애니메이션 속도에 무관한 충돌 탐지 알고리즘
(An Animation Speed-independent Collision Detection Algorithm)

김 형 석 †
(Hyoung Seok Kim)

요 약 본 논문에서는 애니메이션 속도에 무관한 충돌 탐지 알고리즘을 제안한다. 현재까지 개발된 대부분의 충돌 탐지 알고리즘들은 점진적(incremental) 알고리즘들로서, 현 시점에서의 가까운 점(근점)을 찾기 위하여 이전 시점의 근점 주위를 먼저 찾는다. 그런데 만일 움직이는 물체가 충돌 반응에 의해서 큰 토크를 받게 된다면 회전 속도가 증가하게 되어, 다음 시점에서의 실제 근점은 현 시점에서의 근점과는 매우 동떨어져 있어 엉뚱한 위치에서 근점을 찾게 되는 단점을 가진다. 그러므로, 최악의 경우에는 각 시점에서 $O(n^2)$ 시간이 소요될 수 있다. 또한 애니메이션 속도에 따라 이러한 점진적 계산 회수가 변하게 되어 전체적인 알고리즘의 소요 시간이 변하게 되는 단점을 가지고 있다. 본 논문에서는 이러한 문제점을 근본적으로 해결하고자 새로운 방법을 제안하고자 한다. 먼저, 기하학 특성을 내포하는 구면 근점 다이아그램을 생성하고, 이를 이용하여 두 물체간의 단일 거리 함수를 생성한다. 충돌 시점을 효율적으로 찾기 위해서 구간 뉴튼 방법을 거리함수에 적용한다.

키워드 : 충돌 탐지, 구면 근점 다이아그램

Abstract This paper presents an efficient collision detection algorithm the performance of which is independent of animation speed. Most of the previous collision detection algorithms are incremental and discrete methods, which find out the neighborhood of the extreme vertex at the previous time instance in order to get an extreme vertex at each time instance. However, if an object collides with another one with a high torque, then the angular speed becomes faster. Hence, the candidate by the incremental algorithms may be farther from the real extreme vertex at this time instance. Therefore, the worst time complexity may be $O(n^2)$, where n is the number of faces. Moreover, the total time complexity of incremental algorithms is dependent on the time step size of animation because a smaller time step yields more frequent evaluation of Euclidean distance. In this paper, we propose a new method to overcome these drawbacks. We construct a spherical extreme vertex diagram on Gauss Sphere, which has geometric properties, and then generate the distance function of a polyhedron and a plane by using this diagram. In order to efficiently compute the exact collision time, we apply the interval Newton method to the distance function.

Key words : Collision Detection, Spherical Extreme Vertex Diagram

1. 서 론

충돌 탐지 문제는 계산기하학, 로보틱스, 컴퓨터 게임 등에서 필요로 하는 보편적인 문제들이다[1-6]. 충돌 여부를 파악하기 위해서는 물체들간의 효과적인 거리 계산이 정확한 충돌 지점과 시점을 찾는데 핵심적인 역할을 한다. 계산기하학 분야에서는 평면과 3차원 볼록 다면체간의 유클리드 거리 계산 문제가 근점(extreme vertex) 문제로서 잘 알려져 있다. 즉, 평면이 다면체 방향으로 평행 이동할 때 처음으로 만나게 되는 꼭지점을 찾는 문제이다.

Edelsbrunner는 이러한 근점 문제를 전처리 과정으로 $O(n)$의 시간과 공간을 투자하여 $O(\log n)$에 해결하는 알고리즘을 제안하였다[7]. 여기서 n은 다면체 P에 속하는 꼭지점들의 개수이다. 이 알고리즘은 볼록 다면체들간의 포함관계를 설명하는 계층적 자료구조를 사용함에 의해서 그 효율성을 증대시킬 수 있었다. 그러나 이러한 계층적 구조는 고정된 물체들 간의 거리 계산에서는 효율적이지만, 주어진 시간 동안에 움직이는 물체의 근점들의 리스트를 파악하는 데에는 효과적이지 못하다.

· 본 연구는 동의대학교 학술기금(2002AA167)지원으로 수행되었음
† 정 회 원 : 동의대학교 멀티미디어공학과 교수
hskim@deu.ac.kr
논문접수 : 2003년 9월 26일
심사완료 : 2004년 1월 9일

또한 전체적인 소요 시간은 애니메이션의 프레임 속도에 따라 변하게 되는데, 이는 유클리드 거리 계산 횟수가 애니메이션 프레임간의 시간 간격에 반비례함에서 기인한다. 또한, 다면체와 평면과의 명확한 거리 함수를 생성할 수 없다는 단점을 가지고 있는데, 이는 본 논문에서 제시하는 새로운 알고리즘과 구별되는 내용이기도 하다.

보다 일반화된 근점 문제에 대한 해결책이 움직이는 다면체와 평면과의 충돌 탐지 알고리즘에 중요한 역할을 할 것이다. 대부분의 충돌 탐지 알고리즘들은 가장 가까운 점을 찾기 위하여 볼록성과 지역 유사성을 이용하는 이산적(discrete) 기법들이다[8-10]. 이러한 점진적(incremental) 알고리즘들은 $t_0+\Delta t$에서의 근점을 찾기 위해 t_0에서의 근점에 이웃한 꼭지점들을 그 대상으로 여겨 그 이웃한 꼭지점들의 유클리드 거리를 계산한다. 그런데 이러한 방법은 물체의 회전 속도가 애니메이션 시간 간격에 비해서 상대적으로 일정한 한계 이내에 놓여 있다는 가정에서 가능하다. 그래서, 만일 움직이는 물체가 충돌 반응에 의해서 큰 토크를 받게 되어 회전 속도가 크게 증가한다면, $t_0+\Delta t$에서의 실제 근점은 t_0에서의 근점과는 동떨어진 꼭지점인 반면, 이 알고리즘은 t_0에서의 근점의 이웃한 꼭지점들 중에서 현 시점의 근점을 찾게 되는 단점이 있다. 그러므로, 이 시점에서의 근점을 찾기 위해서는 적어도 $O(n^2)$와 같은 시간이 소요됨을 알 수 있으며 따라서 알고리즘이 전체적으로 안정되지 않음을 알 수 있다. 또한, 점진적인 전략을 유지하면서 정확한 근점을 찾기 위해서는 애니메이션 시간 간격을 줄일 수밖에 없게 되는데, 이 방법 또한 안정적이지 못하다[11].

본 논문에서는 강체 운동을 하고 있는 물체에 대한 근점들의 집합(sequence)을 효율적으로 찾는 알고리즘을 제시하며 이를 이용하여 거리함수를 효율적으로 구하는 알고리즘을 개발한다. 이를 위해서 고정되어 있는 물체에 대한 근점 문제와 움직이는 물체에 대한 근점 문제를 동시에 효율적으로 풀기 위하여 구면 상[12]에 근점 다이아그램을 구성하고 이를 이용한다. 이러한 근점 다이아그램을 이용하여 움직이는 물체의 주어진 평면에 대한 거리를 $O(n)$의 선처리 과정을 거쳐 $O(\log n)$ 시간에 구할 수 있다[13-16]. 또한 시간이 지남에 따라 그 근점이 변하게 되는데 이러한 근점들의 시퀀스 v_i's, $i=1,\cdots,m$을 근점 다이아그램의 기하학적 특성을 고려하여 구할 수 있고, 이를 이용하여 꼭지점 v_i가 근점이 되는 구간 $[t_i, t_{i+1}]$를 계산할 수 있다. 그러므로, 그 구간에서 두 물체의 거리 계산은 $O(1)$ 시간에 가능하며, 그 구간에서의 v_i의 움직임만 추적하게 되면 그 구간에서의 두 물체의 거리 함수를 만들 수 있다. 따라서, 본 논문에서 제시되는 방법이 Edelsbrunner 방법보다 효율적임을 알 수 있다. 이렇게 전 부분 구간에 대한 거리 함수를 만들게 되면 전체적인 거리함수를 얻게 되는데, 두 물체의 충돌 시점은 그 거리함수가 처음으로 zero 값을 갖게 되는 시점이므로 이를 효율적으로 구하기 위해서 interval Newton 방법[17]을 사용한다.

