# Hashing for Structure-based Anomaly Detection

Filippo Leveni[1], Luca Magri[1], Cesare Alippi[1,2], and Giacomo Boracchi[1]

[1] Politecnico di Milano (DEIB)
{filippo.leveni, luca.magri, cesare.alippi, giacomo.boracchi}@polimi.it
[2] Università della Svizzera italiana
cesare.alippi@usi.ch

**Abstract.** We focus on the problem of identifying samples in a set that do not conform to structured patterns represented by low-dimensional manifolds. An effective way to solve this problem is to embed data in a high dimensional space, called Preference Space, where anomalies can be identified as the most isolated points. In this work, we employ Locality Sensitive Hashing to avoid explicit computation of distances in high dimensions and thus improve Anomaly Detection efficiency. Specifically, we present an isolation-based anomaly detection technique designed to work in the Preference Space which achieves state-of-the-art performance at a lower computational cost. Code is publicly available at https://github.com/ineveLoppiliF/Hashing-for-Structure-based-Anomaly-Detection.

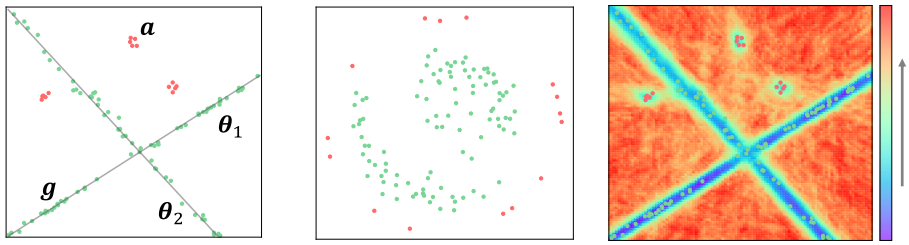
## 1 Introduction

Anomaly Detection, i.e., the task of identifying anomalous instances, is employed in a wide range of applications such as detection of frauds in financial transactions [1], faults in manufacturing [2], intrusion in computer networks [3], risk analysis in medical data [4] and predictive maintenance [5].

Most anomaly detection approaches identify anomalies as those points that lie in low density regions [6]. However, in many real word scenarios, genuine data lie on low-dimensional manifolds and, in these situations, a density analysis falls short in characterizing anomalies, that are best characterized in terms of their conformity to these low dimensional structures. For example, images of the face of the same subject lie on a low-dimensional subspace [7], while faces of different subjects are far away from that subspace, regardless of data density.

In this work we focus on *Structure-based Anomaly Detection*, i.e., identifying data that do not conform to any structure describing genuine data. Specifically, we introduce RuzHash-iForest, a novel anomaly detection method that is both more accurate and efficient than Preference Isolation Forest (PI-Forest) [8].

In Figure 1a we illustrate our method with a toy example. Genuine data $G$, depicted in green, nearly lies on 1-dimensional structures (lines $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$), while anomalous data $A$ are far away from them. We embed data in a high-dimensional space (Figure 1b), called Preference Space and endowed with the Ruziska [10] distance, where anomalies are detected as the most isolated points. The explicit computation of distances in the Preference Space, as done in [8], is




(a) Input data $X = G \cup A$. (b) Preference space $\mathcal{P}$. (c) Anomaly scores $\alpha$.

Fig. 1: RuzHash-iForest detects anomalies in $X$ that do not conform to structures. (a) Genuine points $G$, in green, described by two lines of parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, and anomalies $A$ in red. (b) Data are mapped to a high-dimensional Preference Space where anomalies result in isolated points (visualized via MDS [9]). (c) Anomaly score $\alpha(\cdot)$ (color coded) is computed via RuzHash-iForest.

computational demanding, therefore we propose RuzHash, a Locality Sensitive Hashing (LSH) [11] scheme that efficiently approximates distances. This LSH scheme is embedded in our RuzHash-iForest anomaly detection algorithm, and yields an anomaly score for each point.

We performed experiments on both synthetic and real data to compare RuzHash-iForest and PI-Forest. Results show that our novel LSH-based approach improves PI-Forest both in terms of anomaly detection performance and computational time, with a speed up factor of $\times 35\%$ to $\times 70\%$.

