Behavioral Spherical Harmonics for Long-Range Agents' Interaction

Biagio Cosenza(✉)

Embedded Systems Architecture (AES),
Department of Computer Engineering and Microelectronics (TIME),
Technische Universität Berlin, Berlin, Germany
cosenza@tu-berlin.de

Abstract. We introduce behavioral spherical harmonic (BSH), a novel approach to efficiently and compactly represent the directional-dependent behavior of agent. BSH is based on spherical harmonics to project the directional information of a group of multiple agents to a vector of few coefficients; thus, BSH drastically reduces the complexity of the directional evaluation, as it requires only few agent-group interactions instead of multiple agent-agent ones. We show how the BSH model can efficiently model intricate behaviors such as long-range collision avoidance, reaching interactive performance and avoiding agent congestion on challenging multi-groups scenarios.

Furthermore, we demonstrate how both the innate parallelism and the compact coefficient representation of the BSH model are well suited for GPU architectures, showing performance analysis of our OpenCL implementation.

Keywords: Spherical harmonics · Behavioral model · Agent-based simulation · Long-distance interaction · Collision avoidance · GPGPU

1 Introduction

Agent-Based Simulations (ABSs) include a broad range of domains. Many existing ABS frameworks offer general solutions to common problems shared between different agent domains; however, most challenging problems still stick with domain-specific solutions. An example is long-range interaction between agents, required by many real-case scenarios, but usually solved with context-specific solutions. In crowd simulation, where a crowd is modeled as a set of agents, each agent attempts to reach a target position while avoiding collisions with other agents and static obstacles in the environment. Considering the interaction of all pairs of agents becomes expensive in large crowds, and typically only neighboring agents that lie within a specified radius are take into account, limiting the possibility of look-ahead behaviors. More complex collision avoidance approaches use two algorithms: one for local collision and one for global, long-distance collision. Continuum-based approaches for collision avoidance [16] offer

S. Hunold et al. (Eds.): Euro-Par 2015 Workshops, LNCS 9523, pp. 392–404, 2015.
DOI: 10.1007/978-3-319-27308-2_32

an interesting solution, but they assume that agent directionality is homogenous for agents in the same cell grid; otherwise, important directional information is lost. Therefore, similar solutions are not portable between domains: they work only in contexts where directionality information can be lost without consequences or at least coarsely approximated.

In this paper, we introduce behavioral spherical harmonic (BSH), a novel behavioral model that encodes directionality information usually expressed in an agent's behavior in a compact mathematical formulation. For this purpose, we use spherical harmonics (SH): with a projection step, multiple agent directions are projected to a small set of coefficients. SH functions have useful properties and can be easily combined (e.g., multiplied). Once a new agent needs to calculate his directional-dependent behavior, the BSH can easily reconstruct the directionality information from previously projected agents.

The contribution of this paper are:

- A novel behavioral model (BSH) based on Spherical Harmonics that compactly encodes the directional information of multiple agents
- An application of BSH to support long-range interactions
- A GPU/OpenCL implementation that exploits the natural parallelism and the compact representation of BSH, as tested with several interactive agent simulations on an NVIDIA GPU.

2 Background

Researchers and practitioners of agent-based simulations and modeling have for a long time investigated the use of parallel implementations targeting a wide range of architectures, including multi-cores [15,17], GPUs [6,19], and distributed memory architectures [2–4].

Collision avoidance algorithms have been investigated by several ABS systems, where the motion of each agent is typically governed by some high-level formulation and local interaction rules (e.g., collision avoidance). ClearPath [12] presents a local collision avoidance algorithm that formulates the conditions for collision-free navigation as a quadratic optimization problem. PLEdestrians [11] introduces a bio-mechanically, energy-efficient trajectory for each individual in a multi-agent simulation. Local collision avoidance algorithms in crowd simulation often ignore agents beyond a neighborhood of a certain size. However, this cutoff can result in sharp changes in trajectory when large groups of agents enter or exit these neighborhoods. Such long-range interaction requires more computation than local collision avoidance, even for distributed and parallel simulation. HybridCrowd [7] performs approximate, long-range collision avoidance via two approaches: low-density crowds are modeled with discrete methods, while high-density crowds exploit continuum methods.

