3dSAGER: Geospatial Entity Resolution over 3D Objects

ABSTRACT

Urban environments are continuously mapped and modeled by various data collection platforms, including satellites, unmanned aerial vehicles, and street cameras. The growing availability of 3D geospatial data from multiple modalities has introduced new opportunities and challenges for integrating spatial knowledge at scale, particularly in high-impact domains such as urban planning and rapid disaster management. Geospatial entity resolution is the task of identifying matching spatial objects across different datasets, often collected independently under varying conditions. Existing approaches typically rely on spatial proximity, textual metadata, or external identifiers to determine correspondence. While useful, these signals are often unavailable, unreliable, or misaligned, especially in cross-source scenarios. To address these limitations, we shift the focus to the intrinsic geometry of 3D spatial objects and present 3dSAGER (3D Spatial-Aware Geospatial Entity Resolution), an end-to-end pipeline for geospatial entity resolution over 3D objects. 3dSAGER introduces a novel, spatial-reference-independent featurization mechanism that captures intricate geometric characteristics of matching pairs, enabling robust comparison even across datasets with incompatible coordinate systems where traditional spatial methods fail. As a key component of 3dSAGER, we also propose a new lightweight and interpretable blocking method, BKAFI, that leverages a trained model to efficiently generate high-recall candidate sets. We validate 3dSAGER through extensive experiments on real-world urban datasets, demonstrating significant gains in both accuracy and efficiency over strong baselines. Our empirical study further dissects the contributions of each component, providing insights into their impact and the overall design choices.

ACM Reference Format:

. . 3dSAGER: Geospatial Entity Resolution over 3D Objects. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 17 pages.

1 INTRODUCTION

Entity Resolution (ER) is a data integration task, focusing on identifying and reconciling different data references that correspond to the same real-world entity [15, 20, 26, 27, 77]. Geospatial ER has emerged as a unique branch of ER, focusing on the deduplication of geospatial objects [4, 38, 66, 67]. The task may involve different data modalities, such as textual descriptions and geographic coordinates.

In this work, we present 3dSAGER, a novel end-to-end ER coordinate-agnostic pipeline that extends geospatial ER to the setting of 3D polygon mesh. 3dSAGER is designed for rich geometric representations such as 3D polygon meshes. The work is motivated by the lack of reliable and accurate representations of urban environments. This is particularly crucial in the context of urban disaster management[1] that extensively relies on timely data [80]. Effectively collecting and processing such data involves understanding the heterogeneity of resources, automating the discovery and selection of data sources and dynamically reorganizing applications through model-driven feedback [3]. Consider, for example, a system for early disaster warning, such as earthquakes and their subsequent events, *e.g.*, fires. Such systems need to identify rapidly evolving disaster situations where locality has a great impact. In this example, real-time fire spreading models used in Geographic Information System can identify the threatened areas and the physical structures, *e.g.* houses, that may be impacted. Towards this goal, one can aim at integrating data from various sources, such as satellites, UAVs (Unmanned Aerial Vehicles), and street cameras to assist in coordinating emergency resources effectively [1].

Virtual 3D City Models (V3CMs) offer a detailed data representation technology of geospatial objects, representing environmental and urban virtual information of surfaces and geospatial objects, including buildings, walls, windows, and vegetation [37, 69]. Information on surfaces and objects is mostly stored in a standardized data model that allows format exchange of digital 3D models, among these are *CityGML* [28] and *CityJson* [43]. The data model often combines textual descriptions (*e.g.*, location names) with coordinates (*e.g.*, latitude and longitude), and a textual representation of *polygon mesh*, a geometric object defined by its topology formulated according to a network of interconnected coordinates to form a closed 3D surface [8, 29]. To provide reliable environmental data framework, the models stored in the V3CM are continuously updated as new data is acquired, which may lead to data duplications and conflicts when integrating or analyzing the information [18, 21]. Resolving these duplications is critical for achieving an adequate and reliable representation of the environment.

Existing geospatial ER methods consider coordinates in a pointwise fashion, where the geospatial description of an object is reduced to a single representative point, *e.g.*, its centroid [4, 17, 52, 67]. Building on this, blocking, a preparatory step in ER that aims to discard non-matching pairs, often assumes that different references to the same entity should reside in nearby regions of the coordinate system. However, this assumption breaks when data sources employ different coordinate systems, leading to transformations that disrupt geometric alignment. Consequently, methods that rely on direct coordinate comparisons fail to generalize in such settings. Furthermore, other forms of geospatial information may be limited or unavailable, often due to urgent computing constraints that restrict the extraction of richer contextual data from images. Therefore, a coordinate-agnostic approach that leverages structural properties of objects becomes essential, motivating geometry-oriented featurization beyond 2D projections and metadata.

Example 1. To illustrate the problem, consider Figure 1, which contains on the left an example of CityJson 3D object representation.

[1] https://www.wvi.org/disaster-management/urban-disaster-management

Conference acronym 'XX, ,

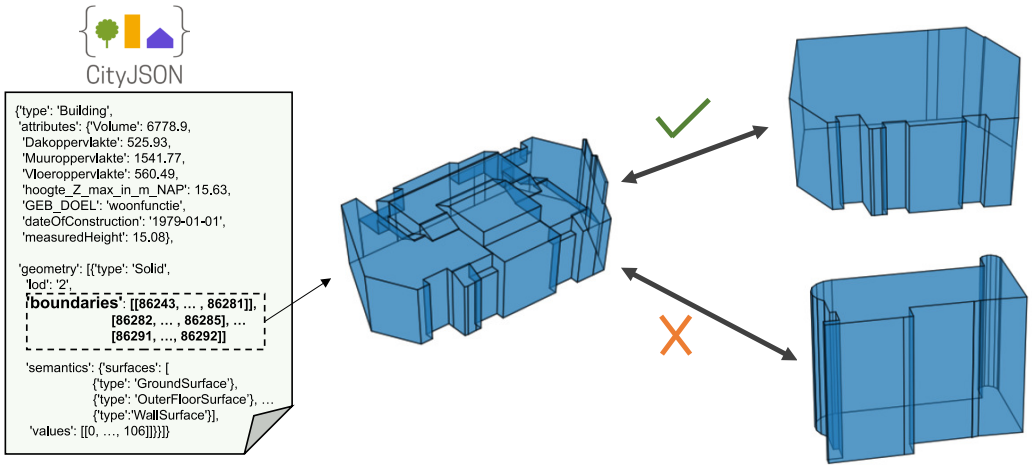


Figure 1: An example of geospatial ER over 3D objects represented as CityJson files.

On its right, a graphical illustration of the represented geospatial object (in this case a building) is provided as polygon mesh. We also provide two additional object representations on the right.

Assume that we aim to match the object on the left to the two objects of the right. The top pair forms a match (marked with ✓), due to their geometrical similarity (shape and boundary intricacies) of the two polygon meshes. In contrast, the bottom pair is identified as a non-match (marked as ✗) due to irregularities, such as sharp edges and protrusions, in their shapes.

Leveraging the complete polygon mesh for geospatial ER for 3D objects enables accurate matching based on intrinsic geometric and topological properties, rather than relying on pointwise representations accompanied with textual descriptions. Therefore, we present 3dSAGER, an end-to-end pipeline for geospatial ER for 3D objects that operates directly on polygon mesh representations of objects. We propose a novel featurization mechanism that transforms raw CityJson geometry into an interpretable feature space, capturing the intricate geometric characteristics between candidate matching pairs. Built upon a model trained over a subset of the data, we employ BKAFI, a lightweight and efficient coordinate-agnostic blocking technique that, to the best of our knowledge, is the first to leverage feature importance scores derived from a downstream matcher to select blocking keys. This approach departs from traditional approaches that extract blocking keys directly from the input data. The property vectors derived from our featurization process are subsequently used to construct pairwise feature vectors, which serve as input to a machine learning model, trained to determine whether a given candidate pair forms a match or not. Our approach is model-agnostic, allowing seamless integration with different models while ensuring both flexibility and interpretability. Our contribution can be summarized as follows.

- A formal definition of geospatial ER for 3D objects and a corresponding end-to-end solution, 3dSAGER, relying solely on the polygon mesh representations without the use of any external metadata or location-based information.
- A novel featurization framework for geospatial ER.
- A newly introduced blocking method, *BKAFI*, which leverages feature importance to select a compact set of geometric properties – offering a scalable and interpretable alternative to traditional coordinate-based blocking.
- A new large-scale benchmark dataset, *The Hague*, built from real-world heterogeneous 3D sources.[2]
- A comprehensive empirical evaluation demonstrating the effectiveness and efficiency of 3dSAGER .
- Publicly available code to facilitate reproducibility.[3]

We provide basic notations and define the problem in Section 2. Then, we propose our solution in Section 3. Section 4 describes our new benchmark, and Section 5 outlines the experimental setup and empirical analysis. Finally, related work is reviewed in Section 6.

2 PROBLEM DEFINITION

We now define the task of geospatial ER for 3D objects, performed over 3D polygon mesh data. Recall that CityJson is a common representation of such objects [43] and thus, for the reminder of the paper we will assume that the 3D objects we have at our disposal are provided as CityJson files. To establish a rigorous basis for our work, we begin by defining the concepts of a *polygon* and a *polygon mesh*, the fundamental representations of 3D objects in our setting.

***Definition* 1.** *A* polygon *is a sequence of n coordinates $p = \{v_1, v_2, \ldots, v_n\}$, such that $v_i \in \mathbb{R}^3$ for every $v_i \in p$. The sequence forms a closed shape through a set of edges, expressed as an adjacency function $\phi : \{v_1, \ldots, v_n\} \times \{v_1, \ldots, v_n\} \to \{0, 1\}$, defined as:*

$$\phi(v_i, v_j) = \begin{cases} 1, & \text{if } j = i + 1 \text{ or } (i = n, j = 1) \\ 0, & \text{otherwise.} \end{cases}$$

The adjacency function ensures that edges exist between consecutive coordinates, including the edge connecting the last coordinate v_n to the first vertex v_1, forming a closed surface. With this definition, we next define a polygon mesh.

[2]https://tinyurl.com/3dSAGERdataset
[3]https://anonymous.4open.science/r/3dSAGER/

***Definition* 2.** *A* polygon mesh *(mesh for short) P is a finite set of m polygons (Definition 1) $P = \{p_1, p_2, \ldots, p_m\}$. The polygons in P collectively form a closed 3D sutrface, such that every edge e belonging to a polygon $p_i \in P$ is shared by exactly one other polygon $p_j \in P$.*

Example 2. Figure 2 provides an example of a simple box-shaped mesh (top) and the corresponding of six quadrilateral polygons (bottom), each defined by four coordinates in $\mathbb{R}^3$.

The objects presented in Figure 1 are also meshes. The visualization was generated from the 'boundries' provided in the CityJson.

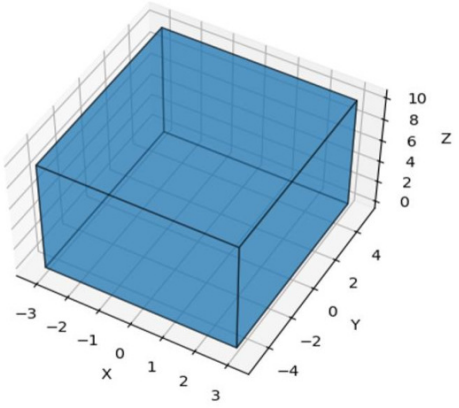
Equipped with the definition of a mesh, we are now ready to define geospatial ER. Let $D^I = \{P^I_1, P^I_2, ..., P^I_{n^I}\}$ and $D^C = \{P^C_1, P^C_2, ..., P^C_{n^C}\}$ be sets of meshes, representing an indexed dataset and a candidate dataset, respectively. We assume a clean-clean setting. This means that there are no matching objects internally in both D^I and D^C, *i.e.*, $\nexists P^I_i, P^I_j \in D^I$ ($\nexists P^C_i, P^C_j \in D^C$) such that P^I_i (P^C_i) and P^I_j (P^C_j) refer to the same real-world entity. The ER objective is to identity objects, in this case 3D objects represented as meshes, that refer to the same *real-world entity*. In the context of our work, such entities can be buildings or other geospatial objects that do not have unique identifiers. Following the common literature for ER [15], we address geospatial ER as a two stage problem consisting of *blocking* and *matching*. Largely speaking, the former aims to reduce the number of object comparisons as comparing all possible $|D^I| \times |D^C|$ pairs can be prohibitively expensive. An effective blocking strategy balances the goal of retaining as many true matches as possible while minimizing the number of non-matching pairs. The matching stage is typically more resource-intensive, as it incorporates nuanced characteristics for better differentiation between matching and non-matching pairs. Stemming mainly from the absence of primary keys, matching relies on semantic cues, may they be geospatial or geometric, to asses whether polygons refer to the same real-world spatial object.

