Doctoral theses at NTNU, 2021:398

Bart Iver van Blokland

A Search for Shape

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of
Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Bart Iver van Blokland

A Search for Shape

Thesis for the Degree of Philosophiae Doctor

Trondheim, December 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Computer Science

ISBN 978-82-326-6170-1 (printed ver.)
ISBN 978-82-326-5954-8 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2021:398

Printed by NTNU Grafisk senter

Abstract

As 3D object collections grow, searching based on shape becomes crucial. 3D capturing has seen a rise in popularity over the past decade and is currently being adopted in consumer mobile hardware such as smartphones and tablets, thus increasing the accessibility of this technology and by extension the volume of 3D scans. New applications based on large 3D object collections are expected to become commonplace and will require 3D object retrieval similar to image based search available in current search engines.

The work documented in this thesis consists of three primary contributions. The first one is the RICI and QUICCI local 3D shape descriptors, which use the novel idea of intersection counts for shape description. They are shown to be highly resistant to clutter and capable of effectively utilising the GPU for efficient generation and comparison of descriptors. Advantages of these descriptors over the previous state of the art include speed, size, descriptiveness and resistance to clutter, which is shown by a new proposed benchmark.

The second primary contribution consists of two indexing schemes, the Hamming tree and the Dissimilarity tree. They are capable of indexing and retrieving binary descriptors (such as the QUICCI descriptor) and respectively use the Hamming and proposed Weighted Hamming distance functions efficiently. The Dissimilarity tree in particular is capable of retrieving nearest neighbour descriptors even when their Hamming distance is large, an aspect where previous approaches tend to scale poorly.

The third major contribution is achieved by combining the proposed QUICCI descriptor and Dissimilarity tree into a complete pipeline for partial 3D object retrieval. The method takes a collection of complete objects, which are indexed using the dissimilarity tree and can subsequently efficiently retrieve objects that are similar to a partial query object.

Thus, it is shown that local descriptors based on shape intersection counts can be applied effectively on tasks such as clutter resistant matching and partial 3D shape retrieval. Highly efficient GPU implementations of the proposed, as well as several popular descriptors, have been made publicly available to the research community and may assist with further developments in the field.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) in partial fulfilment of the requirements for the degree of Philosophiae Doctor (PhD). The thesis is organised as a collection of papers, where Part I provides an overview over the work done as part of the thesis, and Part II includes the papers themselves.

The work was supervised by Theoharis Theoharis, and co-supervised by Anne C. Elster. Financing was provided by both the Faculty of Information Technology and Electrical Engineering, and the Department of Computer Science.

Acknowledgements

A great number of people have contributed to this Thesis by directly or indirectly supporting the research, being a welcome distraction from it, and ensuring it can progress at all.

My deepest gratitude goes to my supervisor Prof. Theoharis Theoharis for his endless advice, guidance, help, and support. In many ways this thesis would not have made it to the finish line without him.

I also deeply appreciate my co-supervisor Anne Elster and her listening ear, her knack for acquiring all kinds of computing powerhouses, the backyard meetings and barbecues, not to mention entrusting her course to me for one semester.

I'm extremely thankful to my parents Ron van Blokland and Wilma Heesterman, and sister Irene for their undying love, encouragement, and support, no matter the circumstances.

I'd also like to extend my sincerest gratitude to Michael Engel, Elmar Eisemann, Lasse Natvig, and Jan Christian Meyer. Thank you for being people to look up to.

Special thanks to Ingulf Helland, Özlem Özgöbek, Magnus Själander, Jost & Elisabeth, Joseph & Ylva, Björn & Miriam, Magnus S., Michael G., Magne, Pablo, David M., Håkon, Daniel, Charlie, David C., Aland, Peder, Bálint, Mateja, and Lars & Paul & Ben for all the hikes, dinners, paddles, trips, and evening play sessions.

I'd in addition like to thank Amund, Truls, Asbjørn, Lasse E., Odd Rune, David M., Nico, Rajiv, Lahiru, Zawadi, Pauline, Igor, Aleksander, Gunnar, Magnus J., Peter, Magnus H., Arthur, Jacob, Rakesh, Halvard, and Johannes for all the wacky lunch discussions and enduring my terrible puns.

Another big thanks goes to Berit Hellan, Rolf Harald Dahl, Birgit Sørgård, Anne Berit Dahl, Jan Grønsberg, Ellen Solberg, Erik Houmb, Camilla Thun Waaden, and Randi Holvik for their ability to quickly turn problems into solutions.

And last but not least, I would like to thank the Norwegian University of Science and Technology (NTNU) for their support for open access publishing, and the Computer Science department and the Faculty of Information Technology and Electrical Engineering for granting me the opportunity to undertake this journey.

You guys are the best.

Contents

Abstract i

Preface iii

Acknowledgements v

Contents vi

I Research Overview 1

1 Introduction 3

2 Background 7
2.1 Local 3D Shape Descriptors 7
2.2 GPU programming with CUDA 12
2.3 Binary Descriptor Indexing 14
2.4 Partial 3D Object Retrieval 15

3 Research Contributions 19
3.1 Contribution Overview 20
3.2 Paper Overview 22
3.3 Technical Contributions 24

4 Discussion 27
4.1 Evaluating Matching Performance of Local Shape Descriptors 27
4.2 Causes of Clutter Sensitivity 32
4.3 Novelty of the RICI and QUICCI descriptors 32
4.4 Indexing Binary Descriptors 33
4.5 GPU implementation of descriptors 34

5 Concluding Remarks 37
5.1 Conclusion 37
5.2 Future Work 37
5.3 Outlook 39

Bibliography 41

II Selected Publications 55

6 Paper A - Quasi Spin Images 57

6.1 Introduction . . . 58
6.2 Background . . . 59
6.3 Quasi Spin Images . . . 61
6.4 Results . . . 63
6.5 Implementation Details . . . 66
6.6 Conclusion . . . 66
6.7 Future Work . . . 67
6.8 References . . . 67

7 Paper B - Microshapes: Efficient Querying of 3D Object Collections based on Local Shape 71
7.1 Introduction . . . 72
7.2 Related Work . . . 73
7.3 Background: Quasi Spin Images . . . 74
7.4 Microshape Images . . . 75
7.5 Experiments . . . 77
7.6 Conclusion and Future Work . . . 80
7.7 References . . . 85

8 Paper C - Radial Intersection Count Image: a Clutter Resistant 3D Shape Descriptor 87
8.1 Introduction . . . 88
8.2 Background and Related Work . . . 89
8.3 Radial Intersection Count Images (RICI) . . . 92
8.4 Evaluation . . . 99
8.5 Observations and Discussion . . . 105
8.6 Conclusion . . . 107
8.7 Acknowledgements . . . 107
8.8 References . . . 107

9 Paper D - An Indexing Scheme and Descriptor for 3D Object Retrieval Based on Local Shape Querying 113
9.1 Introduction . . . 114
9.2 Background and Related Work . . . 115
9.3 Quick Intersection Count Change Image (QUICCI) . . . 117
9.4 Hamming Tree . . . 119
9.5 Weighted Hamming Distance . . . 121
9.6 Evaluation . . . 124
9.7 Issues with FPFH . . . 133
9.8 Conclusion . . . 133
9.9 Acknowledgements . . . 133
9.10 References . . . 133

10 Paper E - Partial 3D Object Retrieval using Local Binary QUICCI Descriptors and Dissimilarity Tree Indexing 137
10.1 Introduction . . . 138
10.2 Related Work . . . 139
10.3 Partial Retrieval Pipeline . . . 142

10.4 Dissimilarity Tree for Indexing Binary Descriptors 144
10.5 Adapting QUICCI Descriptors for Partial Retrieval 148
10.6 Evaluation 149
10.7 Discussion 156
10.8 Conclusion 157
10.9 Acknowledgements 157
10.10 References 157

Part I

Research Overview

Chapter 1

Introduction

The use of 3D capturing has steadily increased over the past decades and has become an important component of a number of applications such as:

- Industrial quality assurance (such as pavements [1], extruded metal parts [2], and piping [3]).
- Nondestructive analysis and preservation of historical artefacts (for example when surveying historic sites [4], [5] or digitising artefacts [6]).
- Video game and cinematic productions (which may be in the form of motion capture of hands [7] or faces [8], or assisting medical rehabilitation through play [9] [10]).
- Autonomous vehicles (for example to detect pedestrians [11] [12], or doing automated underwater inspections [13]).
- Biometrics (most notably for facial recognition [14] [15]).

Moreover, 3D capturing sensors have in recent years made their way into consumer hardware such as smartphones (including the Apple iPhone 12 and 13 Pro [16] [17], and Samsung Galaxy S20+ and Ultra [18]) and tablets (such as the Apple iPad Pro [19]), likely increasing the demand for efficient algorithms to process the recorded data.

Applications making use of 3D data rely on algorithms for tasks such as shape classification, registration, retrieval, and completion. All of these rely on shape similarity as part of their pipeline. The problem of establishing similarity between pairs of shapes is complex. Even for simple shapes, such as pairs of spheres and planes, there may be an infinite number of transformations that align them.

The most commonly used approach to solving the similarity problem is through the establishment of corresponding point pairs on each object's surface. Depending on whether surface normals are given, a minimum of two or three correctly matched pairs need to be found in order to determine an alignment transformation.

The dominant means by which point pair correspondences are found is through the use of local 3D shape descriptors. Their aim is to represent the volume around a given point (known as the *Support Volume*) in an array of constant size, thereby reducing the complexity of determining shape correspondences to an n-dimensional nearest neighbour search.

While shape descriptors have been applied successfully on a wide variety of tasks, methods proposed to date have some limitations. The most notable of which is the tendency of many methods to be computationally inefficient to construct, compare, or both. Moreover, despite the most popular descriptors being *embarrassingly parallel*, the limited number of reference implementations which have been made available by authors have predominantly been single threaded.

Meanwhile, the advent of the General Purpose Graphics Processing Unit (GPGPU, or GPU for short) has the potential to further accelerate descriptor computation, and to date there are few descriptors which have been implemented to run on the GPU, which to our

knowledge is limited to work by Davis et al. [20], Gerlach et al. [21], and Rusu et al. [22]. However, none of these have been designed specifically with GPU hardware in mind for optimal performance.

We thus observed the need for a descriptor with good matching capabilities, while simultaneously being designed for implementation on the GPU.

Primary Contribution 1: The GPU-based RICI and QUICCI descriptors.
The Radial Intersection Count Image (RICI) and Quick Intersection Count Change Image (QUICCI) descriptors are a pair of related descriptors which have been shown to be efficient to compute and compare on the GPU, and highly resistant to clutter.

High performance on the GPU is achieved by requiring less memory bandwidth during descriptor construction and comparison, relative to previous local descriptors such as the Spin Image (SI) [23], 3D Shape Context (3DSC) [24], and the Fast Point Feature Histogram (FPFH) [25]. The clutter resistance of the proposed descriptors is shown using a new benchmark called the *Clutterbox* experiment.

The QUICCI descriptor is a binary image. This led to the conjecture that it may be possible to index such descriptors, accelerating the retrieval of similar descriptors to a given query descriptor.

Methods addressing the problem of indexing binary descriptors have already been proposed (see Section 2.3), however, these proved unsuitable for querying QUICCI descriptors due to the tendency of nearest neighbours to have high Hamming distances. Existing methods have assumed such distances to be low. This led to the development of the second primary contribution:

Primary Contribution 2: Improved descriptor indexing.
The Hamming Tree and Dissimilarity Tree acceleration structures for the retrieval of similar binary descriptors.

The Hamming Tree is a scalable indexing structure for efficiently locating nearest neighbours in terms of Hamming distance. Unfortunately, the Hamming distance function proved to be inadequate for object retrieval using the proposed QUICCI descriptor because of the tendency of the descriptor to be sparse. We therefore also proposed the Weighted Hamming distance function, which was shown to be a more suitable distance function. As this function was incompatible with the Hamming tree, an alternate indexing structure called the Dissimilarity Tree was proposed, capable of retrieving descriptors using the Weighted Hamming distance function.

Using the QUICCI descriptor and the Dissimilarity tree, the final logical step was to extend them into a complete retrieval framework, which is the final primary contribution of the thesis.

Primary Contribution 3: A complete pipeline for partial 3D object retrieval.

The produced pipeline accepts any number of complete objects, indexes them, and subsequently allows content-based querying using partial objects. This combined pipeline is not the first to address the problem of 3D object retrieval, which to date it has seen extensive attention [26] [27]. Current methods can be categorised into retrieval of (partial) rigid objects

(e.g. [28] [29]), non-rigid objects (e.g. [30] [31]), and sketch-based retrieval (e.g. [32] [33]). Approaches to solve these can be classified as feature-based (e.g. [34] [35]), graph or structure-based ([36] [37]), and view-based methods (e.g. [38] [39] [40]). The pipeline presented as part of this thesis aims to retrieve partial rigid objects, using a feature-based approach. Partiality is particularly common for RGB-D images, where due to occlusion only parts of the surface of an object may be available at any given time.

One of the primary benefits of the proposed approach is that it is a complete solution consisting of compact QUICCI descriptors, which can be indexed in large quantities in a Dissimilarity Tree, retrieved efficiently, and is capable of locating accurate matches.

Chapter 2

Background

This chapter outlines the relevant context in which the contributions of this thesis were produced. Section 2.1 first introduces relevant local 3D shape descriptors, followed by Section 2.2 which provides a brief introduction into GPU programming using the CUDA programming model. Section 2.3 subsequently introduces the state of the art in binary descriptor indexing and retrieval, and finally Section 2.4 outlines previous work done on partial 3D shape retrieval.

2.1 Local 3D Shape Descriptors

Determining the similarity of two 3D shapes is fundamental to a number of applications, such as (partial) 3D object retrieval (see Section 2.4), 3D shape classification [38] [41], and 3D shape registration [42] [43]. The dominant means by which current work achieves this is through the use of shape descriptors [44]. Such descriptors can be classified into two main categories. *Global descriptors* aim to compute a single feature descriptor per object, while *Local descriptors* are computed in large numbers for a single object, each describing a portion of its surface. While global shape descriptors can be applied in situations where no clutter or occlusions are present [45], this is not the case for most practical applications. The use of local descriptors has therefore received most attention in the literature. Some examples of global descriptors are PANORAMA [38], Ensemble of Shape Functions [46], Rotational Contour Signatures [47], and the Spatial Structure Circular Descriptor [48].

Local 3D shape descriptors are commonly computed over a surface point, either as part of a triangle mesh or point cloud. The produced descriptor in turn describes the surfaces found in a volume, usually spherical, centred around this point, called its *support volume*.

As a given object commonly has many individual surface points, local shape descriptors are usually generated in large volumes. This can to a certain extent be counteracted by using a keypoint detector [45] [49] [50] at the cost of matching performance becoming constrained by the detector's effectiveness.

A local shape descriptor ideally ought to possess the following properties [51]:

Discriminating Corresponding surface point pairs produce similar descriptors, and vice versa.

Robust Correspondences can still be detected despite noise or other surface disturbances.

Invariant to rotation The orientation of either shape does not affect matching performance.

Invariant to mesh resolution Variations in mesh resolution (mean length of edges in the mesh) does not affect matching performance of the descriptor.

Resistant to clutter The descriptor's matching performance is unaffected by

	other shapes present within the vicinity of the point being described.
Resistant to occlusion	The descriptor's matching performance is unaffected by portions of surfaces being missing near either of two corresponding surface points, for instance due to the inability of a capturing device to register them.
Computationally inexpensive	The descriptor is fast to compute and compare.
Compact	The descriptor's storage requirements are low.

Depending on the circumstances, invariance to scale may also be a desirable trait. However, it may also be seen as a an additional discriminating factor. As its desirability thus depends on the application domain, we do not consider it a universal benefit.

Aside from the discriminative property, resistance to clutter is of particular importance. It has been named as one of the major factors for matching performance deterioration in existing local shape descriptors [51], and a number of methods have been shown to be susceptible to it. The issue stems from the support volume which most local shape descriptors rely upon. Human environments, particularly indoor ones, often contain many different objects within each other's vicinity. When attempting to do object recognition, descriptors computed over surface points on one object can therefore be expected to regularly contain geometry belonging to other clutter objects within their support volumes, affecting the computed descriptor. It has generally been measured using the definition proposed by Johnson et al. [23] as the fraction of surface points in a given (support) volume that are not part of the object of interest, relative to the total number of surface points in that volume.

From this definition, it follows that clutter resistance can also be a factor in partial object retrieval. When a portion of an object is compared to its corresponding larger counterpart, all surfaces in the larger object which are not part of the smaller portion are effectively clutter from the perspective of the retrieval process. In a similar fashion, a pair of objects may contain an intersection of similar geometry, while in both cases the remainder is different, known as partial matching [52]. An example of this is shown in Figure 2.1. In the case of these chairs, the arm and backrests are in a sense clutter when attempting to locate similarities between them, which may occur when retrieving either of the chairs, using the other as a query.

Many descriptors have been proposed over time, and covering all of them in this section is intractable. We will therefore for the remainder of this section focus on the descriptors which were deemed relevant to this thesis.

2.1.1 Spin Image

The Spin Image, proposed by Johnson et al. [23], is a popular descriptor representing variations in point cloud density in the vicinity of an oriented point.

The descriptor, shown in Figure 2.2, is conceptually computed by rotating a plane around a line described by the oriented point described by the descriptor. The plane is subdivided into histogram bins, each counting the number of points intersecting with the plane as it rotates. To reduce the effects of aliasing, bilinear interpolation is used to divide each projected point's contribution over the four neighbouring bins. The Figure shows the effective support volume covered by a single histogram bin, which is a square torus-like

Figure 2.1: Two chairs which have identical legs, but are otherwise different in their designs.

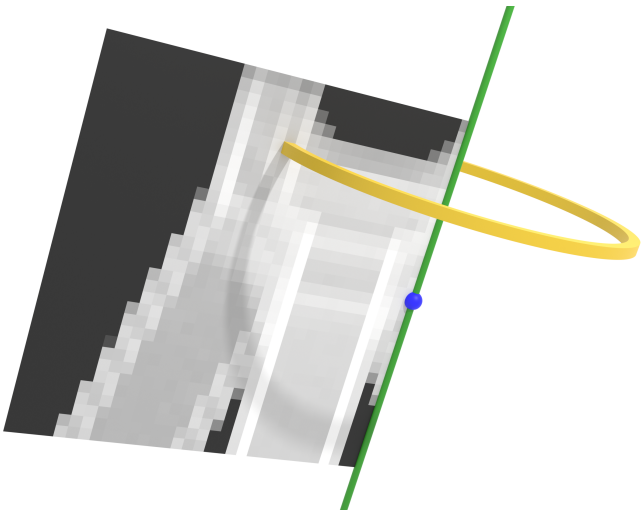


Figure 2.2: Construction of a Spin Image descriptor, computed for a given point (blue), with its vertical axis (green) aligned with the surface normal. The square plane shows a sample spin image descriptor, and the torus-like volume described by a single pixel is indicated (yellow). The volume is produced by rotating the square pixel for one revolution around the vertical (green) axis.

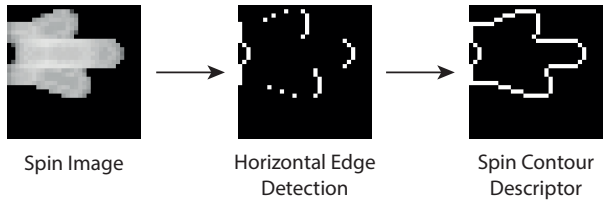


Figure 2.3: Visualisation of the construction of a spin contour descriptor based on a previously computed spin image.

volume. As the descriptor is typically generated as a square image, the support volume of the descriptor is thus a cylinder whose diameter is twice as high as its height. Spin Images are compared by computing their Pearson correlation coefficient.

Johnson et al. also proposed a filter for points whose surface normal deviates from the descriptor's oriented point by more than a given threshold. If the threshold is exceeded, the point's contribution is nullified. The authors claim this filter reduces the influence of self-occlusion and clutter within the support volume. This angle threshold is known as the *support angle*.

Since its publication, a number of variations and improvements to the descriptor have been proposed [53] [54] [55] [56] [57] [58]. Noteworthy contributions include those by Carmichael [59] et al., who observed that the Spin Image effectively estimates the surface area of an object intersecting the volume described by a single histogram bin (also shown in Figure 2.2). They propose a noise free variant of the Spin Image computed from triangle meshes. Additionally, Davis et al. [20] and Gerlach et al. [21] implemented and tested a GPU accelerated version of the Spin Image.

2.1.2 Spin Contour

Another derivative descriptor of the Spin Image is the Spin Contour proposed by Liang et al. [60] [61]. Due to its similarity to some of the work contained in this Thesis, it deserves special attention here. The descriptor is conceptually computed from a Spin Image descriptor at a given oriented point. However, for consistency purposes the authors recommend using the original triangle mesh instead.

The means by which a spin contour descriptor is constructed when using a spin image as a starting point is shown in Figure 2.3. First the horizontal boundaries are computed between zero and nonzero pixels. Additional pixels are subsequently added to this initial set of boundary pixels to form a closed contour. Comparing Spin Contours is the average distance between contour pixels in one image relative to their nearest contour pixel in the other.

2.1.3 3D Shape Context

The 3D Shape Context (3DSC) proposed by Frome et al. [24] is a descriptor which aims to represent variations in the surface density of an object in a spherical volume around an oriented point.

As shown in Figure 2.4, the support volume is divided into a number of smaller subvolumes. Each such subvolume corresponds to a bin in the descriptor. Computing

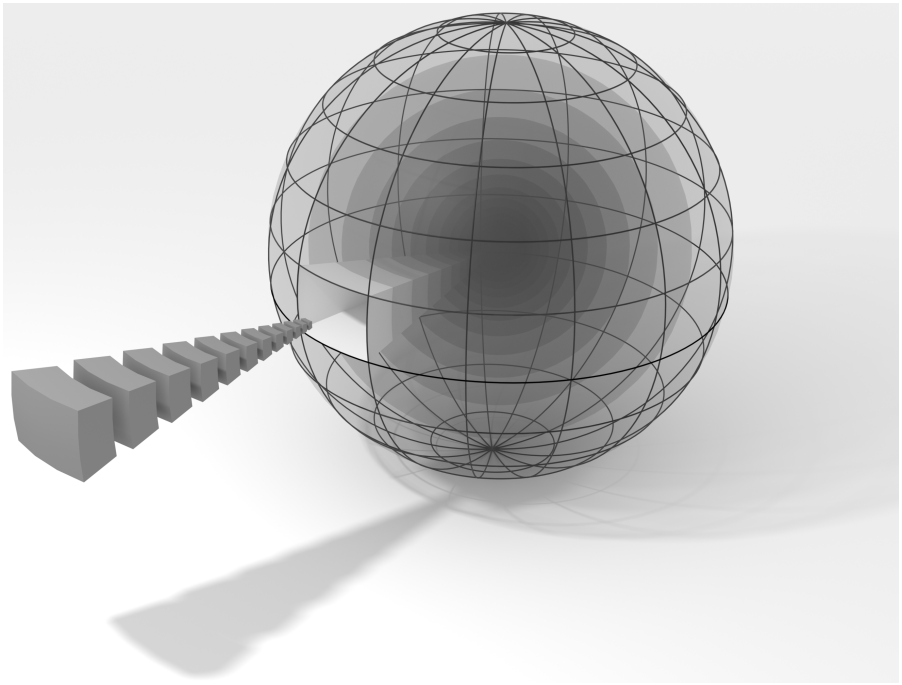


Figure 2.4: Visualisation of how the 3DSC descriptor subdivides its support region. Inside the sphere, layers of smaller spheres are visible. The outer sphere contains longitudinal and latitudinal subdivisions. One such subdivision has been extracted from the support region, showing the shape of each individual descriptor bin.

the descriptor is done by computing the angle and distance from each sample point within the support radius to the surface point for which the descriptor is computed. The bin corresponding to that relative angle and distance is subsequently incremented, albeit scaled to the volume of the histogram bin and local point density.

One of the descriptor's problems is that it is not invariant to rotation. The authors solve this by duplicating a descriptor with n azimuth subdivisions n times, rotating the descriptor's bins one azimuth step at a time. However, a more practical approach is to store the descriptor once, and perform the required rotations at comparison time. Tombari et al. [62] proposed the Unique Shape Context which aims to alleviate this problem, which uses a consistent means to compute a local reference frame to ensure rotation invariance.

2.1.4 Fast Point Feature Histogram

The Fast Point Feature Histogram (FPFH) proposed by Rusu et al. [25] is an extension to the previously proposed Point Feature Histogram [63] [64].

Constructing an FPFH descriptor for a given surface point p is done by first computing a Simplified Point Feature (SPF) for each point in the input point cloud. An SPF of a given point p_i is computed by computing three angular properties for each point within a given radius around p_i. The range of each of those angular properties are divided into 11 equally sized bins, and the number of points falling in each of those bins is counted, the

concatenation of which produces an SPF histogram with 33 bins. The final FPFH descriptor adds the SPF of p to the average SPF of points in the neighbourhood of p, weighted by their distance to p.

2.1.5 Other Descriptors

The descriptors discussed up to this point are those most relevant to the work done as part of this thesis. However, numerous other methods have been proposed to date. Some notable examples of such descriptors are now briefly discussed.

Flint et al. and Darom et al. attempted to exploit the success of the SIFT descriptor [65] for 3D data, for keypoint selection and descriptor construction, resulting in the ThrIFT [66] and LD-SIFT descriptors [54], respectively.

The SHOT descriptor proposed by Salti et al. [67] aims to combine the spatial subdivision of the 3DSC and USC descriptors with a geometric properties histogram such as 3DSC. Rather than a weighted sum of points, each spatial subdivision of the support region tallies points by their normal vector. The descriptor is the concatenation of each histogram for all subdivision volumes. Prakhya et al. have also proposed a binary version of the SHOT descriptor [68].

Guo et al. proposed the Rotational Projection Statistics descriptor [44], constructed by first rotating an object to three specific orientations within a computed reference frame. Next, for each rotation, points from the object surface are projected on to the xy, xz, and yz planes, and counted in a 2D histogram. The descriptor is finally computed by computing several statistical properties over the histograms and concatenating them.

Sun et al. [69] use a shape's heat dissipation over time as a means to identify similar points or regions in an object. The authors also show the method can be used for keypoint detection.

Learning methods such as those proposed by Charles et al. [70] and Zhang et al. [71] have also gained traction in recent years. However, due to neural network architecture constraints, regularisation strategies such as low-resolution voxel grids are commonly required. Combined with their tendency to be computationally expensive [72] means their practical applicability is currently limited.

2.2 GPU programming with CUDA

GPUs were initially introduced as accelerator cards for offloading specific compute and memory intensive graphics related operations from the CPU. These operations were fixed in function and implemented in hardware. The complexity of these operations increased with time, along with demands from the video games industry for finer control over the rendering pipeline. This gave rise to the General Purpose GPU (GPGPU) which is commonplace today.

While the general GPU architecture is largely similar across the primary vendors and architecture generations (NVidia, AMD, ARM, and Intel), implementations that were part of this thesis have used the CUDA programming language, which is limited to NVidia processors. As such this description will focus on that.

A GPU core consists of memory and communication controllers, banks of L2 cache, and a varying number of processing cores referred to as *streaming multiprocessors* (SMs) [73]. A visualisation of the most important GPU core components is shown in Figure 2.5.

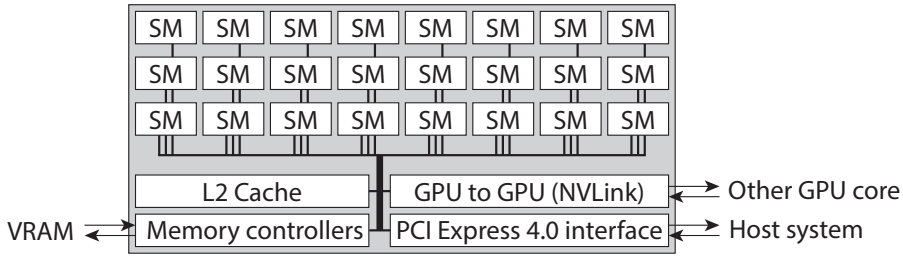


Figure 2.5: GPU Architecture.

The GPU design is intended to execute large quantities of threads, each executing the same procedure, referred to as a *kernel*. During execution, threads are allocated in groups referred to as *blocks* of equal size to SMs. They remain allocated to the SM until every single thread in the block has finished execution, upon which a new block is assigned to take its place. A kernel launch on the GPU causes a *grid* of blocks to be queued, and progressively divided over the available SMs for execution. Each SM has resources to execute one or more blocks simultaneously, depending on the complexity and memory requirements of the kernel being executed.

The architecture of the SM of the current *Ampere* generation is shown in Figure 2.6. It contains four similar execution units, with shared execution pipelines for texture and ray-triangle intersection acceleration instructions, and a shared bank of L1 cache, which can in part be used as a small amount of temporary storage for blocks referred to as *shared memory*.

When a new block is assigned to the SM, the threads within that block are allocated a set of registers in one of the four register files. The number of registers a thread requires is determined at compile time and cannot change during execution. More complex kernels will generally require more registers per thread, which also reduces the total number of threads which can execute on the SM simultaneously. For performance it is therefore advantageous to limit thread complexity whenever possible. Blocks are assigned to SM's until either the register files or shared memory banks are full. The minimum of these therefore determines the number of blocks able to execute simultaneously.

Instructions of threads are executed within the functional units of the SM in batches of 32 threads, called *warps*. At each clock cycle, each of the warp schedulers analyses the program counters of the threads assigned to their register file. From these, an instruction is selected to execute, and assigned to an available execution pipeline which can execute that instruction. Due to memory, synchronisation, or execution unit stalls, there may not always be instructions which can be executed. The fraction of cycles in which a scheduler can execute an instruction is referred to as *occupancy*.

Under ideal conditions, the SM has a sufficient number of threads executing to allow an instruction to execute every single cycle. In this case, *latency hiding* can occur, where the high latencies induced by the memory system are effectively hidden as the occupancy remains high. This can only be achieved if threads make effective use of memory they request from main memory and do not invoke excess memory transactions. Regions of memory which are read from and written to frequently should generally be kept in shared memory for the duration of the block's lifespan.

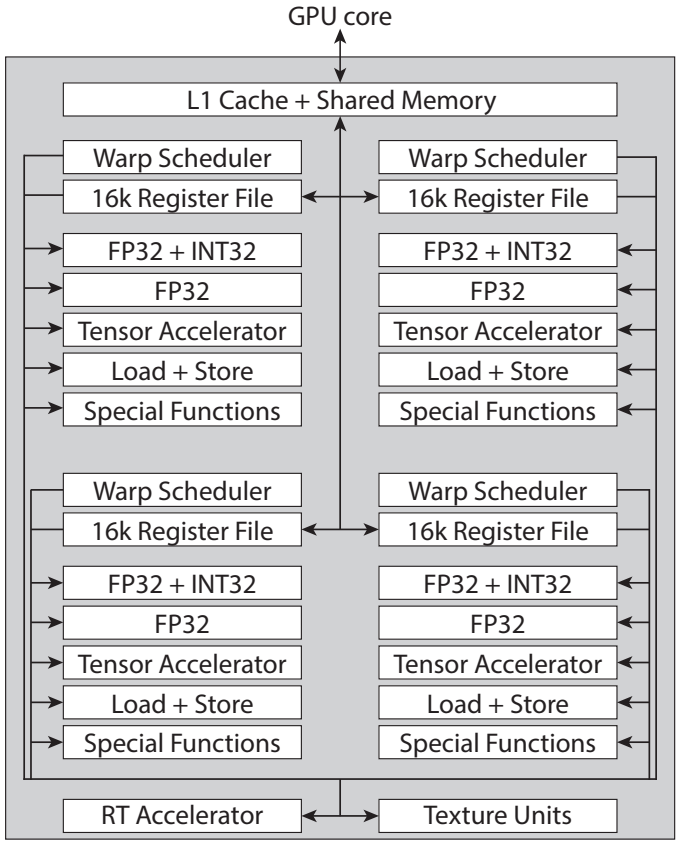


Figure 2.6: SM Architecture.

2.3 Binary Descriptor Indexing

For the purposes of indexing binary descriptors, modern relational databases are primarily able to efficiently locate exact matches. However, ranking descriptors by Hamming distance is not natively supported by any of those[1] commonly in use today.

There exist a number of situations in which locating Nearest Hamming neighbours to a given binary descriptor is desirable. The most noteworthy of these are binary descriptors. A number of such descriptors aimed at retrieving images and 3D shapes have been proposed. Examples include the B-SHOT [68], BRIEF [74], ORB [75], and BRISK [76] descriptors.

Another common application is dimensionality reduction through hashing, which aims to reduce real-valued n-dimensional points to more easily searchable binary values for the purpose of nearest neighbour search. Hashing strategies proposed to date can be categorised into Locality Sensitive Hashing, and data-dependent or learned hashing [77]. Applications of hashing-based search includes image [78] [79] and video retrieval [80] [81].

Locality Sensitive Hashing (LSH), proposed by Har-Peled et al. [82] uses randomised multidimensional planes to project points into a binary bit string. Applications of the method

[1] Neither PostgreSQL, MySQL, or Oracle DB allow sorting by Hamming distance

includes work by Broder et al. [83] [84] and Sadowski et al. [85]. Unfortunately, due to the random nature of LSH, produced hashes may be inaccurate or inefficient [86]. As a result, a large number of bins may be created, or entries are unevenly distributed across hash bins. A number of methods have been proposed, for instance aiming to accelerate retrieval [87] [88], allow other distance functions to be used [89], or reduce storage requirements [90].

Most current research in dimensionality reduction has shifted from unsupervised (such as LSH) to supervised hashing, due to the overall more favourable properties of the produced hashes. Due to its utility in retrieval and machine learning, the field has seen significant attention [91] [92] [93] [94]. More recent work has also employed deep learning for this purpose [79] [95] [96] [97].

For optimal precision, LSH benefits from long hash codes [86] (although this comes at the cost of recall). Additionally, binary descriptors tend to be too volatile to index in a binary tree, thereby creating the need for an indexing structure capable of locating nearest neighbours in Hamming space efficiently in potentially long binary descriptors. A number of methods have been proposed which aim to address it.

Early proposed methods were shown to be efficient, but were limited to retrieve binary descriptors up to a Hamming distance of 2, short bit strings, or both [98] [99] [100] [101].

More recent methods have all followed a similar paradigm in their overall approach. Binary descriptors are subdivided into a number of equally sized substrings. Each substring is subsequently indexed in a corresponding hash table. The initial approaches by Liu et al. [102] and Norouzi et al. [103] requires testing containment of each hash table for all permutations of a query descriptor up to a given Hamming distance. Unfortunately, testing all permutations of a given query rapidly becomes intractable. Subsequent methods have aimed to accelerate the discovery of neighbouring descriptors within hash tables, by for instance using inverted lists [104], using a prefix tree (trie) [105], or a k-means inspired tree [106]. Other variants improving scalability [107] as well as a more generalised version [108] have also been proposed.

While these methods have addressed the deficiencies of the aforementioned early solutions, they generally expect neighbours to have low Hamming distances. Their effectiveness decreases substantially as this distance increases. In these situations, outperforming a linear search becomes difficult.

2.4 Partial 3D Object Retrieval

The problem of partial object retrieval aims to match portions of surfaces to the complete object they belong to. This may for example occur in RGB-D images, where parts of objects are occluded or facing away from the scanner. An example of this is shown in Figure 2.7. Alternatively, only a portion of one object may be similar to another while the remainder is not, such as a set of cutlery with matching handles for the spoons, forks, and knives.

Partial retrieval methods proposed to date have often been divided into three categories; those using Bag of Visual Words (BoVW), View-based, and Part-based methods [109] [52].

BoVW methods use a collection of local descriptors to represent an object surface. When two arbitrary objects contain a subset of surfaces which are similar, the local descriptors computed over those surfaces should be similar too. Thus, partial matches can be found by determining the existence of such a subset. As the volume of descriptors that are generated may be large, some methods also aggregate descriptors using algorithms such as k-means as part of the matching process.

Figure 2.7: Example of a scene containing partial objects, caused by parts of their surface being invisible to the 3D scanner (scanned using a Kinect sensor by Tombari et al. [45]).

Ohbuchi et al. [110] computed SIFT [65] features from range images captured from a number of viewpoints. These features are classified using k-means, and aggregated into a histogram for retrieval. Liu et al. [111] uses a similar approach, clustering Spin Image descriptors instead. Lavoué et al. [29] project the geometry of surface patches centred around keypoints onto the eigenvectors of the Laplace-Beltrami operator. Bronstein et al. [112] proposed the *Shape Google* algorithm aimed at the retrieval of non-rigid shapes, which constructs a vocabulary from descriptors computed using heat kernels, and are indexed using binary hashing. Mohamed et al. [113] extends this by constructing a descriptor based on both the heat diffusion kernel as well as the local Fourier spectrum of the Laplace-Beltrami operator. Savelonas et al. [114] use an extension to the FPFH [25] descriptor and Fisher vectors for both local and global matching. Finally, Dimou et al. [115] extracted features from segmented depth images.

Another means to simplify the complexity in locating partial matches is by segmenting objects into smaller parts. Whereas the object as a whole may be missing portions of its surface, individual parts are less likely to be affected. This design philosophy is partially similar to that of BoVW methods, however, part-based methods effectively cluster vertices by their semantic or physical proximity. BoVW methods tend to cluster the produced descriptors instead. Methods using part-based retrieval includes work by Agathos et al. [116], who represent which segmented parts are connected together as a graph. This graph is subsequently used to locate objects with a similar structure. Tierny et al. [117] also encode relationships between parts, using Reeb graphs instead. Furuya et al. [118] subdivide the bounding box of objects into randomised cuboid subvolumes, using surfaces contained within as object parts. A binary hash is computed from each part, allowing efficient retrieval. Furuya et al. [28] subsequently proposed a learning based method which uses a combination of SPRH and PFH descriptors [119] computed over segmented parts as input. Several neural networks subsequently transform these features in a multidimensional feature space in which

nearest neighbour surface points can be found.

Finally, view-based methods aim to exploit work done in image recognition on three-dimensional surfaces by rendering objects from different viewpoints, and using the produced images as input to the matching algorithm. The most popular approach is using the SIFT descriptor in a variety of ways [120] [121] [122] [123]. Sfikas et al. [124] extracted Dense SIFT [125] keypoints from the previously proposed PANORAMA [38] descriptor, which are clustered into a codebook using k-means. Tashiro et al. [126] more recently proposed a pipeline centred around the SURF [127] local feature descriptor. Aono et al. [123] presented three variations of the same method, each encoding KAZE features [128] extracted from object views. These encoding algorithms were the Vector of Locally Aggregated Descriptors (VLAD) [129], Gaussian of Local Distribution (GOLD) [130], and Fisher Vectors (FV) [131].

Chapter 3

Research Contributions

The thesis is presented as a collection of articles. In this chapter, the research work of these articles is described and contextualised in relation to previous work. The following papers are part of this collection:

Paper A **Quasi Spin Images**
Bart Iver van Blokland, Theoharis Theoharis, Anne C. Elster
In *Proceedings of the Norsk IKT-konferanse for forskning og utdanning (NIKT)*, 2018.

Paper B **Microshapes: Efficient Querying of 3D Object Collections based on Local Shape**
Bart Iver van Blokland, Theoharis Theoharis
In *Proceedings of the Eurographics Workshop on 3D Object Retrieval*, The Eurographics Association, 2018.

Paper C **Radial Intersection Count Image: a Clutter Resistant 3D Shape Descriptor**
Bart Iver van Blokland, Theoharis Theoharis
In *Computers & Graphics, Volume 91*, Elsevier, 2020.
Awarded the *Graphics Replicability Stamp* by the Graphics Replicability Stamp Initiative (GRSI).

Paper D **An Indexing Scheme and Descriptor for 3D Object Retrieval Based on Local Shape Querying**
Bart Iver van Blokland, Theoharis Theoharis
In *Computers & Graphics, Volume 92*, Elsevier, 2020.
Awarded the *Graphics Replicability Stamp* by the Graphics Replicability Stamp Initiative (GRSI).

Paper E **Partial 3D Object Retrieval using Local Binary QUICCI Descriptors and Dissimilarity Tree Indexing**
Bart Iver van Blokland, Theoharis Theoharis
In *Computers & Graphics, Volume 100*, Elsevier, 2021.
Awarded the *Graphics Replicability Stamp* by the Graphics Replicability Stamp Initiative (GRSI).

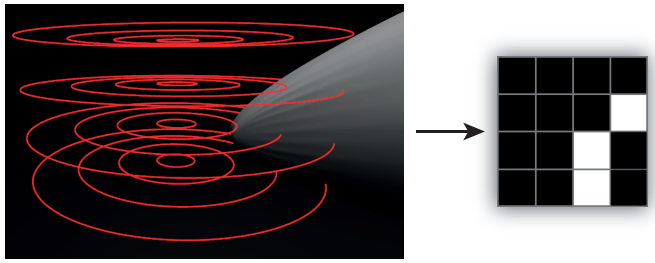


Figure 3.1: Visualisation of the computation of a simple QUICCI descriptor. The intersection count circles are shown in the left, while the produced binary descriptor is shown on the right. In the descriptor, black pixels represent bits set to 0 and white pixels are bits set to 1.

3.1 Contribution Overview

As outlined previously, the work done as part of this thesis can broadly be categorised into three primary contributions. While these contributions are explained in detail in their respective papers [132] [133] [134] [135] [136], we provide a brief explanation here.

3.1.1 Primary Contribution 1: The RICI and QUICCI descriptors

Both the RICI and QUICCI descriptor are computed for an oriented point consisting of a position and a surface normal. Both descriptors share the general means by which they are constructed, which is shown on the left hand side in Figure 3.1. The Figure shows a stack of 4 layers, each containing 4 circles with increasing radii. Some circles intersect the surface of a given object, while others do not. The position and orientation of the circles is derived from the oriented point for which the descriptor is constructed.

The RICI descriptor counts the number of times each circle intersects the surface of the object, creating a histogram where each bin corresponds to exactly one circle. Meanwhile, the QUICCI descriptor denotes whether intersection counts between one circle and its immediate neighbour on the same layer *changed*. Whether such a change occurs is a boolean value, which means the QUICCI descriptor is best stored as a binary image.

In a more formal sense, the descriptors are sampling a function $F(P,h,r)$, where P is the oriented point for which the descriptor is computed, h is the vertical displacement of the layer of circles relative to P's position and direction, and r the radius of the circle within that layer. $F(P,h,r)$ returns the number of times that particular circle intersects the mesh surface. Sampling this function in regular intervals of h and r for P produces the layered structure of circles shown in Figure 3.1.

According to our experiments, measuring changes in intersection counts tends to be more resistant to clutter than comparing absolute intersection counts, as clutter generally expresses itself on the descriptors as regions or patches of added intersections. In addition to the descriptors themselves, several other relevant contributions were proposed:

- The *Clutterbox* experiment for evaluating the effects of clutter on the matching performance of a descriptor.
- Optimisations for efficiently computing RICI and QUICCI descriptors.

- Distance functions for ranking RICI and QUICCI descriptors, one optimised for clutter resistant matching, and the Weighted Hamming distance function for descriptor relevance.

- A variant of the QUICCI construction algorithm which makes the descriptor more suitable for partial shape retrieval.

- Publicly available GPU implementations for the Spin Image, Fast Point Feature Histogram, 3D Shape Context, and the proposed RICI and QUICCI descriptors.

The Weighted Hamming distance function is worth noting in particular. It weights both bit errors (1 should be 0, 0 should be 1) each by a constant weight. The function is asymmetric; the pair of weights are based on a *needle* descriptor, which is being compared to a possibly matching *haystack* descriptor. Each type of bit error is weighted by the number of such errors that can occur. In a sparse descriptor with few bits set to 1, a needle descriptor bit set to 1 being 0 in the haystack results in significantly more distance than a 0 in the needle set to 1 in the haystack. A needle descriptor with exactly half of its bits set to 1 would result in weights effectively equivalent to the original Hamming distance function.

3.1.2 Primary Contribution 2: The Hamming and Dissimilarity Tree

For the purpose of retrieving any binary descriptor (a sequence of one or more bits), two indexing strategies were developed. Although their envisioned purpose was for retrieving similar QUICCI descriptors, both indexing structures can be used for any set of binary descriptors.

The Hamming tree exploits that the number of bits set to 1 in a binary descriptor is sufficient to compute a minimum Hamming distance to another descriptor. For example, if two descriptors respectively have 4 and 6 bits set to 1, their Hamming distance must at least be 2. The tree uses this by grouping descriptors by their bit counts, removing a portion of bits, and repeating the procedure. The minimum Hamming distance can be used to prune irrelevant branches while querying the tree.

While the Hamming tree is solely able to use the Hamming distance function, the Dissimilarity tree is able to retrieve nearest neighbours when using either the regular Hamming or Weighted Hamming distance functions. Its subdivision of descriptors relies on the principle that the distribution of bits within a descriptor is not random. At each node, it subdivides descriptors into two roughly equal sets, one called the *similar* set, and the other the *dissimilar*. The similar set is created by grouping descriptors which all have a particular set of bits set to the same bit value. The remainder is put into the dissimilar set. This procedure is repeated for both of the created subsets to create the tree. Equivalently set bits allow the computation of a minimum distance of a group of indexed descriptors to a query, allowing branches to be pruned.

The strengths of these indexing structures are complimentary. When it is known that similar descriptors can be expected to have a low Hamming distance, the Hamming tree is likely the best choice, while the Dissimilarity tree has been shown to perform well when the Hamming distance between nearest neighbours is high.