Spherical harmonics (SH) is a frequency-space basis for representing functions defined over the sphere. SH has been used in various problems, such as the heat equation, electrical fields, gravitational fields, and modeling the quantum

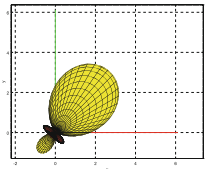
angular momentum of electrons. SH has also been used in computer graphics applications: Cabral et al. [1] first used SH to estimate the integral of the BRDF by expanding the bidirectional reflection coefficient in SH. More recently, Kaplanyan and Dachsbacher [13] used a lattice of harmonical functions to propagate indirect illumination in real-time.

Our approach is similar to a continuum approach where agents' positions and velocities are accumulated in a background grid [16]. However, in our case, velocities are encoded as SH, to eliminate difficulties such as the sensitivity to the number of agent goals [22] or overcrowding in highly dense crowds [16].

3 Behavioral Spherical Harmonics

Traditional agent models such as Reynold's boid model represent an agent with a position and a direction vector. However, cases such as those involving long-range interaction approaches require a way to represent this information for a group of agents (e.g., in a grid-based continuum approach, a group is represented by all agents inside a cell). While agents' positions can be easily approximated by exploiting their locality, directionality information is very hard to approximate: in fact, many existing long-range approaches lose any directionality information.

Table 1. A two-dimensional example of BSH: only original agent directions (a) are projected into SH basis (b) to calculate a vector of coefficients (c).



$s_1(-2,1,0)$,	$d_1(0.45,0.89,0)$	$d_1(0.45,0.89,0)$	
$s_2(-1,-1,0)$,	$d_2(0.45,0.89,0)$	$d_2(0.45,0.89,0)$	
$s_3(-1,1,0)$,	$d_3(0.89,0.45,0)$	$d_3(0.89,0.45,0)$	
$s_4(0,0,0)$,	$d_4(0.62,0.78,0)$	$d_4(0.62,0.78,0)$	$\rightarrow c(1.69, 2.15, 0.0, 1.87,$
$s_5(-2,0.5,0)$,	$d_5(0.78,0.62,0)$	$d_5(0.78,0.62,0)$	$2.91, 0.0, -1.89, 0.0, -0.42)$
$s_6(1,0.5,0)$,	$d_6(0.64,0.77,0)$	$d_6(0.64,0.77,0)$	
(a)		(b)	(c)

In this work, we replace the directionality information of a group of agents with a set of coefficients that represents a spherical function (Table 1). We introduce behavioral spherical harmonics (BSH), a novel approach to represent the directionality information of a group of agents in a compact and efficient way. This section introduces the mathematical background of spherical harmonics, how directions are projected into coefficients and reconstructed from them, and the most interesting properties of BSH. Our use of SH is similar to related work in rendering [9,20] (e.g., we use real-value basis), therefore adopt a similar notation.

3.1 Spherical Harmonics

The spherical harmonics (SH) is a mathematical system analogous to the Fourier transform but defined across the surface of a sphere. SH is the angular portion of the solution to Laplace's equation in spherical coordinates and defines an orthonormal basis over the sphere. In general, SH functions are based on imaginary numbers, but we are only interested in approximating real functions over the sphere (i.e., agent directions). Reference to SH functions are synonymous with references to real spherical harmonic functions.

Given the standard parametrization into spherical coordinates of points on the surface of a unit sphere, the Cartesian coordinates of the point (x, y, z) can be expressed in spherical coordinates using the variables θ and φ:

$$(x, y, z) \rightarrow (\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta)$$

The SH function is traditionally represented by the symbol y

$$y_l^m(\theta,\varphi) = \begin{cases} \sqrt{2}K_l^m \cos(m\varphi)P_l^m(\cos\theta), & m > 0 \\ \sqrt{2}K_l^m \sin(-m\varphi)P_l^{-m}(\cos\theta), & m < 0 \\ K_l^0 P_l^0 \cos(\theta), & m = 0 \end{cases}$$

where P is the associated Legendre polynomial and K_l^m the scaling normalization constants defined as

