Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

SHREC 2025: Partial Retrieval Benchmark

Bart Iver van Blokland[a,*], Isaac Aguirre[b], Ivan Sipiran[b], Benjamin Bustos[c], Silvia Biasotti[d], Giorgio Palmieri[d]

[a]NTNU, Department of Computer Science, Sem Sælands vei 9, Trondheim, 7034, Trøndelag, Norway
[b]University of Chile, Department of Computer Science, Av. Beauchef 851, Santiago, 8370456, Metropolitan Region, Chile
[c]University of Chile, Department of Computer Science & IMFD, Av. Beauchef 851, Santiago, 8370456, Metropolitan Region, Chile
[d]Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes" IMATI - CNR, Via de Marini 16, Genova, 16149, Italy

ARTICLE INFO

Article history:
Received May 12, 2025

Keywords: SHREC 2025, 3D Local Shape Descriptors, ShapeBench, Benchmark

ABSTRACT

Partial retrieval is a long-standing problem in the 3D Object Retrieval community. Its main difficulties arise from how to define 3D local descriptors in a way that makes them effective for partial retrieval and robust to common real-world issues, such as occlusion, noise, or clutter, when dealing with 3D data. This SHREC track is based on the newly proposed ShapeBench benchmark to evaluate the matching performance of local descriptors. We propose an experiment consisting of three increasing levels of difficulty, where we combine different filters to simulate real-world issues related to the partial retrieval task. Our main findings show that classic 3D local descriptors like Spin Image are robust to several of the tested filters (and their combinations), but more recent learned local descriptors like GeDI can be competitive for some specific filters. Finally, no 3D local descriptor was able to successfully handle the hardest level of difficulty.

1. Introduction

Finding similar or relevant objects to a given query input is a fundamental task in multimedia databases. An exact search in this context is, in general, meaningless because two objects in the dataset are identical only in the case where they are digital copies. Two models obtained from the same source (e.g., by 3D scanning the same object twice) will result in different but similar models. In addition to retrieval, similarity search algorithms can be used to implement multimedia mining tasks such as clustering and classification. Thus, it is relevant to study effective methods for representing and searching multimedia objects.

Fig. 1: An example of a partial view from a scene. Note the missing parts on the models.

Among similarity search problems, one of particular interest is the partial retrieval on 3D models. In this task, usually the query input is a partial 3D view, and the problem is to find the corresponding part in a complete or partial 3D model or 3D scene. Figure 1 shows an example of a partial scene. The partial retrieval task is known to be difficult and complex, as previous SHREC tracks on this problem have shown [1, 2].

Practically all real-world 3D captures contain some degree of occlusion, and it is as such one of the most common chal-

*Corresponding author: bart.van.blokland@ntnu.no
e-mail: bart.van.blokland@ntnu.no (Bart Iver van Blokland), example@email.com (Bart Iver van Blokland), isaac.aguirre@ing.uchile.cl (Isaac Aguirre), isipiran@dcc.uchile.cl (Ivan Sipiran), bebustos@dcc.uchile.cl (Benjamin Bustos), silvia@ge.imati.cnr.it (Silvia Biasotti), giorgio.palmieri@ge.imati.cnr.it (Giorgio Palmieri), bart.van.blokland@ntnu.no (Bart Iver van Blokland)

lenges encountered by 3D shape retrieval and recognition methods. The advent of learning-based methods for this task has the opportunity to improve upon the state of the art, and has as of yet not received much attention from the machine learning community. Thus, a systematic benchmarking methodology on this topic is both relevant and timely. Unfortunately, testing the robustness of a given 3D shape retrieval method to various scenarios under which varying degrees of partiality occur is difficult to accomplish using real-world 3D captures. These captures inherently contain various types of noise and capturing artefacts. It is furthermore difficult to achieve quantitative results due to the time and storage requirements for such individual captures.

This SHREC track builds upon the ShapeBench benchmark introduced in previous work [3], which proposed a replicable and scalable methodology for evaluating local 3D shape descriptors. While the original work focused on controlled comparisons of descriptor robustness using synthetic variations applied exclusively to the scene object, our SHREC track significantly extends this evaluation. First, we simulate more realistic and challenging retrieval scenarios by introducing multi-filter pipelines and by applying distortions to both the model and the scene. Second, we introduce a structured notion of difficulty levels, enabling a progressive assessment of descriptor robustness. Third, we include and evaluate several new descriptors, including recent learning-based methods, and analyze their execution times under controlled geometric conditions. Finally, we adapt and optimize the benchmark infrastructure for testing Python-based methods, thus broadening accessibility and enabling the inclusion of deep learning descriptors. Together, these extensions make our benchmark a more comprehensive and realistic testbed for the partial 3D retrieval task.

Seven teams registered for this SHREC track, but only three teams submitted results for evaluation: Ivan Sipiran from U. of Chile [Team 1], Isaac Aguirre from U. of Chile [Team 2], and Bart Iver van Blokland from NTNU [Team 3].

2. The ShapeBench benchmark

This section introduces the ShapeBench benchmark, the dataset used for the benchmark, the evaluation metric, and the combinations of filters selected for this SHREC track.

2.1. The benchmark

ShapeBench [3] is a recent methodology for evaluating local 3D shape descriptors. It evaluates the ability of a descriptor to determine that two surface points are similar under various real-world conditions. These include clutter, occlusion, and noise.

The benchmark measures this by matching corresponding points on two copies of the same object (for historical reasons referred to as the "model" and "scene" object), where the aforementioned adverse conditions are simulated by modifying the scene object using a sequence of *filters*. Each filter applies a procedural modification to the object. A tested method must subsequently correctly identify matching pairs of corresponding model and scene points, where model points are hidden among a large set of random points on other objects. All objects are taken from a set of 790,635 triangle meshes from the Objaverse dataset [4].

This track instead applies filters to both objects, creating a more realistic testing environment. We further extend the benchmark by integrating support for methods implemented in the Python language, which simplifies testing methods utilizing machine learning. The estimation of occlusion and clutter has also been reworked to be faster, in some cases reducing the total execution time of a single benchmark run by several hours.

The Descriptor Distance Index (DDI) [3] is used as the primary metric to evaluate the efficacy of a given method in performing these recognition tasks. Let δ be the dissimilarity function defined over a given 3D local descriptor. Let m be the matching point in the model object, and let s be the matching point in the scene object. Given a set of R random surface points from the dataset, the DDI accumulates the number of points $r \in R$ such that $\delta(m, r) < \delta(m, s)$, i.e., the DDI counts how many random points were considered a better match, i.e., at a lower distance, for m than s, which is the known match. The final DDI score for the 3D local descriptor is the sum of all these values for all selected pairs of points (m, s). We also measure the execution times of the evaluated methods.

2.2. Filters

A filter is a transformation applied on an object. As stated, the purpose of filters is to simulate real-world issues while performing retrieval tasks on digitized objects or scenes. First, we define some terms that will be used for describing the filters:

- Support volume: The region (usually a cylinder or sphere) that contains all the shape information used to compute a local shape descriptor.
- Support radius: The size of the support volume of a local shape descriptor.
- Independent variable: The variable being varied in each filter, to test its effect on the DDI of a local descriptor.