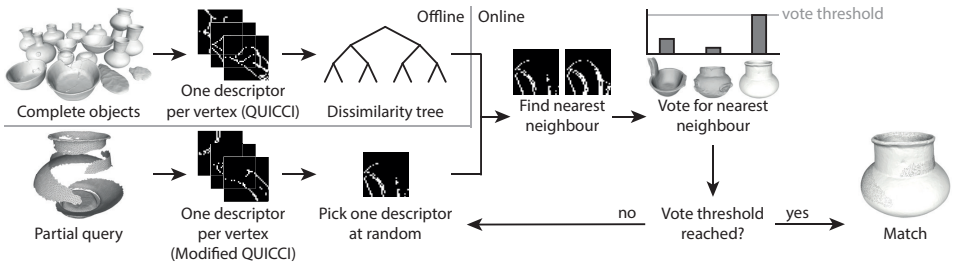


Figure 3.2: Visualisation of the proposed partial retrieval pipeline.

3.1.3 Primary Contribution 3: A complete partial retrieval pipeline

The final primary contribution of this thesis is a pipeline combining most of the contributions listed in this Section, a visualisation of which is shown in Figure 3.2. It relies on the QUICCI descriptor and Dissimilarity tree, and is able to efficiently locate nearest neighbours to partial query objects in a set of previously indexed complete ones. We show that a simple voting scheme is sufficient to achieve near perfect retrieval accuracy under ideal conditions, and good accuracy in a more realistic setting.

One issue of particular note is that partial queries contain surfaces which end abruptly, where the surface of the complete object would otherwise continue. Such edges normally cause responses in QUICCI descriptors, which are not desirable for partial object retrieval. We show that a simple modification to the QUICCI construction algorithm can reduce the average of 84.2 such unwanted bit responses per descriptor to 4.06 bits.

3.2 Paper Overview

Paper A - Quasi Spin Images

This paper is the first to propose the Radial Intersection Count Image descriptor, referred to as a *Quasi Spin Image* in this early paper. The descriptor is explained as a variant of the Spin Image that can be efficiently computed on the GPU and, in contrast to the Spin Image, is free of noise. In the evaluation it is shown that it can be generated significantly faster on the GPU than on a CPU or a GPU reference implementation of the Spin Image descriptor. Its matching performance is measured by computing the degree to which the similarity of image pairs correlate to those of Spin Image descriptors.

The paper is in its essence a progress report on the ongoing development of the RICI descriptor, showing early promising results. However, the descriptor's applicability and evaluation are greatly improved in Paper C. Most notably, while the measured correlation of correlations shows that the RICI and SI descriptors to some extent behave similarly, it does not measure their respective matching capabilities using an absolute measure.

Paper B - Microshapes: Efficient Querying of 3D Object Collections based on Local Shape

This paper proposes the Microshape descriptor, a binary descriptor which responds to decreasing surface intersection counts between pairs of circles laid out in a grid similar to those used to construct RICI descriptors. A distance function which can be used for

comparing QUICCI descriptors is also proposed, and applied in the evaluation on the retrieval of 4 hand drawn local shapes. The queries are executed on a database of objects, where some matches are visualised, and show that retrieved results do indeed contain the desired shapes.

In a similar fashion to Paper A, this paper represents (at the time) ongoing work on the QUICCI descriptor. The difference between the Microshape image and QUICCI descriptor is that the latter also includes rising intersection counts. The proposed scheme for hand drawing local shapes which can subsequently be located in a database of objects is unique to this paper. While the paper's evaluation only does a qualitative analysis, it may be possible to expand on this idea in future work.

Paper C - Radial Intersection Count Image: a Clutter Resistant 3D Shape Descriptor

The Radial Intersection Count Image descriptor is introduced, and an algorithm is given for generating them efficiently on the GPU. A proposed distance function exploits features of the RICI descriptor to allow for clutter resistant matching. To evaluate this, an experimental framework called the *Clutterbox* experiment is proposed, which allows the quantification of matching performance degradation of a descriptor caused by increasing levels of clutter.

The experiment is used to show that the RICI descriptor outperforms the Spin Image and 3DSC descriptors in matching tasks, even with increased levels of clutter. RICI descriptors are also shown to be faster to generate, and when a distance limit is set, much faster to compare than Spin Image and 3DSC descriptors. Finally, testing done as part of the evaluation could not confirm the claimed matching performance improvement of the Spin Image by Johnson et al. [23] when using a Support Angle, as is commonly done in the literature.

Paper D - An Indexing Scheme and Descriptor for 3D Object Retrieval Based on Local Shape Querying

The Quick Intersection Count Change Image (QUICCI) descriptor is proposed, a binary image representing changes in intersection counts between circles. While sharing many properties with the RICI descriptor, its binary nature implies lower storage and bandwidth requirements, improving density and significantly increasing matching speed. Moreover, a slight improvement in matching performance in cluttered scenes was observed.

In order to facilitate efficient retrieval of similar descriptors, the Hamming tree is proposed as an acceleration structure for indexing and locating nearest neighbours to binary descriptors. While the structure is tested on QUICCI descriptors in the paper, the tree can be applied on any binary descriptor. Furthermore, the Weighted Hamming distance function is proposed, which is shown to rank QUICCI descriptors in an improved manner.

Paper E - Partial 3D Object Retrieval using Local Binary QUICCI Descriptors and Dissimilarity Tree Indexing

A complete pipeline for partial 3D object retrieval is presented, using the previously proposed QUICCI descriptor. The pipeline consists of an online and an offline component.

The offline portion consists primarily of an indexing structure called the *Dissimilarity Tree*, which allows the efficient retrieval of QUICCI and other binary descriptors. Most

notably, the tree improves upon the Hamming tree, being capable of efficiently locating nearest neighbours when using the Weighted Hamming distance function.

The pipeline also utilises a proposed modification to the QUICCI descriptor, which improves matching performance in partial retrieval tasks. A voting scheme subsequently allows good retrieval accuracy in reasonably short execution times.

3.3 Technical Contributions

In addition to the aforementioned papers, the research carried out resulted in several libraries and frameworks that allow the main contributions to be adapted and integrated into other projects. All of these have been made freely available to the community under an open source license.

libShapeDescriptor
: A library containing GPU implementations for generating and comparing the RICI, QUICCI, SI, 3DSC, and FPFH descriptors efficiently on the GPU. After its initial implementation, it has been extensively refactored to improve its ease of use. A variety of useful utilities such as fast loaders for common 3D object file formats [1], reading and writing descriptors to disk in compressed files, and functions for visualising applicable descriptors in image files are also included.

  Repository URL: `https://github.com/bartvbl/libShapeDescriptor`

Clutterbox Experiment
: A framework which contains an implementation of, and allows arbitrary shape recognition methods to be tested using, the proposed Clutterbox Experiment in Paper C for quantifying clutter resistance. Support for additional methods can be added by implementing two functions.

  Repository URL: `https://github.com/bartvbl/Clutterbox`

Hamming Tree
: A reference implementation of the Hamming Tree indexing structure proposed in Paper D.

  Repository URL: `https://github.com/bartvbl/Hamming-Tree`

Dissimilarity Tree
: A reference implementation of the Dissilimarlity Tree indexing structure proposed in Paper E.

  Repository URL:
  `https://github.com/bartvbl/Dissimilarity-Tree-Reproduction`

[1] For example, loading an ascii PLY file of over 1GB in size took only a few seconds.

Also noteworthy are the repositories produced as part of the applications for the Graphics Replicability Stamp Initiative (GRSI). These allow all results of Papers C, D, and E to be replicated identically, with only a few exceptions (such as execution times). Each repository contains a script capable of automatically downloading datasets, compiling source code, and invoking any commands necessary to replicate the results shown in each figure of its respective paper.

Paper C Results	`https://github.com/bartvbl/Radial-Intersection-Count-Image-reproduction`
Paper D Results	`https://github.com/bartvbl/Quick-Intersection-Count-Change-Image-Reproduction`
Paper E Results	`https://github.com/bartvbl/Dissimilarity-Tree-Reproduction`

Chapter 4

Discussion

In this chapter, we address several topics of interest. The most important of which pertains to how shape descriptors tend to be evaluated, which is discussed in Section 4.1. We subsequently speculate on what may be the cause of clutter resistance (or the lack thereof) in various descriptors tested as part of this Thesis in Section 4.2. Section 4.3 analysis the contributions of the RICI and QUICCI descriptors in light of previous work. Some lessons which have been learned about indexing binary descriptors are discussed in Section 4.4, and finally various aspects related to the GPU implementations of several descriptors are documented in Section 4.5.

4.1 Evaluating Matching Performance of Local Shape Descriptors

The primary aim of a local shape descriptor is to describe surfaces present in its vicinity, simplifying the process of determining the degree of similarity to other surfaces. When a new descriptor is proposed, its ability to do so must be evaluated. Unfortunately, the evaluations of most papers show a high variation in the matching performance of different algorithms. This suggests that there exist underlying issues with these evaluation strategies, rendering them unable to thoroughly evaluate the quality of a descriptor. While this problem is not easily solved, we can speculate on its causes, and potential avenues by which it may be addressed in the future.

There are a number of factors which can influence the quality of descriptor evaluations: the used dataset (Section 4.1.1), the parameters for generating and comparing descriptors (Section 4.1.2), and the means by which matching performance is measured (Section 4.1.3).

4.1.1 Datasets

The current means by which new descriptors are generally evaluated is by testing their matching capabilities on various benchmark datasets. A wide variety of such datasets has been proposed to date, inlcuding work by Tombari et al. [45] [62], Mian et al. [137], and Koforenko et al. [138]. Additionally, the annual 3D SHape REtrieval Challenge (SHREC) [139] consists of multiple tracks, each targeting a 3D retrieval task [140] [30] [141]. The datasets made available as part of these tracks have also been subsequently used in papers for evaluation purposes [49] [114]. An overview over the most commonly used datasets for descriptor evaluation, as well as some SHREC track datasets used for evaluations in this thesis, is shown in Table 4.1.

These datasets tend to consist of a set of needle objects, one or more of which must be found in a number of haystack objects. The haystack objects are usually scans of the needle objects in various arrangements and poses. The sets can be classified according to several properties:

Noise	Indicates the degree to which the captured 3D surfaces deviate from the ground truth. Such noise may be induced by the scanner(s) used to capture the objects, or added artificially.
Occlusion	Occurs when a particular portion of the object remains invisible to the scanner, causing portions of the reconstructed surface to be missing.
Clutter	The amount of geometry present in the haystack object which is not part of a given needle object, which may affect the ability of a descriptor to correctly match corresponding surface points.
Needle Object Count	The number of needle objects contained in the dataset.
haystack Object Count	The number of haystack objects contained in the dataset.

There are several noteworthy observations which can be made from the listed datasets. First, the number of needle objects is low for the commonly used (first 7) sets. It is not unlikely that this is a major contributor to the volatility in results. Figure 9.8 in Paper D shows that despite no clutter objects having been added, all tested descriptors have widely varying nearest neighbour recognition rates. For example, the Spin Image has experiments where both close to none and nearly all nearest neighbours were correctly identified. Additionally, over a total of 1500 experiments, the difference in matching performance between some of the curves is not large. Authors which therefore test their method on datasets containing few needle and haystack objects likely renders, according to our findings, their evidence more anecdotal than substantial [60] [25] [66].

The low number of needle (and to some extent haystack) objects in these datasets also causes the variation in shapes encountered by the evaluated algorithm to be low. This also somewhat applies to the larger datasets listed, where objects are subdivided into classes of similar shapes. While necessary for object classification, none of the datasets in Table 4.1 can be considered representative for the wide variety of shapes which can be encountered in a human environment. It is also worth noting that none of the datasets contain scans without any of the needle objects, or contain objects which are not part of any of the primary object classes, which would aid in testing for false positives.

The quality of the meshes contained in the datasets can also vary. A notable example encountered as part this thesis was the SHREC'17 ShapeNet Core dataset. Depending on the object, individual faces had been flipped, normal vectors were incorrect, or missing from the source file entirely. Some object files also did not adhere to the file specification. These issues were partially solved, for example by recomputing normals when loading the objects; however, depending on the decisions of other authors, not all results may be exactly comparable to other works.

Finally, while several datasets have perturbed vertex positions in the normal direction in an effort to apply various levels of noise, it may not be representative to all noise which may be encountered in practical settings. For example, when a 3D scanner captures the same surface multiple times, the precise vertex positions chosen may differ between successive scans. Since vertices tend to be used as reference points when matching surfaces, even good quality scans may contain noise from the perspective of the used descriptor. To our knowledge, no current dataset includes noise caused by such retriangulated surfaces.

Dataset	Noise	Occlusion	Clutter	Needle Object Count	Haystack Object Count
Space Time [45] [1]	High	Yes	Multiple Needle Objects, Background not removed.	6	12
Retrieval [45] [1]	Medium to High	No	None	6	18
Bologna [62] [2]	Low	No	Multiple Needle Objects	6	45
Random Views [45] [1]	Medium to High	Yes	Multiple Needle Objects	6	36
Kinect [45] [1]	High	Yes	Multiple Needle Objects, Background not removed.	17	28
UWA [50] [137] [1]	Low	Yes	Needle Objects and Noise	5	50
Queens [142] [3]	Medium	Yes	Multiple Nedle Objects, Other geometry present.	5	80
3DV Pose [143] [4]	Medium	Yes	Multiple Needle Objects	50	294
SHREC'13 Partial Retrieval [144] [5]	Low	Yes	None	7200	360
SHREC'16 Partial Retrieval [123] [6]	Low	Yes	None	42	383
SHREC'16/17 Large Scale [145] [7] [8]	Low	No	None	10266	40924

Table 4.1: Overview over commonly used datasets, as well as some SHREC track datasets used in this thesis.

[1] http://vision.deis.unibo.it/keypoints3d/?page_id=2
[2] http://vision.deis.unibo.it/research/78-cvlab/80-shot
[3] https://code.engineering.queensu.ca/rcvlab-databases/qr3d
[4] http://roboimagedata.compute.dtu.dk/?page_id=131
[5] http://dataset.dcc.uchile.cl/
[6] http://vc.ee.duth.gr/shrec16/
[7] https://vision.princeton.edu/projects/2016/modelnet/shrec16/
[8] https://shapenet.cs.stanford.edu/shrec17/

4.1.2 Parameters

The parameters and overall design of the evaluation methodology also affect the measured matching performance of a descriptor. For example, relevant parameters for Spin-Image like descriptors include the image resolution, the support radius, and the distance function used for comparisons.

Unfortunately, to date no meta-analysis has been done on the selection of such parameters. This complicates the comparison of descriptors, as it is unclear whether the tested configuration yields optimal performance. In some recent work, authors have opted to select the optimal support radius of each method empirically using a portion of a given dataset [146] [147] [148], however, these investigations remain uncommon. The default parameters proposed by the original authors tend to be used instead. The optimal parameters may also vary between scenes, objects, or even individual points.

The need for an aforementioned meta-analysis is also illustrated by our evaluation of the Spin Image's support angle parameter in Paper C. The performed experiments could not confirm the claimed benefits by the original authors, yet the filtering step is commonly used in evaluations in literature.

In addition to optimal parameter selection, chosen parameters must also maintain fairness between methods being evaluated. Parameters such as the support radius may capture different surfaces across descriptors due to variations in the support volume shapes across descriptors (cylindrical, spherical, or otherwise). As a result some descriptors may receive different quantities of information about the surface, or are required to resist a different level of clutter.

One particular case where fairness was relevant for this thesis was in the testing of Papers C and D. While the RICI and QUICCI descriptors are computed from triangle meshes, the Spin Image, 3D Shape Context, and Fast Point Feature Histogram use a point cloud as input. Johnson et al. [23] solved this problem by using the mesh vertices as input points. However, in our experience that is commonly insufficient to adequately describe shape, and does not tend to produce uniformly distributed point clouds. For replicability and simplicity, we settled on a uniform sampling using an average number of point samples per triangle in our evaluation, but the ideal number of sample points may vary from object to object.

4.1.3 Measuring Matching Performance

As discussed in Section 2.1, local shape descriptors can have a number of properties which are generally considered to be beneficial. Some of these, such as invariance to mesh resolution, invariance to rotation, and memory requirements can for most descriptors be shown analytically. However, their ability to be discriminating, robust, and resistant to clutter and occlusion must be evaluated empirically.

The latter of these properties have to date been evaluated by the use of specific datasets which are known to include clutter, partial geometry, noise, or a combination of those. However, and as stated in Paper C as the motivation for the proposed *Clutterbox* experiment, this evaluation methodology tests the combined effects of two or more independent variables. It is thus difficult to deduce, based on the achieved matching performance, what part can for example be attributed to the descriptor's resistance to clutter and what to its innate descriptiveness.

Another potential issue is that the ability of a descriptor to correctly match points can vary on a point by point basis. When point matches are aggregated as average precision,

much of the nuance regarding the situations in which the descriptor excels or experiences difficulties is lost. Understanding these would be highly beneficial to determining which descriptors may be suitable for a given practical application. An improved benchmark should therefore analyse matching performance based on the conditions present within the region around an individual point, not the entire scene.

Furthermore, a scene may contain portions of surfaces which are similar in shape to a part of a given needle object, yet do not belong to the needle object itself. For example, even though a descriptor is able to correctly deduce that the cylindrical handlebars of a bike are similar in shape to the supports of a metal fence, they are semantically unrelated. Despite the descriptor performing its intended purpose, self-similarity within a scene in this manner complicates tasks such as object recognition, known as the sliding problem [149]. As one of the commonly used metrics for evaluating the quality of a descriptor is its ability to correctly identify points belonging to particular 'correct' objects in a scene, the sliding problem can result in cases where the tested descriptor is penalised for correctly identifying a matching surface. Measuring the degree to which the sliding problem affects matching performance is complicated by the need for a ground truth which determines similar points within a scene. Such correspondences are not easily established manually. It is likely that the sliding problem has affected the results presented in Papers C and D, as was noted in the discussion section of Paper C.

One final source of uncertainty when measuring the matching performance of a descriptor are salient or keypoint detectors. These are commonly used to reduce the number of descriptors being generated for a specific object, and aim to locate distinctive points which may be consistently found by that detector on similar surfaces. Tombari et al. [45] showed that keypoint detectors can significantly influence the average matching accuracy of a tested descriptor. This implies the portability of results is further reduced if two evaluations use different salient point detectors, especially since different detectors select different keypoints and may produce a varying quantity of them.

4.1.4 Conclusion

As we have discussed, there exist a number of factors which may influence the matching performance of a given descriptor. There is in addition no set standard for how shape descriptors ought to be evaluated to ensure fairness and accuracy. The resulting landscape causes it to be difficult to determine the strengths and weaknesses of a proposed descriptor based on current evaluation strategies employed.

We therefore consider it a necessity a standardised testing methodology is developed, which is capable of evaluating new descriptors in a more nuanced and quantitatively significant manner. Additionally, a meta-analysis of descriptor parameters should be performed to better understand their effects on performance, and what guidelines should be used for their evaluation in the future.

One reason for the lack of quantitative data in evaluations done thus far may be attributed to the popularity of Point Cloud Library (PCL) [22], which predominantly provides single-threaded, CPU-based implementations of a number of descriptors. The *libShapeDescriptor* library developed as part of this thesis could aid in this respect, as methods have been implemented on the GPU. Paper A showed that this for Spin Images led to much faster execution times.

4.2 Causes of Clutter Sensitivity

In Papers C and D, several of the descriptors mentioned in Section 2.1 were by our testing shown to perform worse in cluttered scenes. In this section, we speculate based on experience and gathered data over the course of the thesis about the possible causes to this, and whether using an alternate distance function may be able to remedy this in a similar fashion to the proposed RICI and QUICCI descriptors.

The effect of clutter on the spin image works in an additive fashion. As each pixel in the image measures the quantity of surface points projected on to them, clutter will manifest as additional points on top of those belonging to the object of interest. Depending on the extent and distribution of clutter in the support volume, regions of pixels in the image may remain unaffected by its effects. Unfortunately, since the noise applied by clutter is additive and unpredictable in magnitude, it is not possible to remove it from the descriptor posthumously or otherwise negate its impact.

When using default settings, the Fast Point Feature Histogram consists of 33 bins. As the descriptor is computed by summing various angular properties within the support volume around a point, adding clutter into this volume will become additive noise in the computed histogram bins. The effect on matching performance should thus be similar as to that experienced by the spin image. However, due to the compact size of the FPFH descriptor, a comparatively large portion of the descriptor can be expected to be affected. While the experiments performed by Guo et al. [51] are quantitatively limited, they do classify the descriptor as being sensitive to clutter, in agreement with our own findings.

The 3DSC descriptor, as with the Spin Image and FPFH descriptors, experiences clutter mostly as additive noise, albeit somewhat negated by the local point density normalisation factor. However, the descriptor's spatial subdivision of its support region may allow an alternate distance function to disregard histogram bins affected by clutter, when compared to a descriptor known to be clutter free. On the other hand, its poor best-case matching performance suggests it is likely not a worthwhile endeavour.

While the descriptor has not been evaluated in our experiments, an alternate distance function for the Spin Contour may be to a certain extent able to counteract the effects of clutter, given that portions of the original contour remain unaffected by clutter.

As discussed in Papers C and D, the variations in intersection counts utilised by the RICI and QUICCI descriptors remain mostly intact by the effects of clutter, allowing the descriptors (when using specific distance functions) to be largely unaffected by its effects.

4.3 Novelty of the RICI and QUICCI descriptors

Both descriptors rely on a grid of circles measuring intersection counts with the surface of an object, as illustrated in Figure 3.1. To our knowledge, they are the first descriptors utilising variations in circle-surface intersection counts for matching purposes.

The resulting descriptors can be thought of to capture surface contours present within the support volume. Contour based retrieval is in itself not a novel concept [150], however, previous methods have relied on object silhouettes [47] [61] [151] [152] [153] or slices [154] [155]. The locality of the computed surface contours in the descriptors means surface details within an object can be captured in a format which is intuitive for humans, which silhouette approaches cannot do. An important downside of slice-based methods is that slice orientations and positions must be chosen in advance. Instead, the proposed descriptors can

capture approximations of such slices in a rotation-invariant fashion.

Additionally, while we have not evaluated this quantitatively, anecdotal evidence in Figures 7.9, 7.10, 7.11, and 7.12 suggests it may also be possible to retrieve desirable surface contours by constructing their descriptors manually.

The Spin Contour developed by Liang et al. [60] [61] is worth mentioning in particular, which happens to share some properties of the RICI and QUICCI descriptors proposed by us. However, despite its similarities, the method primarily captures transformed outlines of objects and must be computed with a comparatively large support radius and image resolution to ensure all relevant outlines are captured. This causes it to behave more as a global rather than local descriptor.

With respect to the proposed *Clutterbox* evaluation, the idea of placing randomly oriented objects near an object of interest is not a unique idea [62]. However, its application for measuring clutter resistance is to our knowledge novel.

4.4 Indexing Binary Descriptors

This section discusses the most important problems encountered during the development of the Hamming and Dissimilarity trees for indexing binary descriptors and their effects on the proposed solutions. These problems are related to the branching factor of the indexing tree, and the query algorithm's ability to prune irrelevant branches. As a result of these, a significant number of potential indexing strategies were discarded as they would result in infeasible memory and/or execution times.

4.4.1 Tree Branching Factor

The branching factor at each indexing tree node has implications on its memory requirements and the execution time of querying the tree. When the branching factor is low or an upper bound is known, pointers to child nodes can be stored compactly inside a fixed size array. When either the branching factor is high, or its upper bound is large or cannot be statically determined, or both, then a sparse container such as a hashmap or dynamically expanding array must be used instead. While these grant flexibility, they also cause CPU cache misses due to the fragmentation of information relevant for querying; in particular hashmaps tend to have much higher memory footprints than arrays.

One example of an indexing strategy that was found to be infeasible for these reasons is a scheme which clusters descriptors by their similar counterparts at lower resolutions. For each indexed descriptor, a pyramid of smaller descriptors is computed, each level converting the level below it into an image at a quarter of the resolution by representing each 2x2 bit region as a single bit using the OR operator[9]. For example, if a leaf node contains a 32x32 bit descriptor, its parent node would be a parent to all descriptors having the same 16x16 bit simplified version. While the branching factor of 2x2 or 4x4 descriptors is sufficiently low that a static array can be allocated containing all permutations, each subsequently taken branch quadruples the number of bits, soon making the size of the required array too large.

Another effect of the high branching factor is that it causes the produced tree structure to be very shallow. This is a problem because it severely limits the speed of retrieval by complicating the discovery of potential neighbours, and a large number of branches must be

[9] In a similar fashion, the scheme where levels in the pyramid are computed by representing 1x2 and 2x1 bit regions in an alternating manner has similar problems.

considered at each node. When the branching factor is high, each branch in turn disqualifies a comparatively low number of descriptors. Moreover, due to the nature of the OR operator, many descriptors tend to have simplified versions mostly consisting of bits set to 1, causing a small number of nodes to have most of the branches that need to be considered when searching for neighbours (and thus an unbalanced tree).

Both the Hamming and Dissimilarity trees solve this problem by having a limited branching factor. The top layer of the Hamming tree has a number of branches at most equal to the length of the descriptor in bits, and, at each successive layer, the number of bits removed from the descriptor at that layer. The Dissimilarity tree is a binary tree, and as such always has a branching factor of 2.

4.4.2 Branch Pruning

Apart from the branching factor, another issue that affects query execution time is the ability of an index and its associated querying algorithm to prune irrelevant branches (sets of descriptors). Unfortunately, the QUICCI descriptor inherently complicates this, particularly when the Weighted Hamming distance function is used.

While similar surfaces tend to produce similar QUICCI descriptors, there may in addition be other surfaces present within the support volume of the descriptor. These surfaces cause bits to be set to 1 in parts of the descriptor other than the region of interest. Locating nearest neighbours is therefore a process of determining the largest common subset of equivalently set bits. As the Weighted Hamming distance function depends on the query descriptor, no clear criterion exists by which descriptors may be clustered, making the construction of an appropriate index tree hard. An inappropriate tree means that more branches need to be visited before it can be established that all neighbours in the database have been retrieved, thus slowing query execution.

The Dissimilarity tree addresses this problem by attempting to cluster descriptors by regions of equivalently set bits, thus allowing hard guarantees about the minimum Weighted Hamming of all descriptors in a branch to the query descriptor.

4.5 GPU implementation of descriptors

This section discusses some lessons and observations that were learned about implementing descriptors as GPU kernels, which may assist in the development and porting of future methods. A total of five methods have been implemented as part of this thesis; the RICI, QUICCI, Spin Image, 3DSC, and FPFH. Some GPU specific terminology will be used throughout this discussion, which is explained in Section 2.2.

Using the GPU for generating and comparing descriptors was primarily motivated by these tasks being independent and requiring high memory throughput. The GPU is ideal for such workloads. Additionally, as indicated in Section 4.1, descriptor evaluations done to date have commonly not tested on a significant number of needle objects, haystack objects, or both. Executing the proposed clutterbox experiment 1500 times during our own evaluation for each tested descriptor would be intractable when using the CPU implementations available in Point Cloud Library (PCL) [22].

Unfortunately, as indicated in Section 2.1, the GPU has to date not been utilised in the evaluation of descriptors. To our knowledge, the only publicly available GPU implementations of several descriptors were in PCL. However, our own testing showed that

(at least at the time) these implementations may not have been entirely correct. This led to the implementation of several well known descriptors and the *libShapeDescriptor* library.

Storing the descriptor being computed in the SM's shared memory banks was found to be beneficial for all implemented methods. This approach requires one descriptor to be computed per thread block for optimal performance. It is also worth noting that the block sizes should be tuned, as larger descriptors reduce the number of thread blocks able to execute on the SM simultaneously.

One potential avenue to improve the rate at which most tested descriptors can be generated and compared is storing their histogram bins at lower precision. For the spin image, FPFH, and 3DSC this would imply using half precision 16-bit floats, while for the RICI descriptor 8 or 16-bit bins could be sufficient. The QUICCI descriptor is already at the lower limit of its precision. As the comparison of descriptors is mostly a memory bound operation, such a reduction will likely result in a significant speedup. This comes at the cost of a potential loss in matching performance, although its impact is not necessarily significant. One issue complicating using lower precision integers for RICI and QUICCI generation is the lack of support on current Nvidia GPU's for atomic integer additions smaller than 32-bit. This can be counteracted by performing the necessary additions using bit masks[10], but adds complexity to the kernel. Similarly, atomic half precision additions are only supported on recent GPU's.

[10]For instance, adding the bit mask `0x00010001` to an `unsigned int` is effectively equivalent to adding 1 to two neighbouring `unsigned short`s.

Chapter 5

Concluding Remarks

5.1 Conclusion

The work done in this thesis shows that variations in shape intersection counts can be encoded in compact, fast and effective descriptors, which we have shown can be successfully applied to clutter resistant recognition and partial retrieval. A new distance function, called Weighted Hamming, has been proposed which is capable of retrieving more relevant QUICCI descriptors than the original Hamming distance.

Taking the QUICCI descriptor as an example, if one were to select 50 representative keypoints for a given object, using 32x32 bit descriptors allows storing over 5 million objects in the memory banks of the Nvidia Tesla V100 GPU. Our measurements show that a linear search should allow these to be searched within a second using that processor. Moreover, when using the default dimensionality settings of popular previously proposed descriptors, the QUICCI descriptor is simultaneously among the most compact (see Table 5.1) and the most descriptive.

Unfortunately, current keypoint detectors are not robust enough and therefore one often saves a local descriptor for every object vertex. Having many descriptors requires indexing and the proposed Hamming and Dissimilarity trees are able to accelerate retrieval of QUICCI and other binary descriptors, as shown by the experiments performed.

A number of these contributions are as part of this thesis made available to the research community. This includes efficient implementations and utilities for the generation and comparison of a number of the descriptors on the GPU, an evaluation framework for measuring the resistance to clutter called the *Clutterbox* experiment, and making *all* publications open access.

5.2 Future Work

There exist avenues for further investigation and improvements to the methods presented in this thesis, which are listed below.

5.2.1 RICI and QUICCI descriptors

As suggested in Paper B (see Chapter 7), while it is possible to compute Microshape or QUICCI descriptors over existing geometry, desirable local surface contours can also be drawn manually. The paper does not formally evaluate this idea, but it does show that the approach has potential. One possible application may be sketch-based retrieval.

Likewise, Figure 7.8 in Paper B (see Chapter 7) shows a hand drawn descriptor detecting objects with repeating elements. It would be interesting to determine whether such repeating elements or self-symmetries can be inferred automatically from the descriptors themselves. Particularly Figure 7.8d from Paper B shows that such patterns can be detected by the descriptor in the presence of complex geometry. Previous work attempting this has in part utilised voxels [156] [157] for this purpose, which does not always yield consistent results.

Descriptor	Default Size (bytes)
3D Shape Context	7,920
Universal Shape Context	7,920
TriSI	2,700
Local Surface Patches	2,312
SHOT	1,408
Spin Image	900
Rotational Projection Statistics	540
Point Feature Histogram	500
Fast Point Feature Histogram	132
THRIFT	128
RICI (64x64)	4,096
RICI (32x32)	1,024
QUICCI (64x64)	512
QUICCI (32x32)	128

Table 5.1: An overview over space requirements of different well-known descriptors, each using the default settings proposed by their respective authors, compared to the descriptors proposed as part of the work in this thesis. Where applicable, 32-bit floating point numbers are used. Additionally, for the RICI descriptor, one byte per pixel is used, which is sufficient precision for most use cases.

The QUICCI descriptor is constructed by locating changes in intersection counts between circles and the object surface. In this thesis, these changes are observed between adjacent circles on the same layer (see Figure 3.1). From the perspective of the descriptor, this implies a change in intersection counts between horizontally neighbouring pixels. However, vertical intersection count changes have not yet been investigated in detail, and may improve matching performance when combined with the horizontal ones.

In a similar fashion, instead of computing a single QUICCI descriptor, a pair of descriptors can be computed, each indicating where positive or negative intersection count changes occurred. This can allow more precise query specification. For instance, when retrieving the handle of a mug, the inner edge of the handle will induce an increase in intersection counts, while the outer a reduction. The ability to specify which type of change is desired thus improves querying precision.

Dinh et al. [55] proposed multi-resolution Spin Images which either accelerate descriptor matching, or to some extent allow surfaces to be compared in a scale invariant manner. Given the shared properties of the proposed QUICCI and RICI descriptors with the original Spin Image, it may be possible to apply some of the ideas for similar purposes on the proposed descriptors.

As shown in Paper E, the QUICCI descriptor combined with the Weighted Hamming distance function may be sensitive to perturbations of vertices. While lower resolution descriptors still resulted in good retrieval performance, an alternate distance function which is better able to account for displacements of intersection count changes within the descriptor may be able to improve its matching capabilities.

5.2.2 Evaluating Descriptors

As discussed in Section 4.1, there is currently a need for improved evaluation procedures for new and previously proposed descriptors.

While, as part of this thesis, several popular descriptors were implemented on the GPU, there exist many others for which to date either only a publicly available CPU implementation exists, or the source code has not been made available at all. Implementing additional descriptors on the GPU would allow a deeper insight into which methods are most suitable for that style of processor. Additionally, existing implementations should be ported to support non-Nvidia GPU's for improved accessibility of the implementations to the research community.

As discussed in Section 4.1, it is currently insufficiently understood in which situations a descriptor performs best. This requires a more thorough, as well as quantitatively significant, benchmark framework to be constructed. Moreover, a meta-analysis of descriptor parameters should be performed, such that their optimal values can be selected. Optimal parameters may also be partially dependent on the mesh surface being captured by the descriptor, which too may prove an interesting avenue of investigation.

With respect to the evaluation of the RICI and QUICCI descriptors, their effectiveness has to date solely been tested on the SHREC'16 [123] and SHREC'17 [158] datasets while evaluating partial retrieval and clutter resistant matching performance, respectively. The descriptors should also be tested on datasets aimed at other matching tasks, as for example face recognition, mesh registration, and symmetry detection.

5.2.3 Indexing Binary Descriptors

The proposed Hamming tree removes a constant number of bits from descriptors for each layer, and categorises the remainder by the number of bits set to 1. It may be possible to improve retrieval execution times by varying the number of bits removed per layer depending on the binary descriptors being indexed.

For retrieval in large scale object datasets (such as 100,000 objects or more), indexing descriptors generated for every vertex of every object is not realistic. Therefore, an evaluation should be done of different keypoint detectors to determine which best improves retrieval performance for RICI and QUICCI descriptors, with respect to the exhaustive case.

5.3 Outlook

A wide adoption of 3D scanning equipment among consumers may allow 3D data to take a more prominent position among media such as images, video, and audio. However, for practical applications, efficiency will be a key metric, particularly on mobile hardware. Utilising graphics processors for this purpose, at least according to our testing, appears to be the logical next step towards this goal.

The size of descriptors ought to receive special attention here, as the reduced storage and bandwidth requirements is a significant factor in achieving efficient implementations on modern processors, in turn allowing their application on larger databases more effectively. Such databases will also benefit from descriptors which are sufficiently distinctive to allow efficient retrieval. The work discussed in this thesis may assist in this effort.

Bibliography

[1] Wenming Cao, Qifan Liu, and Zhiquan He. Review of Pavement Defect Detection Methods. *IEEE Access*, 8:14531–14544, 2020. Conference Name: IEEE Access.

[2] Upendra Mani Tuladhar Park, Hong-Seok. Development of an Inspection System for Defect Detection in Pressed Parts Using Laser Scanned Data. *Procedia Engineering*, 69:931–936, January 2014. Publisher: Elsevier.

[3] Wang Ying, Jin Cuiyun, and Zhang Yanhui. Pipe Defect Detection and Reconstruction Based on 3D Points Acquired by the Circular Structured Light Vision. *Advances in Mechanical Engineering*, 5:670487, January 2013. Publisher: SAGE Publications.

[4] Fabio Remondino. Heritage Recording and 3D Modeling with Photogrammetry and 3D Scanning. *Remote Sensing*, 3(6):1104–1138, June 2011. Number: 6 Publisher: Molecular Diversity Preservation International.

[5] Giovanna Sansoni, Marco Trebeschi, and Franco Docchio. State-of-The-Art and Applications of 3D Imaging Sensors in Industry, Cultural Heritage, Medicine, and Criminal Investigation. *Sensors*, 9(1):568–601, January 2009. Number: 1 Publisher: Molecular Diversity Preservation International.

[6] George Pavlidis, Anestis Koutsoudis, Fotis Arnaoutoglou, Vassilios Tsioukas, and Christodoulos Chamzas. Methods for 3D digitization of Cultural Heritage. *Journal of Cultural Heritage*, 8(1):93–98, January 2007.

[7] Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. Capturing Hands in Action Using Discriminative Salient Points and Physics Simulation. *International Journal of Computer Vision*, 118(2):172–193, June 2016.

[8] Qin Cai, David Gallup, Cha Zhang, and Zhengyou Zhang. 3D Deformable Face Tracking with a Commodity Depth Camera. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, Lecture Notes in Computer Science, pages 229–242, Berlin, Heidelberg, 2010. Springer.

[9] Brook Galna, Dan Jackson, Guy Schofield, Roisin McNaney, Mary Webster, Gillian Barry, Dadirayi Mhiripiri, Madeline Balaam, Patrick Olivier, and Lynn Rochester. Retraining function in people with Parkinson's disease using the Microsoft kinect: game design and pilot testing. *Journal of NeuroEngineering and Rehabilitation*, 11(1):60, April 2014.

[10] Belinda Lange, Chien-Yen Chang, Evan Suma, Bradley Newman, Albert Skip Rizzo, and Mark Bolas. Development and evaluation of low cost game-based balance rehabilitation tool using the microsoft kinect sensor. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1831–1834, August 2011. ISSN: 1558-4615.

[11] Ihn-Sik Weon, Soon-Geul Lee, and Jae-Kwan Ryu. Object Recognition Based Interpolation With 3D LIDAR and Vision for Autonomous Driving of an Intelligent Vehicle. *IEEE Access*, 8:65599–65608, 2020. Conference Name: IEEE Access.

[12] Heng Wang, Bin Wang, Bingbing Liu, Xiaoli Meng, and Guanghong Yang. Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle. *Robotics and Autonomous Systems*, 88:71–78, February 2017.

[13] Dan McLeod, John Jacobson, Mark Hardy, and Carl Embry. Autonomous inspection using an underwater 3D LiDAR. In *2013 OCEANS - San Diego*, pages 1–8, September 2013. ISSN: 0197-7385.

[14] Kevin W. Bowyer, Kyong Chang, and Patrick Flynn. A survey of approaches and challenges in 3D and multi-modal 3D+2D face recognition. *Computer Vision and Image Understanding*, 101(1):1–15, January 2006.

[15] Yinjie Lei, Mohammed Bennamoun, Munawar Hayat, and Yulan Guo. An efficient 3D face recognition approach using local geometrical signatures. *Pattern Recognition*, 47(2):509–524, February 2014.

[16] Technical specifications of the Apple iPhone 12 Pro. `https://www.apple.com/iphone-12-pro/specs/`, July 2021.

[17] Technical specifications of the Apple iPhone 13 Pro. `https://www.apple.com/iphone-13-pro/specs/`, November 2021.

[18] Technical specifications of the Samsung Galaxy S20+ and S20 Ultra. `https://www.samsung.com/global/galaxy/galaxy-s20/specs/`, July 2021.

[19] Technical specifications of the Apple iPad Pro. `https://www.apple.com/ipad-pro/specs/`, July 2021.

[20] Nolan Davis, Dennis Braunreiter, Cezario Tebcherani, and Masatoshi Tanida. 3D object matching on the GPU using spin-image surface matching. In *Advanced Signal Processing Algorithms, Architectures, and Implementations XVIII*, volume 7074, page 707408. International Society for Optics and Photonics, September 2008.

[21] Adam R. Gerlach and Bruce K. Walker. Accelerating robust 3D pose estimation utilizing a graphics processing unit. In *Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*, volume 7878, page 78780V. International Society for Optics and Photonics, January 2011.

[22] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011. ISSN: 1050-4729.

[23] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, May 1999.

[24] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing Objects in Range Data Using Regional Point Descriptors. In Tomáš Pajdla

and Jiří Matas, editors, *Computer Vision - ECCV 2004*, Lecture Notes in Computer Science, pages 224–237, Berlin, Heidelberg, 2004. Springer.

[25] R. B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, May 2009. ISSN: 1050-4729.

[26] Bo Li, Yijuan Lu, Chunyuan Li, Afzal Godil, Tobias Schreck, Masaki Aono, Martin Brutscher, Qiang Chen, Nihad Karim Chowdhury, Bin Fang, Hongbo Fu, Takahiko Furuya, Haisheng Li, Jianzhuang Liu, Henry Johan, Ryuichi Kosaka, Hitoshi Koyanagi, Ryutarou Ohbuchi, Atsushi Tatsuma, Yanjuan Wan, Chaoli Zhang, and Changqing Zou. A comparison of 3D shape retrieval methods based on a large-scale benchmark supporting multimodal queries. *Computer Vision and Image Understanding*, 131:1–27, 2015. Accepted: 2015-02-23T14:29:46Z.

[27] Johan W. H. Tangelder and Remco C. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441, December 2007.

[28] Takahiko Furuya and Ryutarou Ohbuchi. Learning part-in-whole relation of 3D shapes for part-based 3D model retrieval. *Computer Vision and Image Understanding*, 166:102–114, January 2018.

[29] Guillaume Lavoué. *Bag of Words and Local Spectral Descriptor for 3D Partial Shape Retrieval*. The Eurographics Association, 2011. Accepted: 2013-04-25T14:10:26Z ISSN: 1997-0463.

[30] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. V. Nguyen, R. Ohbuchi, Y. Ohkita, Y. Ohishi, F. Porikli, M. Reuter, I. Sipiran, D. Smeets, P. Suetens, H. Tabia, and D. Vandermeulen. *SHREC '11 Track: Shape Retrieval on Non-rigid 3D Watertight Meshes*. The Eurographics Association, 2011. Accepted: 2013-04-25T14:10:27Z ISSN: 1997-0463.

[31] Zhouhui Lian, Afzal Godil, Benjamin Bustos, Mohamed Daoudi, Jeroen Hermans, Shun Kawamura, Yukinori Kurita, Guillaume Lavoué, Hien Van Nguyen, Ryutarou Ohbuchi, Yuki Ohkita, Yuya Ohishi, Fatih Porikli, Martin Reuter, Ivan Sipiran, Dirk Smeets, Paul Suetens, Hedi Tabia, and Dirk Vandermeulen. A comparison of methods for non-rigid 3D shape retrieval. *Pattern Recognition*, 46(1):449–461, January 2013. 116 citations (Crossref) [2021-03-06].

[32] Fang Wang, Le Kang, and Yi Li. Sketch-based 3D shape retrieval using Convolutional Neural Networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1875–1883, June 2015. 86 citations (Crossref) [2021-03-06].

[33] Anh-Phuong Ta, Christian Wolf, Guillaume Lavoue, and Atilla Baskurt. 3D Object Detection and Viewpoint Selection in Sketch Images Using Local Patch-Based Zernike Moments. In *2009 Seventh International Workshop on Content-Based Multimedia Indexing*, pages 189–194, Chania, Crete, June 2009. IEEE.

[34] Ivan Sipiran, Benjamin Bustos, and Tobias Schreck. Data-aware 3D partitioning for generic shape retrieval. *Computers & Graphics*, 37(5):460–472, August 2013.

[35] Sarah Tang and Afzal Godil. An evaluation of local shape descriptors for 3D shape retrieval. In *Three-Dimensional Image Processing (3DIP) and Applications II*, volume 8290, page 82900N. International Society for Optics and Photonics, January 2012.

[36] A. Liu, W. Nie, Y. Gao, and Y. Su. Multi-Modal Clique-Graph Matching for View-Based 3D Model Retrieval. *IEEE Transactions on Image Processing*, 25(5):2103–2116, May 2016. 88 citations (Crossref) [2021-03-06].

[37] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *2003 Shape Modeling International.*, pages 130–139, May 2003.

[38] Panagiotis Papadakis, Ioannis Pratikakis, Theoharis Theoharis, and Stavros Perantonis. PANORAMA: A 3D Shape Descriptor Based on Panoramic Views for Unsupervised 3D Object Retrieval. *International Journal of Computer Vision*, 89(2-3):177–192, September 2010.

[39] Shoki Tashiro, Atsushi Tatsuma, and Masaki Aono. Super-vector coding features extracted from both depth buffer and view-normal-angle images for part-based 3D shape retrieval. *Multimedia Tools and Applications*, 76(21):22059–22076, November 2017. 1 citations (Crossref) [2021-03-11].

[40] A. Liu, W. Nie, Y. Gao, and Y. Su. View-Based 3-D Model Retrieval: A Benchmark. *IEEE Transactions on Cybernetics*, 48(3):916–928, March 2018. Conference Name: IEEE Transactions on Cybernetics.

[41] B. Matei, Ying Shan, H.S. Sawhney, Yi Tan, R. Kumar, D. Huber, and M. Hebert. Rapid object indexing using locality sensitive hashing and joint 3D-signature space estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1111–1126, July 2006.

[42] John Novatnack and Ko Nishino. Scale-Dependent/Invariant Local 3D Shape Descriptors for Fully Automatic Registration of Multiple Sets of Range Images. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, Lecture Notes in Computer Science, pages 440–453, Berlin, Heidelberg, 2008. Springer.

[43] Evdokia Saiti, Antonios Danelakis, and Theoharis Theoharis. Cross-time registration of 3D point clouds. *Computers & Graphics*, 99:139–152, October 2021.

[44] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational Projection Statistics for 3D Local Surface Description and Object Recognition. *International Journal of Computer Vision*, 105(1):63–86, October 2013.

[45] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Performance Evaluation of 3D Keypoint Detectors. *International Journal of Computer Vision*, 102(1):198–220, March 2013.

[46] Walter Wohlkinger and Markus Vincze. Ensemble of shape functions for 3D object classification. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 2987–2992, December 2011.

[47] Jiaqi Yang, Qian Zhang, Ke Xian, Yang Xiao, and Zhiguo Cao. Rotational contour

signatures for both real-valued and binary feature representations of 3D local shape. *Computer Vision and Image Understanding*, 160:133–147, July 2017.

[48] Yue Gao, Qionghai Dai, and Nai-Yao Zhang. 3D model comparison using spatial structure circular descriptor. *Pattern Recognition*, 43(3):1142–1151, March 2010.