Geospatial ER Blocking: Given an indexed dataset D^I and a candidate dataset D^C, blocking step $B(D^I, D^C)$ (or a *blocker*) aims to discard obvious non-matching mesh pairs, thereby reducing the computational complexity of subsequent steps. The outcome of this step is a set of candidate mesh pairs $C_{D^I \times D^C} \in D^I \times D^C$ (C when clear from context).

Geospatial ER Matching: Given a blocker $B(D^I, D^C)$, we now define the matching problem. We assume the existence of an oracle mapping function θ, unknown to us. That is, our goal is to obtain a model $\tilde{\theta}$ to identify duplicate (matching) meshes in C.

Problem 1. Let $D^I = \{P^I_1, P^I_2, ..., P^I_{n^I}\}$ be an index dataset, $D^C = \{P^C_1, P^C_2, ..., P^C_{n^C}\}$ be a candidate dataset and $C \in D^I \times D^C$ a set of mesh pairs obtained by a blocker $B(D^I, D^C)$. The matching problem aims to obtain a model $\tilde{\theta} : C \rightarrow \{0, 1\}$, classifying a pair $(P^I_i, P^C_j) \in C$ as a match (1) or non-match (0).

The focus of our work is in developing (1) a blocker B and (2) a model $\tilde{\theta}$. Specifically, the blocker will be based on a reduced set of features that support maximum recall at a low cost. The model will be a machine-learning model, trained on pairwise feature vectors that encode the geometric relationships between meshes in candidate pairs. Obviously, a "perfect" $\tilde{\theta}$ is one that is equivalent to θ (or at least has equivalent outputs). For the context of this work we assume we have at our disposal a set that was annotated with θ (aka a training set), which can be used to train a model. For simplicity of presentation, this study focuses on clean-clean ER assumption. Yet, the proposed framework is flexible and can be adapted to non clean-clean settings as well. These include settings where duplicates may exist within the indexed dataset D^I, the candidate dataset D^C, or both, enabling broader applicability in diverse real-world contexts.



$p_{\text{Base}} : [(-3, -5, 0), (3, -5, 0), (3, 5, 0), (-3, 5, 0)]$
$p_{\text{Top}} : [(-3, -5, 10), (3, -5, 10), (3, 5, 10), (-3, 5, 10)]$
$p_{\text{Left}} : [(-3, -5, 0), (-3, -5, 10), (-3, 5, 10), (-3, 5, 0)]$
$p_{\text{Right}} : [(3, -5, 0), (3, -5, 10), (3, 5, 10), (3, 5, 0)]$
$p_{\text{Front}} : [(-3, -5, 0), (-3, -5, 10), (3, -5, 10), (3, -5, 0)]$
$p_{\text{Back}} : [(-3, 5, 0), (-3, 5, 10), (3, 5, 10), (3, 5, 0)]$

Figure 2: An example of a simple box-shaped mesh, represented as a set of interconnected polygon faces.

To illustrate the relevance of this setting, we introduce next a case study of a post-disaster management (PDM) scenario. Natural disasters (*e.g.*, earthquakes and floods) require rapid and accurate situational awareness for prioritizing emergency response. A central challenge in such scenarios, especially in urban areas, involves aligning newly captured 3D imagery of damaged areas, often collected by UAVs, against a registry of existing 3D images, as maintained by local authorities. The UAV-acquired datasets, represented by meshes, may lack trustworthy geo-location due to factors such as GPS disruption or the lack of correction infrastructure (e.g., CORS networks) [50, 82], particularly in remote or underdeveloped regions [14, 75, 79]. Moreover, imagery metadata may be missing or inconsistent [74], making traditional matching signals like spatial proximity or external identifiers ineffective. The only reliable information may be the intrinsic geometric structure of captured objects. As a part of our empirical analysis (Section 5.2.5), we analyze the role of matching in supporting PDM.

3 3DSAGER: MATCHING 3D OBJECTS

We now present 3dSAGER (3D Spatial-Aware Geospatial Entity Resolution), our novel end-to-end solution for geospatial ER over

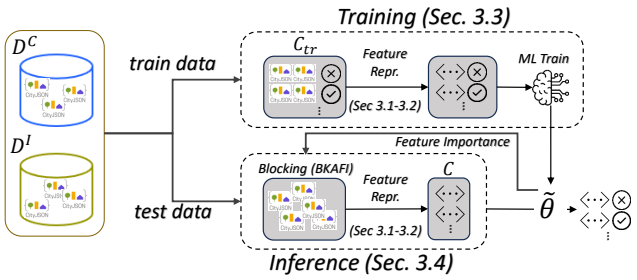


Figure 3: A schematic overview of the full pipeline.

3D objects. As depicted in Figure 3, 3dSAGER comprises two main stages, namely *training* (top panel) and *inference* (bottom panel). Both stages operate over the indexed dataset D^I and the candidate dataset D^C. In the initial step, mesh items from both datasets are transformed into *property vectors* (Section 3.1), which are extracted directly from the polygon mesh geometry, unlike traditional ER pipelines that rely on tabular or metadata-derived features. This featurization is structurally tied to the 3D mesh representation and designed to capture intrinsic geometric characteristics relevant for matching. To the best of our knowledge, this is the first application of such a featurization approach in the context of ER. In the training stage, we use a set of labeled object pairs $C_{tr} \subseteq D^I \times D^C$, represented as property vectors. These pairs undergo a featurization step that converts them into a machine-learning-compatible format, capturing pairwise geometric relationships between the objects (Section 3.2). This representation is then leveraged to train an interpretable binary classifier $\tilde{\theta}$, such as XGBoost [13][4] (Section 3.3). The output of this phase is the trained model $\tilde{\theta}$. The inference phase resembles a traditional ER pipeline: a blocking method (in our case, the proposed *BKAFI*, Section 3.4.1) is first applied to the input datasets, generating a candidate set $C \subseteq D^I \times D^C \setminus C_{tr}$ (excluding training pairs). This filtered set is then passed to the matching phase for final prediction (Section 3.4.2).

3.1 Property Vectors

We introduce first the generation of property vectors. Raw meshes, typically expressed as sequences of 3D coordinates, are not inherently interpretable by machine learning models. Even for human experts, analyzing raw coordinate data provides limited insight into the structure or semantics of spatial objects. This low-level representation lacks the abstraction needed to capture meaningful geometric relationships. While visualization tools (see, *e.g.*, Figures 1 and 2) enable humans to perceive subtle characteristics and nuances within polygons, translating these insights into machine-readable features requires a systematic computational approach. To address this challenge we propose to generate a property vector that encodes various geometric properties of a mesh, enabling its processing by machine learning algorithms later in the pipeline.

The full set of computed geometric properties, denoted by $G = \{g_1, g_2, \ldots, g_n\}$, captures a diverse range of characteristics describing each mesh from multiple perspectives [42]. These properties encompass dimensions, shape descriptors, boundary complexity, and structural indicators – together providing a comprehensive geometric profile. Table 1 presents a selected subset of representative properties used in our pipeline. The rightmost columns of the table provide examples of a candidate mesh from D^C (left) and an object from the index set D^I (right).

[4]Gradient-boosted decision trees, *e.g.*, XGBoost, are considered state-of-the-art for tabular data [51].

Table 1: Object Properties Computed over each Polygon. The `Ind. Example` and `Cand. Example` columns contain the property values computed for the matching objects in Figure 1.

		Cand. Example	Ind. Example
1. Size and Dimensions			
`area`	Total surface area of the mesh.	292412	271745
`volume`	3D volume of the mesh.	6778859	1269.7
`height_diff`	Height difference between vertices.	29.7	11.6
`num_vertices`	Number of unique vertices defining the mesh.	194	48
2. Perimeter and Boundary Characteristics			
`perimeter`	Total length of polygon's boundary.	114043.9	114003.7
`circumference`	Boundary length.	8.6	5.2
`perimeter_index`	Ratio of perimeter to area indicating boundary complexity	1.7	1.6
3. Shape Complexity and Geometry			
`convex_hull_area`	Area of the polygon's convex hull comparing irregularity.	108013.9	107877.8
`ave_centroid_distance`	Average distance of boundary points from the centroid.	14077.1	16336.8
`shape_index`	Ratio reflecting compactness of the shape.	4.1	5.6
`fractality`	Self-similarity, indicating shape irregularity.	0.0	0.0
`elongation`	Degree of shape's elongation.	2.8	3.1
`hemisphericality`	Degree of similarity to a hemisphere.	17.8	8.3
`cubeness`	Degree of similarity to a cube.	10.6	6.4
4. Symmetry and Alignment			
`axes_symmetry`	Symmetry relative to principal axes.	6967.7	8854.1
5. Structural and Functional Properties			
`density`	Compactness or mass distribution within the polygon's area/volume.	25640.3	23836.6
`num_floors`	Number of distinct floors.	25	49

For each mesh p in D^I and D^C, we compute a property vector $G_p = (g_1(p), g_2(p), \ldots, g_n(p))$, where each $g_i(p)$ corresponds to a specific geometric property. The columns `Cand. Example` and `Ind. Example` in Table 1 provide the actual computed values for the real matching pair illustrated in Figure 1, where the middle object originates from the candidate dataset D^C, while the object on the right comes from the indexed dataset D^I. Although these two objects differ visually and numerically in several geometric properties, they are still correctly identified as a match. For instance, the `num_vertices` property varies significantly (194 vs. 48), while others such as `perimeter` and `convex_hull_area` remain closely aligned. These variations illustrate the role of our property-based approach in tolerating moderate differences while leveraging consistent structural signals across sources. To mitigate the effect of scale disparities across properties while preserving relative magnitudes, we apply a logarithmic normalization to the computed features using the $\log(1+x)$ transformation [12, 49]. Formally, we normalize a property value $g_i(p)$ for a mesh p to be $\tilde{g}_i(p) = \log(1 + g_i(p))$.

3.2 Pairwise Feature Vectors

Our goal is to identify matches between objects despite property discrepancies, leveraging both shared and differing geometric characteristics. With a property vector representing a single mesh, the next step is to generate feature vectors that encapsulate the pairwise relationships between corresponding object properties. Formally, we use $F(P_1, P_2)$ to denote a feature vector over the pair of meshes $P_1 \in D^I$ and $P_2 \in D^C$. To capture the mutuality of object properties,

we introduce a *division* operator that computes the element-wise ratio between corresponding entries in the two property vectors.

***Definition* 3** (Division Operator)**.** For meshes $P_1 \in D^I$ and $P_2 \in D^C$, let $G_{P_1} = (g_1(P_1), g_2(P_1), \ldots, g_n(P_1))$ be a property vector of P_1 and $G_{P_2} = (g_1(P_2), g_2(P_2), \ldots, g_n(P_2))$ be a property vector of P_2. The *division operator*, denoted by $G_{P_1} \div G_{P_2}$, is defined as follows.

$$F_{\div}(P_1, P_2) = G_{P_1} \div G_{P_2} \tag{1}$$

where

$$F_{\div}(P_1, P_2)[i] = \frac{g_i(P_1)}{g_i(P_2)}, \quad \forall g_i(P_1) \in G_{P_1}, \quad g_i(P_2) \in G_{P_2}.$$

An entry in the feature vector $F_{\div}(P_1, P_2)$ encodes the proportional variation of a property in the two objects. By expressing pairwise relationships as ratios (rather than absolute differences), this representation remains meaningful even when the objects originate from sources with differing scales or systematic offsets, thereby enhancing robustness to discrepancies across data sources.