본 논문의 구성은 다음과 같다. 2절에서는 본 논문에서 제시되는 알고리즘의 기본 아이디어인 근점 다이아그램에 대해서 설명하는데, 이를 이용하여 두 물체의 가까운 점(근점) 찾는 문제를 구면상의 point location 문제로 전환할 수 있게 됨을 보인다. 3절에서는 쿼터니언 곡선에 관한 간단한 고찰과 물체의 움직임을 표현하는 구면상의 곡선을 살펴본다. 4절에서는 두 물체의 거리함수를 생성하는 과정을 소개한다. 이러한 방법에 의한 결과를 5절에서 보인다.

2. 근점 문제

P를 n개의 반공간(half-space)의 교집합으로 정의되는 볼록 다면체라고 하자. 그러면 P의 표면은 n개의 면 F_i, $i=1,\cdots,n$으로 구성되며, 각 면 F_i는 법선 단위 벡터 nF_i를 가진다. P가 평면 H에 멀리 떨어져 있다고 가정을 한다면, $P \cap H = \emptyset$. 이때, H의 법선 단위 벡터를 nH라고 표기하며 이 벡터의 방향은 P가 놓여 있지 않는 다른 반공간을 가리킨다고 가정한다(그림 1 참조).

h_1을 평면 H에 속하는 점이라고 한다면 평면 H는 $\langle x-h_1, nH\rangle=0$와 같이 정의된다. 이때, $\langle\cdot,\cdot\rangle$은 두 벡터의 내적을 의미한다. 여기에서 3차원 공간을 정의역으로 하며 실수 값을 갖는 하나의 함수 $D(p)=\langle p-h_1, nH\rangle$를 정의하자. 점 p가 다면체 P가 놓여 있는 반공간에 속한다면 음수 값을 가지며, 반대편에 있다면 양수 값을 가지게 된다. 다면체 P가 nH의 반대편에 있기 때문에 P의 모든 꼭지점들은 모두 음수값을 갖는다. 여기에서 H에 대한 P의 근점 v_h는 평면으로부터

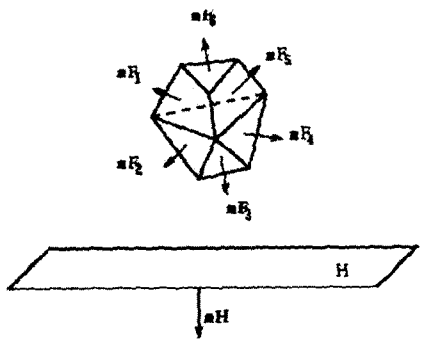


그림 1 다면체와 평면에서 정의되는 단위 법선 벡터

가장 작은 거리를 갖기 때문에 v_h는 다음을 만족한다.

$$\langle v_h - h_1, nH\rangle = \min\{\ |\langle v_i - h_1, nH\rangle|\ ,\ i=1,\cdots,m\}$$
$$= \max\{\ \langle v_i - h_1, nH\rangle\ ,\ i=1,\cdots,m\ \}$$

이때, m은 P에 있는 꼭지점들의 개수이다. 이는 다음 수식과 동치가 된다.

$$\langle v_h, nH\rangle = \max\{\ \langle v_i, nH\rangle\ ,\ i=1,\cdots,m\ \}$$

그러므로, 근점 문제는 P의 꼭지점들에 의해 정의되는 $\langle x, nH\rangle$ 값 중에서 최대값을 구하는 문제로 전환된다. 이러한 최대치 문제는 하나의 선형 프로그래밍 문제로서 이에 대한 일반적인 해법이 널리 알려져 있다[18].

본 논문에서는 이러한 근점을 보다 효율적으로 찾기 위하여, 점-평면 이중성(point-plane duality)을 이용한다. 먼저 점 $p=(p_1, p_2, p_3)$를 공간상에 있는 하나의 평면 $\langle p, x\rangle = p_1x_1 + p_2x_2 + p_3x_3 = 1$ 으로 대응하는 변환 T를 생각하자[18]. 이 변환의 역 또한 정의되는데 평면의 방정식이 $q_1x_1 + q_2x_2 + q_3x_3 = 1$인 평면은 점 $q=(q_1, q_2, q_3)$로 대응됨을 알 수 있다. 따라서 이러한 변환을 볼록 다면체에 적용하면 새로운 볼록 다면체인 D를 얻을 수 있다. P의 각 꼭지점 v_h는 D의 하나의 면 Dv_h에 대응되며 P의 각 면 F_h는 D의 하나의 꼭지점 DF_h에 대응됨을 알 수 있다(그림 2 참조). 일반성 유지를 위해서 다면체 P는 원점을 내포한다고 가정할 수 있다. 만약, P가 원점을 포함하지 않는다면 좌표계를 전체적으로 평행이동을 하면 된다.

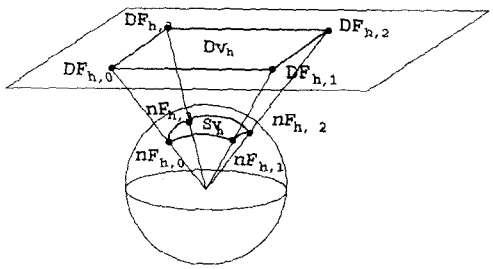


그림 2 이중 다면체의 하나의 면 D_{v_h}과 구면 근점 영역 Sv_h

볼록 다면체 P의 하나의 꼭지점인 $v_h=(x_h, y_h, z_h)$에 이웃한 면이 k개 있다면, 이를 반시계 방향으로 순서를 매겨 면들의 사이클 $(F_{h,1}, F_{h,2}, \cdots F_{h,k})$을 구성할 수 있다. 이때, $F_{h,i}$와 $F_{h,i+1}$는 P의 하나의 선분을 공유하고 있다. 꼭지점 v_h는 변환에 의해서 다음의 평면 $\langle v_h, x\rangle = x_hx_1 + y_hx_2 + z_hx_3 = 1$으로 대응된다. $nF_{h,j}$가 다면체 P의 면 $F_{h,j}$의 단위 법선 벡터이기 때문에 $F_{h,j}$를 포함하는 평면은 다음의 방정식을 만족한다.

$$\langle nF_{h,j}, x\rangle = d_{h,j} \text{ or } \langle \frac{nF_{h,j}}{d_{h,j}}, x\rangle = 1,$$

이때, $d_{h,j}$는 원점으로부터 면 $F_{h,j}$까지의 거리이다. 또한 v_h가 면 $F_{h,j}$의 꼭지점이기 때문에

$$\langle v_h, \frac{nF_{h,j}}{d_{h,j}}\rangle = 1.$$

이는 곧, $DF_{h,j} = nF_{h,j}/d_{h,j}$가 평면 $\langle v_h, x\rangle = 1$에 속한다는 사실을 알리고 있다. $DF_{h,j}$는 P의 면 $F_{h,j}$에 대응되며 이중성 다면체 D의 한 면 Dv_h의 꼭지점이다. 더군다나,

$$\langle v_h, x\rangle = \|v_h\| \langle \frac{v_h}{\|v_h\|}, x\rangle = 1 \text{ or } \langle \frac{v_h}{\|v_h\|}, x\rangle = \frac{1}{\|v_h\|}.$$