[49] Ivan Sipiran and Benjamin Bustos. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer*, 27(11):963, July 2011.

[50] A. Mian, M. Bennamoun, and R. Owens. On the Repeatability and Quality of Keypoints for Local Feature-based 3D Object Retrieval from Cluttered Scenes. *International Journal of Computer Vision*, 89(2):348–361, September 2010.

[51] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A Comprehensive Performance Evaluation of 3D Local Feature Descriptors. *International Journal of Computer Vision*, 116(1):66–89, January 2016.

[52] Zhen-Bao Liu, Shu-Hui Bu, Kun Zhou, Shu-Ming Gao, Jun-Wei Han, and Jun Wu. A Survey on Partial Retrieval of 3D Shapes. *Journal of Computer Science and Technology*, 28(5):836–851, September 2013.

[53] Jrgen Assfalg, Marco Bertini, Alberto Del Bimbo, and Pietro Pala. Content-Based Retrieval of 3-D Objects Using Spin Image Signatures. *IEEE Transactions on Multimedia*, 9(3):589–599, April 2007. Conference Name: IEEE Transactions on Multimedia.

[54] Tal Darom and Yosi Keller. Scale-Invariant Features for 3-D Mesh Models. *IEEE Transactions on Image Processing*, 21(5):2758–2769, May 2012. Conference Name: IEEE Transactions on Image Processing.

[55] H.Q. Dinh and S. Kropac. Multi-Resolution Spin-Images. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 863–870, June 2006. ISSN: 1063-6919.

[56] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Jianwei Wan, and Min Lu. A novel local surface feature for 3D object recognition under clutter and occlusion. *Information Sciences*, 293:196–213, February 2015.

[57] Giuliano Pasqualotto, Pietro Zanuttigh, and Guido M. Cortelazzo. Combining color and shape descriptors for 3D model retrieval. *Signal Processing: Image Communication*, 28(6):608–623, July 2013.

[58] S. Ruiz-Correa, L.G. Shapiro, and M. Melia. A new signature-based method for efficient 3-D object recognition. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, December 2001. ISSN: 1063-6919.

[59] O. Carmichael, D. Huber, and M. Hebert. Large data sets and confusing scenes in 3-D surface matching and recognition. In *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062)*, pages 358–367, October 1999.

[60] Luming Liang, Andrzej Szymczak, and Mingqiang Wei. Geodesic spin contour for partial near-isometric matching. *Computers & Graphics*, 46:156–171, February 2015.

[61] Luming Liang, Mingqiang Wei, Andrzej Szymczak, Wai-Man Pang, and Meng Wang. Spin Contour. *IEEE Transactions on Multimedia*, 18(11):2282–2292, November 2016. Conference Name: IEEE Transactions on Multimedia.

[62] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique shape context for 3d data description. In *Proceedings of the ACM workshop on 3D object retrieval - 3DOR '10*, page 57, Firenze, Italy, 2010. ACM Press.

[63] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391, September 2008. ISSN: 2153-0866.

[64] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Learning informative point classes for the acquisition of object model maps. In *Robotics and Vision 2008 10th International Conference on Control, Automation*, pages 643–650, December 2008.

[65] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[66] Alex Flint, Anthony Dick, and Anton van den Hengel. Thrift: Local 3D Structure Recognition. In *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)*, pages 182–188, December 2007.

[67] Samuele Salti, Federico Tombari, and Luigi Di Stefano. SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, August 2014.

[68] Sai Manoj Prakhya, Bingbing Liu, and Weisi Lin. B-SHOT: A binary feature descriptor for fast and efficient keypoint matching on 3D point clouds. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1929–1934, September 2015.

[69] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *Computer Graphics Forum*, 28(5):1383–1392, 2009. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2009.01515.x.

[70] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, July 2017. ISSN: 1063-6919.

[71] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. ShellNet: Efficient Point Cloud Convolutional Neural Networks Using Concentric Shells Statistics. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1607–1616, October 2019. ISSN: 2380-7504.

[72] Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompat-

siaris. Deep Learning Advances in Computer Vision with 3D Data: A Survey. *ACM Computing Surveys*, 50(2):20:1–20:38, April 2017.

[73] NVIDIA Ampere Architecture In-Depth. `https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/`, May 2020.

[74] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, Lecture Notes in Computer Science, pages 778–792, Berlin, Heidelberg, 2010. Springer.

[75] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, November 2011. ISSN: 2380-7504.

[76] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. BRISK: Binary Robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, November 2011. ISSN: 2380-7504.

[77] Jingdong Wang, Ting Zhang, jingkuan song, Nicu Sebe, and Heng Tao Shen. A Survey on Learning to Hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):769–790, April 2018. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[78] Meina Kan, Dong Xu, Shiguang Shan, and Xilin Chen. Semisupervised Hashing via Kernel Hyperplane Learning for Scalable Image Search. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(4):704–713, April 2014. Conference Name: IEEE Transactions on Circuits and Systems for Video Technology.

[79] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1556–1564, June 2015. ISSN: 1063-6919.

[80] André Araujo and Bernd Girod. Large-Scale Video Retrieval Using Image Queries. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(6):1406–1420, June 2018. Conference Name: IEEE Transactions on Circuits and Systems for Video Technology.

[81] Dejin Zhang, Qin Zou, Hong Lin, Xin Xu, Li He, Rong Gui, and Qingquan Li. Automatic pavement defect detection using 3D laser profiling technology. *Automation in Construction*, 96:350–365, December 2018.

[82] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. *Theory of Computing*, 8(1):321–350, July 2012. Publisher: Theory of Computing Exchange.

[83] A.Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29, June 1997.

[84] Andrei Z. Broder. Identifying and Filtering Near-Duplicate Documents. In Raffaele

Giancarlo and David Sankoff, editors, *Combinatorial Pattern Matching*, Lecture Notes in Computer Science, pages 1–10, Berlin, Heidelberg, 2000. Springer.

[85] Caitlin Sadowski and Greg Levin. SimHash: Hash-based Similarity Detection. *Google Technical Report*, page 10, December 2007.

[86] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-Supervised Hashing for Large-Scale Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2393–2406, December 2012. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[87] Anirban Dasgupta, Ravi Kumar, and Tamas Sarlos. Fast locality-sensitive hashing. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 1073–1081, New York, NY, USA, August 2011. Association for Computing Machinery.

[88] Mayank Bawa, Tyson Condie, and Prasanna Ganesan. LSH Forest: Self-Tuning Indexes for Similarity Search. In *World Wide Web Conference (WWW)*, Chiba, Japan, 2005.

[89] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, SCG '04, pages 253–262, New York, NY, USA, June 2004. Association for Computing Machinery.

[90] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 950–961, Vienna, Austria, September 2007. VLDB Endowment.

[91] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, NIPS'08, pages 1753–1760, Red Hook, NY, USA, December 2008. Curran Associates Inc.

[92] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised Discrete Hashing. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 37–45, June 2015. ISSN: 1063-6919.

[93] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, and David Suter. Fast Supervised Hashing with Decision Trees for High-Dimensional Data. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1971–1978, June 2014. ISSN: 1063-6919.

[94] Go Irie, Zhenguo Li, Xiao-Ming Wu, and Shih-Fu Chang. Locally Linear Hashing for Extracting Non-linear Manifolds. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2123–2130, June 2014. ISSN: 1063-6919.

[95] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep Supervised Hashing for Fast Image Retrieval. *International Journal of Computer Vision*, 127(9):1217–1234, September 2019.

[96] Jian Zhang and Yuxin Peng. SSDH: Semi-Supervised Deep Hashing for Large Scale

Image Retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):212–225, January 2019. Conference Name: IEEE Transactions on Circuits and Systems for Video Technology.

[97] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3270–3278, June 2015. ISSN: 1063-6919.

[98] Gerth Stølting Brodal and Leszek Gasieniec. Approximate dictionary queries. In Dan Hirschberg and Gene Myers, editors, *Combinatorial Pattern Matching*, Lecture Notes in Computer Science, pages 65–74, Berlin, Heidelberg, 1996. Springer.

[99] S. Brin. Near Neighbor Search in Large Metric Spaces. Zurich, Switzerland, 1995.

[100] Abdullah N. Arslan and Ömer Eğecioğlu. Dictionary look-up within small edit distance. *International Journal of Foundations of Computer Science*, 15(01):57–71, February 2004. Publisher: World Scientific Publishing Co.

[101] Moritz G. Maaß and Johannes Nowak. Text Indexing with Errors. In Alberto Apostolico, Maxime Crochemore, and Kunsoo Park, editors, *Combinatorial Pattern Matching*, Lecture Notes in Computer Science, pages 21–32, Berlin, Heidelberg, 2005. Springer.

[102] Alex X. Liu, Ke Shen, and Eric Torng. Large scale Hamming distance query processing. In *2011 IEEE 27th International Conference on Data Engineering*, pages 553–564, April 2011. ISSN: 2375-026X.

[103] M. Norouzi, A. Punjani, and D. J. Fleet. Fast search in Hamming space with multi-index hashing. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3108–3115, June 2012. ISSN: 1063-6919.

[104] Timothy Chappell, Shlomo Geva, and Guido Zuccon. Approximate Nearest-Neighbour Search with Inverted Signature Slice Lists. In Allan Hanbury, Gabriella Kazai, Andreas Rauber, and Norbert Fuhr, editors, *Advances in Information Retrieval*, Lecture Notes in Computer Science, pages 147–158, Cham, 2015. Springer International Publishing.

[105] E. M. Reina, K. Q. Pu, and F. Z. Qureshi. An Index Structure for Fast Range Search in Hamming Space. In *2017 14th Conference on Computer and Robot Vision (CRV)*, pages 8–15, May 2017.

[106] Marius Muja and David G. Lowe. Fast Matching of Binary Features. In *2012 Ninth Conference on Computer and Robot Vision*, pages 404–410, May 2012.

[107] Jingkuan Song, Heng Tao Shen, Jianfeng Wang, Zi Huang, Nicu Sebe, and Jingdong Wang. A Distance-Computation-Free Search Scheme for Binary Code Databases. *IEEE Transactions on Multimedia*, 18(3):484–495, March 2016. Conference Name: IEEE Transactions on Multimedia.

[108] Jianbin Qin, Chuan Xiao, Yaoshu Wang, Wei Wang, Xuemin Lin, Yoshiharu Ishikawa, and Guoren Wang. Generalizing the Pigeonhole Principle for Similarity Search in Hamming Space. *IEEE Transactions on Knowledge and Data Engineering*, 33(2):489–

505, February 2021. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[109] Michalis A. Savelonas, Ioannis Pratikakis, and Konstantinos Sfikas. An overview of partial 3D object retrieval methodologies. *Multimedia Tools and Applications*, 74(24):11783–11808, December 2015.

[110] Ryutarou Ohbuchi, Kunio Osada, Takahiko Furuya, and Tomohisa Banno. Salient local visual features for shape-based 3D model retrieval. In *2008 IEEE International Conference on Shape Modeling and Applications*, pages 93–102, June 2008.

[111] Yi Liu, Hongbin Zha, and Hong Qin. Shape Topics: A Compact Representation and New Algorithms for 3D Partial Shape Retrieval. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2025–2032, June 2006.

[112] Alexander M. Bronstein, Michael M. Bronstein, Leonidas J. Guibas, and Maks Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics*, 30(1):1:1–1:20, February 2011.

[113] Hela Haj Mohamed and Samir Belaid. Algorithm BOSS (Bag-of-Salient local Spectrums) for non-rigid and partial 3D object retrieval. *Neurocomputing*, 168:790–798, November 2015.

[114] Michalis A. Savelonas, Ioannis Pratikakis, and Konstantinos Sfikas. Fisher encoding of differential fast point feature histograms for partial 3D object retrieval. *Pattern Recognition*, 55:114–124, July 2016.

[115] Dimitrios Dimou and Konstantinos Moustakas. Fast 3D Scene Segmentation and Partial Object Retrieval Using Local Geometric Surface Features. In Marina L. Gavrilova, C.J. Kenneth Tan, and Alexei Sourin, editors, *Transactions on Computational Science XXXVI: Special Issue on Cyberworlds and Cybersecurity*, Lecture Notes in Computer Science, pages 79–98. Springer, Berlin, Heidelberg, 2020.

[116] Alexander Agathos, Ioannis Pratikakis, Panagiotis Papadakis, Stavros Perantonis, Philip Azariadis, and Nickolas S. Sapidis. 3D articulated object retrieval using a graph-based representation. *The Visual Computer*, 26(10):1301–1319, October 2010.

[117] Julien Tierny, Jean-Philippe Vandeborre, and Mohamed Daoudi. Partial 3D Shape Retrieval by Reeb Pattern Unfolding. *Computer Graphics Forum*, 28(1):41–55, 2009.

[118] T Furuya, S Kurabe, and R Ohbuchi. Randomized Sub-Volume Partitioning for Part-Based 3D Model Retrieval. *3DOR '15 Proceedings of the 2015 Eurographics Workshop on 3D Object Retrieval*, pages 15–22, 2015.

[119] E. Wahl, U. Hillenbrand, and G. Hirzinger. Surflet-pair-relation histograms: a statistical 3D-shape representation for rapid classification. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 474–481, October 2003.

[120] H. Dutagaci, A. Godil, A. Axenopoulos, P. Daras, T. Furuya, and R. Ohbuchi. *SHREC'09 Track: Querying with Partial Models*. The Eurographics Association, 2009. Accepted: 2013-10-21T18:00:33Z ISSN: 1997-0463.

[121] R. Ohbuchi and T. Furuya. Scale-weighted dense bag of visual features for 3D model retrieval from a partial view 3D model. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 63–70, September 2009.

[122] Zhouhui Lian, Afzal Godil, Xianfang Sun, and Jianguo Xiao. CM-BOF: visual similarity-based 3D shape retrieval using Clock Matching and Bag-of-Features. *Machine Vision and Applications*, 24(8):1685–1704, November 2013.

[123] Ioannis Pratikakis, Michalis A. Savelonas, Fotis Arnaoutoglou, George Ioannakis, Anestis Koutsoudis, Theoharis Theoharis, Minh-Triet Tran, Vinh-Tiep Nguyen, V.-K. Pham, Hai-Dang Nguyen, Hoang-An Le, Ba-Huu Tran, Huu-Quan To, Minh-Bao Truong, Thuyen Van Phan, Minh-Duc Nguyen, Thanh-An Than, Cu-Khoi-Nguyen Mac, Minh N. Do, Anh-Duc Duong, Takahiko Furuya, Ryutarou Ohbuchi, Masaki Aono, Shoki Tashiro, David Pickup, Xianfang Sun, Paul L. Rosin, and Ralph R. Martin. SHREC'16 Track: Partial Shape Queries for 3D Object Retrieval. *Eurographics Workshop on 3D Object Retrieval*, page 10 pages, 2016. Artwork Size: 10 pages ISBN: 9783038680048 Publisher: The Eurographics Association.

[124] Konstantinos Sfikas, Ioannis Pratikakis, Anestis Koutsoudis, Michalis Savelonas, and Theoharis Theoharis. Partial matching of 3D cultural heritage objects using panoramic views. *Multimedia Tools and Applications*, 75(7):3693–3707, April 2016.

[125] Takahiko Furuya and Ryutarou Ohbuchi. Dense sampling and fast encoding for 3D model retrieval using bag-of-visual features. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '09, pages 1–8, New York, NY, USA, July 2009. Association for Computing Machinery.

[126] A. Godil, H. Dutagaci, B. Bustos, S. Choi, S. Dong, T. Furuya, H. Li, N. Link, A. Moriyama, R. Meruane, R. Ohbuchi, D. Paulus, T. Schreck, V. Seib, I. Sipiran, H. Yin, and C. Zhang. Range Scans based 3D Shape Retrieval. 2015. Accepted: 2015-04-27T11:03:43Z Publisher: The Eurographics Association.

[127] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[128] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. KAZE Features. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, Lecture Notes in Computer Science, pages 214–227, Berlin, Heidelberg, 2012. Springer.

[129] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, June 2010. ISSN: 1063-6919.

[130] Giuseppe Serra, Costantino Grana, Marco Manfredi, and Rita Cucchiara. GOLD: Gaussians of Local Descriptors for image representation. *Computer Vision and Image Understanding*, 134:22–32, 2015.

[131] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the Fisher Kernel

for Large-Scale Image Classification. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, Lecture Notes in Computer Science, pages 143–156, Berlin, Heidelberg, 2010. Springer.

[132] Bart Iver van Blokland and Theoharis Theoharis. *Microshapes: Efficient Querying of 3D Object Collections based on Local Shape*. The Eurographics Association, 2018. Accepted: 2018-04-14T18:28:38Z ISSN: 1997-0471.

[133] Bart Iver van Blokland, Theoharis Theoharis, and Anne C. Elster. Quasi Spin Images. *Norsk IKT-konferanse for forskning og utdanning*, August 2018.

[134] Bart Iver van Blokland and Theoharis Theoharis. Radial intersection count image: A clutter resistant 3D shape descriptor. *Computers & Graphics*, 91:118–128, October 2020.

[135] Bart Iver van Blokland and Theoharis Theoharis. An indexing scheme and descriptor for 3D object retrieval based on local shape querying. *Computers & Graphics*, 92:55–66, November 2020.

[136] Bart Iver van Blokland and Theoharis Theoharis. Partial 3D object retrieval using local binary QUICCI descriptors and dissimilarity tree indexing. *Computers & Graphics*, 100:32–42, November 2021.

[137] A.S. Mian, M. Bennamoun, and R. Owens. Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1584–1601, October 2006. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[138] Lilita Kiforenko, Bertram Drost, Federico Tombari, Norbert Krüger, and Anders Glent Buch. A performance evaluation of point pair features. *Computer Vision and Image Understanding*, 166:66–80, January 2018.

[139] Website of the 3d shape retrieval challenge (SHREC). `https://www.shrec.net/`, August 2021.

[140] Helin Dutagaci, Afzal A. Godil, A. Axenopoulos, P. Daras, T. Furuya, and R. Ohbuchi. SHREC 2009 - Shape Retrieval Contest of Partial 3D Models. March 2009. Last Modified: 2017-02-19T20:02-05:00.

[141] Andrea Giachetti, Silvia Biasotti, Elia Moscoso Thompson, Luigi Fraccarollo, Quang Nguyen, Hai-Dang Nguyen, Minh-Triet Tran, Gerasimos Arvanitis, Ioannis Romanelis, Vlasis Fotis, Konstantinos Moustakas, Claudio Tortorici, Naoufel Werghi, and Stefano Berretti. *SHREC 2020 Track: River Gravel Characterization*. The Eurographics Association, 2020. Accepted: 2020-09-03T09:50:27Z ISSN: 1997-0471.

[142] Babak Taati, Michel Bondy, Piotr Jasiobedzki, and Michael Greenspan. Variable Dimensional Local Shape Descriptors for Object Recognition in Range Data. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, October 2007. ISSN: 2380-7504.

[143] Thomas Sølund, Anders Glent Buch, Norbert Krüger, and Henrik Aanæs. A Large-Scale 3D Object Recognition Dataset. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 73–82, October 2016.

[144] I. Sipiran, R. Meruane, B. Bustos, T. Schreck, H. Johan, B. Li, and Y. Lu. SHREC'13 Track: Large-Scale Partial Shape Retrieval Using Simulated Range Images. *Eurographics 2013 Workshop on 3D Object Retrieval*, page 8 pages, 2013. Artwork Size: 8 pages ISBN: 9783905674446 Publisher: The Eurographics Association.

[145] Manolis Savva, Fisher Yu, Hao Su, Asako Kanezaki, Takahiko Furuya, Ryutarou Ohbuchi, Zhichao Zhou, Rui Yu, Song Bai, Xiang Bai, Masaki Aono, Atsushi Tatsuma, S. Thermos, A. Axenopoulos, G. Th. Papadopoulos, P. Daras, Xiao Deng, Zhouhui Lian, Bo Li, Henry Johan, Yijuan Lu, and Sanjeev Mk. Large-scale 3d shape retrieval from shapenet core55: Shrec'17 track. In *Proceedings of the Workshop on 3D Object Retrieval*, 3Dor '17, page 39–50, Goslar, DEU, 2017. Eurographics Association.

[146] Rongrong Lu, Feng Zhu, Yingming Hao, and Qingxiao Wu. Simple and efficient improvement of spin image for three-dimensional object recognition. *Optical Engineering*, 55(11):113102, November 2016. Publisher: International Society for Optics and Photonics.

[147] Anders G. Buch, Henrik G. Petersen, and Norbert Krüger. Local shape feature fusion for improved matching, pose estimation and 3D object recognition. *SpringerPlus*, 5(1):297, March 2016.

[148] Huan Zhao, Minjie Tang, and Han Ding. HoPPF: A novel local surface descriptor for 3D object recognition. *Pattern Recognition*, 103:107272, July 2020.

[149] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy. Geometrically stable sampling for the ICP algorithm. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 260–267, October 2003.

[150] Hui Chen and Bir Bhanu. Contour Matching for 3D Ear Recognition. In *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*, volume 1, pages 123–128, January 2005.

[151] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum*, 22(3):223–232, 2003.

[152] Antonio Adán, Pilar Merchán, and Santiago Salamanca. 3D scene retrieval and recognition with Depth Gradient Images. *Pattern Recognition Letters*, 32(9):1337–1353, July 2011.

[153] Leonardo Chang, Miguel Arias-Estrada, José Hernández-Palancar, and L. Enrique Sucar. Partial Shape Matching and Retrieval under Occlusion and Noise. In Eduardo Bayro-Corrochano and Edwin Hancock, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, Lecture Notes in Computer Science, pages 151–158, Cham, 2014. Springer International Publishing.

[154] Pu Jiantao, Liu Yi, Xin Guyu, Zha Hongbin, L. Weibin, and Y. Uehara. 3D model retrieval based on 2D slice similarity measurements. In *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004.*, pages 95–101, September 2004.

[155] I. O. Taybi, T. Gadi, and R. Alaoui. A New Partial 3D Object Indexing and Retrieval Approach Combining 2D slices and Apriori Algorithm. *Scientific Visualization*, 12(2), 2020.

[156] Martin Bokeloh, Michael Wand, Vladlen Koltun, and Hans-Peter Seidel. Pattern-aware shape deformation using sliding dockers. *ACM Transactions on Graphics*, 30(6):1–10, December 2011.

[157] Martin Bokeloh, Michael Wand, and Hans-Peter Seidel. A connection between partial symmetry and inverse procedural modeling. *ACM Transactions on Graphics*, 29(4):104:1–104:10, July 2010.

[158] Manolis Savva, Fisher Yu, Hao Su, Asako Kanezaki, Takahiko Furuya, Ryutarou Ohbuchi, Zhichao Zhou, Rui Yu, Song Bai, Xiang Bai, Masaki Aono, Atsushi Tatsuma, S. Thermos, A. Axenopoulos, G. Th Papadopoulos, P. Daras, Xiao Deng, Zhouhui Lian, Bo Li, Henry Johan, Yijuan Lu, and Sanjeev Mk. *Large-Scale 3D Shape Retrieval from ShapeNet Core55*. The Eurographics Association, 2017. Accepted: 2017-04-22T17:17:41Z ISSN: 1997-0471.

Part II

Selected Publications

Chapter 6

Paper A - Quasi Spin Images

Authors

Bart Iver van Blokland, Theoharis Theoharis, and Anne C. Elster.

Published in

Proceedings of the Norsk IKT-konferanse for forskning og utdanning (NIKT), 2018

Quasi Spin Images

Bart Iver van Blokland[1], Theoharis Theoharis[1], Anne C. Elster[1]
1) Norwegian University of Science and Technology, Norway

Abstract

The increasing adoption of 3D capturing equipment, now also found in mobile devices, means that 3D content is increasingly prevalent. Common operations on such data, including 3D object recognition and retrieval, are based on the measurement of similarity between 3D objects. A common way to measure object similarity is through local shape descriptors, which aim to do part-to-part matching by describing portions of an object's shape. The Spin Image is one of the local descriptors most suitable for use in scenes with high degrees of clutter and occlusion but its practical use has been hampered by high computational demands. The rise in processing power of the GPU represents an opportunity to significantly improve the generation and comparison performance of descriptors, such as the Spin Image, thereby increasing the practical applicability of methods making use of it. In this paper we introduce a GPU-based Quasi Spin Image (QSI) algorithm, a variation of the original Spin Image, and show that a speedup of an order of magnitude relative to a reference CPU implementation can be achieved in terms of the image generation rate. In addition, the QSI is noise free, can be computed consistently, and a preliminary evaluation shows it correlates well relative to the original Spin Image.

6.1 Introduction

Local shape descriptors are essential for measuring the similarity of 3D objects, and are at the heart of operations such as 3D object retrieval and recognition. These operations are essential as 3D object collections grow in size. A classic descriptor, the spin image (SI), is advantageous for many object classes in terms of accuracy and has thus been employed in many applications. In fact the SI has been considered to be the de facto benchmark for the evaluation of local surface features [14] [33], and has been listed among the descriptors most robust to clutter, varying mesh resolution, and clutter [13]. Unfortunately, the SI is hampered by high computational demands, thus restricting its applicability.

Graphics Processing Units (GPUs) have in recent years seen increased use in many applications, and have shifted from processors aimed primarily at accelerating rendering procedures to extremely high-throughput co-processors (also referred to as General Purpose GPUs, or GPGPUs). The highly parallel and throughput oriented architecture of GPUs has

allowed for a number of techniques to be significantly accelerated, increasing their utility. For instance, deep learning has seen performance increases by a factor of 50 or more [20]. The fields of game physics, computational biophysics [15] and medical image processing [17] [21] are also good examples.

Effective utilisation of GPU compute resources in part requires designing an algorithm with the hardware in mind. For instance, groups of threads should ensure their memory requests exhibit high spatial locality to minimise wasted memory bandwidth, as well as minimise thread divergence.

One observation which can be made regarding local shape descriptors is that their implementations are consistently written for the CPU. Moreover, descriptors produced by local descriptor generation algorithms are generally independent of one another, and have similar or identical generation processes [13] [10]. The characteristics of their computation are well aligned with the workloads GPUs have been designed for, i.e. similar operations over many data items that require high throughput.

Meanwhile, previous work within the fields of symmetry detection and 3D object retrieval has primarily focused on improving the accuracy of local shape descriptors for 3D object similarity evaluation.

This paper shows the potential of utilising the GPU for generating local mesh descriptors. We use the spin image as an example to show that designing local shape descriptors with the GPU in mind can greatly benefit execution times. This in turn allows for greater practical use of these methods. To this end, we propose the Quasi Spin Image (QSI) descriptor, a descriptor similar to the SI, which has the following advantages:

- The QSI exhibits properties favourable for execution on the GPU.
- The QSI is noise-free.
- The QSI consistently produces the same image, given the same input model and parameters.

The contributions of this paper include:

- The novel QSI local shape descriptor.
- A GPU implementation of the QSI with good memory, memory bandwidth, and performance characteristics.
- A GPU implementation of the original SI descriptor, along with a detailed execution performance evaluation against the QSI on a recent benchmark.

The remainder of this paper is organised as follows: In Section 6.2 we outline the original spin image descriptor and some other related work. In Section 6.3 we introduce the novel Quasi Spin Image descriptor. We finally evaluate the proposed method in Section 6.4.

6.2 Background

The spin image is a local histogram descriptor initially proposed by Johnson *et al.* [18]. It is either created from a point cloud or from a uniformly sampled triangle mesh. Its generation conceptually involves rotating a square plane around a given vertex (referred

to as the spin vertex) directed along a given normal vector (referred to as the spin normal) for one revolution. The plane is divided into an equal number of bins along the horizontal and vertical axes, and its physical size is referred to as the support radius. When rotated, each bin forms a torus-like volume as shown in Figure 6.1. The spin image is conceptually constructed by measuring the total area of the input mesh intersecting each of these torus-like volumes. The spin vertex and spin normal combined define a line, referred to as the central axis.

In order to generate a single image, five parameters must be given: the spin vertex (which usually lies on the surface of the sample model), the spin normal (usually the surface normal at the location of the spin vertex), the image width in bins (pixels), the support radius S_r, and the support angle.

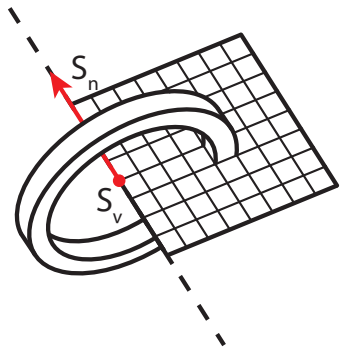


Figure 6.1: Spin image computation: the volume captured by rotating a spin image bin around the central axis. S_v and S_n are the spin vertex and the spin normal respectively. The dashed line represents the central axis.

Accurately calculating the area of a mesh intersecting the aforementioned torus-like volume (shown in Figure 6.1) for each spin image bin is complex, and thus time consuming. The authors of the original paper instead opted for using uniformly distributed surface point samples. The idea is that a linear increase in the area intersecting the torus-like volume implies a linear increase in the number of point samples intersecting it, thus approximating the computation of area. This approximation can however cause a reduction in matching performance, as shown by Carmichael *et al.* [2].

While the spin image descriptor has been shown to perform well in scenes with significant quantities of clutter [13] [10] [18], and has been used successfully in a number of applications [5] [2] [16] [3], several alternate forms have been proposed to address some of its shortcomings or improve its matching performance.

Effective matching of spin images requires setting a support radius parameter [18]. Additionally, comparing spin image pairs requires computing the Pearson correlation coefficient, which can be computationally costly. Dinh *et al.* [7] addressed these issues by proposing a multiscale spin image. Their method generates downsampled spin images for faster matching or matching at different support radii, albeit not simultaneously.

Another method attempting to simplify the image comparison process, is the spin image signature method proposed by Assfalg *et al.* [1]. This method computes a signature from each spin image, thereby significantly reducing the computation time necessary to compare two images. However, the spin image generation time was not addressed.

Finally, Davis *et al.* and Gerlach *et al.* showed significant speedups can be achieved when using the GPU for comparing spin images, reporting speedups of one to two orders of magnitude compared to a CPU implementation [5] [11]. However, neither of these show implementations for generating spin images on the GPU.

6.3 Quasi Spin Images

6.3.1 Motivation

An important observation regarding spin images is that the content of an image is independent of any other image. This in turn means the generation of spin images can be run in parallel across a number of threads. Typically, a large number of images are generated for a particular mesh to increase the probability of finding a matching pair when comparing them. Moreover, the process of computing individual images is identical. For these reasons, the generation algorithm exhibits favourable characteristics for its implementation on GPUs.

Profiling our GPU implementation of the spin image algorithm showed the main bottleneck to be the large volume of memory transactions required to generate individual images. This quantity of transactions is mainly caused by two factors.

First, in order to obtain a representative image which accurately portrays the support volume of the spin image, a significant number of uniformly sampled surface points are needed to ensure that the produced images can be matched against other images.

Second, in the original SI generation method, point samples are divided over the four adjacent pixels using bilinear interpolation. On a GPU this causes four reads and four writes to memory per pixel update.

Therefore, limiting the number of memory transactions necessary per image is the primary issue which must be addressed for an effective implementation of the spin image generation algorithm.

CPUs, as opposed to GPUs, are capable of containing an entire spin image in L1 cache. Therefore the effects of the large quantities of transactions on performance are not as significant. The main reason for this is the good spatial locality of these requests relative to the size of the cache. In contrast, on a GPU, these transactions generally require explicit memory transactions, and accesses tend to be spread over a range of cache lines. This in turn causes both high pressure on the memory bus while simultaneously not fully utilising the available bandwidth.

Unfortunately, keeping an image in the GPU's shared memory and only committing it to memory upon its completion severely limits the number of blocks which can be active per GPU streaming multiprocessor, significantly reducing performance. Therefore, the only means to address these problems is to reduce the number of required memory transactions.

6.3.2 Definition

We thus propose the Quasi Spin Image descriptor (QSI). A QSI is formed by counting the number of *intersections* of the model geometry with circles defined by points (pixel centres) on the QSI plane and the central axis, as shown in Figure 6.2, as opposed to measuring *area* as done by the SI. This definition effectively yields a stack of layers, each containing a number of circles with linearly increasing radii. The number of intersections between each circle and the mesh surface can be represented in an image, as shown in Figure 6.3.

While the definition of the QSI deviates from the original SI, the images produced are

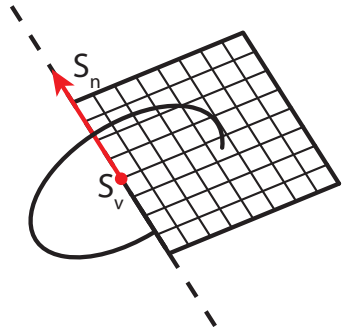


Figure 6.2: Quasi spin image computation: the circle captured by rotating a spin image bin around the central axis. S_v and S_n are the spin vertex and the spin normal respectively. The dashed line represents the central axis.

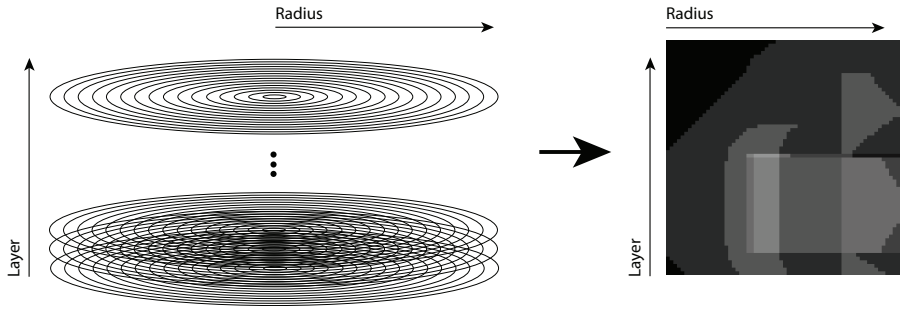


Figure 6.3: The correspondence between intersection circles and the produced QSI image.

similar visually, especially when compared to other descriptors. A visual comparison of QSI and SI is shown in Figure 6.6.

The main advantage of QSI relative to the SI is that it requires significantly fewer memory transactions. This is due to the fact that when generating an SI, projecting an additional surface on to the image on average requires each affected pixel to be updated more than once. This is mainly caused by projected point contributions being spread over neighbouring pixels using bilinear interpolation, as discussed previously. In contrast, the QSI only requires one additional update per rasterised pixel.

Additionally, since the number of intersections between a circle and a triangle mesh is an integer value, the values of individual QSI pixels can be computed consistently and are free of noise. This is in contrast to the SI method which uses a (uniformly sampled) point cloud as input instead, which is inherently noisy.

The QSI requires the same parameters as the SI, most important of which are the spin vertex S_v and spin normal S_n. The only difference is that since the QSI uses a triangle mesh as input directly, no uniform mesh sampling is needed. A sample count does not therefore need to be set. The separation between layers and the separation between radii of circles within layers is defined by a single constant value computed from the support radius and the image width.

6.4 Results

We evaluated the QSI and SI GPU implementations with two distinct experiments:

- Execution times of GPU implementations of QSI and SI generation algorithms compared to a reference CPU implementation.
- The similarity between the correlations computed over different SI and QSI.

Each of these experiments are described in detail below.

6.4.1 QSI / SI Generation Execution Times on GPU and CPU

The performance of the SI and QSI generation implementations were tested by applying them on the training models from the SHREC17 dataset [31], which includes over 35,000 models. Each algorithm was executed 10 times per model in the dataset, and the execution times were subsequently averaged.

For each implementation, only the time spent on computing the images themselves was measured. The time requirements for loading, partitioning, and uniform surface sampling (in case of the original SI method) were not included in the execution time measurements. These processing steps only represent a minor or negligible portion of the total execution time.

A reference single-threaded implementation, part of the command line tools from Point Cloud Library, was used as a CPU baseline to compare against [29]. To the best of our knowledge, this implementation is the fastest one currently available at the time of writing.

For both the SI and QSI method, one spin image was generated per vertex/surface normal present in the model. This is a means of keeping the number of generated images per model consistent between the two methods and is identical to the approach used in the original spin image paper by Johnson *et al.* [18].

A noteworthy issue is that the SHREC17 benchmark does not guarantee the models to be at similar scale, while at the same time different models may exhibit different features at different scales.

Both SI and QSI require a support radius parameter to be set. In the original paper by Johnson *et al.* this radius was calculated by assuming that the optimal bin size (the physical width of a spin image bin/SI pixel) was equivalent to the mesh resolution, arguing that most features present in the model would be at this scale. However, we do not consider this argument to be valid anymore given the wide variety in mesh resolution across models available today.

We instead choose the support radius in such a way that the image covers as much of the model as possible, while simultaneously ensuring high-level features within the model remain visible on the images themselves. To this effect we use a support radius that creates images based on the scale of the model, rather than using a constant support radius. To achieve this, we first compute the axis-aligned bounding box of the model. We subsequently determine the length of the side of a cube whose volume is equal to the volume of this bounding box. The length of the side p of this cube is used as the "support diameter", which is halved to produce the support radius. Equation 6.1 computes the support radius S_r, in a manner that satisfies the above requirements:

$$S_r = \frac{\sqrt[3]{BBox.x \cdot BBox.y \cdot BBox.z}}{2} \tag{6.1}$$

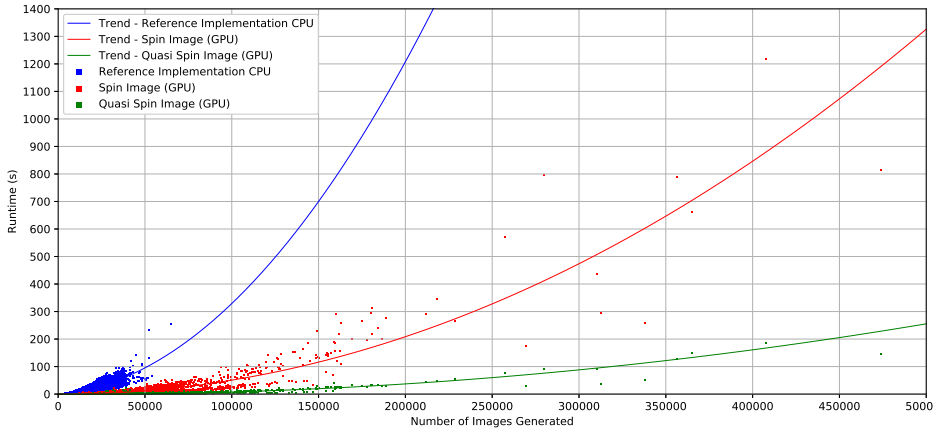


Figure 6.4: A comparison of the execution times of the proposed and reference implementations.

The original SI algorithm also requires a sample count parameter to be given. We set this number equal to three times the model's triangle count, uniformly distributed across the model. For many models, this was the minimum number of samples needed to produce an image of satisfactory quality.

The image size was set to 64x64 bins for the GPU tests (both SI and QSI). For stability reasons, the image size of the reference CPU implementation was set to 8x8 pixels. Here it should be noted that we could not detect a measurable performance difference between using 8x8 and 64x64 pixels, most likely because both image sizes are able to fit in the CPU's L1 cache.

The results of the SI and QSI generation times on the GPU are shown in Figure 6.4 along with the reference CPU implementation.

The average speedup of GPU QSI compared to GPU SI calculated over the models in the SHREC17 training set was 3.44.

The trend lines in Figure 6.4 show that when generating 200,000 images, the GPU QSI implementation outperforms the reference CPU one by approximately a factor of 35.

It's also worth noting that in *"A Comprehensive Review of Local Feature Descriptors"* by Quo *et al.* a performance evaluation is listed of an SI generation method implemented in MATLAB. For a model of 100,000 points, they measured an execution time of approximately 750ms in order to generate a single image. Our GPU implementation of the SI method, for models of an equivalent number of points, generates images at approximately 4700 images per second. This implies our GPU implementation is approximately 3500 times faster.

6.4.2 Correlation Between QSI and SI

It is interesting to investigate how QSI relates to the original SI. To this effect we used the models of the SHREC14 benchmark [14] [9] [1].

The devised experiment analyses one model in the benchmark at a time. For each vertex/normal pair in the model, an image is generated using each method, resulting in two

[1]The SHREC17 benchmark was not used here as a number of models in the set did not contain normals. Properly defined models are crucial to good matching performance.

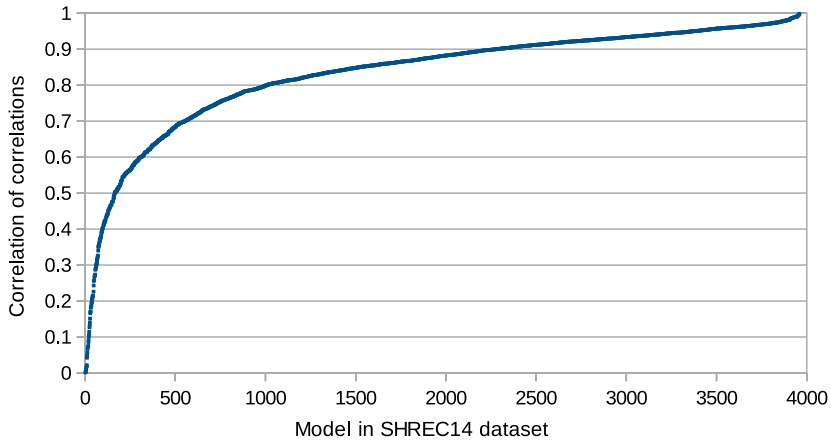
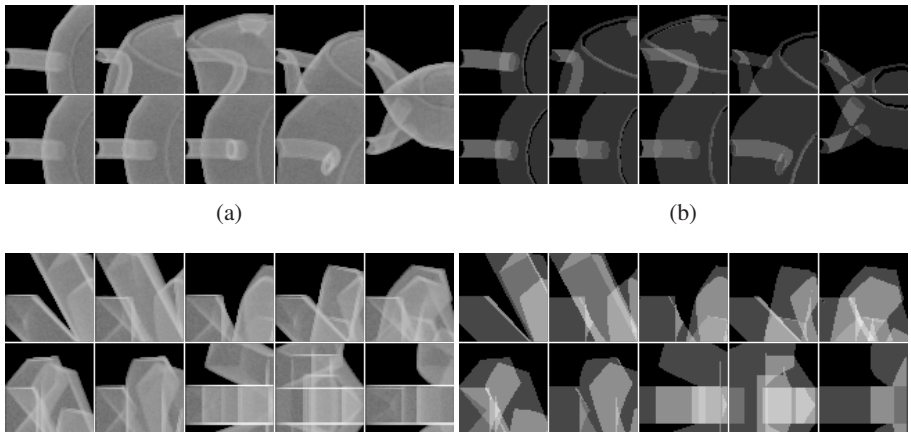


Figure 6.5: The correlation between the calculated correlations of SI and QSI for each model in the SHREC14 benchmark.

images per vertex/normal. The original SI method compared images by calculating the Pearson correlation coefficient. To estimate the similarity of QSI to SI, we thus estimate the Pearson correlation coefficient for the image pair generated by each method for the same vertex/normal. We compared the correlation values of image pairs generated by each method for each unique vertex/normal. We subsequently calculated the Pearson correlation coefficient over the resulting sequence of correlations. This yields a single correlation value representing the overall similarity of correlations calculated for each method for that model.

The resulting values for the entire benchmark are shown in Figure 6.5. The graph shows that 42.4% of the models have a correlation coefficient over 0.9, and 74.5% have 0.8 or higher.

Figure 6.6 shows a visual comparison of images generated by both methods. The images were generated from a Utah teapot and a Chrystalline structure. The following observations can be made from these Figures. First, since both methods generate images in cylindrical coordinate space, the produced shapes are similar. Second, an additive response can be observed in images generated by both methods, where specific parts of the model appear superimposed on the image. An example of this is the teapot handle visible on the top row of Figure 6.6b. Both the original SI and QSI methods exhibit this behaviour, due to the inherent greater area intersected by a bin and greater number of surface intersections encountered, respectively. Third, despite the fact that 1,000,000 point samples were used for both objects using the SI method, a level of noise is still visibly present. The images from our method are, apart from some single-pixel rounding errors, free of noise. Moreover, given the same input model and settings, the generated images of QSI are deterministic. Finally, the original SI method exhibits responses from surfaces orthogonal to the spin image plane, as in these cases greater numbers of point samples project to similar cylindrical coordinates. This is particularly visible on the images of the chrystalline structure (Figure 6.6c).



(a) (b)

(c) (d)

Figure 6.6: A visual comparison between the original SI (Figures 6.6a and 6.6c) and proposed QSI method (Figures 6.6b and 6.6d). All grayscale values have been logarithmically scaled for clarity.

6.5 Implementation Details

All testing took place on a system with an Intel Core i7-5820K processor and an NVidia Quadro P5000 graphics card. The GPU algorithms have been implemented using CUDA 9.0.

6.5.1 QSI Implementation

In order to cull geometry wherever possible as well as simplifying work division, we used a voxel space subdivision (we also used this in the SI implementation).

We cull flat triangles before rasterising them, as these are prone to cause rounding errors during the intersection test.

Finally, our implementation stores the values of individual bins in unsigned short variables, which proved more than adequate for the tested benchmark.

6.5.2 Model Scaling

When computing the QSI, one needs to convert from model space to QSI pixel space; this operation requires a division for all pixel updates in the case of SI and most pixel updates in the case of QSI, which is expensive. Instead, at pre-processing time the model is scaled so that one distance unit is equivalent to the physical size of a pixel/bin on the spin image plane.

6.6 Conclusion

It was shown that the workload characteristics of the generation of local shape descriptors, such as spin images, are suitable for GPU implementation (as they consist of similar operations that must be performed over multiple data items) and can thus offer significant

speedups over conventional CPU implementations. Moreover, targeted alterations to the local descriptor creation algorithm, such as QSI, can further improve the utilisation of the GPU hardware.

The quasi spin image local shape descriptor introduced in this paper is not only better capable of exploiting the available GPU resources, but also offers a number of other advantages. A reference GPU implementation of the original spin image algorithm is also given.

