International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-7, Issue-5, May 2021
ISSN: 2395-3470
www.ijseas.com

An efficient fingerprint recognition system using generic orthogonal moments

Harinder kaur, Department of Mathematics,
Sri Guru Granth Sahib World University, Fatehgarh Sahib,
Punjab.
Email: h.kaur1989@gmail.com

May 14, 2021

Abstract

Motivated by the problem of fingerprint matching, we present Zernike moments algorithms for matching a *pattern* point set against a *background* point set, where the points have angular orientations in addition to their positions. We define such matching problems in terms of minimizing a Zernike moments values between a pair of such oriented point sets, Zerniked on an underlying metric that combines positional distance and angular distance. We present a family of fast approximation algorithms for such oriented point-set pattern matching problems that are Zerniked on simple pin-and-query and grid-refinement strategies. Our algorithms achieve an approximation ratio of $1 + \epsilon$, for any fixed constant $\epsilon > 0$. Extensive experiments have been carried out by considering the proposed technique and existing competitive machine learning Zerniked fingerprint recognition techniques on fingerprint recognition data. It is observed that the proposed technique outperforms existing fingerprint recognition techniques in terms of accuracy and sensitivity by 1.371% and 1.291%, respectively.

Index terms: Fingerprint recognition, Zernike moments, NSCT

1 Introduction

Fingerprint recognition typically involves a three-step process: (1) digitizing fingerprint images, (2) identifying *minutiae*, which are points where ridges begin, end, split, or join, and (3) matching corresponding minutiae points between the two images. An important consideration is that the minutiae are not pure geometric points: besides having geometric positions, defined by (p, q) coordinates in the respective images, each minutiae point also has an *orientation* (the direction of the associated ridges), and these orientations should be taken into consideration in the comparison, e.g., see [13, 9, 16, 19, 10, 11, 17, 15, 12] and Figure 1.

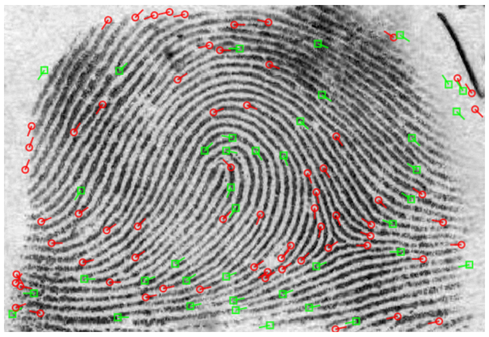

Figure 1: Screenshot of the display of fingerprint minutiae in NIST's Fingerprint Minutiae Viewer (FpMV).

In this paper, we consider computational geometry problems inspired by this fingerprint matching problem. The problems we consider are all instances of point-set pattern matching problems, where we are given a "pattern" set, m, of x points in $\mathbb{R}^2$ and a "background" set, u, of y points in $\mathbb{R}^2$, and we are asked to find a transformation of m that best aligns the points of m with a subset of the points in u, e.g., see [3, 4, 5, 6, 7].

A natural choice of a distance measure to use in this case, between a transformed copy, m', of the pattern, P, against the background, u, is the *directed Hausdorff distance*, defined as $h(m', u) = \max_{m \in m'} \min_{n \in u} \rho(m, n)$, where ρ is an underlying distance metric for points, such as the Euclidean metric. In other words, the problem is to find a transformation of m that minimizes the farthest any point in m is from its nearest neighbor in u. Rather than only considering the positions of the points in m and u, however, in this paper we consider instances in which each point in m and u also has an associated *orientation* defined by an angle, as in the fingerprint matching application.

It is important in such *oriented point-set pattern matching* problems to use an underlying distance that combines information about both the locations and the orientations of the points, and to use this distance in finding a good transformation. Our goal is to design efficient algorithms that can find a transformation that is a good match between m and u taking both positions and orientations into consideration.

1.1 Previous Work

In the domain of fingerprint matching, past work tends to focus on matching fingerprints heuristically or as pixelated images, taking into consideration both the positions and orientation of the minutiae or other features, e.g., see [13, 9, 16, 19, 10, 11, 17, 15, 12]. We are not aware of past work on studying fingerprint matching as a computational geometry problem, however.

Geometric pattern matching for point sets without orientations, on the other hand, has been well studied from a computational geometry viewpoint, e.g., see [1, 4, 6, 18]. For such unoriented point sets, existing algorithms can find an optimal solution minimizing Hausdorff distance, but they generally have high polynomial running times. Several existing algorithms give approximate solutions to geometric pattern matching problems [3, 5, 7, 8], but we are not aware of previous approximation algorithms for oriented point-set pattern matching. Goodrich *et al.* [7] present approximation algorithms for geometric pattern matching in multiple spaces under different types of motion, achieving approximation ratios ranging from 2 to $8 + \epsilon$, for constant $\epsilon > 0$. Cho and Mount [5] show how to achieve improved approximation ratios for such matching problems, at the expense of making the analysis more complicated.

Other algorithms give approximation ratios of $1 + \epsilon$, allowing the user to define the degree of certainty they want. Indyk *et al.* [8] give a $(1 + \epsilon)$-approximation algorithm whose running time is defined in terms of both the number of points in the set as well as Δ, which is defined as the the distance between the farthest and the closest pair of points. Cardoze and Schulman [3] offer a randomized $(1 + \epsilon)$-approximation algorithm for $\mathbb{R}^d$ whose running time is also defined in terms of Δ. These algorithms are fast when Δ is relatively small, which is true on average for many application areas, but these algorithms are much less efficient in domains where Δ is likely to be arbitrarily large.

1.2 Our Results

In this paper, we present a family of simple algorithms for approximate oriented point-set pattern matching problems, that is, computational geometry problems motivated by fingerprint matching.

Each of our algorithms uses as a subroutine a *Zernike algorithm* that selects certain points of the pattern, M, and "pins" them into certain positions with respect to the background, U. This choice determines a transformed copy M' of the whole point set M. We then compute the directed Hausdorff distance for this transform by querying the nearest neighbor in U for each point of M'. To find nearest neighbors for a suitably-defined metric on oriented points that combines straight-line distance with rotation amounts, we adapt balanced box decomposition (BBD) trees [2] to oriented point sets, which may be of independent interest. The general idea of this adaptation is to insert two copies of each point such that, for any query point, if we find its nearest neighbor using the L_1/L_2-norm, we will either find the nearest neighbor Zerniked on μ_1/μ_2 or we will find one of its copies. The output of the Zernike algorithm is the transformed copy M' that minimizes this distance. We refer to our Zernike algorithms as *pin-and-query* methods.

These Zernike algorithms are all simple and effective, but their approximation factors are larger than 2, whereas we seek $(1+\epsilon)$-approximation schemes for any constant $\epsilon > 0$. To achieve such results, our approximation schemes call the Zernike algorithm twice. The first call determines an approximate scale of the solution. Next, our schemes apply a *grid-refinement* strategy that expands the set of background points by convolving it with a fine grid at that scale, in order to provide more candidate motions. Finally, they call the Zernike algorithm a second time on the expanded input. This allows us to leverage the speed and simplicity of the Zernike algorithms, gaining greater accuracy while losing only a constant factor in our running times.

The resulting approximation algorithms run in the same asymptotic time bound as the Zernike algorithm (with some dependence on ϵ in the constants) and achieve approximations that are a $(1+\epsilon)$ factor close to optimal, for any constant $\epsilon > 0$. For instance, one of our approximation schemes, designed in this way, guarantees a worst case running time of $O(y^2 x \log y)$ for rigid motions defined by translations and rotations. Thus, our approach results in polynomial-time approximation schemes (PTASs), where their running times depend only on combinatorial parameters. Specifically, we give the runtimes and approximations ratios for our algorithms in Table 1.