P의 꼭지점 v_h에 대응되는 볼록 다각형 Dv_h는 원점으로부터 거리가 $\frac{1}{\|v_h\|}$인 평면 $\langle v_h, x\rangle = 1$에 속함을 알 수 있다. 일반적으로, 변환 T는 원점으로부터 거리가 d인 점 P를 원점으로부터 거리가 $\frac{1}{d}$이며 원점을 시점으로 하며 점 P를 종점으로 하는 벡터에 수직인 평면으로 대응시킨다. 이는 곧, 다면체 D 또한 원점을 포함하는 볼록 다면체임을 뜻한다. 더군다나, Dv_h 또한 D의 한 면이 되고 형태는 볼록 다각형이다. 이러한 점-평면 이중성에 의해서, $DF_{h,j}$ 들은 볼록 다각형의 Dv_h의 꼭지점들이 되며 두 꼭지점 $DF_{h,j}$와 $DF_{h,j+1}$을 연결하는 선분 $DE_{h,j}$는 Dv_h의 하나의 선분이 된다. 원점에서 출발하여 $DF_{h,j}$를 향하는 사선(ray)은 가우스(Gauss) 구를 $nF_{h,j}$에서 만난다. 그러므로, Dv_h는 가우스 구면에 구면 영역 S_{v_h}로 투영된다. 이 구면 영역 S_{v_h}는 구면 점들과 구면 선분들에 의한 싸이클 $(nF_{h,0}, SE_{h,0}, nF_{h,1}, SE_{h,1}, \cdots, nF_{h,k-1}, SE_{h,k-1}, nF_{h,0})$로 표현될 수 있다. 이때, 사용되는 구면 선분 $SE_{h,j}$는 구면 점들 $nF_{h,j}$과 $nF_{h,j+1}$를 잇는 대원(great circle)의 부분(segment)이다. 본 논문에서는 이러한 구면 영역 S_{v_h}를 다면체 P의 꼭지점 v_h에 대응되는 *구면 근점 영역*(*spherical extreme region*)이라 한다. 볼록 다면체 P에 있는 모든 꼭지점들 v_i에 대응되는 구면 근점 영역들 S_{v_i}은 가우스 구면을 m개의 영역으로 분할(partition)하게 된다. 이러한 분할을 *구면 근점 다이아그램*(*spherical extreme vertex diagram*)이라 한다.

우리는 이러한 구면 근점 다이아그램을 이용하여 선형 프로그래밍 문제를 사선 투사(ray shooting) 문제로 전환할 수 있다. 즉, 원점에서 출발하고 nH를 향하는 사선 R이 만나는 볼록 다면체의 면 Dv_h를 구하는 것과

동일하다. m 개의 점 p_i, $i=1, \cdots, m$를 사선 R을 포함하는 직선 L과 다면체의 면 Dv_h를 포함하는 평면 H_i : $\langle v_i, x\rangle = 1$과의 교점이라 하고, d_i를 원점으로부터의 p_i 까지의 거리라고 하자. 그러면, 그 점들은 nH에 의해서 $p_i = d_i\ nH$ 로 표현된다. 각 점들은 역 변환에 의해서 원점까지의 거리가 $1/d_i$이며, 사선 R과는 수직인 평면 $T^{-1}(p_i)$: $\langle p_i, x\rangle = 1$에 변환된다. 즉,

$$\langle nH, x\rangle = \frac{1}{d_i}.$$

점 p_i가 평면 H_i에 포함되기 때문에 점 p_i는 다음의 평면의 방정식을 만족한다.

$$\langle nH, v_i\rangle = \frac{1}{d_i}. \quad (1)$$

따라서, P의 각 꼭지점 v_i는 평면 $T^{-1}(p_i)$: $\langle nH, x\rangle = 1/d_i$에 속하게 된다. 사선 R이 듀얼 면 Dv_h와는 점 p_h에 만난다고 가정하자. 듀얼 다면체 D가 원점을 내포하는 반공간의 교집합으로 구성되기 때문에 사선 R이 듀얼 면 Dv_h을 포함하는 평면 H_h와 교차한다고 생각하는 것과 동일하다. 따라서, p_h는 사선 R 위에 있는 모든 점 p_i 들 중에서 원점으로부터 가장 가까운 거리에 있는 점이다. 그러므로,

$$\langle v_h, nH\rangle = \frac{1}{d_h} \geq \frac{1}{d_i} = \langle v_i, nH\rangle, \text{ for all } i$$

따라서, 목적 함수 $z(x) = \langle x, nH\rangle$는 v_h에서 최대값을 가짐을 알 수 있다. 이는 곧, 선형 프로그래밍 문제(Linear Programming Problem)가 사선 투사 문제(Ray-Shooting Problem)로 전환됨을 보인 것이다.

이러한 듀얼 면 Dv_h를 효율적으로 찾기 위하여, 가우스 구면에 듀얼 다면체 D를 투영하여 얻어지는 구면 근접 다이아그램을 이용한다[18]. nH를 지나는 사선 R이 듀얼 면 Dv_h을 지나간다면, 그 사선 R은 Dv_h에 대응되는 구면 근점 영역 S_{v_h}를 nH를 통해 지나간다. 이는 곧, nH는 구면 근점 영역 S_{v_h}에 속함을 의미한다. 따라서, 사선 투사 문제(Ray-Shooting Problem)도 구면 위치 파악 문제(spherical point location problem)으로 전환될 수 있음을 의미한다. 이러한 구면 위치 파악 문제는 $O(\log n)$ 시간에 풀 수 있음은 본 연구자의 선행연구에서 알 수 있다[19]. 만약 nH가 구면 근점 다이아그램의 한 꼭지점 $nF_{h,j}$과 일치 하다면, 그 평면 H는 다면체 P의 한 면인 $F_{h,j}$와 평행한 상태를 의미한다. 이는 곧, 면 $F_{h,j}$의 모든 점들이 평면 H에 대한 근점(extreme vertex)들임을 알 수 있다. 만약 nH가 구면 근점 영역들 S_{v_h}와 S_{v_i} 사이를 공유하는 구면 선분 위에 놓여 있다면, S_{v_h}에 대응되는 꼭지점 v_h와 S_{v_i}에 대응되는 꼭지점 v_i 모두가 평면 H에 대한 근점 들임을 의미한다. 또한, 만약 nH가 구면 근점 영역 S_{v_h}에 속한다면, S_{v_h}에 대응되는 꼭지점 v_h가 그 시점에서의 근점임을 알 수 있다.

3. 쿼터니언과 구면 곡선

3.1 구면 곡선

다면체의 강체 운동은 일반적으로 평행이동과 회전운동으로 구성된다. H가 무한 평면이기 때문에 H에 대한 P의 근점은 두 물체의 평행이동에는 변하지 않는다는 성질을 가지고 있다. 그러므로, 근점들의 리스트를 조사하기 위해서는 다면체의 회전운동만을 고려하면 된다. 다면체 P의 회전운동을 설명하기 위해서 단위 쿼터니언 표기법을 사용한다[20-23]. 4차원에 속하는 단위 쿼터니언 Q는 실수부분과 3차원 벡터부분으로 나타내어진다. Q의 실수부분은 q_0로 표현되며, 벡터부분은 q로 표현되는데, 단위 쿼터니언은 간단하게 $Q=(q_0, q)$로 나타낸다. 이러한 쿼터니언을 사용하여 회전운동을 설명하도록 한다. 임의의 3차원 벡터 v를 회전축 r을 중심으로 회전각 $\|r\|$ 만큼 회전한 후 생성되는 벡터 v'은 다음의 수식으로 표현되어진다.

$$v' = QvQ^{-1}, \qquad Q = (\cos\frac{\|r\|}{2}, \frac{r}{\|r\|}\sin\frac{\|r\|}{2})$$

$Q(t)$를 시간의 흐름에 따라 변하는 다면체 P의 방향을 나타내는 단위 쿼터니언 함수라 하자. 다면체 P가 $Q(t)$를 따라 회전을 하는 반면, 무한 평면 H는 고정되어 있다. 무한 평면 H에 대한 다면체 P의 근점들은 H에 대한 P의 상관 방향(orientation)에 종속적이다. 그러므로, 위 상황을 상대적으로 P가 고정되어 있고 H가 $-Q(t)$ 만큼 회전한다고 생각할 수 있다. 따라서, 무한 평면의 단위 법선 벡터 nH가 $-Q(t)$에 의해서 구면 근점 다이아그램 위에서 하나의 구면 곡선 $nH(t)$을 생성하게 된다. 이 곡선은 시간이 흐름에 따라 여러 구면 근점 영역들을 지나게 되는데, 이에 따라 근점들이 변하게 된다.