For the evaluation of local 3D shape descriptors in this track, we use ShapeBench with combinations of the following filters:

- Occlusion: This filter chooses a random viewing direction from which the scene is viewed, and removes all geometry that is not visible from that point of view. The independent variable is the area of the remaining mesh that intersects the support volume divided by the area of the unmodified mesh intersecting the support volume. Figure 2a shows an example of the application of this filter.
- Clutter: A physics simulator randomly places objects on top of the input scene, simulating how they collide with other objects and how gravity affects them. The independent variable is the area of clutter objects that intersects the support volume, divided by the area intersecting the support volume that belongs to the object being recognized. Figure 2b shows an example of the result of this filter.
- Gaussian noise: Simulates various sources of noise introduced in the capture process. This filter displaces the position of all vertices by a distance that follows a normal

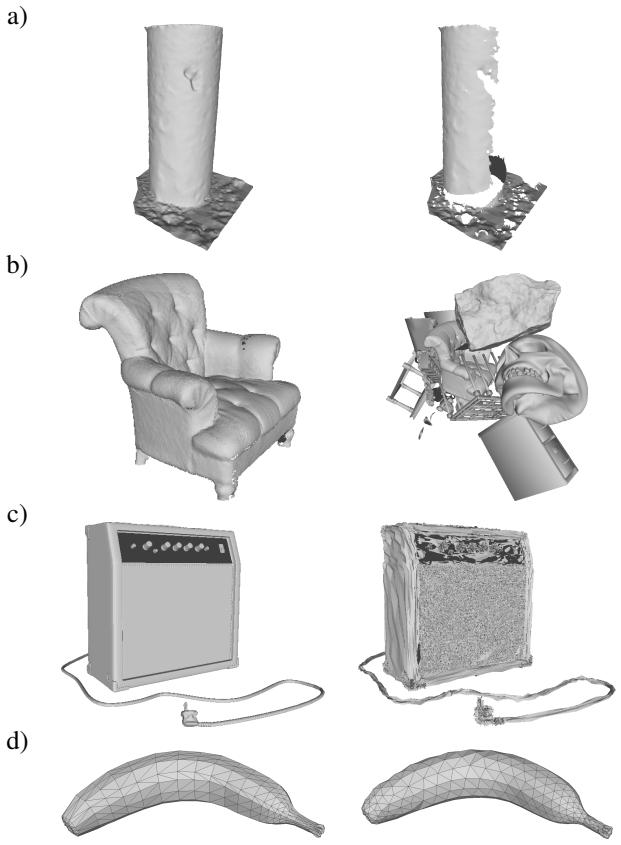


Fig. 2: Illustration of the effects of individual filters. The model object is on the left, and the scene is on the right. From top to bottom, the effects of the occlusion, clutter, Gaussian noise, and vertex perturbation filters are shown.

distribution, using a fixed value for the standard deviation. The independent variable is the standard deviation of the noise function. Figure 2c shows an example of this filter.

- Vertex Perturbation: Simulates capturing the mesh multiple times by displacing triangle vertices, while keeping the mesh's overall shape intact. The independent variable is the distance to the closest corresponding vertex in the modified mesh. An example is shown in Figure 2d.

2.2.1. Levels of difficulty

We define three different levels of increasing difficulty for the partial retrieval task. Level 1 tests common sources of matching inaccuracies in isolation. Level 2 tests combinations of these that are often observed in practical applications. Level 3 aims to present a combination of these that can be expected in captures of real-world environments. Table 1 lists the filter configuration on each experiment that is done at each difficulty level. Figure 3 visualises Experiments 4 to 8, and Figure 4 depicts the effects of applying the filters as defined in Experiment 9.

The experiments use variations of the aforementioned filters in order to reduce the dimensionality and interpretability of the results. Experiments 6, 8, and 9 apply Gaussian noise with a fixed standard deviation instead of one chosen at random. Experiment 9 also applies the clutter filter with only 2 clutter objects instead of the usual 10. Finally, when occlusion is applied to both the model and the scene, the occlusion fraction of the overlapping area is used as the independent variable.

Level	# Ex.	Filters applied on model	Filters applied on scene
Level 1	Ex. 1		Occlusion
	Ex. 2		Clutter
	Ex. 3		Gaussian noise
Level 2	Ex. 4		Occlusion + Gaussian noise
	Ex. 5	Occlusion	Occlusion
	Ex. 6	Occlusion + Fixed Gaussian noise	Occlusion + Fixed Gaussian noise
	Ex. 7	Occlusion	Occlusion + Clutter
	Ex. 8	Occlusion + Fixed Gaussian noise	Occlusion + Clutter + Fixed Gaussian noise
Level 3	Ex. 9		Occlusion + Two clutter objects + Fixed Gaussian noise + Vertex perturbation

Table 1: Levels of filtering, and their tested filter configurations

The Descriptor Distance Index (DDI) metric is used by Shapebench [3] to measure the effect of each filter configuration on the matching performance of a local 3D shape descriptor.

2.3. Execution Time

We have also extended the benchmark with a new process for measuring the execution time of a tested method. Deciding the optimal method to use for 3D shape recognition is often a balance between its matching capabilities, and its execution time. In cases where latency is essential, or processing power is limited, a faster method that is less capable may be desirable.

Recent work has predominantly measured the time to generate a single descriptor for a given surface as a function of the support radius [5, 6, 7, 8, 9], though the vertex or triangle count [10, 11, 12], or case studies [13, 14] have also been used.

Figure 5 shows observed execution times as a function of the support radius. Figure 5b demonstrates that the execution time can vary by roughly a factor of two for the same radius. This variation is caused by that the time cost for processing a point or triangle that lies within the support volume can be different to that of one which lies outside of it. The location of the support volume varies the proportion of in- and excluded geometry for a given surface, and thus the execution time. Figure 5a shows that this variation disappears when this proportion is constant, as is the case with surface points on a sphere.

Understanding the performance characteristics of a method therefore requires measuring the execution time cost of geometry inside and outside the support volume separately. This approach deviates from previous work, which has generally disregarded the cost of excluding geometry as being something all methods need to do, with the implicit assumption that this cost is approximately the same for all methods. We use synthetic meshes, which allow this proportion to be controlled.

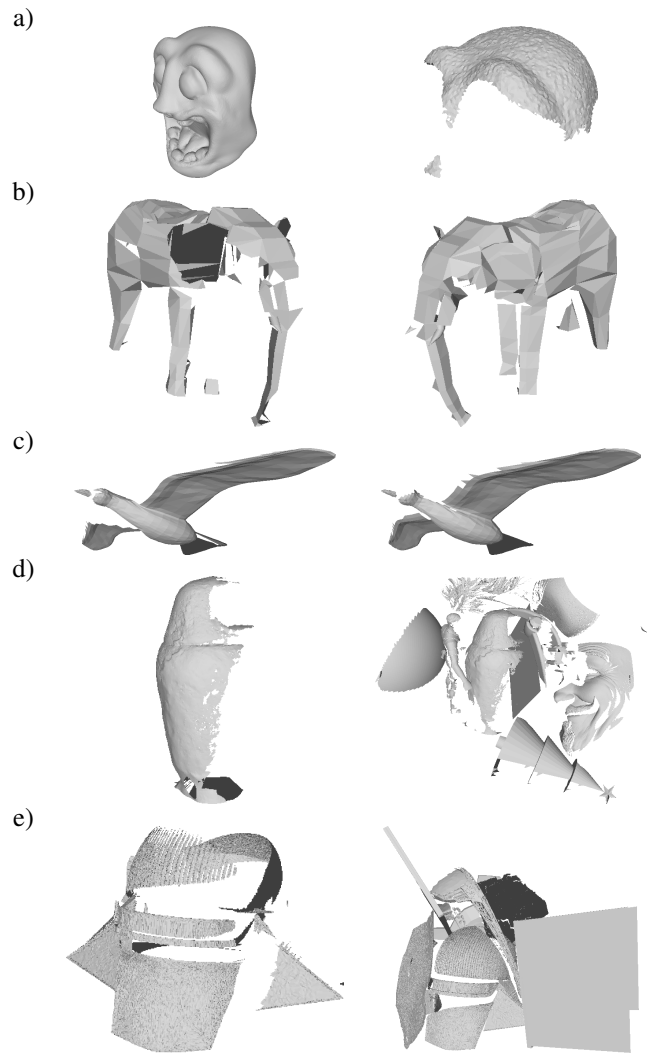
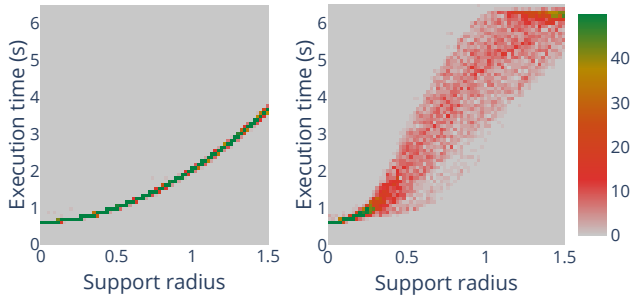


Fig. 3: Illustration of the experiments at Level 2. The model is on the left, and the scene is on the right. Refer to Table 1 for the filter configurations used. Here Figures a) to e) correspond to experiments 4 to 8, respectively.

