Quasi Spin Images

Bart Iver van Blokland   Theoharis Theoharis   Anne C. Elster

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.

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 [1] [2], and has been listed among the descriptors most robust to clutter, varying mesh resolution, and clutter [3]. 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 [4]. The fields of game physics, computational biophysics [5] and medical image processing [6] [7] are also good examples.

This paper was presented at the NIK-2018 conference; see http://www.nik.no/.

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 [3] [8]. 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 to 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 2 we outline the original spin image descriptor and some other related work. In Section 3 we introduce the novel quasi spin image descriptor. We finally evaluate the proposed method in Section 4.

2 Background

The spin image is a local histogram descriptor initially proposed by Johnson *et al.* [9]. 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 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.

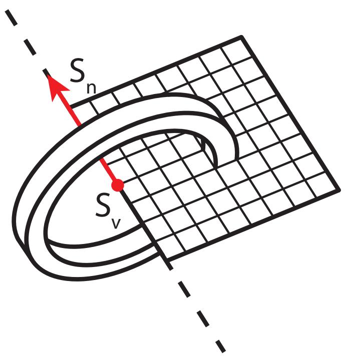

Figure 1: Spin image computation: the volume captured by rotating a spin image bin around the central axis. S_v and S_n are the spin vertex and the spin normal respectively. The dashed line represents the central axis.

Accurately calculating the area of a mesh intersecting the aforementioned torus-like volume (shown in Figure 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.* [10].

While the spin image descriptor has been shown to perform well in scenes with significant quantities of clutter [3] [8] [9], and has been used successfully in a number of applications [11] [12] [13] [14], 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 [9]. Additionally, comparing spin image pairs requires computing the Pearson correlation coefficient, which can be computationally costly. Dinh *et al.* [15] 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.* [16]. 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 [17] [18]. However, neither of these show implementations for generating spin images on the GPU.

3 Quasi Spin Images

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

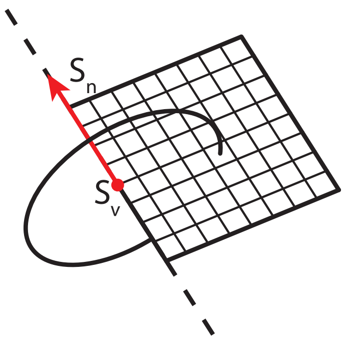

Figure 2: Quasi spin image computation: the circle captured by rotating a spin image bin around the central axis. S_v and S_n are the spin vertex and the spin normal respectively. The dashed line represents the central axis.

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.

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 2, as opposed to

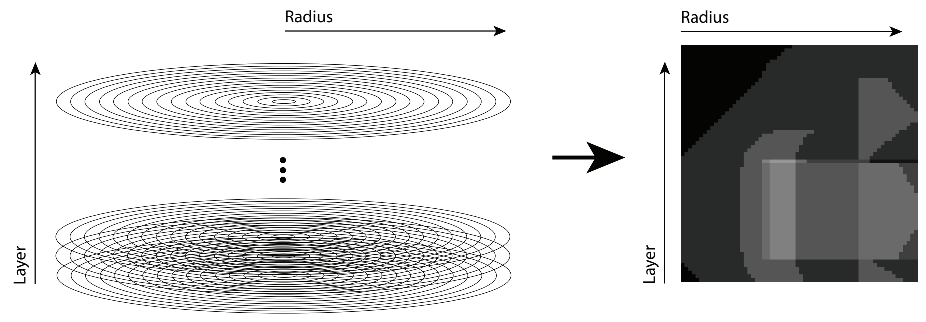

Figure 3: The correspondence between intersection circles and the produced QSI image.

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 3.

While the definition of the QSI deviates from the original SI, the images produced are similar visually, especially when compared to other descriptors. A visual comparison of QSI and SI is shown in Figure 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.

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.

QSI / SI Generation Execution Times on GPU and CPU

The performance of the SI and QSI generation implementations were tested by applying them on training models from the SHREC17 dataset [19]. 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 [20]. To the best of our knowledge, this implementation is the fastest one available today.

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.* [9].

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 1 computes the support radius S_r, in a manner that satisfies the above requirements:

$$S_r = \frac{\sqrt[3]{BBox.x \cdot BBox.y \cdot BBox.z}}{2} \tag{1}$$

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 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 4 show that when generating 200,000 images, the GPU QSI implementation outperforms the reference CPU one by approximately a factor of 35.