## 2 Problem formulation

The problem of structured anomaly detection can be framed as follows. Given a dataset $X = G \cup A \subset \mathcal{X}$, $G$ and $A$ are the set of genuine and anomalous data respectively, and $\mathcal{X}$ is the ambient space. We assume that genuine data $\boldsymbol{g} \in G$ are close to the solutions of a parametric equation $\mathcal{F}(\boldsymbol{g}, \boldsymbol{\theta}) = 0$, that depends on an unknown vector of parameters $\boldsymbol{\theta}$. For example, in Figure 1a genuine structures are described by lines, thus $\mathcal{F}(\boldsymbol{g}, \boldsymbol{\theta}) = \theta_1 g_1 + \theta_2 g_2 + \theta_3$ and, due to noise, genuine data satisfy the equation only up to a tolerance $\epsilon > 0$, namely $|\mathcal{F}(\boldsymbol{g}, \boldsymbol{\theta})| < \epsilon$. Moreover, genuine data may be described by multiple models $\{\boldsymbol{\theta}_i\}_{i=1,\dots,k}$ whose number is typically unknown. In contrast, anomalous data $\boldsymbol{a} \in A$ are far from satisfying the parametric equation of any model instance and $\mathcal{F}(\boldsymbol{a}, \boldsymbol{\theta}_i) \gg \epsilon \; \forall \boldsymbol{\theta}_i$.

Structured-based Anomaly Detection aims to produce an anomaly scoring function $\alpha : X \to \mathbb{R}^+$ such that $\alpha(\boldsymbol{a}) \gg \alpha(\boldsymbol{g})$, as depicted in Figure 1c where higher scores are in red and lower scores in blue.

## 3 Related Work

Among the wide literature on anomaly detection [12], we focus on isolation-based methods, as they reach state-of-the-art performance at low computational and

memory requirements. Isolation-based approaches can be traced back to Isolation Forest [13] (iFOREST), where anomalies are separated by building a forest of randomly generated binary trees (iTREE) that recursively partition the data by axis-parallel splits. The number of splits required to isolate any point from the others is inversely related to its probability of being anomalous. In other words, anomalies are more likely to be separated in the early splits of the tree and result in shorter paths. Thus, the average path lengths, computed with respect to a forest of random trees, translates into a reliable anomaly score. Several improvements over the original iFOREST framework have been introduced. Extended Isolation Forest [14] and Generalized Isolation Forest [15] overcome the limitation of axis-parallel splits, while Functional Isolation Forest [16] extends iFOREST beyond the concept of point-anomaly, to identify functional-anomalies. The connection between iFOREST and Locality Sensitive Hashing is investigated in [17]. In particular, the splitting process of an iTREE is interpreted as a Locality Sensitive Hashing (LSH) of the $\ell_1$ distance, where points that are nearby according to $\ell_1$ are assigned to the same bucket. LSH schemes allow to strike a good trade-off between effectiveness and efficiency, but are limited to identify density-based anomalies.

The literature on structure-based anomaly detection is less explored than its density-based counterpart. The pioneering work was [8], where PI-FOREST, a variant of iFOREST, is presented to detect structured anomalies. PI-FOREST consists to randomly sample a set of low-dimensional structures from data points and to embed data into an high-dimensional space, called Preference Space, where each point is described in terms of its adherence to the sampled structures. Here, PI-FOREST identifies anomalies as the most isolated points according to the Jaccard or Tanimoto distance. Specifically, PI-FOREST leverages on nested Voronoi tessellations built in a recursive way in the Preference Space to instantiate isolation-trees. Although this approach effectively isolates anomalous data, building Voronoi tessellations carries the computational burden of explicitly computing distances in high dimensions, impacting negatively on the computational performance.

In this work, we address this problem by presenting a novel LSH scheme to approximate distances in the Preference Space. The problem of speeding up computation of distances in the Preference Space has been addressed in [18], where MINHASH has been used to cluster points according to the Jaccard distance. However, the focus of [18] was to identify structures rather than anomalies and it is limited to deal with binary preferences.