Fig. 4: Illustration of Experiment 9, at Level 3 Occlusion + Two clutter objects + Fixed Gaussian noise + Vertex perturbation). The model is on the left, and the scene is on the right.

The first two of these (type a and b, as shown in Figure 6) place meshes at randomly chosen locations inside the support volume. What is being varied between these is the distribution of the geometry. Type a spreads it out uniformly, while type b concentrates it. Type c exclusively places geometry outside the support region, with a uniform distribution. Finally, type d (not pictured directly) uses a mesh similar to that of type a, but



(a) Sphere (b) Mesh (statue)

Fig. 5: Scatterplots showing the variation of execution times when computing the SHOT descriptor 25 times for a randomly selected vertex and support radius. Each input point cloud has 5M points. A heatmap visualisation is used to highlight clusters of in total 10k sample points.

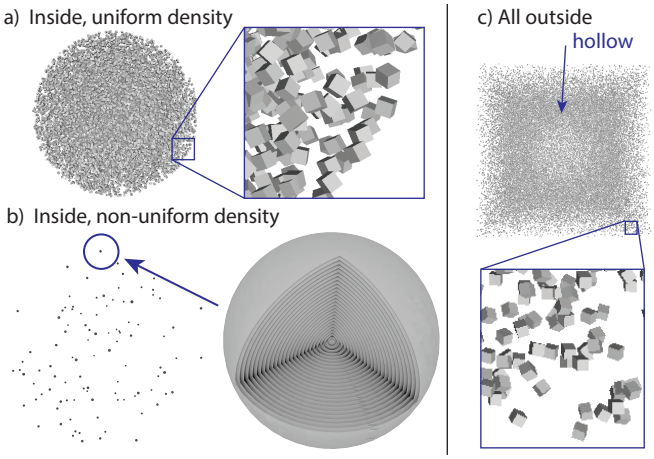


Fig. 6: Types of synthetic meshes generated by the benchmark.

does not request the method to generate any descriptors. This allows the estimation of the method's overhead, assuming its implementation does not have an explicit check for this.

For measuring the execution time itself, we limit the running of the descriptor method to a single thread. While descriptors would in a practical setting primarily be computed in parallel, we also wish to be able to compare against methods implemented in Python. These are inherently single-threaded. Boosting of the CPU was disabled, and the benchmark thread was locked to a single core through the operating system. The number of descriptors being generated at a time is fixed, and are computed in a single batch. This ensures acceleration structures (if the method uses them) are only computed once per scene.

One noteworthy consideration is the observation that triangle and point cloud resolution are somewhat independent of one another. Assuming that uniform surface sampling is used, only the area of a mesh determines the point cloud resolution, not the number of triangles that are used to describe it. The density of triangles can vary greatly across a given mesh, which makes it difficult to compare the execution times of point cloud and triangle based methods directly.

3. Methods

Among the 3D local descriptors considered in track, the GeDI (Section 3.1) and COPS (Section 3.2) methods are learning-based methods, while MICI (Section 3.3) is a more traditional histogram-based method. In the evaluation we include four descriptors used in the original ShapeBench [3]: Spin Image [15], Radial Intersection Count Image (RICI) [11], Quick Intersection Count Change Image (QUICCI) [12], and Signature of Histograms of OrienTations (SHOT) descriptor [9].

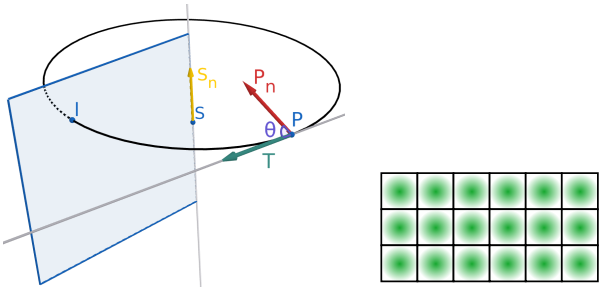
3.1. General and Distinctive Learned Descriptors (GeDI) (Ivan Sipiran)

GeDi [16] introduces a learned descriptor for local 3D point cloud patches that is compact and distinctive. A patch $X \subset \mathcal{R}^3$ is defined as a set of 3D points within a fixed radius r from a central point $\hat{x}$ in the original point cloud P. To accommodate varying point densities and ensure uniform input size for learning, the method performs a random sampling of m points per patch, with resampling if fewer points are present. This process yields a consistent structure for batch processing and model training. To achieve invariance to transformations and improve the robustness of the descriptor, the method estimates a local reference frame (LRF) using the TOLDI algorithm [17]. Finally, the method downsamples the path to $n < m$ points for computational efficiency.

The canonicalisation step transforms these sampled points to a normalized coordinate frame relative to the patch centre and radius. Specifically, points are first rotated into the LRF and then normalized for translation and scale invariance. The canonicalised point set serves as input to a deep network Φ_Θ, which learns to produce a descriptor $f \in \mathcal{R}^d$ with unit norm. The network design is based on PointNet++ [18], which uses hierarchical receptive fields to capture geometric patterns at multiple spatial scales.

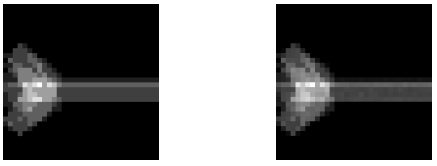
To keep geometric consistency and solve possible inaccuracies in LRF estimation, the method introduces QNet, a spatial transformer network that outputs a unit quaternion representing a rotation in $SO(3)$. Unlike matrix-based transformation networks, QNet inherently produces valid rotations without requiring additional regularization terms or computationally expensive orthogonalization steps. QNet is trained jointly with the main descriptor network, providing an efficient and integrated solution to compensate for canonicalization noise while preserving the spatial properties critical for geometric learning.

The training procedure uses a siamese network architecture with shared weights accross branches, processing pairs of corresponding patches sampled from overlapping regions of different point clouds. Descriptors are learned using a hard contrastive loss that emphasizes discrimination between matching and non-matching patches. Negative sampling is conducted by excluding samples within a predefined radius around anchor points, ensuring spatial distinctiveness. This training strategy, combined with randomized patch sampling, promotes robustness, supports large minibatch training, and leads to improved generalization across varying point cloud configurations.



(a) (b)

Fig. 7: In (a): visualization of the point projection and weighting procedure, and in (b) approximate visualisation of the weighting of point samples to the sum being accumulated in each pixel bin of the MICI descriptor.



(a) Triangle input (RICI) (b) Point cloud input (MICI)

Fig. 8: Descriptors generated using the RICI and MICI methodologies: despite differing input modalities, approximately equivalent descriptors are produced.

3.2. Comprehensive model for Parts Segmentation (Isaac Aguirre)

COPS [19] integrates semantics extracted from visual concepts and 3D geometry to effectively identify object parts. It renders a 3D point cloud from multiple viewpoints, delivering the resulting image outputs into DINOv2 [20] to extract high-level features. These features are then backprojected onto the corresponding points in the original point cloud. Finally, a geometry-aware feature aggregation process clusters points into parts and assigns them labels.

This procedure can also be used to compute features/descriptors for each point, and it is expected that geometrically similar parts will produce similar outputs. For the experiments, DINOv2 with registers [21] is used, which is an improved version of DINOv2, specifically in its small variant.

3.3. Multimodal Intersection Count Image (MICI) (Bart Iver van Blokland)

The Multimodal Intersection Count Image (MICI) is an approximation of the previously proposed RICI [11] descriptor. The RICI and MICI methods both aim to compute the number of intersections between a circle described by each pixel in the image, and the surface of an object. Where they differ is that while RICI requires a triangle mesh as input, MICI uses a point cloud (this is the “MICI PointCloud” variant). The combination of the RICI and MICI methods allows triangle meshes and point clouds to be compared across both modalities interchangeably (this is the “MICI Triangle” variant).