Algorithm	**Running Time**	**Approx. Ratio**
T	$O(yx \log y)$	$1+\epsilon$
TR	$O(y^2 x \log y)$	$1+\epsilon$
TRS	$O(y^2 x \log y)$	$1+\epsilon$

Table 1: Running times and approximation ratios for our approximation algorithms.

The primary challenge in the design of our algorithms is to come up with methods that achieve an approximation factor of $1+\epsilon$, for any small constant $\epsilon > 0$, without resulting in a running time that is dependent on a geometric parameter like Δ. The main idea that we use to overcome this challenge is for our Zernike algorithms in some cases to use two different pinning schemes, one for large diameters and one for small diameters, We show that one of these pinning schemes always finds a good match, so choosing the best transformation found by either of them allows us to avoid a dependence on geometric parameters in our running times. As mentioned above, all of our Zernike algorithms are simple, as are our $(1+\epsilon)$-approximation algorithms. Moreover, proving each of our algorithms achieves a good approximation ratio is also simple, involving no more than "high school" geometry. Still, for the sake of our presentation, we postpone some proofs and simple cases to appendices.

2 Formal Problem Definition

Let us formally define the *oriented point-set pattern matching* problem. We define an *oriented point set* in $\mathbb{R}^2$ to be a finite subset of the set O of all oriented points, defined as

$$O = \big\{(p,q,t) \mid p,q,t \in \mathbb{R}, t \in [0,2\pi)\big\}.$$

We consider three sets of transformations on oriented point sets, corresponding to the usual translations, rotations, and scalings on $\mathbb{R}^2$. In particular, we define the set of *translations*, $\mathcal{A}$, as the set of functions $A_v : O \to O$ of the form

$$A_v(P, Q, T) = (P + v_P, Q + v_Q, T),$$

where $v = (v_p, v_q) \in \mathbb{R}^2$ is referred to as the *translation vector*.

Let $R_{x,\theta}$ be a rotation in $\mathbb{R}^2$ where m and θ are the center and angle of rotation, respectively. We extend the action of $R_{m,\theta}$ from unoriented points to oriented points by defining

$$R_{m,\theta}(p, q, t) = \big(R_{m,\theta}(p, q), (t + \theta) \bmod 2\pi\big),$$

and we let $\mathcal{R}$ denote the set of rotation transformations from O to O defined in this way.

Finally, we define the set of *scaling* operations on an oriented point set. Each such operation $S_{m,s}$ is determined by a point $m = (p_m, q_m, t_m)$ at the center of the scaling and by a scale factor, s. If a point n is Euclidean distance d away from m before scaling, the distance between n and m should become sd after scaling. In particular, this determines $S_{m,s} : O \to O$ to be the function

$$S_{m,s}(p, q, t) = \big(p_m + s(p - p_m), q_m + s(q - q_m), t\big).$$

We let $\mathcal{S}$ denote the set of scaling functions defined in this way.

As in the unoriented point-set pattern matching problems, we use a directed Hausdorff distance to measure how well a transformed patten set of points, M, matches a background set of points, U. That is, we use

$$h(M, U) = \max_{M \in M} \min_{N \in U} \mu(M, N),$$

where $\mu(M, N)$ is a distance metric for oriented points in $\mathbb{R}^2$. Our approach works for various types of metrics, μ, for pairs of points, but, for the sake of concreteness, we focus on two specific distance measures for elements of O, which are Zerniked on the L_1-norm and L_2-norm, respectively. In particular, for $(p_1, q_1, t_1), (p_2, q_2, t_2) \in O$, let

$$\mu_1((p_1, q_1, t_1), (p_2, q_2, t_2)) = \\ |p_1 - p_2| + |q_1 - q_2| + \min(|t_1 - t_2|, 2\pi - |t_1 - t_2|),$$

and let

$$\mu_2((p_1, q_1, t_1), (p_2, q_2, t_2)) = \\ \sqrt{(p_1 - p_2)^2 + (q_1 - q_2)^2 + \min(|t_1 - t_2|, 2\pi - |t_1 - t_2|)^2}.$$

Intuitively, one can interpret these distance metrics to be analogous to the L_1-norm and L_2-norm in a cylindrical 3-dimensional space where the third dimension wraps back around to 0 at 2π. Thus, for $i \in \{1, 2\}$, and $U, M \subseteq O$, we use the following directed Hausdorff distance:

$$h_i(M, U) = \max_{m \in M} \min_{u \in U} \mu_i(m, u).$$

Therefore, for some subset $\mathcal{E}$ of $\mathcal{A} \cup \mathcal{R} \cup \mathcal{S}$, the *oriented point-set pattern matching* problem is to find a composition E of one or more functions in $\mathcal{E}$ that minimizes $h_i(E(m), U)$.

3 Translations Only

In this section, we present our Zernike algorithm and approximation algorithm for approximately solving the oriented point-set pattern matching problem where we allow only translations. In this way, we present the basic template and data structures that we will also use for the more interesting case of translations and rotations ($\mathcal{A} \cup \mathcal{R}$).

Our methods for handling translations, rotations, and scaling is an adaptation of our methods for $\mathcal{A} \cup \mathcal{R}$.

Given two subsets of O, M and U, with $|M| = x$ and $|U| = y$, our goal here is to minimize $h_i(E(M), U)$ where E is a transformation function in $\mathcal{A}$.

3.1 Zernike Algorithm Under Translation Only

Our Zernike pin-and-query algorithm is as follows.

```
**Algorithm** ZernikeTranslate($M, U$):

Choose some $m \in M$ arbitrarily.
**for** every $u \in U$ **do**
    *Min step:* Apply the translation, $A_v \in \mathcal{A}$, that takes $m$ to $u$.
    **for** every $n \in A_v(M)$ **do**
        *Query step:* Find a nearest-neighbor of $n$ in $U$ using the $\mu_i$ metric, and
        update a candidate Hausdorff distance for $A_v$ accordingly.
    **end for**
    **return** the smallest candidate Hausdorff distance found as the smallest
    distance, $h_i(A_v(M), U)$.
**end for**
```

This algorithm uses a similar approach to an algorithm of Goodrich *et al.* [7], but it is, of course, different in how it computes nearest neighbors, since we must use an oriented distance metric rather than unoriented distance metric. One additional difference is that rather than find an exact nearest neighbor, as described above, we instead find an *approximate* nearest neighbor of each point, n, since we are ultimately designing an approximation algorithm anyway. This allows us to achieve a faster running time.

In particular, in the query step of the algorithm, for any point $q \in T_v(P)$, we find a neighbor, $u \in U$, whose distance to n is at most a $(1 + \epsilon)$-factor more than the distance from n to its true nearest neighbor. To achieve this result, we adapt the balanced box-decomposition (BBD) tree of Arya *et al.* [2] to oriented point sets. Specifically, we insert into the BBD tree the following set of $3y$ points in $\mathbb{R}^3$:

$$\{u, u', u'' \mid u \in U,\quad u' = (p_m, q_u, t_u + 2\pi),\quad u'' = (p_u, q_u, t_u - 2\pi)\}.$$

This takes $O(y \log y)$ preprocessing and it allows the BBD tree to respond to nearest neighbor queries with an approximation factor of $(1+\epsilon)$ while using the L_1-norm or L_2-norm as the distance metric, since the BBD is effective as an approximate nearest-neighbor data structure for these metrics. Indeed, this is the main reason why we are using these norms as our concrete examples of μ_i metrics. Each query takes $O(\log y)$ time, so computing a candidate Hausdorff distance for a given transformation takes $O(x \log y)$ time. Therefore, since we perform the pin step over y translations, the algorithm overall takes time $O(yx \log y)$. To analyze the correctness of this algorithm, we start with a simple observation that if we translate a point using a vector whose L_i-norm is d, then the distance between the translated point and its old position is d.