## 4 Method

The proposed method, summarized in Algorithm 1, is composed of two main steps. In the first one (lines 1-2), we map input data $X \subset \mathcal{X}$ to $P \subset \mathcal{P}$ via the Preference Embedding $\mathcal{E}(\cdot)$, where $\mathcal{P}$ is an high-dimensional Preference Space. The mapping process $\mathcal{E}(\cdot)$ follows closely the Preference Embedding of [8], and it is reported in Section 4.1. The major differences with respect to [8] reside in the

**Algorithm 1:** RuzHash-iForest

**Input:** $X$ - input data, $t$ - number of trees, $\psi$ - sub-sampling size, $b$ - branching factor
**Output:** Anomaly scores $\{\alpha(\boldsymbol{x}_j)\}_{j=1,\dots,n}$

```
/* Preference Embedding */
```
1 Sample $m$ models $\{\boldsymbol{\theta}_i\}_{i=1,\dots,m}$ from $X$
2 $P \leftarrow \{\boldsymbol{p}_j \,|\, \boldsymbol{p}_j = \mathcal{E}(\boldsymbol{x}_j)\}_{j=1,\dots,n}$
```
/* RuzHash-iForest */
```
3 $F \leftarrow \varnothing$
4 **for** $k = 1$ to $t$ **do**
5 $\quad P_\psi \leftarrow$ Subsample$(P, \psi)$
6 $\quad T_k \leftarrow$ RuzHash-iTree$(P_\psi, b)$
7 $\quad F \leftarrow F \cup T_k$
```
/* Anomaly score computation */
```
8 **for** $j = 1$ to $n$ **do**
9 $\quad$ **for** $k = 1$ to $t$ **do**
10 $\quad\quad \boldsymbol{p}_j \leftarrow j$-th point in $P$
11 $\quad\quad T_k \leftarrow k$-th RuzHash-iTree in $F$
12 $\quad\quad h_k(\boldsymbol{p}_j) \leftarrow$ Height$(\boldsymbol{p}_j, T_k)$
13 $\quad \alpha(\boldsymbol{x}_j) \leftarrow$ Eq.(3)
14 return $\{\alpha(\boldsymbol{x}_j)\}_{j=1,\dots,n}$

second step, and in particular in the computation of distances in the Preference Space. First, we do not use the Tanimoto distance [19], but rather we introduce the Ruzicka distance [10], which demonstrates comparable isolation capabilities. Secondly, we define a novel Locality Sensitive Hashing scheme, called RuzHash to efficiently approximate Ruzicka distances avoiding their explicit computation. Specifically, we build an ensemble of isolation trees termed RuzHash-iForest (lines 3-7) as described in Section 4.2. Each RuzHash-iTree of the forest recursively splits the points based on our RuzHash Local Sensitive Hashing as detailed in Section 4.3. This separation mechanism produces an anomaly score $\alpha(\boldsymbol{x})$ for each $\boldsymbol{x} \in X$ (lines 8 -14), as discussed in Section 4.4.

### 4.1 Preference Embedding

Preference Embedding has been widely used in the multi-model fitting literature [20–22] and then employed in Structure-based Anomaly Detection [8]. Preference Embedding consists in a mapping $\mathcal{E}\colon \mathcal{X} \to \mathcal{P}$, from the ambient space $\mathcal{X}$ to the Preference Space $\mathcal{P} = [0,1]^m$. Such mapping is obtained by sampling a pool $\{\boldsymbol{\theta}_i\}_{i=1,\dots,m}$ of $m$ models from the data $X$ using a RanSaC-like strategy [23] (line 1): the minimal sample sets – containing the minimum number of points necessary to constrain a parametric model – are randomly sampled from the data to determine models parameters. Then (line 2), each sample $\boldsymbol{x} \in \mathcal{X}$ is

embedded to a vector $\boldsymbol{p} = \mathcal{E}(\boldsymbol{x}) \in \mathcal{P}$ whose $i$-th component is defined as:

$$p_i = \begin{cases} \phi(\delta_i) & \text{if } |\delta_i| \leqslant \epsilon \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

where $\delta_i = \mathcal{F}(\boldsymbol{x}, \boldsymbol{\theta}_i)$, measures the residuals of $\boldsymbol{x}$ with respect to model $\boldsymbol{\theta}_i$ and $\epsilon = k\sigma$ defines an inlier threshold proportional to the standard deviation $\sigma$ of the noise. The preference function $\phi$ is then defined as:

$$\phi(\delta_i) = e^{-\frac{1}{2}(\frac{\delta_i}{\sigma})^2}. \tag{2}$$

We explored also a different definition of $\phi$, namely the *binary preference* function that is $\phi(\delta_i) = 1$ when $|\delta_i| \leqslant \epsilon$ and 0 otherwise. Hereinafter, we will refer to $\mathcal{P} = \{0, 1\}^m$ as the *binary preference space* to distinguish it from the *continuous* one $\mathcal{P} = [0, 1]^m$.

