_{k=1}^n \lambda^{-j(n+1-k)} z_{n+1-k}}{\lambda^{-j(n+1)} \sum_{k=1}^n \lambda^{j(n+1-k)} z_{n+1-k}} = \frac{\lambda^{2j}}{\varphi_j(z_1, z_2, \ldots, z_n)}.$$

2.3 An index for matching polygons

Exact matching of similar polygons Assume that a collection of different polygons $Z_1, Z_2, \ldots, Z_m$ of n edges is given. By a preprocessing step we compute pairs $(\ell, \varphi_j(Z_\ell)^n)$. Assume that a query polygon W is given and that the objective is to find all the polygons in the collection such that $W = f(Z_\ell)$ for some unknown similarity transformation f. This corresponds to all the polygons such that $\varphi_j(Z_\ell)^n = \varphi_j(W)^n$ or $\varphi_j(Z_\ell)^n = \varphi_j(W)^{-n}$ (Propositions 1 and 2). Since the probability of collision is zero (Remark 4), all the R polygons mapped to $\varphi_j(Z_\ell)^n$ or $\varphi_j(Z_\ell)^{-n}$ should be similar to the query polygon, and can be found in $O(n+R)$ operations, where R is the number of polygons mapped to $\varphi_j(Z_\ell)^n$ or $\varphi_j(Z_\ell)^{-n}$. Notice that the bound in the running time is time independent of m, the size of the collection.

Matching similar polygons under noisy conditions A slightly more general setup is when there is an unknown noise function at the matching. The image of the query polygon is a similarity transformation f plus noise, namely $W = (f(z_1+\Delta z_1), f(z_2+\Delta z_2), \ldots, f(z_n+\Delta z_n))$. In this case, instead of retrieving just the polygons mapped to $\varphi_j(W)^n$ as above, we retrieve all the polygons within a certain Euclidean distance of the image of the query. That is, if r is the tolerance, then we inspect all the polygons such that $|\varphi_j(W)^n - \varphi_j(Z_\ell)^n|^2 \leq r$.

Proposition 3 below gives a precise bound for the tolerated noise.

Proposition 3. *Let $Z = (z_1, \ldots, z_n) \in \mathbb{C}^n \setminus \{0\}$ be a polygon. Consider an integer $j \in \mathbb{Z} \setminus \frac{n}{2}\mathbb{Z}$ such that $\sum_{l=1}^n \lambda^{-jl} z_l \neq 0$. Let ρ be a positive real number such that $n\rho < \mu := |\sum_{l=1}^n \lambda^{-jl} z_l|$. Then for any $\Delta Z = (\Delta z_1, \ldots, \Delta z_n)$ with $|\Delta z_k| < \rho$, $k = 1, \ldots, n$, we have*

$$|\varphi_j(z_1, \ldots, z_n) - \varphi_j(z_1 + \Delta z_1, \ldots, z_n + \Delta z_n)| \leq \frac{2n\rho \sum_{l=1}^n |z_l|}{\mu(\mu - n\rho)}.$$

Proof.

$$\begin{aligned}
&\varphi_j(z_1, \ldots, z_n) - \varphi_j(z_1 + \Delta z_1, \ldots, z_n + \Delta z_n) \\
&= \frac{\sum_{l=1}^n \lambda^{jl} z_l}{\sum_{l=1}^n \lambda^{-jl} z_l} - \frac{\sum_{k=1}^n \lambda^{jk}(z_k + \Delta z_k)}{\sum_{k=1}^n \lambda^{-jk}(z_k + \Delta z_k)} \\
&= \frac{\sum_{l=1}^n \lambda^{jl} z_l}{\sum_{l=1}^n \lambda^{-jl} z_l} - \frac{\sum_{k=1}^n \lambda^{jk} z_k + \sum_{k=1}^n \lambda^{jk} \Delta z_k}{\sum_{k=1}^n \lambda^{-jk} z_k + \sum_{k=1}^n \lambda^{-jk} \Delta z_k} \\
&= \frac{\sum_{l=1}^n \lambda^{jl} z_l \sum_{k=1}^n \lambda^{-jk} \Delta z_k - \sum_{l=1}^n \lambda^{-jl} z_l \sum_{k=1}^n \lambda^{jk} \Delta z_k}{\sum_{l=1}^n \lambda^{-jl} z_l \, (\sum_{k=1}^n \lambda^{-jk} z_k + \sum_{k=1}^n \lambda^{-jk} \Delta z_k)} \qquad (2) \\
&= \frac{\sum_{k,l=1}^n (\lambda^{j(l-k)} - \lambda^{j(k-l)}) z_l \Delta z_k}{\sum_{l=1}^n \lambda^{-jl} z_l \, (\sum_{k=1}^n \lambda^{-jk} z_k + \sum_{k=1}^n \lambda^{-jk} \Delta z_k)} \\
&= \frac{2i \sum_{k,l=1}^n \sin\left(\frac{2\pi j(l-k)}{n}\right) z_l \Delta z_k}{\sum_{l=1}^n \lambda^{-jl} z_l \, (\sum_{k=1}^n \lambda^{-jk} z_k + \sum_{k=1}^n \lambda^{-jk} \Delta z_k)}.
\end{aligned}$$