Systematic Discrepancy: Intuitively, one might expect that matching mesh references should exhibit closely aligned property values. In such cases, the ratios computed by $F_{\div}(\cdot)$ would be approximately 1. However, since the datasets originate from distinct sources with varying acquisition methods, timestamps, and modeling conventions, consistent patterns of variation may arise, referred to as systematic discrepancy. As all meshes in D^C come from the same source, they may collectively deviate from those in D^I in specific properties. For instance, one dataset may systematically underestimate dimensions such as area or height, or apply different resolutions or coordinate transformations. These deviations are not random noise, but rather reflect structured biases introduced during data collection. We formally define systematic discrepancy next.

***Definition* 4** ((ϵ, δ)-*Systematic Discrepancy*)**.** Let D^I and D^C denote the indexed and candidate sets, respectively, and let $g \in G$ be a geometric property. We say that D^I and D^C exhibit an (ϵ, δ)-*systematic discrepancy* with respect to g if there exists a ratio $r_g \in \mathbb{R}^+$ such that at least a $(1-\delta)$ fraction of the known matching pairs (P_1, P_2), with $P_1 \in D^I$ and $P_2 \in D^C$, satisfy

$$r_g - \epsilon \leq \frac{g(P_1)}{g(P_2)} \leq r_g + \epsilon. \tag{2}$$

ϵ represents the tolerance allowed for the ratio of geometric property values between matching pairs. A smaller value of ϵ indicates a stricter condition for the matching pairs, implying that the properties of the pairs should be more similar to be considered a valid match. δ reflects the efficacy of the tolerance condition. A smaller value of δ (closer to 0) implies that a higher proportion of matching pairs must meet the condition, resulting in a stricter requirement for the consistency of the geometric properties.

Another perspective on (ϵ, δ)-systematic discrepancy is through the lens of statistical dispersion. Given a set of ratios $\left\{\frac{g(P_1)}{g(P_2)}\right\}$ for matching pairs (P_1, P_2) and a property g, the standard deviation σ of this ratio distribution provides an empirical way to estimate a reasonable ϵ value. In particular, if the ratios are normally distributed, setting $\epsilon = 2\sigma$ would cover roughly 95% of the values, corresponding to $\delta \approx 0.05$. Thus, the standard deviation offers a data-driven approach to quantify and validate systematic discrepancy, with lower σ implying stronger consistency across sources.

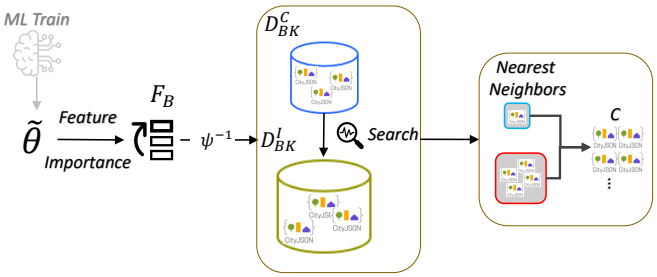


Figure 4: An illustration of BKAFI.

The property values in Table 1 exemplify the phenomenon of *systematic discrepancy*. For instance, the `convex_hull_area` property only slightly differs between the candidate (108,013.9) and indexed (107,877.8) objects. Conversely, properties like `axes_symmetry` exhibit more noticeable differences, despite the objects being a confirmed match (Figure 1). Such patterns, when consistently observed across numerous matching pairs, indicate systematic (rather than random) discrepancies. While certain ratios may be inferred through direct computation over the training data, more intricate relationships between different properties are not always apparent through simple observation. Instead, we delegate the identification of such geometric consistencies to a learning algorithm, which can effectively capture and leverage these complex patterns.

3.3 3dSAGER Training Phase

The feature vectors of objects in the training set C_{tr} serve as input to a machine learning model $\tilde{\theta}$ for classification. During training, the model learns to map the input feature vectors to binary labels, where a label of 1 indicates that the mesh pair corresponds to the same real-world entity, and 0 otherwise. The training process optimizes the model parameters by minimizing a loss function that measures the variation between the predicted and true labels. 3dSAGER is machine learning model-agnostic, offering a flexible and adaptable framework that can be incorporated into any model. We emphasize the use of interpretable models, enabling the utilization of feature importance scores in the blocking phase.

3.4 3dSAGER Inference Phase

Equipped with a trained model $\tilde{\theta}$, we now describe the inference process for blocking (Section 3.4.1) and matching (Section 3.4.2).

3.4.1 Blocking based on Feature Importance. We introduce a new blocking method, *Blocking Key as Feature Importance* (*BKAFI*), tailored for geospatial ER for 3D objects. The key insight is that a small subset of geometric properties captures most of the information needed for matching. Hence, instead of searching the full feature space, BKAFI restricts the search to a space defined by the most informative features. To identify these, we trace back the top-ranked features from the trained model $\tilde{\theta}$ to their originating geometric properties (Section 3.2). For each candidate object, we then retrieve similar objects from D^I based on their values in this reduced space.

Algorithm 1 BKAFI: Blocking Key as Feature Importance.

Require: Trained model $\tilde{\theta}$, candidate set D^C, $|F_B|$ search space size, index set D^I, nearest neighbor parameter k

1: Initialize modified candidate set $D^C_{BK} \leftarrow \emptyset$
2: Initialize modified indexed set $D^I_{BK} \leftarrow \emptyset$
3: Initialize set of candidate meshpairs $C \leftarrow \emptyset$
4: $F_B \leftarrow$ Extract top $|F_B|$ features with the highest feature importance scores from $\tilde{\theta}$
5: $G' \leftarrow \{\psi^{-1}(f) \forall f \in F_B\}$
6: **for** each $P \in D^C$ **do**
7: $\quad G_{P_{BK}} \leftarrow G_P[G']$
8: $\quad$ Add $G_{P_{BK}}$ to D^C_{BK}
9: **end for**
10: **for** each $P \in D^I$ **do**
11: $\quad G_{P_{BK}} \leftarrow G_P[G']$
12: $\quad$ Add $G_{P_{BK}}$ to D^I_{BK}
13: **end for**
14: $BKAFI_{Ind} \leftarrow Index(D^I_{BK})$
15: **for** each $G_{P_{BK}} \in D^C_{BK}$ **do**
16: $\quad P \leftarrow$ Map $G_{P_{BK}}$ to its corresponding item in D^C
17: $\quad k_{G_{P_{BK}}} \leftarrow$ Retrieve$(k, G_{P_{BK}}, BKAFI_{Ind})$
18: $\quad$ **for** each $G_{P_I} \in k_{G_{P_{BK}}}$ **do**
19: $\quad\quad$ Map G_{P_I} back to its corresponding item in $P_I \in D^I$ and add (P, P_I) to C
20: $\quad$ **end for**
21: **end for**
22: **return** C

BKAFI introduces several key differences from traditional blocking methods. It avoids reliance on coordinate data, operates over features extracted from polygon mesh geometry, and, critically, derives blocking keys from the trained matcher's feature importance scores rather than manually selected attributes. This approach not only improves adaptability to non-standard 3D mesh data, but also ensures that the blocking process remains lightweight and efficient.

The *BKAFI* algorithm (see Algorithm 1 for pseudocode and Figure 4 for illustration) receives as input a trained model $\tilde{\theta}$, candidate set D^C, and index set D^I. F_B denotes the subset of features of size ($|F_B|$). k is the number of candidate pairs generated per each object in D^C, which is also provided to the algorithm as input.

The algorithm extracts the $|F_B|$ features with the highest feature importance scores from $\tilde{\theta}$ (Line 4). Next, the original geometric properties corresponding to the features in F_B are mapped using an inverse function ψ^{-1}, resulting in the set G' (Line 5). The algorithm then iterates through each candidate object $P \in D^C \cup D^I$, projects its properties onto the selected subset F_B to obtain the property vector $G_{P_{BK}}$, and adds it to the candidate set D^C_{BK} and index set D^I_{BK} (Lines 7–8 and 11-12, respectively). The index set D^I_{BK} is then indexed and stored in $BKAFI_{Ind}$ (Line 14). The algorithm continues by iterating through each modified feature vector $G_{P_{BK}}$ in D^C_{BK} (Lines 16-19). For each vector, the corresponding object P is mapped back from the feature vector to its item in D^C and the k nearest neighbors are retrieved from the indexed set $BKAFI_{Ind}$ (Line 17). The algorithm then maps each nearest neighbor back to the corresponding item in D^I and adds the candidate pair (P, P_I) to the set C (Line 19). Finally, the set of candidate pairs C is returned as the output of the algorithm (Line 22).

We use a *KDTree* index [6, 65], a proven efficient and effective indexing method for low-dimensional spaces, to index the set D^I in the $|F_B|$-dimensional feature space. Its construction (Line 14) takes $O(|D^I| \log |D^I|)$ [6], which is a one-time cost and typically negligible compared to the cost of performing nearest neighbor searches for all candidates in D^C. Under the assumption of low dimensionality, each query takes $O(\log |D^I|)$, making the method scalable for large $|D^C|$. While the parameter k has little impact on the query complexity, it directly impacts the number of candidate pairs generated per item, which in turn affects the cost of the downstream matching phase.

To mitigate pair over-generation, we refrain from returning exactly k nearest neighbors. Instead, we incorporate a retrieval-time similarity-based pruning, allowing early process termination whenever the distance in the sub-property feature space is above a calibrated threshold, derived from training data. This adjustment reduces false positives and supports more efficient matching without compromising recall. Specifically, we compute distances between matching pairs in the training set and set a high threshold, *e.g.*, 95-th percentile of the distribution. At inference time, retrieval of candidates proceeds in increasing distance order until either k neighbors are collected or the distance to the next neighbor exceeds the threshold. It is worth noting that a higher quantile increases robustness to noise but may miss potential matches, while a lower quantile favors recall at the risk of introducing false positives.

As for feature set size $|F_B|$, we empirically observe (Section 5.2.4) that smaller subsets of informative features, selected in a data-driven manner, yield faster and more effective search.

3.4.2 Matching. The matching process is performed using the candidate set C, which was obtained in the blocking phase. The model $\tilde{\theta}$ (Section 3.3), is then used for prediction. For each pair in C, we run the featurization process (Section 3.1), and subsequently pass it through the trained model to generate a match/ non-match decision.

4 NEW BENCHMARK FOR 3D GEOSPATIAL ER

To the best of our knowledge, geospatial ER for 3D objects has not been previously studied, and no standardized benchmarks exist in the literature. To address this gap, we curated a new benchmark dataset from the city of *The Hague*, constructed from two distinct sources, enabling a natural partition into candidate and indexed sets. Below, we detail the dataset construction process and describe its different variants. Key statistics are summarized in Table 2.

Data Sources: The 3D City Model *The Hague 2022* (the candidate set) is maintained by the municipality of *The Hague* as part of the Open Data platform.[5] It is derived from 2021 aerial photographs and structured according to the Register of Buildings and Addresses (*BAG*). The raw data comprises 46 CityJson files, each describing a distinct district, where the coordinate system used in all files is RDnew, the national coordinate system of the Netherlands.

[5] https://denhaag.dataplatform.nl/

Table 2: Statistics of training and test sets averaged over three random seeds. Candidate and index sizes are shown only for test sets, as they are relevant to the blocking evaluation. The training set is used solely for learning the matcher.

Set	Size	Candidates	Index	Total Pairs
Train	Small	–	–	14,178
	Large	–	–	60,379
Test	Small	3,507	9,985	13,789
	Large	15,588	97,437	52,586

3DBAG (the index set) is a large-scale openly available 3D building dataset, containing over 10 Million buildings across the Netherlands [63]. The data was automatically generated from three sources of the Topographic Register of the Netherlands – *BAG*,[6] the National Height Model of the Netherlands (AHN),[7] and *TOP10NL*[8] – using various methods such as aerial photography, terrestrial surveying, and airborne laser scanning. The dataset is publicly available for download[9] and is partitioned into spatial tiles for efficient access. Since our focus is on *The Hague*, we selected 246 tiles covering more than 100,000 buildings within the city's boundaries, providing spatial overlap with the first data source.