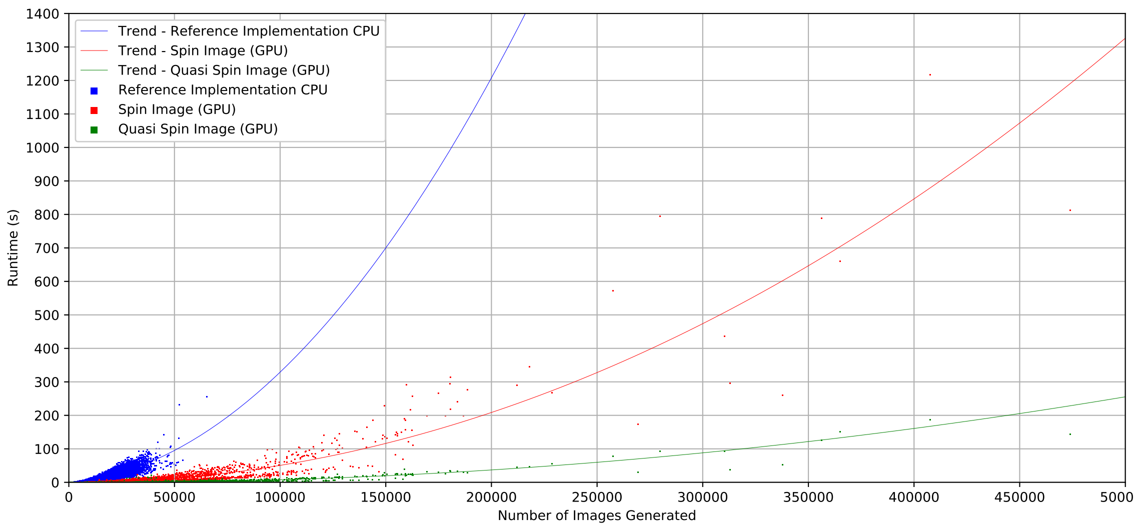

Figure 4: A comparison of the execution times of the proposed and reference implementations.

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.

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 [21] [22] [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 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 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 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

[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.

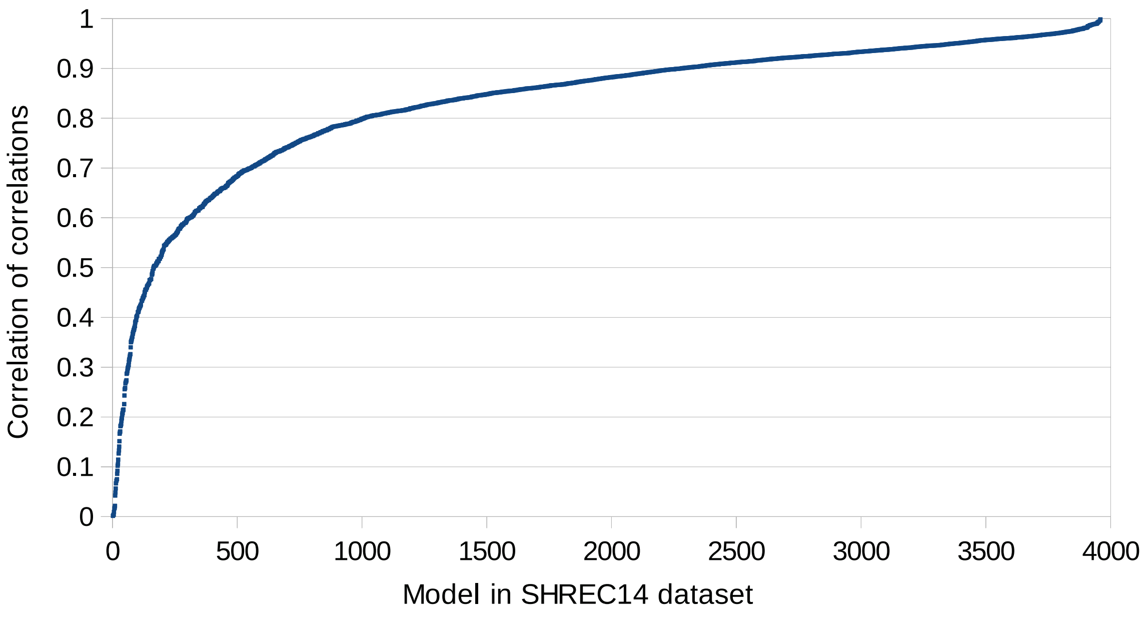

Figure 5: The correlation between the calculated correlations of SI and QSI for each model in the SHREC14 benchmark.

the model appear superimposed on the image. An example of this is the teapot handle visible on the top row of Figure 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 6c).

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.

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.