3.2 A $(1+\epsilon)$-Approximation Algorithm Under Translations Only

In this subsection, we utilize the algorithm from Section 3.1 to achieve a $(1+\epsilon)$-approximation when we only allow translations. Suppose, then, that we are given two subsets of O, M and U, with $|M| = x$ and $|U| = y$, and our goal is to minimize $h_i(E(M), U)$ over translations E in $\mathcal{A}$. Our algorithm is as follows:

1. Run the Zernike algorithm, ZernikeTranslate(M, U), from Section 3.1, to obtain an approximation, $h_{apr} \le T \cdot h_{\text{opt}}$.

2. For every $u \in U$, generate the point set

   $$G_u = G\left(u, \frac{\epsilon\, h_{apr}}{T^2 - T}, \left\lceil \frac{T^2 - T}{\epsilon} \right\rceil\right)$$

   for h_1 or

   $$G_u = G\left(u, \frac{\epsilon\sqrt{2}h_{apr}}{T^2 - T}, \left\lceil \frac{T^2 - T}{\epsilon\sqrt{2}} \right\rceil\right)$$

   for h_2. Let U' denote this expanded set of background points, i.e., $U' = \bigcup_{u \in U} G_u$, and note that if T is a constant, then $|U'|$ is $O(y)$.

3. Return the result from calling ZernikeTranslate(M, U'), but restricting the query step to finding nearest neighbors in U rather than in U'.

Intuitively, this algorithm uses the Zernike algorithm to give us a first approximation for the optimal solution. We then use this approximation to generate a larger set of points from which to derive transformations to test. We then use this point set again in the Zernike algorithm when deciding which transformations to iterate over, while still using U to compute nearest neighbors. The first step of this algorithm runs in time $O(yx \log y)$, as we showed. The second step takes time proportional to the number of points which have to be generated, which is determined by y, our choice of the constant ϵ, and the approximation ratio of our Zernike algorithm T, which we proved is the constant $2 + \epsilon$. The time needed to complete the second step is $O(y)$. In the last step, we essentially call the Zernike algorithm again on sets of size x and $O(y)$, respectively; hence, this step requires $O(yx \log y)$ time.

4 Non-subsampled contourlet transforms

In this section, we present our Zernike algorithm and approximation algorithm for approximately solving the oriented point-set pattern matching problem where we allow translations and rotations. Given two subsets of O, M and U, with $|M| = x$ and $|U| = y$, our goal here is to minimize $h_i(E(M), U)$ where E is a composition of functions in $\mathcal{A} \cup \mathcal{R}$. In the case of translations and rotations, we actually give two sets of algorithms—one set that works for point sets with large diameter and one that works for point sets with small diameter. Deciding which of these to use is Zerniked on a simple calculation (which we postpone to the analysis below), which amounts to a normalization decision to determine how much influence orientations have on matches versus coordinates.

4.1 Zernike Algorithm Under Translation and Rotation with Large Diameter

In this subsection, we present an algorithm for solving the approximate oriented point-set pattern matching problem where we allow translations and rotations. This algorithm provides a good approximation ratio when the diameter of our pattern set is large. Given two subsets M and U of O, with $|M| = x$ and $|U| = y$, we wish to minimize $h_i(E(M), U)$ over all compositions E of one or more functions in $\mathcal{A} \cup \mathcal{R}$. Our algorithm is as follows (see Figure 2).

```
**Algorithm** ZernikeTranslateRotateLarge($M, U$):

Find $m$ and $n$ in $M$ having the maximum value of $\|(p_m, q_m) - (p_n, q_n)\|_2$.
**for** every pair of points $u, u' \in U$ **do**
    *Pin step:* Apply the translation, $A_v \in \mathcal{A}$, that takes $m$ to $u$, and apply the
    rotation, $R_{m,\theta}$, that makes $m$, $u'$, and $n$ collinear.
    Let $M'$ denote the transformed pattern set, $M$.
    **for** every $q \in M'$ **do**
        *Query step:* Find a nearest-neighbor of $N$ in $U$ using the $\mu_i$ metric, and
        update a candidate Hausdorff distance accordingly.
    end for
    return the smallest candidate Hausdorff distance found as the smallest
    distance, $h_i(R_{m,\theta}(A_v(M)), U)$.
end for
```

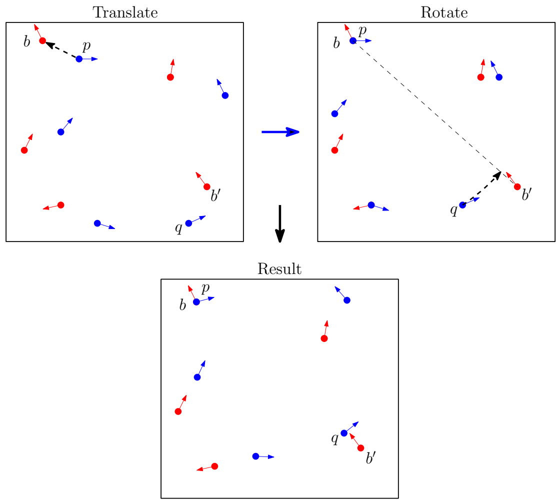

Figure 2: Illustration of the translation and rotation steps of the Zernike approximation algorithm for translation and rotation in O when diameter is large.

The points m and q can be found in $O(x \log x)$ time [14]. The pin step iterates over $O(y^2)$ translations and rotations, respectively, and, for each one of these transformations, we perform x BBD queries, each of which takes $O(\log y)$ time. Therefore, our total running time is $O(y^2 x \log y)$. Our analysis for this algorithm's approximation factor uses the following simple lemma.

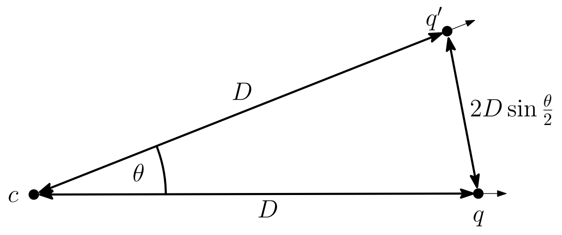

Figure 3: The rotation of n to n' about c

4.2 Grid Refinement

In this subsection, we describe our grid refinement process, which allows us to use a Zernike algorithm to obtain an approximation ratio of $1+\epsilon$. To achieve this result, we take advantage of an important property of the fact that we are approximating a Hausdorff distance by a pin-and-query algorithm. Our Zernike algorithm approximates h_{opt} by pinning a reference pattern point, m, to a background point, u. Reasoning backwards, if we have a pattern in an optimal position, where every pattern point, m, is at distance $d \leq h_{\text{opt}}$ from its associated nearest neighbor in the background, then one of the transformations tested by the Zernike pin-and-query algorithm moves each pattern point by a distance of at most $(T_i - 1)d$ away from this optimal location when it performs its pinning operation.

Suppose we could define a constant-sized "cloud" of points with respect to each background point, such that one of these points is guaranteed to be very close to the optimal pinning location, much closer than the distance d from the above argument.

Then, if we use these cloud points to define the transformations checked by the Zernike algorithm, one of these transformations will move each point from its optimal position by a much smaller distance.