Both data sources include unique object identifiers, enabling a direct mapping between them after standardizing key formats to resolve format inconsistencies. To guarantee data quality and consistency, we retained only objects (buildings) composed of at least 10 polygons. This threshold eliminates incomplete or overly simplified structures, ensuring that the retained objects exhibit sufficient geometric complexity for meaningful evaluation.

Dataset Variants: To ensure robust evaluation, we generate two dataset variants by varying the training and testing configurations across three random seeds. Specifically, we sample training sets of the following sizes: *small* (0.1) and *large* (0.6). These ratios are applied to the intersection of candidate and index object identifiers to determine the number of matching training samples.

For training, we construct different pair sets for the blocking and matching components. For blocking we enforce that every object in the candidate set has a corresponding match in the index setby retaining, in the candidate set, only objects that appear in both sources. For each such candidate object, its corresponding index object is included to form a positive pair. We apply negative sampling by generating two non-matching pairs for each matching pair, selected at random. For matching, to yield a challenging and realistic evaluation scenario, we use the output of the best-performing blocking configuration (see Section 5.2). Specifically, we generate a candidate pair set by retrieving the top-3 nearest neighbors for each candidate object in the property space, with the true match always included, even if it is not among the top-3 results.

To construct the test data, we follow a two-step strategy over both blocking and matching. For blocking, we generate candidate and index sets by identifying test candidate objects from the two source intersection, excluding any identifiers used in the training data. For each test size (*small* (0.1) and *large* (1.0)) we randomly sample a subset of remaining candidates, where the ratio in parentheses indicates the fraction of the remaining candidate pool used. To evaluate BKAFI's reduction power we augment the candidate set so that 20% of candidates do not have a matching object in the index set. The corresponding index set is constructed by copying the sampled test candidate set, then augmenting it with random samples from the full index object pool. The size of this additional sample set is determined by applying the same test ratio to the total number of index objects. For matching, we sample from the unused portion of the intersection between the two sources, excluding any object identifiers that were present in the training pairs. For each test size, we sample a set of matching test pairs by selecting candidate objects and pairing them with their corresponding index matches. To form a set of non-matching pairs, we again use the best-performing blocking configuration to retrieve the top-3 nearest index objects for each candidate in the sampled test set. These top-3 results form the set of candidate pairs for evaluation, ensuring the inclusion of the true match. This design ensures a realistic yet challenging evaluation scenario for the matcher.

The reported values (Table 2) represent averages computed over the three different random seeds employed during data partitioning. To the best of our knowledge, this is the first work to leverage polygon meshes for geospatial ER, introducing a new modality to the domain. To facilitate further research the dataset is released as a benchmark for geospatial ER for 3D objects.[10]

5 EMPIRICAL EVALUATION

Equipped with the new benchmark, we now evaluate 3dSAGER. Section 5.1 describes the experimental setup, followed by blocking and matching evaluation (sections 5.2 and 5.3, respectively).

5.1 Experimental Setup

We outline the experimental framework, starting with a description of an additional dataset used for our case study. We then provide implementation details (Section 5.1.2), describe the evaluation metrics (Section 5.1.3), present the baselines (Section 5.1.4) and conclude with key design choices (Section 5.1.5).

5.1.1 Case Study Data. In addition to our new benchmark (Section 4), we collaborated with a third-party partner to obtain real-world data from a post-disaster scenario. This data is proprietary and thus cannot be released publicly. It was collected from two distinct sources and consists of 3D objects represented in LOD2. The data was generated from point clouds and approximate building footprints, with roofs reconstructed using plane adjustment techniques. We retained only objects with at least 10 polygons, and applied the same partitioning policy as introduced in Section 4, resulting in 5,472 and 3,705 total pairs for Train and Test, respectively. For the blocking component analysis, the test set is split into 938 candidate objects and 9,617 index objects. We refer to this dataset as PDM (Post-Disaster Management).

[6] https://www.kadaster.nl/zakelijk/registraties/basisregistraties/bag
[7] https://www.ahn.nl/
[8] https://www.kadaster.nl/zakelijk/producten/geo-informatie/topnl
[9] https://3dbag.nl/en/download
[10] https://tinyurl.com/3dSAGERdataset

5.1.2 Implementation Details. All experiments were conducted on a server running CentOS 7, equipped with 2 Nvidia Quadro RTX 6000 GPUs. To ensure robust evaluation, each experiment was repeated across three random initialization seeds. Our implementation is publicly available, and the full list of features used in our experiments can be found in the repository configuration file.

To instantiate the model $\tilde{\theta}$ (see Section 3.3), we evaluated a variety of machine learning algorithms on the $Hague_{small}$ dataset, including `RandomForest` (RF) [10], `AdaBoost` (AB) [23], `Gradient Boosting` (GB) [24], `Bagging` (B) [9], and `ExtremeGradient Boosting` (XGB) [13]. We also evaluated a neural baseline implemented as a multilayer perceptron (MLP) with 2 or 3 fully connected layers. The XGBoost model uses the official open-source implementation,[11] while all other models rely on the Scikit-learn library [59].[12] Each model was trained using grid search with 5-fold cross-validation over a predefined hyperparameter space, with configuration details provided in our GitHub repository. The `Bagging` model performance dominated the alternatives (see Section 5.3.2) and was adopted as the matcher of choice for the evaluation.

5.1.3 Evaluation Metrics. In line with prior work [58], we evaluate blocking effectiveness considering the tradeoff between match coverage and pruning efficiency. The former is measured by *Pair Completeness* (PC), the proportion of matching pairs retained after the blocking step (corresponds to recall). The second is *Reduction Ratio* (RR), the reduction in the number of pairwise comparisons relative to the full cross-product of the candidate and index sets [56]. Unless stated otherwise, RR is governed by the parameter k, set to a maximum of 20, without applying the distance threshold-based pruning (Section 3.4.1). This value reflects realistic candidate list sizes commonly used in downstream matching tasks [47, 53]. Efficiency is evaluated in terms of execution time (in seconds).

For matching, in line with prior work [15, 47], we evaluate performance using precision, recall, and F1-score. We also analyze scalability using runtime and memory usage. This includes measuring training and inference time, as well as model size, offering a more comprehensive view of deployment feasibility.

For the case study (PDM dataset, see Section 5.1.1), we focus on a specific measurable KPI, **Mutual Registration Accuracy** (MRA) [72], which can be derived from Recall@K.

5.1.4 Baselines. Since geospatial ER for 3D objects has not been previously addressed in the literature, there are no established baselines for direct comparison. To enable meaningful evaluation, we adapt methods from related tasks and construct competitive alternatives to benchmark the performance of 3dSAGER.

***Blocking Baselines*:** We evaluate the following baselines:

• **ViT**: Visual representations of meshes are generated by rendering objects as PNG images (see Figure 1). Embedding vectors are extracted using pretrained Vision Transformer (ViT) models [19], followed by similarity search using Faiss [35] to retrieve potential matches. We experiment with two variants: *ViT-B/32* and *ViT-L/14*. This approach tests whether image-based representations effectively captures structural similarity between buildings, offering a blocking strategy that is independent of explicit spatial information.

• **ViT+Contrastive**: This baseline builds on the ViT setup by fine-tuning the visual encoder using contrastive learning. We use training set pairs (as described in Section 4), where each pair consists of a candidate and an index object rendered as PNG images. The model architecture reuses the visual encoder from a pretrained Vision Transformer (ViT), and is trained with a contrastive loss (Cosine Embedding Loss) to pull matching pairs closer in the embedding space while pushing apart non-matching ones. We use the CLIP implementation of both *ViT-B/32* and *ViT-L/14* as the backbone and train for 8 and 5 epochs per random seed, respectively, after observing that the contrastive loss converges within these ranges. After training, the encoder is used to extract refined embeddings for similarity-based blocking with Faiss.

***Matching Baselines*:** We use the following baselines:

• **ViT+Contrastive**: Similar to the blocking setup, we fine-tune a visual encoder using contrastive learning to bring matching object pairs closer in the embedding space. However, in this case, the goal is to perform binary classification (match or non-match). Specifically, we build a lightweight binary classifier on top of a frozen visual encoder (e.g., `ViT-B/32` or `ViT-L/14`) from the CLIP model [64]. The classifier, implemented as a two-layer feedforward network with ReLU activation, receives the absolute difference between the image embeddings as input and learns to predict the matching decision boundary. The model is trained using binary cross-entropy loss (`BCEWithLogitsLoss`) over labeled image pairs, with training data constructed from matching and non-matching object pairs (see Section 4). Training is conducted using the AdamW optimizer with a learning rate of 1e−5 and a batch size of 4.

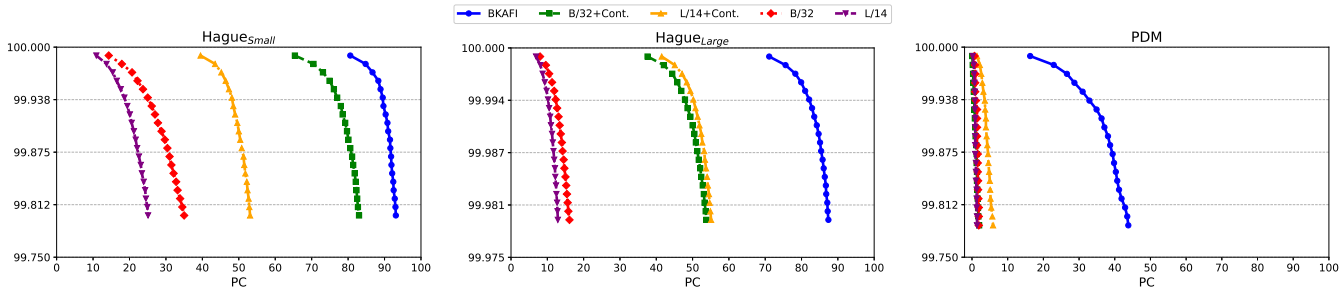
• **LLM (GPT-4o)**: Given the predominant performance of GPT-4o over other LLMs in traditional ER problems [62], we implemented an LLM-baseline for geospatial ER for 3D objects. Given the supervised learning setup, we provided examples in the prompt (in-context learning). The performance of GPT-4o was inferior and thus omitted from the results. This may be attributed to the model's inability to effectively process raw polygon mesh representations.

5.1.5 Hyperparameters and Design Choices. For BKAFI, aside from the detailed component analysis in Section 5.2.4, we report results using the value of $|F_B|$ (i.e., the number of selected blocking key functions) that achieved the best area under the PC@k curve (AUC) on the training set. In addition to comparing different model architectures for matching, we employ the `Bagging` model for the ablation study. Feature importance scores computation, used as a preparatory step for blocking, are performed using a `RandomForest` model, chosen for its efficiency and fast training time.

5.2 Blocking Evaluation

This section presents the experimental analysis of BKAFI, our blocking approach. We begin by comparing BKAFI with baseline methods (Section 5.2.1), followed by an ablation study analyzing its key components, namely normalization (Section 5.2.2), and the property selection criterion used to define the blocking keys (Section 5.2.3). We also analyze the effect of $|F_B|$, the number of properties dictating the search space size and Cardinality, in Section 5.2.4. Finally, we provide a case-study analysis over the PDM dataset (Section 5.1.1).

[11] https://xgboost.readthedocs.io/
[12] https://scikit-learn.org/



(a) *Hague$_{small}$* (b) *Hague$_{large}$* (c) *PDM dataset*

Figure 5: RR vs. PC comparison between BKAFI and baseline methods across the different datasets.

5.2.1 BKAFI vs. Baselines. We start by comparing BKAFI against the baseline methods (Section 5.1.4), along the dimensions of effectiveness and runtim. Our analysis shows that BKAFI outperforms all baselines over all experiment settings and does that 1-2 order of magnitudes faster.

Effectiveness: Figure 5 compares BKAFI with the four baseline models (Section 5.1.4) in terms of RR versus PC, with both metrics reported as percentages. Each RR value corresponds to a specific k used in the candidate generation step. For example, for *Hague$_{small}$*, setting $k = 20$ yields an RR of $100 \cdot \left(1 - \frac{20 \times 3{,}507}{3{,}507 \times 9{,}985}\right) \approx 99.7997$. Intuitively, being farther to the right on the plot is better, as it indicates higher PC score for a given RR value.

