Computers & Graphics xxx (2017) xxx–xxx

Contents lists available at ScienceDirect

Computers & Graphics

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

Special Issue on CAD/Graphics 2017

Exclusive grouped spatial hashing

Q1 Weiwei Duan, Jianxin Luo*, Guiqiang Ni, Bin Tang, Qi Hu, Yi Gao

PLA University of Science and Technology, Haifu Lane #1 Nanjing, China

ARTICLE INFO

Article history:
Received 15 June 2017
Revised 15 August 2017
Accepted 15 August 2017
Available online xxx

Keywords:
Exclusive grouped hashing
Multidimensional hashing
Spatial data
Repetitive data

ABSTRACT

A novel multidimensional hashing scheme, named the Exclusive Grouped Spatial Hashing (EGSH), which compresses repetitive spatial data into several compact tables while retaining efficient random access, is presented. EGSH represents a multi-level hashing without any losses. Moreover, EGSH compresses a group of repetitive elements into the same entry of the hash tables, while it uses a coverage table to mark the corresponding hash tables of the compressed data. Although, prior hashing work is related to hash collisions mitigation, here a full use of these collisions is obtained and therefore the spatial data compression rate is improved. The performance of exclusive grouped spatial hashing is presented in 2D and 3D graphic examples.

1. Introduction

The compressing and storing of spatial data represents a fundamental issue in computer graphics. Many graphics applications involve spatial data that generally include a large number of repetitive elements. For instance, 2D and 3D textures are represented by spatial data which are often repetitive. A proper compromise between efficient storage and access performance of spatial data has become a hot research topic. Traditional hash algorithms typically perform sequential probes into the hash table. The varying number of probes per query leads to GPU inefficiency, due to the SIMD (Single Instruction Multiple Data) parallelism all threads wait for the worst-case number of probes.

In 2006, Lefebvre and Hoppe proposed for the first time the use of a GPU to access a hash table using a perfect hashing [1]. The proposed method has a constant look-up time and simple access to the data from the GPU. However, the hash table construction is expensive because the item location depends on the locations of previous items. In the perfect hashing scheme, all defined items should be packed into different locations of the hash table. In other words, even the repetitive items should be stored in different entries of the hash table.

Uniform spatial partitioning data structures, such as quadtrees and octrees, often provide efficient storage for repetitive data compression. These data structures usually contain unused entries in their hierarchies, and maintain a costly sequence of parent-to-child pointer links. Choi et al. [2] proposed a linkless octree to encode the subdivided nodes using a perfect hashing without explicit parent-to-child pointer links. However, the hierarchical structures are still inefficient for random node access on the GPUs.

In this paper, we present a novel hashing scheme named Exclusive Grouped Spatial Hashing (EGSH) that efficiently compresses repetitive data into tiny compact hash tables without losses while maintaining simple random access to the GPUs. The term “Grouped” refers to repetitive data with the same values which are considered as a group of elements. A group of repetitive elements are compressed into the same entry of the hash tables. The term “Exclusive” denotes that each entry of the hash tables should store maximum one group of elements, i.e. elements of different groups cannot be packed into the same entry. The term “Spatial” means that hashing is used for point queries in multidimensional datasets, which can be efficiently implemented for GPUs.

In order to store the repetitive data efficiently, a sequence of similar multidimensional hash functions, which iteratively pack defined elements into hash tables were defined by:

$$h_i(p) = M_0 p \bmod \overline{m}_i, 1 \leqslant i \leqslant k \tag{1}$$

where the parameter k represents the total number of iterations. In the ith iteration, the hash function $h_i(p)$ maps defined elements to the hash table whose size, labeled as m_i, depends on the number of groups in the uncompressed data. For each entry q in the hash table, we select an element value whose attributive group has the most repetitive elements amongst all data mapped to the position q. Thus, each entry of the hash table can replace as many repetitive elements as possible. The uncompressed elements are pushed into the next iteration. EGSH compresses all repetitive data using several tiny hash tables, which enables efficient random access to the GPUs.

---

* Corresponding author.
E-mail address: luojianxin555@163.com (J. Luo).

http://dx.doi.org/10.1016/j.cag.2017.08.012
0097-8493/© 2017 Published by Elsevier Ltd.

Please cite this article as: W. Duan et al., Exclusive grouped spatial hashing, Computers & Graphics (2017), http://dx.doi.org/10.1016/j.cag.2017.08.012

