Optimal grid size for random fixed radius ball collision detection

Pratham Gohil

December 16, 2023

Overview

A random fixed radius ball collision is a model where there are balls of same radius generated on a space and the collisions between the balls have to be decided. The naive way of doing this is comparing each ball to every other ball which results in $O(n^2)$ time complexity. We can partition the space into grids so that only balls close by are compared leading to many computation increases. This can even result in $O(n)$ expected time complexity. However this requires the gird box size to be the radius of the balls. If the balls are small, the grid size will be too large and The computation will increase as the algorithm has to traveerse between each grid cell. In this paper, we calculate the optimal grid size so that the grid isn't too large to cause inefficiencies in traversal, and neither too small to result in $O(n^2)$ time complexity.

Algorithm

To detect the collisions, we will have to check for collisions in each adjacent cells. In 1-dimensional case, there are two adjacent cells for all the mid cells. We could instead check the collisions between current cell and the next cell instead of checking for previous cell as well. Doing this will reduce computation. This leaves 2 adjacent cell collision checking for 1-dimensional case and 4 and 8 adjacent cell collision checking for 2-dimensional and 3-dimensional cases respectively. The algorithm performs $n * (n - 1)$ collision detection calculations for n balls selected.

Modelling the Problem

We begin with a simple 1-dimensionl case with N grid partitions and K particles. Here every ball is given the Random Variable $X_i \in \{1, 2, \ldots, N\}$ denoting the position of a balls in the grid, where $i \in \{1, 2, \ldots, K\}$ is the ball number.

The number of balls in each cell is:

$$P_k = \sum_{i \in \{1,2,\ldots,K\}} (X_i == k) \qquad k \in \{1, 2, \ldots, N\}$$

Here $(X_i == k)$ denotes:

$$(X_i == k) = \begin{cases} 1 & X_i = k \\ 0 & X_i \neq k \end{cases}$$

Hence, $\mathbb{P}[X_i == k]$ denotes the probability of ith ball to be in kth partition. In this case, we'll consider this probability to be constant for all partitions:

$$\mathbb{P}[X_i == k] = \frac{1}{N}$$

We take the access time for each partition to be T_p and time for each collision detection between two balls to be T_d. These will have to be calculated either by experimentation or analysing assembly code. According to the algorithm, we'll have to perform C_p number of partition access and C_d number of collision detection calculations. Here,

$$C_p = N$$

and,

$$C_d = \sum_{k \in \{1,2,\ldots,N-1\}} \left[\left(\sum_{i \in \{1,2,\ldots,K\}} ((X_i == k) + (X_i == k+1)) \right) \left(\sum_{i \in \{1,2,\ldots,K\}} ((X_i == k) + (X_i == k+1)) - 1 \right) \right]$$

The Total time taken is denoted as T,

$$T = T_p * C_p + T_d * C_d$$

So the expected value of T is:

$$\begin{aligned} \mathbb{E}[T] &= T_p * \mathbb{E}[C_p] + T_d * \mathbb{E}[C_d] \\ \mathbb{E}[C_p] &= N \\ \mathbb{E}[C_d] &= \frac{4K(K-1)(N-1)}{N^2} \end{aligned}$$

Hence,

$$\text{1-dimensional}: \ \mathbb{E}[T] = T_p * [N] + T_d * \left[\frac{4K(K-1)(N-1)}{N^2} \right]$$

$$\text{2-dimensional}: \ \mathbb{E}[T] = T_p * [NM] + T_d * \left[\frac{16K(K-1)(N-1)(M-1)}{N^2M^2} \right]$$

$$\text{3-dimensional}: \ \mathbb{E}[T] = T_p * [NMO] + T_d * \left[\frac{64K(K-1)(N-1)(M-1)(O-1)}{N^2M^2O^2} \right]$$

Optimizing

We find the minima of the $\mathbb{E}[T]$ for certain N, NM and NMO value for 1-diimensional, 2-dimensional and 3-dimensional cases respectively. Also, we assume $N - 1 \approx N$

$$\frac{\partial \mathbb{E}[T]}{\partial N} = 0 = T_p - T_d * \left[\frac{4K(K-1)}{N^2}\right]$$

Hence, optimal value of $\mathbb{E}[T]$ occurs at:

$$\text{1-dimensional}: \ N = 2\sqrt{K(K-1)\frac{T_d}{T_p}}$$

$$\text{2-dimensional}: \ NM = 4\sqrt{K(K-1)\frac{T_d}{T_p}}$$

$$\text{3-dimensional}: \ NMO = 8\sqrt{K(K-1)\frac{T_d}{T_p}}$$

And when we substitute these values, we get:

$$\text{1-dimensional}: \ \mathbb{E}[T] = 4\sqrt{K(K-1)T_dT_p}$$

$$\text{2-dimensional}: \ \mathbb{E}[T] = 8\sqrt{K(K-1)T_dT_p}$$

$$\text{3-dimensional}: \ \mathbb{E}[T] = 16\sqrt{K(K-1)T_dT_p}$$

This shows that the time complexity of the algorithm approaches $O(K)$.

Conclusion

The expected time complexity of the collision detection algorithm approached $O(K)$ given the values of T_d and T_p are known. If not, they have to be calculated.