We first observe that BKAFI dominates the baselines for all datasets. For example, over *Hague$_{large}$* BKAFI achieves $PC = 81\%$ for $k = 5$ ($RR \approx 99.9949\%$), compared to 46.9% and 49.4% contrastive learning-based baselines, `B/32+Cont.` and `L/14+Cont.`, respectively. All methods experience a drop in PC while moving from *Hague$_{small}$* to *Hague$_{large}$*. This is expected, as a larger index set (97, 437 objects compared to 9, 985) inherently contain more potential options per object from the candidate set, making the blocking task more challenging. Interestingly, the performance degradation is more pronounced for baseline methods, particularly the raw ViT-based models, indicating their limited scalability to larger search spaces. In contrast, BKAFI maintains robust performance across dataset sizes, with only moderate declines in PC, especially in the low-k range. This suggests that BKAFI is better equipped to handle increased data complexity and scale.

On the *PDM* dataset, which contains simpler geomteries and is thus more challenging for image-based similarity alone, the gap widens further. BKAFI reaches $PC = 31.2\%$ at $RR \approx 99.948\%$ ($k = 5$), whereas the best baseline, `L/14+Cont.`, only achieves 3.2%, and others below 2%. This sharp contrast highlights the advantage of BKAFI's geometry-aware blocking mechanism, particularly where visual embeddings struggle to capture fine-grained structural cues.

Results demonstrate that BKAFI not only scales well with dataset size but also generalizes better to diverse 3D modeling conditions, making it a reliable choice for handling geospatial ER for 3D objects.

Execution Time: Table 3 provides a runtime comparison of all methods across different dataset sizes. BKAFI is substantially faster than baselines. For example, on the *Hague$_{large}$* dataset, BKAFI completes the blocking step in just 1.08 seconds, compared to 279.43 and 421.29 seconds for the `B/32+Cont.` and `L/14+Cont.` baselines, respectively, amounting to over two orders of magnitude speedup.

Table 3: Blocking Execution Time (in seconds)

Method	Hague$_{\text{small}}$	Hague$_{\text{large}}$	PDM
BKAFI	**0.22**	**1.08**	**0.08**
B/32+Cont.	5.09	279.43	1.29
L/14+Cont.	9.22	421.29	2.50
B/32	2.69	176.75	1.30
L/14	4.91	264.77	2.58

BKAFI's speedup may be attributed to its compact feature representation (Section 3.4.1). While the baseline models operate in high-dimensional embedding spaces (512 and 768 for B/32 and L/14, respectively), BKAFI uses only the most important features as a blocking key. Also, ViT-based baselines incur overhead due to the exhaustive embedding generation for all objects, a step involving forward passes through a heavy neural backbone. For the fine-tuned variants, the cost is further amplified due to a separate training phase over image pairs. Image rendering, a non-negligible one-time preprocessing step, is excluded from the reported runtime. The comparative analysis underscores BKAFI's suitability for large-scale deployments, where efficiency and scalability are critical.

5.2.2 Effect of Normalization. We now asses property value normalization impact (Section 3.1). For ease of presentation, we report PC@k ($k \in \{1, 5, 20\}$) instead of RR, as it offers a direct view of recall at fixed candidate list sizes. As shown in Table 4, normalization consistently improves PC@k across all datasets, particularly at lower k values. For instance, on *Hague$_{small}$*, PC@1 improves from 29.5% to 80.6%, and on *Hague$_{large}$*, from 59.2% to 71.1%. Gains at $k = 20$ are smaller but still present. On the *PDM* dataset, although the absolute PC values are lower, normalization still leads to a measurable gain. For example, PC@5 increases from 28.8% to 31.0%. These results suggest that normalization is a simple yet effective preprocessing step for enhancing property similarity-based blocking.

5.2.3 Blocking Key Selection Criterion. An essential component of BKAFI is the selection of blocking key functions. This step determines the subset of features F_B used to construct the similarity

Table 4: PC@k with and without normalization.

Dataset	Normalization	PC@1	PC@5	PC@20
$\text{Hague}_{\text{small}}$	Yes	80.6	88.9	93.1
	No	29.5	67.4	88.1
$\text{Hague}_{\text{large}}$	Yes	71.1	81.0	87.4
	No	59.2	78.1	86.3
PDM	Yes	16.3	31.0	43.8
	No	14.6	28.8	41.2

space for nearest neighbor retrieval. We propose to use the top-$|F_B|$ features according to their importance scores extracted from the trained model $\tilde{\theta}$. We also explore a statistical selection strategy, where features are ranked based on the standard deviation of their corresponding property ratios across matching pairs, in ascending order. Intuitively, features with lower standard deviation, indicating more stable ratios between matched objects, might be more informative for identifying true matches.

Table 5: PC@k over different blocking key criteria.

Dataset	Selection Criterion	PC@1	PC@5	PC@20
$\text{Hague}_{\text{small}}$	Feature Importance	80.6	88.9	93.1
	Std	40.2	64.3	80.6
$\text{Hague}_{\text{large}}$	Feature Importance	71.1	81.0	87.4
	Std	29.9	48.6	66.1
PDM	Feature Importance	16.3	31.0	43.8
	Std	11.4	22.1	35.3

Table 5 presents PC@k of both selection strategies. For $Hague_{small}$, feature importance consistently outperforms standard deviation across all k values. This dominance is even more evident for $Hague_{large}$, particularly at lower k: PC@1 improves from 29.9% (standard deviation) to 71.1% (feature importance). A similar trend appears on the *PDM* dataset, where PC@1 improves from 11.4% to 16.3%, and PC@5 from 22.1% to 31.0%. While standard deviation may serve as a lightweight, training-free proxy for feature selection, especially in smaller datasets, feature importance provides robust performance, particularly in larger and complex settings.

5.2.4 Effect of Search Space Size ($|F_B|$). A key parameter of BKAFI is the number of selected blocking key features ($|F_B|$). This value determines the dimensionality of the property space in which nearest neighbor retrieval is performed. At runtime, only the top $|F_B|$ properties are used to construct the vector representations of candidate and index objects.

PC@k vs. Search Space Size ($|F_B|$): Figure 6 shows how PC@k varies with $|F_B|$ for $k \in \{1, 5, 20\}$ over $Hague_{large}$. We observe a steep increase in PC as $|F_B|$ grows from 1 to 2–4, with performance peaking early. In all cases, the highest PC is achieved using a compact feature subset ($|F_B| = 3$), after which PC steadily declines and stabalizes at $|F_B| \approx 14$.

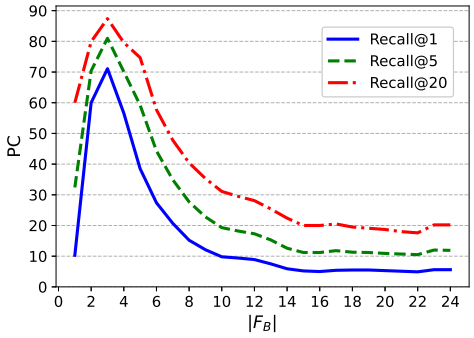


Figure 6: PC@k vs. search space size $|F_B|$ for $k \in \{1, 5, 20\}$ across $\text{Hague}_{\text{large}}$.

This trend can be attributed to the nature of the blocking strategy: while incorporating a small number of discriminative properties improves candidate filtering, adding too many properties leads to over-restriction of the search space. As the size of a blocking key increases, fewer candidate pairs survive the filter, resulting in PC loss. This is especially pronounced for PC@1, where early elimination of good candidates becomes more detrimental. These results highlight the importance of a compact and selective property space in blocking. An aggressive property expansion harms PC due to reduced overlap, reinforcing the value of principled feature selection in the BKAFI framework.

Execution Time vs. Search Space Size ($|F_B|$) and Cardinality: Figure 8 presents the execution time as a function of the dataset cardinality (the cross product of he candidate and index set), for three different $|F_B|$ values. The cardinality values shown on the X-axis refer to four variants of *The Hague* dataset: the previously defined *small* and *large* versions (see Table 2), as well as two additional variants created specifically for this analysis. Each subplot corresponds to a different value of $|F_B| \in \{3, 15, 24\}$, which dictates the dimensionality of the sub-property vectors.

We observe that runtime increases with both data volume and feature dimensionality, as expected. However, the rate of increase is moderate for lower-dimensional setups (e.g., $|F_B| = 3$), where even large-scale blocking remains computationally efficient. For the $|F_B| = 24$ setting shows a steep rise in execution time for the largest dataset, exceeding 50 seconds. These findings reinforce that BKAFI efficiently scales when operating in compact subspaces, a design supported by our earlier results showing that high recall can be achieved using a small number of features (see Figure 6).

Figure 8 presents a more detailed analysis over all possible $|F_B|$ values, for the small and large variants of *The Hague* dataset. Specifically As expected, runtime generally increases with $|F_B|$, reflecting the added cost of higher-dimensional similarity search and candidate filtering. This growth is most prominent for the large variant, where execution time rises sharply as $|F_B|$ increases. The observed trend reinforces the conclusions from Figure 6: near-optimal PC is achieved with a small number of features, which also yields faster runtime. That means, BKAFI benefits from compact feature spaces both in accuracy and scalability.

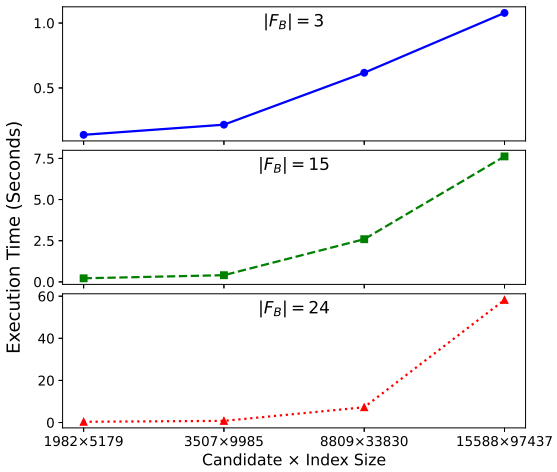


Figure 7: Execution time vs. cardinality of *The Hague* dataset variants, with different search space size values ($|F_B|$).

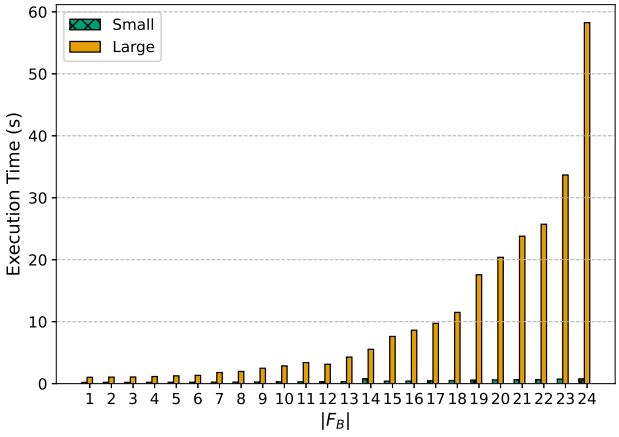


Figure 8: Execution time vs. $|F_B|$ for *The Hague* datasets.

We note that the plot exhibits occasional irregularities, *e.g.*, small dips or spikes in execution time for certain $|F_B|$ values. These fluctuations likely stem from implementation-level effects such as system caching, multithreading variability, or how the underlying similarity structures are partitioned in memory. Despite these, the overall trend remains clear: smaller $|F_B|$ leads to significantly more efficient blocking.

To summarize, our empirical analysis shows that low $|F_B|$ values not only achieve higher recall, but also lead to lower search times, making them preferable both in terms of effectiveness and efficiency.