This can be advantageous in application domains such as bin picking, where it may be necessary to locate a known CAD object in a 3D scan. Because descriptors can be extracted from

the triangle mesh directly, the lossy step of uniformly sampling the mesh into a point cloud can be avoided. The extracted descriptors can subsequently be compared to those computed from points in a point cloud captured by a 3D scanner.

To estimate the intersection count per bin, MICI accumulates points from the input point cloud onto a plane subdivided into a grid of pixels. A visual representation of this procedure is shown in Figure 7(a). A descriptor is computed for the point S and surface normal S_n. A point P with surface normal P_n is projected in cylindrical coordinate space onto the descriptor, yielding point I that determines which pixel P contributes to.

Two factors weigh the contribution. The first of which is a 2D Gaussian function whose mean is centered in the corresponding pixel and has a standard deviation of 0.1. This aims to focus the contributions close to where the circles used by the original RICI descriptors would be. These Gaussian weights are visualised in Figure 7(b). The second weighting factor is the cosine of the angle θ between the circle tangent T and input point cloud normal vector P_n. For a given surface, as the angle between these vectors decreases, more points will be encountered in the proximity of the circle. Reducing the weight of these by the cosine accounts for this. The combination of these factors results, under ideal conditions, in a descriptor that is visually nearly indistinguishable from a similar one computed for a triangle mesh, as is shown in Figure 8.

All point contributions are accumulated in a 2D histogram. The final step in the feature extraction process is to convert the accumulated floating point values into a discrete number of intersections. This is done by dividing the contents of each bin by a constant factor c that depends on the density of the input point cloud, and thus the method and settings by which the point cloud is acquired. For this benchmark, we determined c experimentally as the factor that minimises the difference between all nonzero bins for the same descriptor computed using the MICI and RICI method for a large set of sample descriptors.

4. Results and discussions

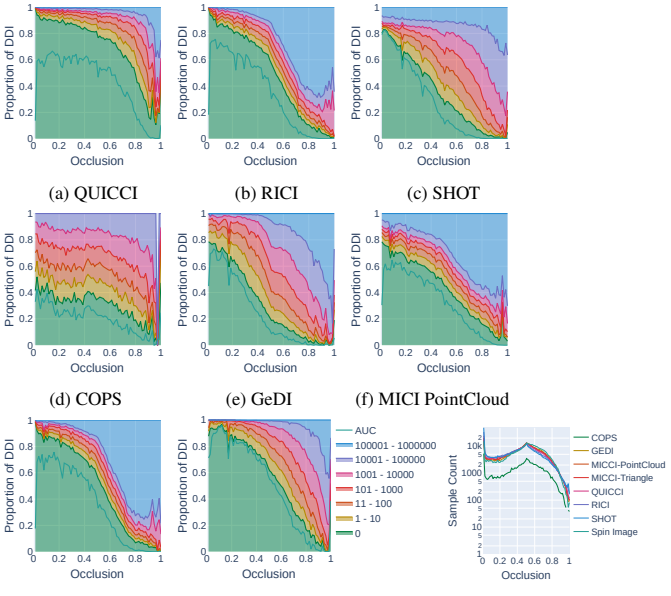
The results of the experiments defined in Table 1 are now presented for all participating methods. For these results, the following parameters have been used:

- Parameters for filters:
  - Fixed Gaussian noise: standard deviation of 0.001.
  - Vertex perturbation (alternate triangulation in the original): same as original ShapeBench [3].
  - Multi-view occlusion: angle between viewpoints varies between 0 and 90 degrees.
  - For Experiment 9, the number of clutter objects was reduced to 2 instead of the usual 10 for a clutter filter.
- Parameters for methods:
  - QUICCI: support radius 0.39.
  - RICI: support radius 0.255.
  - SHOT: support radius 0.15.
  - COPS: support radius 0.5 (for training).
  - GeDI: support radius 0.5 (for training).
  - Spin image: support radius 0.81.
  - MICI: Level threshold set to 166.6, support radius was 0.5 (MICI Triangle was also run at this support radius to be able to compare maximum achievable performance vs point cloud performance).

One other parameter of note is that vertex counts of point clouds provided to GeDI and COPS by the benchmark were scaled to 10% and 5%, respectively. Running both of these methods at full resolution proved intractably slow. The reference descriptor set of COPS was also limited to 250,000 descriptors for a similar reason. The latter does cause some problems with comparing its performance to other methods. However, based on experience the DDI=0 line should approximately be correct, but other subdivisions may shift had the full resolution been used instead. While this measure provides these methods with less information, we believe any practical application of them would require similar measures. Comparisons to other methods should therefore be possible.

4.1. Level 1 experiments

The first level investigates the effect of specific adverse conditions in isolation of others, where each experiment applies a single filter. For each experiment, the observed effect on the DDI metric is shown for each tested method. The colours show the distribution of DDI values, with green representing DDI = 0. A higher proportion of low DDI values corresponds to a more effective local descriptor. Therefore, the greener the chart, the more effective the local descriptor is. Each chart also contains the commonly used Area under Precision-Recall curves (AUC) metric.

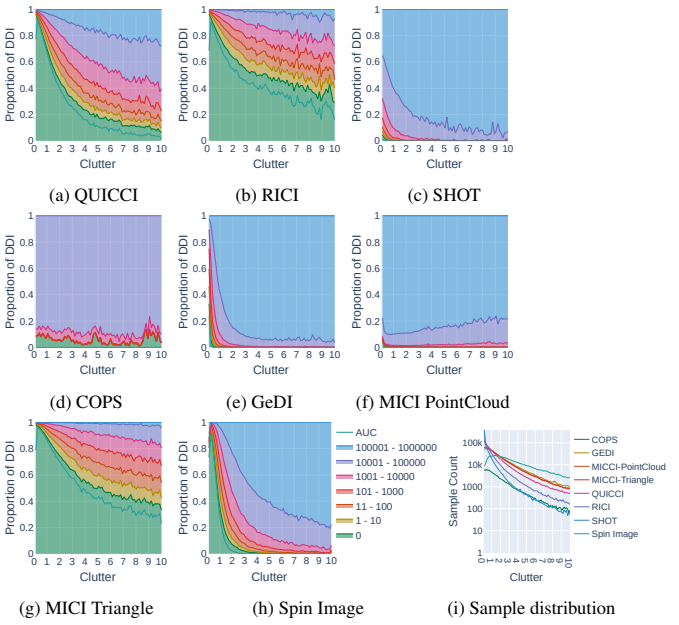


(a) QUICCI (b) RICI (c) SHOT

(d) COPS (e) GeDI (f) MICI PointCloud

(g) MICI Triangle (h) Spin Image (i) Sample distribution

Fig. 9: Results for Experiment 1 (Occlusion). Figure 9i shows the number of sample points per histogram bin.

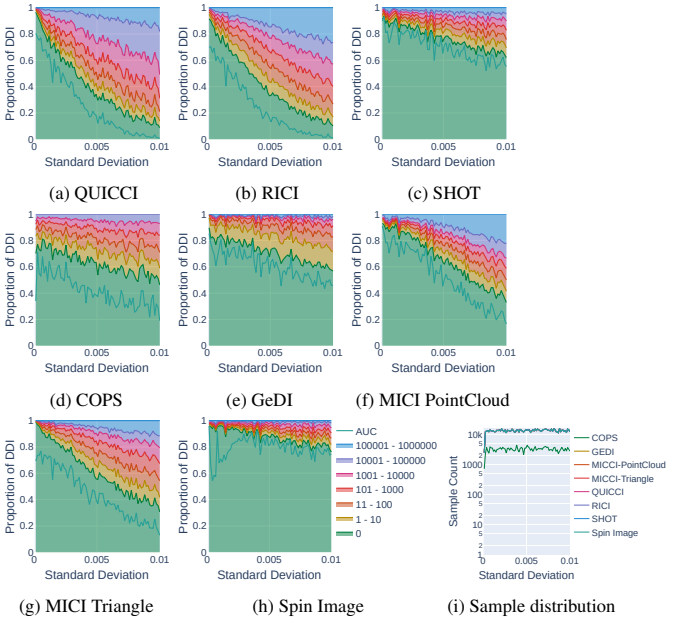


(a) QUICCI (b) RICI (c) SHOT

(d) COPS (e) GeDI (f) MICI PointCloud

(g) MICI Triangle (h) Spin Image (i) Sample distribution

Fig. 10: Results for Experiment 2 (Clutter). Figure 10i shows the number of sample points per histogram bin.