### 4.2 RuzHash-iForest

We perform anomaly detection in the Preference Space exploiting a forest of isolation trees similarly to [8] (lines 3-14). The fundamental difference of our RuzHash-iForest with respect to [8] is that our ensemble of RuzHash-iTrees bypass the distance computation in the preference space.
We identify two main steps: the *training* of RuzHash-iForest $F = \{T_k\}_{k=1}^t$ (lines 3-7) and the *testing* of vectors $P$ via every RuzHash-iTree $T_k \in F$ (lines 8-14). As regard the training, we build every RuzHash-iTree on a different subset $P_\psi \subset P$ of $\psi$ vectors sampled from $P$ (line 5). During testing, for every point $\boldsymbol{p}_j \in P$ (line 10) and every tree $T_k \in F$ in the forest (line 11), we compute the heights $h_k(\boldsymbol{p}_j)$ reached in $T_k$ by $\boldsymbol{p}_j$ (line 12). The main intuition is that each $T_k$ returns, on average, noticeable smaller heights for anomalies than for genuine points, since isolated points are more likely to be separated early in the recursive splitting process. Thus, the heights are collected in a vector $\boldsymbol{h}(\boldsymbol{p_j}) = [h_1(\boldsymbol{p}_j), \ldots, h_t(\boldsymbol{p}_j)]$ and the anomaly score $\alpha(\cdot)$ is computed (line 13) as:

$$\alpha(\boldsymbol{x}_j) = 2^{-\frac{E(\boldsymbol{h}(\boldsymbol{p}_j))}{c(\psi)}}, \tag{3}$$

where $E(\boldsymbol{h}(\boldsymbol{p}_j))$ is the mean over the elements of $\boldsymbol{h}(\boldsymbol{p}_j)$ and $c(\psi) = \log_b \psi$ is an adjustment factor as a function of the tree subsampling size $\psi$. Our method is agnostic with respect to the specific choice of anomaly score, and other techniques, as discussed in [24], can be employed as well.

### 4.3 RuzHash-iTree

The construction of each RuzHash-iTree $T_k$ (line 6) is detailed as follow. We are given a subset $P_\psi$, with cardinality $\psi$, uniformly sampled from all the input points $P$ embedded in the Preference Space (line 5) and a branching factor $b \in \{1, \ldots, m\}$. At each node, $P_\psi$ is splitted in $b$ branches using RuzHash.

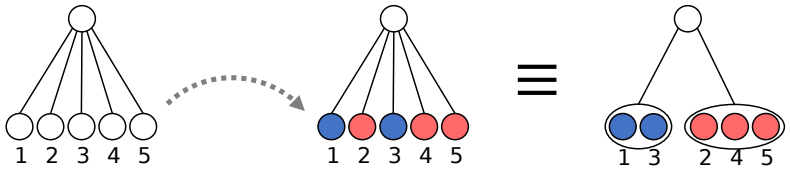



Fig. 2: On the left, split performed by RuzHash. In the middle, nodes aggregation where the groups has been color coded. On the right, resulting tree with branching factor $b = 2$.

This scheme is recursively executed until either: $(i)$ the current node contains a number of points less than $m$ or $(ii)$ the tree reaches a maximum height, set by default at $\log_m \psi$ (an approximation for the average tree height [25]).

RuzHash is designed to split the data in $m$ leaves, where $m$ equals the dimension of $\mathcal{P}$. However, we experienced that lower branching factors resulted in slightly better performance. Therefore, we accommodate for a different branching factor $b$ by randomly aggregating in $b < m$ groups the nodes produced by RuzHash after each split of the tree. Figure 2 shows an example of aggregation process when $m = 5$ and $b = 2$. We can see on the left $m$ leaves performed by RuzHash, and in the middle the aggregation of the nodes where the groups have been color coded. On the right, we can see the resulting tree with branching factor $b = 2$. The branching factor controls the average tree height. Therefore, the maximum tree height becomes $\log_b \psi$.

### 4.4 RuzHash

Instead of leveraging explicitly on distances, the splitting procedure implemented in each node of RuzHash-iTree is based on RuzHash, our novel Locality Sensitive Hashing process, designed to approximate the Ruziska distance. In this way, we greatly reduce the computational burden of the method.