3.2 구면 곡선 속도의 상계

단위 쿼터니언 $Q(t) = (q_0(t), q_1(t), q_2(t), q_3(t))$가 주어졌을 때, 역 단위 쿼터니언은 $Q^{-1}(t) = (q_0(t), -q_1(t), -q_2(t), -q_3(t))$으로 표현된다. 따라서, 구면 곡선은 $nH(t)$는 다음과 같이 표현된다.

$$nH(t) = Q^{-1}(t) \cdot nH \cdot Q(t) = \begin{bmatrix} 2\cdot(q_0(t)q_2(t) - q_1(t)q_3(t)) \\ -2\cdot(q_0(t)q_1(t) + q_2(t)q_3(t)) \\ -q_0^2(t) + q_1^2(t) + q_2^2(t) - q_3^2(t) \end{bmatrix}^T$$

주어진 시간 구간 $[t_s, t_e]$ 동안의 구면 곡선의 길이

$A_{nH(t)}(t_s, t_e)$는 다음과 같이 정의된다.

$$A_{nH(t)}(t_s, t_e) = \int_{t_s}^{t_e} \|nH'(t)\| dt,$$

이때, 사용되는 $\|nH'(t)\|$는 구면 곡선 $nH(t)$의 속력이다. 일반적으로 속력이 근호를 가지고 있기 때문에 해석학적인 곡선의 길이를 구하는 것은 힘들다. 따라서, 속력의 상계를 구한다면 전체적인 곡선의 길이의 상계를 부등식을 이용하여 구할 수 있게 된다.

먼저, 구면 곡선의 도함수를 구하면,

$$nH'(t) = \begin{bmatrix} q_0'(t)q_2(t) + q_0(t)q_2'(t) q_1'(t)q_3(t) - q_1(t)q_3'(t) \\ -q_0'(t)q_1(t) - q_0(t)q_1'(t)q_2'(t)q_3(t) - q_2(t)q_3'(t) \\ -q_0'(t)q_0(t) + q_1'(t)q_1(t) + q_2'(t)q_2(t) - q_3'(t)q_3(t) \end{bmatrix}^T$$

이러한 도함수 크기의 상계를 살펴보면,

$$\frac{\|nH'(t)\|^2}{4} =$$

$$(q_0'(t))^2 \cdot (q_0^2(t) + q_1^2(t) + q_2^2(t)) + (q_1'(t))^2 \cdot (q_0^2(t) + q_1^2(t) + q_3^2(t))$$
$$+ (q_2'(t))^2 \cdot (q_0^2(t) + q_2^2(t) + q_3^2(t)) + (q_3'(t))^2 \cdot (q_1^2(t) + q_2^2(t) + q_3^2(t))$$
$$+ 2 \cdot q_0'(t) \cdot q_3'(t) \cdot q_0(t) \cdot q_3(t) - 2 \cdot q_0'(t) \cdot q_1'(t) \cdot q_2(t) \cdot q_3(t)$$
$$+ 2 \cdot q_0'(t) \cdot q_2'(t) \cdot q_1(t) \cdot q_3(t) - 2 \cdot q_2'(t) \cdot q_3'(t) \cdot q_0(t) \cdot q_1(t)$$
$$+ 2 \cdot q_1'(t) \cdot q_3'(t) \cdot q_0(t) \cdot q_2(t) + 2 \cdot q_1'(t) \cdot q_2'(t) \cdot q_1(t) \cdot q_2(t)$$
$$\le (q_0'(t))^2 \cdot (q_0^2(t) + q_1^2(t) + q_2^2(t)) + (q_1'(t))^2 \cdot (q_0^2(t) + q_1^2(t) + q_3^2(t))$$
$$+ (q_2'(t))^2 \cdot (q_0^2(t) + q_2^2(t) + q_3^2(t)) + (q_3'(t))^2 \cdot (q_1^2(t) + q_2^2(t) + q_3^2(t))$$
$$+ \frac{1}{2} \cdot (|q_0'(t)|^2 + |q_3'(t)|^2) \cdot (|q_0(t)|^2 + |q_3(t)|^2)$$
$$+ \frac{1}{2} \cdot (|q_0'(t)|^2 + |q_1'(t)|^2) \cdot (|q_2(t)|^2 + |q_3(t)|^2)$$
$$+ \frac{1}{2} \cdot (|q_0'(t)|^2 + |q_2'(t)|^2) \cdot (|q_1(t)|^2 + |q_3(t)|^2)$$
$$+ \frac{1}{2} \cdot (|q_2'(t)|^2 + |q_3'(t)|^2) \cdot (|q_0(t)|^2 + |q_1(t)|^2)$$
$$+ \frac{1}{2} \cdot (|q_1'(t)|^2 + |q_3'(t)|^2) \cdot (|q_0(t)|^2 + |q_2(t)|^2)$$
$$+ \frac{1}{2} \cdot (|q_1'(t)|^2 + |q_2'(t)|^2) \cdot (|q_1(t)|^2 + |q_2(t)|^2)$$
$$= \frac{3}{2} \cdot ((|q_0'(t)|^2 + |q_1'(t)|^2 + |q_2'(t)|^2 + |q_3'(t)|^2)) \cdot (q_0^2(t) + q_1^2(t) + q_2^2(t) + q_3^2(t))$$
$$= \frac{3}{2} \cdot \|Q'(t)\|^2 \cdot \|Q(t)\|^2$$
$$= \frac{3}{2} \cdot \|Q'(t)\|^2.$$

그러므로,

$$\|nH'(t)\| \le \sqrt{6} \cdot \|Q'(t)\|.$$

그러면 단위 쿼터니언 함수 $Q(t)$의 속력의 한계를 계산한다면 전체적인 한계치를 유도할 수 있다. 단위 쿼터니언 함수의 도함수 $Q'(t)$는 다음의 수식을 만족한다 [21,22].

$$Q'(t) = \frac{1}{2} Q(t) \cdot w(t)$$

이때, 사용되는 $w(t)$는 단위 쿼터니언 함수 $Q(t)$의 각속도 함수이다. 쿼터니언 곱셈 연산에 의해서 위 수식은 다음으로 정리될 수 있다.

$$Q'(t) = \frac{1}{2}(-\langle q(t), w(t)\rangle,\ q_0(t)w(t) + q(t) \times w(t)),$$

이때, $\langle q(t), w(t)\rangle$와 $q(t) \times w(t)$는 두 벡터 $q(t)$와 $w(t)$의 내적과 외적을 의미한다. 위 식을 정리하면

$$\|Q'(t)\|^2 = \frac{1}{4}\langle q(t), w(t)\rangle^2 + \frac{1}{4}\|q_0(t) \cdot w(t) + q(t) \times w(t)\|^2.$$

우변의 두 번째 식을 정리하면 다음과 같은 간단한 식을 얻을 수 있다.

$$\|q_0(t) \cdot w(t) + q(t) \times w(t)\|^2$$
$$= \langle\ q_0(t) \cdot w(t) + q(t) \times w(t),\ q_0(t) \cdot w(t) + q(t) \times w(t)\ \rangle$$
$$= \|q_0(t) \cdot w(t)\|^2 + \|q(t) \times w(t)\|^2$$

왜냐하면, $w(t)$와 $q(t) \times w(t)$는 수직이기 때문에 $\langle w(t), q(t) \times w(t)\rangle = 0$이다. 위 수식을 이용하면 다음의 관계식을 얻을 수 있다.