Hence

$$\begin{aligned}
&|\varphi_j(z_1, \ldots, z_n) - \varphi_j(z_1 + \Delta z_1, \ldots, z_n + \Delta z_n)| \\
&\leq \frac{2 \sum_{k,l=1}^n |z_l| \, |\Delta z_k|}{|\sum_{l=1}^n \lambda^{-jl} z_l| \, (|\sum_{k=1}^n \lambda^{-jk} z_k| - \sum_{k=1}^n |\Delta z_k|)} \qquad (3) \\
&< \frac{2n\rho \sum_{l=1}^n |z_l|}{|\sum_{l=1}^n \lambda^{-jl} z_l| \, \left(\left|\sum_{l=1}^n \lambda^{-jl} z_l\right| - n\rho\right)}.
\end{aligned}$$

Notice that bounds for the noise depend on the frequency response of the polygon. That is, we cannot input a given tolerance and obtain a proper searching radius. The maximum noise allowed is intrinsic to the polygon.

3 Experiments in polygon matching with noise

This section shows the results obtained when we test the algorithm using an extensive set of polygons of four classes. Figure 2 shows some sample polygons of the four classes we considered, respectively convex, star-shaped, Jordan, and general polygons. We have found that the more complex polygons (e.g., general polygons) are more easily distinguished. We discuss it below.

For evaluating our identification method, we generated a large set of polygons in each of the four considered classes, each class with 100,000 polygons. The polygons were generated with integer coordinates in a grid of size 1024×768 using the software kindly provided by Martin Held, according to the heuristics described in [3].

For each polygon Z_ℓ in the collection we computed and stored all $\xi_\ell = \varphi_j(Z_\ell)^n$, $j = 1, \ldots, \lfloor (n-1)/2 \rfloor$, and indexed them using a $2d$-tree. Each polygon was mapped to $\lfloor (n-1)/2 \rfloor$ points in the complex plane. After the mapping, our polygon collection is transformed to a point collection; each point in the collection corresponds to a polygon. The $2d$-tree is used as an inverted index to back-link the points to the original polygons.

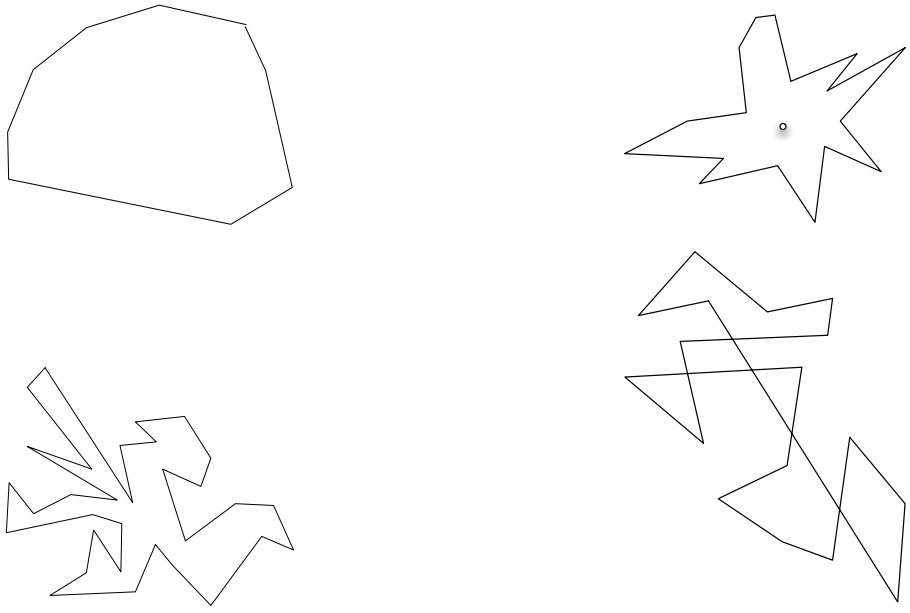

Fig. 2. Example polygons. Convex, star shaped, simple (Jordan) and general (with self-intersections).