The GPU implementation for the generation of SI images was shown to outperform a reference CPU implementation by an order of magnitude, bringing the possibility of real-time applications within reach. Moreover, our GPU QSI implementation further outperforms the GPU SI implementation on average by over a factor of 3.

The source code for our implementations of SI and GSI are being made publicly available as part of this paper to serve the research community in benchmarking and further developing GPU based descriptors.

6.7 Future Work

While the generation efficiency, noise-free, and consistent generation properties of the QSI have been shown in this paper, a more complete numerical analysis of its matching capabilities using a recent benchmark should be done. As Figure 6.5 shows that correlations between similarity values produced when comparing images using each method are similar, and can therefore be expected to produce similar results in a matching performance experiment.

Moreover, since the largest deviations in pixel intensities between the QSI and SI occur on pixels intersecting surfaces near orthogonal to the SI plane, means performing an analysis including the support angle could yield closer matching performance between the two methods.

Additionally, since the QSI is noise free (as opposed to the spin image), it may be worth investigating other means for measuring image similarity, potentially improving its matching capabilities.

6.8 References

[1] Jrgen Assfalg, Marco Bertini, Alberto Del Bimbo, and Pietro Pala. Content-based retrieval of 3-d objects using spin image signatures. *IEEE Transactions on Multimedia*, 9(3):589–599, 2007.

[2] Jürgen Assfalg, Alberto Del Bimbo, and Pietro Pala. Spin images for retrieval of 3d objects by local and global similarity. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 906–909. IEEE, 2004.

[3] Ryan E Breighner, David R Holmes III, Shuai Leng, Kai-Nan An, Cynthia H McCollough, and Kristin D Zhao. Relative accuracy of spin-image-based registration of partial capitate bones in 4dct of the wrist. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 4(6):360–367, 2016.

[4] Owen Carmichael, Daniel Huber, and Martial Hebert. Large data sets and confusing

scenes in 3-d surface matching and recognition. In *3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on*, pages 358–367. IEEE, 1999.

[5] Cristina Conde, Roberto Cipolla, Licesio J Rodríguez-Aragón, Ángel Serrano, and Enrique Cabello. 3d facial feature location with spin images. In *MVA*, pages 418–421. Citeseer, 2005.

[6] Nolan Davis, Dennis Braunreiter, Cezario Tebcherani, and Masatoshi Tanida. 3d object matching on the gpu using spin-image surface matching. In *Advanced Signal Processing Algorithms, Architectures, and Implementations XVIII*, volume 7074, page 707408. International Society for Optics and Photonics, 2008.

[7] H Quynh Dinh and Steven Kropac. Multi-resolution spin-images. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 863–870. IEEE, 2006.

[8] Adam R Gerlach and Bruce K Walker. Accelerating robust 3d pose estimation utilizing a graphics processing unit. In *Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*, volume 7878, page 78780V. International Society for Optics and Photonics, 2011.

[9] Afzal A Godil and Chunyuan Li. Shrec'14 track: Large scale comprehensive 3d shape retrieval. In *Co-event of the 35rd Annual Conference of the European Association for Computer Graphics (Eurographics 2014)*, 2014.

[10] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3d object recognition in cluttered scenes with local surface features: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287, 2014.

[11] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89, 2016.

[12] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3d local surface description and object recognition. *International journal of computer vision*, 105(1):63–86, 2013.

[13] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[14] Bo Li, Yijuan Lu, Chunyuan Li, Afzal Godil, Tobias Schreck, Masaki Aono, Martin Burtscher, Qiang Chen, Nihad Karim Chowdhury, Bin Fang, et al. A comparison of 3d shape retrieval methods based on a large-scale benchmark supporting multimodal queries. *Computer Vision and Image Understanding*, 131:1–27, 2015.

[15] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, May 2008.

[16] Surya Prakash et al. False mapped feature removal in spin images based 3d ear recognition. In *Signal Processing and Integrated Networks (SPIN), 2016 3rd International Conference on*, pages 620–623. IEEE, 2016.

[17] Guillem Pratx and Lei Xing. Gpu computing in medical physics: A review. *Medical physics*, 38(5):2685–2697, 2011.

[18] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[19] Manolis Savva, Fisher Yu, Hao Su, Asako Kanezaki, Takahiko Furuya, Ryutarou Ohbuchi, Zhichao Zhou, Rui Yu, Song Bai, Xiang Bai, Masaki Aono, Atsushi Tatsuma, S. Thermos, A. Axenopoulos, G. Th. Papadopoulos, P. Daras, Xiao Deng, Lian Zhouhui, Bo Li, Henry Johan, Yijuan Lu, and Sanjeev Mk. Shrec'17 track large-scale 3d shape retrieval from shapenet core55. In *Proceedings of the Eurographics Workshop on 3D Object Retrieval*, 2017.

[20] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[21] Erik Smistad, Thomas L Falch, Mohammadmehdi Bozorgi, Anne C Elster, and Frank Lindseth. Medical image segmentation on gpus–a comprehensive review. *Medical image analysis*, 20(1):1–18, 2015.

[22] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010.

Chapter 7

Paper B - Microshapes: Efficient Querying of 3D Object Collections based on Local Shape

Authors

Bart Iver van Blokland and Theoharis Theoharis.

Published in

Proceedings of the Eurographics Workshop on 3D Object Retrieval, 2018

Microshapes: Efficient Querying of 3D Object Collections based on Local Shape

Bart Iver van Blokland[1], Theoharis Theoharis[1]
1) Norwegian University of Science and Technology, Norway

Abstract

Content-based querying of 3D object collections has the intrinsic difficulty of creating the query object and previous approaches have concentrated in producing global simplifications such as sketches. In contrast, in this paper, the concept of querying 3D object collections based on local shape is introduced. *Microshapes* are very promising in terms of generality and applicability and is based on a variation of the spin image descriptor. This descriptor uses intersection counts to determine the presence of boundaries in the support volume. These boundaries can be used to recognise local shape similarity. Queries based on this descriptor are general, easy to specify and robust to geometric clutter.

7.1 Introduction

A large number of objects, in particular man-made ones, inherently exhibit smaller shapes which together define the appearance of the whole object. For instance, shelving units are commonly constructed using rectangular planks, which in turn can be considered to contain straight corners as well as flat surfaces of various sizes.

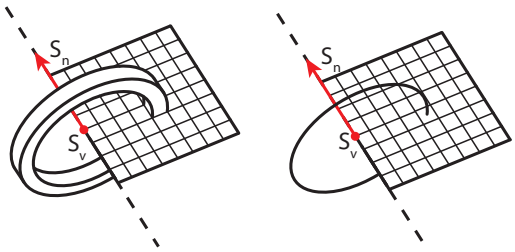
On a semantic level, it is possible to describe objects by combinations of such "Microshapes", examples of which include *circular*, *rounded corner*, *slight bend*, and *concave edge*.

Local shape descriptors which have been proposed to date primarily aim at creating a summary of their support volume [10] [13]. They exploit qualitative properties which are a *result* of microshapes. For instance, the curvature at a specific surface point on a model is the *result* of the shape being convex or concave at that location, rather than the cause.

In this paper, we present a novel method which is, amongst others, able to detect the occurrence of specific microshapes within an object, as defined by a 2D search query image which can be created intuitively and efficiently.

In addition to its matching capability, our method has some distinct advantages:

- Capable of matching occluded sections of a mesh
- Invariant to pose



(a) (b)

Figure 7.1: An illustration of the main difference between the original spin image, which measures the mesh area intersecting a torus-like volume generated for each pixel shown in 7.1a, and a quasi spin image, which measures the number of intersections made by a circle with the mesh surface for each pixel, as shown in 7.1b.

- Resistant to geometric clutter
- Easily compressible and efficiently comparable due to its binary nature

7.2 Related Work

Of previously proposed approaches, those in the Bag-of-Visual-Words (BoVW) shape retrieval [3] category can be considered to be closest to Microshape images. These methods commonly compute a vector of descriptors from a sample object, which is in turn used to locate similar points or surfaces in other models. Several distinct approaches have been proposed in this paradigm which were shown to have state-of-the-art performance. These include using Global Fisher vectors [10] and image features computed from panoramic views [12]. While our method can potentially be applied in the BoVW paradigm, we consider Microshape query construction sufficiently intuitive such that queries can be formulated and used for querying directly. Using a query model is therefore not a necessity. Moreover, Microshapes search for curves or lines, where existing methods have focused on matching surface patches.

In terms of using curves for querying objects, sketch-based methods could to some extent be considered to use a similar approach to the one presented in this paper. A wide variety of such methods have previously been proposed, whose general goal is to match a user sketch of a desired object to objects in a database. This is commonly done by generating views of each mesh in the database from different angles, and comparing the shape of outlines and edges to those drawn in the sketched query through various means [13] [15] [5]. However, these methods match entire sketches against entire models, and as such do not allow smaller shapes to be located.

The spin image, initially proposed by Johnson *et al.* [18], is a descriptor generated by rotating a square plane divided into pixels around a central axis for one revolution, and subsequently measuring the area of the mesh intersecting with the torus-like volume generated by each pixel (as shown in Figure 7.1a). In the author's implementation, the area computation is approximated by accumulating uniformly sampled points instead. The descriptor has been shown to be noise resistant, and perform well in cluttered scenes. Moreover, due to the spin image's use of a cylindrical coordinate system centred around the

spin vertex, it is pose invariant. This property is inherited by our method.

A number of methods have been derived from the original spin image in order to address various aspects of its weaknesses. Such methods include multi-resolution spin images, proposed by Dinh *et al*. [7], which aimed to address their lack of scale invariance, and computing signatures from spin images for faster matching, as proposed by Assfalg *et al*. [1]. Additionally, a version of the spin image which does not use point samples to approximate area intersected per pixel was described by Carmichael *et al*. [2].

A more recent method derived from the spin image, which bears some similarities to the one presented in this paper, is the Spin Contour proposed by Liuang *et al*. [9]. The spin contour is generated by computing a spin image, and subsequently locating the contour around all nonzero pixels on that image, which can be used for matching. However, the method only looks at the extreme outlines of a shape in cylindrical coordinate space, and, as we show in this paper, this discards curves present within the object which can potentially be used for matching. Moreover, the parts of a model which could be considered similar are not guaranteed to be part of the spin outline.

7.3 Background: Quasi Spin Images

The Microshape Image (MSI), introduced in this paper as an efficient query tool (see next Section), is a derivation of the Quasi Spin Image (QSI) descriptor, which was proposed as an efficient GPU-based alternative to Spin Images in [14]. We briefly describe the QSI below.

The QSI is constructed around a 3D point (referred to as the spin vertex), and a surface normal at this point (referred to as the spin normal). The spin vertex and spin normal combined describe a line, referred to as the central axis. Computing a QSI involves placing square plane (referred to as the spin plane) divided into pixels along the central axis, rotating it for one revolution, and in the process counting the number of intersections between each pixel centre and the mesh surface. A visualisation of this is shown in Figure 7.1b.

These intersections are effectively performed in a cylindrical coordinate space defined by the spin vertex and the central axis. Let α be a variable denoting the distance to the central axis and β denote distance along the central axis (the spin normal) from the origin (the spin vertex). Imagine a sequence of planes P placed perpendicular to the central axis at equal increments of β. Then, for each such plane, imagine concentric circles centred at the central axis with radii corresponding to linearly increasing values of α. Each plane corresponds to a row of the QSI and each circle corresponds to a pixel within that row. The number of intersections of such a circle with the mesh surface gives the value of the corresponding QSI pixel. One aspect worth observing here is that the intersections of one of the above planes and the mesh surface produces two-dimensional curves.

Using intersection counts yields images which can both be computed consistently and are inherently free of noise. In contrast, the original spin image algorithm created a histogram of uniformly sampled point samples, and is therefore noisy. This is not acceptable in our case, as our method compares the values of neighbouring pixels (see next section). Moreover, because pixels in the original Spin Image represent sums of areas, distinguishing internal borders of objects is not possible. The properties exploited by our proposed descriptor therefore only exist in QSI images.

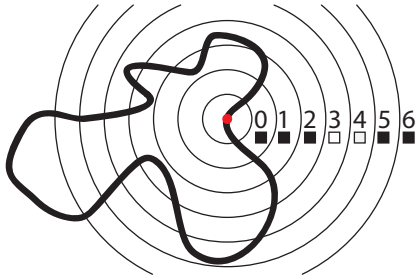


Figure 7.2: The generation of a single row in the MSI image. The portion of a shape intersecting with a plane described by a point along the central axis and the spin normal is shown. Circles with linearly increasing radii are intersected with the mesh. The computed values of the corresponding row of pixels in the MSI image are shown below each row index.

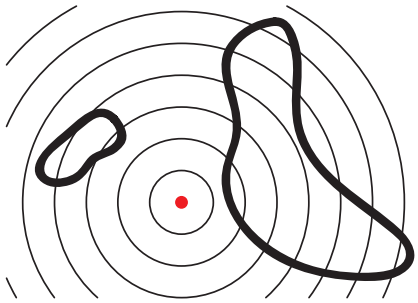


Figure 7.3: The object on the left causes a decrease in the number of encountered intersections with increasing circle radii, irrespective of the presence of the object on the right.

7.4 Microshape Images

The use of the MSI for querying 3D object collections based on local shape, was inspired by a specific observation in the behaviour of the QSI, which exploits its noise-free and consistency properties. Note that the intersection of the object mesh with the plane P used when computing a QSI, produces planar curves, as shown in Figure 7.2. The key observation which the MSI image exploits, is that the number of intersections with these curves, and thus the mesh surface, encountered by circles at linearly increasing radii only decreases *whenever a specific section of the mesh is no longer encountered.*

Since the MSI contains only the changes in the intersection counts of the QSI, it represents local shape more directly and is therefore more suitable as a local shape descriptor.

A reduction in intersection count can be observed in the vast majority of cases, irrespective of the presence of clutter, as shown in Figure 7.3. The only exception is whenever from one radius to the next, a section of the mesh which is no longer encountered, is replaced by another. However, this requires some rather specific circumstances and was not found to be a significant problem in practice. We therefore consider the MSI image to be resistant, albeit not immune, to clutter.

Moreover, the radii at which intersection counts decrease tend to be relatively similar across neighbouring rows of MSI pixels, describing shapes which are present in the object. An example of this is shown in Figure 7.4.

A database of microshape images corresponding to each vertex of a collection of 3D objects is queried by computing their distances from a constructed query MSI. We will refer to the query MSI as the "needle image" and to a database MSI as a "haystack image". Algorithm 1 describes the Offline and Online parts of the proposed query method.

Algorithm 1: Microshapes: offline and online algorithms

Offline

for *every 3D object O in haystack* **do**

  for *every vertex v in O* **do**

    - Compute QSI for v on GPU (Section 7.3)
    - Compute microshape image m from QSI (Section 7.4.1)
    - Store m as vertex v data

Online

- Design needle (query) microshape image q
- Compute distance function between q and the microshape image of every vertex of every object in the collection (Section 7.4.2)
- Produce ranked retrieval list of 3D objects based on distances of vertices from q

We have identified two limitations of our method. First, as MSI inherits some of the properties of QSI, the MSI are not scale invariant. However, the distance function is lenient and allows a certain degree of scale variations. Second, since MSI are discrete images and thus susceptible to aliasing effects, sampling detailed geometry which exhibit rapidly varying intersection counts (high frequencies), details can potentially be missed. But it should be noted that since MSI are bit vectors, they require 16 times less storage space compared to QSI, one can afford to generate them at a higher spatial resolution.

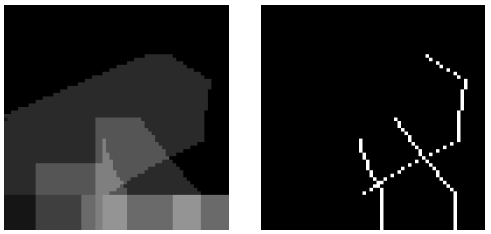
7.4.1 Microshape Computation

Computing a microshape image (MSI) consists of two steps. First, for a given vertex (spin vertex) and its associated normal (spin normal), a QSI is generated. Then, the QSI is converted into a binary MSI as follows. The value of each pixel in the QSI (number of intersections) is compared to its right neighbour. If the number of intersections in the right pixel is smaller than those in the left pixel, the left pixel is set to "active" (1), and "inactive" (0) otherwise.

A sample QSI and its corresponding MSI is shown in Figure 7.4.

7.4.2 Microshape Distance Function

The main idea behind our distance function is to evaluate to which degree the pattern in needle MSI is contained in a haystack MSI. We have identified four properties a distance function should exhibit to achieve our goal, and our implementation satisfies



(a) (b)

Figure 7.4: A sample quasi spin image (QSI) (7.4a) with its corresponding microshape image (MSI) (7.4b).

these requirements.

First, if active pixels in the haystack MSI are close to active pixels in the needle MSI then a low distance value must be produced. Second, the distance function should allow for minor differences in active pixels to occur; this is to allow for rounding errors and minute differences in shape. Third, in order to avoid taking geometric clutter into account, we should *only* be concerned with the haystack MSI patterns that correspond to active patterns in the needle MSI. Fourth, we should avoid matches resulting from haystack MSI geometric clutter close to the needle MSI pattern.

The above requirements resulted in the following distance algorithm, which works on a row by row basis. It is worth mentioning here that because MSIs are binary, our implementation stores and processes them as bit vectors.

For each row, we look at the location of the active bits in the needle MSI, and for each of these bits locate the closest active bit in the same row of the haystack MSI; this covers the first and second requirements. To satisfy the third requirement, we seek haystack MSI matches in a width of 2 pixels from the active needle pixels; unmatched haystack active pixels are given the maximum penalty of 2, since they have not been matched within the 2-pixel band. To address the fourth requirement, we calculate the hamming distance between needle and haystack MSIs within the 2-pixel band around each active needle pixel and add this to the distance measure of the row.

Our implementation uses 64x64 images, which means an entire row of an MSI can be stored in a single unsigned 64-bit integer. Our comparison method thus uses boolean bitwise operations, which allows it to be implemented efficiently.

The distance algorithm can be implemented efficiently by taking the bit string representing a needle image row and iteratively filtering away bits for which corresponding bits are located in the haystack row at increasing distances (up to 2 pixels in each direction). This can be done through a series of bit shifts, boolean operators, and the population count instruction (which counts the number of set bits in a bit string).

7.5 Experiments

We evaluated the potential of our method by designing query MSI images to retrieve objects from the SHREC17 training dataset [31] based on the local shape characteristics described by the MSI.

We generated an MSI for each vertex/normal present in each model, and created a

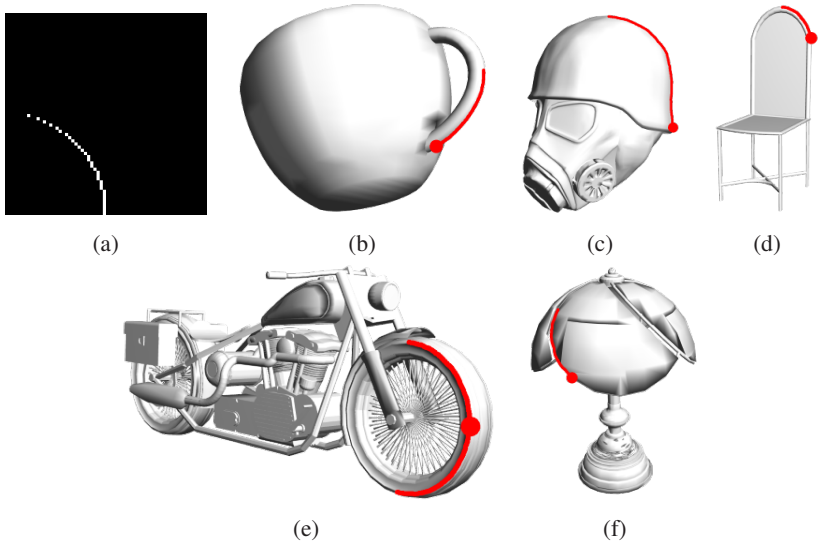


Figure 7.5: Search results produced by a query for a perfect quarter circle, as shown in 7.5a. The meshes shown in 7.5b to 7.5d are example objects from the top ranks of the retrieval list.

database of the produced MSIs. The size of the spin plane was set to be half of the side of a cube whose volume is equivalent to the axis-aligned bounding box of the input model. This implies that MSI are generated at a scale close to that of the model. The resolution of the MSI was set to 64x64 pixels.

For each query MSI, we computed a distance score against all vertex MSIs and sorted the corresponding objects by ascending score. In Figures 7.5 to 7.8, we have indicated the location of the detected vertex with a red dot, and any matched microshapes with red lines in example objects from the top ranks of the retrieval lists of particular interest. The top 20 unique objects are shown in Figures 7.9, 7.10, 7.11, and 7.12.

It should be noted that our search algorithm matches individual vertices. Some objects contain repeating shapes or patterns, and will therefore often appear multiple times in the retrieval list. Only the top appearance of each object has been indicated in the results list.

The generality of the approach should be observed here; simply giving the main characteristic of a local shape results in objects that possess it from various distinct object classes. This is a complex endeavour with global content-based queries. Figure 7.6a indicates that queries can be formulated which do not require the desired local shape to be originating from the sampled vertex.

It is particularly worth mentioning that the query MSIs were created in a simple image editor in the order of 1 minute each.

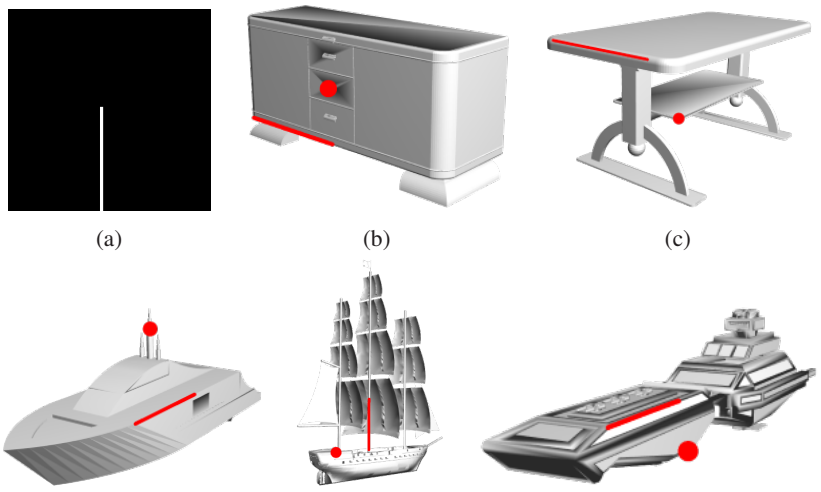


Figure 7.6: Search results returned by our algorithm based on the query MSI image shown in Figure 7.6a, representing a long straight edge some distance away from the sample point.

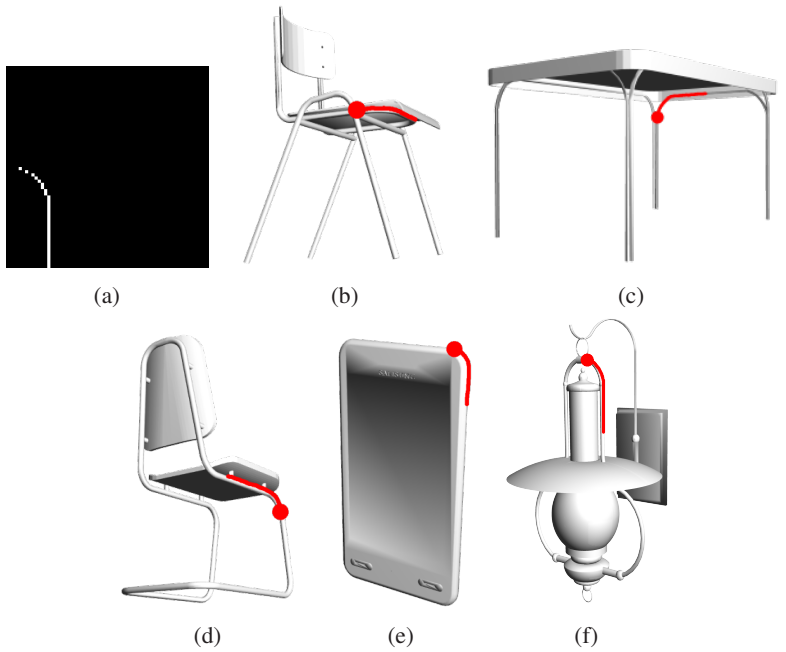


Figure 7.7: Search results returned by our algorithm based on the query MSI image shown in Figure 7.7a, representing a rounded corner followed by a straight edge.

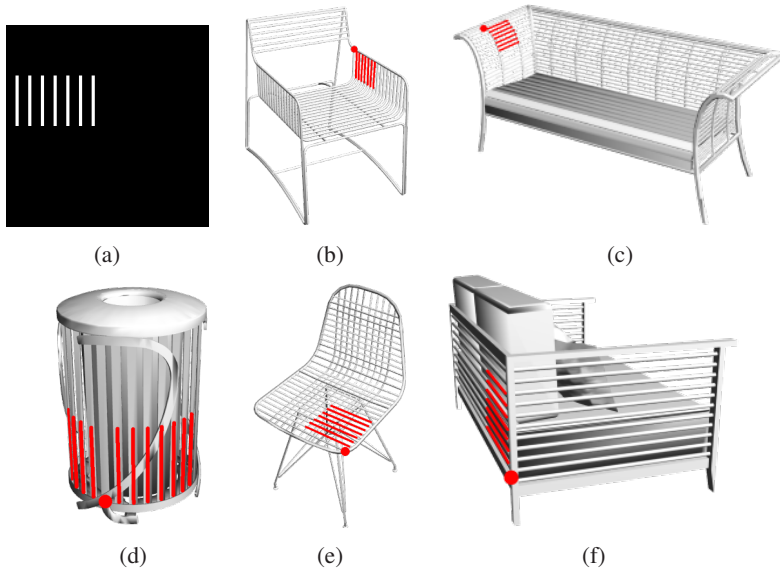


Figure 7.8: Search results returned by our algorithm based on the query MSI image shown in Figure 7.8a. The image represents a "grating"-like pattern.

7.6 Conclusion and Future Work

We have strong indications that querying based on microshapes is very general, widely applicable and quite simple in terms of the user interface required. Its local nature appears to simplify the problem of query formulation when searching 3D object collections based on content. A descriptor derived from a variant of the spin image that can be effectively computed on the GPU proved a highly suitable microshape descriptor.

While initial qualitative experiments indicate that the microshape method is very promising, a more thorough evaluation is required leading to quantitative results. For example, we can annotate the 3D objects of a certain collection with the microshapes (out of a finite set) that each contains and then perform microshape queries and count false positives and false negatives, thus leading to the standard retrieval metrics.

We believe the work presented in this paper opens the possibility for semantically querying 3D object collections, based on individual local features of a desired object. This is in contrast to conventional querying approaches, which concentrate on global appearance and thus do not offer the possibility of specifying detailed desired local shape characteristics.

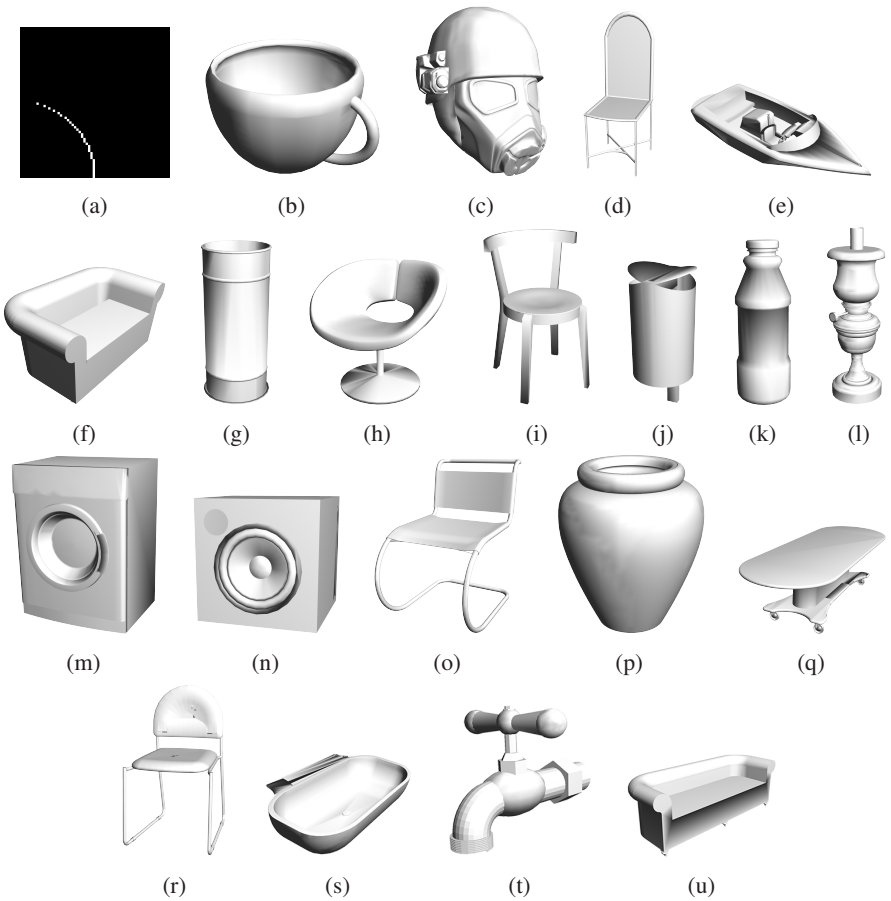


Figure 7.9: The top 20 unique models (shown in order) in the search results returned by our algorithm for the "quarter circle" query image shown in 7.9a.

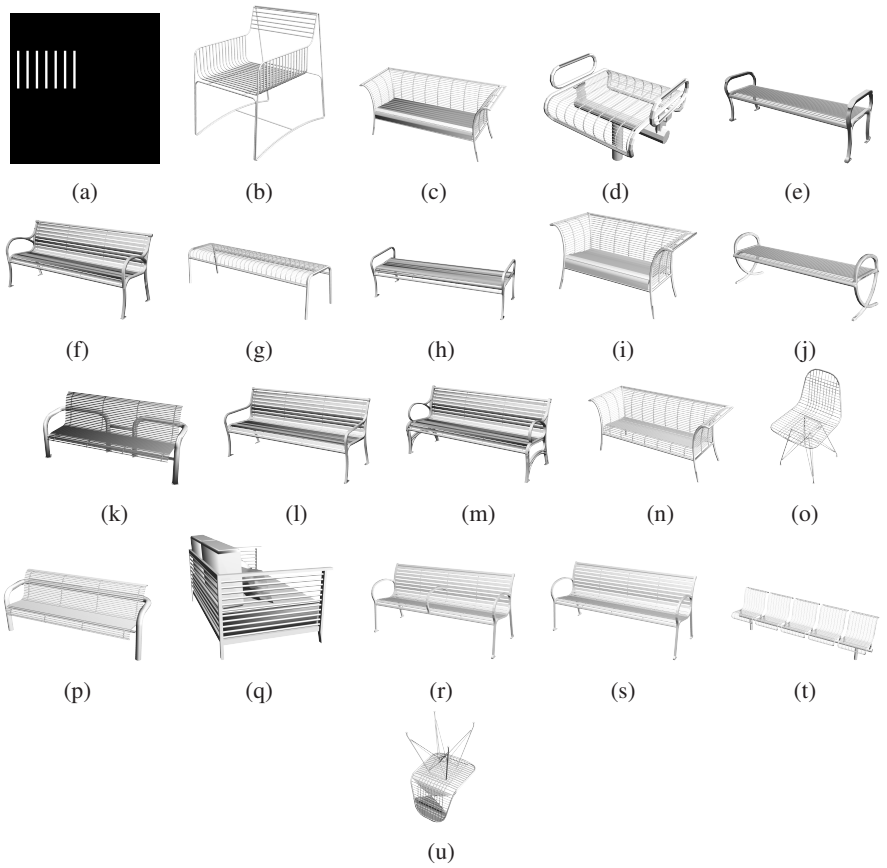


Figure 7.10: The top 20 unique models (shown in order) in the search results returned by our algorithm for the "grating" query image shown in 7.10a.

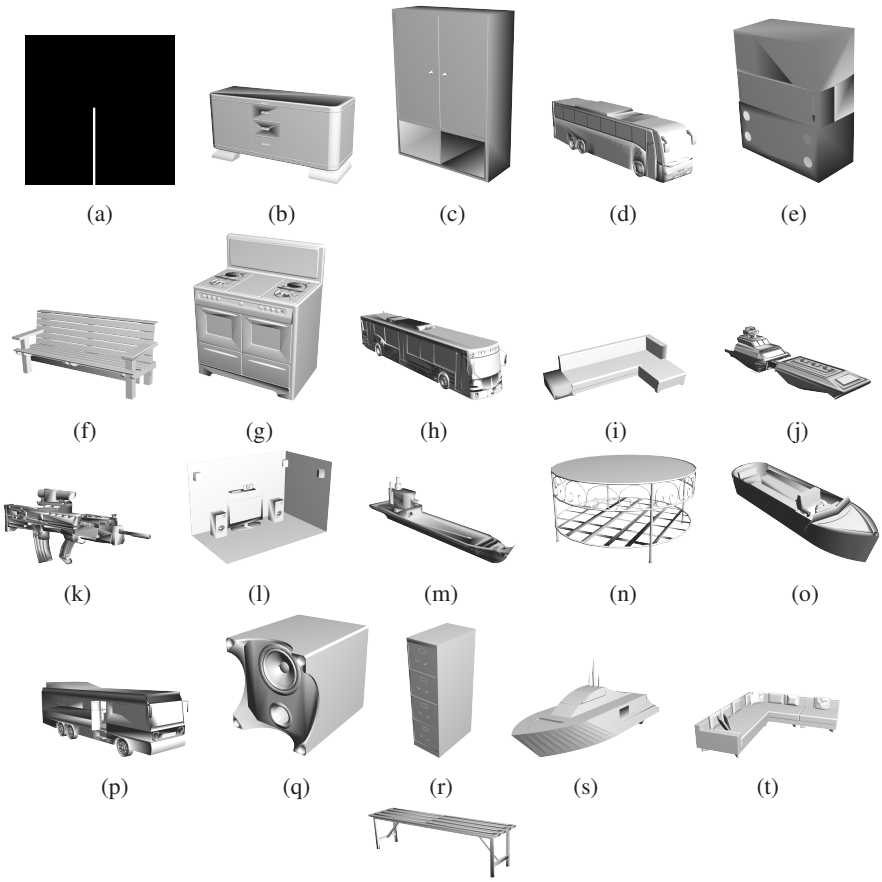


Figure 7.11: The top 20 unique models (shown in order) in the search results returned by our algorithm for the "long straight edge" query image shown in 7.11a.

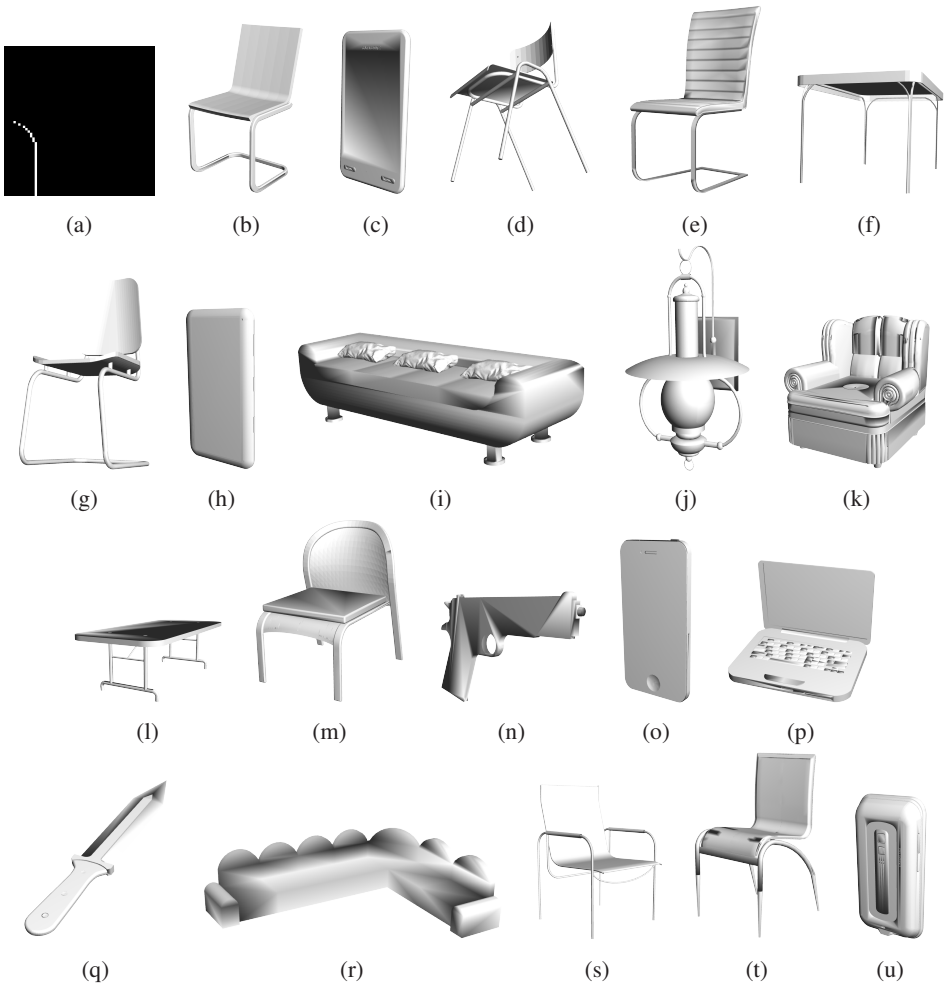


Figure 7.12: The top 20 unique models (shown in order) in the search results returned by our algorithm for the "rounded corner" query image shown in 7.12a.

7.7 References

[1] Jürgen Assfalg, Marco Bertini, Alberto Del Bimbo, and Pietro Pala. Content-based retrieval of 3-d objects using spin image signatures. *IEEE Transactions on Multimedia*, 9(3):589–599, 2007.

[2] O. Carmichael, D. Huber, and M. Hebert. Large data sets and confusing scenes in 3-d surface matching and recognition. In *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062)*, pages 358–367, 1999.

[3] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[4] H Quynh Dinh and Steven Kropac. Multi-resolution spin-images. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 863–870. IEEE, 2006.

[5] Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. Sketch-based shape retrieval. *ACM Trans. Graph.*, 31(4):31–1, 2012.

[6] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3d object recognition in cluttered scenes with local surface features: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287, 2014.

[7] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89, 2016.

[8] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[9] Luming Liang, Mingqiang Wei, Andrzej Szymczak, Wai-Man Pang, and Meng Wang. Spin contour. *IEEE Transactions on Multimedia*, 18(11):2282–2292, 2016.

[10] I Pratikakis, M Spagnuolo, T Theoharis, L Van Gool, and R Veltkamp. Partial 3d object retrieval combining local shape descriptors with global fisher vectors. 2015.

[11] Manolis Savva, Fisher Yu, Hao Su, Asako Kanezaki, Takahiko Furuya, Ryutarou Ohbuchi, Zhichao Zhou, Rui Yu, Song Bai, Xiang Bai, Masaki Aono, Atsushi Tatsuma, S. Thermos, A. Axenopoulos, G. Th. Papadopoulos, P. Daras, Xiao Deng, Lian Zhouhui, Bo Li, Henry Johan, Yijuan Lu, and Sanjeev. Mk. Shrec'17 track large-scale 3d shape retrieval from shapenet core55. In *Proceedings of the Eurographics Workshop on 3D Object Retrieval*, 2017.

[12] Konstantinos Sfikas, Theoharis Theoharis, and Ioannis Pratikakis. 3d object retrieval via range image queries in a bag-of-visual-words context. *The Visual Computer*, 29(12):1351–1361, 2013.

[13] Tianjia Shao, Weiwei Xu, Kangkang Yin, Jingdong Wang, Kun Zhou, and Baining Guo. Discriminative sketch-based 3d model retrieval via robust shape matching. In *Computer Graphics Forum*, volume 30. Wiley Online Library, 2011.

[14] Bart van Blokland, Theoharis Theoharis, and Anne Catherine Elster. Efficient quasi spin image generation on the gpu. (unpublished).

[15] Fang Wang, Le Kang, and Yi Li. Sketch-based 3d shape retrieval using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1875–1883. IEEE, 2015.

Chapter 8

Paper C - Radial Intersection Count Image: a Clutter Resistant 3D Shape Descriptor

Authors

Bart Iver van Blokland and Theoharis Theoharis

Published in

Computers & Graphics Volume 91, Pages 118-128, 2020, Elsevier

Awards

This paper has been awarded the *Graphics Replicability Stamp* by the Graphics Replicability Stamp Initiative (GRSI).

Radial Intersection Count Image: a Clutter Resistant 3D Shape Descriptor

Bart Iver van Blokland[1], Theoharis Theoharis[1]
1) Norwegian University of Science and Technology, Norway

Abstract

A novel shape descriptor for cluttered scenes is presented, the Radial Intersection Count Image (RICI), and is shown to significantly outperform the classic Spin Image (SI) and 3D Shape Context (3DSC) in both uncluttered and, more significantly, cluttered scenes. It is also faster to compute and compare. The clutter resistance of the RICI is mainly due to the design of a novel distance function, capable of disregarding clutter to a great extent. As opposed to the SI and 3DSC, which both count point samples, the RICI uses intersection counts with the mesh surface, and is therefore noise-free. For efficient RICI construction, novel algorithms of general interest were developed. These include an efficient circle-triangle intersection algorithm and an algorithm for projecting a point into SI-like (α, β) coordinates. The 'clutterbox experiment' is also introduced as a better way of evaluating descriptors' response to clutter. The SI, 3DSC, and RICI are evaluated in this framework and the advantage of the RICI is clearly demonstrated.

8.1 Introduction

Local shape descriptors have seen extensive use in a wide variety of applications where determining shape correspondences are beneficial or even required. Such applications include registration [24] [22] [32], shape segmentation [25] [15] [31], and retrieval [8] [4].

Many local 3D shape descriptor methods rely on the surfaces present in the volume around a point to compute the degree to which two points are similar. This also makes them susceptible to any unwanted geometry present in the neighbourhood, commonly referred to as *clutter*. For this reason, clutter has been named as a major factor degrading the performance of current descriptors [13].

The degree to which different descriptors are capable of resisting the negative effects of clutter varies. One classical method which has shown to be significantly resistant to clutter is the Spin Image [18] (SI). This descriptor is invariant under rigid transformations, and has been applied successfully for applications such as shape registration [16] and facial recognition [19].

In this paper, we present the Radial Intersection Count Image (RICI) combined with a novel distance function. The new descriptor shares the original concept of the Spin Image but is advantageous in terms of its generation speed and clutter resistance.

In order to show the effectiveness of the RICI, we propose a repeatable experiment aimed at quantifying the effects of clutter on the matching performance of 3D shape descriptors. The main advantage of this evaluation method is that it can be used with datasets of any size, and ensures scenes are cluttered with natural shapes.

In summary, the contributions of this paper are:

1. The novel RICI descriptor and an accompanying distance function, capable of resisting clutter.

2. Algorithms for efficient generation of RICI descriptors, also capable of accelerating SI construction.

3. The clutterbox experiment for quantifying the effects of clutter.

4. Evidence that the Support Angle filter proposed in the original SI paper does not necessarily improve matching performance.

5. Freely available GPU implementations for generating and comparing Spin Image, 3DSC, and RICI descriptors, as well as an implementation of the proposed clutterbox experiment.

8.2 Background and Related Work

Numerous local shape descriptors have been proposed to date [13]. The Spin Image has been the foundation for a number of methods, which attempt to improve its matching performance or other limitations. Clutter is a major challenge for object descriptors and few methods have addressed it.

8.2.1 Spin Images

The Spin Image [18], originally presented by Johnson et al., is a classic descriptor generated from an oriented point cloud (vertices with position and normal).

An SI is constructed around an oriented point, the position of which is in this paper referred to as the Spin Vertex S_v. The corresponding normal is referred to as the Spin Normal S_n. The combined oriented point describes a line, which is called the Central Axis.

Computing the descriptor involves placing a square plane whose left side is on the Central Axis, with the Spin Vertex at its vertical halfway point. This plane is subsequently subdivided into $(N_{bins} \times N_{bins})$ equivalently sized bins, and rotated for one revolution around the Central Axis. As the plane rotates, the number of point samples intersecting each bin is counted. The descriptor itself is a histogram of the resulting value of each bin, which can be visualised as an image.

In practice, the locations where point samples will intersect with the rotating square can be computed directly as two-dimensional cylindrical coordinates. Here the α coordinate refers to the distance from the point sample to the closest point on the Central Axis, and the β coordinate refers to the distance from this closest point to the Spin Vertex. The projection of a given point P is shown in Figure 8.1.

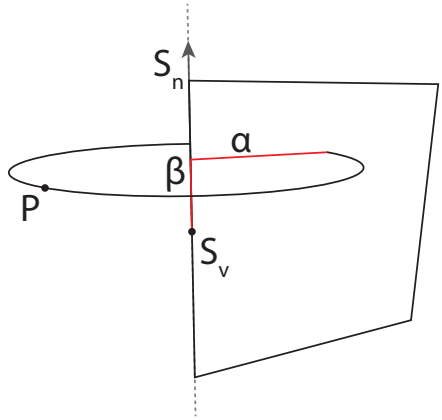


Figure 8.1: A visualisation of the α and β coordinates corresponding to a given point P, relative to the Spin Vertex S_v and Spin Normal S_n. The Central Axis; the line described by the Spin Vertex and Spin Normal is also shown.

The physical width and height of the square plane is the Support Radius of the descriptor. By rotating the plane around the Central Axis, a cylindrical volume is created, which represents the Support Volume of the descriptor. Additionally, point sample contributions are divided over nearby bins using bilinear interpolation to reduce the effects of aliasing.

Johnson et al. also describe a prefiltering step called the Support Angle, where a sample oriented point is not included in the computation of the descriptor if the angle between its normal vector and the Spin Normal exceeds a set threshold.