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}$$

l and m are integer constant indexes, also called the *order* or *band* (l), and the degree (m), respectively. To generate all the SH orthonormal basis functions, the parameters l and m are defined slightly differently from the Legendre polynomials: l is still a positive integer starting from 0, but m takes signed integer values from $-l$ to l. It is possible to think of the SH functions occurring in a specific order so that we can use a simpler one-dimensional vector:

$$y_l^m(\theta,\varphi) = y_i(\theta,\varphi) \text{ where } i = l(l+1) + m$$

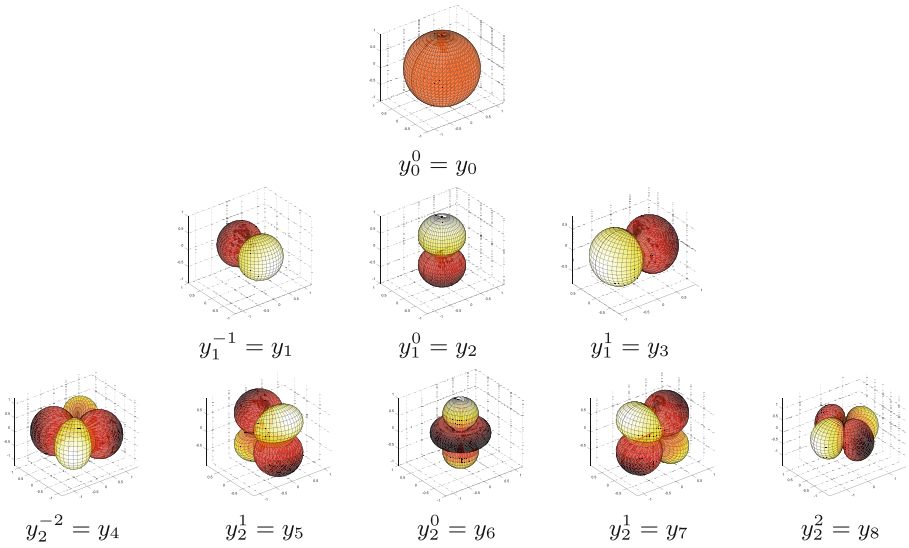
In this paper, we use 3rd-order SH representation, therefore using only 9 basis functions as shown in Table 2; a list of the first 5 SH bands are available in [20].

3.2 Projection and Reconstruction

Let f be the spherical function we want to project to SH coefficients (we will show later how f will represent directional information of multiple agents). To calculate a single coefficient for a specific band, we integrate the product of the function $f(d)$, where d is the direction, with the SH basis function y:

$$c_l^m = \int f(d)y_l^m(s)ds$$

Table 2. Spherical harmonic basis functions for the first three bands.



To reconstruct the function f at a given direction d, we define $\hat{f}(d)$ by performing the reverse process and summing scaled copies of the corresponding SH functions:

$$\hat{f}(d) = \sum_{l=0}^{n} \sum_{m=-l}^{l} c_l^m y_l^m(d) = \sum_{i=0}^{n^2} c_i y_i(d)$$

which is increasingly accurate as the number of bands n increases[1]. Note that the projection to nth order SH generates n^2 coefficients.

Evaluating the approximate function $\hat{f}$ at direction d simply requires the dot product between the n^2 coefficients and the vector of evaluated basis functions $y_i(d)$. The first coefficient ($i = 0$), called DC term, represents the average value of the function over the sphere.

Properties. SH functions have many interesting properties. SH bases are orthogonal, which means that the inner product of any two distinct basis functions is zero. An integration performed over the product of two SH functions is the same as a dot product of their coefficients as a result of the SH basis functions' othonormality. To project a scalar function $f(s)$ defined over a sphere S against the basis function of the SH requires a simple integration over S: $c_l^m = \int f(s) y_l^m(s) ds$. SH functions are also rotationally invariant. Other properties such as the double and triple product integral and the double product projection are covered by [20].

[1] We use n for the number of bands and N for the number of agents.