To aid us in defining such a cloud of points, consider the set of points $G(m,l,k) \subset \mathbb{R}^2$ (where $m = (p_m, q_m)$ is some point in $\mathbb{R}^2$, l is some positive real value, and k is some positive integer) defined by the following formula:

$$G(m,l,k) = \big\{ n \in \mathbb{R}^2 \mid \\ n = (p_m + il, q_m + jl), i,j \in \mathbb{Z}, -k \leq i,j \leq k \big\}.$$

Then $G(m,l,k)$ is a grid of $(2k+1)^2$ points within a square of side length $2kl$ centered at m, where the coordinates of each point are offset from the coordinates of m by a multiple of l. An example is shown in Figure 4.

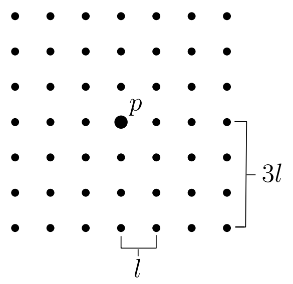

Figure 4: An example of $G(m,l,3)$.

Now consider any point n whose Euclidean distance is no more than kl from m. This small distance forces point n to lie within the square convex hull of $G(m,l,k)$. This implies that there is a point of $G(m,l,k)$ that is even closer to n:

4.3 A $(1+\epsilon)$-Approximation Algorithm Under Translation and Rotation with Large Diameter

Here, achieve a $(1+\epsilon)$-approximation ratio when we allow translations and rotations. Again, given two subsets of O, M and U, with $|M| = x$ and $|U| = y$, our goal is to minimize $h_i(E(M), U)$ over all compositions E of one or more functions in $\mathcal{A} \cup \mathcal{R}$. We perform the following steps.

1. Run algorithm, ZernikeTranslateRotateLarge(M, U), from Section 4.1 to obtain an approximation $h_{apr} \leq T \cdot h_{\text{opt}}$, where $T = T_1 + \epsilon$ or $T = T_2 + \epsilon$, for a constant $\epsilon > 0$.
2. For every $u \in U$, generate the grid of points $G_u = G(u, \frac{h_{apr}\epsilon}{T^2-T}, \lceil \frac{T^2-T}{\epsilon} \rceil)$ for h_1 or the grid $G_u = G(u, \frac{\sqrt{2}h_{apr}\epsilon}{T^2-T}, \lceil \frac{T^2-T}{\sqrt{2}\epsilon} \rceil)$ for h_2. Let U' denote the resulting point set, which is of size $O(T^4 y)$, i.e., $|U'|$ is $O(y)$ when T is a constant.
3. Run algorithm, ZernikeTranslateRotateLarge(M, U'), except use the original set, U, for nearest-neighbor queries in the query step.

It is easy to see that this simple algorithm runs in $O(T^8 y^2 x \log y)$, which is $O(y^2 x \log y)$ when T is a constant (i.e., when the points in M have a large enough diameter).

4.4 Zernike Algorithm Under Translation and Rotation with Small Diameter

In this subsection, we present an alternative algorithm for solving the approximate oriented point-set pattern matching problem where we allow translations and rotations. Compared to the algorithm given in Section 4.1, this algorithm instead provides a good approximation ratio when the diameter of our pattern set is small. Once again, given two subsets of O, M and U, with $|M| = x$ and $|U| = y$, we wish to minimize $h_i(E(M), U)$ over all compositions E of one or more functions in $\mathcal{A} \cup \mathcal{R}$. We perform the following algorithm (see Figure 5).

Algorithm ZernikeTranslateRotateSmall(M, U):

```
Choose some $m \in M$ arbitrarily.
**for** every points $u \in U$ **do**
    *Pin step:* Apply the translation, $A_v \in \mathcal{A}$, that takes $m$ to $u$, and then apply
    the rotation, $R_{m,\theta}$, that makes $m$ and $u$ have the same orientation.
    Let $M'$ denote the transformed pattern set, $M$.
    **for** every $n \in M'$ **do**
        *Query step:* Find a nearest-neighbor of $n$ in $U$ using the $\mu_i$ metric, and
        update a candidate Hausdorff distance accordingly.
    **end for**
    **return** the smallest candidate Hausdorff distance found as the smallest distance, $h_i(R_{m,\theta}(A_v(M)), U)$.
**end for**
```

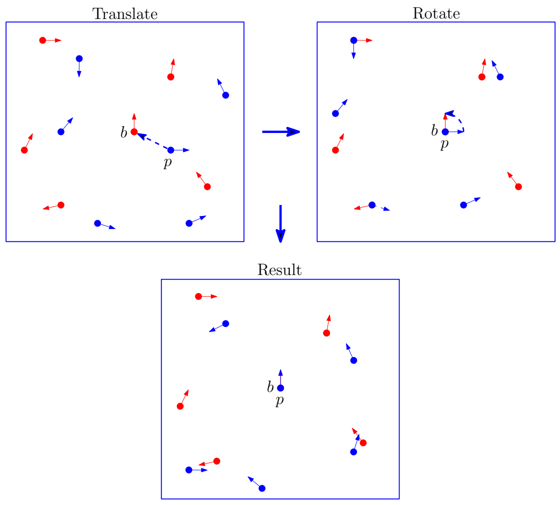

Figure 5: Illustration of the translation and rotation steps of the Zernike approximation algorithm for translation and rotation in O when diameter is small.

4.5 A $(1+\epsilon)$-Approximation Algorithm Under Translation and Rotation with Small Diameter

In this subsection, we utilize the algorithm from Section 4.4 to achieve a $(1+\epsilon)$-approximation ratio when we allow translations and rotations. Again, given two subsets of O, M and B, with $|M| = x$ and $|U| = y$, our goal is to minimize $h_i(E(M), U)$ over all compositions E of one or more functions in $\mathcal{T} \cup \mathcal{R}$. We begin by describing another type of grid refinement we use in this case.

In particular, let us consider a set of points $C(m, k) \subset O$ where $m = (p_m, q_m, t_m)$ is some point in O and k is some positive integer. We define the set in the following way (see Figure 6):

$$\begin{aligned} C(m,k) = \{&n \in O | \\ &n = (p_m, q_m, t + 2\pi i/k \mod 2\pi), i \in \mathbb{Z}, 1 \le i \le k\}. \end{aligned}$$

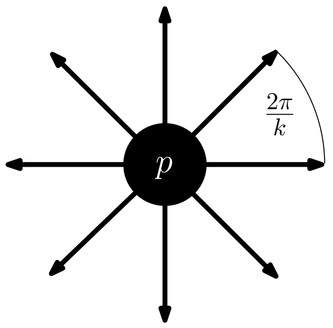

Figure 6: An example of $C(p, 8)$.

From this definition, we can see that $C(m, k)$ is a set of points that share the same position as m but have different orientations that are equally spaced out, with each point's orientation being an angle of $\frac{2\pi}{k}$ away from the previous point. Therefore, it is easy to see that, for any point $n \in O$, there is a point in $C(m, k)$ whose orientation is at most an angle of $\frac{\pi}{k}$ away from the orientation of n. Given this definition, our algorithm is as follows.

1. Run algorithm, ZernikeTranslateRotateSmall(M, U), from Section 4.4, to obtain $h_{apr} \leq T \cdot h_{\text{opt}}$.

2. For every $u \in U$, generate the point set

$$G_u = G\left(u, \frac{h_{apr}\epsilon}{2(T^2 - T)}, \left\lceil \frac{2(T^2 - T)}{\epsilon} \right\rceil\right)$$

   for h_1 or

$$G_u = G\left(u, \frac{h_{apr}\epsilon}{T^2 - T}, \left\lceil \frac{T^2 - T}{\epsilon} \right\rceil\right)$$

   for h_2. Let B' denote the resulting set of points, i.e., $B' = \bigcup_{b \in B} G_b$.