5.2.5 Mutual Registration Accuracy for Post Disaster Management: In post-disaster scenarios such as the one captured by the PDM dataset (Section 5.1.1), decision-making must account for strict resource constraints as computational resources are often limited due to urgency, infrastructure damage, or cost considerations [50, 82]. Therefore, data-driven prioritization is critical. The PDM literature uses registration-based analysis [72] and in particular, mutual registration accuracy (MRA), as a decision-making tool. Basically, MRA reports on the success of identifying same real-world entities in a symmetric manner, both from the indexed dataset to the candidate dataset and vice versa, given limited matching resources. In a clean-clean setting (see Section 5.3.3 for the impact of contamination on the analysis) where $D^C \subset D^I$ MRA under budget constraints is translated into PC (see Section 5.1.3).

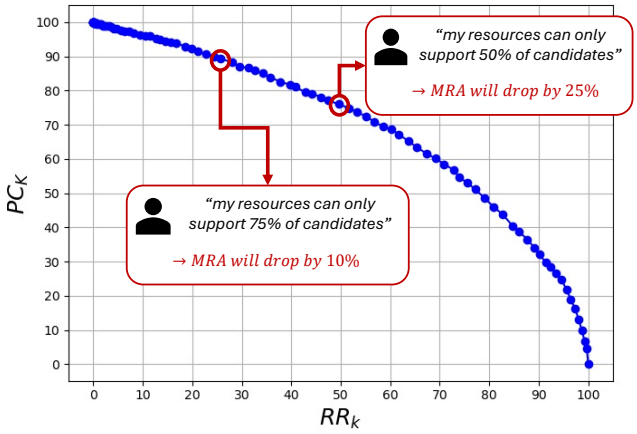


Figure 9: PC_k vs. RR_k on the PDM dataset, showing how threshold similarity-based pruning affects recall.

BKAFI has an optional pruning mechanism (see Section 3.4.1) to accommodate resource constraints requiring smaller candidate sets. In this context, we use the following two metrics. The *relative reduction ratio* (RR_k) measures the proportion of comparisons saved relative to the original blocking size: $RR_k = 1 - \frac{|C|}{k \cdot |D^C|}$. The *relative pair completeness* (PC_k) captures the pair completeness after pruning, normalized by the unpruned value: $PC_k = \frac{\text{PC}(C)}{\text{PC}(k)}$, where $\text{PC}(C)$ and $\text{PC}(k)$ denote the number of true matches after and before pruning, respectively. These metrics help in estimating the MRA (recall in this case) loss with respect to constraints stated by disaster management experts, *e.g.*, being able to handle only 75% of the original candidate set size.

Figure 9 illustrates this tradeoff for our case study (the PDM dataset). Each point corresponds to a different similarity threshold, tracing the curve of achieved PC_k as a function of RR_k. Minimal pruning (*i.e.*, low RR_k) retains nearly all true matches, resulting in near-perfect relative recall. Base on the figure, we can observe that if an expert wants to remove 25% of the candidate pairs due to different constraints, the MRA (recall) drops by only ~10%. In circumstances that demand retaining only half of the candidates, the MRA (recall) drops by ~25%.

5.3 Matching Evaluation

In this section, we evaluate the matching component of 3dSAGER. We begin by comparing its performance with baseline methods (Section 5.3.1). Next, we assess different machine learning models used as the backbone of 3dSAGER (Section 5.3.2). Finally, we investigate performance over different contamination levels in Section 5.3.3.

Finally, we analyze the phenomenon of systematic discrepancy in Section 5.3.5.

5.3.1 3dSAGER VS. Baselines. We compare 3dSAGER to baselines. **Performance:** Table 6 reports the matching performance of 3dSAGER and ViT-based baselines across three datasets. 3dSAGER consistently achieves the highest scores across all settings. On $Hague_{small}$ and $Hague_{large}$, it reaches perfect precision with F1 scores of 93.0 and 92.8, respectively, outperforming both *ViT-B/32* and *ViT-L/14*. On the *PDM* dataset, 3dSAGER maintains strong performance ($F1 = 89.6$), while *ViT-B/32* degrades significantly ($F1 = 63.9$). Results for *ViT-L/14* on this dataset are omitted, as the model failed to converge, likely due to insufficient signal in the polygon mesh input and sensitivity to domain-specific variance.

Table 6: Matching performance across datasets.

Dataset	Model	Pr.	Re.	F1
$Hague_{small}$	3dSAGER	**100.0**	**87.0**	**93.0**
	ViT-B/32	92.4	85.2	88.6
	ViT-L/14	94.9	80.4	87.0
$Hague_{large}$	3dSAGER	**100.0**	**86.6**	**92.8**
	ViT-B/32	88.4	83.8	86.1
	ViT-L/14	87.6	83.2	85.3
PDM	3dSAGER	**94.2**	**85.5**	**89.6**
	ViT-B/32	66.6	61.5	63.9
	ViT-L/14	-	-	-

Table 7: Runtime and space complexity on $Hague_{large}$.

Model	Vector Gen. (s) Train / Test	Train (s)	Inf. (s)	Model (MB)	# Params / Nodes
3dSAGER (Bagging)	446.5 / 270.7	511.5	0.55	30.88	438,746
ViT-B/32	–	14,421	529	343.5	87,915,009
ViT-L/14	–	53,055	2,402	1,188	304,064,769

Complexity: Table 7 compares the runtime (in seconds) and model complexity of 3dSAGER (using the best-performing *Bagging* model) against ViT-based baselines. Experiments were conducted on the $Hague_{large}$ dataset using a fixed random seed, with 60,393 training pairs and 52,600 test pairs. For 3dSAGER , we report the time required to generate property vectors for both splits. This step is unique to our pipeline and thus omitted for the baselines. Feature vector construction is computationally negligible and not reported separately. The training time for 3dSAGER includes grid search over a predefined hyperparameter space, meaning the actual per-configuration training time is substantially lower.

For 3dSAGER, Evaluation is extremely fast, taking only a fraction of a second for all test pairs. The learned model has a compact footprint with around 439K nodes and a file size of just 30.88MB. For the ViT-based models, we report the cumulative training time across all epochs. In our setup, embeddings are recomputed in each epoch (8 for ViT-B/32 and 5 for ViT-L/14), though in practice these can be cached to reduce overhead. Inference time refers to embedding generation and similarity scoring. ViT-B/32 and ViT-L/14 are significantly larger in size, with 88M and 304M parameters respectively. We do not account for the image rendering stage required to extract mesh images from CityJSON files for these baselines, as it is performed once and remains constant across runs.

The results highlight the efficiency advantage of 3dSAGER. Training and inference are at least an order of magnitude faster than ViT-based models, with model size over 10× smaller than *ViT-B/32* and 40× smaller than *ViT-L/14*. Property vector generation, while non-negligible, is a one-time cost that scales linearly with the number of objects, making it suitable for large-scale use. In contrast, *ViT* models require expensive fine-tuning and retain high inference overhead even with cached embeddings. While 3dSAGER consistently outperforms ViT-based baselines in matching accuracy, its key strength lies in its lightweight design, combining fast execution with a compact footprint. Together with the efficient BKAFI strategy (Section 5.2.1), these qualities make 3dSAGER a scalable and practical solution for real-world geospatial ER at scale.

5.3.2 Backbone Model Comparison. Table 8 compares the performance of various machine learning models used within 3dSAGER for $Hague_{small}$ and *PDM*. On $Hague_{small}$, *Bagging* achieves the highest F1 score (93.0), with perfect precision, while *XGBoost* and *Gradient Boosting* also perform strongly, with F1 scores above 89.5. On the more challenging, real-world, *PDM* dataset, *XGBoost* yields the best overall performance ($F1 = 89.6$), followed by *Gradient Boosting* and *Bagging*. *MLP* underperforms on both datasets, particularly in terms of recall, suggesting that neural models are less suited to the low-dimensional, structured feature space of our setting. Overall, these results highlight the effectiveness and robustness of ensemble tree-based models for geospatial ER for 3D objects.

Table 8: Matching performance of different ML models in 3dSAGER across datasets.

Model	$Hague_{small}$			PDM		
	P	R	F1	P	R	F1
Rand. Forest	89.5	87.8	88.7	**95.9**	80.0	87.2
AdaBoost	85.9	85.7	85.8	94.2	80.3	86.7
Gr. Boosting	89.8	**89.3**	89.5	94.7	84.1	89.1
Bagging	**100.0**	87.0	**93.0**	95.6	82.4	88.5
XGBoost	93.6	91.2	92.4	94.2	**85.5**	**89.6**
MLP	62.6	68.2	65.2	93.4	56.4	70.0

5.3.3 3dSAGER Over Contaminated datasets. To evaluate the robustness of 3dSAGER in settings where the dataset is contaminated, we simulate controlled contamination by introducing cross-source reference swaps into the $Hague_{small}$ dataset. This is done by swapping a percentage of items between the candidate and index sets, in accordance with a predefined contamination level, varying from 0.05 to 0.5 in increments of 0.05. This setup weakens the systematic discrepancy assumption, as candidate-index alignment becomes noisier, a condition that mirrors post-disaster urban reconstruction scenarios where data sources may be heterogeneous or partially

overlapping. We evaluate the F1 score for three classifiers (Random Forest, Bagging, and XGBoost) as the backbone matcher.

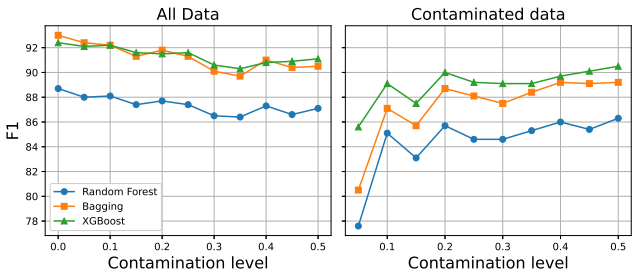


Figure 10: Robustness of 3dSAGER to increasing contamination levels on the $Hague_{small}$ dataset. Left: overall F1. Right: F1 over contaminated samples only.

Figure 10 (left) provides the overall F1 score across all test samples as a function of contamination level. Increasing contamination degrades overall performance due to the difficulty of learning consistent matching patterns under noisy alignment. Still, 3dSAGER exhibits robust behavior, with all backbone models maintaining a relatively slight performance drop (*e.g.*, XGBoost decreases by only 1.3, from 92.4 at 0% contamination to 91.1 at 50%).

When isolating performance over the contaminated portion of the test set (Figure 10, right), we observe the opposite trend: F1 improves as contamination increases. This suggests that as the model is exposed to more contaminated examples during training, it becomes better equipped to identify such cases at test time. For all backbone models, 3dSAGER stabilizes after a relatively small amount of contamination, indicating its capacity to adapt quickly to noisy conditions.

5.3.4 Dirty-Clean Setting: Mixed-Source Transferability Analysis. To further challenge our framework, we depart from the clean-clean assumption by introducing a mixed-source setting in which the candidate and index sets are no longer strictly disjoint. Specifically, we select objects from the candidate set that have known matches in the index set and move their corresponding matches into the candidate set. This process is repeated with increased contamination level ranging from 0.05 to 0.5. By the, we introduce ambiguity into the matching structure, as some entities now appear in the same source (candidate set). Beyond simulating noisy alignment, this setup also serves to examine the *transferability* of the trained matcher. Here, the pipeline is executed entirely within the contaminated candidate set, and the resulting matcher is then applied to the cross-source test set. This allows us to assess whether knowledge learned in a noisy, within-source scenario can be effectively transferred to the standard cross-source matching setting.

Figure 11 summarizes the results. All three backbone matchers experience only a slight drop in F1 when moving from the clean-clean setting (contamination level 0) to the highest contamination level (0.5), indicating that the transfer from the noisy training environment to the clean test setting is largely successful. The Random Forest matcher shows a gradual decline from about 88.7 to 87.8, whereas Bagging and XGBoost maintain are relatively consistent, with no clear sensitivity to the contamination level. These findings

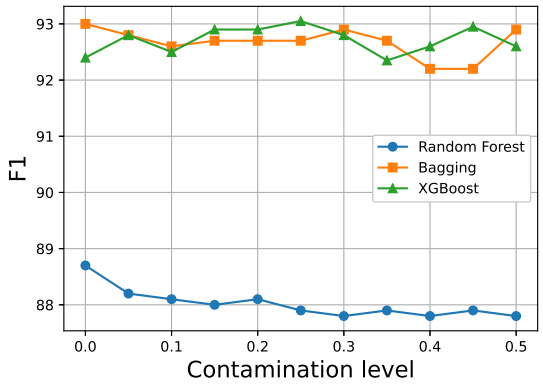


Figure 11: Performance (F1 score) of Random Forest, Bagging, and XGBoost backbone matchers in the dirty-clean setting across different contamination levels.

confirm our initial expectation that contamination in training leads to only minor degradation in clean-test performance.