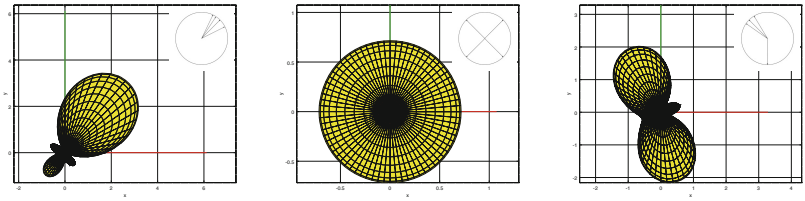


Fig. 1. Three different agent group orientations resulting in different SH functions.

3.3 Behavioral Spherical Harmonics

We apply SH functions to a group of agents whose directions will be replaced by an approximated function $\hat{f}(d)$. In a simulation using grid-based space subdivision, SH is applied for each cell, whose agent directions will be replaced by a vector of coefficients. Expressing directionality as per-cell spherical functions has different advantages. First, a single SH function replaces all agent directions contained in the cell, thus requiring far fewer interaction evaluations: once the agent directions are encoded into coefficients, successive evaluations are independent of the number of agents. Second, SH functions are more expressive than other representations, as they effectively approximate all directional information for the projected directions. For example, SH can distinguish different agent orientations, as depicted in Fig. 1:

Long-Range Collision Avoidance. Earlier agent models such as Reynolds' boid model have no long range collision avoidance: collision is local as it reacts to boids (agents) within a limited neighborhood. In this paper we introduce a behavioral model (BSH) that also uses agent's direction (i.e., velocity) to implement long-range agent interactions. BSH is calculated by summing the local collision-avoidance model plus an *avoidance* direction that expresses a long-range repulsion force. This force depends on the distance between the agent and the cell, the angle between agent's direction and the cell, and the precalculated cell's directional information. As we will show later, the directional contribution is critical to avoid congestion in long-range, multi-group scenarios (see Figs. 3, 4 and 5).

Collision Avoidance with Static Obstacles. BSH is also able to model obstacles, in the same way we express agent directionality in grid cells. This work uses a simple approach: to project obstacle information within the SH if each cell contains an obstacle. For each obstacle, depending on its shape and how agents should avoid it, a number of directional avoidance forces are projected to the SH function. For simplicity, the obstacle is placed in the center of the cell. As obstacles are static their contributions to their cells' respective SH are precalculated. Evaluating the avoidance force also takes in account of the (square of the) distance between the agent and the cell.

4 Implementation

We have implemented our agent-based simulation based on BSH in OpenCL [14] and tested on a GPU. It comprises 8 steps:

1. Calculate each agent's grid position;
2. Calculate each agent's grid hash and index;
3. Sort key-value pair (grid hash, index);
4. Reorder position and velocity vectors by sorted index and store the cell start and end indices;
5. Project velocity vectors into SH representation for each cell;
6. Simulate pairwise agent interaction;
7. Evaluate SH in the direction of agent and reconstruct velocity;
8. Draw agents (optional, requires OpenGL binding).

Steps 1–4 and 6 are common in many GPU-based agent simulations that use spatial hashing (grid) and are briefly discussed in the next section. Our approach introduces two new steps, 5 and 7, discussed in Sects. 4.2 and 4.3. Step 8 is straightforward and required only when visualization is enabled.

4.1 Spatial Subdivision with Grid

Naïve implementations compute the pairwise interaction between all agents, which has a complexity of $O(N^2)$ for N agents. A more efficient approach uses spatial subdivision such as the three-dimensional uniform grid, which divides the simulation space in uniformly sized, non-overlapping cells. Each agent is assigned to exactly one cell, depending on its position. The cell position is then hashed to give it a unique, sortable, consecutively-numbered value and stored a key-value pair with the agent's *id* as key and the cell position hash as value. This approach has been largely used in literature [5,6,8,18]. Our implementation of the spatial grid is similar to the NVIDIA particle simulation implementation [10]: steps 1–4 and 6 are essentially the same. The pairwise interaction between agents in step 6, as for related work, happens only if they are at close range: each agent looks at other agents in its own cell and the 26 surrounding cells (steps 5 and 7 take long-range interaction into account). The sorted index of the agents is used to reorder position and velocity data. To quickly locate the agents in a cell, two additional buffers containing the reordered agent's start and end indices, respectively, are needed. Additional details can be found in [10].