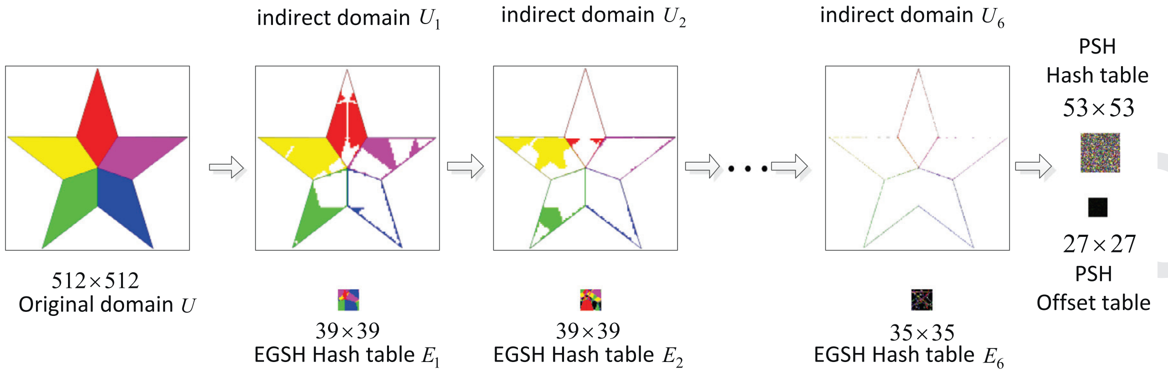

Fig. 1. Representation of exclusive grouped hashes for repetitive spatial data compression in a 2D image. The original 512^2 domain U contains a set of 86,885 pixels, which can be divided into 1507 groups according to their values. Exclusive grouped hashes were used to iteratively compress as many repetitive data as possible into the same entry of small hash tables with a size of $39^2 (= 1521 > 1507)$ until the uncompressed data became sparse enough. Then, the remaining uncompressed sparse data were packed into the indirect domain U_6 using additional perfect spatial hashing.

The main contributions of this work are the use of very small hash tables to store repetitive data, and the retainment of fast random access to the GPUs. Previous research was mostly based on hash collisions mitigation, here we present a novel many-to-one hashing scheme that uses these collisions. For each entry in the hash tables, one position can replace a large number of repetitive elements, which generally can be equal to dozens, hundreds or be even larger depending on the application scenario. The hash tables do not contain unused entries and the construction process can be precomputed on static spatial data. EGSH greatly reduces storage requirements without any losses. Moreover, EGSH is simple to understand and easy to implement. When EGSH is implemented in the GPU, only three shader instructions are needed to achieve efficient random access.

Since the paper focuses on constant-time access compression, EGSH was compared with the perfect spatial hashing (PSH) scheme proposed by Lefebvre and Hoppe [1]. The experiments on different graphic datasets have shown that two schemes have similar lookup times, while EGSH has a better performance of both the construction time and storage requirement.

2. Related work

Perfect hashing. Hash tables are commonly used in computer graphics. Since the mid-eighties, many studies have been dedicated to the realization of perfect hash tables [3–7]. These approaches were very complex, which caused unreasonable space and time costs. Thus just hundreds of elements could be used in practice. The first practical scheme that achieved a good average-case performance on large datasets was presented by Fox et al. [8]. They ordered the search for hash functions based on the degree of the vertices in a graph that represented word dependencies.

Sager [4] presented a hash to map string keys. The hash contained three respective hash functions together with two auxiliary tables. Lefebvre and Hoppe [1] extended the basic framework of Sager [4] to 2D and 3D spatial spaces. Different from Sager [4], Lefebvre and Hoppe used only one single auxiliary table and two respective hash functions. They packed the sparse spatial data into a compact hash table using a perfect hashing for the first time. By achieving an efficient access to hash tables for GPUs, it provided a far-reaching influence on later hashing research. Since the item location depends on the locations of previous items, the inherent sequential construction of hash tables is very costly. Since each hash entry stores only one element, the perfect hashing does not have a good performance on highly repetitive data.

Parallel hashing. Based on the well-known Cuckoo hash proposed by Pagh and Rodler [9], Alcantara et al. [10] presented the first real-time hashing scheme with parallel hash table construction in the GPU. Since the cuckoo hashing is performed within a small and fast on-chip memory, hash tables can be constructed and accessed at interactive rates, outperforming the previous sequential construction schemes. Due to the iterative insertion of elements into hash tables, in the case of collision, already inserted keys are removed. Since the hash construction might cause a failure especially at high load factors, they restart the process and rechoose a new hash function.

García et al. [11] introduced a new parallel hashing by exploiting spatial data coherence. They adapt the Robin Hood hashing of [12] for the quick rejection of empty keys. Their scheme provides a high load factor and fast access with a very low failure rate.

However, previous hashing schemes do not consider the issue of repetitive data compression in the GPU. The repetitive elements are still stored on different positions in the hash tables.

Spatial partitioning. In the standard computer graphic technique, the spatial partitioning structures, such as quadtrees and octrees, are commonly used for image/volume encoding and compression, especially when the spatial datasets have a highly repetitive structure. However, the parent-to-child pointer linkers, produced by subdividing each spatial cell into child cells, leads to a waste of storage.

To increase the efficiency, researchers [13,14] used spatial hashing techniques to encode quadtrees and octrees. Andrysco and Tricoche [15] presented pointerless octrees in 2010. Choi et al. [2] created a linkless octree without storing explicit parent-to-child links. Their method encodes the subdivided nodes using a perfect spatial hashing while retaining coarse-to-fine hierarchical structures. Scandolo et al. [16] proposed a lossless compression scheme for high-resolution data based on sparse quadtree encoding. Using the local choosing of the most prominent value, they produced a sparse encoding in the form of a hierarchy and obtained high compression rates.

Laine and Karras [17] presented a sparse voxel octree (SVO) which can efficiently carve out empty space. Kampe et al. [18] constructed a directed acyclic graph (SVDAG) from an SVO by simply merging identical subtrees. As the SVDAG allows nodes to share pointers to identical regions of space, the storage cost has been improved. Villanueva et al. [19] made an extension of the sparse voxel DAG. They showed a symmetry-aware sparse voxel DAG (SSVDAG) by merging subtrees that are identical up to a similarity transform. Dado et al. [20] and Dolonius et al. [21] both showed how to apply DAG compression to non-binary data. These approaches have a

Please cite this article as: W. Duan et al., Exclusive grouped spatial hashing, Computers & Graphics (2017), http://dx.doi.org/10.1016/j.cag.2017.08.012