For querying we took 1,000 polygons in each collection and randomly perturbed *each* vertex with $\pm r$ pixels in the x and y coordinates, with $1 \leq r \leq 25$.

The resulting polygons were mapped using the same $\lfloor (n-1)/2 \rfloor$ functions. We searched for the nearest neighbor of each one of the resulting points in the corresponding collection. Figure 3 show the recall as a function of the noise (measured in pixels) for various setups. We first notice that using a single, fixed invariant produces a low recall. Remember that the noise bounds depend on the polygon's response to a frequency. Taking all the invariants φ_j for indexing and requiring any one of them to match the nearest neighbor of the query gives excellent results. Remember, by remark 4, that false positive matches are improbable, although due to noise, false positives are possible. For the plot, we considered the $\lfloor (n-1)/2 \rfloor$ candidates, one for each invariant, and checked if this list contains the matching polygon.

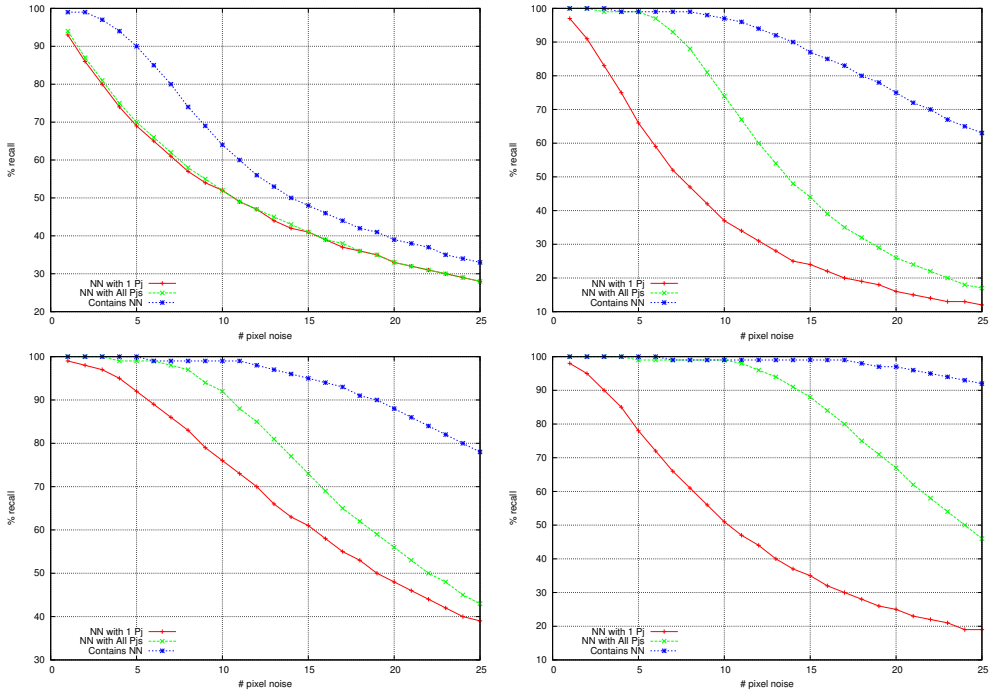

Fig. 3. Searching for noisy polygons. A graph is displayed for each class of polygons. The first class was a set of convex polygons, the second was of star shaped polygons, the third was of simple or Jordan polygons and the fourth class was of polygons without restrictions. The plots show the recall considering *one of the* φ_j (NN with 1Pj), *all of them* (NN with all Pj) and *any of them* (contains NN) respectively. Each point in the plot is the average of 1,000 queries. As the shape is more complex, the identification is easier.

The lowest recall we obtained was for convex polygons, which is consistent with the theoretical results because they have the lowest frequency responses.

On the other hand, general polygons have a higher recall because there will be at least one high-frequency response. As per the cyclic shifts, recall proposition 2, there was no difference in recall when the query was cyclically shifted. We experimented with polygons having between 16 and 32 sides. We saw no significant difference in the plots and only reported the results for 32 sided polygons. The total searching time is negligible, a few milliseconds in a laptop.

3.1 Fixing recall for convex polygons

Observing the disparate results in recall for convex and general polygons and knowing the relationship of the performance and the frequency response of the polygons, we transformed the polygons before storing them. Let Π a random permutation, fixed beforehand. We applied Π to each one of the polygons before computing invariants φ_j, $Z_\Pi = (\Pi(z_1), \Pi(z_2), \ldots, \Pi(z_n))$. At query time, we applied the same permutation to the query. With this change, all the indexed polygons responded equally, obtaining the same recall as generalized polygons.