The descriptor's core idea is that a pair of points with identical surfaces surrounding them, and assuming both have been uniformly sampled, will have proportional quantities of projected points in similar locations. Images can thus be compared using statistical correlation.

8.2.2 Methods related to the SI

One of the major issues with the Spin Image is its volatility. Uniform sampling of triangle meshes as well as scans from 3D capture devices are inherently noisy. Carmichael et al. proposed a method to address this by computing the exact area of the support region intersecting each pixel [2].

Other methods aim to address specific limitations of the spin image. Assfalg et al. proposed the spin image signature aimed at simplifying the ease of image retrieval from a large database [1]. Dinh et al. aimed at addressing the issue of selecting bin sizes by creating a spin image variant with variable sized histogram bins [7], although their solution involves the manual setting of parameters.

An alternate spin image variant, proposed by Guo et al. used three spin images per vertex rather than a single one for better matching performance [14]. Accelerating spin image generation using a GPU was first proposed by Davis et al. [5] [11]. Alternate derivative methods include Spin Contours, proposed by Liang et al. [20] and colour spin images by Pasqualotto et al. [26].

8.2.3 The 3D Shape Context

The 3D Shape Context, proposed by Frome et al. [11], is a histogram descriptor constructed by accumulating points by their spherical coordinates and distance relative to an oriented reference point in a spherical support region. The support region is divided into J equally spaced spherical wedges, centred around the central axis described by the reference oriented point (similar to the SI). Each wedge is subsequently divided into K elevation divisions. The bin volumes are finally created by the intersection volume of each radial and elevation divisions with the volume bounded by two of L successive spheres with exponentially increasing radii.

The descriptor has a degree of freedom around the Central Axis, which the Authors solve by generating J different descriptors for each vertex, where each of the wedges has been offset by a multiple of the angle $\frac{2\pi}{J}$. However, due to its self-symmetry, this step is unnecessary for descriptors used for querying.

8.2.4 Other Clutter-Resistant Shape Matching Methods

Some methods which have been proposed to date, in addition to the Spin Image and 3DSC, have been shown to perform better in cluttered scenes than others [13] [23].

Mian et al. presented a method which creates a three-dimensional grid of voxels based on two randomly selected vertices, referred to as a Tensor [23]. Their results outperform the Spin Image, and show resistance to clutter being present in the scene.

The THRIFT descriptor, proposed by Flint et al. [10], uses an approach similar to the Scale-Invariant Feature Transform (SIFT) by Lowe et al [30]. The method aims to find distinctive points which can be detected reliably under a wide range of conditions. This is accomplished by computing a three-dimensional density map of the input point cloud, and selects interest points by locating local maxima of the Hessian matrix.

Local surface patches, proposed by Chen et al. [3], is a two-dimensional histogram descriptor generated from points in an oriented point cloud. Each descriptor accumulates points in a spherical support volume, by their shape index and the cosine of the angles between their normal vectors. The authors only test their method on range images, and do not expose the descriptor to significant levels of clutter themselves. However, experiments performed in the review by Guo et al. [13] suggest that this method performs well in cluttered scenes.

Unfortunately, the above works on clutter resistant descriptors used very small datasets for testing their methods (1 to 56 objects). Therefore, the provided results may be statistically biased, since the proposed descriptors were not subjected to a sufficiently wide range of possible surface features. The datasets used were also not made public, making it difficult to compare their results. In addition, some used very similar objects (such as cars), presumably for ease of creation, which is not representative of all forms of clutter that can be encountered in a real scene.

8.2.5 Learning Approaches

More recent shape matching methods have attempted to utilise Neural Networks. One of the major hurdles these methods need to overcome is the inherent irregularity present in 3D shape data, as opposed to more regular data such as images on which learning methods have been applied successfully.

To this end, many methods, such as the PPFNet proposed by Deng et al. [6], make use of existing descriptors or features in a pre-processing step to regularise the input to the neural network. PPFNet specifically uses point pair features, and was shown to outperform many current state-of-the-art handcrafted methods.

Another regularisation approach is the voxelisation of the input point cloud or mesh, which has amongst others been exploited in the 3DMatch method proposed by Zeng et al. [33], who successfully apply their proposed method on point cloud alignment and keypoint matching, outperforming both handcrafted and earlier learning methods.

While these learning methods show great promise, their applicability depends highly on the used dataset for training, and may require retraining for new environments. Moreover, current learning methods tend to be highly computationally expensive, which can limit their applicability to small datasets only [17].

8.3 Radial Intersection Count Images (RICI)

The novel RICI descriptor is now detailed, which shares some conceptual similarities with the original Spin Image, and has preliminarily been proposed as a quasi Spin Image [30].

8.3.1 RICI Generation

A RICI descriptor is a 2D histogram of integers. It is constructed around an oriented point, and has a Central Axis around which a square plane is conceptually rotated, similar to the Spin Image. The square plane is divided into ($N_{bins} \times N_{bins}$) bins, producing a histogram which can be visualised as a grayscale image.

The primary difference between the RICI and the SI is what is counted in each histogram bin. In Spin Images, projected point samples are accumulated to create an estimate of the surface area intersecting each bin or pixel as the square plane is rotated for a full revolution. In contrast, RICI bins count the number of intersections of circles with the surfaces of the scene and are thus integers.

The conceptual construction method, i.e. the relationship between the aforementioned intersection circles and the produced descriptor is visualised in Figure 8.3. Consider a set of circles that are centred at fixed distances from the Spin Vertex on the Central Axis and have a fixed number of radii. Each bin in the RICI image stores the number of intersections of the corresponding circle with the surfaces of the scene. RICI rows thus represent circles on the same plane, and RICI columns circles with equivalent radii.

The remainder of this section presents a method for efficiently computing RICI descriptors. The general idea is to iterate over each triangle in the scene, and determine the set of circles in cylindrical coordinates (see Figure 8.1) which will intersect with it. This implies a complexity of O(T), where T is the number of triangles in the scene, as in the worst case, the number of circles is fixed and equal to the resolution of a RICI image. The bins corresponding to these circles are incremented. Note that cylindrical projections will not preserve the linearity of a triangle's edges (as shown in Figure 8.2), thus not allowing the use of common rasterisation methods. Instead we exploit a circle-triangle intersection algorithm in order to determine the correct projections.

To summarise, a RICI image is generated by iterating over each triangle in the scene, and in turn each triangle is processed in 3 steps:

1. Project the triangle vertices into cylindrical coordinate space, as described in Section

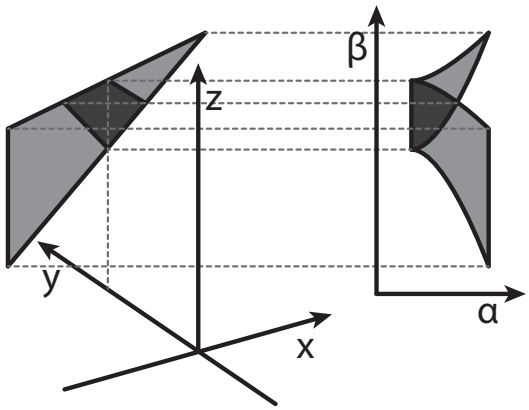


Figure 8.2: A triangle depicted alongside its projection in cylindrical coordinate space. The area in which circles centred and directed along the z-axis intersect the triangle twice is coloured in dark grey. Sizes may not be to scale.

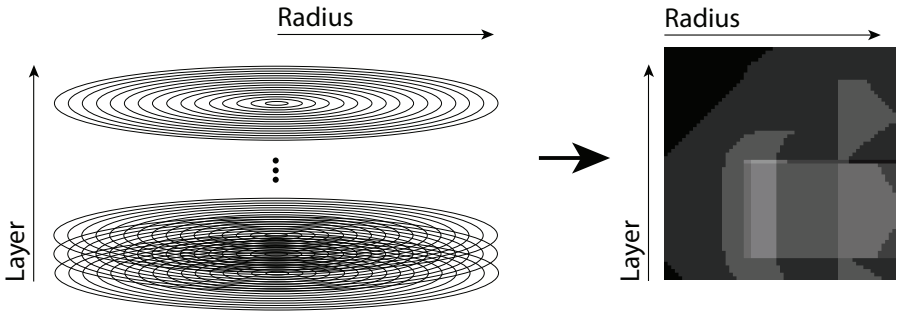


Figure 8.3: A visualisation of the construction of a RICI image.

8.3.1.1.

2. Using the circle-triangle intersection method outlined in Section 8.3.1.2, compute the range of α coordinates which will intersect with the triangle for each β coordinate in the triangle's β-extent.

3. Increment the histogram bins that correspond to these intersections.

8.3.1.1 Projecting Vertices into Cylindrical Coordinate Space

An efficient method for projecting points from Euclidean coordinates into cylindrical coordinates is presented. Apart from the RICI, this method can also be applied directly in the construction of SI descriptors.

The algorithm projects a point $P = (P_x, P_y, P_z)$ by computing two transformations. First, a translation that moves the Spin Vertex $S_v = (S_{vx}, S_{vy}, S_{vz})$ to the origin (Equation 8.2), and second, a rotation which aligns the Spin Normal $S_n = (S_{nx}, S_{ny}, S_{nz})$ with the z-axis. The projected point's α and β coordinates can be computed trivially afterwards.

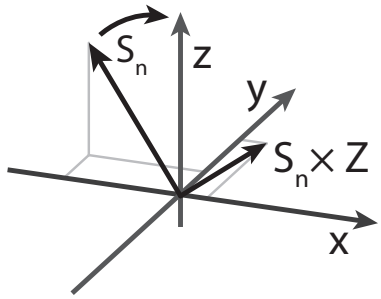
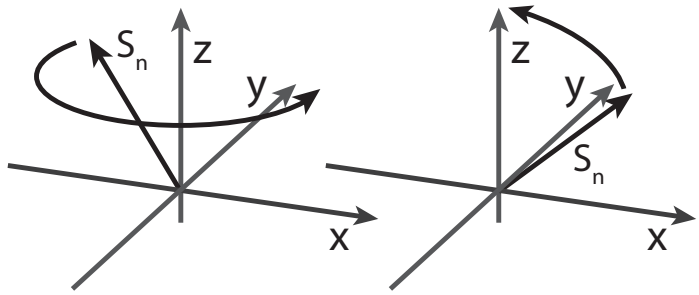


Figure 8.4: Direct approach for vector alignment. First, compute the vector product $S_n \times Z$ between the spin normal S_n and z-axis. Second, rotate S_n around $S_n \times Z$ to align it with the z-axis.



(a) Rotation 1: Align the spin normal with the XZ-plane by a rotation around the Z-axis (Equation 8.3)

(b) Rotation 2: Align the spin normal with the Z-axis by a rotation around the Y-axis (Equation 8.4)

Figure 8.5: Visual representation of the rotations that form our alignment method.

For the z-axis alignment transformation, a common technique for aligning two vectors consists of a vector product followed by a rotation (shown in Figure 8.4). While the vector product itself is inexpensive (due to one of the vectors being the z-axis) the subsequent alignment rotation requires a relatively expensive multiplication with a 3x3 matrix.

Our alignment method instead uses two rotations, exploiting the observation that only distance must be preserved for the α coordinate. We align the spin normal with the xz-plane using a rotation around the z-axis (see Figure 8.5a and Equation 8.3). We then align the transformed normal with the z-axis by a rotation around the y-axis (Figure 8.5b and Equation 8.4).

$$\begin{aligned}[N_{ax}, N_{ay}] &= Normalize[S_{nx}, S_{ny}] \\ [N_{bx}, N_{bz}] &= Normalize[S_{nx}, S_{nz}]\end{aligned} \tag{8.1}$$

$$\begin{aligned}P'_x &= P_x - S_{vx} \\ P'_y &= P_y - S_{vy} \\ P'_z &= P_z - S_{vz}\end{aligned} \tag{8.2}$$

$$\begin{aligned} P''_x &= N_{ax} \cdot P'_x + N_{ay} \cdot P'_y \\ P''_y &= -N_{ay} \cdot P'_x + N_{ax} \cdot P'_y \end{aligned} \tag{8.3}$$

$$\begin{aligned} T_x &= N_{bz} \cdot P''_x - N_{bx} \cdot P'_z \\ T_y &= P''_y \\ T_z &= N_{bx} \cdot P''_x + N_{bz} \cdot P'_z \end{aligned} \tag{8.4}$$

$$\begin{aligned} \alpha_i &= |(T_x, T_y)| \\ \beta_i &= T_z \end{aligned} \tag{8.5}$$

The coefficients of the rotation transformations N_a and N_b can be calculated inexpensively from components of the spin normal S_n, as shown in Equation 8.1. When both coefficients of either N_a or N_b are zero, that rotation step is unnecessary and an identity rotation is used instead. The key here is that, considering a two-dimensional coordinate system xy, the coordinates of a normalised vector represent the sine and cosine values of a rotation which aligns that vector with the x-axis. These normalised coordinates can therefore be used directly for this purpose.

It should be noted that since the rotation coefficients only depend on the spin normal, they are constant for the entire spin image. Therefore they only need to be computed once per image, essentially taking this computation out of the inner loop. This is the primary reason for the method's efficiency compared to previous work.

8.3.1.2 Circle-Triangle Intersection

A circle-triangle intersection test can result in four outcomes; no intersection, one intersection, two intersections, or infinite intersections. However, due to floating point rounding errors, handling the latter, while possible, is not feasible in practice and is thus not addressed by the proposed algorithm.

Our algorithm starts off with the triangle vertices in cylindrical coordinate space. For a given β coordinate, it determines the range of α coordinates which result in a single or double intersection. This information is subsequently used to "rasterise" a row of pixels for the triangle in the RICI descriptor.

The method operates in three distinct stages. First, the triangle is intersected with the plane π of the circle, which is parallel to the xy plane, as shown in Figure 8.6. Next, the triangle vertices are rotated around the z-axis in order to further simplify subsequent computations. Finally, the ranges of circle radii in which respectively single and double intersections occur, are calculated.

Prior to detailing these stages individually, we will outline the geometric background used in the intersection test calculations.

Figure 8.6 shows a given β coordinate. The triangle being tested is defined by its transformed vertices T_0, T_1, and T_2, using the previously described alignment transformation. Here all points with equal β coordinates lie on the plane π.

Where the triangle intersects the plane, it forms an intersection line segment E_0E_1, which defines a line L. The range of α coordinates either intersecting the triangle once or twice can

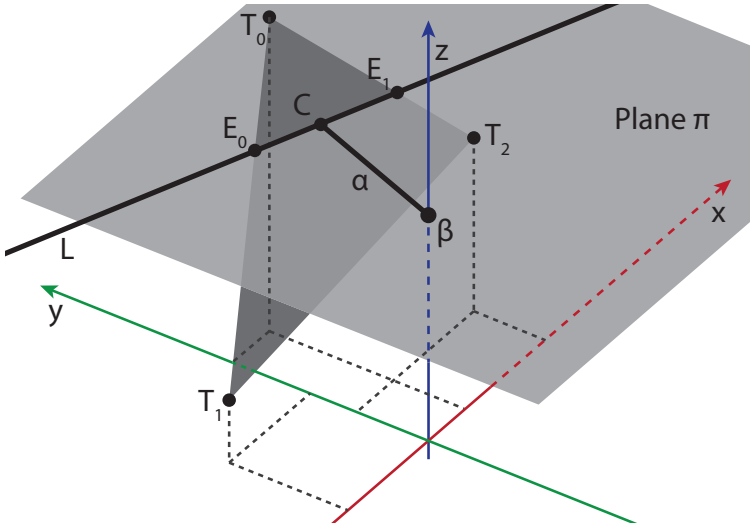


Figure 8.6: A triangle defined by the vertices T_0, T_1, and T_2 intersecting with the horizontal plane through an arbitrary coordinate β on the z axis.

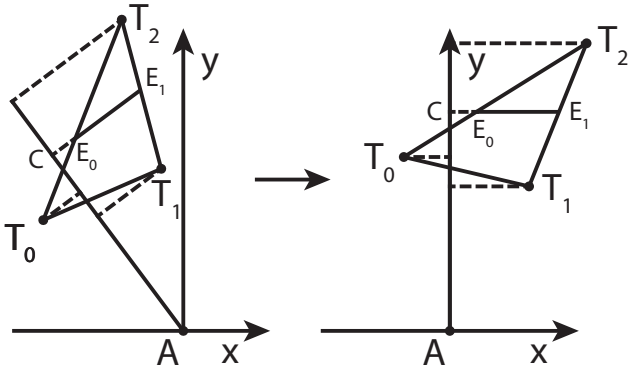


Figure 8.7: Aligning an $E_0\vec{E}_1$ vector with the x-axis. Any value of C can be chosen for which an $E_0\vec{E}_1$ vector exists for this purpose. A sample $E_0\vec{E}_1$ vector has been indicated in the Figure. Point A represents the point on the Central Axis marked by β in Figure 8.6.

be calculated by determining which radii intersect with this line segment. This reduces the determination of intersection distances to a two-dimensional problem.

For single intersections, the lower and upper bounds of radii is $[min(|E_0|,|E_1|),max(|E_0|,|E_1|)]$. Note that the 2D coordinates of E_0 and E_1 are equivalent to the vectors $\beta\vec{E}_0$ and $\beta\vec{E}_1$, respectively.

A double intersection occurs when the closest point to β on line L is also on the line segment $E_0\,E_1$. When double intersections exist, the range of radii in which they occur is $[|C|,min(|E_0|,|E_1|)]$.

Given the aforementioned background, the next step of our method is aligning the vector $\beta\vec{C}$ with the y-axis, as illustrated in Figure 8.7. The objective of this step is to simplify the

remaining calculations for the intersection test. Alignment is done by normalising the vector between E_0 and E_1, and subsequently rotating the triangle vertices around the z-axis; the coordinates of the normalised vector can be used directly as sine and cosine coefficients for the rotation.

At this stage, determining the existence of a double intersection is inexpensive, and can be achieved by comparing signs of the x components of the aligned E_0 and E_1 coordinates. Different signs indicate that a double intersection exists. If so, the length of $\beta\vec{C}$ (the rotated y-coordinate of C) represents the lower bound of radii which correspond to double intersections.

The intersection test itself can be done by comparing a given radius against the computed ranges, which yields an intersection count corresponding to that radius.

Summarising, computing the range of values of α that will result in a single or double intersection for a given value of β involves the following steps:

1. Determine the intersection points E_0 and E_1 for any value of value of β where L is defined, as shown in Figure 8.6.

2. Rotate E_0 and E_1 around the z-axis such that the vector $\vec{E_0E_1}$ is aligned with the x-axis (as shown in Figure 8.7).

3. Determine the distance of E_0 and E_1 from the z-axis.

4. The range of circle radii in which single intersections occur is $[min(|E_0|,|E_1|),max(|E_0|,|E_1|)]$.

5. Determine the existence of a double intersection by comparing the signs of the x-coordinates of E_0 and E_1. If they are different then a double intersection exists.

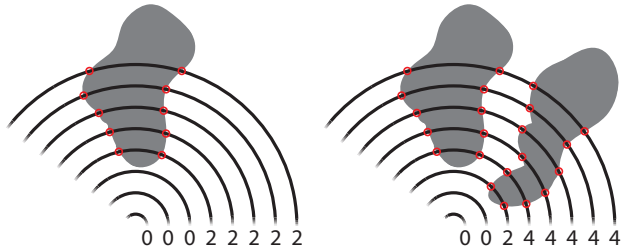
6. If a double intersection exists, the range of α coordinates (circle radii) corresponding to the double intersection is the y-coordinate of either E_0 or E_1 and the shortest distance between the z-axis and E_0 or E_1.

8.3.2 A Clutter-Resistant RICI Distance Function

Spin Images, by their nature of being generated from oriented point clouds, are inherently noisy. They have as such relied on statistical correlation to compute similarity. The idea here is that two matching bins tend to have proportionally similar accumulated sample counts. Unfortunately, this method is susceptible to the effects of clutter. Additional geometry present in the support volume causes portions of the image to receive additional projected point samples, which consequently negatively affects the computed correlation value.

When it comes to comparing RICIs, one important downside of the Pearson Correlation Coefficient is that it is not defined for sequences of constant values. While this scenario is unlikely to occur for Spin Images, there exist situations in which RICIs consist solely of pixels with equivalent intersection counts. For these situations, the Pearson correlation coefficient is undefined, and therefore an insufficient solution for comparing RICIs. Handling these edge cases separately is possible, but results in a solution that requires balancing awarded scores against normal situations.

Meanwhile, the RICI does not have the aforementioned issue of noise, and is as such not bound solely to using statistical methods for measuring similarity. For these reasons



(a) Intersection counts without clutter

(b) Intersection counts with clutter

Figure 8.8: Demonstration of changes in intersection counts generally being unaffected by clutter. A portion of a single layer of intersection circles is shown. Intersections with the shape surface have been marked.

we propose a new distance function, which is by design able to resist some of the negative effects of clutter, primarily by exploiting features of the RICI.

First, the distance function does not consider the values of pixels in the RICI. Instead, *changes* in pixel values (i.e. intersection counts which show up as edges in the RICI) are compared. As RICIs are free of noise, it is possible to interpret pixel values directly. The main advantage of this approach is that changes in intersection counts are largely unaffected by clutter. The reason for this can be seen in Figure 8.8.

In Figure 8.8a, a cross section is shown of an arbitrary 3D shape. On the same plane, circles are drawn with increasing radii, similar to how RICI images are computed. The numbers below each circle indicate the number of intersections they encounter, which corresponds to the value of their respective pixels in the RICI image.

Similarly, Figure 8.8b shows the same situation in which a clutter object has been added. From the intersection counts can be seen that even though the absolute intersection counts have now changed, the change in intersection counts from the third to the fourth circle, caused by the original object, is still present.

Second, when searching, our distance function treats the *needle* (query) and the *haystack* image asymmetrically, in contrast to the Pearson correlation coefficient. One can use the needle image to deduce what features to look for in a given haystack image.

This asymmetry consists of only computing a sum of squared differences distance on pixels where there are *changes* in the needle RICI image.

Returning to Figure 8.8, we'll assume that Figure 8.8a shows a cross section of the needle object that we are attempting to locate in the cluttered haystack scene shown in Figure 8.8b. In our needle image, only the increased intersection counts from the third to the fourth circle are relevant. Including other pixels is not relevant, as there are no changes in the needle image's intersection counts. We can therefore ignore these pixels in our distance computation. This also means any clutter present in the haystack image is ignored by this method.

The proposed Clutter Resistant Distance function $CRD(needleRICI, haystackRICI)$ is shown in Equation 8.7, and the corresponding pseudocode is given in Listing 1. Note here that the distance function is positive, but not symmetric. It has a complexity of O(1), because comparing a descriptor pair requires a fixed number of operations.

```
def clutterResistantDistance(needle, haystack):
    score = 0
    for row r in [0..N_bins]:
        # Skip first column
        for column c in [1..N_bins]:
            needleDelta = needle[r][c] - needle[r][c-1]
            haystackDelta = haystack[r][c] - haystack[r][c-1]
            if needleDelta != 0:
                score += (needleDelta - haystackDelta) *
                         (needleDelta - haystackDelta)
    return score
```

Listing 1: Pseudocode for our proposed method for computing the distance between two RICI images.

$$D(rici, r, c) = rici(r, c) - rici(r, c-1) \tag{8.6}$$

$$CRD(n,h) = \sum_{r=0}^{N_{bins}} \sum_{c=1}^{N_{bins}} \begin{cases} (D(n,r,c) - D(h,r,c))^2, & \text{if } D(n,r,c) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{8.7}$$

8.4 Evaluation

The proposed method has been evaluated in terms of its clutter resistance, generation speed, and matching performance. Where applicable, we compare our method against the two most referenced among those listed in survey [13] as being clutter resistant. These are the Spin Image[1] and the 3D Shape Context. It is worth noting that the survey also observes that popular descriptors such as the Fast Point Feature Histogram [29], Unique Signatures of Histograms [33], and Rotational Projection Statistics [14], do not exhibit optimal performance under cluttered conditions. We have therefore implemented the above two most referenced clutter resistant methods on the GPU, to allow a direct comparison on the same dataset.

The novel Clutterbox Experiment is proposed in order to evaluate the effect of clutter on the descriptors' matching performance.

8.4.1 The Clutterbox Experiment

In previous work, clutter has typically been defined as the proportion of area within the support volume that does not belong to the object being recognised. Greater proportions of clutter generally imply worse descriptor performance. The expression used in previous work, initially proposed by Johnson et al. [18] is shown in Equation 8.8. Here A_{all} is the surface area of all objects within the support volume and A_{object} is the surface area of the object of interest.

$$clutter = \frac{A_{all} - A_{object}}{A_{all}} \tag{8.8}$$

The objective of the proposed evaluation method, which we call the "clutterbox

[1] [33] and [14] also support the SI as a clutter resistant descriptor.

experiment", is to measure the relationship between increasing levels of clutter and the resulting performance of the descriptor being tested.

In previous clutter experiments, clutter has generally been evaluated by measuring descriptor performance against levels of clutter present at points in a scene without controlling the points' identities. However, this measures the effects of two parameters combined; the descriptor's ability to recognise the desired shape, and the level of clutter present around it. Ideally an evaluation of the effects of clutter should control the former of these parameters, while varying the latter. This is the primary objective that the clutterbox experiment addresses.

Varying clutter levels in the neighbourhood of an object can be done trivially by adding triangles, points, spheres, or cubes in random locations and sizes around an object. However, this kind of clutter is not representative of the clutter that can be expected in a realistic 3D scene. The clutterbox experiment therefore inserts complete objects rather than random noise. This results in a more natural distribution of clutter in the scene, and therefore more directly measures the effect of clutter that can be expected of a given descriptor when applied in a practical context.

The clutterbox experiment is executed a large number of times by varying objects and their transformations, in order to provide robust results, independent of object type.

The steps of the experiment are outlined below:

1. Define the clutterbox as a cube of side s.
2. Select n objects at random from a large object collection.
3. Scale and translate each object such that it fits exactly inside a unit sphere.
4. Pick one of the n objects at random. This is the reference object.
5. Compute the reference descriptor set $\{RD\}$, by computing one descriptor for each unique vertex of the reference object.
6. For each of the n objects in random order, but starting with the reference object:
    a) Place the object within the clutterbox, at a randomly chosen orientation and position, with the constraint that the bounding sphere fits entirely within the clutterbox.
    b) Compute the set of cluttered descriptors $\{CD\}$, by computing one descriptor for each unique vertex of the combined mesh in the clutterbox.
    c) For each $d \in \{RD\}$, create a list of ranked distances to all $c \in \{CD\}$. Keep the rank where the corresponding cluttered descriptor was found in the ranked list ($0 \leq rank \leq |\{CD\}| - 1$). Note that lower ranks are better.
    d) Create a histogram where bin i holds the number of times the correct vertex is found in the search results at rank i.

Thus the output of the clutterbox experiment is a list of histograms, one for each level of clutter. A visualisation of a sequence of scenes with increasing clutter generated by the above experiment is shown in Figure 8.9.

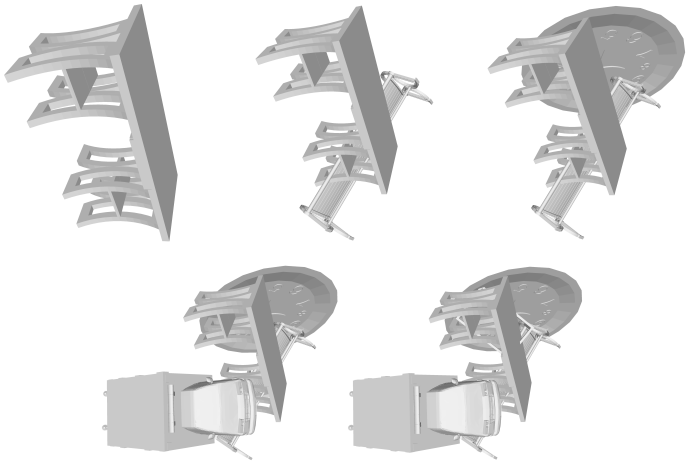


Figure 8.9: Visual representation of the increasing number of clutter objects added into the clutterbox. The leftmost image only contains the reference object.

8.4.2 Clutter Resistance Evaluation

We used the clutterbox experiment to quantify the effects of clutter on the SI and 3DSC versus the proposed RICI descriptor. For our object collection, we selected the combined SHREC2017 dataset [31], which consists of 51,162 triangle meshes.

In the case of the SI and 3DSC, the combined triangle mesh of the reference and clutter objects was sampled into a point cloud before generating their descriptors; RICI descriptors are generated from the triangle mesh directly. For optimal performance, SI and 3DSC require a high number of samples to ensure a low level of noise in the produced descriptors. However, one cannot increase the sample count indefinitely as that results in a lower generation rate. Based on our experimental evidence on the given dataset, we feel that 10 samples per triangle is a reasonable point on this trade-off.

While Johnson et al. define the bin size (thus the support radius) of the SI to be equal to the mesh resolution, we do not believe their reasoning holds any longer for present day 3D objects. Similar objects can have significant variance in their resolution. As such, making the support radius dependent on the mesh resolution is not a guarantee for better matching performance. We therefore use a constant support radius for all tested methods, set to 0.3 units, relative to the bounding unit sphere, for all scenes in the experiment for ease of reproducibility. For the 3DSC, we set the minimum support radius to $r_{min} = 0.048$ units, which is proportionally the same as the one originally used by Frome et al. [11].

We executed the experiment 1,500 times, iteratively cluttering a scene with $n = 1$ (the reference object only), $n = 5$, and $n = 10$ objects, into a clutterbox of size $s = 3$. The size of the RICI and SI descriptors N_{bins} was set to 64x64 bins, while the 3DSC descriptor's dimensions were left the same as those used in previous work ($J = 15$, $K = 11$, $L = 12$ [13] [11]). A more detailed discussion on size settings can be found in Section 8.5.2. In order to visualise the histograms generated by the clutterbox experiment, we opted to compute the fraction of the bin representing rank 0 in the histogram against the sum of all bins (all search results). For clarity, each sequence of such fractions has been sorted individually to produce

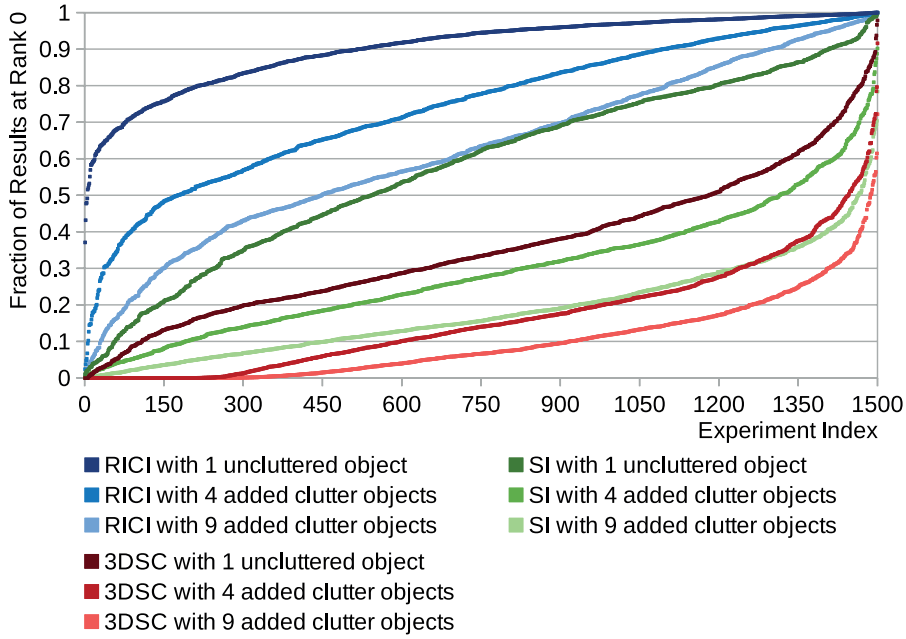


Figure 8.10: Percentage of search results for all tested methods that ended up at rank 0 for each of the 1500 performed experiments.

monotonically increasing curves. The results are shown in Figure 8.10.

The support angle parameter used to generate the SI results in Figure 8.10 requires further elaboration. In their original SI paper, Johnson et al. claim this filter reduces the effects of self-occlusion and clutter. However, our testing which compared using a support angle filter to not filtering any input points (Figure 8.11) could not confirm this. All SI results in this paper therefore do not apply any support angle filter, as this favours the SI.

While Figure 8.10 shows that our RICI descriptor clearly outperforms both the SI and 3DSC in scenes that contain clutter (see Equation 8.8), it is also relevant to gain insight in the relationship between descriptor performance and the specific clutter level present in the support region. Figure 8.12 shows a heatmap plot of the fractional area of clutter present in the support volume around each Spin Vertex, versus the rank of the corresponding descriptor in the haystack. It can be observed that the RICI trends towards lower ranks than the SI and 3DSC, even at high levels of clutter. Furthermore, while the 3DSC generally does not outperform the SI, it appears more clutter resistant than the SI at extreme clutter levels ($>$ 90%).

The heatmaps have been computed over 73.5 million search results extracted from scenes with 4 added clutter objects, based on the results of the Clutterbox experiment.

It is not expected that a RICI image would be very dependent on mesh resolution (which may be related to scanning) as intersection counts should in most cases not be very sensitive to that.

The experiment was implemented using C++, with the descriptor generation and search kernels written in CUDA 10.0. The code was written in such a way that given a dataset of objects, a single random seed determines all randomly chosen parameters, making all

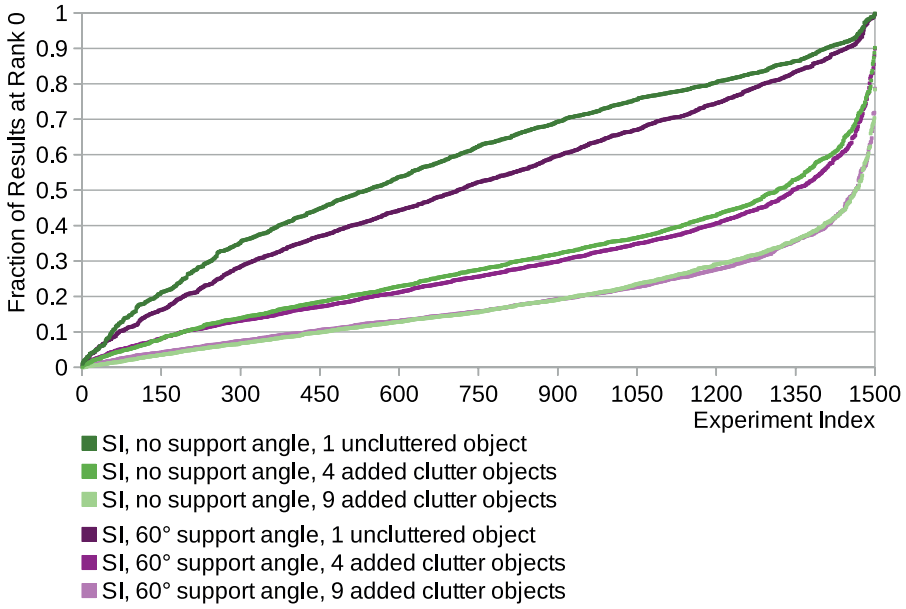


Figure 8.11: Percentage of SI search results that ended up at rank 0 for each of the 1500 performed experiments for two different support angles.

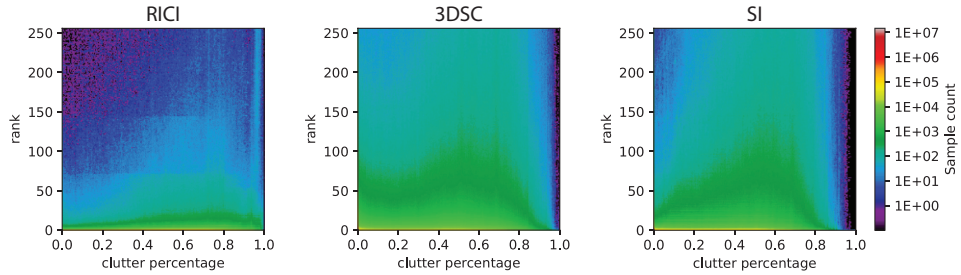


Figure 8.12: Visualisation of the clutter resistance of RICI, SI, and 3DSC. Colours are mapped using a logarithmic function (colours toward the red end of the spectrum lower in the images is better). A pixel's colour represents the number of search results, i.e. descriptors, that ended up in the specific rank in relation to the amount of clutter within their support volume.

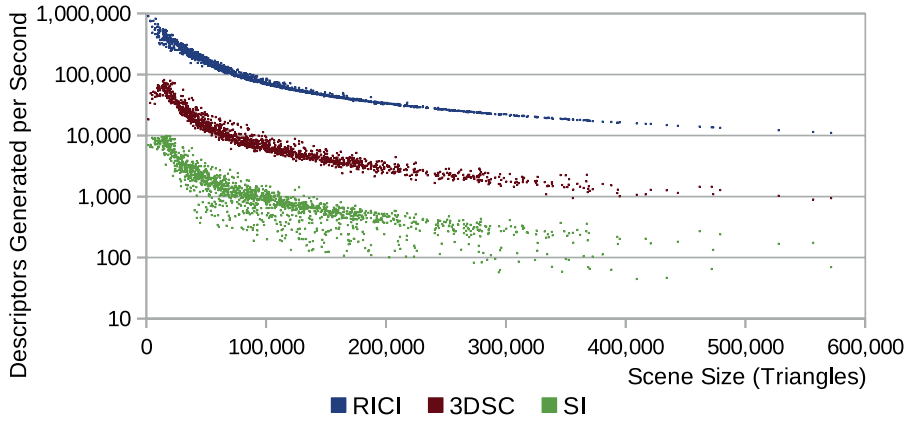


Figure 8.13: Relationship between the number of triangles present in the scene, and the rate at which our implementations generate RICI, SI, and 3DSC descriptors.

results reproducible. The experiment was executed on a combination of Nvidia Tesla cards (P100 16GB, V100 16GB, and V100 SXM3 32GB). All time-based results were exclusively gathered on the latter. One relevant implementation detail is that in cases where multiple search results have the same distance (which may occur due to reasons such as object self-similarity), we use the highest (best) rank of the matched haystack image for the sake of consistency.

8.4.3 Generation Performance

Figure 8.13 shows the difference in the rate at which the RICI, SI, and 3DSC descriptors are generated. As can be seen, the RICI is approximately one order of magnitude faster than the 3DSC, and two orders faster than the SI for the given settings.

8.4.3.1 Performance of Point Projection Algorithm

The largest portion of the computational effort involved in the RICI and SI generation algorithms require projecting points into cylindrical coordinate space. We have proposed an efficient algorithm for this, as outlined in Section 8.3.1.1.

A similar algorithm is included in Point Cloud Library [29], as part of the Spin Image generation implementation. To the best of our knowledge, this was up to now the most efficient implementation available. We therefore compare our projection algorithm against this previous work.

We evaluate both algorithms using a microbenchmark which projects a sequence of $1 \cdot 10^9$ randomly generated points. To ensure a fair comparison, all code unrelated to point projection has been removed from the Point Cloud Library SI generation implementation. The results are shown in Table 8.1.

It's worth noting that points are projected into cylindrical coordinates relative to the same oriented point. Our method can therefore precompute the values of N_{ax}, N_{ay}, N_{bx}, and N_{bz}, as outlined in section 8.3.1.1. Both methods were tested on an Intel Core i7-8750H CPU.

PCL (s)	Proposed method (s)
7.559	3.084

Table 8.1: Point projection algorithm average execution times for projecting $1 \cdot 10^9$ points.

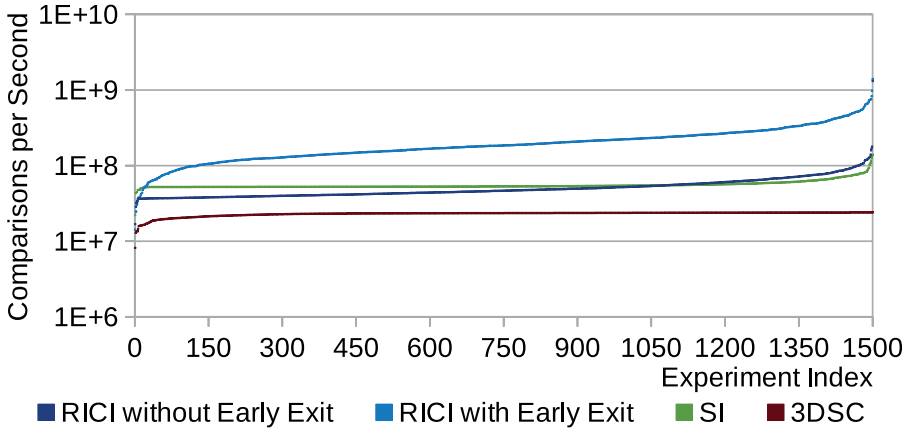


Figure 8.14: Image matching rates in a scene with 5 objects. For clarity, each sequence has been sorted individually to produce a monotonically increasing curve.

8.4.4 Matching Rate

The rates of evaluating the distance functions for each method are shown in Figure 8.14. As can be seen, the RICI distance function's execution times are similar to the SI's Pearson correlation coefficient, while 3DSC is significantly slower.

For all methods, the bandwidth of the GPU memory bus is the main factor limiting the comparison rate. As our proposed distance function relies on computing the difference between neighbouring pixels, this would in a naive implementation, have required double the bandwidth. Instead, we use specialised "shuffle instructions" to read the value of neighbouring pixels without having to resort to another memory transaction, thereby halving the needed memory bandwidth. The result is a kernel whose memory bandwidth requirements, and consequently execution time, is similar to the Pearson Correlation Coefficient used to compare Spin Images.

We further optimised our implementation by using an early exit condition. Since the distance score can only go up for every subsequent pixel being processed, if the only objective is determining whether the distance between two images is smaller than some given threshold distance (as is the case in many retrieval applications), it is possible to cease execution when a predetermined distance threshold is exceeded. In our clutterbox experiment, this threshold can be trivially precomputed. Utilising this early exit condition resulted on average in a 4.2 times speedup over the SI distance function.

8.5 Observations and Discussion

There are several topics and observations that may be relevant for the interpretation of the presented results.

8.5.1 Analysis of Experimental Results

While analysing the results presented in Section 8.4, we made several observations that are relevant to their interpretation. Figure 8.15 contains a visualisation of a subset of these.

Figure 8.15a shows the result set where RICI experienced the smallest decrease in matching performance between 0 and 9 added clutter objects in the scene. It is also possible to observe the clutter resistant properties of RICI. The seat part of the desk chair is significantly cluttered, while the wheels experience relatively small amounts of clutter (and remain visible). All three methods are capable of reasonably recognising these exposed wheels, however, the SI and 3DSC descriptors in large part fail to recognise the cluttered seat part.

Figure 8.15b shows the result set where RICI experienced the largest drop in performance between the scenes with 0 and 9 added clutter objects. The primary cause of this drop is due to the cuboid-like shape and low level of details on the police van, which causes a low number of changes in intersection counts. In turn, the produced RICI images become relatively susceptible to clutter.

Figure 8.15c shows the experiment where RICI performed worst on the uncluttered reference object. The particular object, a bookshelf, has high levels of self-similarity; a property which is also, to varying degrees, present in other objects in the CAD-oriented SHREC2017 dataset. Thus any local descriptor would rank vertices belonging to self-similar regions equally and whether they end up at Rank 0 is a matter of luck. One would expect to find them within the top s ranks, where s is the number of self-similar vertices. On the other hand, this is a useful tool for detecting self-similar regions.

To investigate this further we visualised the results of an experiment where the reference object had countable symmetric features, as shown in Figure 8.16. As opposed to Figure 8.15, we highlighted in red those vertices that were detected in the top s ranks instead of only rank 0. For instance, vertices in the table's legs are expected to constitute 12 self-similar partitions (6 legs with a symmetric front and backside each), which are all detected in the top 12 results, as shown in Figure 8.16d. Also all vertices in the base of the tabletop are correctly detected within the top 4 results (4-way symmetry).

In contrast to Figure 8.15c, Figure 8.15d shows the experiment in which RICI had the highest recognition rate in the uncluttered scene. Little matching performance is lost after adding significant amounts of clutter.

In Figure 8.15e the experiment whose drop in matching performance was closest to the total average of all performed 1500 experiments is shown. Worth noting here is the relatively low drop in recognition performance between the uncluttered scene, and the scene with 9 added clutter objects.

Finally, in Figure 8.15f a rare phenomenon is shown where matching performance slightly improves between 4 and 9 added clutter objects.

8.5.2 Performance of 3DSC

As can be seen in Figure 8.10, in contrast to the results obtained in previous work [11] [13], the SI generally outperforms the 3DSC descriptor. The primary cause of this is that in previous work, the SI resolution was set to the 15x15 bins used originally by Johnson et al. [18]. In contrast, we used a resolution of 64x64 bins for parity with the RICI descriptor, which we also consider to be a resolution more suitable to the capabilities of modern processors. This significant increase in resolution meant the SI descriptor in our testing

performs better than 3DSC with our chosen settings.

The decision to use the same bin dimensions for 3DSC as in previous work was primarily motivated by a tradeoff between comparison performance and GPU hardware limitations. Our implementation makes use of shared memory when comparing 3DSC descriptors, due to the needle and haystack descriptor both being accessed once for each radial division. Current GPU shared memory pools allow fitting of approximately 2 image pairs sized at default settings simultaneously, which implies the number of bins can either be left intact, or doubled, or performance can be expected to be suboptimal. While it would be possible to double the number of bins in the 3DSC descriptor (which would make its memory requirements equal to the SI and RICI) leading to an increase in matching performance, the matching rate would decrease below acceptable levels because of the distance algorithm used. We therefore consider the used settings to be the best balance between quality and execution time for 3DSC.