5.3.5 Systematic Discrepancy Discussion. Recall that (ϵ, δ)-systematic discrepancy (Definition 4) characterizes how consistently a geometric property exhibits a stable ratio r_g across matching pairs. Here, ϵ defines the tolerance around r_g, and δ denotes the fraction of matches exceeding this tolerance. We next qualitatively explore this behavior by analyzing how δ varies with ϵ for several top-ranked properties (based on feature importance scores from a single run on $Hague_{large}$ using the best-performing matcher).

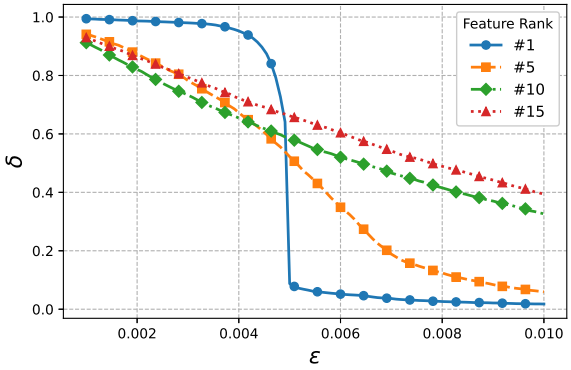


Figure 12: (ϵ, δ) curves for selected geometric properties (indexed by importance rank). Lower curves reflect stronger systematic discrepancy.

Figure 12 shows (ϵ, δ) plots for four representative geometric properties, ranked by feature importance, first, fith, etc. Each curve reflects the trade-off between tolerance (ϵ) and the proportion of matching pairs that violate the discrepancy condition (δ). A sharper drop in δ (*e.g.*, 1 and 5) indicates properties with strong systematic patterns, meaning that most matching pairs follow a consistent ratio, even under tight ϵ bounds. In contrast, smoother curves (e.g.,

10 and 15) reveal properties with higher dispersion and weaker consistency across sources. These trends support the use of systematic discrepancy correction for select properties, especially those with low δ under small ϵ values.

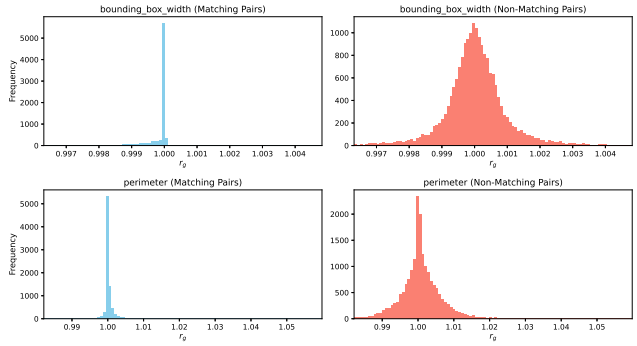


Figure 13: Ratio distributions for top features among matching vs. non-matching pairs.

To further elucidate the meaning of ratio distribution, we present Figure 13 that compares the ratio distributions of the two top-ranked geometric properties – `bounding_box_width` and `perimeter` – between matching and non-matching pairs. As seen on the left, the ratios for matching pairs are highly concentrated around 1.0, indicating strong consistency. Conversely, the distributions for non-matching pairs (right) are notably broader and more irregular, lacking the sharp central peak. This highlights a key motivation behind our geometric featurization: for certain properties, matching pairs exhibit distinctive and predictable proportional patterns that are absent in non-matching pairs. This disparity enables effective discrimination by the downstream model.

6 RELATED WORK

ER is among the most widely-studied data integration tasks, aimed at identifying correspondences between data records that refer to the same real-world entity [15, 16, 27, 77]. ER typically consists of two main steps: blocking and matching. The former reduces the number of candidate comparisons by grouping likely matches based on predefined attributes or heuristics [45, 55, 57]. The latter determines whether each candidate pair is indeed a true match. Solutions for ER have evolved over the years, starting from relying on string similarity between textual elements [33, 34, 44, 48] and probabilistic approaches [22], and followed by rule-based methods [68, 70]. As the complexity of data sources increased and the volume of records grew, ER methods adopted machine learning techniques, offering greater flexibility in modeling complex relationships in data [7, 41]. Recent approaches increasingly rely on deep learning [36, 53], especially leveraging pretrained language models for textual data [46, 47, 60, 61, 76]. We now provide a more detailed discussion on Geospatial ER, 3D Data Featurization and, given the novelty of BKAFI, blocking techniques.

Geospatial ER: Geospatial ER and its extensions exemplify use cases that extend beyond tabular or textual data sources [38, 66]. Some existing works focus on matching objects based on geographic information, (*i.e.,* coordinates), obtained from sensors [2, 71]. Another line of works suggests to incorporate signals such as textual location descriptions. Balsebre *et al.* [4] presented Geo-ER, in which graph neural networks generate a unified entity representations based on object textual attributes along with their location. Barret *et al.* [5] introduced GeoAlign, a system that allows users to customize the desired similarity measures and attributes to be used for the matching decision. Closest to our work is the one by Shah *et al.* [67] that use polygons in 2D space, rather than a single coordinate, to model an object. They describe spatial relationships between entities based on distance measures, as well as spatial proximity-based features, such as overlaps and containment, enabling more nuanced matching decisions. All existing works operate in 2D space and rely on flat spatial representations, such as centroids, bounding boxes, or polygon overlays, along with metadata like textual descriptions. While these representations suffice for relatively simple geospatial objects, they fall short in capturing the structural complexity of real-world 3D environments. We use 3D polygon mesh representations, encoding detailed geometric and topological information about spatial entities. These richer representations enable the extraction of intrinsic geometric properties that are inaccessible in 2D. The rich representation motivates a shift in matching paradigm: from location-based or textual similarity to similarity in a geometric property space derived directly from raw 3D representation.

3D Data Featurization: Several works explore the use of geometric features over 3D spatial building data, primarily in the context of urban modeling. For example, Labetski *et al.* [42] review a wide range of applications in which 3D building metrics are used for tasks like energy consumption estimation and urban morphology analysis, motivating the use of 3D metrics in reducing ambiguities and supporting city complexity parametrization. Jaljolie *et al.* [32] propose a spatial data structure for 3D land management, introducing hierarchical 3D primitives and topological reasoning for volumetric parcels. They also develop a topological-based algorithm to classify spatial relationships among complex 3D geometries [31]. While these works demonstrate the utility of geometric and topological 3D features in land administration and urban planning, they do not address ER or operate directly on raw polygon mesh inputs. Our work fills this gap by generating geometric featurization from raw 3D, which dictates our novel problem formulation. This connection between raw 3D data and derived feature space is key to the design of BKAFI, which leverages this structure to enable efficient and interpretable candidate generation.

Blocking Techniques: Traditional blocking methods rely on cues from structured data, *e.g.*, q-grams [39, 54, 58]. In our case, such signals are absent, as inputs are unstructured polygon meshes. Modern approaches increasingly rely on embedding-based representations to capture semantic similarity between entities [11, 73, 81]. These embeddings project entities into a continuous vector space, where geometric distance serves as a proxy for similarity, a notion that has also been adopted in related ER sub-tasks, such as active learning [25, 30]. This vector-based reasoning is also central to unsupervised pipelines such as ZeroER [78], which construct similarity-based feature vectors over entity pairs to enable learning without labeled data. Existing works propose using variations of contrastive learning to train a blocker [11, 30, 73]. These approaches

operate on tabular data with textual attributes and typically employ attention-based encoders to generate record embeddings. The training objective is to pull embeddings of matching pairs closer in the vector space while pushing apart non-matching pairs. In contrast, Bilenko *et al.* [7] introduce adaptive blocking functions based on data-derived predicates. In geospatial ER, blocking methods often assume that references to the same entity lie near each other in the coordinate space, leading to pointwise similarity techniques [4, 38, 40]. However, in the absence of coordinate systems or in the presence of spatial distortions, performance significantly deteriorates, motivating the need for alternative strategies. BKAFI departs from prior trainable blocking methods in several key ways. First, while existing approaches learn blocking functions from raw textual attributes or embedding distances (*e.g.*, using contrastive learning), BKAFI leverages a trained matcher to identify the most informative geometric properties. Blocking keys are selected based on feature importance scores, focusing on a meaningful, learned subspace. Second, while traditional geospatial approaches assume spatial proximity based on flat representations such as object centroids, BKAFI operates entirely in the geometric feature space derived from 3D polygon meshes. This design allows BKAFI to remain effective in settings where coordinate systems are unreliable or unavailable, while also ensuring both efficiency and interpretability.

7 CONCLUSIONS

In this work, we introduce 3dSAGER, an end-to-end pipeline for geospatial entity resolution over 3D objects. We levarage a novel featurization process that captures the geometric characteristics of 3D polygon meshes and serves as the foundation for both the matching model and our novel blocking technique, BKAFI. To support evaluation and encourage future research, we curated a new benchmark for geospatial ER over 3D objects. Empirical results on the new benchmark, alongside a proprietary dataset, demonstrate the effectiveness and efficiency of our approach in addressing the challenges of both components of the geospatial ER pipelines

In future work, we plan to experiment with datasets exhibiting varying levels of detail, and expanding the design of the pairwise feature vectors to capture a broader range of spatial characteristics. In particular, we aim to explore reference semantics using a graph representation learning framework capable of modeling richer structural information embedded in 3D models, such as windows and doors within walls and roofs. Another future work direction will examine properties and guarantees of BKAFI, with the aim of extending its applicability to other vector-search applications.

ACKNOWLEDGEMENTS

We acknowledge the use of OpenAI's ChatGPT for assistance in the preparation of this manuscript. Specifically, the tool was used for rephrasing parts of the text, converting raw experimental results into LaTeX tables, and generating Python code for plotting figures.

REFERENCES

[1] Niall M. Adams, Martin Field, Erol Gelenbe, David J. Hand, Nicholas R. Jennings, David S. Leslie, David Nicholson, Sarvapali D. Ramchurn, Stephen J. Roberts, and Alex Rogers. 2008. The ALADDIN project : intelligent agents for disaster management. https://api.semanticscholar.org/CorpusID:18836574

[2] Sandrine Balley, Christine Parent, and Stefano Spaccapietra. 2004. Modelling geographic data with multiple representations. *International Journal of Geographical Information Science* 18, 4 (2004), 327–352.

[3] Daniel Balouek-Thomert, Eddy Caron, Laurent Lefevre, and Manish Parashar. 2022. Towards a methodology for building dynamic urgent applications on continuum computing platforms. In *2022 First Combined International Workshop on Interactive Urgent Supercomputing (CIW-IUS)*. IEEE, 1–6.

[4] Pasquale Balsebre, Dezhong Yao, Gao Cong, and Zhen Hai. 2022. Geospatial entity resolution. In *Proceedings of the ACM Web Conference 2022*. 3061–3070.

[5] Nelly Barret, Fabien Duchateau, Franck Favetta, and Ludovic Moncla. 2019. Spatial entity matching with geoalign (demo paper). In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 580–583.

[6] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 9 (1975), 509–517.

[7] Mikhail Bilenko and Raymond J Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 39–48.

[8] Vasileios Bouzas, Hugo Ledoux, and Liangliang Nan. 2020. Structure-aware building mesh polygonization. *ISPRS Journal of Photogrammetry and Remote Sensing* 167 (2020), 432–442.

[9] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24 (1996), 123–140.

[10] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[11] Alexander Brinkmann, Roee Shraga, and Christina Bizer. 2024. Sc-block: Supervised contrastive blocking within entity resolution pipelines. In *European Semantic Web Conference*. Springer, 121–142.