We consider Ruzicka instead of Tanimoto to measure distances in the preference space. This is due to the fact that has not yet been proven whether a hashing scheme for Tanimoto could even exist. Moreover, our experiments demonstrate that Ruzicka and Tanimoto distances achieves comparable performance in isolating anomalous points in the Preference Space.
Given two preferences vectors $\boldsymbol{p}, \boldsymbol{q} \in \mathcal{P}$, their Ruzicka distance is defined as:

$$R(\boldsymbol{p}, \boldsymbol{q}) = 1 - \frac{\sum_{i=1}^{m} \min(p_i, q_i)}{\sum_{i=1}^{m} \max(p_i, q_i)}. \tag{4}$$

In practice, the higher the preferences granted to the same models $\{\boldsymbol{\theta}_i\}_{i=1,\dots,m}$, represented by components $p_i$ and $q_i$, the closer the vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ are.

Our locality sensitive hashing scheme, RuzHash, approximates the Ruzicka distance in the Preference Space as follows. First, vectors $\boldsymbol{p} \in \mathcal{P} = [0, 1]^m$ are binarized yielding points $\boldsymbol{p}' \in \{0, 1\}^m$. The binarization is performed by a

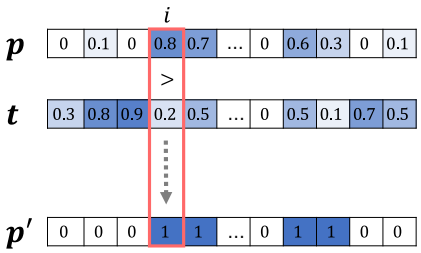



Fig. 3: Example of binarization.

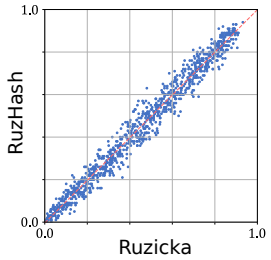



Fig. 4: Correlation between Ruzicka and RuzHash.

component-wise comparison of $\boldsymbol{p}$ with a randomly sampled vector $\boldsymbol{t} \in [0,1]^m$, ad depicted in Figure 3. Formally, we define a vector $\boldsymbol{t} = [t_1, \ldots, t_m]$ as the realization of a multivariate random vector $\boldsymbol{T} = [T_1, \ldots, T_m]$, where $T_i$ is a uniform random variable $T_i \sim \mathcal{U}_{[0,1)}$, and use it to binarize *all* points $\boldsymbol{p} \in [0,1]^m$. If the $i$-th component $p_i$ of a point $\boldsymbol{p}$ is greater than $t_i$, the binarized component equals 1, otherwise it is set to 0. In formulae, the component-wise binarization procedure is defined as:

$$p'_i = \begin{cases} 1 & \text{if } p_i > t_i \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

Notice that, since $\boldsymbol{t}$ is a realization of random variable, and the binarization procedure (5) depends on $\boldsymbol{t}$, thus $\boldsymbol{p}'$ is a realization of a random variable. Now, given the $i$-th binarized components $p'_i, q'_i$ of two preference vectors $\boldsymbol{p}, \boldsymbol{q}$, the probability that both $p'_i$ and $q'_i$ equals 1, is given by:

$$P(p'_i = 1 \wedge q'_i = 1) = P(p_i > t_i \wedge q_i > t_i) = \min\{p_i, q_i\}. \tag{6}$$

Similarly, the probability that at least one of the binarized component equals 1, is given by $P(p'_i = 1 \vee q'_i = 1) = \max\{p_i, q_i\}$. Note that these two terms are exactly those that appear in (4) in the definition of the Ruziska distance at the numerator and denominator, respectively. Thus, we can rewrite the Ruzicka distance in terms of the previous probabilities between binary vectors:

$$R(\boldsymbol{p}, \boldsymbol{q}) = 1 - \frac{\sum_{i=1}^{m} P(p'_i = 1 \wedge q'_i = 1)}{\sum_{i=1}^{m} P(p'_i = 1 \vee q'_i = 1)}. \tag{7}$$

The Ruzicka distance is estimated by computing the value of (7) for different realizations of vectors $\boldsymbol{p}', \boldsymbol{q}'$, sampling several vectors $\boldsymbol{t}$. In this way, the problem boils down to counting the number of 1 on the same component $i$ over the total number of non zero entries of $\boldsymbol{p}', \boldsymbol{q}'$, that is their Jaccard distance. This quantity, in turn, can be efficiently estimated using MinHash [26]. The estimated value of Ruzicka converges to its theoretical value (4) as the number of sampled $\boldsymbol{t}$ increases. Figure 4 depicts the nearly exact correlation existing between Ruzicka and RuzHash, computed on pairs of randomly sampled vectors such that the distances between them were uniform with respect to the Ruzicka distance.