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

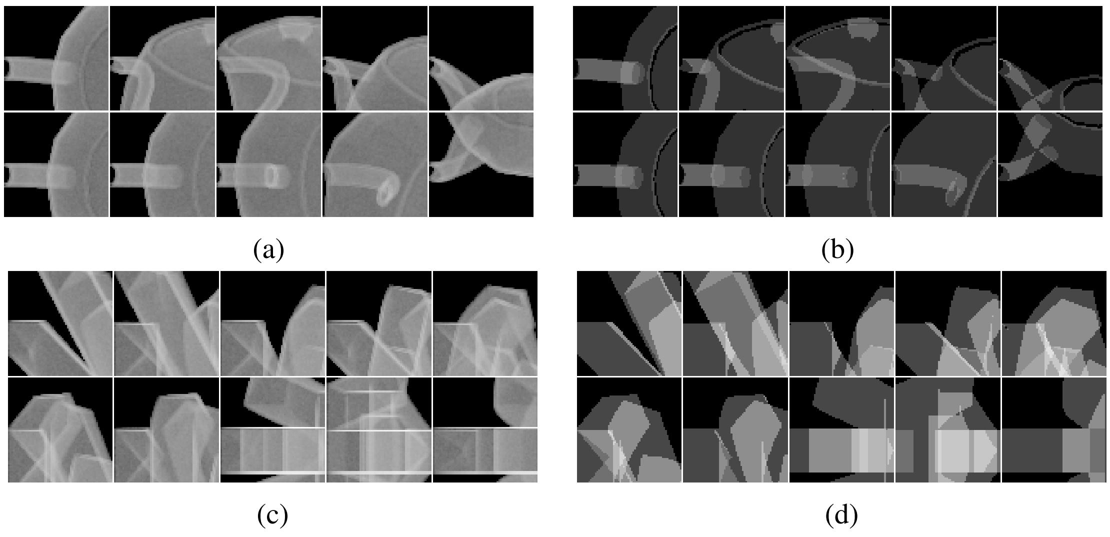
Figure 6: A visual comparison between the original SI (Figures 6a and 6c) and proposed QSI method (Figures 6b and 6d). All grayscale values have been logarithmically scaled for clarity.

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 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.

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 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.

References

[1] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3d local surface description and object recognition," *International journal of computer vision*, vol. 105, no. 1, pp. 63–86, 2013.

[2] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European conference on computer vision*. Springer, 2010, pp. 356–369.

[3] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3d local feature descriptors," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 66–89, 2016.

[4] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[5] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "Gpu computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, May 2008.

[6] G. Pratx and L. Xing, "Gpu computing in medical physics: A review," *Medical physics*, vol. 38, no. 5, pp. 2685–2697, 2011.

[7] E. Smistad, T. L. Falch, M. Bozorgi, A. C. Elster, and F. Lindseth, "Medical image segmentation on gpus–a comprehensive review," *Medical image analysis*, vol. 20, no. 1, pp. 1–18, 2015.

[8] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3d object recognition in cluttered scenes with local surface features: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2270–2287, 2014.

[9] 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*, vol. 21, no. 5, pp. 433–449, 1999.

[10] O. Carmichael, D. Huber, and M. 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*. IEEE, 1999, pp. 358–367.

[11] C. Conde, R. Cipolla, L. J. Rodríguez-Aragón, Á. Serrano, and E. Cabello, "3d facial feature location with spin images." in *MVA*. Citeseer, 2005, pp. 418–421.

[12] J. Assfalg, A. Del Bimbo, and P. 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*, vol. 3. IEEE, 2004, pp. 906–909.

[13] S. 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*. IEEE, 2016, pp. 620–623.

[14] R. E. Breighner, D. R. Holmes III, S. Leng, K.-N. An, C. H. McCollough, and K. 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*, vol. 4, no. 6, pp. 360–367, 2016.

[15] H. Q. Dinh and S. Kropac, "Multi-resolution spin-images," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 863–870.

[16] J. Assfalg, M. Bertini, A. Del Bimbo, and P. Pala, "Content-based retrieval of 3-d objects using spin image signatures," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 589–599, 2007.

[17] N. Davis, D. Braunreiter, C. Tebcherani, and M. Tanida, "3d object matching on the gpu using spin-image surface matching," in *Advanced Signal Processing Algorithms, Architectures, and Implementations XVIII*, vol. 7074. International Society for Optics and Photonics, 2008, p. 707408.

[18] A. R. Gerlach and B. K. Walker, "Accelerating robust 3d pose estimation utilizing a graphics processing unit," in *Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*, vol. 7878. International Society for Optics and Photonics, 2011, p. 78780V.

[19] M. Savva, F. Yu, H. Su, A. Kanezaki, T. Furuya, R. Ohbuchi, Z. Zhou, R. Yu, S. Bai, X. Bai, M. Aono, A. Tatsuma, S. Thermos, A. Axenopoulos, G. T. Papadopoulos, P. Daras, X. Deng, L. Zhouhui, B. Li, H. Johan, Y. Lu, and S. Mk, "Shrec17 track large-scale 3d shape retrieval from shapenet core55," in *Proceedings of the Eurographics Workshop on 3D Object Retrieval*, 2017.

[20] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[21] B. Li, Y. Lu, C. Li, A. Godil, T. Schreck, M. Aono, M. Burtscher, Q. Chen, N. K. Chowdhury, B. Fang *et al.*, "A comparison of 3d shape retrieval methods based on a large-scale benchmark supporting multimodal queries," *Computer Vision and Image Understanding*, vol. 131, pp. 1–27, 2015.

[22] A. A. Godil and C. Li, "Shrec14 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.