Agent movement is restricted to the simulation space limited by a bounding box. A negative force is used to keep agents from moving out of the box by keeping them away from the borders.

4.2 Projection of Directionality into SH Coefficients

The first additional kernel is the projection of each agent direction into a SH representation. For this purpose, we used the 3rd-order, numerical evaluation

code from Sloan [21], which produces 9 coefficients. The coefficient from band 0 is a constant value and is not actually stored in the output vector. The evaluation returns an OpenCL $float8$ with the coefficients for a specific point on the sphere.

```
__kernel void SHEval3(float4* vel_in, float8* SHCoeff)
{
  float4 vel = vel_in[get_global_id(0)];
  float8 pSH;
  float fC0,fC1,fS0,fS1,fTmpA,fTmpB,fTmpC;
  float fZ2 = vel.z*vel.z;
  pSH.s1 = 0.4886025119029199f*vel.z;
  pSH.s5 = 0.9461746957575601f*fZ2 + -0.3153915652525201f;
  fC0 = vel.x;
  fS0 = vel.y;
  fTmpA = -0.48860251190292f;
  pSH.s2 = fTmpA*fC0;
  pSH.s0 = fTmpA*fS0;
  fTmpB = -1.092548430592079f*vel.z;
  pSH.s6 = fTmpB*fC0;
  pSH.s4 = fTmpB*fS0;
  fC1 = vel.x*fC0 - vel.y*fS0;
  fS1 = vel.x*fS0 + vel.y*fC0;
  fTmpC = 0.5462742152960395f;
  pSH.s7 = fTmpC*fC1;
  pSH.s3 = fTmpC*fS1;
  SHCoeff[get_global_id(0)] = pSH;
}
```

The result vector with the coefficients from the evaluation is then weighted by the scalar values of each component of an agent's direction vector. The weighted SH is summed up per cell using an optimized reduction kernel.

4.3 Reconstruction of the Avoidance Direction

BSH calculates the final avoidance force by evaluating the contribution of all cell coefficients along with those of other factors. The reconstruction kernel loads the SH coefficients for each cell; to improve performance, all agents in the same OpenCL workgroup move the coefficients into the fast local memory (i.e., loop blocking).

The avoidance force also accounts for the distance from the agent to the cell position (i.e., the center of the cell), and the direction of the agent itself. This calculation is performed on the SH coefficients and exploits some of the SH properties we have seen before. First, the contribution of c_{cell} is weighted to be inversely proportional to the distance from a to c with a scalar-vector product

$$\boldsymbol{c_d} = \frac{1}{\|\boldsymbol{p_a} - \boldsymbol{p_{cell}}\|} \boldsymbol{c_{cell}}$$

where p_a and p_c are, respectively, the position of a and c.

Successively, the resulting coefficients c_d are multiplied by a harmonical representation of the agent's direction vector (i.e., the avoidance force is stronger when the cell is in front of the agent and zero when behind it). Let $\boldsymbol{c_a}$ be the coefficients of agent directional SH functions. We recall that the product of two

SH functions is equal to the dot product of the two coefficient vectors; therefore our avoidance force for agent a and cell c is

$$R_{a \leftrightarrow c} = \hat{f}(d_{a \leftrightarrow c}) = \sum_{i=0}^{n} c_a(i) \cdot c_d(i) y_i(d_{a \leftrightarrow c})$$

For each agent a, the final long-distance avoidance force is calculated by summing up the contributions for all cells c in C:

$$R_a = \sum^{c \in C} R_{a \leftrightarrow c}$$

and is added to the other agent rules (e.g., local collision avoidance).