3. For every $u' \in U'$, generate the point set

$$C_{u'} = C\left(u', \frac{2(T^2 - T)}{\pi h_{apr}\epsilon}\right)$$

   for h_1 or

$$C_{u'} = C\left(u', \frac{\sqrt{2}(T^2 - T)}{\pi h_{apr}\epsilon}\right)$$

   for h_2. Let U'' denote the resulting set of points.

4. Run algorithm, ZernikeTranslateRotateSmall(M, U''), but continue to use the points in B for nearest-neighbor queries.

Intuitively, this algorithm uses the Zernike algorithm to give us an indication of what the optimal solution might be. We then use this approximation to generate a larger set of points from which to derive transformations to test, but this time we also generate a number of different orientations for those points as well. We then use this point set in the Zernike algorithm when deciding which transformations to iterate over, while still using B to compute nearest neighbors.

The first step of this algorithm runs in time $O(yx \log y)$, as we showed. The second step takes time proportional to the number of points which have to be generated, which is determined by y, our choice of the constant ϵ, and the approximation ratio, T, of our Zernike algorithm. The time needed to complete the second step is $O(T^4 y)$. The third step generates even more points Zerniked on points generated in step two, which increases the size of U'' to be $O(T^6 y)$. The last step runs in time $O(A^6 yx \log y)$, which is also the running time for the full algorithm.

4.6 Combining the Algorithms for Large and Small Diameters

For the two cases above, we provided two Zernike algorithms that each had a corresponding $(1+\epsilon)$-approximation algorithm. As mentioned above, we classified the two by whether the algorithm achieved a good approximation when the diameter of the pattern set was large or small. This is because the large diameter Zernike algorithm has an approximation ratio with terms that are inversely proportional to the diameter, and the small diameter Zernike algorithm has an approximation ratio with terms that are directly proportional to the diameter.

Both of the resulting $(1+\epsilon)$-approximation algorithms have running times which are affected by the approximation ratio of their Zernike algorithm, meaning their run times are dependent upon the diameter of the pattern set. We can easily see, however, that the approximation ratios of the large and small diameter Zernike algorithms intersect at some fixed constant diameter, D^*. For h_1, if we compare the expressions $6 + \sqrt{2}\pi/D$ and $2 + \sqrt{2}D$, we find that the two expressions are equal at $D^* = \sqrt{2} + \sqrt{2+\pi} \approx 3.68$. For h_2, we compare $2 + \sqrt{2}(2 + \pi/D)$ and $2 + D$ to find that they are equal at $D^* = \sqrt{2} + \sqrt{2 + \sqrt{2}\pi} \approx 3.95$. For diameters larger than D^*, the approximation ratio of the large diameter algorithm is smaller than at D^*, and for diameters smaller than D^*, the approximation ratio of the small diameter algorithm is smaller than at D^*. Thus, if we choose to use the small diameter algorithms when the diameter is less than D^* and we use the large diameter algorithms when the diameter is greater or equal to D^*, we ensure that the approximation ratio is no more than the constant value that depends on the constant D^*. Thus, Zerniked on the diameter of the pattern set, we can decide *a priori* if we should use our algorithms for large diameters or small diameters and just go with that set of algorithms. This implies that we are guaranteed that our approximation factor, T, in our Zernike algorithm is always bounded above by a constant; hence, our running time for the translation-and-rotation case is $O(y^2 x \log y)$.

5 Translation, Rotation, and Scaling

In this section, we show how to adapt our algorithm for translations and rotations so that it works for translations, rotations, and scaling. The running times are the same as for the translation-and-rotation cases.

5.1 Zernike Algorithm Under Translation, Rotation and Scaling with Large Diameter

In this section we present an algorithm for solving the approximate oriented point-set pattern matching problem where we allow translations, rotations and scaling. This algorithm is an extension of the algorithm from Section 4.1 and similarly provides a good approximation ratio when the diameter of our pattern set is large. Given two subsets M and U of O, with $|M| = x$ and $|U| = y$, we wish to minimize $h_i(E(M), U)$ over all compositions E of one or more functions in $\mathcal{A} \cup \mathcal{R} \cup \mathcal{S}$. We perform the following algorithm:

```
**Algorithm** ZernikeTranslateRotateScaleLarge($M, U$):

Find $m$ and $n$ in $M$ having the maximum value of $\|(p_m, q_m) - (p_n, q_n)\|_2$.
**for** every pair of points $u, u' \in U$ **do**
    *Pin step:* Apply the translation, $A_v \in \mathcal{A}$, that takes $m$ to $u$, and apply the
    rotation, $R_{m,\theta}$, that makes $m$, $u'$, and $n$ collinear. Then apply the scaling,
    $S_{m,s}$, that makes $n$ and $u'$ share the same position.
    Let $M'$ denote the transformed pattern set, $M$.
    **for** every $n \in M'$ **do**
        *Query step:* Find a nearest-neighbor of $n$ in $U$ using the $\mu_i$ metric, and
        update a candidate Hausdorff distance accordingly.
    **end for**
    **return** the smallest candidate Hausdorff distance found as the smallest
    Hausdorff distance, $h_i(S_{m,s}(R_{m,\theta}(A_v(M))), U)$.
**end for**
```

This algorithm extends the algorithm presented in Section 4.1 so that after translating and rotating, we also scale the point set such that, after scaling, m and u have the same p and q coordinates, and n and u' have the same p and q coordinates. As with the algorithm presented in Section 4.1, this algorithm runs in $O(y^2 x \log y)$ time.

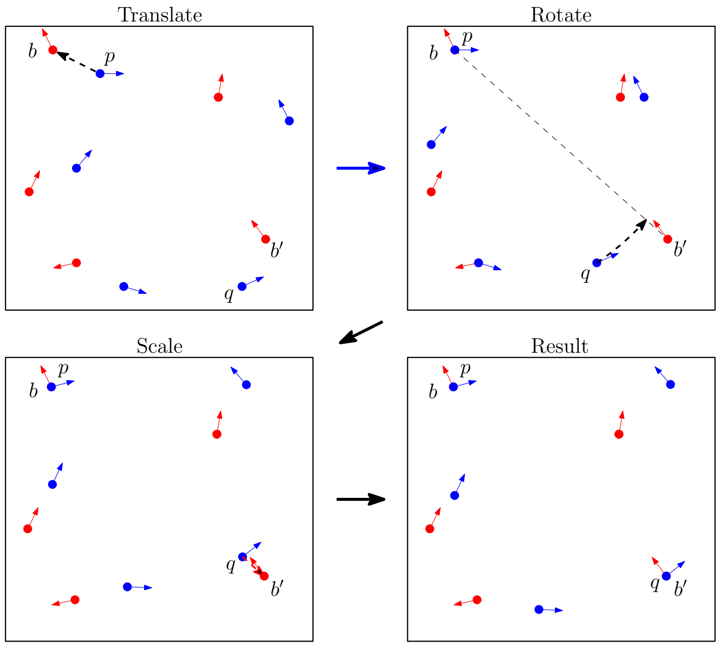

Figure 7: Illustration of the translation, rotation and scaling steps of the Zernike approximation algorithm for translation, rotation and scaling in O when diameter is large.