4 Final remarks

Polygon matching under similarities is a fundamental problem at the core of many applications. In [9] they define the problem of finding the attitude of a spaceship, that is, finding which star appears in the objective of a camera. Stars are codified as polygons, using as vertices the surrounding stars. More precisely, for each star, the k-nearest stars define a polygon, with the center the target star. The algorithm in [9] is akin to brute-force. They compute thousands of perturbations of each polygon to boost the recall, and the corresponding invariants φ_j defined in [7] are stored with rounded decimals. The query polygon is searched for by exact matching. Hence if it coincides with one of the stored perturbations, a match is reported. The above procedure is wasteful; for each star, there will be a blob of points associated.

Using the heuristics defined herein, we report better recall rates than [9] by using the (k)nearest neighbors instead of exact searching and the random permutation before indexing. We store only one complex number, instead of a blob of points, for each star. The above allows to dramatically reduce space usage for a star index for spatial navigation.

We plan to use polygon indexing as a building block for robust point set retrieval under similarities, with applications to image and multimedia retrieval, computer vision, and robotics.

References

1. E.M. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kedem, and J.S.B. Mitchell, *An efficiently computable metric for comparing polygonal shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence. **13** (1991), no. 3, 209–216.

2. M.J. Atallah, C.C. Ribeiro, and S. Lifschitz, *Computing some distance functions between polygons*, Pattern Recognition **24** (1991), no. 8, 775–781.
3. T. Auer and M. Held, *Heuristics for the generation of random polygons*, Proc. 8th Canad. Conf. Comput. Geometry (J.-R. Sack F. Fiala, E. Kranakis, ed.), Carleton University Press, 1996, pp. 38–43.
4. N. Barnes, G. Loy, and D. Shaw, *The regular polygon detector*, Pattern Recognition **43** (2010), no. 3, 592–602.
5. T. Biedl, S. Durocher, and J. Snoeyink, *Reconstructing polygons from scanner data*, Theoretical Computer Science **412** (2011), no. 32, 4161–4172, Algorithms and Computation.
6. K. Buchin, M. Buchin, and C. Wenk, *Computing the Fréchet distance between simple polygons*, Computational Geometry **41** (2008), no. 1–2, 2–20, Special Issue on the 22nd European Workshop on Computational Geometry (EuroCG).
7. Edgar Chávez, Ana C. Chávez-Cáliz, and Jorge L. López-López, *Affine invariants of generalized polygons and matching under affine transformations*, Comput. Geom. **58** (2016), 60–69.
8. J. Chris Ficher, D. Ruoff, and J. Shilleto, *Perpendicular polygons*, Amer. Math. Monthly **92** (1985), no. 1, 23–37.
9. E Antonio Hernández, Miguel A Alonso, Edgar Chávez, David H Covarrubias, and Roberto Conte, *Robust polygon recognition method with similarity invariants applied to star identification*, Advances in Space Research **59** (2017), no. 4, 1095–1111.
10. J. Huang, *A new model for general polygon matching problems*, Precision Engineering **33** (2009), no. 4, 534–541.
11. S. Kaygin and M.M. Bulut, *Shape recognition using attributed string matching with polygon vertices as the primitives*, Pattern Recognition Letters **23** (2002), no. 1–3, 287–294.
12. K. Leszczynski and S. Loose, *A polygon matching algorithm and its applications to verification of radiation field placement in radiotherapy*, International Journal of Bio-Medical Computing **40** (1995), no. 1, 59–67.
13. H. Liu and Z. Wang, *PLDD: Point-lines distance distribution for detection of arbitrary triangles, regular polygons and circles*, Journal of Visual Communication and Image Representation **25** (2014), no. 2, 273–284.
14. J.L. López-López, *The area as a natural pseudo-Hermitian structure on the spaces of plane polygons and curves*, Diff. Geom. Appl. **28** (2010), no. 5, 582–592.
15. E.C. McCreath, *Partial matching of planar polygons under translation and rotation*, Proceedings of the 20th Canadian Conference on Computational Geometry, 2008, pp. 47–50.
16. E. Neustupnỳ, *Polygons in archaeology*, Památky archeologické **87** (1996), 112–136.
17. W.H. Tsai and S.S. Yu, *Attributed string matching with merging for shape recognition*, IEEE Trans. Pattern Anal. Machine Intell. **7** (1985), no. 4, 453–462.
18. R.C. Veltkamp and M. Hagedoorn, *State of the art in shape matching*, Principles of Visual Information Retrieval (M.S. Lew, ed.), Springer, 2001, pp. 87–119.