(a) QUICCI (b) RICI (c) SHOT

(d) COPS (e) GeDI (f) MICI PointCloud

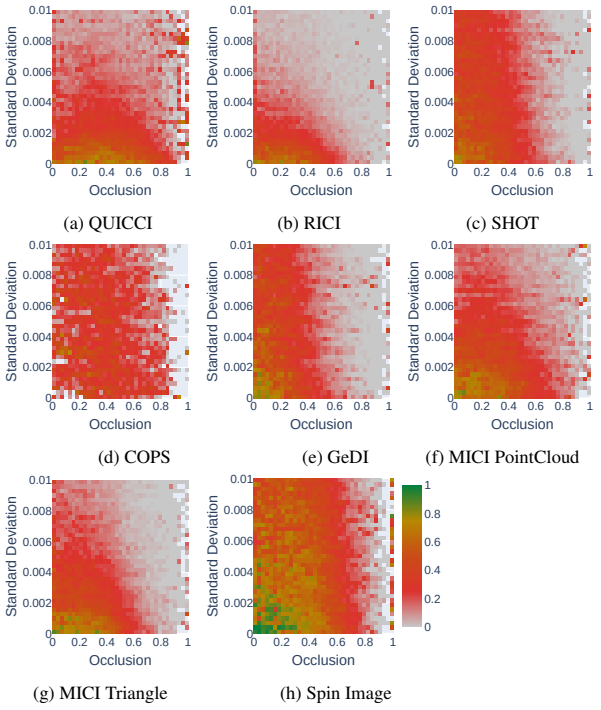
(g) MICI Triangle (h) Spin Image (i) Sample distribution

Fig. 11: Results for Experiment 3 (Gaussian Noise). Figure 11i shows the number of sample points per histogram bin.

The effects of occlusion are shown in Figure 9. The Spin Image, RICI, and QUICCI outperformed the other methods. GeDI and SHOT rely on point cloud neighbourhoods for their shape representation, which are degraded by the filter. Visual descriptors like COPS are also not robust to geometrical occlusion.

The results for clutter can be seen in Figure 10. Here MICI Triangle and RICI outperform the other methods, followed by QUICCI. These descriptors were specifically designed to be robust to clutter, which is evident here. The other methods show little to no ability to resist clutter.

When subjected to normally distributed vertex perturbations,

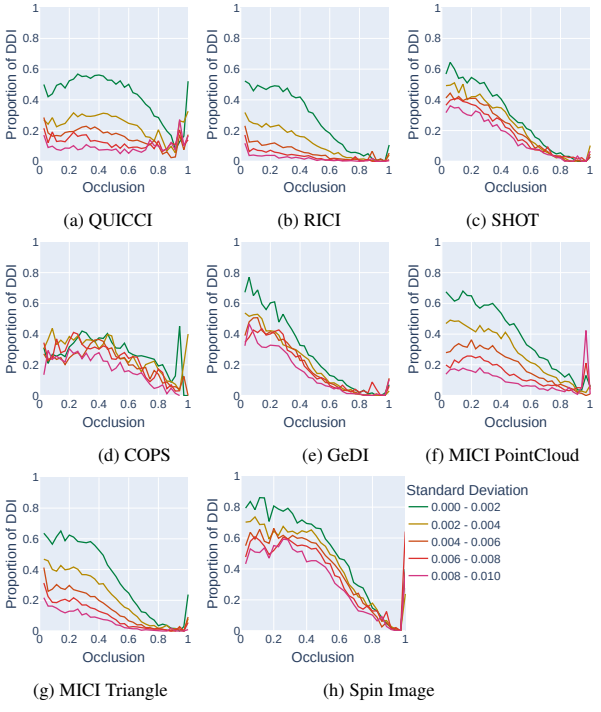


(a) QUICCI (b) RICI (c) SHOT

(d) COPS (e) GeDI (f) MICI PointCloud

(g) MICI Triangle (h) Spin Image

Fig. 12: Heatmaps of results for experiment 4 (Occlusion + Gaussian noise).



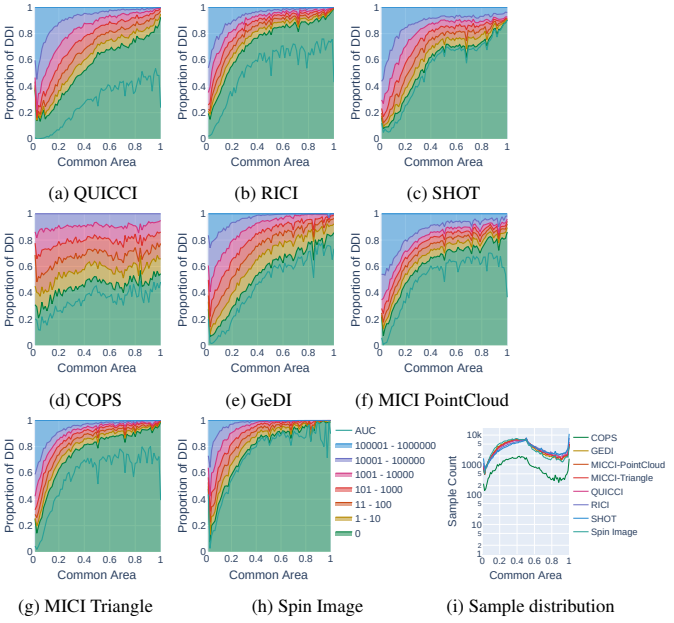
(a) QUICCI (b) RICI (c) SHOT

(d) COPS (e) GeDI (f) MICI PointCloud

(g) MICI Triangle (h) Spin Image

Fig. 13: Line curves of results for experiment 4 (Occlusion + Gaussian noise).

Figure 11 shows that the Spin Image is the most robust in this test. This can be explained by its subdivision of contributions of incoming vertices having a smoothing effect, and large support radius. In contrast, SHOT has impressive performance despite its small support radius. Learned neural networks work as smoothed regression functions, which could explain their robustness to noise. COPS may also benefit here from that visual

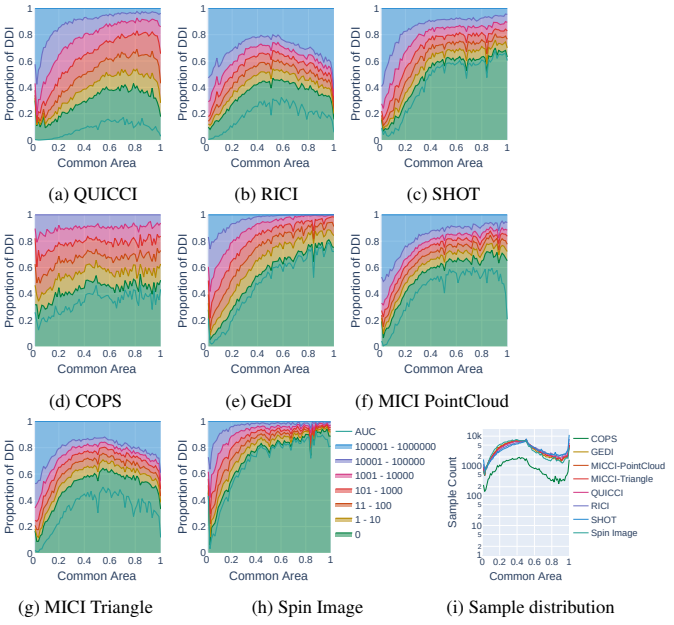


(a) QUICCI (b) RICI (c) SHOT

(d) COPS (e) GeDI (f) MICI PointCloud

(g) MICI Triangle (h) Spin Image (i) Sample distribution

Fig. 14: Results for Experiment 5 (occlusion on both meshes). Figure 14i shows the number of sample points per histogram bin.