$$4 \cdot \|Q'(t)\|^2 = \langle q(t), w(t)\rangle^2 + \|q_0(t) \cdot w(t)\|^2 + \|q(t) \times w(t)\|^2$$
$$\le \|q(t)\|^2\|w(t)\|^2 + q_0^2(t)\|w(t)\|^2 + \|q(t)\|^2\|w(t)\|^2$$
$$= \|q(t)\|^2\|w(t)\|^2 + (q_0^2(t) + \|q(t)\|^2) \cdot \|w(t)\|^2$$
$$\le 2 \cdot \|Q(t)\|^2\|w(t)\|^2$$
$$\le 2 \cdot \|w(t)\|^2$$

따라서, 구면 곡선의 속력의 상계는 다음 수식으로 정리될 수 있으며

$$\|nH'(t)\| \le \sqrt{3} \cdot \|w(t)\|,$$

이를 이용하면, 전체적인 곡선의 길이의 한계는 다음과 같은 관계식을 얻는다.

$$A_{nH(t)}(t_s, t_e) = \int_{t_s}^{t_e} \|nH'(t)\| dt \le \sqrt{3} \cdot w(t) \cdot (t_e - t_s)$$

4. 거리 함수

다면체 P가 단위 쿼터니언 $Q(t)$에 따라 회전운동을 하게 되면 구면 곡선 $nH(t)$는 구면 근점 다이아그램의 여러 구면 근점 영역들을 지나간다. 그 곡선이 하나의 영역 S_{v_i}를 지나고 있는 동안에는 S_{v_i}에 대응되는 꼭지점 v_i가 다면체 P의 근점이다. 따라서, H에 대한 v_i의 거리가 H에 대한 다면체 P의 거리가 된다. 그러므로, 애니메이션 동안에 구면 곡선 $nH(t)$가 지나가는 영역들, 그 영역을 들어갈 때의 시점과 나올 때의 시점을 알고 있다면, 평면 H에 대한 다면체 P의 거리함수를 구할 수 있게 된다.

일반적으로 곡선과 다이아그램의 영역과의 교점을 구하기 위해서는 다이아그램의 각 영역의 모든 선분들과의 교점을 모두 조사하여야 한다. 그런데 이러한 교차점 테스트들 대부분은 수치적 근사 방법들을 사용하기 때

문에 경우에 따라서는 많은 시간이 소요된다. 또한 본 논문에서 제시되는 다이아그램이 구면에서 정의되는 것이기 때문에 위와 같은 평면에서 사용되는 방법들을 그대로 사용하기에는 기하학적인 문제가 있어 어려움을 겪게된다. 따라서, 이러한 문제를 극복하기 위하여 애니메이션에 사용되는 다른 방법을 제안한다. 주어진 시간 구간 동안, 구면 곡선 $nH(t)$가 하나의 구면 영역에만 속하든지 아니면 이웃한 영역에만 속하는지를 결정하는 방법이다. 하나의 영역에만 속하는 시간 구간을 *동일 영역 구간(same-region interval)*, 이웃한 영역에만 속하는 시간 구간을 *이웃 영역 구간(adjacent-region interval)*이라 한다. 만약, 주어진 시간 구간이 동일 영역 구간도 이웃 영역 구간도 아니면 그 구간을 *미결정 구간(undetermined interval)*이라 한다. 우리는 애니메이션 구간을 세 가지 종류의 부분 구간들의 집합으로 나누고 이러한 분류를 효과적으로 수행하기 위하여 $nH(t)$의 곡선 길이와 구면 상에서 타원을 정의하는 구면 거리를 비교한다.

4.1 시간 구간 분류법

주어진 시간 구간이 $I_{s,e}=[t_s, t_e]$라고 하자. S_s와 S_e는 시점 t_s와 종점 t_e일 때의 구면 곡선의 위치 $nH(t_s)$와 $nH(t_e)$들이 속하는 구면 영역들이라 하자. 만약, S_s가 S_e와 일치하지 않거나 이웃하지 않다면, 그 구간 $I_{s,e}$ 동안 구면 곡선 $nH(t)$는 S_s와 S_e와는 다른 여러 영역들을 지나야 할 것이다. 이러한 경우에는 그 구간을 두 개의 작은 시간 구간으로 분할하여 동일한 테스트를 그 부분 구간들에 반복해서 적용한다. 그렇지 않다면, 그 구간 $I_{s,e}$ 동안 구면 곡선 $nH(t)$가 S_s와 S_e의 합집합에 놓일 가능성이 있게 된다. 먼저 그 구간이 이웃 영역 구간인지를 다음의 방법을 통하여 확인하도록 한다.

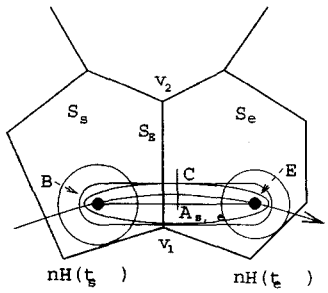
① 이웃 영역 구간(Adjacent-Region Interval)

만약, S_s가 S_e에 구면 선분 $S_E=(v_1, v_2)$을 통해 이웃한다고 가정하자. 그리고, $nH(t_s)$와 $nH(t_e)$를 지나는 대원호 $A_{s,e}$를 생각하자. 만약, $A_{s,e}$가 구면 선분 S_E와 교차한다면, S_s와 S_e의 합집합에 속하는 구면 타원을 만들 수 있다. 먼저, 대원호 $A_{s,e}$의 라운드 바운딩 영역 B를 생각하자. B의 구면 수평선들은 $A_{s,e}$에 평행하며 양쪽 라운드 부분은 지오데식(geodesic) 거리 Sr을 반지름으로 하는 구면 반원으로 구성된다(그림 3 참조). 이때,

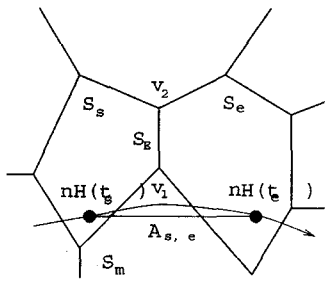
$$Sr = \min\{r_s, r_e, r_{v_1}, r_{v_2}\},$$

$$r_s = \min\{\, G(nH(t_s), x) \mid x \in Bd(S_s)\,\},$$

$$r_e = \min\{\, G(nH(t_e), x) \mid x \in Bd(S_e)\,\},$$



(a) 기준선 $A_{s,e}$가 선분 S_E를 지나는 경우



(b) $A_{s,e}$가 S_E를 지나지 않는 경우

그림 3 이웃 영역 구간

$$r_{v_1} = G(S_{v_1}, A_{s,e}),$$

$$r_{v_2} = G(S_{v_2}, A_{s,e}).$$

그리고, $G(x,y)$는 가우스 구면 상에서 정의되는 두 점 x와 y의 Geodesic 거리인데, $G(x,y)=\cos^{-1}\langle x, y\rangle$으로 계산된다. C를 B의 경계선과 $A_{s,e}$의 이등분선과의 교점이라 하자. 우리는 구면 타원 E를 다음과 같이 정의할 수 있다.

$$E=\{\, x\in S^2 \mid G(nH(t_s), x)+G(nH(t_e), x) = GS_C \,\}.$$

이때, GS_C는 $nH(t_s)$와 $nH(t_e)$의 C까지의 거리의 합이다. 이제 우리는 타원 E가 바운딩 영역 B에 속하는지를 다음과 같이 쉽게 확인할 수 있다.