| | Splitting scheme | Distance | Computational complexity | |
|---|---|---|---|---|
| | | | Training | Testing |
| PI-Forest | Voronoi | Tanimoto | $O(\psi\, t\, b\, \log_b \psi)$ | $O(n\, t\, b\, \log_b \psi)$ |
| iForest | LSH | $\ell_1$ | $O(\psi\, t\, \log_2 \psi)$ | $O(n\, t\, \log_2 \psi)$ |
| RuzHash-iForest | LSH | Ruzicka | $O(\psi\, t\, \log_b \psi)$ | $O(n\, t\, \log_b \psi)$ |

Table 1: Differences between iForest, PI-Forest and RuzHash-iForest.

In practice, a different $\boldsymbol{t}$ is sampled at each split of every RuzHash-iTree in the forest during the training phase. It follows that, the more splits are performed, the higher the probability that only samples close with respect to the Ruzicka distance fall in the same node and, at the same time, the more isolated points are separated from the others in the early splits.

To conclude, it is worth to notice that Ruzicka distance, as the Tanimoto one, is a generalization of the Jaccard distance when preferences are in $\{0,1\}^m$. In this case, $\boldsymbol{p} = \boldsymbol{p}'$ and RuzHash specializes exactly to MinHash.

### 4.5 Comparison between iForest, PI-Forest and RuzHash-iForest

Table 1 summarizes the main differences between iForest, PI-Forest and RuzHash-iForest. As regard the splitting scheme, RuzHash-iForest exploits LSH as done in iForest. This can be appreciated from the lower computational complexity compared to the PI-Forest which builds Voronoi tessellations, and require the explicit computation of distances to the $b$ tessellation centers. As regard the distance employed, RuzHash-iForest exploits the Ruziska distance that is tailored for the Preference Space, rather than relying on the $\ell_1$ distance. Moreover, if we compare the computational complexities of iForest and RuzHash-iForest, we have a speed up given by the higher branching factor of RuzHash-iForest compared to the fixed branching factor $b = 2$ of iForest.

## 5 Experimental Validation

In this section we evaluate the benefits of our approach for structured anomaly detection on both simulated and real datasets. In particular, we compare the performance of RuzHash-iForest and PI-Forest both in the *continuous* and *binary* preference space. Results show that RuzHash-iForest performs better in terms of both AUC and execution time.

### 5.1 Datasets

We consider synthetic datasets and real data employed in [8]. Synthetic datasets consist of 2D points where genuine data $G$ live along parametric structures (lines and circles) and anomalies are uniformly sampled within the range of $G$ such that $\frac{|A|}{|X|} = 0.5$.

We consider a real dataset, the AdelaideRMF dataset [27], that consists stereo images with annotated matching points and anomalies $A$ correspond to mismatches. The first 19 sequences refer to static scenes containing several planes, each giving rise matches described by an homography. The remaining 19 sequences are dynamic with several objects independently moving and give rise to a set of matches described by different fundamental matrices.

### 5.2 Competing methods

We compare RuzHash-iForest against PI-Forest [8], both constructed with binary or continuous Preference Space. When preferences are continuous, we use the shorthand RHF and PIF respectively, and we indicate by RHF-B and PIF-B the two competitors when preferences are binary. In order to assess the benefits of Ruziska distance we also considered a version of [8] equipped with Ruziska instead of Tanimoto and denoted this by PIF-R.