(a) QUICCI (b) RICI (c) SHOT

(d) COPS (e) GeDI (f) MICI PointCloud

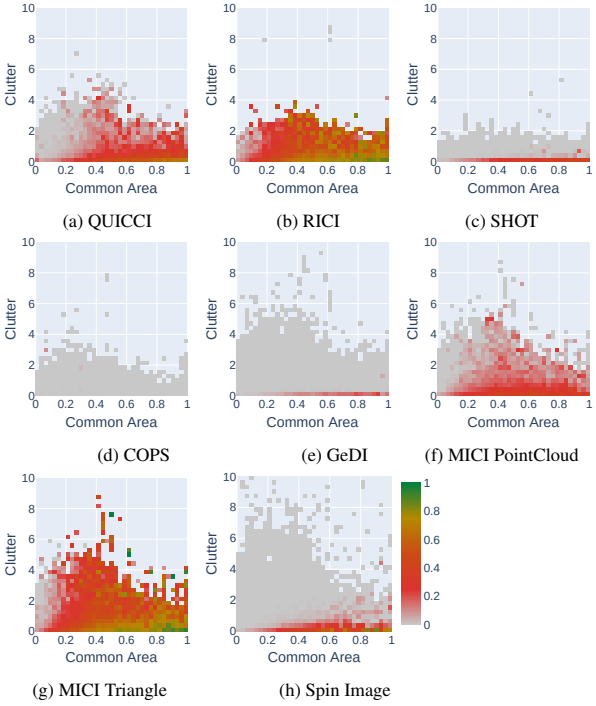
(g) MICI Triangle (h) Spin Image (i) Sample distribution

Fig. 15: Results for Experiment 6 (occlusion and Gaussian noise on both meshes). Figure 15i shows the number of sample points per histogram bin.

features are independent to geometrical noise. Finally, QUICCI and RICI are more susceptible to changes in the geometry, and thus they obtain worse results compared to the other methods.

4.2. Level 2 experiments

The second level experiments investigate the effect of combinations of filters. Two sets of plots (heatmaps and line curves) are computed for each experiment, showing the same results from different perspectives. To simplify visualisation, these charts focus on the fraction of cases where the DDI = 0. In the heatmaps, this proportion is represented by a colour map, where green indicates a proportion equal to 1 (the optimal result). The line charts group results by their values on the vertical axis.

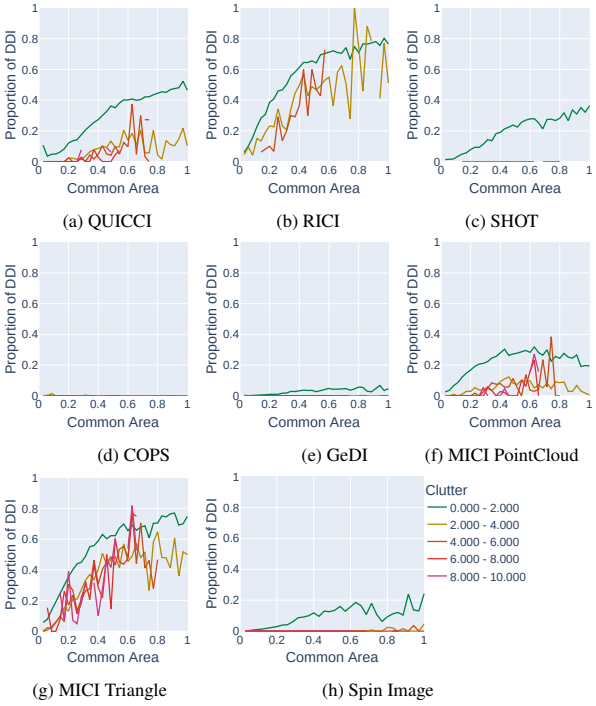


(a) QUICCI (b) RICI (c) SHOT

(d) COPS (e) GeDI (f) MICI PointCloud

(g) MICI Triangle (h) Spin Image

Fig. 16: Results for Experiment 7 (occlusion on both objects, clutter in scene).



(a) QUICCI (b) RICI (c) SHOT

(d) COPS (e) GeDI (f) MICI PointCloud

(g) MICI Triangle (h) Spin Image

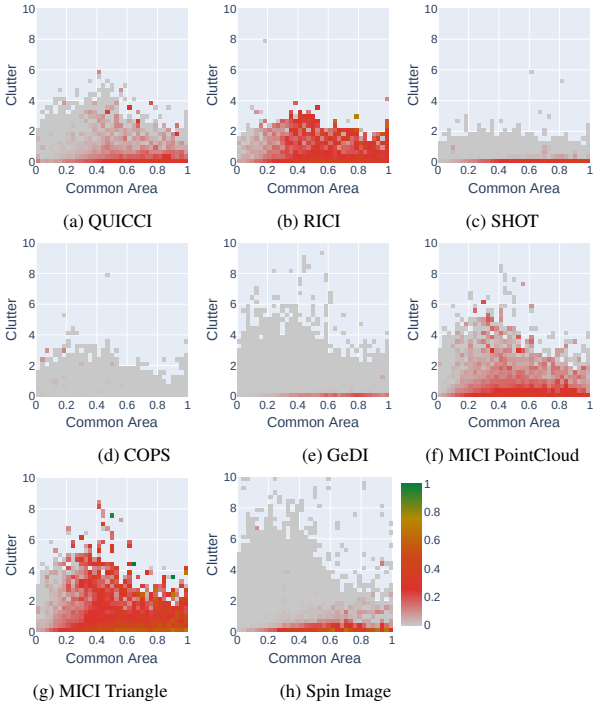
Fig. 17: Results for Experiment 7 (occlusion on both objects, clutter in scene).

When Occlusion is combined with Gaussian noise, Figures 12 and 13 show that the Spin Image obtains the best results, followed by MICI (both versions). As Spin Image behaves well in both filters separately, one would expect it to have good robustness to both filters. GeDI also shows some robustness for

low levels of noise and occlusion, which is also inherited from its robustness to Gaussian noise.

Experiments 5 and 6 both apply an occlusion filter on the model and scene objects, but ensure the maximum angle between the viewing directions is at most 90°. The occlusion factor for the surface visible from both points of view is used as the independent variable. Where the two experiments differ is that Gaussian noise with a fixed standard deviation is applied in experiment 6.

The results for experiment 5 are shown in Figure 14, and those for experiment 6 in Figure 15. In the case of the former, the conclusions are in line with those from experiment 1, although performance here is generally better. We conjecture that this is improvement caused by the filter generating occluded meshes from similar points of view, inadvertently making the model and scene objects more similar to one another than corresponding objects would be in experiment 1. When noise is added in experiment 6, the methods which were found to suffer most from its effects in experiment 3 are also those most affected by it here.



(a) QUICCI (b) RICI (c) SHOT

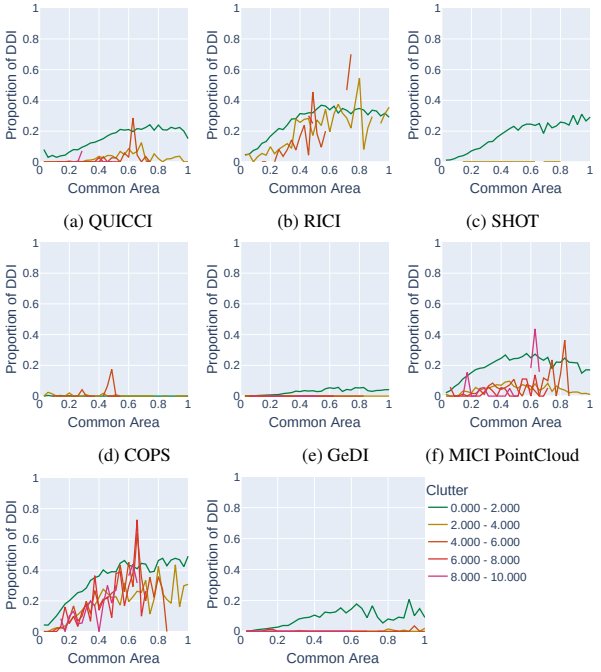
(d) COPS (e) GeDI (f) MICI PointCloud

(g) MICI Triangle (h) Spin Image

Fig. 18: Results for Experiment 8 (occlusion and Gaussian noise on both objects, clutter in scene).