$nH(t)$의 곡선 길이를 GS_C와 비교하여, 주어진 시간 구간 $I_{s,e}$가 이웃 영역 구간인지 아닌지를 결정하도록 하자. M이 시간 구간 $I_{s,e}$ 동안에 구면 곡선 $nH(t)$의 속도의 상계(upper bound)라고 할 때, 만약 부등식 $M(t_e-t_s)\le GS_C$을 만족한다면, 그 구간 동안의 구면 곡선 $nH(t)$의 곡선 길이 $A_{nH(t)}(t_s, t_e)$는 다음과 같이 GS_C보다 작게 된다.

$$A_{nH(t)}(t_s, t_e)=\int_{t_s}^{t_e}\|nH'(t)\|\,dt \le M(t_e-t_s) \le GS_C.$$

이는 곧, 시간 구간 $I_{s,e}$ 동안의 구면 곡선 부분이 구면 타원 E안에 속한다는 사실이다. 따라서, 그 곡선 부분은

S_s와 S_e의 합집합에 속하게 되며 그 시간 구간 $I_{s,e}$는 이웃 영역 구간으로 분류된다. 만약 곡선의 길이가 GS_C 보다 크다면 곡선 부분이 합집합에 속한다고 단언할 수 없다. 이런 경우에는 미결정 구간이라 여기고 그 구간을 분할하여 다시 부분 구간들에 대해서 적용한다.

② 동일 영역 구간(Same-Region Interval)

구면 영역 S_s가 영역 S_e와 동일하다고 가정하자. S_s의 경계 $Bd(S_s)$가 m_s개의 순서가 있는 구면 선분들의 리스트 $(S_{e_1}, S_{e_2}, \cdots, S_{e_m})$로 표현된다고 하자. S_s에 속하는 구면 타원을 정의하기 위하여, $nH(t_s)$와 $nH(t_e)$까지의 거리의 합이 최소가 되게 하는 경계 $Bd(S_s)$ 위에 있는 점 x_s를 다음과 같이 구할 수 있다(그림 4참조).

$$G(nH(t_s), x_s) + G(nH(t_e), x_s) =$$
$$\min\{\ G(nH(t_s), x) + G(nH(t_e), x) \mid x \in Bd(S_s)\ \}$$

구면 근점 영역이 구면 상에서 볼록 다각형이며 각 영역의 모든 구면 선분들이 대원호이기 때문에 위 방정식을 만족하는 점 x_s는 $O(m_s)$ 시간에 구할 수 있다. 그러면, S_s에 속하는 타원 중에서 가장 큰 타원 E_s을 다음과 같이 정의할 수 있다.

$$E_s = \{x \in S^2 \mid G(nH(t_s), x) + G(nH(t_e), x) = G_s\},$$

이때, $G_s = G(nH(t_s), x_s) + G(nH(t_e), x_s)$.

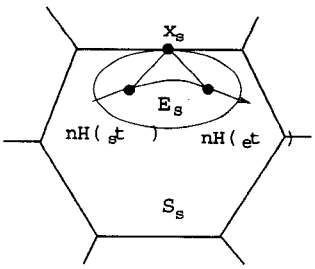


그림 4 동일 영역 구간

주어진 시간 구간 $I_{s,e}$가 동일-영역-구간인지 아닌지를 확인하기 위하여, 그 구간 동안의 곡선 길이와 G_s 값과 비교한다. 만약, 그 구간이 다음 부등식 $M(t_e - t_s) \le G_s$을 만족하다면, 그 구간 동안의 곡선의 길이는 다음과 같이 G_s보다 작게 된다.

$$A_{nH(t)}(t_s, t_e) = \int_{t_s}^{t_e} \|nH'(t)\|\ dt \le M(t_e - t_s) \le G_s.$$

따라서, 그 곡선은 타원 E_s 내부에 놓이게 되면 전체적으로는 구면 영역 S_s 내부에 속하게 된다. 그러므로 그 시간 구간 $I_{s,e}$은 동일-영역-구간으로 분류된다. 만약, 그 구간이 그 부등식을 만족하지 않는다면, 현시점에서는 미결정 구간으로 분류된다.

4.2 근접 거리 함수

주어진 애니메이션 시간 구간 $I=[0,1]$을 동일-영역-구간과 이웃-영역-구간의 분할로 나뉘어질 수 있도록 4.1절에서 상술된 시간 구간 분류법을 반복적으로 적용한다. 그러면 주어진 시간 구간 I는 $I = I_1 \cup I_2 \cup \cdots \cup I_n$와 같이 부분 구간들의 분할(partition)로 표현된다. 이때 각 부분 구간 I_i 는 동일-영역-구간이거나 이웃-영역-구간이다. 동일 영역 구간 동안의 구면 곡선의 부분은 하나의 구면 근점 영역 S_i에만 속하게 되는 반면, 이웃 영역 구간 동안의 구면 곡선 부분은 이웃하는 두 개의 구면 근점 영역들 S_i와 S_j 모두를 지난다. 따라서, S_i에 대응되는 다면체 P의 꼭지점 v_i가 그 동일 영역 구간 동안 근점이 되며, 이웃 영역 구간일 경우에는 v_i와 v_j 중 하나가 근점이 됨을 알 수 있다. 그러므로, 거리 함수 $D(t)$를 구하기 위해서는 그 구면 영역에 대응되는 근점들의 움직임만 조사하면 충분하다.

계산의 편의를 위해서 무한 평면은 xy 평면이라 가정하며 다면체 P의 각 꼭지점 v_i는 다면체의 무게 중심 c_p에 대한 상대적인 좌표 $r_i = v_i - c_p$과 같이 표현된다고 하자. 그러면 다면체 P의 강체운동을 수행함에 따라, v_i의 위치 $p_i(t)$는 다음과 같이 표현된다.

$$p_i(t) = T(t) + Q(t)\ r_i\ Q^{-1}(t),$$

이때, 는 다면체 P의 평행이동을 나타내는 수식이다. 따라서, 꼭지점 v_i의 무한 평면 $T(t) = (T_1(t), T_2(t), T_3(t))$ H에 대한 거리 $d_i(t)$는 $p_i(t)$의 z-성분이며 다음과 같은 식으로 표현된다.

$$d_i(t) = T_3(t) + 2r_{i1}(-q_0(t)q_2(t) + q_1(t)q_3(t))$$
$$+ 2r_{i2}(q_0(t)q_1(t) + q_2(t)q_3(t))$$
$$+ r_{i3}(q_0^2(t) - q_1^2(t) - q_2^2(t)(t) + q_3^2(t)).$$

그러므로, 시간 구간의 종류에 따라 그 구간에서의 다면체 P와 평면 H과의 거리함수는 다음과 같이 정리된다.

$$D(t) = \begin{cases} d_i(t), & \text{동일영역구간} \\ \min\{d_i(t), d_{j+1}(t)\}, & \text{이웃영역구간} \end{cases}$$

5. 연속 기법의 충돌검사법

본 절에서는 시간 구간 분류법과 구간 Newton 방법을 이용하는 효율적인 연속 기법의 충돌 검사 방법을 소개한다. 본 논문에서 제시되는 연속적 충돌 탐지 알고리즘은 시간 구간 분할 과정, 부분 구간에서의 충돌 존재 여부 조사 과정, 그리고 렌더링 과정으로 구성되어 있다. 첫 번째 과정으로, 주어진 시간 구간 [0, 1]으로부

터 첫 번째 동일 영역 구간인 $[0, t_{s_1}]$을 얻을 때까지 분할을 반복한다. 경우에 따라서는 $t_{s_1}=1$일 수도 있다. 이 과정을 자세히 설명하면, 우선 주어진 구간 [0, 1]이 어떤 구간인지 확인한다. 만약 그 구간이 동일 영역 구간이면 분할과정을 그만하고, 그렇지 않다면 그 구간을 분할하여 두 번째 부분 구간 [0.5, 1.0]을 스택에 저장한다. 그리고 첫 번째 부분 구간인 [0, 0.5]에 대해서 구간 분류법을 적용시킨다. 만약 그 구간이 동일 영역 구간이 아니면 위와 같은 구간 분할법과 구간 분류법을 반복해서 적용하여 첫 번째 동일영역 구간을 얻고 나머지 부분 구간들은 스택에 저장한다. 그 부분 구간 동안의 구면 곡선 $nH(t)$는 시작점 $nH(0)$가 속하는 구면 영역 S_0에 속하게 되며 이때 대응되는 v_0가 가장 가까운 점이 되며 이 점과 평면까지의 거리가 두 물체 사이의 거리 함수를 표현하게 된다.