5 Results

We tested our BSH implementation on a machine equipped with an AMD FX-6300 3.5Ghz processor and an NVIDIA GTX 960 GPU. The OpenCL code has been executed on the GPU, together with the OpenGL rendering code comprising vertex, fragment and geometry shaders. We tested different agent boid scenarios where our long-distance BSH-based model replaced the classical local-only Reynold's model. Test benchmarks focused on two aspects: performance and model evaluation. For performance, we investigated the scalability of the approach, comparing alternative bruteforce non-local approaches and analyzing the per-kernel runtime. For model evaluation, we compared BSH with a local model in known test scenarios where the lack of long-distance interaction leads to agent congestion.

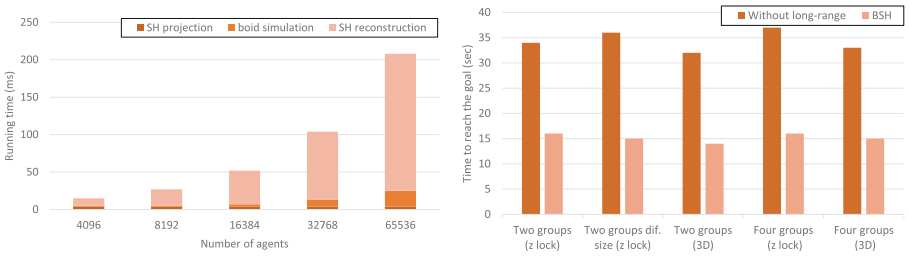


Fig. 2. Per-kernel runtime for the two-group z-lock scenario (left) and simulation runtime time for five tested scenarios (right) (Color figure online).

Per-Kernel Analysis and Agent Scalability. We first tested the per-kernel performance of our implementation on a crowd-like, two-group agent scene where agents were constrained to move only on the xy plane (z lock). Agents are initially positioned randomly in two spheres in the simulation space. Every agent has an assigned goal position at the opposite side of the simulation space and

tries to move toward it. The initial directions (velocities) of all agents are set at a fixed value and aligned in the direction of the goal.

The spatial subdivision works better if the agent density is uniformly distributed over the uniform grid. In our test cases, density is higher at the position of the initial and final placement therefore performance degrades at these points. The reconstruction kernel performance is limited by the global memory access, an aspect greatly improved by our local memory cell-prefetching strategy. From this analysis, we conclude that there is still space for improvement, e.g., by changing the spatial subdivision and handling empty cells, which do not contribute to the result but are currently loaded into the memory.

Figure 2 (left) shows the time spent in SH construction and evaluation with different sizes. BSH long-range interaction complexity is $O(kN)$ instead of $O(N^2)$, where N is the number of agents and k the number of cells.

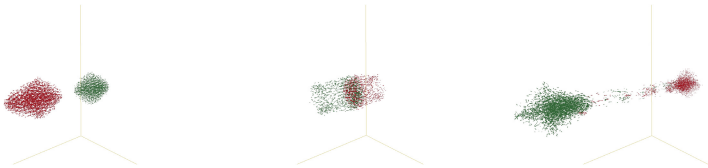


Fig. 3. Two-groups scenario without long-range interaction. $t = 3s$, $10s$ and $20s$ (Color figure online).

Behavioral Comparison with Local Models. We tried to replicate the same test benchmark of Golas et al. [7], where diametrically opposed groups try to move in opposite directions in space. Early detection of distant groups, combined with knowledge of agent direction, enable them to avoid the congestion in the center of the space. Tests were performed in both 3D and 2D (z axis locked) space, as the latter is both common in crowd simulations and more challenging for congestion because of agents' lack of freedom. We repeated the same test with two and four groups; runtimes to reach the goal (simulation times) are shown in Fig. 2 (right).

Figure 3 shows that, in the two-group scenario without long-range interaction, a congestion happens at time $t = 10s$. Figure 4, instead, shows respectively two equally sized (top) and different sized (bottom) groups that diametrically try to move in opposite directions using the BSH model. By enabling our BSH-based long-range interaction, each group reaches the target point in 15 s and no congestion happen.