8.6 Conclusion

In this paper, a clutter resistant shape descriptor, RICI, is presented and evaluated using a novel evaluation framework for such descriptors, called the clutterbox experiment. Novel algorithms for cylindrical coordinate projection, circle-triangle intersection, and the rasterization of triangles in cylindrical coordinates were presented. The largest quantitative evaluation of the SI, 3DSC, and RICI methods to date is also made, along with a useful observation for the SI support angle.

The main advantages of RICI are its noise-free nature and generation speed, while the related distance function makes it clutter resistant. We anticipate that the proposed clutterbox experiment, which is being made public, will aid future benchmarking of shape descriptors for cluttered scenes.

8.7 Acknowledgements

The authors would like to thank the HPC-Lab leader and PI behind the "Tensor-GPU" project, Prof. Anne C. Elster, for access to the Nvidia DGX-2 system used in the experiments performed as part of this paper. Additionally, the authors would like to thank the IDUN cluster at NTNU for the provision of additional computing resources.

8.8 References

[1] Jrgen Assfalg, Marco Bertini, Alberto Del Bimbo, and Pietro Pala. Content-based retrieval of 3-d objects using spin image signatures. *IEEE Transactions on Multimedia*, 9(3):589–599, 2007.

[2] Owen Carmichael, Daniel Huber, and Martial Hebert. Large data sets and confusing scenes in 3-d surface matching and recognition. In *3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on*, pages 358–367. IEEE, 1999.

[3] Hui Chen and Bir Bhanu. 3d free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, 28(10):1252–1262, 2007.

[4] Daniela Craciun, Guillaume Levieux, and Matthieu Montes. Shape Similarity System

driven by Digital Elevation Models for Non-rigid Shape Retrieval. In Ioannis Pratikakis, Florent Dupont, and Maks Ovsjanikov, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2017.

[5] Nolan Davis, Dennis Braunreiter, Cezario Tebcherani, and Masatoshi Tanida. 3d object matching on the gpu using spin-image surface matching. In *Advanced Signal Processing Algorithms, Architectures, and Implementations XVIII*, volume 7074, page 707408. International Society for Optics and Photonics, 2008.

[6] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–205, 2018.

[7] H Quynh Dinh and Steven Kropac. Multi-resolution spin-images. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 863–870. IEEE, 2006.

[8] Cong Feng, Andrei C. Jalba, and Alexandru C. Telea. A Descriptor for Voxel Shapes Based on the Skeleton Cut Space. In A. Ferreira, A. Giachetti, and D. Giorgi, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2016.

[9] Alex Flint, Anthony Dick, and Anton Van Den Hengel. Thrift: Local 3d structure recognition. In *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)*, pages 182–188. IEEE, 2007.

[10] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *European conference on computer vision*, pages 224–237. Springer, 2004.

[11] Adam R Gerlach and Bruce K Walker. Accelerating robust 3d pose estimation utilizing a graphics processing unit. In *Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*, volume 7878, page 78780V. International Society for Optics and Photonics, 2011.

[12] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89, 2016.

[13] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3d local surface description and object recognition. *International journal of computer vision*, 105(1):63–86, 2013.

[14] Yulan Guo, Ferdous Ahmed Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Trisi: A distinctive local surface descriptor for 3d modeling and object recognition. In *GRAPP/IVAPP*, pages 86–93, 2013.

[15] Ruizhen Hu, Lubin Fan, and Ligang Liu. Co-segmentation of 3d shapes via subspace clustering. In *Computer graphics forum*, volume 31, pages 1703–1713. Wiley Online Library, 2012.

[16] Daniel F Huber and Martial Hebert. Fully automatic registration of multiple 3d data

sets. *Image and Vision Computing*, 21(7):637–650, 2003.

[17] Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. Deep learning advances in computer vision with 3d data: A survey. *ACM Computing Surveys (CSUR)*, 50(2):1–38, 2017.

[18] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[19] Ioannis A Kakadiaris, Georgios Passalis, George Toderici, Mohammed N Murtuza, Yunliang Lu, Nikos Karampatziakis, and Theoharis Theoharis. Three-dimensional face recognition in the presence of facial expressions: An annotated deformable model approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(4):640–649, 2007.

[20] Luming Liang, Andrzej Szymczak, and Mingqiang Wei. Geodesic spin contour for partial near-isometric matching. *Computers & Graphics*, 46:156–171, 2015.

[21] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[22] Sotiris Malassiotis and Michael G Strintzis. Snapshots: A novel local surface descriptor and matching algorithm for robust 3d surface alignment. *IEEE Transactions on pattern analysis and machine intelligence*, 29(7):1285–1290, 2007.

[23] Ajmal S Mian, Mohammed Bennamoun, and Robyn Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1584–1601, 2006.

[24] John Novatnack and Ko Nishino. Scale-dependent/invariant local 3d shape descriptors for fully automatic registration of multiple sets of range images. In *European conference on computer vision*, pages 440–453. Springer, 2008.

[25] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J Mitra. Exploration of continuous variability in collections of 3d shapes. *ACM Transactions on Graphics (TOG)*, 30(4):33, 2011.

[26] Giuliano Pasqualotto, Pietro Zanuttigh, and Guido M Cortelazzo. Combining color and shape descriptors for 3d model retrieval. *Signal Processing: Image Communication*, 28(6):608–623, 2013.

[27] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[28] Manolis Savva, Fisher Yu, Hao Su, Asako Kanezaki, Takahiko Furuya, Ryutarou Ohbuchi, Zhichao Zhou, Rui Yu, Song Bai, Xiang Bai, Masaki Aono, Atsushi Tatsuma, S. Thermos, A. Axenopoulos, G. Th. Papadopoulos, P. Daras, Xiao Deng, Lian Zhouhui, Bo Li, Henry Johan, Yijuan Lu, and Sanjeev. Mk. Shrec'17 track large-scale 3d shape retrieval from shapenet core55. In *Proceedings of the Eurographics Workshop on 3D Object Retrieval*, 2017.

[29] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010.

[30] Bart Iver van Blokland, Theoharis Theoharis, and Anne C Elster. Quasi spin images. 2018.

[31] Zizhao Wu, Yunhai Wang, Ruyang Shou, Baoquan Chen, and Xinguo Liu. Unsupervised co-segmentation of 3d shapes via affinity aggregation spectral clustering. *Computers & Graphics*, 37(6):628–637, 2013.

[32] Sameh M Yamany and Aly A Farag. Surface signatures: an orientation independent free-form surface representation scheme for the purpose of objects registration and matching. *IEEE transactions on pattern analysis and machine intelligence*, 24(8):1105–1120, 2002.

[33] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1802–1811, 2017.

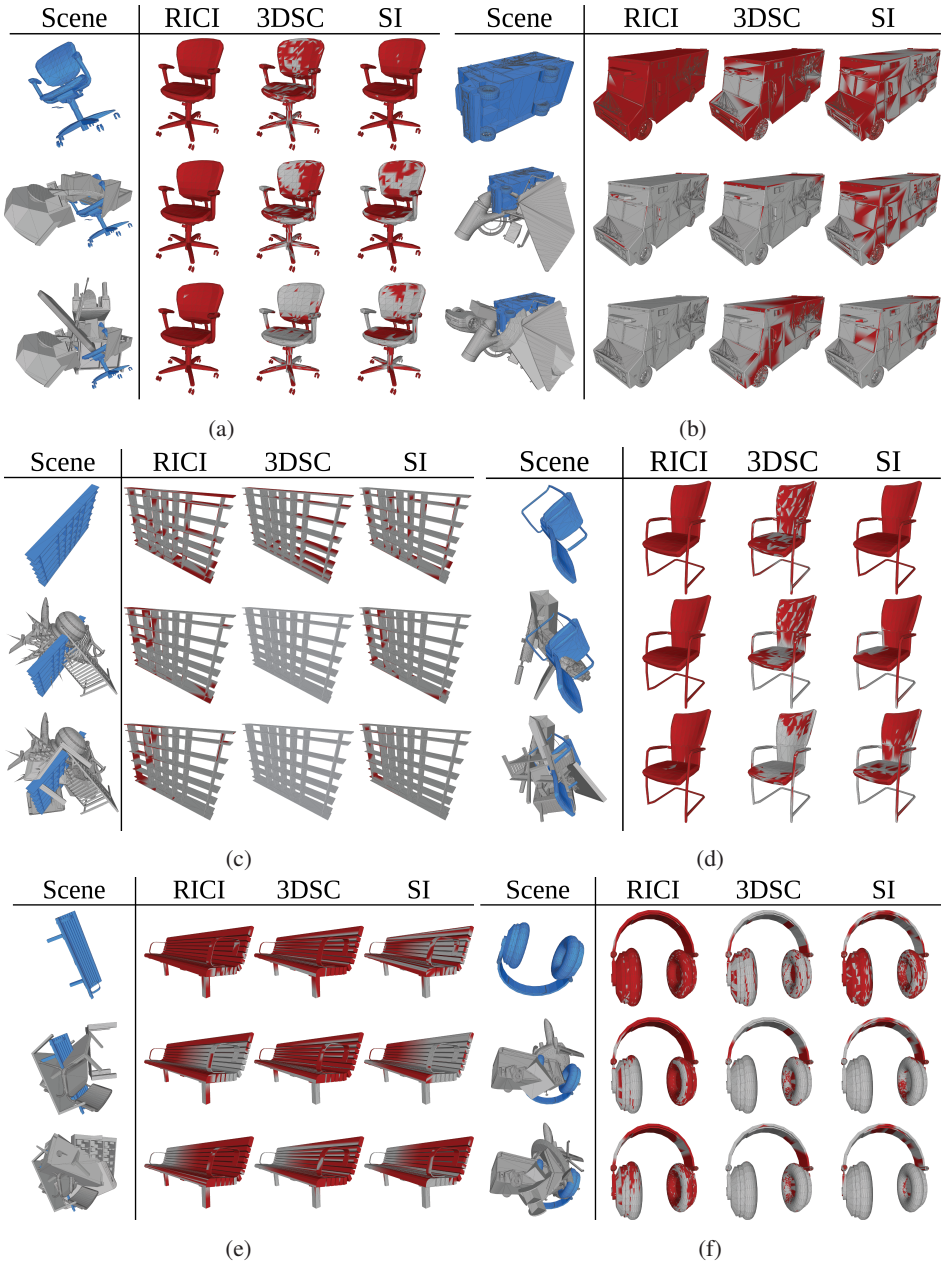
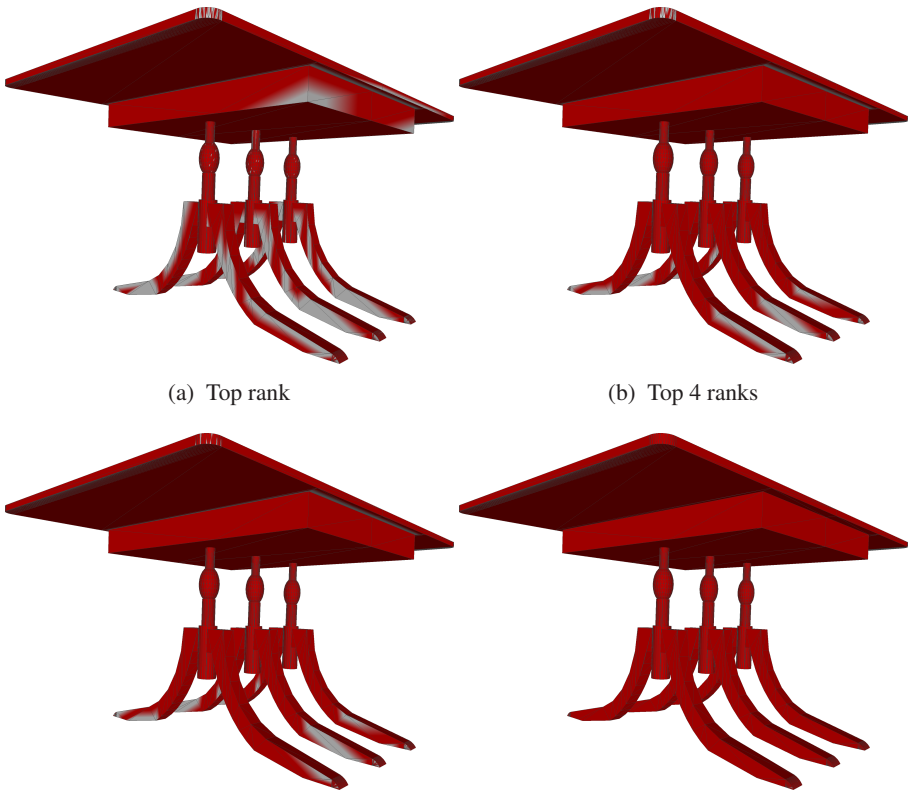


Figure 8.15: Visualised results from 6 selected experiments. For each of the 6 subfigures, the Clutterbox scene (with 1, 5, and 10 objects) is shown on the left hand side, with the reference object highlighted in blue. Vertices correctly ranked at index 0 are highlighted in red, other vertices are coloured grey.



(a) Top rank (b) Top 4 ranks

(c) Top 6 ranks (d) Top 12 ranks

Figure 8.16: Symmetric object whose vertices were present in the top s ranks of the search results, with varying values of s.

Chapter 9

Paper D - An Indexing Scheme and Descriptor for 3D Object Retrieval Based on Local Shape Querying

Authors

Bart Iver van Blokland and Theoharis Theoharis

Published in

Computers & Graphics Volume 92, Pages 55-66, 2020, Elsevier

Awards

This paper has been awarded the *Graphics Replicability Stamp* by the Graphics Replicability Stamp Initiative (GRSI).

An Indexing Scheme and Descriptor for 3D Object Retrieval Based on Local Shape Querying

Bart Iver van Blokland[1], Theoharis Theoharis[1]
1) Norwegian University of Science and Technology, Norway

Abstract

A binary descriptor indexing scheme based on Hamming distance called the Hamming tree for local shape queries is presented. A new binary clutter resistant descriptor named Quick Intersection Count Change Image (QUICCI) is also introduced. This local shape descriptor is extremely small and fast to compare. Additionally, a novel distance function called Weighted Hamming applicable to QUICCI images is proposed for retrieval applications. The effectiveness of the indexing scheme and QUICCI is demonstrated on 828 million QUICCI images derived from the SHREC2017 dataset, while the clutter resistance of QUICCI is shown using the clutterbox experiment.

9.1 Introduction

The problem of shape retrieval has thus far primarily been posed as an object based one. Many proposed algorithms aim to answer queries such as ‘find all chairs’, or ‘find buses similar to this sample bus’. However, objects are not a single large shape; they are the sum of many small details that combined produce a larger, more complex whole. For instance, a wheelbarrow may contain shapes such as a slightly bent flat surface, a curved cylinder, or a large disc. Individually these shapes may not be unique to that object, but their specific combination and arrangement makes it an object useful for garden work.

It may be argued that querying of such smaller (partial) shapes fall under the existing class of partial object retrieval. Thus one can pose retrieval queries such as ’find all objects that contain a spout like this’, which would presumably retrieve teapots (as well as other objects with spouts). Unfortunately, partial retrieval requires the availability of the partial shape that is to be retrieved.

However, in many cases it is useful to be able to pose more general geometric queries such as ’retrieve objects containing an S-bend’ for finding bottles with that specific cross-section. This could be easily specified by drawing such a curve in 2D thus not necessitating the existence of a partial query object.

One important problem with this type of approach, which would have to describe shape at a very low level, is the sheer volume of local descriptors that would be generated, potentially one for every vertex. Not only would they require a large amount of storage but it would also be quite slow to search them.

We thus propose:

- A robust and efficient novel local binary shape descriptor (called QUICCI)
- An efficient novel indexing scheme called Hamming Tree for bit strings such as QUICCI
- A novel distance function used for retrieval of bit strings (called Weighted Hamming distance)

After an introduction to relevant background material in Section 9.2, each of these contributions are described in the above listed order in Sections 9.3, 9.4, and 9.5, respectively. The various methods are evaluated in Section 9.6, and some specifics are discussed in Section 9.7.

9.2 Background and Related Work

This section is divided in two parts, corresponding to the main contributions of the paper: indexing bit strings and local shape descriptors.

9.2.1 Indexing Bit Strings

The need for indexing collections of bit strings primarily stems from two main categories of methods; those utilising dimensionality reduction to map higher dimensional descriptors on to shorter binary vectors, and binary feature descriptors.

Dimensionality reduction is often done through a method utilising Locality-Sensitive Hashing (LSH), initially described by Har-Peled et al. [24]. These aim to represent larger, more varied feature vectors in shorter bit strings, where similar feature vectors will produce similar bit strings, thereby significantly reducing the search space for finding closest neighbours. Popular methods applying LSH include Minhash proposed by Broder et al. [12] (as well as a more scalable variant [11]), and Simhash [42] by Sadowski et al.

A number of binary feature descriptors have been proposed aimed at various retrieval applications. For images, the most popular binary features proposed to date include BRIEF by Calonder et al. [13], a rotation invariant extension named ORB by Rublee et al. [40], and a both rotation and scale invariant keypoint descriptor called BRISK by Leutenegger et al. [27]. An example of a binary descriptor for 3D point matching is B-SHOT, proposed by Prakhya et al. [37]. The lengths of these descriptors vary between 128 and 512 bits.

While LSH derived methods are capable of significantly reducing dimensionality in the source data, large quantities of indexed data may cause a high number of hash bins to be created. However, not all hash bins may receive a similar number of entries, and the creation of all possible bins for a given bit string length is not always feasible, thereby creating the need to discover the existence of nearby hash bins with relatively low Hamming distances. This discovery process can be costly, particularly when no close neighbours exist. Binary feature descriptors are inherently longer, and for that reason face a similar problem.

Retrieval from large collections of bit strings, where each bit string is ranked by hamming distance from a query string, is known in the literature as the *n Nearest Neighbours* Hamming problem, and a variety of methods have been proposed [10] [9] [3] [31].

However, these early methods are limited in their design to the efficient retrieval of neighbours with Hamming distances of up to 2, support for short bit sequences only, or both. More recent methods have addressed the problem more effectively, and do not exhibit the aforementioned problems.

Norouzi et al. proposed the Multi-Index Hashing (MIH) algorithm [32]. The method works by dividing all indexed bit strings into equally sized disjoint substrings, and constructing a hash table for each set of corresponding substrings. These can be queried by subdividing the query string in a similar fashion, and querying each hash table for all permutations of the query substring within a given Hamming distance. The set of candidate matches can be refined when testing subsequent hash tables, as strings which surpass the Hamming distance limit can be excluded prematurely. The authors show that MIH outperforms the most significant previous work, however, the requirement that all permutations within a given Hamming distance must be tried on hash sets becomes a performance bottleneck when this limit is high.

Chappell et al. proposed a system for approximate nearest-neighbour search of bit strings [14] aimed at locating such nearest neighbour hashes by creating inverted lists of smaller bit string "slices", similar to the divisions done in MIH. However, for larger collections of longer bit strings, such as binary descriptors, the method does not scale due to each slice list increasing in size linearly.

Reina et al. [39] presented an improved variation of MIH. This is a hybrid indexing scheme, where a *trie* (prefix tree) is used to store the index tree itself, and a separate hash table is exploited to prune tree branches during a query by checking a specific bit string's existence in the index when the tree node's common prefix has reached a given Hamming distance limit. In similar fashion to MIH, bit strings are divided into substrings, and from each corresponding substring a separate index is constructed. While the method is shown to outperform MIH, it is hampered by the fact that for its efficiency (the pruning of branches which are known not to contain matches) it relies on the existence of a fixed Hamming distance limit. Constructing a querying algorithm which does not contain this optimisation is possible, but as the authors themselves state, this would significantly degrade querying performance. Moreover, creating one index for each subdivision in the input string, as well as the corresponding hash table and trie that each of these includes, adds significant storage overhead.

The Hamming Tree data structure proposed in this paper commands a significantly lower overhead, as only a single indexing structure is created. The proposed querying algorithm can dynamically cut off the querying process and does not necessarily require a Hamming distance limit to be set.

9.2.2 Local 3D Shape Descriptors

Local approaches to 3D object retrieval are advantageous to global methods due to their inherent resistance to clutter and shape variations. The field is well developed, and numerous descriptors have been proposed to date, e.g. [33] [10] [14] [12].

One popular descriptor is the Fast Point Feature Histogram (FPFH) [41]. It is constructed in two phases. First, a Simplified Point Feature Histogram (SPFH) is computed for each point in the scene, by constructing a Darboux frame to each neighbour in the point's vicinity,

and accumulate its components over a fixed number of bins. Next, the FPFH descriptor of a point is constructed by adding the average SPFH histogram of the point's neighbours (albeit also weighted by distance to the point itself) to the point's own SPFH.

While many such descriptors have been shown to perform well at recognition tasks, one of their primary challenges is the presence of geometry unrelated to the queried shapes within the support volume of a descriptor, referred to as "clutter" [13]. Not every descriptor is equally resistant to the negative effects of clutter on matching performance.

One example of a descriptor that has been shown to resist clutter is the classic Spin Image (SI) proposed by Johnson et al. [18]. An SI is generated by projecting points uniformly sampled from a surface on to a rotating square plane whose side is on the axis of rotation. The plane is divided into square bins, which count the number of point samples projecting onto them, thus creating a 2D histogram. Similar surfaces will result in a high correlation between their Spin Images.

An extension to the Spin Image which is related to the descriptor proposed in this paper, is the Spin Contour descriptor proposed by Liang et al [20]. The authors post-process high resolution Spin Images by detecting edges between zero and nonzero histogram bins. The resulting outlines, called "Spin Contours", are used for shape detection. However, the Spin Contour can only be used to represent an object in its entirety, due to its inability to detect edges within a Spin Image. Moreover, due to the method's reliance on outlines, its clutter resistance is not expected to be good.

Another descriptor shown to be resistant to clutter is the 3D Shape Context (3DSC) proposed by Lowe et al [30]. 3DSC has a spherical support volume, which is subdivided into bins through horizontal and vertical cuts, as well as spheres placed within it. Point samples of the surrounding region intersecting each bin are subsequently accumulated, creating a histogram. 3DSC descriptors are compared using a Euclidean distance function.

The Radial Intersection Count Image (RICI) [34] is a descriptor aimed at shape matching in highly cluttered scenes. A set of three dimensional circles are defined with centers along the normal to a vertex and with varying radii. The number of intersections between each circle and the mesh surface is counted, resulting in a 2D histogram. This is similar to the arrangement of circles seen in Figure 9.1. Comparing exact changes in intersection counts from a circle to its neighbour can be used to determine correspondence between RICI descriptors. The authors also propose a distance function that is capable of largely disregarding clutter within the support region, and show that this results in better matching performance in heavily cluttered scenes.

9.3 Quick Intersection Count Change Image (QUICCI)

In contrast to the previously proposed RICI descriptor, which stores integers representing intersection counts, the QUICCI descriptor stores booleans representing changes in intersection counts. The differences between the two descriptors also propagate to their distance functions, which due to the different representations require them to be tailored specifically to each method.

Circles are arranged in layers, each layer containing circles of increasing radii by a constant amount increment. The distance between circle layers, and the radius increment between circles within a layer, are equal. One thus forms a cylindrical "grid", where the y-coordinate corresponds to a layer of circles, and the x-coordinate to a circle within that layer. These coordinates in turn can be used to create an image. A visualisation of this is

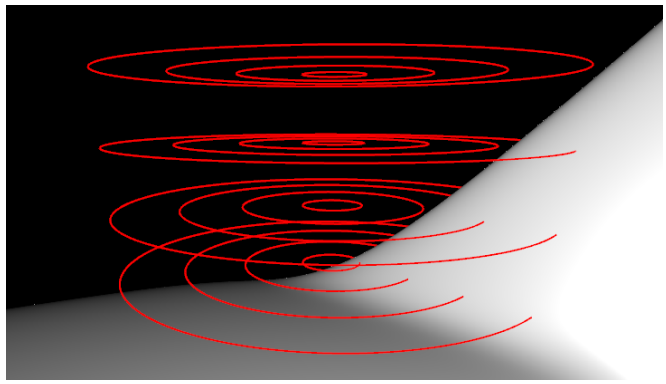


Figure 9.1: Visualisation of the "layers of circles" used for the construction of a 4x4 RICI, or a 3x4 QUICCI descriptor (pairs of circles are used, thus the effective width is one less than the number of circles per layer). The 4 layers containing 4 circles each can be seen in the image, some of which intersect with the object surface towards the right side. The circles combined form a cylindrical support volume.

shown in Figure 9.1.

The descriptor is constructed around an oriented point, consisting of a vertex and a normal, referred to as the Reference Vertex and Reference Normal for the remainder of this paper. The oriented point defines a three-dimensional line, called the Central Axis. All circles are orthogonal to the Reference Normal, and centred around the Central Axis. The Reference Vertex lies at the exact centre of the support region.

Computing a QUICCI descriptor for a Reference Vertex involves intersecting all circles with the mesh surface, and subsequently subtracting each circle's intersection count from that of its smaller neighbour in the same layer, as illustrated in Figure 9.2. If this difference is nonzero, the corresponding bit will be set to 1, else to 0. This implies that a layer with C circles will result in a QUICCI descriptor of $C-1$ bits.

The function for comparing two QUICCI descriptors is asymmetric, and distinguishes between a needle image (describing the shape that should be located) and a haystack image (describing any other shape to which a similarity score should be computed). Intersection count changes present in the needle image are characteristic to the shapes being queried, and this can be exploited by only including those specific bits in the distance computation. Due to the QUICCI image's tendency to be sparse, this excludes the majority of the image's bits from the distance computation, making it resistant to clutter (see Section 9.6.2). An algebraic representation of the distance function is shown in Equation 9.1.

$$D_{QUICCI}(I_n, I_h) = \sum_{r=0}^{N} \sum_{c=0}^{N} ((I_n[r,c] \oplus I_h[r,c]) \wedge I_n[r,c]) \tag{9.1}$$

Where I_n and I_h are the needle and haystack images, respectively, $I[r,c]$ denotes the bit at row r and column c of image I, and N denotes the QUICCI image width. A lower distance value indicates that the two images are more similar. The $\oplus$ and $\wedge$ operators denote the bitwise XOR and AND functions, respectively.

Constructing the QUICCI descriptor as a binary image has significant advantages. Many of the previously discussed local shape descriptors use 32-bit floating point or integer

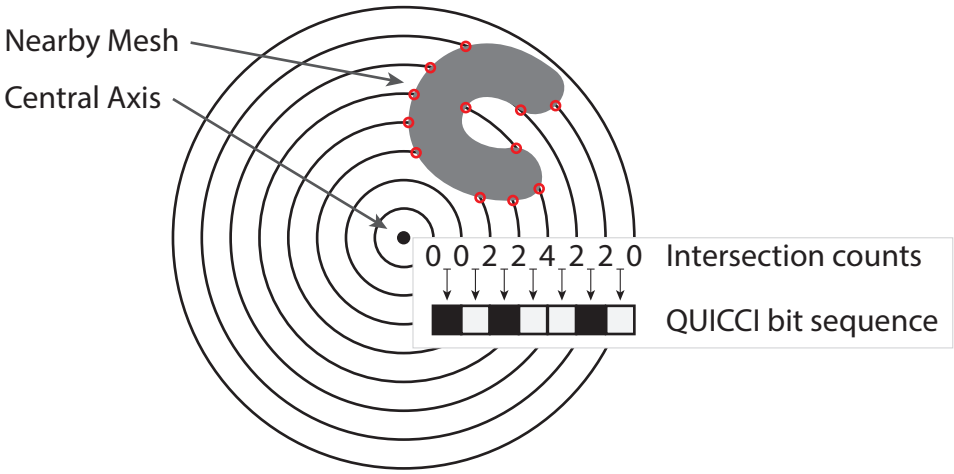


Figure 9.2: Visualisation of the construction of a single row of a QUICCI descriptor. First, intersections between circles with increasing radii and a mesh surface are counted (intersection points are indicated in red). Next, neighbouring intersection counts are compared. If they are different, the corresponding bit in the QUICCI image is set to 1 (white), otherwise to 0 (black).

representations. The QUICCI descriptor thus uses about an order of magnitude less memory. This smaller size means both less disk storage and significantly faster comparison rates, mainly due to the smaller memory bandwidth requirements.

Moreover, QUICCI descriptors can be constructed efficiently on the GPU due to its advantageous memory access patterns and bandwidth requirements. The intersection computation between the circles and the mesh surface is the most demanding part of this process, which can be done using the efficient algorithm presented in [34].

9.4 Hamming Tree

The Hamming Tree is introduced as a means for indexing bit strings of arbitrary length, such as QUICCI images, for the purpose of k-Nearest-Neighbour searches using the Hamming distance function [16] as a ranking metric. In this paper, the method is discussed and tested only on the proposed QUICCI descriptor, where the rows of the complete 2D image are concatenated to produce a 1D bit string that can be indexed and queried. However, the application of the tree is not limited to it and can be used for indexing arbitrary bit strings. For this reason the explanations in this section will use QUICCI images as an example, but the contents of the tree being indexed is referred to as “bit string” throughout the paper.

The observation central to the design of the tree is that the total set bit count of a bit string can be used to compute a lower bound of the Hamming distance between a given pair of bit strings. For example, two bit strings with 3 and 5 bits set, respectively, must have a Hamming distance of at least 2. This minimum distance can subsequently be used as a heuristic for navigating a tree, where the set bit count of consecutive, equally sized, substrings determines which next branch to pick. Each branch taken will place stricter requirements on the distribution of set bits within the string, increasing the probability a match is found.

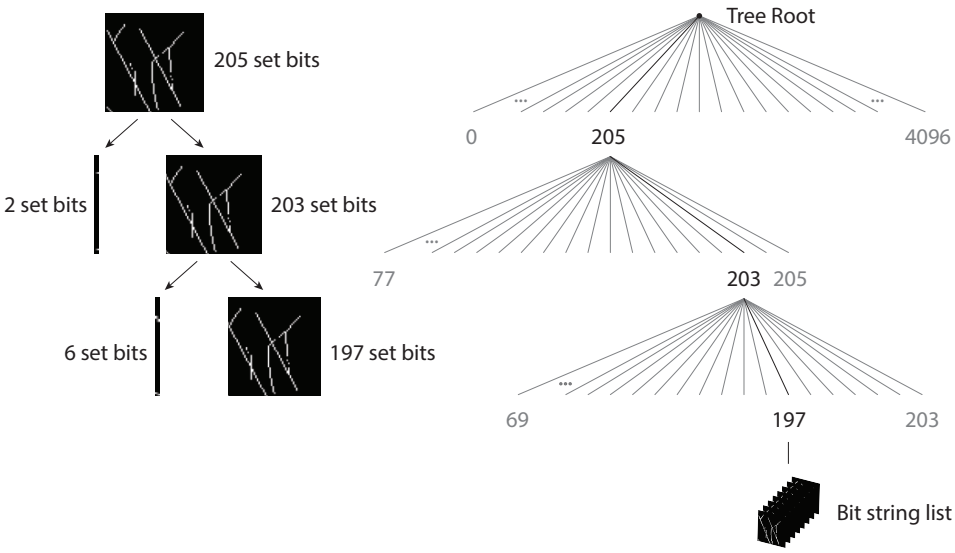


Figure 9.3: Visual representation of navigating a Hamming tree. On the left hand side, a 64×64 QUICCI image is shown, where two columns are removed at each step (128 bits). The right hand side shows the corresponding path in the tree.

9.4.1 Hamming Tree Construction

Construction of the tree is done by iteratively inserting bit strings, dynamically expanding the tree where necessary. It consists of two node types; internal nodes and leaf nodes. Internal nodes contain references to leaf nodes and other internal nodes. Leaf nodes in turn contain a list of bit strings. Initially, the tree consists of one root internal node, and one leaf node for each possible bit count. When the length of all bit strings being indexed is N, that implies the root node has $N+1$ children. For levels underneath the root node, the branching factor is at most the number of set bits corresponding to that node plus one.

The tree is navigated (both during insertion and querying) by iteratively cutting off a fixed number of bits from the front of the bit string. After removal, the number of set bits in the remaining string determines the next branch to take in the tree. This process has been visualised in Figure 9.3. While this approach does not place requirements on the exact positions of set bits, it aims to filter the indexed bit strings by those whose distribution of bits are similar to a given query string, thereby increasing the likelihood a relevant match is found.

Thus to insert a new bit string into a Hamming tree, one navigates down to the leaf node corresponding to the bit string. The new bit string is then inserted in the list of that leaf node. If afterwards the count of that list exceeds a constant threshold, the leaf node is replaced by an internal node and the list of bit strings is distributed among the lists of its new child leaf nodes.

9.4.2 Querying the Hamming Tree

Our algorithm for querying a Hamming tree takes in a needle bit string, a Hamming tree, and the maximum number of search results that should be returned as input, and returns a list of bit strings whose Hamming distance is closest to the provided needle string.

It first attempts to locate an exact match for the needle string and subsequently widens the search so as to include the nearest matches within the requested search result limit count.

Internally, the algorithm maintains a list and a priority queue. The list stores the best search results that have been found up to that point and its size is limited by the search result limit parameter. The priority queue contains unvisited internal tree nodes (initialised with the tree's root node), sorted by the minimum possible distance between the needle and all possible descriptors in the subtree rooted at a node.

The query algorithm visits one internal node at a time, until the node at the front of the unvisited node queue (with the lowest minimum distance) has a greater distance than the worst entry in the search result list, or the unvisited node queue is empty (a similar strategy to the one adopted by Chappell et al. [7] [14]). This process is illustrated in Figure 9.4.

Visiting an internal node involves iterating over the node's outgoing edges. When there is a bit string list at the end of that edge, compute the Hamming distance between the needle and haystack strings contained within. Any strings which improve the list of search results are inserted into the search result list. When the outgoing edge points to an internal node, the minimum distance to that node is computed (as a function of the needle string and the node's position in the tree), and if that minimum distance is lower than the current worst entry in the search result list, it is inserted in the unvisited node queue. This condition prevents the unvisited node queue from growing indefinitely, and excludes bit strings that are certainly not going to be part of the search results anyway.

The Hamming Tree is capable of efficiently locating all bit strings which have low distance scores relative to a needle string. However, as the distances get larger than a few bit flips, the number of permutations, and thus nodes that need to be visited, increases to such a degree that it may be necessary to visit a significant part of the tree before the algorithm can ensure that no better search results exist. It is therefore advisable to set a distance threshold that is used in conjunction with the worst search result score, and set this threshold as low as possible when querying a Hamming tree.

In terms of complexity, in the worst case, a completely unbalanced tree is in effect equivalent to a linked list. Insertion therefore has constant complexity ($O(1)$), while search is linear ($O(n)$).

9.5 Weighted Hamming Distance

With respect to QUICCI, for a proportion of needle images, the previously proposed indexing strategy is capable of locating QUICCI images containing similar shapes as to the ones requested in the needle. However, it is not universally applicable for this task. Most notably, needle images which are close to fully saturated with set (1) or unset (0) bits are likely to yield search results containing irrelevant shapes. An example of such a needle image and the corresponding search results can be seen in Figure 9.5. The needle image shown in the Figure is also a good example of a local shape query of the kind described in Section 9.1.

The cause of this problem is that the Hamming distance considers each bit to be equivalent in importance. However, when the number of set bits in a needle image is low, for the purpose of shape retrieval, it is more important that the bits set in the needle

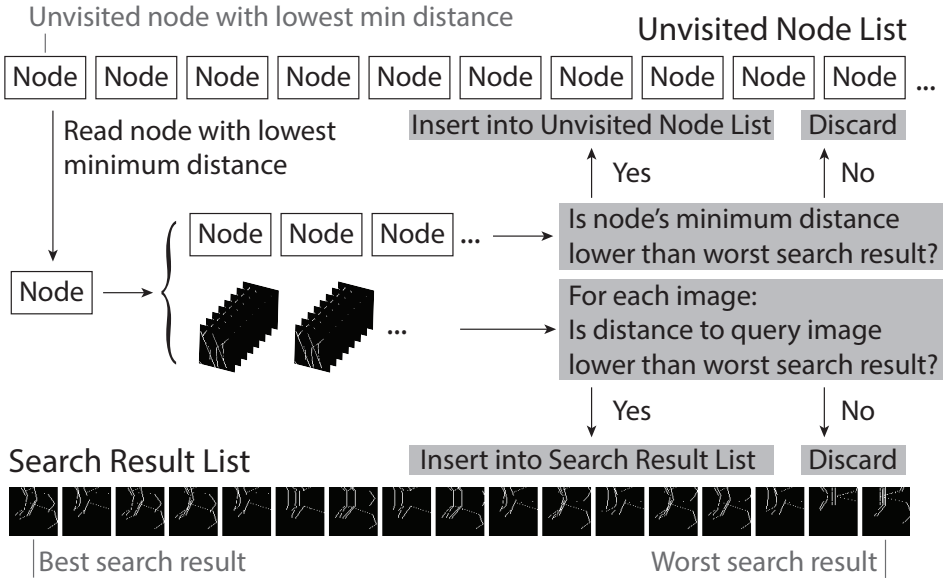


Figure 9.4: Visualisation of the Hamming tree query algorithm. At each iteration, the contents of the node with the lowest minimum distance in the unvisited node queue, which consists of child internal nodes and bit string lists, is inserted into the univisited node list and search result list, iff there is a possibility that they can potentially improve the search results.

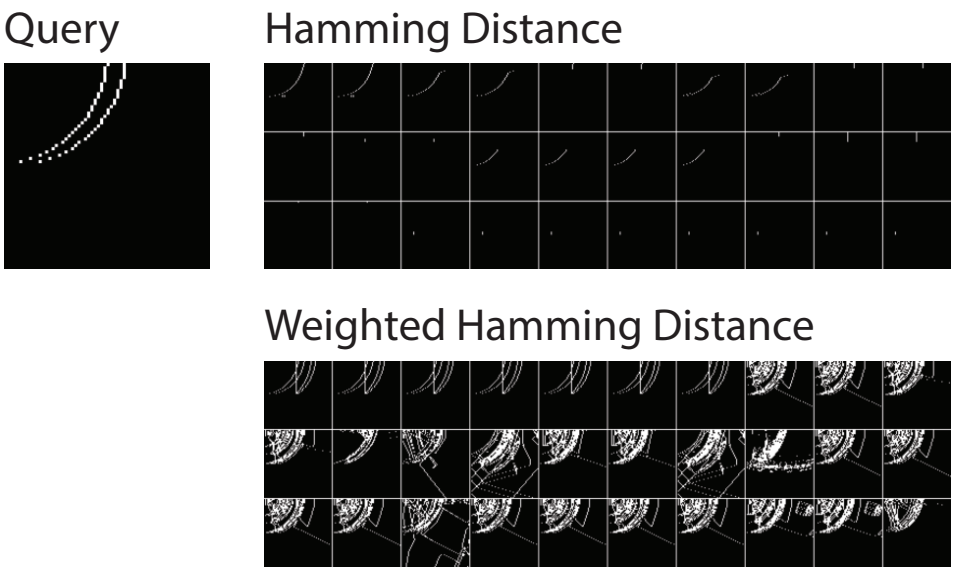


Figure 9.5: Top 30 search results for the shown needle image when search results are ranked using Hamming distance (above) and the proposed Weighted Hamming (below) distance functions.

are also set in the haystack image than unset needle bits being unset in the haystack image. We therefore observe that the lower the number of set bits in the needle image, the more important it is for these bits to be set in a haystack image. The opposite also holds true for needle images nearly saturated with set bits.

With this observation, we propose an alternate distance function, called "Weighted Hamming", which can be used for ranking QUICCI search results. The function broadly resembles Hamming distance, but the distance cost for the two types of bit mismatches (incorrectly set and incorrectly unset) are weighted differently, depending on the proportion of set to unset bits in the needle image. The definition of this function is given in Equation 9.2.

$$\begin{aligned} D_{WH}(I_n, I_h) = &\frac{\sum_{r=0}^{N}\sum_{c=0}^{N}(I_n[r,c](1-I_h[r,c]))}{max(\sum_{r=0}^{N}\sum_{c=0}^{N}I_n[r,c],1)} \\ &+\frac{\sum_{r=0}^{N}\sum_{c=0}^{N}((1-I_n[r,c])I_h[r,c])}{max(N-\sum_{r=0}^{N}\sum_{c=0}^{N}I_n[r,c],1)} \end{aligned} \tag{9.2}$$

Here I_n and I_h are respectively the needle and haystack images being compared, $I[r,c]$ represents the bit at row r and column c of a needle or haystack image I, and N is the width of the QUICCI image in bits.

It's worth noting that removing the denominators of the fractions in the Equation makes it equivalent to the regular Hamming distance function. Moreover, the weighted Hamming distance function is effectively a hybrid between Hamming distance and the clutter-resistant QUICCI distance function used for locating shapes in clutter heavy scenes shown in Equation 9.1. When the second term in Equation 9.2 is nullified, its ranking becomes equivalent to the clutter-resistant distance function. However, the removal of the second term also means there is a possibility for false positives, where a high variation in intersection counts may cause the desired needle bits to be set accidentally in a given haystack image, even though the surroundings of the corresponding haystack point does not actually contain the shapes requested by the query. We therefore consider the function given in Equation 9.2 to be more suitable for retrieval purposes. An in-depth evaluation of this claim is done in Section 9.6.5.

9.5.1 Indexing for Weighted Hamming

The remainder of this section is dedicated to the construction of an index that allows querying using the presented weighted Hamming distance function, and a discussion of insights and some negative results that were obtained in the process. It is assumed that needle images will generally have a low number of bits set, otherwise a regular Hamming tree is likely a more suitable solution.

A good indexing strategy is closely tied to the distance function used. For weighted Hamming, this means that since the function primarily looks for the bits which are set in the needle image, the indexing structure should focus on discovering those in haystack images. One observation that can be made for QUICCI images is that edges of 3D geometry tend to create line or curve responses in QUICCIs. Thus, groups of bits that are in close proximity to one another in a QUICCI image are likely to be related.

There are a number of ways in which this can be exploited, however, a problem is the exponential increase of permutations in the possible arrangements of a group of bits. However, it can be observed that due to the image's construction, these lines have a tendency

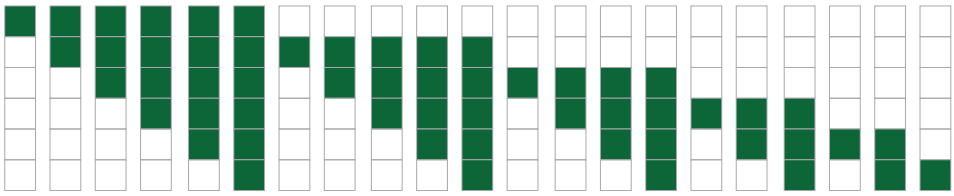


Figure 9.6: All 21 possible sequences of consecutively set bits in a single column of a 6 bit high image. White represents unset bits, whereas those marked green are set. A single column may contain multiple (albeit non-overlapping and separated by at least one unset bit) such sequences.

to be vertical. This allows a relatively simplistic approach where permutation counts remain within reasonable limits.

The indexing algorithm detects segments of consecutively set bits within each column of the QUICCI image. Vertical bit sequences are advantageous due to their aforementioned common occurrence, and limited number of permutations in which they can occur within a given column.

For every possible bit sequence, an inverted list is created of all images which contain that exact bit sequence (with the same starting position and length). As an example, all possible consecutive bit sequences that can be found in an image that is 6 bits high are shown in Figure 9.6.

Querying involves combining the contents of all lists whose bit sequence overlaps by at least 1 bit with the given needle image. Since the total number of set bits for each image is also stored with each entry in the inverted lists, the exact weighted Hamming distance can be computed and used to rank results.

Unfortunately, the major issue of this approach is also the main advantage of the Weighted Hamming function; the value of matching set bits between the needle and haystack images far outweighs the cost of mismatched unset needle bits. The query algorithm must therefore consider all haystack bit sequences that overlap by at least one bit with the needle image, and cannot preemptively disregard entries. This causes many inverted lists to be searched, incurring long query execution times (typically resulting in a cost similar to a sequential search). Furthermore, the storage requirements of this index are high due to the inverted list construction.

9.6 Evaluation

All experiments involving time measurements in this section were executed on the same hardware. For CPU implementations, an Intel Xeon Platinum 8168 was used, and GPU kernels were executed on an NVidia Tesla V100 SXM3. The remainder of the results were in part gathered on the NTNU IDUN Cluster [50] computing cluster.

9.6.1 Hamming Tree Search Acceleration

The Hamming Tree was implemented in C++. Nodes and image lists stored on disk are compressed using the LZMA2 algorithm [9]. This was selected after empirically testing a number of state-of-the-art compression methods; LZMA2 yielded good compression ratios and speed for QUICCI images.

A Hamming Tree was constructed over the first 12,500 objects of the SHREC2017 dataset [31], which resulted in a total of 828.5 million QUICCI images. The resolution of the QUICCI images was set to 64x64 bits, and the support radius to 0.3 (for consistency all objects are translated and scaled to fit into a unit sphere prior to QUICCI generation). The number of bits removed at each level of the Hamming tree was set to 128 bits, or 2 image columns. The threshold at which leaf nodes (image lists) are split was set to 256 images, which balances index compactness with granularity. 1000 queries were executed on the constructed Hamming Tree. The needle images were randomly selected from the entire SHREC2017 QUICCI dataset (51,109 objects).

While the algorithm can to some extent be parallelised, the testing was done using a single threaded implementation. The time from the start of each query to when all nearest neighbours up to each Hamming distance were located using the Hamming Tree was measured. These timings were averaged across the 1000 queries for every value of Hamming distance. For comparison, the cost of performing a linear search through the set of QUICCI images is also reported; this has a constant time as it has to traverse the entire list of 828.5 million QUICCI images. The results are shown in Figure 9.7.

The Figure shows that, particularly for small bit distances, the Hamming Tree is very effective at reducing query times. This makes it a good candidate for neighbour discovery when using LSH-derived methods. Query execution times are highly dependent on the presence of close neighbours to a given needle image and the number of search results requested, but generally follow the timing pattern shown.