두 번째 과정으로, 이 시간 구간 동안 충돌이 발생하는지 아닌지 확인하기 위하여, 그 거리함수에 구간 (interval) Newton 방법을 적용시킨다. 일반적으로 충돌 시점이라 함은 거리 함수가 zero를 갖는 값 중에서 현 시점에 제일 가까운 zero 값이다. 방정식의 근을 구하는 해법은 많이 있으나 우리가 원하는 조건을 만족하면서 정확한 결과를 내는 방법으로는 구간 Newton 방법이다. 이 방법은 적은 회수의 계산에 의해서 근의 존재성을 알 수 있을 뿐만 아니라 존재하는 근들에 대한 또 다른 정렬 연산 없이도 첫 번째 근을 알려주기 때문에 본 충돌 탐지 알고리즘에 타당한 해법임을 알 수 있다.

마지막 과정으로, 첫 번째 동일 영역 구간인 $[0, t_{s_1}]$ 동안에 충돌이 발생하지 않는다면, 시점 t_s까지는 모든 시점에서의 거리 계산과 충돌 여부 검사를 하지 않고 단지 움직이는 물체를 렌더링만 실행한다. 그리고 난 후, 시뮬레이션 시간이 t_s에 이르게 되면 스택에서 제일 위의 내용인 부분 구간 $[t_{s_1}, t_{s_2}]$을 꺼낸다. 만약, 이 부분 구간이 동일 영역 구간이거나 이웃 영역 구간이면, 두 번째 과정을 수행한다. 그렇지 않다면, 첫 번째 과정을 반복적으로 수행하여 동일 영역 구간이거나 이웃 영역 구간을 얻도록 한다.

만약, 첫 번째 동일 영역 구간인 $[0, t_{s_1}]$ 동안에 충돌이 t_c 시점에서 발생한다면, 시점 t_c까지는 거리 계산을 하지 않고 단지 움직이는 물체를 렌더링만 실행한다. 그리고 난 후, 시뮬레이션 시간이 t_c이 되면 충격량과 토크를 계산한다. 즉, 다면체의 새로운 평행이동 및 회전 운동 변환을 얻게 된다. 그러므로, 가우스 구면 상에 새로운 구면 곡선을 얻게 되고, 이 곡선을 가지고 이 절에서 언급된 모든 과정을 반복해서 적용한다. 만약 충돌 반응에 의한 결과로서 나타나는 토크가 커서 각속도가 갑자기 증가한다면 기존의 점진적 기법으로는 좋은 결과를 뽑아 낼 수 없음을 알 수 있다. 왜냐하면, 그러한 방법들은 움직이는 물체의 속도가 한정되어 있다는 가정에서 시작하기 때문이다. 반면, 4절에서 언급된 구간 분류법을 통한 충돌 검사법은 물체의 속도에 무관하게 적용되기 때문에 본 알고리즘의 수행 결과는 시뮬레이션 구간 단위가 변하더라도 영향을 받지 않음을 알 수 있다.

(a) (b)

그림 5 다면체, 그에 대응되는 구면 근점 다이아그램, 구면 곡선

6. 수행 결과

우리는 시간 구간 [0.0, 3.0] 동안에 하나의 다면체의 움직임을 시뮬레이션하고 평면과 충돌 여부 및 전체적인 알고리즘의 성능을 평가한다. 그림 5(a)에 있는 다면체는 속도 $v=(10.0,5.0,15.0)$, 각속도 $w=(-1.789, 1.123, -0.789)$로 움직임을 시작한다. 그림 5(b)는 그 다면체에 의해서 생성되는 구면 근점 다이아그램을 부여주고 있으며 초기 각속도에 의해서는 생성되는 구면 곡선 $nH(t)$가 구면 상의 하나의 원으로 표현됨을 보여주고 있다. 이때, 적용되는 구면 곡선의 속도의 상계는 $\sqrt{3}\cdot\|w\|$으로 정의되며 그 값은 3.905446이고, 구간 Newton 방법의 근사 한계치(threshold)로서는 $\varepsilon=0.000001$을 사용하였다. 그 다면체는 $t_{a_1}=1.647786$에서 첫 번째 충돌을 하고 새로운 속도 $v'=(10.0,5.0,12.28)$와 새로운 각속도 $w'=(1.12,1.31,-0.21)$을 가지게 된다. 그리고 난 후 두 번째 충돌 시점 $t_{a_2}=2.918145$까지는 새로운 운동으로 상공을 날고 있다. 이때의 새로운 속도와 각속도는 $v''=(10.0,5.0,5.93)$, $w''=(-0.89,-6.44,-0.49)$이다.

표 1은 이산적 충돌 알고리즘(기존)과 본 논문에서 제시하는 연속적 알고리즘에 의해서 구해진 이 다면체의 충돌 시점들을 보여주고 있다. 연속적 방법에 의한

표 1 [0.0, 3.0] 동안의 충돌 시점

시뮬레이션 시간 간격	$\Delta t=0.005$		$\Delta t=0.001$	
충돌 시점	t_{a_1}	t_{a_2}	t_{a_1}	t_{a_2}
연속적 알고리즘	1.647786	2.918145	1.647786	2.918145
이산적 알고리즘	1.646488	2.923164	1.647981	2.916781

표 2 연속적 알고리즘과 이산적 알고리즘의 소요시간 비교

	연속적 알고리즘		이산적 알고리즘	
다면체(면의 수)	$\Delta t=0.005$	$\Delta t=0.001$	$\Delta t=0.005$	$\Delta t=0.001$
Cube(6)	0.159435	0.158708	2.496181	12.479868
Octahedron(8)	0.156491	0.162084	2.491545	12.507955
Con(13)	0.307315	0.307915	2.612410	12.800396
Pot(50)	0.411429	0.423491	5.450509	27.569424
Sphere(64)	1.299005	1.292649	6.478204	31.792561
Sphere(100)	1.482894	1.497369	11.416183	50.046286
Sphere(200)	4.379463	4.377843	17.176759	80.620888

충돌 시점들은 시뮬레이션 시간 간격 크기에 변화가 없음을 알 수 있다. 왜냐하면, 연속적 방법은 Newton 방법을 적용할 때 사용되는 시간 구간이 시뮬레이션 시간 간격 크기에 의해서 정해지는 것이 아니고 다면체의 초기 각속도에 의해서 정의되는 구면 곡선이 지나가는 구면 근점 영역에 의해서 결정된 부분 구간들로서 정해지기 때문이다. 반면, 기존의 이산적 기법은 시뮬레이션 시간 간격 크기에 의해서 분할된 부분 구간들을 사용하기 때문에 충돌 시점은 수치적 오차를 가지게 된다.