In the more challenging four-group scenario, shown in Fig. 5: four groups of agents in a circular position try to reach the opposite position. Early, long-distance obstacle detection avoids the congestion so that each group *rotates* around the center instead of having a congestion there. We obtained interactive performance and reproduced the behavior modeled in [7] with a much simpler

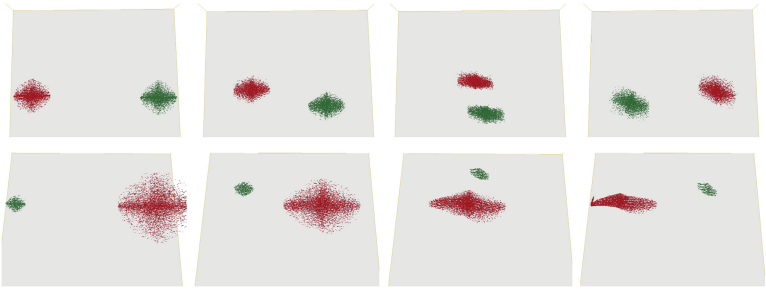


Fig. 4. BSH-based long-distance collision avoidance with two groups (z-lock). Equally sized (top) and unequal groups (bottom) are shown (Color figure online).

method than the original study's mixed approach, which used both continuum and discrete methods.

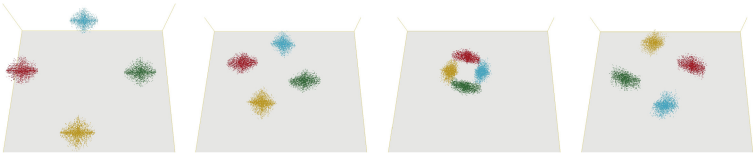


Fig. 5. BSH-based long-distance collision avoidance with four groups (z-lock) (Color figure online).

BSH is also quite flexible and can adapt to various obstacle-avoidance scenarios. We implemented two experimental scenarios where obstacle avoidance, typically based on vector fields, was replaced with few SH functions. Figure 6 illustrates the two examples discussed here. The advantages with respect of approaches based on vector fields is that they require few coefficients to describe an obstacle. For examples, the column from Fig. 6 (left) spans exactly one cell on the xz plane and 10 cells on the y axis; therefore, this scenario needs only 10 SH functions to simulate a repulsive, avoidance force for the three columns. The tunnel is modeled with directional SH repulsing only agents from one direction.

Fig. 6. Two obstacle avoidance scenarios: (left) two groups of agents avoiding three columns and (right) two one-way tunnels where agents are allowed to enter only the tunnel on their respective right side.

6 Conclusion

Behavioral spherical harmonics (BSH) is a novel approach for representing directional-dependent agent behaviors. BSH exploits the use of spherical harmonics to model the directional contribution of multiple agents with an approximated spherical function, which only requires a small coefficient vector. The approach is effective in drastically reducing the complexity of the directional evaluation, as it requires only a few agent-cell interactions instead of multiple interactions between agents.

We tested our BSH model on a long-range collision detection scenario. Our GPU parallel implementation reached interactive performance and replicated the test benchmark of state-of-the art approaches using a 3rd-order BSH that replaced all directional per-cell information with only 9 coefficients. BSH are also flexible enough to simulate more advanced obstacle collision avoidance scenarios (instead of classical vector fields-based solutions) and can be easily implemented on existing distributed and parallel simulation frameworks.

The source code is available under the BSD simplified license[2].