[12] FENG Changyong, WANG Hongyue, LU Naiji, CHEN Tian, HE Hua, LU Ying, and M TU Xin. 2014. Log-transformation and its implications for data analysis. *Shanghai archives of psychiatry* 26, 2 (2014), 105.

[13] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[14] F Chiabrando, F Giulio Tonolo, and Andrea Lingua. 2019. Uav direct georeferencing approach in an emergency mapping context. the 2016 central Italy earthquake case study. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2019), 247–253.

[15] Peter Christen and Peter Christen. 2012. *The data matching process*. Springer.

[16] William W Cohen and Jacob Richman. 2002. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 475–480.

[17] Yue Deng, An Luo, Jiping Liu, and Yong Wang. 2019. Point of interest matching between different geospatial datasets. *ISPRS International Journal of Geo-Information* 8, 10 (2019), 435.

[18] Jürgen Döllner and Benjamin Hagedorn. 2007. Integrating urban GIS, CAD, and BIM data by service-based virtual 3D city models. In *Urban and regional data management*. CRC Press, 157–170.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR* (2021).

[20] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. 2006. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering* 19, 1 (2006), 1–16.

[21] Anna Erving, Petri Rönnholm, and Milka Nuikka. 2009. Data integration from different sources to create 3D virtual model. *3D-ARCH* 2009 (2009), 3D.

[22] Ivan P Fellegi and Alan B Sunter. 1969. A theory for record linkage. *J. Amer. Statist. Assoc.* 64, 328 (1969), 1183–1210.

[23] Yoav Freund, Robert E Schapire, et al. 1996. Experiments with a new boosting algorithm. In *icml*, Vol. 96. Citeseer, 148–156.

[24] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[25] Bar Genossar, Avigdor Gal, and Roee Shraga. 2023. The battleship approach to the low resource entity matching problem. *Proceedings of the ACM on Management of Data* 1, 4 (2023), 1–25.

[26] Lise Getoor and Christopher P Diehl. 2005. Link mining: a survey. *Acm Sigkdd Explorations Newsletter* 7, 2 (2005), 3–12.

[27] Lise Getoor and Ashwin Machanavajjhala. 2012. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment* 5, 12 (2012), 2018–2019.

[28] Gerhard Gröger and Lutz Plümer. 2012. CityGML–Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing* 71 (2012), 12–33.

[29] Thomas Holzmann, Martin R Oswald, Marc Pollefeys, Friedrich Fraundorfer, and Horst Bischof. 2017. Plane-based surface regularization for urban 3d construction. In *Proceedings 28th British Machine Vision Conference, 2017 (BMVC)*. 1–9.

[30] Arjit Jain, Sunita Sarawagi, and Prithviraj Sen. 2021. Deep indexed active learning for matching heterogeneous entity representations. *Proceedings of the VLDB Endowment* 15, 1 (2021), 31–45.

[31] Ruba Jaljolie, Kirsikka Riekkinen, and Sagi Dalyot. 2021. A topological-based approach for determining spatial relationships of complex volumetric parcels in land administration systems. *Land Use Policy* 109 (2021), 105637.

[32] Ruba Jaljolie, Peter Van Oosterom, and Sagi Dalyot. 2018. Spatial data structure and functionalities for 3d land management system implementation: Israel case study. *Isprs international journal of Geo-information* 7, 1 (2018), 10.

[33] Matthew A Jaro. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *J. Amer. Statist. Assoc.* 84, 406 (1989), 414–420.

[34] Matthew A Jaro. 1995. Probabilistic linkage of large public health data files. *Statistics in medicine* 14, 5-7 (1995), 491–498.

[35] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[36] Muhammad Ebraheem Saravanan Thirumuruganathan Shafiq Joty and Mourad Ouzzani Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *Proceedings of the VLDB Endowment* 11, 11 (2018).

[37] Dušan Jovanović, Stevan Milovanov, Igor Ruskovski, Miro Govedarica, Dubravka Sladić, Aleksandra Radulović, and Vladimir Pajić. 2020. Building virtual 3D city model for smart cities applications: A case study on campus area of the university of novi sad. *ISPRS International Journal of Geo-Information* 9, 8 (2020), 476.

[38] Hyunmo Kang, Vivek Sehgal, and Lise Getoor. 2007. Geoddupe: A novel interface for interactive entity resolution in geospatial data. In *2007 11th International Conference Information Visualization (IV'07)*. IEEE, 489–496.

[39] Batya Kenig and Avigdor Gal. 2013. MFIBlocks: An effective blocking algorithm for entity resolution. *Information Systems* 38, 6 (2013), 908–926.

[40] Mohammad Khodizadeh-Nahari, Nasser Ghadiri, Ahmad Baraani-Dastjerdi, and Jörg-Rüdiger Sack. 2021. A novel similarity measure for spatial entity resolution based on data granularity model: Managing inconsistencies in place descriptions. *Applied Intelligence* 51, 8 (2021), 6104–6123.

[41] Pradap Konda et al. 2016. Magellan: Toward building entity matching management systems. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1197–1208.

[42] Anna Labetski, Stelios Vitalis, Filip Biljecki, Ken Arroyo Ohori, and Jantien Stoter. 2023. 3D building metrics for urban morphology. *International Journal of Geographical Information Science* 37, 1 (2023), 36–67.

[43] Hugo Ledoux, Ken Arroyo Ohori, Kavisha Kumar, Balázs Dukai, Anna Labetski, and Stelios Vitalis. 2019. CityJSON: A compact and easy-to-use encoding of the CityGML data model. *Open Geospatial Data, Software and Standards* 4, 1 (2019), 1–12.

[44] Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.

[45] Bo-Han Li, Yi Liu, An-Man Zhang, Wen-Huan Wang, and Shuo Wan. 2020. A survey on blocking technology of entity resolution. *Journal of Computer Science and Technology* 35 (2020), 769–793.

[46] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment* 14, 1 (2020), 50–60.

[47] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, Jin Wang, Wataru Hirota, and Wang-Chiew Tan. 2021. Deep entity matching: Challenges and opportunities. *Journal of Data and Information Quality (JDIQ)* 13, 1 (2021), 1–17.

[48] Dekang Lin et al. 1998. An information-theoretic definition of similarity. In *ICML*, Vol. 98. 296–304.

[49] Willard G Manning and John Mullahy. 2001. Estimating log models: to transform or not to transform? *Journal of health economics* 20, 4 (2001), 461–494.

[50] Thomas Manzini, Priyankari Perali, Raisa Karnik, and Robin Murphy. 2024. Crasar-u-droids: A large scale benchmark dataset for building alignment and damage assessment in georectified suas imagery. *arXiv preprint arXiv:2407.17673* (2024).

[51] Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. 2023. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems* 36 (2023), 76336–76369.

[52] Anthony Morana, Thomas Morel, Bilal Berjawi, and Fabien Duchateau. [n. d.]. Geobench: a geospatial integration tool for building a spatial entity matching benchmark. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*.

[53] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*. 19–34.

[54] George Papadakis, George Alexiou, George Papastefanatos, and Georgia Koutrika. 2015. Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data. *Proceedings of the VLDB Endowment* 9, 4 (2015), 312–323.

[55] George Papadakis, Ekaterini Ioannou, Claudia Niederée, Themis Palpanas, and Wolfgang Nejdl. 2011. Eliminating the redundancy in blocking-based entity resolution methods. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*. 85–94.

[56] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2019. A survey of blocking and filtering techniques for entity resolution. *arXiv preprint arXiv:1905.06167* (2019).

[57] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys (CSUR)* 53, 2 (2020), 1–42.

[58] Derek Paulsen, Yash Govind, and AnHai Doan. 2023. Sparkly: A simple yet surprisingly strong TF/IDF blocker for entity matching. *Proceedings of the VLDB Endowment* 16, 6 (2023), 1507–1519.

[59] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[60] Ralph Peeters and Christian Bizer. 2023. Using chatgpt for entity matching. In *European Conference on Advances in Databases and Information Systems*. Springer, 221–230.

[61] Ralph Peeters, Aaron Steiner, and Christian Bizer. 2023. Entity matching using large language models. *arXiv preprint arXiv:2310.11244* (2023).

[62] Ralph Peeters, Aaron Steiner, and Christian Bizer. 2025. Entity matching using large language models. *OpenProceedings* 2 (2025), 529–541.

[63] Ravi Peters, Balázs Dukai, Stelios Vitalis, Jordi van Liempt, and Jantien Stoter. 2022. Automated 3D reconstruction of LoD2 and LoD1 models for all 10 million buildings of the Netherlands. *Photogrammetric Engineering & Remote Sensing* 88, 3 (2022), 165–170.

[64] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PmLR, 8748–8763.

[65] Parikshit Ram and Kaushik Sinha. 2019. Revisiting kd-tree for nearest neighbor search. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining*. 1378–1388.

[66] Vivek Sehgal, Lise Getoor, and Peter D Viechnicki. 2006. Entity resolution in geospatial data integration. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*. 83–90.

[67] Setu Shah, Vamsi Meduri, and Mohamed Sarwat. 2021. GEM: An efficient entity matching framework for geospatial data. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*. 346–349.

[68] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Synthesizing entity matching rules by examples. *Proceedings of the VLDB Endowment* 11, 2 (2017), 189–202.

[69] Surendra Pal Singh, Kamal Jain, and V Ravibabu Mandla. 2013. Virtual 3D city modeling: techniques and applications. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 40 (2013), 73–91.

[70] Parag Singla and Pedro Domingos. 2006. Entity resolution with markov logic. In *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 572–582.

[71] PG Tabarro, Jacynthe Pouliot, Richard Fortier, and L-M Losier. 2017. A WebGIS to support GPR 3D data acquisition: A first step for the integration of underground utility networks in 3D city models. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2017), 43–48.

[72] Gary KL Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C Langbein, Yonghuai Liu, David Marshall, Ralph R Martin, Xian-Fang Sun, and Paul L Rosin. 2012. Registration of 3D point clouds and meshes: A survey from rigid to nonrigid. *IEEE transactions on visualization and computer graphics* 19, 7 (2012), 1199–1217.

[73] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep learning for blocking in entity matching: a design space exploration. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2459–2472.

[74] Ali Osman Ulusoy and Joseph L Mundy. 2014. Image-based 4-d reconstruction using 3-d change detection. In *European Conference on Computer Vision*. Springer, 31–45.

[75] Styliani Verykokou, Anastasios Doulamis, George Athanasiou, Charalabos Ioannidis, and Angelos Amditis. 2016. UAV-based 3D modelling of disaster scenes for Urban Search and Rescue. In *2016 IEEE International Conference on Imaging Systems and Techniques (IST)*. IEEE, 106–111.

[76] Pengfei Wang, Xiaocan Zeng, Lu Chen, Fan Ye, Yuren Mao, Junhao Zhu, and Yunjun Gao. 2022. Promptem: prompt-tuning for low-resource generalized entity matching. *arXiv preprint arXiv:2207.04802* (2022).

[77] William E Winkler. 2002. *Methods for record linkage and bayesian networks*. Technical Report. Statistical Research Division, US Census Bureau.

[78] Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. 2020. Zeroer: Entity resolution using zero labeled examples. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1149–1164.

[79] Ningli Xu, Debao Huang, Shuang Song, Xiao Ling, Chris Strasbaugh, Alper Yilmaz, Halil Sezen, and Rongjun Qin. 2021. A volumetric change detection framework using UAV oblique photogrammetry–a case study of ultra-high-resolution monitoring of progressive building collapse. *International Journal of Digital Earth* 14,

11 (2021), 1705–1720.
[80] Manzhu Yu, Chaowei Yang, and Yun Li. 2018. Big data in natural disaster management: a review. *Geosciences* 8, 5 (2018), 165.
[81] Wei Zhang, Hao Wei, Bunyamin Sisman, Xin Luna Dong, Christos Faloutsos, and Davd Page. 2020. Autoblock: A hands-off blocking framework for entity matching. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 744–752.
[82] Xiangyu Zhuo, Tobias Koch, Franz Kurz, Friedrich Fraundorfer, and Peter Reinartz. 2017. Automatic UAV image geo-registration by matching UAV images to georeferenced image data. *Remote Sensing* 9, 4 (2017), 376.