It is worth noting here that the average number of set bits per QUICCI image in the created dataset was measured to be 610.1. When the Hamming Tree reaches parity with a linear search at a Hamming distance of about 800, the relevance of the search results is not expected to be high. Moreover, the vast majority of the algorithm's execution time is spent on reading and decompressing files stored on disk. This applies to both the sequential and the indexed query implementations. Chappell et al. [14] performed all searches in memory, which complicates direct comparison of the two implementations.

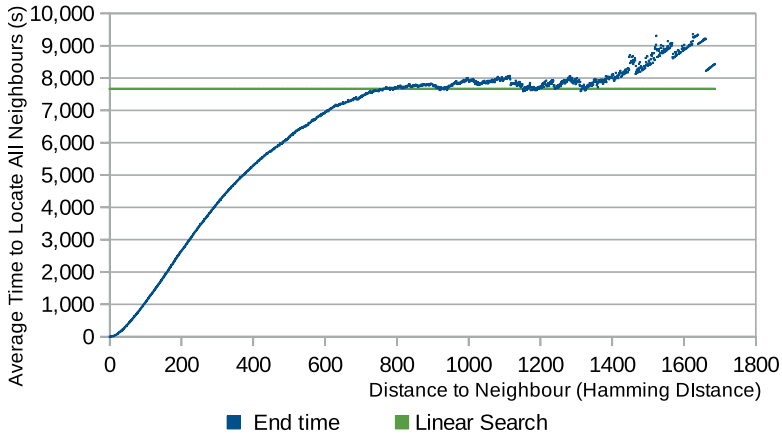


Figure 9.7: Chart showing the average time in seconds required to locate all neighbours up to a given Hamming distance for a Hamming Tree and a linear search.

9.6.2 QUICCI Descriptiveness and Clutter Resistance

The descriptiveness and clutter resistance of the QUICCI descriptor is evaluated using the Clutterbox experiment proposed in [34], and used to compare the performance of QUICCI against the RICI, SI, 3DSC, and FPFH descriptors. The RICI descriptor was chosen due to its similarity to the QUICCI descriptor, SI and 3DSC for being the most referenced methods known for their clutter resistance [13], and FPFH is an example of a popular descriptor.

The experiment aims to quantify the clutter resistance of the descriptors by measuring the response of a tested descriptor to increasing levels of clutter. The experiment is executed a large number of times by varying objects and their transformations, in order to provide robust results independent of object type.

The Clutterbox experiment is performed using the following steps:

1. Define a cube volume whose edge size is s.
2. From a large object collection, draw n objects at random.
3. Fit each of the randomly chosen objects in a unit sphere centred around the origin.
4. From the selected objects, select one at random to be what is referred to as the "reference object".
5. Compute a descriptor for each unique vertex in the reference object, thereby creating the set of reference descriptors $\{RD\}$.
6. Iterating over the list of chosen objects in a random order, though always starting with the reference object, do the following for each:
    a) Place the object at a randomly chosen orientation and position whose bounding sphere fits entirely within the cube volume.
    b) Compute a descriptor for each unique vertex present in the combined mesh present inside the cube volume. The result is a set of cluttered descriptors $\{CD\}$.
    c) For each $d \in \{RD\}$, compute a list of distances for each $c \in \{CD\}$, and sort it. Locate the corresponding cluttered descriptor in this list, and store its rank in the list ($0 \leq rank \leq |\{CD\}| - 1$). Note that lower ranks are better, and rank 0 is the front/top of the list.
    d) From the computed list ranks, construct a histogram where bin i stores the number of occurrences where the corresponding cluttered descriptor was found at rank i.

The procedure is repeated for each tested descriptor, where all randomly selected values are kept constant. The result of the experiment is therefore a list of histograms, one for each level of clutter. While performing the experiment, the parameters listed in Table 9.1 were used.

Parameter	Value
Clutterbox size	$s = 3$
Object counts	$n = 1$, $n = 5$, $n = 10$
Support radius (all descriptors)	$r = 0.3$ [1]
QUICCI resolution	63x64 bits [2]
RICI / SI resolution	64x64 pixels
SI support angle	180° (disabled) [3]
3DSC minimum support radius	$r_{min} = 0.048$ [4]
3DSC bin dimensions	$J = 15$, $K = 11$, $L = 12$ [5]
FPFH bins per feature	11 [6]
Mesh sampling resolution	10 point samples per triangle in mesh [7]

Table 9.1: Parameters that were used during the evaluation of the different methods.

The clutterbox experiment was executed 1,500 times on objects from the SHREC2017 dataset [31], which contains a total of 51,162 triangle meshes. An exception has been made to the FPFH descriptor, which was only executed 500 times due to excessive execution times. For clarity, all curves of this descriptor have been stretched for easier comparison against other descriptors.

To visualise the resulting histograms, the fraction of search results correctly being ranked as the best match for each uncluttered reference descriptor (at rank 0) was computed for each descriptor and clutter object count. The produced measurements exhibited a high degree of noise, which did not allow the data to be displayed in a comprehensive manner. Each sequence was therefore sorted individually to produce a monotonically increasing curve, for the purpose of chart readability. The result of this is shown in Figure 9.8. The clutterbox experiment was implemented in C++, and the tested descriptors have been implemented on the GPU using CUDA 10.1.

The results show that the QUICCI descriptor outperforms previous work in terms of resistance to clutter. However, it is also advantageous to investigate the relationship between the degree of clutter present in the support radius, and the resulting matching performance of each descriptor. To this end, a set of heatmaps was created from the search results of $n = 5$ (4 added clutter objects), showing this relationship. This result set corresponds to a total of 70.0M needle descriptors and associated search results. These can be seen in Figure 9.9. The vertical axis in these heatmaps represents the rank where the correct search result was found (lower rank is better), and the horizontal axis denotes the fraction of clutter (the proportion of surface area in the descriptor's support region not belonging to the object being queried). Higher fractions of clutter generally imply greater difficulty for a given descriptor to correctly identify the correct matching vertex.

[1] Note that all objects are first fit inside a unit sphere.
[2] Corresponds to the equivalent RICI resolution.
[3] We have not found evidence for its claimed benefits.
[4] Proportionally equivalent to previous work.
[5] Equivalent to previous work [13] [11].
[6] Equivalent to previous work.
[7] SI, 3DSC, and FPFH require point clouds, necessitating uniform sampling.

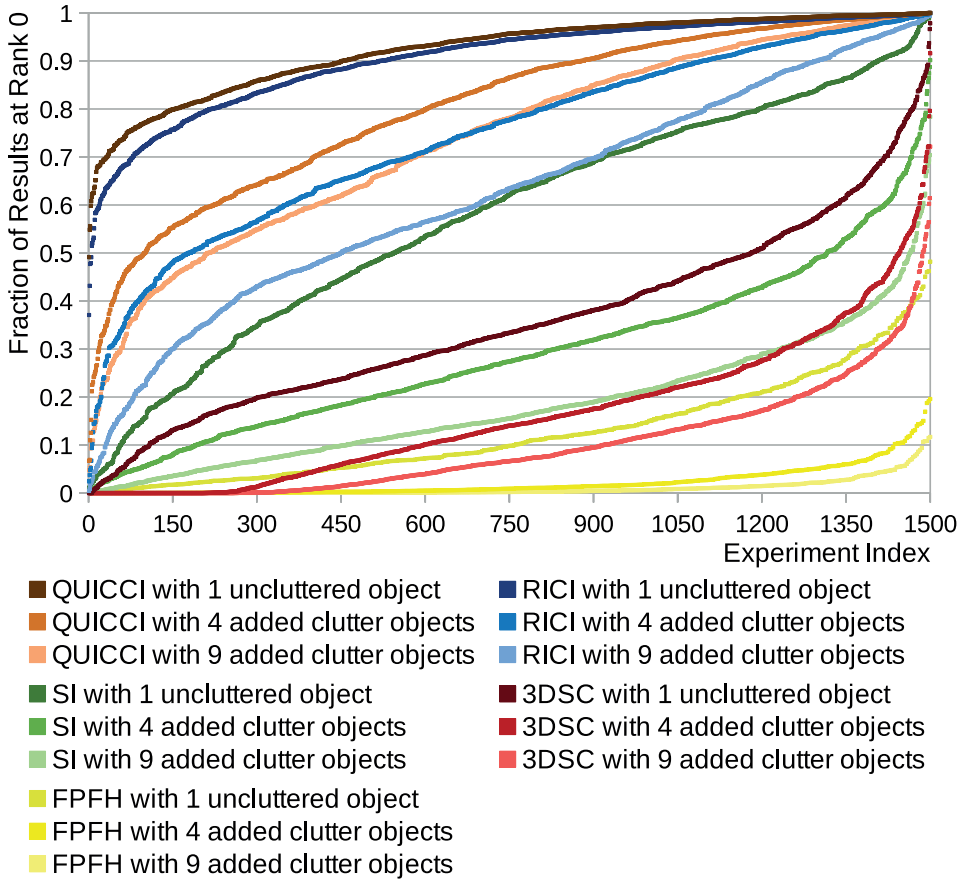


Figure 9.8: Fraction of search results that were correctly ranked at the top of the list of search results for each tested descriptor and added clutter object count. Each sequence has been sorted individually to create monotonically increasing curves.

From these heatmaps it can be seen that the QUICCI descriptor has similar characteristics to RICI in terms of clutter resistance, albeit with slightly better performance. This reflects the observations from the results of Figure 9.8. One possible reason why QUICCI outperforms RICI, is that RICI compares absolute intersection counts, while QUICCI only looks at differences. In the presence of clutter, these absolute values may become noisy, and consequently reduce matching performance.

The FPFH heatmap has a distinct appearance relative to the other methods, which can be attributed to its poor matching performance, particularly in cluttered scenes. The heatmap only counts results that appeared in the top 256 ranks, and shows that even in situations with low fractions of clutter, very few results end up within the top 256 ranks shown in the image.

9.6.3 QUIICI Comparison Rate

The number of QUICCI image comparisons performed per second was also measured during the experiments and compared to the other descriptors. In similar fashion to Figure 9.8,

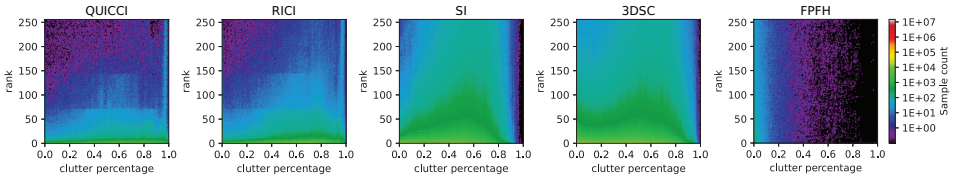


Figure 9.9: Heatmaps showing the relationship between varying degrees of clutter and each descriptor's matching performance. The horizontal axis represents fraction of area in the support region not part of the matched object, while the vertical axis denotes ranks in the list of search results (where lower is better).

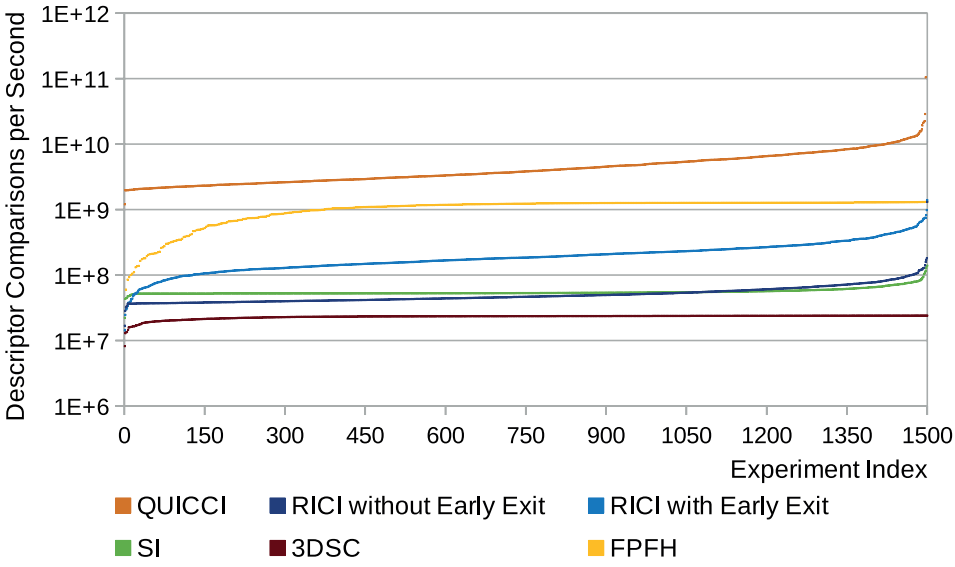


Figure 9.10: Comparison of the number of descriptor pairs each method can compare per second. For readability, each sequence has been sorted individually to create monotonically increasing curves.

there was a degree of noise present in the results, and sorting each shown curve individually allowed for the best chart readability. The results are shown in Figure 9.10. As can be seen, many billions of comparisons can be done per second and, on average, outperforms previous work by over an order of magnitude. This is due to the binary nature of QUICCI.

For the RICI measurements, two variants of the distance function are tested. When an upper distance bound is known, as is often the case in retrieval applications, distance computation can cease early as this value can only grow. Results with early exit disabled serve as a baseline execution time, whereas those with the early exit enabled represent a best case. While this early exit could also be implemented for QUICCI images, it is not expected to improve performance much, if at all, due to the additional instruction overhead. Instruction counts are more relevant for QUICCI than RICI, as many QUICCI bits can be compared with a single bitwise instruction, whereas RICI compares each pixel individually.

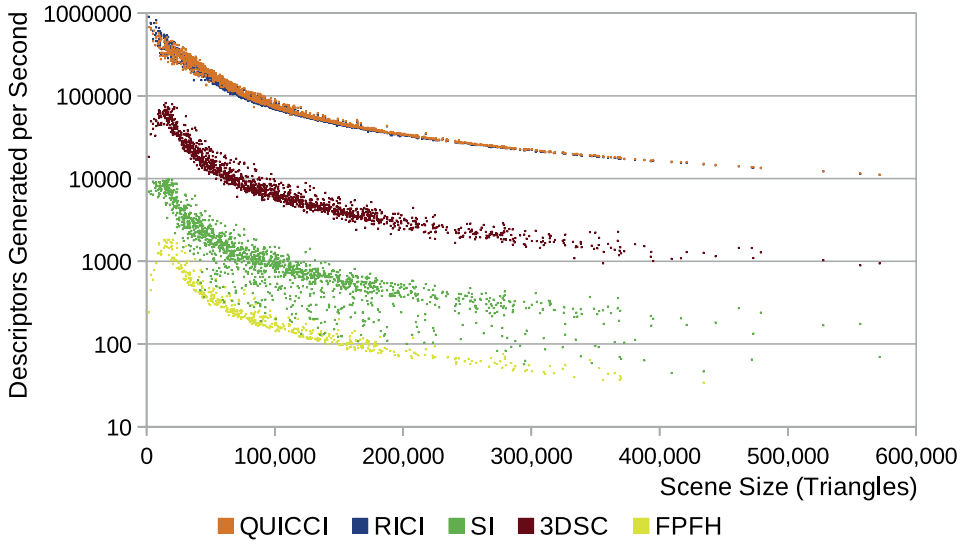


Figure 9.11: A plot showing the relationship between scene size measured in triangles and the rate at which descriptors are computed per second.

9.6.4 QUICCI Generation Rate

Finally, the rate at which the tested descriptors are computed was measured during the performed experiments. The results are shown in Figure 9.11. The chart shows that QUICCI and RICI descriptors can be generated at similar speeds, which is about an order of magnitude better than the next best descriptor.

9.6.5 Weighted Hamming

An experiment was constructed in order to quantify our claim that the Weighted Hamming distance function is superior for retrieval purposes of QUICCI images over the clutter resistant distance and Hamming distance functions. The premise of this experiment is to evaluate the distance values returned by each distance function. We compare two different settings: where the surface points being compared have distinctly different support regions and where the support regions are quite similar.

The values returned by the distance functions where object surfaces are distinctly different gives insight into the range of distances that can be expected to be returned by the distance function under "nominal" conditions.

With that background, one can then investigate what happens to the distance values when point pairs have varying degrees of similarity. In order to obtain quantitative results, it must be possible to generate these varying degrees of similarity automatically [8]. It is possible to simulate variations in geometric similarity through the addition of geometry, whose shape does not necessarily matter. In the devised experiment, spheres are added touching on randomly sampled points on the object's surface. An example of an object with

[8]There exist various ways of generating variants of similar shapes, notably those utilising shape grammars [23] [15]. However, the complexity of constructing these shape grammars tends to be high, while the variety of local surfaces produced is often low due to the reuse of a limited set of "aphabet" shapes.

spheres added to its surface in this manner is shown in Figure 9.12.

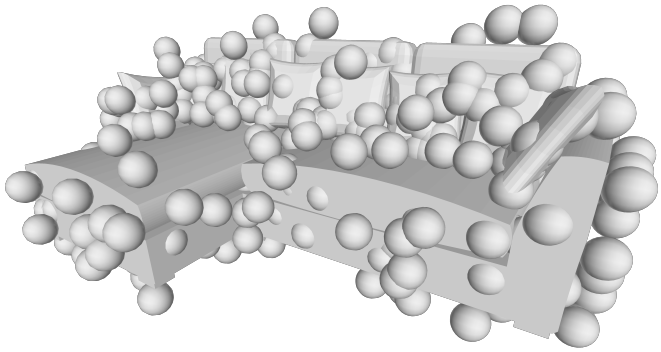


Figure 9.12: A visualisation of an object on which 500 spheres have been placed (the highest number used in the described experiment).

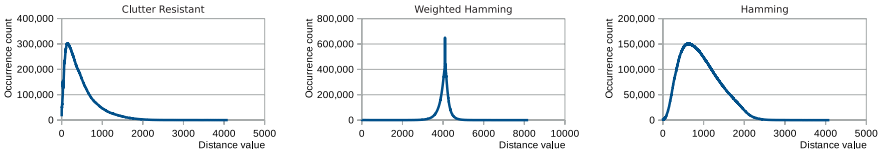
For computing distance values under "nominal" conditions, two objects were selected from the (same as previously used) SHREC2017 dataset [31] at random, and scaled to fit within a unit sphere. For each unique vertex in each object, a QUICCI descriptor was computed with the same generation parameters as in Section 9.6.2. Each pair of QUICCI descriptors corresponding to vertices with the same index[9] across the 2 objects (which have a random degree of similarity) was used to compute the distance value for each of the 3 distance functions. These values were used to construct the histograms of Figure 9.13a. This process was repeated for 10,000 object pairs, generating 176.2M image pairs.

The next step is to check the stability of the distance functions across the same object vertices as the environment of the vertices is changed. To this end, a random object from the SHREC2017 dataset is selected and fitted within a unit sphere. For each object vertex, a QUICCI descriptor is computed. Next, 10 random points on the object's surface are chosen and normal vectors are computed for these points by interpolation. At each of these points, a sphere of radius 0.05 units is placed such that it touches the selected sample point. This is achieved by displacing the sphere's origin by its radius along the point's normal. After each step of adding 10 spheres (up to a limit of 500 spheres), QUICCI descriptors are computed for the vertices of the original object. Distance values are then computed for each of the 3 distance functions between corresponding vertex QUICCI descriptors of the original and modified objects. After repeating this experiment for 1000 random objects from SHREC2017, histograms of the combined distance values are computed (from a total of 26.79M QUICCI images), see Figure 9.13b.

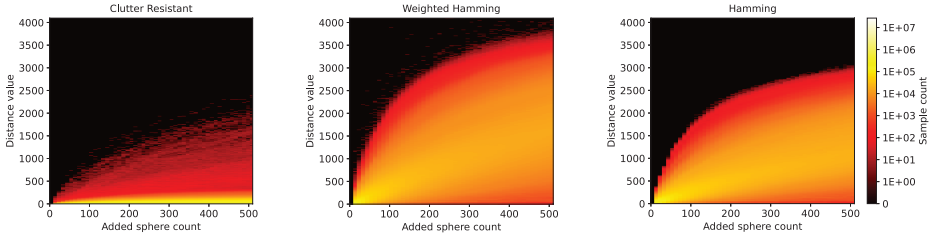
A good distance function should clearly discriminate between relevant and irrelevant descriptors with respect to a query. For the presented experiment, objects with fewer spheres applied to their surface should be considered closer to their original (unmodified) version by a given distance function. As can be seen from Figure 9.13b, this is indeed the case; all tested distance functions return on average lower scores for objects with fewer spheres.

However, the performance of these distance functions varies when it comes to their ability to discount images not relevant to the needle. For example, when comparing results for Hamming Distance, in Figure 9.13a (right) and 9.13b (right) it can be observed that

[9]The number of generated images is bounded by the object with the fewest vertices.



(a) Distribution of measured distance values under nominal conditions for each tested distance function.



(b) Visualisation of all 51 distance function response histograms produced when comparing object pairs having varying degrees of surface similarity. Each column represents a single histogram similar to the one shown in Figure 9.13a, corresponding to the distance value distribution when the modified object has a set number of spheres applied to its surface.

Figure 9.13: Visualisation of the histograms of distance function responses that were obtained as part of the evaluation of these functions under both nominal and similar surface conditions.

the histograms (columns of Figure 9.13b (right)) quickly approach the histogram of Figure 9.13a (right) for random vertices.

At a glance, the clutter resistant distance function appears to have the same issue. However, closer inspection of the data shows that the cause of this behaviour is the commonly low number of set bits present in needle images. As the distance function is bounded by the number of set bits present in the image (with the exception of cases where none are set), computed distances have a tendency to be low. However, the vast majority of scores ends up being the highest possible score that the distance function allows for that particular needle image.

While this behaviour is effective at discerning close matches (as demonstrated in Section 9.6.2), it is less advantageous for retrieval purposes, where more granularity is desirable for the purpose of ranking search results. The Weighted Hamming distance function is the one of the three tested functions which is the most capable of this. Moreover, of the three, it is also the one which shows the clearest separation between distances of matching surfaces relative to distances measured under 'nominal' conditions. A significant amount of variation can be applied before the range of computed scores reaches the same territory as the one shown in the nominal occurrence chart above. Moreover, under these circumstances only a small fraction of results overlaps with this nominal range. For these reasons we conclude that, among the tested distance measures, the Weighted Hamming distance is most suitable for the purposes of retrieval.

9.7 Issues with FPFH

Some issues were discovered while testing the well-known FPFH descriptor. The most notable of which pertains to the equation used to construct the FPFH during the second stage of generation, listed in Equation 4 in [41], reproduced here as Equation 9.3.

$$FPFH(p) = SPF(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_k}\cdot SPF(p_k) \tag{9.3}$$

The Equation computes an FPFH descriptor at point p, using a set of SPF histograms that were computed for each point in a previous step, and includes all k neighbours present within the support radius.

Of specific interest here is the distance weighting component $\frac{1}{\omega_k}$, which discounts the contribution of each point neighbour's SPF histogram by the distance to the point p for which the $FPFH$ histogram is computed. The issue is that, as this distance is not normalised, the weighting between the left ($SPF(p)$) and right ($\sum_{i=1}^{k}\frac{1}{\omega_k}\cdot SPF(p_k)$) terms of the equation depend on the scale of the object.

Also worth noting is that the original FPFH paper does not give a distance function to compare descriptors. We have used Pearson correlation in our implementation.

Finally, one detail that we noted in the currently available GPU implementation of Point Cloud Library [29] is that the aforementioned weighted distance factor uses the squared distance as a the value of ω_k, which deviates from the original paper.

9.8 Conclusion

This paper addressed the problem of querying by local shape. A new binary descriptor, QUICCI, is proposed which is robust to clutter, highly descriptive and quite small in size. To overcome the cost of searching the huge number of such descriptors that result from an object collection, a binary image indexing scheme, the Hamming Tree, was proposed which can significantly accelerate searching, especially for small Hamming distances. The effectiveness of an indexing structure is, however, highly dependent on the distance function used. The Weighted Hamming distance function is also proposed, which can be used to rank QUICCI descriptors in a retrieval setting.

9.9 Acknowledgements

The authors would like to thank the HPC-Lab leader and PI behind the "Tensor-GPU" project, Prof. Anne C. Elster, for access to the Nvidia DGX-2 system used in the experiments performed as part of this paper. Additionally, the authors would like to thank the IDUN cluster at NTNU for the provision of additional computing resources.

9.10 References

[1] Abdullah N. Arslan and Ömer Eğecioğlu. Dictionary look-up within small edit distance. *International Journal of Foundations of Computer Science*, 15(1):57–71, 2004.

[2] Sergey Brin. Near neighbor search in large metric spaces. *very large data bases*, pages 574–584, 1995.

[3] Gerth Stølting Brodal and Leszek Gasieniec. Approximate dictionary queries. In *CPM '96 Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching*, pages 65–74, 1996.

[4] Andrei Z. Broder. Identifying and filtering near-duplicate documents. *combinatorial pattern matching*, pages 1–10, 2000.

[5] A.Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29, 1997.

[6] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: binary robust independent elementary features. In *ECCV'10 Proceedings of the 11th European conference on Computer vision: Part IV*, volume 6314, pages 778–792, 2010.

[7] Timothy Chappell, Shlomo Geva, Anthony Nguyen, and Guido Zuccon. Efficient top-k retrieval with signatures. In *Proceedings of the 18th Australasian Document Computing Symposium*, ADCS '13, page 10–17. Association for Computing Machinery, Dec 2013.

[8] Timothy Chappell, Shlomo Geva, and Guido Zuccon. Approximate nearest-neighbour search with inverted signature slice lists. *european conference on information retrieval*, pages 147–158, 2015.

[9] Igor Pavlov Conor McCarthy. Fast lzma2 library, 2018.

[10] Alex Flint, Anthony Dick, and Anton Van Den Hengel. Thrift: Local 3d structure recognition. In *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)*, pages 182–188. IEEE, 2007.

[11] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *European conference on computer vision*, pages 224–237. Springer, 2004.

[12] A. Giachetti and C. Lovato. Radial symmetry detection and shape characterization with the multiscale area projection transform. *Computer Graphics Forum*, 31(5):1669–1678, 2012.

[13] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89, 2016.

[14] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3d local surface description and object recognition. *International journal of computer vision*, 105(1):63–86, 2013.

[15] Jean-David Génevaux, Éric Galin, Eric Guérin, Adrien Peytavie, and Bedrich Benes. Terrain generation using procedural models based on hydrology. *international conference on computer graphics and interactive techniques*, 32(4):143, 2013.

[16] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, Apr 1950.

[17] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, Jul 2012.

[18] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[19] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, 2011.

[20] Luming Liang, Andrzej Szymczak, and Mingqiang Wei. Geodesic spin contour for partial near-isometric matching. *Computers & Graphics*, 46:156–171, 2015.

[21] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[22] Moritz G. Maaí and Johannes Nowak. Text indexing with errors. *Journal of Discrete Algorithms*, 5(4):662–681, 2007.

[23] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. *international conference on computer graphics and interactive techniques*, 25(3):614–623, 2006.

[24] Mohammad Norouzi, Ali Punjani, and David J. Fleet. Fast search in hamming space with multi-index hashing. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3108–3115, 2012.

[25] Sai Manoj Prakhya, Bingbing Liu, and Weisi Lin. B-shot: A binary feature descriptor for fast and efficient keypoint matching on 3d point clouds. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1929–1934. IEEE, 2015.

[26] E. M. Reina, K. Q. Pu, and F. Z. Qureshi. An index structure for fast range search in hamming space. In *2017 14th Conference on Computer and Robot Vision (CRV)*, pages 8–15, 2017.

[27] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[28] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[29] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[30] Caitlin Sadowski and Greg Levin. Simhash: Hash-based similarity detection. *Technical report, Google*, 2007.

[31] Manolis Savva, Fisher Yu, Hao Su, Asako Kanezaki, Takahiko Furuya, Ryutarou Ohbuchi, Zhichao Zhou, Rui Yu, Song Bai, Xiang Bai, Masaki Aono, Atsushi Tatsuma, S. Thermos, A. Axenopoulos, G. Th. Papadopoulos, P. Daras, Xiao Deng, Lian Zhouhui, Bo Li, Henry Johan, Yijuan Lu, and Sanjeev. Mk. Shrec'17 track large-scale 3d shape retrieval from shapenet core55. In *Proceedings of the Eurographics Workshop on 3D Object Retrieval*, 2017.

[32] Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure, 2019.

[33] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010.

[34] Bart Iver van Blokland and Theoharis Theoharis. Radial intersection count image: A clutter resistant 3d shape descriptor. *Computers & Graphics*, 91:118 – 128, 2020.

Chapter 10

Paper E - Partial 3D Object Retrieval using Local Binary QUICCI Descriptors and Dissimilarity Tree Indexing

Authors

Bart Iver van Blokland and Theoharis Theoharis

Published in

Computers & Graphics Volume 100, Pages 32-42, 2021, Elsevier

Awards

This paper has been awarded the *Graphics Replicability Stamp* by the Graphics Replicability Stamp Initiative (GRSI).

Partial 3D Object Retrieval using Local Binary QUICCI Descriptors and Dissimilarity Tree Indexing

Bart Iver van Blokland[1], Theoharis Theoharis[1]
1) Norwegian University of Science and Technology, Norway

Abstract

A complete pipeline is presented for accurate and efficient partial 3D object retrieval based on Quick Intersection Count Change Image (QUICCI) binary local descriptors and a novel indexing tree. It is shown how a modification to the QUICCI query descriptor makes it ideal for partial retrieval. An indexing structure called Dissimilarity Tree is proposed which can significantly accelerate searching the large space of local descriptors; this is applicable to QUICCI and other binary descriptors. The index exploits the distribution of bits within descriptors for efficient retrieval. The retrieval pipeline is tested on the artificial part of SHREC'16 dataset with near-ideal retrieval results.

10.1 Introduction

There exist many circumstances in which it is desirable to determine which larger object a smaller surface patch belongs to; occlusions and missing parts can result in this problem. This problem is known as *Partial 3D Object Retrieval*, and a number of methods have been proposed to date which address it [23] [29] [43] and finds application in areas such as archaeology [47] [19].

One successful strategy for partial 3D object retrieval is using the descriptiveness of local shape descriptors, as local surface similarity tends to be maintained when other parts of the object are missing. A problem with retrieval using local shape descriptors is the large number of such descriptors that are generated, potentially one for every vertex. This can be somewhat counteracted by using a salient point detector, but then the retrieval quality is affected by the consistency of this detector. An efficient indexing scheme is therefore called for.

To address this issue, a complete pipeline is presented in this paper which is capable of indexing and retrieving arbitrary 3D objects based on partial queries. Under ideal circumstances the system can achieve near perfect retrieval, even with low degrees of partiality, within reasonable time constraints.

The pipeline utilises the recently introduced Quick Intersection Count Change Image (QUICCI) descriptor [52] whose binary nature makes it storage-efficient and fast to compare.

As part of this complete partial retrieval pipeline, the following novelties are introduced:

1. An indexing scheme called "Dissimilarity Tree" for efficiently retrieving binary descriptors, especially nearest neighbours with high Hamming distance.

2. An algorithm for accelerating partial 3D object retrieval using the aforementioned indexing scheme.

3. An adaptation of the QUICCI descriptor generation process to greatly improve its performance in partial 3D object retrieval applications.

The primary descriptor and distance function used in this pipeline, along with relevant background, is given in Section 10.2. The pipeline is described at a high level in Section 10.3, and the two other main contributions which are used in this pipeline are detailed in Sections 10.4 and 10.5. The various methods are evaluated in Section 10.6, and some aspects of those are discussed in Section 10.7.

10.2 Related Work

The problem of Partial Object Retrieval has to date received significant attention, both using global and using local descriptors. A number of binary descriptor indexing strategies have also been proposed. This work builds upon the QUICCI local descriptor and the Weighted Hamming distance function, which are discussed in detail.

10.2.1 Partial Object Retrieval

Partial Object Retrieval approaches presented to date can be divided into three main categories; Bag of Visual Words (BoVW) based, View-based, and Part-based [29] [43]. Other methods also exist, addressing particular applications such as CAD shape retrieval [5].

BoVW based methods use local feature descriptors to exploit that from the perspective of a local neighbourhood, shapes in a query remain similar to those in the corresponding object in a database. Lavoué et al. [26] segment a surface into small patches, and compute a codebook for each patch. Object classification is subsequently done by matching new patches against words in the codebook. Savelonas et al. [45] propose an extension to the FPFH [41] descriptor called *dFPFH*, which is used for both local and global matching in their retrieval pipeline. Ohbuchi et al. [34] combine the BoVW and the view-based paradigm by computing a bag of features over range images of an object rendered from different viewpoints, and comparing features of a query against those in a codebook. More recently, Dimou et al. [18] used features computed from patches from segmented depth images.

Part-based methods use segmentation to divide a shape into smaller distinct patches, computing one feature vector for each of them, then match these against a database of feature vectors from other parts. Agathos et al. [1] used a graph of segmented parts to locate objects with a similar structure. Tierny et al. [51] used Reeb graphs for both segmenting and encoding relationships between surface patches for partial object retrieval. Furuya et al. [21] proposed the RSVP algorithm, which partitions an object into random cuboid volumes, and describes each partition as a binary string, against which other parts can be matched. They, and others, [22] later utilised a Siamese-like network pair to project handcrafted features

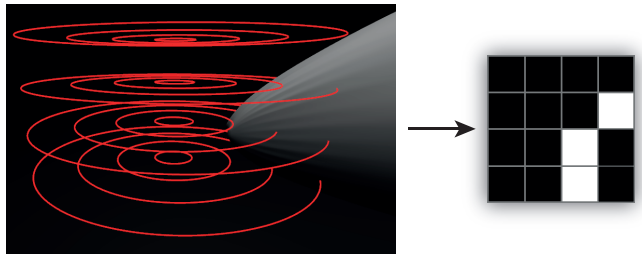


Figure 10.1: Visualisation of a 4x4-bit QUICCI descriptor construction along with the corresponding generated descriptor. White pixels in the descriptor image correspond to a bit value set to 1 (i.e. intersection counts changed), and 0 otherwise.

extracted from segmented parts into a common feature space, allowing for fast surface patch comparison.

View-based methods are able to adapt work on image matching and recognition to 3D shapes. Examples of such methods include work by Axenopoulos et al. [20], who proposed a combination of several features computed from object silhouette outlines to create the Compact Multi-View Descriptor (CMVD). The SIFT local feature [30] is used by several methods on images extracted from 3D shapes [20] [33]. One specific example is work by Sfikas et al. [47], where the Authors used the PANORAMA descriptor [35] for matching parts of archaeological to complete objects. More recently, Tashiro et al. [23] proposed a pipeline relying extensively on the SURF [6] local feature descriptor.

In the SHREC'16 Partial 3D Object Retrieval track [38], a number of additional view-based methods were introduced. Aono et al. presented three variant methods which each encoded KAZE features [2] extracted from different object views with Vector of Locally Aggregated Descriptors (VLAD) [25], Gaussian of Local Distribution (GOLD) [46], and Fisher Vectors (FV) [36]. Pickup et al. used a variant of the view-based method by Lian et al. [28], using rendered views and SIFT descriptors to find matching points.

10.2.2 The QUICCI Local Binary Descriptor

The Quick Intersection Count Change Image, proposed by van Blokland et al. [52], is a binary descriptor which captures changes in intersection counts between circles and an object's surface. These circles are laid out in layers, where each layer contains circles with linearly increasing radii. A visualisation of this structure can be seen in Figure 10.1.

As can be seen in the Figure, a grid of 4x4 circles is intersecting a 3D surface. A total of 5 circles intersect with this surface, and the remainder do not. To its right the corresponding QUICCI descriptor is shown, where black pixels indicate a bit value of 0, and white a value of 1. Note that each bit has a corresponding circle, where the bit in the bottom left corner of the descriptor is mapped to the innermost circle on the bottom layer.

Each bit in the descriptor denotes whether the number of intersections between the circle corresponding to that bit, and the circle one step smaller on the same layer, has changed. In the Figure, the bottom right 2x2 bits all have corresponding circles which intersect the object surface, which causes a response in the bottom half of column 3, but not in the bottom half of column 4, as the intersection counts did not change.

The resulting descriptor will commonly show outlines of surfaces present near the

oriented point around which the descriptor is generated. This point lies at the centre of the grid of circles, which on the descriptor corresponds to the grid point closest to the arrow's head in Figure 10.1.

10.2.3 Weighted Hamming Distance Function

There exist two possible bit errors when comparing a pair of binary descriptors (corresponding to a query shape and a target shape from a database, respectively) using a bitwise distance function such as Hamming distance. A type A error occurs when a bit set to 1 in the query is set to 0 in the target, and a type B error represents the case where a bit set to 0 in the query is set to 1 in the target. The Hamming distance function considers both of these bit errors as equivalent in importance.

Meanwhile, the Weighted Hamming distance function proposed by van Blokland et al. [52] observes that it may not always be desirable to weigh both types of bit errors equally. In the case of QUICCI descriptors, bits set to 1 represent surface outlines. A good match must also contain these bits, but may also include others due to responses from other geometry. For QUICCI descriptors the type A error is therefore more important than the type B error.

The Weighted Hamming distance function normalises the contribution of each bit error type by the total number of such errors that *can* occur, thereby weighting the importance of each bit error type equally as a group. Thus the Weighted Hamming distance function is asymmetric. In a sparse descriptor, this implies that a type A error is weighted much more than a type B error. The distance function is listed in Equation 10.1.

$$\begin{aligned}\delta_{WH}(D_q, D_t) = &\frac{\sum_{r=1}^{N}\sum_{c=1}^{N}(D_q[r,c](1-D_t[r,c]))}{max(\sum_{r=1}^{N}\sum_{c=1}^{N}D_q[r,c],1)} \\ &+\frac{\sum_{r=1}^{N}\sum_{c=1}^{N}((1-D_q[r,c])D_t[r,c])}{max(N-\sum_{r=1}^{N}\sum_{c=1}^{N}D_q[r,c],1)}\end{aligned} \tag{10.1}$$

Where D_q and D_t are respectively the query and target descriptors being compared, $D[r,c]$ represents the bit at row r and column c of descriptor D, and the size of the descriptor is NxN bits.

Experiments by van Blokland et al. showed that using the Weighted Hamming distance function resulted in improved retrieval performance relative to Hamming distance of QUICCI descriptors when additive noise was applied.

10.2.4 Indexing of Binary Descriptors

The need for indexing binary descriptors commonly arises in algorithms representing shape features as binary descriptors, but also in other fields such as dimensionality reduction through Locality-Sensitive Hashing (LSH) [24]. Some popular methods utilising LSH include Minhash proposed by Broder et al. [12] [11] and Simhash [42] by Sadowski et al.

There exist a number of methods which produce and compare binary shape features, such as BRIEF by Calonder et al. [13], an extension called ORB by Rublee et al. [40], and BRISK by Leutenegger et al. [27]. A binary descriptor which has specifically been proposed for 3D point matching is B-SHOT by Prakhya et al. [37], and the aforementioned QUICCI descriptor by van Blokland et al. [52].

Local binary descriptors are often produced in large quantities, which raises the need for

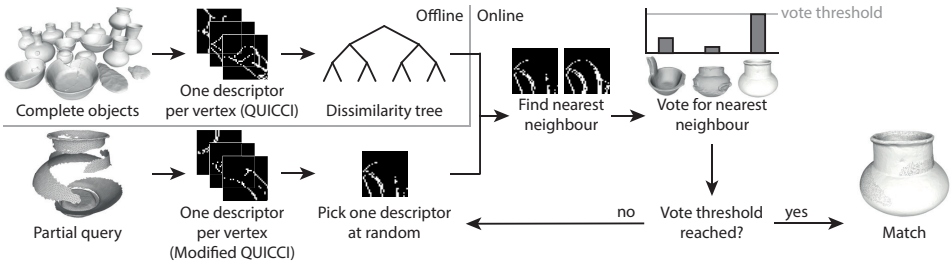


Figure 10.2: An overview of the proposed partial retrieval pipeline. A dissimilarity tree is constructed offline over the QUICCI descriptors of a set of complete objects. Querying these objects with a partial object involves computing *modified* QUICCI descriptors for each vertex, iteratively, selecting one such query descriptor at random, determining the closest indexed descriptor to the one randomly selected, and finally voting for the object from which the nearest neighbour originated. When an object has reached a set number of votes, it is deemed the closest match to the partial query.

acceleration structures capable of efficiently locating nearest neighbours in Hamming space. A number of methods have been proposed for this purpose [10] [9] [3] [31]. Unfortunately, these initial attempts only support short descriptors, are limited to the retrieval of neighbours up to a Hamming distance of 2, or both. This significantly limits their applicability.

More recent work includes the Multi-Index Hashing (MIH) algorithm proposed by Norouzi et al. [32], which subdivides descriptors into regions, building inverted hash tables for each subdivision. Chappell et al. [14] proposed a similar approach, instead using inverted lists. An improved variation of MIH was presented by Reina et al. [39], utilising a prefix tree to store the index itself, and a separate hash table for pruning irrelevant branches while querying.

The Hamming Tree proposed by van Blokland et al. [52] exploits the notion that descriptors with a low Hamming distance must by necessity have a similar number of bits set to 1. The tree first categorises descriptors by the total number of 1 bits, then divides descriptors into regions, categorising them by the number of bits set to 1 within each region.

Unfortunately, the previously introduced binary indexes typically assume that the nearest neighbour to a query in the database has a low Hamming distance, which is not a property which can be assumed consistently. This issue is particularly significant for the application of QUICCI descriptors on the problem of partial retrieval. The discussed indexing strategies tend to scale poorly with increasing distance from a descriptor to its nearest neighbour, which makes their application intractable when this distance is high.

Moreover, they cannot be adapted to use the Weighted Hamming distance function, which is a highly desirable property for the application of QUICCI descriptors.

10.3 Partial Retrieval Pipeline

The proposed partial retrieval pipeline consists of an online and offline component, and is visualised in Figure 10.2. In the offline phase, one QUICCI descriptor is extracted from each vertex in a set of complete objects. Using these descriptors, a dissimilarity tree is constructed, which is described in detail in Section 10.4.

The online phase takes a partial query object as input. In similar fashion to the offline phase, one descriptor is computed for each of the object's vertices, albeit a *modified* version

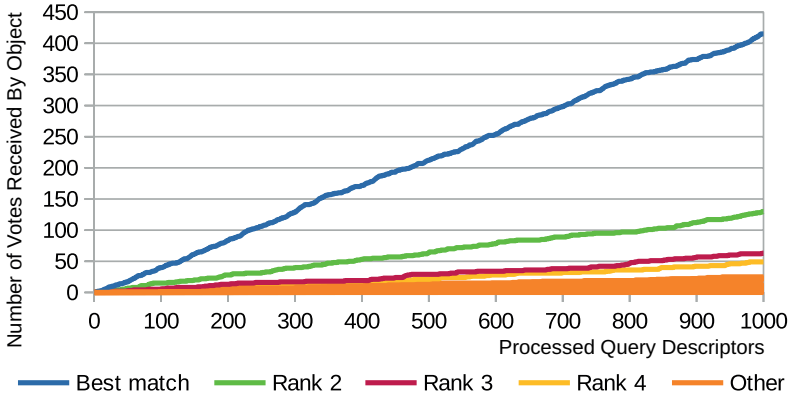


Figure 10.3: A visualisation of vote counts received by objects as more query descriptors are processed, while using the proposed pipeline to locate the nearest neighbour of a single partial query object. Each line represents a single object from the SHREC'16 partial retrieval dataset [38]. For example, the best match (blue line) received 300 of the first 700 votes cast. The linear nature of these curves suggests that the probability that a particular object will receive a vote is approximately constant across the voting process.

of the QUICCI descriptor is used here, see Section 10.5. The voting steps outlined below are then repeated until an object match is found or all vertices have been exhausted.

One descriptor is first selected from the set of query descriptors at random. Using the dissimilarity tree structure, the nearest neighbour in terms of Weighted Hamming Distance is found in the set of descriptors extracted from the complete objects. Next, a vote is cast for the object containing the found nearest neighbour. This process is repeated until an object has received a predefined number of votes (threshold), upon which this object is considered the best match for the partial query. Our evaluation shows that a threshold of 10 votes is sufficient. Additional search results can be obtained by evaluating additional queries until the desired number of other objects reaches the vote threshold.

The motivation for using a voting threshold to exit the counting process can be seen in Figure 10.3, which shows the number of votes received by different objects with respect to the number of processed query descriptors using the proposed method. As this relationship is approximately linear, it is possible to terminate the search early by using a vote threshold.

For each randomly selected query descriptor, only the nearest neighbour descriptor is considered. The motivation for not considering other neighbours in addition to the nearest is shown in Figure 10.4. The Figure shows the average Weighted Hamming distance scores of the closest 50 matches to 1,000 descriptor queries. For legibility, distance scores for each neighbour have been normalised relative to the Weighted Hamming distance of the nearest neighbour (search result index 0) of the same query. The Figure shows that the average distance score to the query descriptor increases between the nearest and second nearest neighbours by over a factor of 7, where further neighbours exhibit similar scores. We therefore conclude only the nearest neighbour to the query is relevant to the retrieval process.

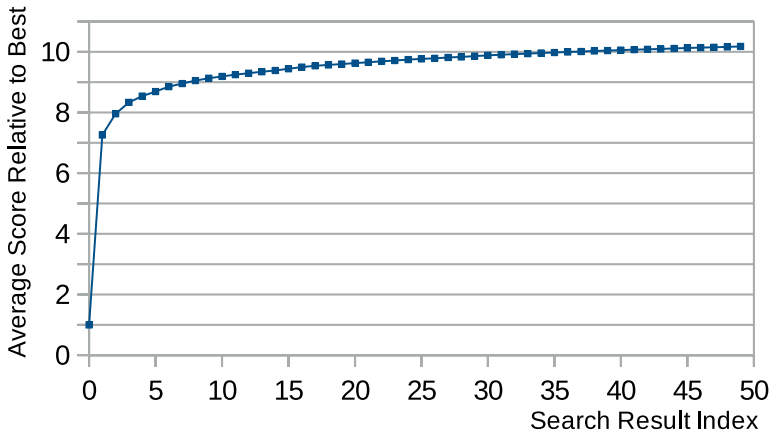


Figure 10.4: Average Weighted Hamming distance of the top 50 descriptor search results (where the top search result has index 0) of 1,000 descriptor queries normalised to the distance score of the top search result. As the average distance between the top and second ranked search result is high, it is unlikely that any result but the best result is relevant to the query.