표 2는 시간 구간 [0.0, 3.0] 동안에 위와 같은 동일한 운동을 적용하였을 때, 평면과 처음으로 충돌하는 시점을 찾는데 소요되는 시간을 보여주고 있다. 본 논문에서 제시된 연속적 기법에 의하면 전체적인 소요 시간이 시뮬레이션 시간 간격 크기에 무관함을 알 수 있다. 즉, $\Delta t=0.005$이든 $\Delta t=0.001$이든 상관없이 다면체의 면의 수에만 영향을 받는다는 사실이다. 반면, 기존의 이산적 기법들은 $\Delta t=0.001$인 경우의 전체적인 소요 시간은 $\Delta t=0.005$의 경우에 소요되는 시간보다 5 배 가량 됨을 알 수 있다. 본 논문에서 제시하는 연속적 기법은 거리 함수 생성 과정에서 미리 근점들을 알고 있기 때문에 시뮬레이션 각 시점에서 따로 근점을 계산할 필요가 없다. 그러므로, 각 시점에서의 거리 계산은 한 번에 가능하게 된다. 이것은 곧, 거리 계산에 소요되는 시간이 시뮬레이션 시간 간격 크기에 무관함을 의미하는 것이며, 따라서 시뮬레이션 시간 간격 크기가 연속적 충돌 탐지 알고리즘의 전체적인 소요시간에 영향을 미치지 못함을 의미한다. 반면, 이산적 기법들은 매 시점에서 근점을 찾고 다시 거리를 계산하여 충돌 여부를 파악한다. 이 알고리즘들이 빠른 계산 능력을 가지고 있다고 하여도, 충돌 여부 조사를 위한 계산 회수가 시뮬레이션 시간 간격 크기에 반비례하기 때문에 이산적 기법의 전체적인 알고리즘의 소요시간은 시뮬레이션 시간 간격 크기에 영향을 받는다.

7. 결 론

본 논문에서는 강체 운동을 하고 있는 물체와 평면과의 충돌 여부를 효과적으로 처리하는 새로운 개념의 연속적 충돌 탐지 기법을 소개하였다. 본 알고리즘의 핵심은 기하학적 특징을 내포하는 구면 근점 다이아그램을 구성하여 두 물체간의 거리함수를 생성하였다는 점이다. 그러므로, 연속적 충돌 탐지 알고리즘은 정확한 충돌 시점을 파악함에 있어 다른 알고리즘들에 비해 두 가지 장점을 가진다. 첫 번째로, 각 시점에서 두 물체간의 거리를 계산함에 있어서, 연속적 기법은 거리함수로부터 그 값을 추출하면 되는 반면, 기존의 이산적 기법들은 $O(\log n)$ 시간의 거리 계산 알고리즘을 적용시켜야 한다. 이때, n은 다면체를 구성하는 면의 수이다. 경우에 따라서는 $O(1)$ 시간에도 가능하다고 주장하는 알고리즘들이 있지만 상술한 바와 같이 충돌 이후의 갑작스런 각속도의 변화에 원치 않는 결과를 초래할 수 있는 문제점을 가지고 있기 때문에 일반적인 시간 복잡도라고는 할 수 없다. 그러므로, 본 알고리즘의 전체적인 소요 시간은 시뮬레이션 시간 간격 크기에 무관하며, 동일한 물체와 여러 번 충돌하는 환경에 매우 적합하다. 두 번째로, 본 알고리즘은 시뮬레이션 시간 간격 보다 넓은 시간 부분 구간을 interval Newton 방법에 적용한다는 점이다. 여기에 사용되는 시간 부분 구간은 구면 곡선이 지나가는 영역들에 의해서 결정되기 때문에 이 또한 시뮬레이션 시간 간격 크기와는 무관하다. 그러므로, 충돌 시점들은 표 1에서 보았던 것처럼 시뮬레이션 시간 간격 크기와 무관하며 주어진 오차 한도 ϵ 내에서 계산될 수 있게 된다.

참 고 문 헌

[1] James E. Bobrow, "A Direct Minimization Approach for Obtaining the distance between Convex Polyheda," *The International Journal of Robotics Research*, Vol. 8, No. 3, pp. 65-76, 1989.

[2] Ming C. Lin and John F. Canny, "A New Fast Algorithm for Collision Detection," *Proceed. of Computer Animation*, pp. 43-55, 1993.

[3] F. Dai, "Collision-Free Motion of an Articulated Kinematic Chain in a Dynamic Environment," *IEEE Computer Graphics and Applications*, Vol. 9, No. 1, pp. 70-74, 1989.

[4] M. Herman, "Fast three-dimensional collision-free motion planning," *Proceed. of IEEE International Conference on Robotics and Automation 2*, pp. 1056-1063 (1986).

[5] A. Pentland, "Computational Complexity versus Simulated Environment," *SIGGRAPH '93*, pp. 185-192 (1990).

[6] Miguel A. Otaduy and Ming C. Lin, "CLODs : Dual Hierarchies for Multiresolution Collision Detection," *Eurographics Symposium on Geometry Processing*, 2003.

[7] H. Edelsbrunner, *Algorithms in Computational Geometry*, Springer-Verlag, 1987.

[8] Ming C. Lin, *Efficient Collision Detection for Animation and Robotics*, PhD thesis, Univ. of California, Berkeley, 1993.

[9] Ming C. Lin and John F. Canny, "A Fast Algorithm for Incremental Distance Calculation," *Proceed. of IEEE International Conference on Robotics and Automation*, pp. 1008-1014 (1991).

[10] Madhav K. Ponamgi and Dinesh Manocha and Ming C. Lin, "Incremental Algorithms for Collision Detection Between Polygonal Models," *IEEE Transaction on Visualization and Computer Graphics*, Vol. 3, No. 1, pp. 51-64 (1997).

[11] David. Baraff, "Interactive Simulation of Solid Rigid Bodies," *IEEE Computer Graphics and Application*, (May), pp. 63-75 (1995).

[12] Manfredo P. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, New Jersey, 1976.

[13] Hyoung Seok Kim, *Collision Detection Using Spherical Voronoi Diagrams*, Ph. D. Thesis, KAIST, 1998.

[14] Hyoung Seok Kim, Hong Oh kim, and Sung Yong Shin, "An Efficient Algorithm for Determining the Extreme Vertices of a Moving 3D Convex with Respect to a Plane," *Journal of Computers and Mathematics with Applications*, Vol. 36, No. 3, pp. 55-61 (1998).

[15] Hyoung Seok Kim, "A Sequence of the Extreme Vertices of a Moving Regular Polyhedron Using Spherical Voronoi Diagram," *Journal of Korea Multimedia Society*, Vol. 3, No. 3, pp. 298-308 (2000).

[16] Hyoung Seok Kim, "A Linear-time Algorithm for Computing the Spherical Voronoi Diagram of Unit Normal Vectors of a Convex Polyhedron," *Journal of KISS: Computer Systems and Theory*, Vol. 27, No. 10, pp. 835-839 (2000).

[17] Ramon E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, 1966.

[18] Ketan Mulmuley, *Computational Geometry: An Introduction Through Randomized Algorithms*, Prentice-Hall, New Jersey, 1994.

[19] Leo. J. Guibas and Jorge Stolfi and Kenneth L. Clarkson, "Solving Related Two- and Three-Dimensional Linear Programming Problems in Logarithmic Time," *Theoretical Computer Science*, Vol. 49, pp. 81-84 (1987).

[20] J. L. Junkins and J. D. Turner, "Optimal Continuous Torque Attitide Maneuvers," *J. Guidance and Control*, Vol. 3, No. 3, pp. 210-217 (1980).

[21] W. R. Hamilton, *Elements of Quaternions (Volume I, II)*, Chelsea Publishing Company, 1969.

[22] J. L. Junkins and J. D. Turner, *Optimal Spacecraft Rotational Maneuvers*, Elsevier, 1986.

[23] Myoung Jun Kim and Myung Soo Kim and Sung Yong Shin, "A general construction scheme for unit quaternion curves with simple high order derivatives," *SIGGRAPH*, pp. 369-376 (1995).

김 형 석

1990년 연세대학교 이과대학 수학과(학사). 1992년 한국과학기술원 응용수학과(석사, 매듭이론 전공). 1998년 한국과학기술원 응용수학과(박사, 컴퓨터그래픽스 전공). 1998년 3월~1999년 2월 한국전자통신연구원 컴퓨터그래픽스연구팀 Post Doc. 1999년~현재 동의대학교 공과대학 멀티미디어공학전공 조교수. 2004년 1월~현재 조지아공대, College of Computing, 방문교수. 관심분야는 컴퓨터그래픽스, 컴퓨터게임, 계산기하학