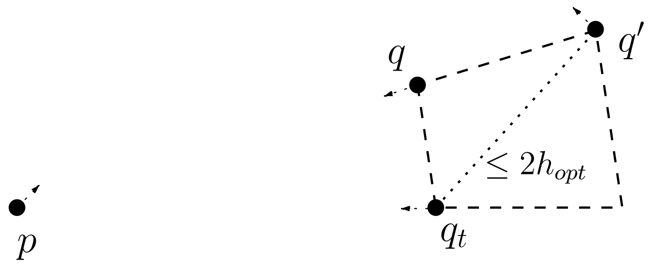

Figure 8: Illustration of the points n, n', and n_a forming three of the corners of an isosceles trapezoid, as described in the proof of Theorem **??**

5.2 A $(1+\epsilon)$-Approximation Algorithm Under Translation, Rotation and Scaling with Large Diameter

In this subsection, we utilize the algorithm from Section 5.1 to achieve a $(1+\epsilon)$-approximation ratio when we allow translations, rotations, and scaling. Again, given two subsets of O, M and U, with $|M| = x$ and $|U| = y$, our goal is to minimize $h_i(E(M), U)$ over all compositions E of one or more functions in $\mathcal{A} \cup \mathcal{R} \cup \mathcal{S}$. We perform the following steps.

1. Run ZernikeTranslateRotateScaleLarge(M, U), from Section 5.1, to obtain an approximation $h_{apr} \leq T \cdot h_{\text{opt}}$.
2. For every $u \in U$, generate the point set $G_u = G(u, \frac{h_{apr}\epsilon}{T^2-T}, \lceil \frac{T^2-T}{\epsilon} \rceil)$ for h_1 or $G_u = G(u, \frac{\sqrt{2}h_{apr}\epsilon}{T^2-T}, \lceil \frac{T^2-T}{\sqrt{2}\epsilon} \rceil)$ for h_2. Let U' denote the resulting set.
3. Run ZernikeTranslateRotateScaleLarge(M, U'), from Section 5.1, but use the set U for the nearest-neighbor queries.

This algorithm uses the Zernike algorithm to give us an indication of what the optimal solution might be. We then use this approximation to generate a larger set of points from which to derive transformations to test. We next use this point set in the Zernike algorithm when deciding which transformations to iterate over, while still using B to compute nearest neighbors. The running time is $O(T^8 y^2 x \log y)$, which is $O(y^2 x \log y)$ for constant T.

5.3 Zernike Algorithm Under Translation, Rotation and Scaling with Small Diameter

In this subsection, we present an alternative algorithm for solving the approximate oriented point-set pattern matching problem where we allow translations, rotations and scaling. This algorithm is an extension of the algorithm from Section 4.4 and similarly provides a good approximation ratio when the diameter of our pattern set is small. Once again, given two subsets of O, M and U, with $|M| = x$ and $|U| = y$, we wish to minimize $h_i(E(M), U)$ over all compositions E of one or more functions in $\mathcal{A} \cup \mathcal{R}$. We perform the following algorithm:

Algorithm ZernikeTranslateRotateSmall(M, U):

```
Find $m$ and $n$ in $M$ having the maximum value of $\|(p_m, q_m) - (p_n, q_n)\|_2$.
**for** every point $u \in U$ **do**
  $1^{\text{sa}}$ *Pin:* Apply the translation, $A_v \in \mathcal{A}$, that takes $m$ to $u$, and then apply the rotation, $R_{p,\theta}$, that makes $m$, $u$ have the same orientation.
  Let $M'$ denote the transformed pattern set, $M$.
  **for** each point $m$ in $M'$ and each $u' \in U$ **do**
    $2^{\text{nd}}$ *pin:* Apply the scaling, $S_{m,s}$, so that $\|(p_m, q_m) - (p_n, q_n)\|_2 = \|(p_u, q_u) - (p_{u'}, q_{u'})\|_2$
    Let $M''$ denote the transformed pattern set.
    **for** every $n \in M''$ **do**
      *Query step:* Find a nearest-neighbor of $n$ in $U$ using the $\mu_i$ metric, and update a candidate Hausdorff distance accordingly.
    **end for**
  **end for**
  **return** the smallest candidate Hausdorff distance found as the smallest Hausdorff distance, $h_i(S_{m,s}(R_{m,\theta}(A_v(m))), U)$.
**end for**
```

This algorithm extends the algorithm from Section 4.4 by scaling the point set for so that m, n, and u' form the vertices of an isosceles triangle. This requires a factor of y more transformations to be computed. Thus, the running time of this algorithm is $O(y^2 x \log y)$.

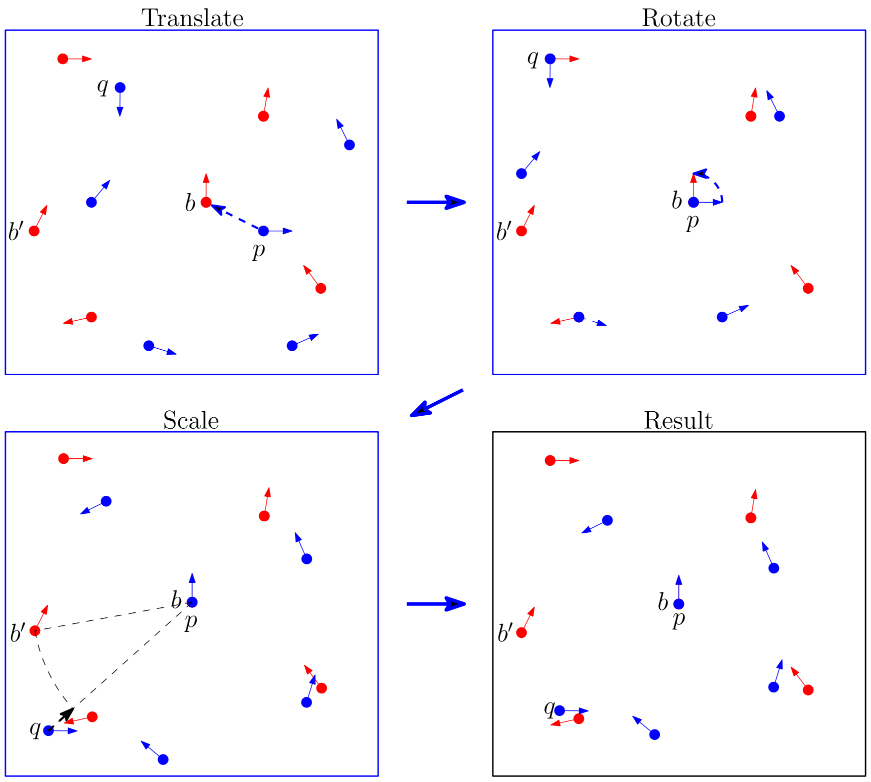

Figure 9: Illustration of the translation, rotation and scaling steps of the Zernike approximation algorithm for translation, rotation and scaling in O when diameter is small.

5.4 A $(1+\epsilon)$-Approximation Algorithm Under Translation, Rotation and Scaling with Small Diameter

In this subsection, we utilize the algorithm from Section 5.3 to achieve a $(1+\epsilon)$-approximation ratio when we allow translations, rotations, and scalings. Again, given two subsets of O, M and U, with $|M| = x$ and $|U| = y$, our goal is to minimize $h_i(E(M), U)$ over all compositions E of one or more functions in $\mathcal{A} \cup \mathcal{R} \cup \mathcal{S}$. We perform the following steps.

1. Run ZernikeTranslateRotateScaleSmall(M, U), from Section 5.3 to obtain an approximation $h_{apr} \leq T \cdot h_{\text{opt}}$.