The filters applied in experiments 7 and 8 are similar to those of 5 and 6, except for the addition of a clutter filter being to the scene object. The results for experiment 7 are shown in Figures 16 and 17, and those for experiment 8 in Figures 18 and 19.

As was shown in experiment 2, the COPS, SHOT, GeDI, MICI PointCloud, and to a lesser extent the Spin Image, are all affected by clutter. The methods that are the least robust to it are also those which suffer the most in both of these experiments. After adding noise in experiment 8, a similar drop is observed as to the one from experiment 5 to 6. Out of all tested



(a) QUICCI (b) RICI (c) SHOT

(d) COPS (e) GeDI (f) MICI PointCloud

(g) MICI Triangle (h) Spin Image

Fig. 19: Results for Experiment 8 (occlusion and Gaussian noise on both objects, clutter in scene).

methods, MICI Triangle and RICI are the only methods that can handle all applied filters.

4.3. Level 3 experiment

The third level aims to simulate a combination of artefacts commonly found in real-world 3D captures. Tested methods are subjected to a combination of occlusion, a fixed amount of Gaussian noise, vertex perturbations, and a small amount of clutter. The observed values for these results have been classified into a ‘high‘ and ‘low‘ category for easier interpretation. The results are shown in Figure 20.

No single 3D local descriptor could effectively handle high levels of all filters combined. MICI-Triangle and the Spin Image perform best with high levels of clutter, although the overall effectiveness remains low. GeDI performs well when not much clutter is present.

4.4. Execution Times

The results for the measured execution times are shown in Figures 21 and 22. It should be noted that these charts show results for triangle, point cloud, and learning based methods. Learning based methods utilise the GPU, while the remainder were run exclusively single-threaded on the CPU. Results for descriptor generation throughput for CPU and GPU-based methods can therefore not be compared directly.

All execution time results were measured on a system with an AMD Ryzen 9 3900X CPU and an Nvidia Quadro P5000 GPU. CPU frequency boosting was disabled to ensure the processor maintained a constant execution speed (3.8GHz). Execution was further limited to a single core through the operating system to avoid slowdowns from core switching.

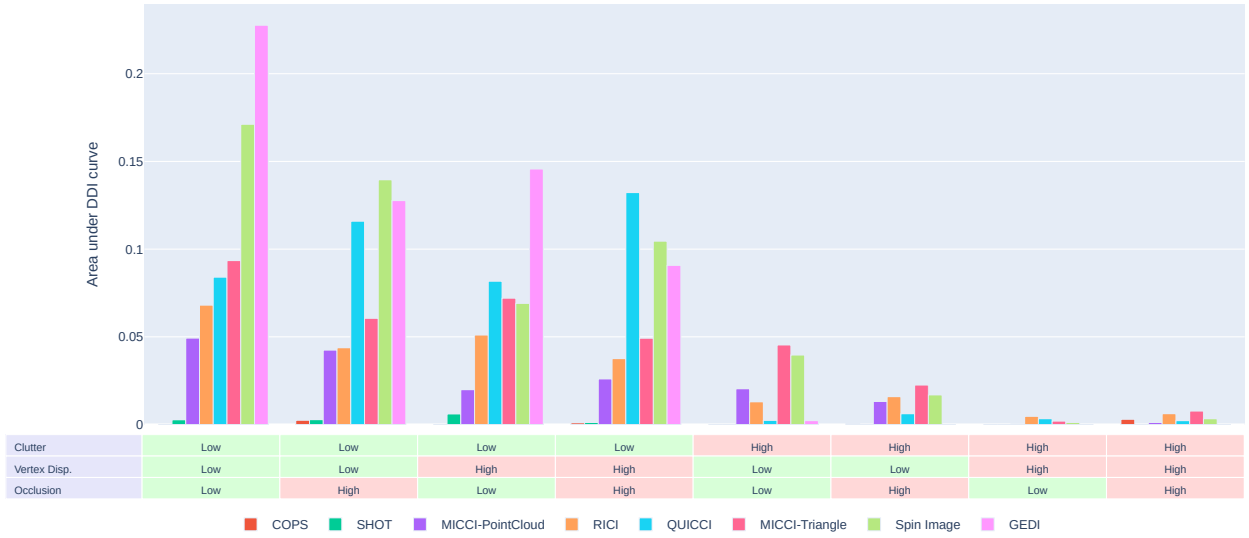
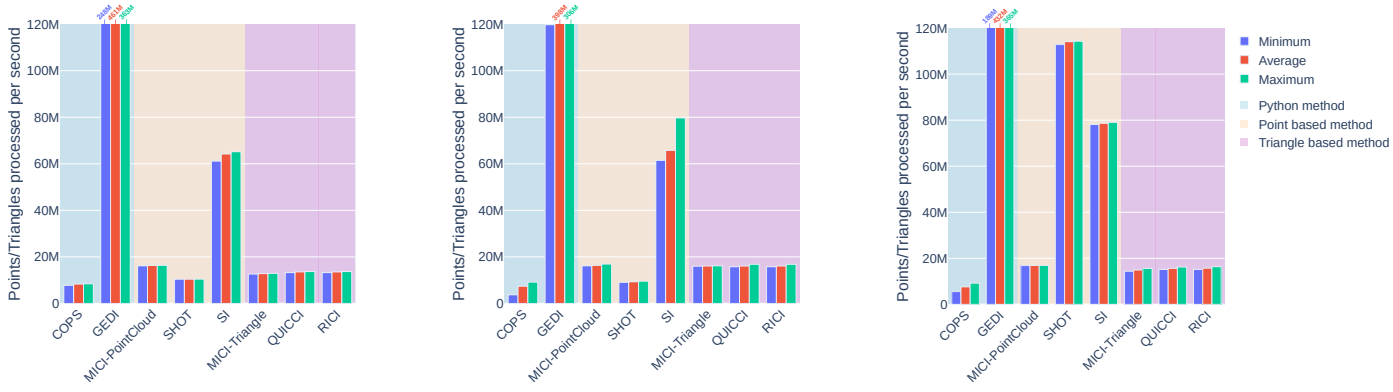


Fig. 20: Results for Experiment 9. DDI=0 curves are computed for all data points that fall above or below threshold values for each of the three independent variables in this experiment. Occlusion values above 0.3 are labelled as ‘high‘, and ‘low‘ otherwise. Clutter values between 0 and 1 are considered ‘low‘, and ‘high‘ between 1 and 2. Vertex displacements between 0.07 and 0.15 are considered ‘high‘.



(a) Uniform geometry density (b) Concentrated (high) geometry density (c) All geometry outside support volume

Fig. 21: Throughput of descriptor generation for different synthetic scenes. In Fig. 21a uniform density geometry within the support volume; In Fig. 21b high density geometry within the support volume; In Fig. 21c geometry outside the support volume.

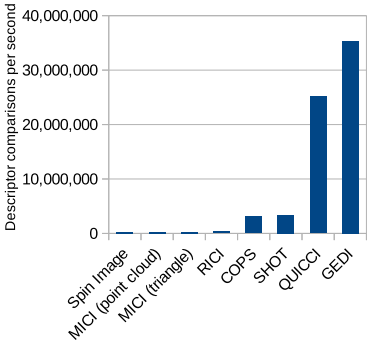


Fig. 22: Number of descriptors compared per second for each tested method.

In terms of the observed generation speed, GEDI is clearly the fastest of the learning-based methods. The Spin Image is the fastest of the CPU-based implementations. The only exception can be seen in Figure 21c, where the SHOT descriptor appears to be much faster at discarding points outside its support volume. Cylindrical support volume methods such as the Spin Image and MICI must perform more calculations to determine whether a point or triangle intersects their support region, and receive a comparatively small uplift. Only the synthetic meshes were used for measuring execution time.