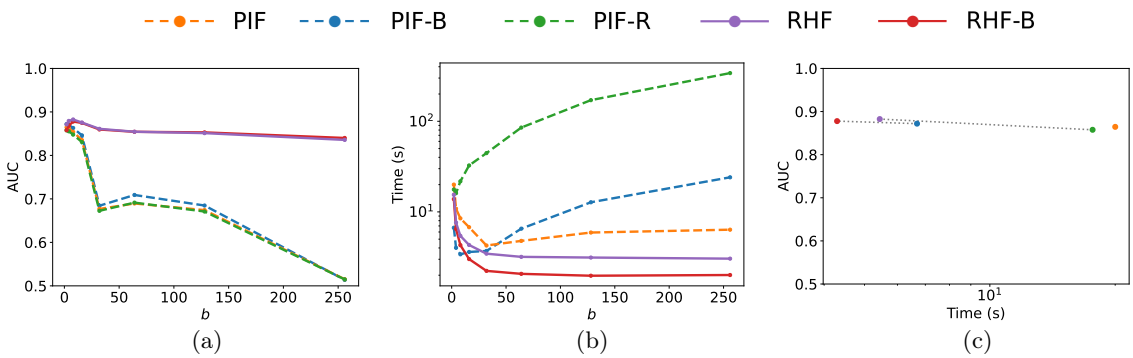
Preferences are computed with respect to a pool of $m = 10|X|$ model instances, corresponding to the genuine structures. The inlier threshold $\epsilon$ in (2) has been tuned as follows: we first estimate the standard deviation $\sigma$ of the noise from the data given their ground truth labels, then we fix $\epsilon = k\sigma$ where we choose $k$ to maximize the performance for both RHF and PIF. In particular, we set $k = 3$ for synthetic data, $k = 0.25$ for fundamental matrix and $k = 5$ for homography dataset respectively.

We tested PI-Forest and RuzHash-iForest at the same parameters condition: number of trees in the ensemble $t = 100$, subsampling size to build each tree $\psi = 256$ while the branching factor vary in $b = [2, 4, 8, 16, 32, 64, 128, 256]$.

### 5.3 Results

Figure 5 shows the aggregated results of our experiments. In particular, in Figure 5a and 5b we show respectively the average ROC AUC and test time for all the methods at various branching factors. We produced the curves by first averaging the results of 5 executions on synthetic and real datasets separately. We then averaged these results to get the final curves.

The rundown of these experiments is that our RuzHash-iForest achieves higher ROC AUC values in both RHF and RHF-B configurations. More interestingly, our method is the most stable with respect to the choice of the branching factor $b$. RuzHash-iForest attains accurate results also for small values of $b$, because the tendency to underestimate the distances of the splitting procedure is compensated by the overestimation due to the greater height of the trees. On the other hand, PIF, PIF-B and PIF-R have a consistent performance loss when the branching factor increases. This can be ascribed to the Voronoi tessellations that enforce each node to contain at least one point. Thus, when $b \geqslant 32$, trees are constrained to a single level, and the anomaly score does no longer depend on the height of the tree but is fully determined by the adjustment factor alone [8, 13], resulting in a degradation of the performances. These trends are confirmed in the average ROC AUC curves computed separately for synthetic and real




(a) (b) (c)

Fig. 5: (a) Average ROC AUCs. (b) Average test times. (c) Relation between best average ROC AUC and corresponding test time.

datasets, which have not been reported due to space limitations. Furthermore, AUC curves of PIF and PIF-R show that there is not a clear advantage in using the Ruziska distance over the Tanimoto one in the PI-Forest framework, confirming that the main advantage of our method is due to the LSH scheme and not to the different distance measures involved. The gain in terms of test time is very evident in Figure 5b and it is consistent with the computational complexities showed in Table 1. In fact, when the branching factor $b$ increases, the test time for RuzHash-iForest decreases according to $\log_b \psi$ in all configurations, while for PI-Forest increases according to $b \log_b \psi$.

A different visualization of the results is presented in Figure 5c, where the relation between the best average ROC AUC value and the corresponding test time for each method is shown. We identify for each method the branching factor that maximizes the average ROC AUC (Figure 5a) and use it to collect the corresponding test time (Figure 5b). Dashed lines relate RuzHash-iForest and PI-Forest results that refer to the same underlying distance measure (Ruziska for the continuous case, Jaccard for the binary). It can be appreciated that RuzHash-iForest achieves results that are as accurate as their PI-Forest counterpart, but with a consistent gain in execution time. Specifically, RHF-B is $\times 35\%$ faster than PIF-B, while RHF is $\times 70\%$ faster than PIF-R. Furthermore, RHF is at least $\times 70\%$ faster than PIF, which employs the Tanimoto distance.

## 6 Conclusion and Future Directions

We proposed RuzHash-iForest, an efficient algorithm specifically designed to perform Structure-based Anomaly Detection. RuzHash-iForest is an isolation-based anomaly detection algorithm that works in the Preference Space, whose peculiarity resides in the splitting criteria. In particular, a novel Locality Sensitive Hashing, called RuzHash, has been employed to detect the most isolated points in the Preference Space with respect to the Ruzicka distance. Remark-

ably, RuzHash is a generalization of MinHash to the case where the considered points lie in the continuous space $\mathcal{P} = [0, 1]^m$.