10.4 Dissimilarity Tree for Indexing Binary Descriptors

The Hamming Tree [52] and other binary descriptor indexing structures proposed in previous work (see Section 10.2.3) have the ability to efficiently locate descriptors similar to a given query when the Hamming distance to those matches is generally low. However, in applications where this distance is large, the search time of these methods tends to increase significantly.

This problem has a high likelihood of occurring when using QUICCI descriptors for partial object retrieval. In this case, a partial query descriptor will generally contain a subset of the bits set compared to those for the correctly matching descriptor of the complete object.

A new tree indexing structure, the *Dissimilarity Tree*, is proposed here which is capable of efficiently retrieving nearest neighbours when the distance to these neighbours is high. The Dissimilarity Tree is also capable of supporting the Weighted Hamming distance function for querying, which has been shown to be superior for QUICCI descriptor ranking [52]. The Dissimilarity Tree can be used for arbitrary binary descriptors, but in the remainder of this paper will be explained in the context of QUICCI descriptors.

The Dissimilarity Tree is a binary tree that exploits the assumption that bits set to 1 in a binary descriptor are not distributed randomly, and aims to cluster descriptors which have similar bits set. It does so by attempting to create subtrees where patterns of bits are consistently set in all contained descriptors.

When it is known that a specific bit will have a consistent value (0 or 1) across all descriptors in a subtree, it is possible to compute the (Weighted) Hamming distance that will be incurred for that bit for a given query descriptor. The more effectively this can be done, the greater the ability of the search algorithm to prune irrelevant branches.

For each node in the Dissimilarity Tree, two characteristic binary images are computed, one representing the bitwise sum (OR) of all descriptors contained in both subtrees and one representing the bitwise product (AND). These allow a minimum distance to be computed

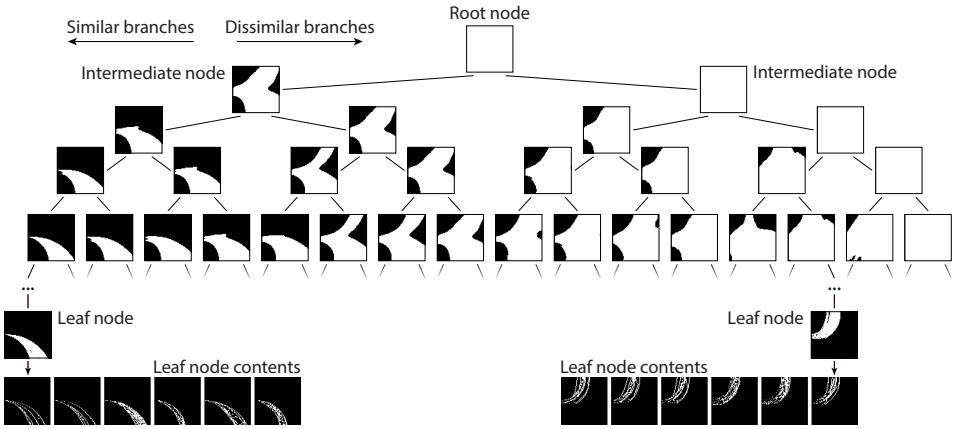


Figure 10.5: Visualisation showing the four top layers of the Dissimilarity tree generated from all descriptors in the SHREC'16 dataset. Each node is represented by its sum image. All outgoing branches from nodes directed to the left represent the similar branch of that node, and those directed to the right are those that are dissimilar. Two examples of leaf nodes are also shown, along with a subset of the descriptors contained within each.

from the query descriptor to all descriptors contained in that particular node, as the sum and product descriptors denote which bits in a subtree are consistently set to 0 and 1, respectively.

At each node, the set of descriptors is partitioned into two roughly equal subsets. The *similar* subset contains descriptors which maximise the number of bits consistently set to 0 or 1. The remaining descriptors form the *dissimilar* subset. A visualisation of an example dissimilarity tree can be seen in Figure 10.5.

For descriptors which are either highly sparse or dense, the product and sum images respectively do not provide meaningful value to the querying process when the Weighted Hamming distance function is used. Since QUICCI descriptors are highly sparse, we omit the product image. Section 10.4.1 details the algorithm for constructing a Dissimilarity Tree while Section 10.4.2 describes the querying process.

10.4.1 Dissimilarity Tree Construction

The tree construction algorithm described in this section details how a dissimilarity tree can be constructed from a fixed set of descriptors. However, it should be possible to construct a tree incrementally by recomputing only the affected parts of the tree, in a similar fashion to the heapify algorithm [17].

The root of the tree represents the set of all descriptors in the index. The tree construction algorithm divides this set into two approximately equally sized disjoint subsets. The *similar* subset of these contains descriptors where regions of bits set consistently to 0 or 1 are created, by moving descriptors where those bits are set otherwise to the *dissimilar* set. For each of these subsets a new node is created, and the procedure is repeated recursively until the set of descriptors represented by a node is smaller than a threshold. Pseudocode for the construction algorithm is shown in Listing 2.

In order to maximise the size of the region of bits set to a consistent value, the number of descriptors in a given set is determined for which a specific bit is set to the value 1, which

```
def buildDissimilarityTree(node, descriptors):
    if descriptors.length <= maxDescriptorsPerLeaf:
        markAsLeafNode(node)
        return
    similarDescriptors = descriptors.copy()
    dissimilarDescriptors = create_set() # empty
    # Count popularity of set bits in descriptors
    counts = computeBitResponseCounts(descriptors)
    bitOrder = sortAscending(counts)

    # Split descriptors into similar and
    # dissimilar subsets
    for bitIndex in bitOrder:
        for descriptor in similarDescriptors:
            if descriptor[bitIndex] == 1:
                similarDescriptors.remove(descriptor)
                dissimilarDescriptors.insert(descriptor)

    # Recurse for similar and dissimilar branches
    node.similarBranch = create_node()
    buildDissimilarityTree(node.similarBranch,
                           similarDescriptors)
    node.dissimilarBranch = create_node()
    buildDissimilarityTree(node.dissimilarBranch,
                           dissimilarDescriptors)

    node.productImage = computeAND(descriptors)
    node.sumImage = computeOR(descriptors)
```

Listing 2: Pseudocode of the Dissimilarity Tree construction algorithm.

will be referred to as the *popularity* of that bit. The bit popularity can be visualised in a heatmap, an example of which, computed over the entirety of the SHREC'16 Partial Object Retrieval dataset, is shown in Figure 10.6. As can be seen in the Figure, there are areas where bits are frequently set to 1 (middle left), and others less so (top and bottom left, middle right).

The popularity heatmap in Figure 10.6 represents the occurrence counts for the root node of Figure 10.5. In the latter Figure, the sum image of the node along the similar branch shows that the less common areas of the heatmap have been cut away as part of the first subdivision, leaving zero-valued areas in the sum image.

The division strategy starts by computing the aforementioned bit popularity heatmap. Next, bit positions within the descriptor are sorted in order of ascending popularity, the *bitOrder*.

Next all descriptors of the current node are placed in a the *similarDescriptors* set and the *dissimilarDescriptors* set is initalised to the empty set. Then, for each bit position in *bitOrder* starting from the least popular one, all descriptors are found in *similarDescriptors* which have that particular bit set to 1. These descriptors are moved over to *dissimilarDescriptors*. Moving all descriptors that have a specific bit set may result in a tree which is not perfectly balanced, however, only moving a part of the descriptors which have a particular bit set to 1

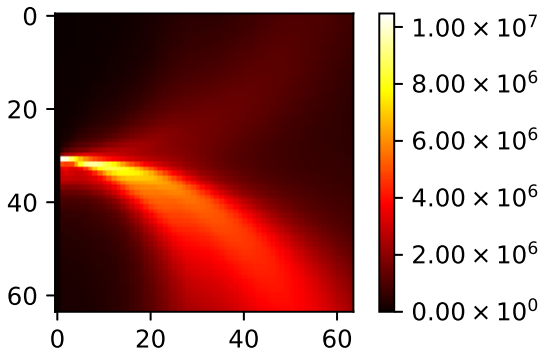


Figure 10.6: Heatmap showing occurrence counts of bits of all descriptors of complete objects in the SHREC'16 partial retrieval dataset.

does not yield the advantage of that particular bit being set to 0 in the similar node's sum image.

If the node being visited contains a set of descriptors which has fewer descriptors than a set threshold, the subdivision can stop. By constructing a number of indices, it was determined that the value of 32 for this threshold yields optimal execution times for QUICCI descriptors.

10.4.2 Dissimilarity Tree Querying

As mentioned previously, the sum and product images of each node allow a minimum distance to be computed between the query descriptor and all descriptors contained within both branches of a particular node. This is possible because the sum and product images by definition represent all descriptors contained within the node having a particular bit set to 0 or 1, respectively.

If for instance a particular bit in the query descriptor is set to 1, and the sum image of a node has that same bit set to 0, every single descriptor contained within the node will incur a distance penalty at that bit.

By summing up all such universal distances using a distance function, such as Hamming or Weighted Hamming, a minimum distance can be computed between the query descriptor and all descriptors contained in the node being considered, thus allowing to make decisions on culling a subtree.

The querying algorithm of the dissimilarity tree works in an iterative fashion, and is outlined in Listing 3. A priority queue is kept of open nodes, sorted by their minimum distance. The queue is initialised to the root node. During each iteration, the node with the lowest minimum distance is removed from the queue. If the node is a leaf node, the list of descriptors contained is searched for matches. If the node is not a leaf node, the minimum distance to the similar and dissimilar branch nodes is computed, and both are inserted into the queue.

Meanwhile, a list of fixed size is kept with the closest matching descriptors found up to that point. The list is sorted by each result's Weighted Hamming distance to the query. When the list of search results has the desired size and the worst result in the list is lower

```
def queryDissimilarityTree(query, rootNode, resultCount):
    openQueue = create_priority_queue()
    openQueue.insert(rootNode)
    searchResults = create_list()
    # While we need more results and nodes
    # that can improve the results exist
    while len(searchResults) < resultCount
          and searchResults[-1].distance > openQueue[0].minDistance:
        node = openQueue.remove(0)
        if isLeafNode(node):
            searchResults.append(node.descriptors)
            searchResults.computeDistanceTo(query)
            searchResults.sortByDistance()
            searchResults.shrinkTo(resultCount)
        else: # node is intermediate node
            node.similarBranch.minDistance
                = minWHDist(query, node.similarBranch)
            node.dissimilarBranch.minDistance
                = minWHDist(query, node.dissimilarBranch)

            # Only consider child nodes if
            # they can improve search results
            if node.similarBranch.minDistance
               < searchResults[-1].distance:
                openQueue.insert(node.similarBranch)
            if node.dissimilarBranch.minDistance
               < searchResults[-1].distance:
                openQueue.insert(node.dissimilarBranch)

# Compute minimum weighted Hamming distance
def minWHDist(query, node):
    return sum(popcnt(query and not node.sumImage))
              * (len(query) / popcnt(query))
         + sum(popcnt(not query and node.productImage))
              * (len(query) /
                (len(query) - popcnt(query))
```

Listing 3: Pseudocode showing the main steps of the Dissimilarity Tree querying algorithm. The procedure takes a query descriptor, the root of a Dissimilarity Tree, and the desired number of closest search results to retrieve.

than the minimum distance to the next unvisited open node, all search results have been found and the search can terminate.

10.5 Adapting QUICCI Descriptors for Partial Retrieval

Partial objects that constitute queries are generally not closed surfaces. They commonly contain surface discontinuities, which we shall call *boundaries*. Thus if the QUICCI descriptor is used on vertices of such a partial object, one would get responses to boundaries,

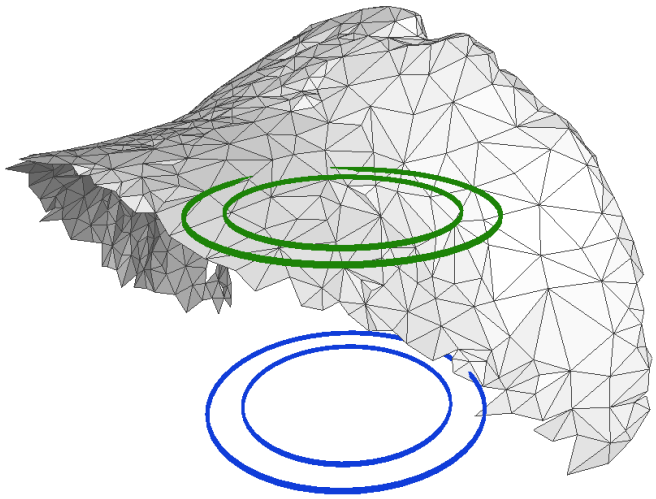


Figure 10.7: A query mesh along with two pairs of hypothetical circles used during the construction of the QUICCI descriptor. The green circles on top indicate an intersection count change of 2, while the blue circle pair on the bottom indicates an intersection count change of 1.

that have no match in the corresponding complete object.

An example of a partial query object can be seen in Figure 10.7, where the intersections count changes of successive circle pairs result in QUICCI descriptor responses. The green ones will result in a response which has a match in the corresponding complete object but the blue ones (across a boundary) will not. Ideally, these boundary responses should be filtered out.

Fortunately, a slight modification to the computation process of QUICCI descriptors for partial query objects, effectively filters out most interference caused by boundaries. This is based on the fact that boundaries result in intersection count changes by 1, whereas closed surfaces result in intersection count changes by 2. Thus, during QUICCI construction of partial query objects we only record intersection deltas of at least 2.

Figure 10.8 shows a comparison between the existing QUICCI descriptor and the proposed modification for partial query objects, along with the matching descriptor for the corresponding complete object. The proposed change filters out nearly all responses induced by boundaries, leaving responses belonging to the shape being queried. Ideally, the response set of the modified query image is a subset of the corresponding set for the complete object.

10.6 Evaluation

The proposed partial retrieval pipeline was evaluated on a set of real 3D object scans. The primary dataset chosen for this purpose is the SHREC'16 Partial Object Retrieval track dataset [38], which consists of a variety of historic artefacts, primarily ceramic pottery. We create additional partial query objects from this dataset to form a new augmented dataset of partial query objects, detailed in Section 10.6.1.

The three primary contributions are subsequently evaluated individually. The Dissimilarity Tree is evaluated in Section 10.6.2. The proposed QUICCI modification for partial query

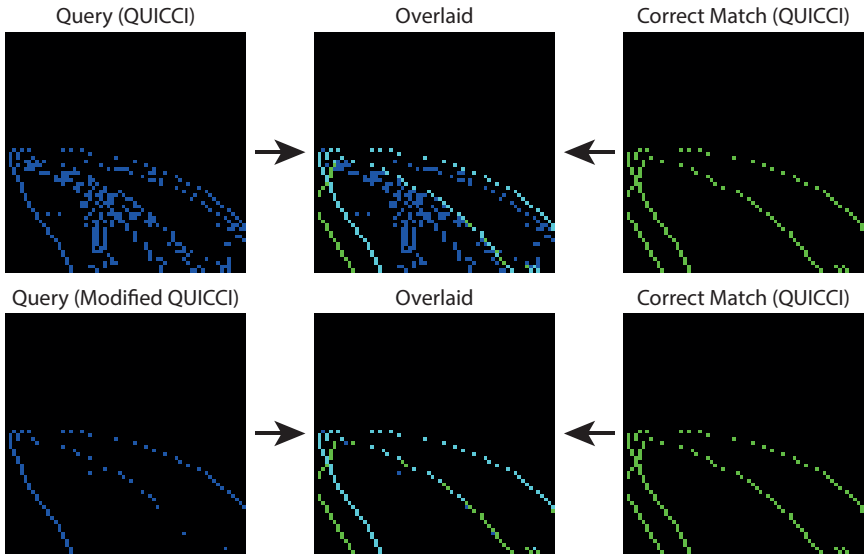


Figure 10.8: QUICCI descriptors for a vertex from a partial query object (left side), and its corresponding matching vertex in the complete object (right side). The top left is an original QUICCI descriptor while the bottom left is a modified QUICCI. Both query-match pairs are shown overlaid on top of each other in the middle, and show a significant reduction in noise in the query descriptor when using the modified QUICCI.

objects is evaluated for its ability to filter responses to query boundaries in Section 10.6.3, and for its matching capabilities in Section 10.6.4. The complete partial retrieval pipeline is finally externally evaluated using the augmented SHREC'16 dataset in Section 10.6.5 and on part of the original SHREC'16 query objects in Section 10.6.6.

All algorithms presented were implemented in C++ and CUDA where applicable. All implementations were executed on a machine with an AMD R9 3900X 12-core CPU and an Nvidia GeForce RTX 3090 GPU. The authors intend to make source code publicly available, and apply for the Graphics Replicability Stamp (GRSI) [4] upon publication.

Unless stated otherwise, the QUICCI descriptor resolution was set to 64x64 bits for all experiments. The support radius used was 100 units, which was found to be able to capture shapes in the local area. This trend is visible in the heatmap shown in Figure 10.6. All objects in the SHREC'16 dataset have been scanned at the same scale, and thus no scale alteration or correction was required.

10.6.1 SHREC'16 Dataset Augmentation

The SHREC'16 Partial Retrieval track dataset [38] has been chosen for the evaluation of the proposed retrieval pipeline. This allows direct comparison to results from other methods which were evaluated using this benchmark. While the dataset contains a variety of query objects, their quantity is limited, and is therefore augmented.

The used augmentation is similar to the one used in the SHREC'13 track for partial object retrieval [49]; a first partial query set is created by generating meshes of all triangles in view from a random viewpoint. We used this to create one partial query mesh for each

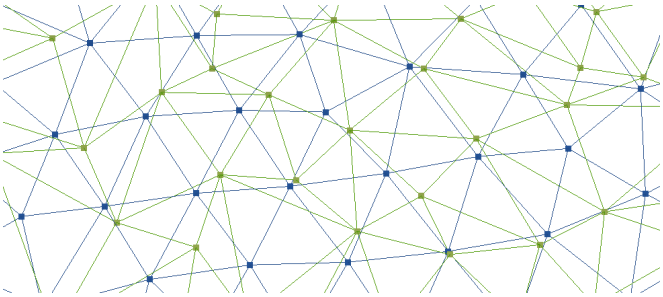


Figure 10.9: A part of two identical surfaces shown in wireframe form overlaid on top of one another, where one of the two has been remeshed. Vertices of both meshes have been highlighted.

object in the SHREC'16 dataset, thereby creating 383 query objects. This query dataset is called $AUGMENTED_{Best}$.

However, while the extracted query meshes give a good indication for *best case* retrieval performance, a more realistic retrieval scenario could involve subsequent scans of the same object with different triangulations. We therefore also generated a second augmented dataset by remeshing all meshes in $AUGMENTED_{Best}$, creating the $AUGMENTED_{Rem}$ query dataset. In particular, we have used the remeshing algorithm proposed by Botsch et al. [8] as it works on non-watertight meshes, which has been made available as part of PMP library [48]. An example of the effect of this remeshing step can be seen in Figure 10.9.

The generated query datasets will be made available upon publication.

10.6.2 Dissimilarity Tree

The effectiveness of the dissimilarity tree index structure was evaluated by querying a tree constructed over all descriptors from all complete objects in the SHREC'16 dataset, which amounts to a total of 36.5M indexed descriptors. A set of 100,000 unique descriptors was randomly selected from the descriptors of all objects in the $AUGMENTED_{Best}$ set.

Each descriptor was subsequently used to query the tree. The resulting execution time of each query was counted in a histogram with bins of 0.1 seconds. As a reference, the first 2,500 queries were also used to measure the execution time of a sequential search, which resulted in another execution time distribution histogram. The results are shown in Figure 10.10.

The histograms show that significant speedups are achieved using the proposed dissimilarity tree structure over a sequential search. Out of the 100,000 queries, only 25 took longer than the average sequential search.

The perf profiling tool showed that for a given query, on average 26.2% of execution time is spent on visiting intermediate nodes, and 56.1% is spent on visiting leaf nodes. The remainder is spent on open node queue management. Visiting leaf nodes is almost entirely (99.0%) spent on the computation of weighted hamming distances.

10.6.3 Modified QUICCI evaluation

We evaluate here the effect of the modification to the QUICCI descriptor proposed in Section 10.5, which removes descriptor responses to object boundaries that typically exist in the

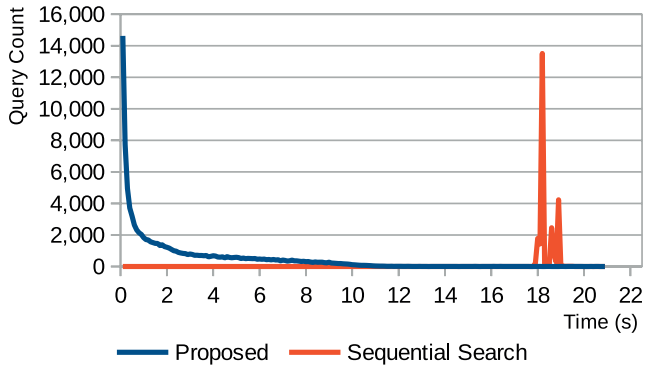


Figure 10.10: Histogram with bins of 0.1s showing the distribution of execution times of 64x64 bit queries using indexed and sequential searches. All sequential search occurrence counts have been scaled up by a factor of 10 for legibility.

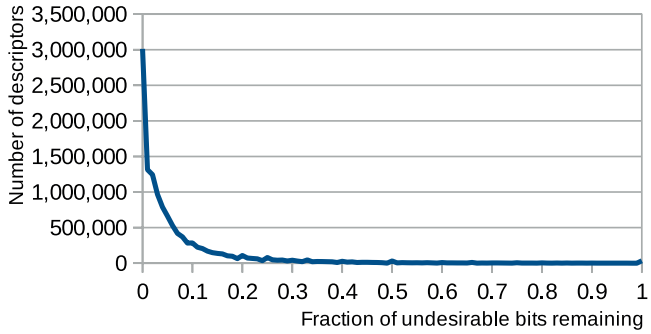


Figure 10.11: A histogram with bins of size 0.01, computed over 12.3M 64x64 bit partial query object descriptors, showing the fraction of undesirable (most boundary response) bits remaining in a modified QUICCI descriptor over the corresponding number of bits in the original QUICCI.

partial query objects only.

The $AUGMENTED_{Best}$ set is used to evaluate this modification. As each partial query object is extracted from a dataset object, there exists an exact correspondence between their vertices (ground truth). It is subsequently possible to determine the exact bits which are set in a query descriptor, but not in the correctly matching descriptor of the complete object, which are thus *undesirable*.

The chart in Figure 10.11 shows that in 78.0% of the tested descriptors, the number of undesirable bits is reduced to under 10%. In 24.5% of all cases, the undesirable bits are removed entirely. The shown results have been computed over a set of 12.4M partial query descriptors, from which a relatively small number (105,448) have been excluded for not containing any query boundary responses (to avoid divisions by 0) or unreliable correspondence between vertices.

However, the modification also removes some bits which are set in both the partial and complete descriptors, and are thus desirable. A histogram over the fraction of bits set to 1 in both descriptors relative to the total number of such bits in the complete descriptor is shown

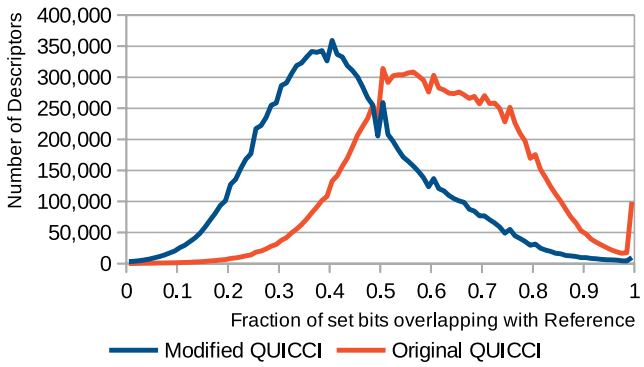


Figure 10.12: A histogram with bins of size 0.01, computed over 12.3M 64x64 bit partial query object descriptors, showing the distribution of fractional overlap of bits set to 1 in a partial query descriptor relative to the complete object from which the query was extracted.

QUICCI	$AUGMENTED_{Best}$	$AUGMENTED_{Rem}$
Original	0.72	0.17
Modified	0.99	0.49

Table 10.1: The fraction of correctly retrieved nearest neighbours when using combinations of the original and modified QUICCI, measured using the $AUGMENTED_{Best}$ and $AUGMENTED_{Rem}$ query object sets.

in Figure 10.12.

The Figure shows that the average of fractional overlap decreases from 61.5% to 42.7% when using the QUICCI modification, a loss of 30.6%. However, while the fraction of desirable bits in query descriptors decreased, the average number of undesirable bits decreased even further, from an average of 84.2 bits per descriptor to 4.06 bits. Thus in the modified QUICCI, responses in the query descriptor can, to a high degree, also be expected to be present in the descriptor of the corresponding complete object.

10.6.4 Modified QUICCI for Partial Object Retrieval

While Section 10.6.3 showed the proposed modification to the QUICCI descriptor to produce more reliable query descriptors, its effect on matching performance must be evaluated too.

A distance score for each of the query objects in $AUGMENTED_{Best}$ and $AUGMENTED_{Rem}$ was computed for each of the complete objects in the SHREC'16 dataset. The distance score of an object pair was computed by summing the distances of each descriptor in the query object to its nearest neighbour in the set of descriptors of the complete object. Next, all objects were ranked by their total distance to the query object. The results are outlined in Table 10.1. As can be seen, the nearest neighbour matching performance improves significantly when using the modified QUICCI descriptors, for both the $AUGMENTED_{Best}$ and $AUGMENTED_{Rem}$ datasets.

Confusion matrices were also computed across the query and complete objects, see Figure 10.13. Each row in these matrices represents the scores of a single query object to each complete object it was compared against. For each of these rows, the distance scores

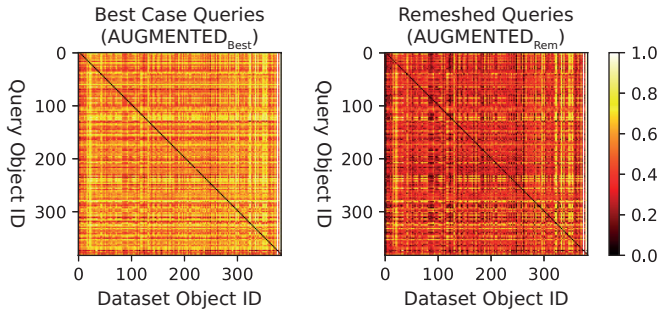


Figure 10.13: Confusion matrices showing summed Weighted Hamming distances from all objects in each of the augmented query object datasets to all objects in the SHREC'16 dataset. Since one partial query object is computed from each SHREC'16 dataset object, objects with matching IDs should correspond, and therefore have a low distance (leading diagonal). Distances of each row are normalised for legibility. Both query object datasets contain a visible desirable leading diagonal of low distances, although this is less pronounced for the remeshed query objects, which is also reflected in worse nearest neighbour matching performance.

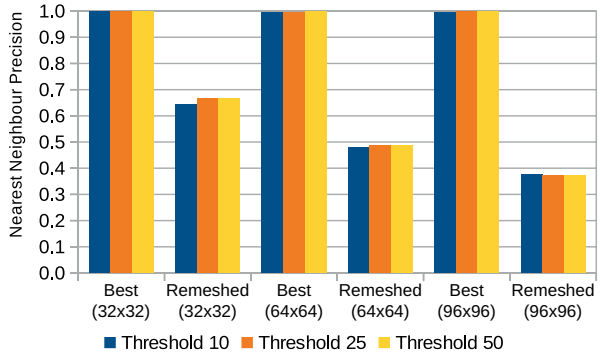


Figure 10.14: Fraction of correct nearest neighbour object matches for several voting thresholds, using the proposed partial object retrieval pipeline, tested using the $AUGMENTED_{Best}$ (Best) and $AUGMENTED_{Rem}$ (Remeshed) partial query object datasets.

have been normalised to the range [0, 1].

The confusion matrix from the $AUGMENTED_{Best}$ set shows a clear distinction between partial query and complete objects. For the remeshed partial queries $AUGMENTED_{Rem}$, the nearest neighbour distance scores naturally increase.

10.6.5 Partial Retrieval Pipeline

Considering the proposed partial object retrieval pipeline, in this section we consider the nearest neighbour retrieval performance and the effect of the threshold parameter, as well as the execution times for querying objects.

Figure 10.14 shows the effect of the threshold parameter on nearest object neighbour retrieval performance for objects from $AUGMENTED_{Best}$ and $AUGMENTED_{Rem}$, for three different descriptor resolutions. As can be seen, the method is almost resilient to this

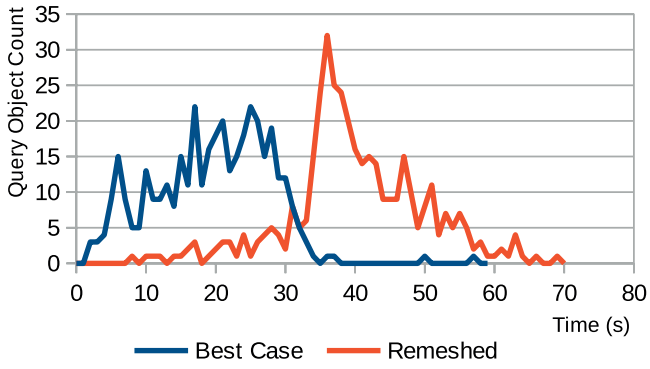


Figure 10.15: Execution times of the $AUGMENTED_{Best}$ and $AUGMENTED_{Rem}$ query meshes when using them in the proposed pipeline with a descriptor resolution of 64x64 and a vote threshold of 10.

parameter across resolutions and a low value can be used.

While there is little variation in the results of query objects from $AUGMENTED_{Best}$, different descriptor resolution yields a significant variation in matching performance for the query objects of $AUGMENTED_{Rem}$.

The execution times of the queries are shown in Figure 10.15 using a vote count threshold of 10 and descriptor resolution of 64x64. It is also worth noting that queries based on different descriptors can be executed in parallel, although most of this acceleration was lost due to ensuring that results are reproducible. As can be seen in the Figure, there is a significant difference in the execution times of query objects from $AUGMENTED_{Best}$ and $AUGMENTED_{Rem}$. These results are discussed further in Section 10.7.

10.6.6 SHREC'16 Partial Retrieval Performance

The proposed partial retrieval pipeline is compared against the results presented in the SHREC'16 Partial Shape Query track [38] as well as the results for the equivalent benchmark presented by Savelonas et al. [45], which also includes results for the PANORAMA descriptor by Sfikas et al. [47] and Global Fisher features [44]. As the source code of these works was not available, we have used the results from the referenced papers. Dimou et al. [18] have also tested their work against this dataset, but no nearest neighbour retrieval performance was provided.

The majority of the SHREC'16 benchmark focuses on the classification of objects into classes rather than specific object retrieval. Only the *artificial queries*, which are culled versions of the database objects, have matching objects in the database. Fortunately, they also provide Nearest Neighbour data. Because the proposed retrieval pipeline is intended for exact part-in-whole matching, this comparison focuses on Nearest Neighbour. The results are shown in Figure 10.16.

As shown in the Figure, the proposed method is able to correctly identify all partial queries in the benchmark, across multiple descriptor resolutions. While Tran et al. also accomplish this, their method uses the Iterative Closest Point (ICP) algorithm [15] [7] 512 times per candidate match [38]. While no execution times are listed, we estimate that our method is likely to run faster.

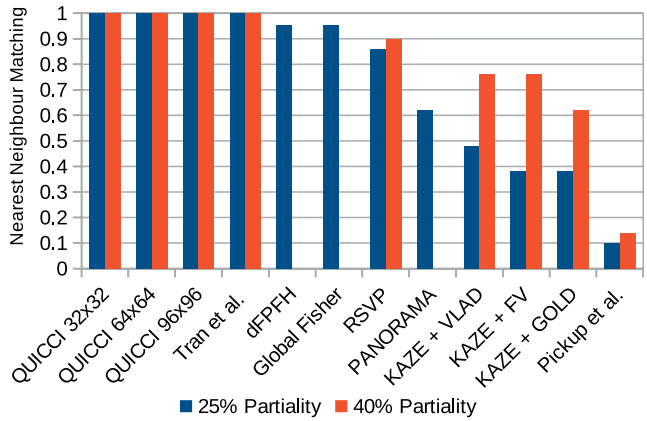


Figure 10.16: Comparison of nearest neighbour retrieval performance of the Virtual Hampson Museum collection. Query objects with 25% and 40% partiality were used. Results for the Tran et al., RSVP, KAZE+VLAD, KAZE+FV, KAZE+GOLD, and Pickup et al. methods are taken from [38] and dFPFH, Global Fisher, and PANORAMA were taken from [45]. The latter does not show results for the 40% partiality queries which are therefore missing.

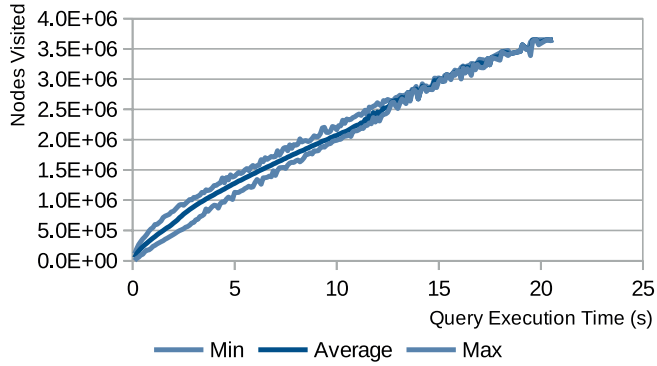


Figure 10.17: The relationship between the execution time of a query using the dissimilarity tree, and the number of tree nodes visited during that query's execution. For each time slice of 0.1 second, the minimum, maximum, and average number of nodes visited for all queries which executed within that time slice is shown.

10.7 Discussion

Figure 10.17 shows that there appears to be a linear relationship between the query execution time and the number of tree nodes visited by the algorithm. As the querying algorithm iterates until it determines that no nodes with smaller distances than the ones found are present in the Dissimilarity Tree, one can conclude that the more dissimilar a query descriptor is from its nearest neighbour descriptors in the tree, the longer querying will take.

Figure 10.14 shows that there is a non-insignificant effect on partial object retrieval performance when using remeshed versions of the query objects. The severity of this effect varies across different descriptor resolutions.

The cause of this loss in matching performance is that the positions of vertices on the object surfaces are slightly shifted causing changes in intersection counts to occur elsewhere on the descriptor, which can be seen in Figure 10.9.

Because the distance between each vertex and its closest neighbour in the remeshed mesh is small, and based on anecdotal evidence, the resulting effect is that corresponding QUICCI descriptors on the unmodified and remeshed object contain portions of bits which have either been shifted left or right by 1 bit. As the Weighted Hamming distance function only considers bits in exactly the same position, this incurs a distance penalty on what would otherwise have been a good match.

The probability of this distance penalty occurring is diminished when the descriptor resolution is lowered. As the distances between QUICCI intersection circles is increased to cover the same support radius, the probability of bit shifts decreases, thereby resulting in the improved matching performance observed in Figure 10.14.

The distance penalty also has the downside of increasing query execution times. As indicated in Figure 10.17, we observed a relationship between the similarity of a query descriptor and the nearest neighbour in the set of complete object descriptors, and the execution time of that query. When the distance to the nearest neighbour increases, so does the execution time of the dissimilarity tree search algorithm. This increase is visible in Figure 10.15.

Given the extremely promising nature of the retrieval results of the partial query objects from the $AUGMENTED_{Best}$ dataset, we conjecture that if in future work a distance function is found which can remedy the aforementioned distance penalty issue, it should be possible to both significantly increase the matching performance of remeshed queries while simultaneously reduce query times.

10.8 Conclusion

A small modification to the QUICCI descriptor was shown to be advantageous for partial retrieval tasks. An indexing scheme for binary descriptors called *Dissimilarity Tree* was also proposed, and was shown to greatly reduce nearest neighbour retrieval time. Finally, an accurate and efficient search algorithm for partial 3D object retrieval using the aforementioned Dissimilarity indexing structure was proposed.

10.9 Acknowledgements

The authors would like to thank the HPC-Lab leader and PI behind the "Tensor-GPU" project, Prof. Anne C. Elster, for access to the Nvidia DGX-2 system used in the experiments performed as part of this paper. Additionally, the authors would like to thank the IDUN cluster [50] at NTNU for the provision of additional computing resources. Multiple images in this work were captured using Meshlab [16].

10.10 References

[1] Alexander Agathos, Ioannis Pratikakis, Panagiotis Papadakis, Stavros Perantonis, Philip Azariadis, and Nickolas S. Sapidis. 3D articulated object retrieval using a graph-based representation. *The Visual Computer*, 26(10):1301–1319, October 2010.

[2] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison. Kaze features. In *European conference on computer vision*, pages 214–227. Springer, 2012.

[3] Abdullah N. Arslan and Ömer Eğecioğlu. Dictionary look-up within small edit distance. *International Journal of Foundations of Computer Science*, 15(1):57–71, 2004.

[4] Marco Attene, Marc Alexa, Klaus Mueller, Helwig Hauser, Joaquim Jorge, Konrad Polthier, and Vadim Shapiro. Graphics replicability stamp initiative, 2021.

[5] Jing Bai, Shuming Gao, Weihua Tang, Yusheng Liu, and Song Guo. Design reuse oriented partial retrieval of CAD models. *Computer-Aided Design*, 42(12):1069–1084, December 2010.

[6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[7] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[8] Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 185–192, 2004.

[9] Sergey Brin. Near neighbor search in large metric spaces. *very large data bases*, pages 574–584, 1995.

[10] Gerth Stølting Brodal and Leszek Gasieniec. Approximate dictionary queries. In *CPM '96 Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching*, pages 65–74, 1996.

[11] Andrei Z. Broder. Identifying and filtering near-duplicate documents. *combinatorial pattern matching*, pages 1–10, 2000.

[12] A.Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29, 1997.

[13] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: binary robust independent elementary features. In *ECCV'10 Proceedings of the 11th European conference on Computer vision: Part IV*, volume 6314, pages 778–792, 2010.

[14] Timothy Chappell, Shlomo Geva, and Guido Zuccon. Approximate nearest-neighbour search with inverted signature slice lists. *european conference on information retrieval*, pages 147–158, 2015.

[15] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *1991 IEEE International Conference on Robotics and Automation Proceedings*, pages 2724–2729 vol.3, April 1991.

[16] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008.

[17] Thomas T. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 2009.

[18] Dimitrios Dimou and Konstantinos Moustakas. Fast 3D Scene Segmentation and Partial Object Retrieval Using Local Geometric Surface Features. In Marina L. Gavrilova, C.J. Kenneth Tan, and Alexei Sourin, editors, *Transactions on Computational Science XXXVI: Special Issue on Cyberworlds and Cybersecurity*, Lecture Notes in Computer Science, pages 79–98. Springer, Berlin, Heidelberg, 2020.

[19] Guoguang Du, Mingquan Zhou, Congli Yin, Juan Zhang, Zhongke Wu, and Wuyang Shui. An automatic positioning algorithm for archaeological fragments. In *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1*, VRCAI '16, pages 431–439, New York, NY, USA, December 2016. Association for Computing Machinery.

[20] Helin Dutagaci, Afzal A. Godil, A. Axenopoulos, P. Daras, T. Furuya, and R. Ohbuchi. SHREC 2009 - Shape Retrieval Contest of Partial 3D Models. *Eurographics Workshop on 3D Object Retrieval(2009)*, March 2009.

[21] T Furuya, S Kurabe, and R Ohbuchi. Randomized Sub-Volume Partitioning for Part-Based 3D Model Retrieval. *3DOR '15 Proceedings of the 2015 Eurographics Workshop on 3D Object Retrieval*, pages 15–22, 2015.

[22] Takahiko Furuya and Ryutarou Ohbuchi. Learning part-in-whole relation of 3D shapes for part-based 3D model retrieval. *Computer Vision and Image Understanding*, 166:102–114, January 2018.

[23] Afzal Godil, Helin Dutagaci, Benjamin Bustos, Sunghyun Choi, Suchuan Dong, Takahiko Furuya, Haisheng Li, Norman Link, A Moriyama, Rafael Meruane, et al. Range scans based 3d shape retrieval. *Proceedings of the 2015 Eurographics Workshop on 3D Object Retrieval*, pages 153–160, 2015.

[24] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, Jul 2012.

[25] Herve Jegou, Matthijs Douze, Cordelia Schmid, and Patrick Perez. Aggregating local descriptors into a compact image representation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, 2010.

[26] Guillaume Lavoué. Combination of bag-of-words descriptors for robust partial shape retrieval. *The Visual Computer*, 28(9):931–942, September 2012.

[27] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, 2011.

[28] Zhouhui Lian, Afzal Godil, Xianfang Sun, and Jianguo Xiao. Cm-bof: visual similarity-based 3d shape retrieval using clock matching and bag-of-features. In *Machine Vision and Applications archive*, volume 24, pages 1685–1704, 2013.

[29] Zhen-Bao Liu, Shu-Hui Bu, Kun Zhou, Shu-Ming Gao, Jun-Wei Han, and Jun Wu. A

Survey on Partial Retrieval of 3D Shapes. *Journal of Computer Science and Technology*, 28(5):836–851, September 2013.

[30] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[31] Moritz G. Maaí and Johannes Nowak. Text indexing with errors. *Journal of Discrete Algorithms*, 5(4):662–681, 2007.

[32] Mohammad Norouzi, Ali Punjani, and David J. Fleet. Fast search in hamming space with multi-index hashing. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3108–3115, 2012.

[33] R. Ohbuchi and T. Furuya. Scale-weighted dense bag of visual features for 3D model retrieval from a partial view 3D model. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 63–70, September 2009.

[34] Ryutarou Ohbuchi, Kunio Osada, Takahiko Furuya, and Tomohisa Banno. Salient local visual features for shape-based 3D model retrieval. In *2008 IEEE International Conference on Shape Modeling and Applications*, pages 93–102, June 2008.

[35] Panagiotis Papadakis, Ioannis Pratikakis, Theoharis Theoharis, and Stavros Perantonis. PANORAMA: A 3D Shape Descriptor Based on Panoramic Views for Unsupervised 3D Object Retrieval. *International Journal of Computer Vision*, 89(2):177–192, September 2010.

[36] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV'10 Proceedings of the 11th European conference on Computer vision: Part IV*, volume 6314, pages 143–156, 2010.

[37] Sai Manoj Prakhya, Bingbing Liu, and Weisi Lin. B-shot: A binary feature descriptor for fast and efficient keypoint matching on 3d point clouds. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1929–1934. IEEE, 2015.

[38] Ioannis Pratikakis, Michalis A. Savelonas, Fotis Arnaoutoglou, George Ioannakis, Anestis Koutsoudis, Theoharis Theoharis, Minh-Triet Tran, Vinh-Tiep Nguyen, V.-K. Pham, Hai-Dang Nguyen, and et al. Shrec'16 track: Partial shape queries for 3d object retrieval. *Eurographics Workshop on 3D Object Retrieval*, page 10 pages, 2016.

[39] E. M. Reina, K. Q. Pu, and F. Z. Qureshi. An index structure for fast range search in hamming space. In *2017 14th Conference on Computer and Robot Vision (CRV)*, pages 8–15, 2017.

[40] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[41] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[42] Caitlin Sadowski and Greg Levin. Simhash: Hash-based similarity detection. *Technical report, Google*, 2007.

[43] Michalis A. Savelonas, Ioannis Pratikakis, and Konstantinos Sfikas. An overview of partial 3D object retrieval methodologies. *Multimedia Tools and Applications*, 74(24):11783–11808, December 2015.

[44] Michalis A. Savelonas, Ioannis Pratikakis, and Konstantinos Sfikas. Partial 3D Object Retrieval combining Local Shape Descriptors with Global Fisher Vectors. *3DOR '15 Proceedings of the 2015 Eurographics Workshop on 3D Object Retrieval*, 2015.

[45] Michalis A. Savelonas, Ioannis Pratikakis, and Konstantinos Sfikas. Fisher encoding of differential fast point feature histograms for partial 3d object retrieval. *Pattern Recognition*, 55:114–124, Jul 2016.

[46] Giuseppe Serra, Costantino Grana, Marco Manfredi, and Rita Cucchiara. Gold: Gaussians of local descriptors for image representation. *Computer Vision and Image Understanding*, 134:22–32, 2015. Image Understanding for Real-world Distributed Video Networks.

[47] Konstantinos Sfikas, Ioannis Pratikakis, Anestis Koutsoudis, Michalis Savelonas, and Theoharis Theoharis. Partial matching of 3D cultural heritage objects using panoramic views. *Multimedia Tools and Applications*, 75(7):3693–3707, April 2016.

[48] Daniel Sieger and Mario Botsch. The polygon mesh processing library, 2020. http://www.pmp-library.org.

[49] I. Sipiran, R. Meruane, B. Bustos, T. Schreck, H. Johan, B. Li, and Y. Lu. SHREC'13 Track: Large-Scale Partial Shape Retrieval Using Simulated Range Images. *Eurographics 2013 Workshop on 3D Object Retrieval*, page 8 pages, 2013.

[50] Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure, 2019.

[51] Julien Tierny, Jean-Philippe Vandeborre, and Mohamed Daoudi. Partial 3D Shape Retrieval by Reeb Pattern Unfolding. *Computer Graphics Forum*, 28(1):41–55, 2009.

[52] Bart Iver van Blokland and Theoharis Theoharis. An indexing scheme and descriptor for 3d object retrieval based on local shape querying. *Computers & Graphics*, 92:55–66, 2020.

ISBN 978-82-326-6170-1 (printed ver.)
ISBN 978-82-326-5954-8 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU
Norwegian University of
Science and Technology