References

1. Cabral, B., Max, N., Springmeyer, R.: Bidirectional reflection functions from surface bump maps. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987, pp. 273–281 (1987)
2. Cordasco, G., De Chiara, R., Mancuso, A., Mazzeo, D., Scarano, V., Spagnuolo, C.: A framework for distributing agent-based simulations. In: Alexander, M., D'Ambra, P., Belloum, A., Bosilca, G., Cannataro, M., Danelutto, M., Di Martino, B., Gerndt, M., Jeannot, E., Namyst, R., Roman, J., Scott, S.L., Traff, J.L., Vallée, G., Weidendorfer, J. (eds.) Euro-Par 2011, Part I. LNCS, vol. 7155, pp. 460–470. Springer, Heidelberg (2012)
3. Cordasco, G., Milone, F., Spagnuolo, C., Vicidomini, L.: Exploiting D-Mason on parallel platforms: a novel communication strategy. In: Lopes, L., Žilinskas, J., Costan, A., Cascella, R.G., Kecskemeti, G., Jeannot, E., Alexander, M., Hunold, S., Scott, S.L., Lankes, S., Lengauer, C., Carretero, J., Breitbart, J., Cannataro, M., Ricci, L., Benkner, S., Petit, S., Scarano, V., Gracia, J. (eds.) Euro-Par 2014, Part I. LNCS, vol. 8805, pp. 407–417. Springer, Heidelberg (2014)
4. Cosenza, B., Cordasco, G., Chiara, R.D., Scarano, V.: Distributed load balancing for parallel agent-based simulations. In: 19th International Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 62–69 (2011)
5. Erra, U., Frola, B., Scarano, V.: BehaveRT: a GPU-based library for autonomous characters. In: Boulic, R., Chrysanthou, Y., Komura, T. (eds.) MIG 2010. LNCS, vol. 6459, pp. 194–205. Springer, Heidelberg (2010)
6. Erra, U., Frola, B., Scarano, V., Couzin, I.: An efficient GPU implementation for large scale individual-based simulation of collective behavior. In: International Workshop on High Performance Computational Systems Biology (HIBI), pp. 51–58 (2009)

[2] http://bcosenza.github.io/bsh.

7. Golas, A., Narain, R., Curtis, S., Lin, M.C.: Hybrid long-range collision avoidance for crowd simulation. IEEE Trans. Visual. Comput. Graphics **20**(7), 1022–1034 (2014)
8. Grasso, I., Ritter, M., Cosenza, B., Benger, W., Hofstetter, G., Fahringer, T.: Point distribution tensor computation on heterogeneous systems. Procedia Comput. Sci. (ICCS) **51**, 160–169 (2015)
9. Green, R.: Spherical harmonic lighting: the gritty details. In: GDC, vol. 56 (2003)
10. Green, S.: Particle simulation using cuda. Technical report, NVIDIA (2010)
11. Guy, S.J., Chhugani, J., Curtis, S., Dubey, P., Lin, M., Manocha, D.: Pledestrians: a least-effort approach to crowd simulation. In: ACM SIGGRAPH/EG Symposium on Computer Animation, pp. 119–128 (2010)
12. Guy, S.J., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., Dubey, P.: Clearpath: highly parallel collision avoidance for multi-agent simulation. In: ACM SIGGRAPH/EG Symposium on Computer Animation, pp. 177–187. SCA (2009)
13. Kaplanyan, A., Dachsbacher, C.: Cascaded light propagation volumes for real-time indirect illumination. In: ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, pp. 99–107. I3D (2010)
14. Khronos Group: Khronos Group. The OpenCL 2.0 specification (2014)
15. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: Mason: a multiagent simulation environment. Simulation **81**(7), 517–527 (2005)
16. Narain, R., Golas, A., Curtis, S., Lin, M.C.: Aggregate dynamics for dense crowd simulation. In: ACM SIGGRAPH Asia, pp. 122:1–122:8 (2009)
17. North, M.J., Collier, N.T., Vos, J.R.: Experiences creating three implementations of the repast agent modeling toolkit. ACM Trans. Model. Comput. Simul. **16**(1), 1–25 (2006)
18. Passos, E.B., Joselli, M., Zamith, M., Clua, E.W.G., Montenegro, A., Conci, A., Feijo, B.: A bidimensional data structure and spatial optimization for supermassive crowd simulation on GPU. Comput. Entertain. **7**(4), 60:1–60:15 (2010)
19. Richmond, P., Walker, D.C., Coakley, S., Romano, D.M.: High performance cellular level agent-based simulation with FLAME for the GPU. Briefings Bioinform. **11**(3), 334–347 (2010)
20. Sloan, P.P.: Stupid Spherical Harmonics (SH) Tricks (2008)
21. Sloan, P.P.: Efficient spherical harmonic evaluation. J. Comput. Graphics Tech. (JCGT) **2**(2), 84–90 (2013)
22. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. ACM Trans. Graph. **25**(3), 1160–1168 (2006)