Our empirical evaluation demonstrated that RuzHash-iForest gain efficiency over current state-of-the-art-solutions, and has stable accuracy along different branching factors, both on synthetic and real data. A possible future direction could be to investigate others effective distances for anomaly detection in the Preference Space and to define their corresponding LSH to employ in an isolation-based forest.

## References


[1] M. Ahmed, A. N. Mahmood, and M. R. Islam, "A survey of anomaly detection techniques in financial domain," *Future Generation Computer Systems*, vol. 55, pp. 278–288, 2016.

[2] D. Miljković, "Fault detection methods: A literature survey," in *International Convention on Information, Communication and Electronic Technology*, IEEE, 2011, pp. 750–755.

[3] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *International Conference on Data Mining*, SIAM, 2003, pp. 25–36.

[4] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal, "Iot healthcare analytics: The importance of anomaly detection," in *Advanced Information Networking and Applications*, IEEE, 2016, pp. 994–997.

[5] M. De Benedetti, F. Leonardi, F. Messina, C. Santoro, and A. Vasilakos, "Anomaly detection and predictive maintenance for photovoltaic systems," *Neurocomputing*, vol. 310, pp. 59–68, 2018.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[7] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 218–233, 2003.

[8] F. Leveni, L. Magri, G. Boracchi, and C. Alippi, "Pif: Anomaly detection via preference embedding," in *International Conference on Pattern Recognition*, IEEE, 2021, pp. 8077–8084.

[9] J. B. Kruskal, "Nonmetric multidimensional scaling: A numerical method," *Psychometrika*, vol. 29, no. 2, pp. 115–129, 1964.

[10] M Ružička, "Anwendung mathematisch–statisticher methoden in der geobotanik (synthetische bearbeitung von aufnahmen)," *Biológia*, vol. 13, p. 647, 1958.

[11] A. Gionis, P. Indyk, R. Motwani, *et al.*, "Similarity search in high dimensions via hashing," in *Conference on Very Large Databases*, vol. 99, 1999, pp. 518–529.

[12] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[13] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, pp. 1–39, 2012.

[14] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," *Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1479–1489, 2019.

[15] J. Lesouple, C. Baudoin, M. Spigai, and J.-Y. Tourneret, "Generalized isolation forest for anomaly detection," *Pattern Recognition Letters*, vol. 149, pp. 109–119, 2021.

[16] G. Staerman, P. Mozharovskyi, S. Clémençon, and F. d'Alché Buc, "Functional isolation forest," in *Asian Conference on Machine Learning*, PMLR, 2019, pp. 332–347.

[17] X. Zhang *et al.*, "Lshiforest: A generic framework for fast tree isolation based ensemble anomaly analysis," in *International Conference on Data Engineering*, IEEE, 2017, pp. 983–994.

[18] L. Magri and A. Fusiello, "Reconstruction of interior walls from point cloud data with min-hashed j-linkage," in *International Conference on 3D Vision*, IEEE, 2018, pp. 131–139.

[19] A. H. Lipkus, "A proof of the triangle inequality for the tanimoto distance," *Journal of Mathematical Chemistry*, vol. 26, no. 1-3, pp. 263–265, 1999.

[20] R. Toldo and A. Fusiello, "Robust multiple structures estimation with J-Linkage," in *European Conference on Computer Vision*, 2008, pp. 537–547.

[21] L. Magri and A. Fusiello, "T-Linkage: A continuous relaxation of J-Linkage for multi-model fitting," in *Computer Vision and Pattern Recognition Conference*, Jun. 2014, pp. 3954–3961.

[22] L. Magri, F. Leveni, and G. Boracchi, "Multilink: Multi-class structure recovery via agglomerative clustering and model selection," in *Computer Vision and Pattern Recognition Conference*, 2021, pp. 1853–1862.

[23] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[24] A. Mensi and M. Bicego, "Enhanced anomaly scores for isolation forests," *Pattern Recognition*, vol. 120, p. 108 115, 2021.

[25] D. E. Knuth, *The art of computer programming: Sorting and Searching*. Addison-Wesley Professional, 1998, vol. 3.

[26] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," in *Symposium on Theory of Computing*, 1998, pp. 327–336.

[27] H. S. Wong, T.-J. Chin, J. Yu, and D. Suter, "Dynamic and hierarchical multi-structure geometric model fitting," in *International Conference on Computer Vision*, IEEE, 2011, pp. 1044–1051.