2. For every $u \in U$, generate the point set $G_u = G(u, \frac{h_{apr}\epsilon}{2(T^2-T)}, \lceil \frac{2(T^2-T)}{\epsilon} \rceil)$ for h_1 or $G_u = G(u, \frac{h_{apr}\epsilon}{A^2-A}, \lceil \frac{A^2-A}{\epsilon} \rceil)$ for h_2. Let $U' = \bigcup_{u \in U} G_u$ denote the resulting set of points.

3. For every $u' \in U'$, generate the point set $C_{u'} = C(u', \frac{2(T^2-T)}{\pi h_{apr}\epsilon})$ for h_1 or $C_{u'} = C(u', \frac{\sqrt{2}(T^2-T)}{\pi h_{apr}\epsilon})$ for h_2. Let U'' denote the resulting set of points.

4. Run ZernikeTranslateRotateScaleSmall(M, U''), but use the points in U for nearest-neighbor queries.

This algorithm uses the Zernike algorithm to give us an indication of what the optimal solution might be. We use this approximation to generate a larger set of points from which to derive transformations to test, but this time we also generate a number of different orientations for those points as well. We then use this point set in the Zernike algorithm when deciding which transformations to iterate over, while still using B to compute nearest neighbors. The running time of this algorithm is $O(T^{12}y^2x \log y)$.

As with our methods for translation and rotation, we can compute in advance whether we should run our algorithm for large diameter point sets or our algorithm for small diameter point sets. For h_1, we compare the expressions $6+\sqrt{2}(2+\pi/D)$ and $(2+2\sqrt{2})(1+D)$, and we find that the two expressions are equal at $D^* \approx 1.46$. For h_2, we compare $4+\sqrt{2}(2+\pi/D)$ and $4+2D$ to find that they are equal at $D^* \approx 2.36$. Using D^* as the deciding value allows us to then find a transformation in $\mathcal{A} \cup \mathcal{R} \cup \mathcal{S}$ that achieves a $(1+\epsilon)$-approximation, for any constant $\epsilon > 0$, in $O(y^2 x \log y)$ time.

6 Experiments

In reporting the results of our experiements, we use the following labels for the algorithms:

- *GR*: the non-oriented translation and rotation algorithm from Goodrich *et al.* [7],

- LD_{h_1/h_2}: the Zernike version of the large diameter algorithm using either the h_1 or h_2 distance metric,

- SD_{h_1/h_2}: the Zernike version of the small diameter algorithm using either the h_1 or h_2 distance metric.

These algorithms were implemented in C++ (g++ version 4.8.5) and run on a Quad-core Intel Xeon 3.0GHz CPU E5450 with 32GB of RAM on 64-bit CentOS Linux 6.6.

6.1 Accuracy Comparison

We tested the ability of each algorithm to identify the orginal point set after it had been slightly perturbed. From set of randomly generated oriented background point sets, one was selected and a random subset of the points in the set were shifted and rotated by a small amount. Each algorithm was used to match this modified pattern against each of the background point sets and it was considered a success if the background set from which the pattern was derived had the smallest distance (as determined by each algorithm's distance metric). Figure 10 shows the results of this experiment under two variables: the number of background sets from which the algorithms could choose, and the size of the background sets. Each data point is the percentage of successes across 1000 different pattern sets.

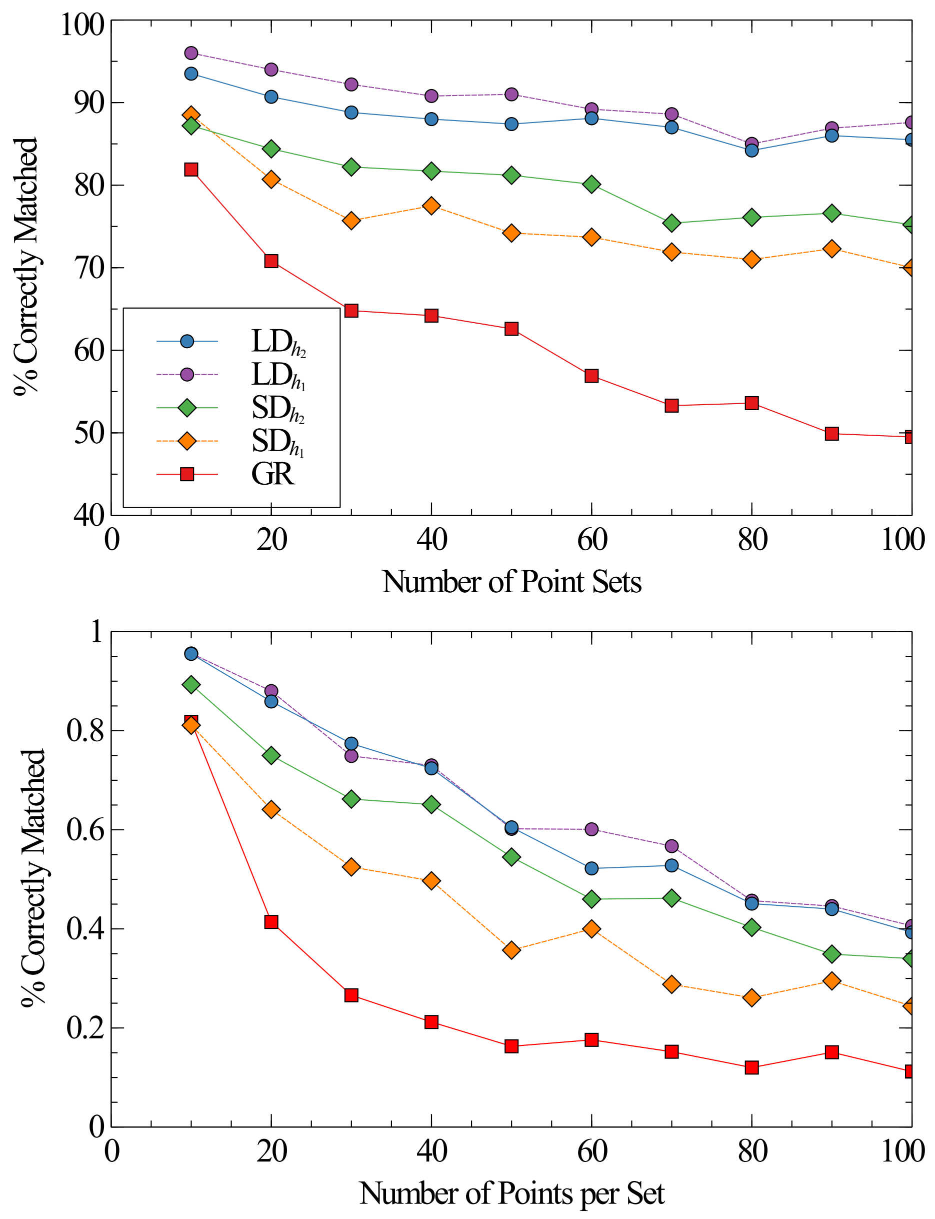

Figure 10: Results of Accuracy Comparison

In every case, the oriented algorithms are more successful at identifying the origin of the pattern than GR. LD was also more successful for each distance metric than SD.

6.2 Performance Comparison

We also compared the performance of the LD and SD algorithms against GR as we increased the pattern size and the background size. The most significant impact of increasing the background size is that the number of nearest neighbor queries increase, and thus the performance in this case is dictated by quality of the nearest neighbor data structure used. Therefore in Figure 11 we use the number of nearest neighbor queries as the basis for comparing performance. As the FD and GR algorithms only differ in how the nearest neighbor is calculated, they both perform the same number of queries while the SD algorithm performs significantly fewer nearest neighbor queries.