In terms of comparison speed as shown in Figure 22, GEDI is again the fastest method. However, here QUICCI takes a comparatively close second place. These results have all used CPU implementations, and are thus comparable. The wide margin between GEDI, QUICCI, and the others can be explained by

their comparatively small size. The gap between them may be due to GEDI requiring fewer instructions to be compared, and its ability to better utilise vector instructions.

The estimated overhead was effectively zero for all methods. We have therefore not included these results in a separate chart.

4.5. Discussion

From the obtained results it is clear that clutter is particularly challenging for deep learning because there is a change of paradigm from a single object to a scene. Figures 13c and 15c highlight that SHOT is also not clutter resistant. As soon as the clutter filter is added, its performance decreases immediately. The same effect is observed in the clutter vs occlusion and clutter vs Gaussian noise heatmaps of the original paper.

The MICI methodology also displays exceptionally poor matching performance in cluttered scenes, while its corresponding triangle input version performs well. This can be attributed to the benchmark enforcing a maximum number of vertices per sampled point cloud, which results in a much lower point density per pixel when a number of clutter objects are added into the scene. Had this limit not been in place, we conjecture that performance penalty relative to the triangle input would have been similar to the one observed in the occlusion experiment.

Overall, it looks like methods tailored for local description, such as QUICCI, are less sensitive to dramatic perturbation. However, QUICCI and RICI "focus" on specific places where they expect to see changes in the intersection count. Adding noise causes those locations to become misaligned, and thus no longer match. This also explains their somewhat poor resistance to Gaussian noise, but it is these characteristics that also allow them to ignore any clutter. Interestingly, the higher support radius MICI Triangle variant performed better than RICI, despite only using a different support radius. This implies that the methodology for determining this radius used by ShapeBench may not be optimal, and that a higher radius means that the same vertex shifts caused by Gaussian noise are not as impactful compared to methods using a smaller radius instead.

Note that the current benchmark is evaluated independently of the grouping of the models into classes. Our approach to the analysis based on the DDI measure makes this benchmark considerably different from the SHREC 2013 one [1].

Another interesting observation is that our results were generated using a different root random seed from the original ShapeBench paper, but the resulting charts were highly similar (with some amount of noise depending on the filter used). This is evidence that the results are sufficiently quantitative.

5. Conclusions

This SHREC 2025 track on partial retrieval evaluated seven methods, including two methods rooted in deep learning. Generally, QUICCI, Spin Images and GeDI are robust to noise and occlusions. The benchmark is fully open and self-contained, permitting the modularity of testing other descriptors/methods. However, the size of the dataset (approximately 8.9 TB) requires adequate storage space and for some methods was necessary to decimate the data.

As a future development, we would consider reducing the dataset to 200-300 GB and analysing how well results are maintained when scaling down the number of objects used in the evaluation. In addition, from the experiments, it appears that a challenge for deep learning methods is the creation of ad hoc datasets that enable training a model over complex scenes. This suggests that, being completely deterministic, this benchmark could be adapted to generate the training scenes.

Acknowledgments

This work was funded by ANID - FONDECYT - Project 1230448 (Benjamin Bustos and Ivan Sipiran), and by ANID - Millennium Science Initiative Program - Code ICN17_002 (Benjamin Bustos). This work was partially supported by the Italian Ministry of Business and Made in Italy (MIMIT) project, House of Emerging Technologies Genoa (CTEGE): Digital factory for culture, 2023-2025 (Silvia Biasotti and Giorgio Palmieri). A total of 28 202 CPU hours were provided by the IDUN cluster [22] (Bart Iver van Blokland). This work was also supported by National Center for Artificial Intelligence CENIA FB210017, Basal ANID, Chile.

References

[1] Sipiran, I, Meruane, R, Bustos, B, Schreck, T, Johan, H, Li, B, et al. SHREC'13 track: Large-scale partial shape retrieval using simulated range images. In: Eurographics 2013 Workshop on 3D Object Retrieval (3DOR'13). Eurographics Association; 2013, p. 81–88.

[2] Marini, S, Paraboschi, L, Biasotti, S. Shape retrieval contest 2007: Partial matching track. SHREC (in conjunction with IEEE Shape Modeling International) 2007;:13–16.

[3] van Blokland, BI. Shapebench: A new approach to benchmarking local 3D shape descriptors. Computers & Graphics 2024;124:104052.

[4] Deitke, M, Schwenk, D, Salvador, J, Weihs, L, Michel, O, VanderBilt, E, et al. Objaverse: A Universe of Annotated 3D Objects . In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Los Alamitos, CA, USA: IEEE Computer Society; 2023, p. 13142–13153.

[5] Hao, L, Yang, X, Xu, K, Yi, W, Shen, Y, Wang, H. Rotational voxels statistics histogram for both real-valued and binary feature representations of 3d local shape. Journal of Visual Communication and Image Representation 2023;93:103817.

[6] Bibissi, DL, Yang, J, Quan, S, Zhang, Y. Dual spin-image: A bi-directional spin-image variant using multi-scale radii for 3d local shape description. Computers & Graphics 2022;103:180–191.

[7] Zhao, H, Tang, M, Ding, H. Hoppf: A novel local surface descriptor for 3d object recognition. Pattern Recognition 2020;103:107272.

[8] Yang, J, Zhang, Q, Xiao, Y, Cao, Z. Toldi: An effective and robust approach for 3d local shape description. Pattern Recognition 2017;65:175–187.

[9] Salti, S, Tombari, F, Di Stefano, L. Shot: Unique signatures of histograms for surface and texture description. Computer Vision and Image Understanding 2014;125:251–264.

[10] Guo, Y, Bennamoun, M, Sohel, F, Lu, M, Wan, J, Kwok, NM. A comprehensive performance evaluation of 3d local feature descriptors. International journal of computer vision 2016;116:66–89.

[11] van Blokland, BI, Theoharis, T. Radial intersection count image: A clutter resistant 3D shape descriptor. Computers & Graphics 2020;91:118–128.

[12] van Blokland, BI, Theoharis, T. An indexing scheme and descriptor for 3D object retrieval based on local shape querying. Computers & Graphics 2020;92:55–66.

[13] Logoglu, KB, Kalkan, S, Temizel, A. Cospair: colored histograms of spatial concentric surflet-pairs for 3d object recognition. Robotics and Autonomous Systems 2016;75:558–570.

[14] Guo, Y, Sohel, F, Bennamoun, M, Wan, J, Lu, M. A novel local surface feature for 3d object recognition under clutter and occlusion. Information Sciences 2015;293:196–213.
[15] Johnson, A, Hebert, M. Using spin images for efficient object recognition in cluttered 3D scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence 1999;21(5):433–449.
[16] Poiesi, F, Boscaini, D. Learning general and distinctive 3D local deep descriptors for point cloud registration. IEEE Transactions on Pattern Analysis and Machine Intelligence 2023;45(3):3979–3985.
[17] Yang, J, Zhang, Q, Xiao, Y, Cao, Z. Toldi: An effective and robust approach for 3D local shape description. Pattern Recognition 2017;65:175–187.
[18] Qi, CR, Yi, L, Su, H, Guibas, LJ. Pointnet++: deep hierarchical feature learning on point sets in a metric space. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17; Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964; 2017, p. 5105–5114.
[19] Garosi, M, Tedoldi, R, Boscaini, D, Mancini, M, Sebe, N, Poiesi, F. 3D part segmentation via geometric aggregation of 2D visual features. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2025;.
[20] Oquab, M, Darcet, T, Moutakanni, T, Vo, HV, Szafraniec, M, Khalidov, V, et al. DINOv2: Learning robust visual features without supervision. 2023.
[21] Darcet, T, Oquab, M, Mairal, J, Bojanowski, P. Vision transformers need registers. 2023.
[22] Själander, M, Jahre, M, Tufte, G, Reissmann, N. EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. 2024. URL: http://arxiv.org/abs/1912.05848; arXiv:1912.05848 [cs].