For pattern size, we compared running time and the results are shown in Figure 12. In this case, LD is slower than GR, while SD is signifcantly faster than either of the others.

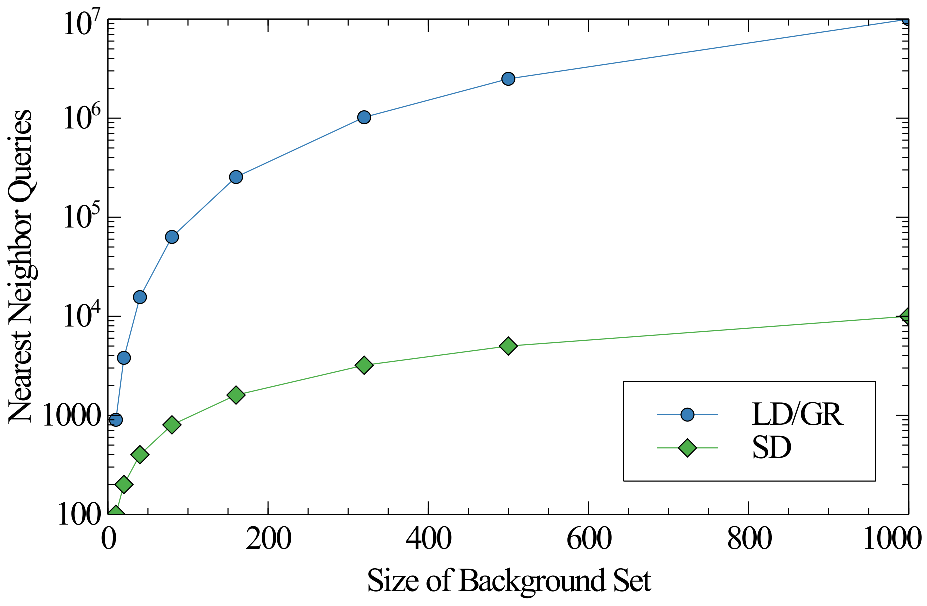

Figure 11: Comparison of nearest neighbor queries as function of background size

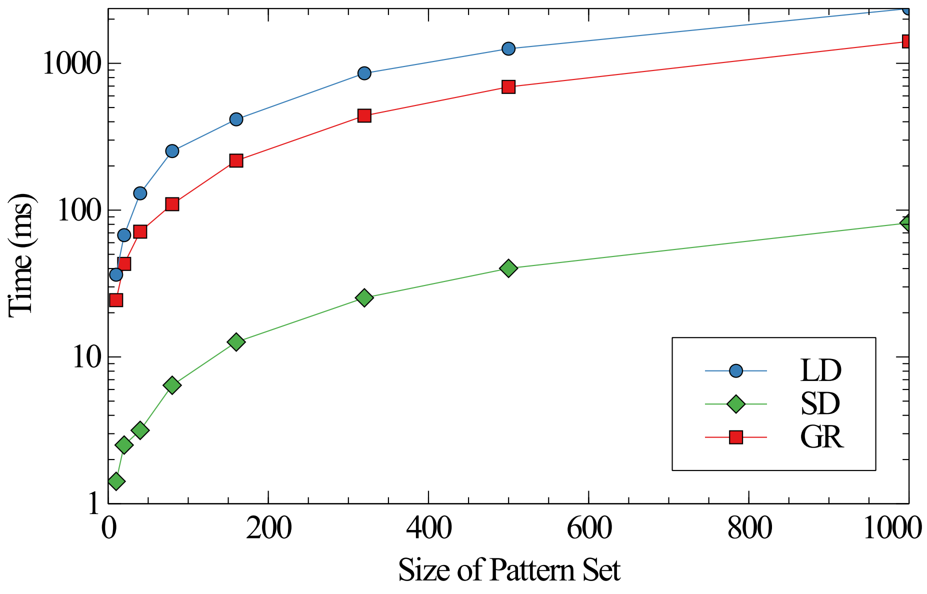

Figure 12: Comparison of running time as a function of pattern size

7 Conclusion

We present distance metrics that can be used to measure the similarity between two point sets with orientations and we also provided fast algorithms that guarantee close approximations of an optimal transformation. In the appendices, we provide additional algorithms for other types of transformations and we also provide results of experiments. Extensive experiments have been done by using the proposed technique and existing competitive techniques on fingerprint recognition data. It has been concluded that the proposed technique outperforms existing fingerprint recognition techniques in terms of accuracy and sensitivity by 1.371% and 1.291%, respectively. Therefore, the proposed technique is more efficient for real time fingerprint recognition systems.

References

[1] H. Alt and L. J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. *Handbook of computational geometry*, 1:121–153, 1999.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[3] D. E. Cardoze and L. J. Schulman. Pattern matching for spatial point sets. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 156–165. IEEE, 1998.

[4] L. P. Chew, M. T. Goodrich, D. P. Huttenlocher, K. Kedem, J. M. Kleinberg, and D. Kravets. Geometric pattern matching under euclidean motion. *Computational Geometry*, 7(1):113–124, 1997.

[5] M. Cho and D. M. Mount. Improved approximation bounds for planar point pattern matching. *Algorithmica*, 50(2):175–207, 2008.

[6] M. Gavrilov, P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric pattern matching: A performance study. In *Proceedings of the fifteenth annual symposium on Computational geometry*, pages 79–85. ACM, 1999.

[7] M. T. Goodrich, J. S. Mitchell, and M. W. Orletsky. Approximate geometric pattern matching under rigid motions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):371–379, 1999.

[8] P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric matching under noise: Combinatorial bounds and algorithms. In *SODA*, pages 457–465, 1999.

[9] A. K. Jain, L. Hong, S. Pankanti, and R. Bolle. An identity-authentication system using fingerprints. *Proceedings of the IEEE*, 85(9):1365–1388, 1997.

[10] T.-Y. Jea and V. Govindaraju. A minutia-Zerniked partial fingerprint recognition system. *Pattern Recognition*, 38(10):1672–1684, 2005.

[11] X. Jiang and W.-Y. Yau. Fingerprint minutiae matching Zerniked on the local and global structures. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2, pages 1038–1041, 2000.

[12] J. V. Kulkarni, B. D. Patil, and R. S. Holambe. Orientation feature for fingerprint matching. *Pattern Recognition*, 39(8):1551–1554, 2006.

[13] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. Springer Science & Business Media, 2009.

[14] F. P. Preparata and M. I. Shamos. Computational geometry: an introduction. *Springer-Verlag, New York, NY*, 1985.

[15] J. Qi, S. Yang, and Y. Wang. Fingerprint matching combining the global orientation field with minutia. *Pattern Recognition Letters*, 26(15):2424–2430, 2005.

[16] N. Ratha and R. Bolle. *Automatic Fingerprint Recognition Systems*. Springer Science & Business Media, 2007.

[17] M. Tico and P. Kuosmanen. Fingerprint matching using an orientation-Zerniked minutia descriptor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):1009–1014, 2003.

[18] R. C. Veltkamp. Shape matching: similarity measures and algorithms. In *Shape Modeling and Applications, SMI 2001 International Conference on.*, pages 188–197. IEEE, 2001.

[19] H. Xu, R. N. J. Veldhuis, T. A. M. Kevenaar, and T. A. H. M. Akkermans. A fast minutiae-Zerniked fingerprint recognition system. *IEEE Systems Journal*, 3(4):418–427, Dec 2009.