한국 CAD/CAM학회 논문집
제 11권 제 2호 2006년 4월 pp. 97-106

학술논문

삼차원 구의 보로노이 다이어그램 계산을 위한 두 가지 알고리듬 및 단백질구조해석에의 응용

김동욱*, 조영송**, 김덕수***

Two Algorithms for Constructing the Voronoi Diagram for 3D Spheres and Applications to Protein Structure Analysis

Kim, D.*, Cho, Y.** and Kim, D.-S.***

ABSTRACT

Voronoi diagrams have been known for numerous important applications in science and engineering including CAD/CAM. Especially, the Voronoi diagram for 3D spheres has been known as very useful tool to analyze spatial structural properties of molecules or materials modeled by a set of spherical atoms. In this paper, we present two algorithms, the edge-tracing algorithm and the region-expansion algorithm, for constructing the Voronoi diagram of 3D spheres and applications to protein structure analysis. The basic scheme of the *edge-tracing algorithm* is to follow Voronoi edges until the construction is completed in $O(mn)$ time in the worst-case, where m and n are the numbers of edges and spheres, respectively. On the other hand, the *region-expansion algorithm* constructs the desired Voronoi diagram by expanding Voronoi regions for one sphere after another via a series of topology operations, starting from the ordinary Voronoi diagram for the centers of spheres. It turns out that the region-expansion algorithm also has the worst-case time complexity of $O(mn)$. The Voronoi diagram for 3D spheres can play key roles in various analyses of protein structures such as the pocket recognition, molecular surface construction, and protein-protein interaction interface construction.

Key words : Voronoi diagrams, edge-tracing, region-expansion, computational geometry, protein structure

1. 서 론

20세기 초반 이후로 보로노이 다이어그램은 계산기하학을 비롯하여 과학 및 공학의 다양한 분야에서 중요한 주제로 자리잡고 있다[15]. 특히 삼차원에서 구의 보로노이 다이어그램은 원자 단위로 모델링 되는 분자 생물학 및 재료 분야의 다양한 응용 문제에 적용시킬 수 있기 때문에 그 중요성은 더욱 높아지고 있다. 예를 들어, 단백질 및 RNA의 구조를 분석하는데 원자들의 공간 구조를 해석할 수 있는 계산 도구가 필요한 것으로 잘 알려져 있으며[6,9,10,16], 신소재의 개발 시에도 원자들 사이의 공간 구조를 분석하는 것이 매우 근본적이면서도 중요한 문제인 것으로 알려져 있다[12,14,17].

특히 삼차원에서 구의 보로노이 다이어그램은 단백질의 구조적 특성을 분석하는데 매우 중요한 역할을 하는 molecular surface를 정확하고 효율적으로 계산하는데 핵심적으로 이용될 수 있으며[22], 신약 개발 시에 매우 중요한 문제로 알려져 있는 단백질의 포켓을 찾는 문제에 매우 효과적으로 적용할 수 있다[24]. 그리고 단백질들 사이의 상호작용을 보다 정확하고 체계적으로 분석하는데 중요하게 이용되는 상호작용 인터페이스를 정의하고 구축하는 데에도 매우 중요하게 응용될 수 있다[21].

그러나 평면 및 일반 차원에서 점 집합에 대한 보로노이 다이어그램의 계산법 및 특성에 대한 매우 많은 연구들이 진행되어온 반면에, 삼차원 구에 대한 보

*성회원, 한양대학교 Voronoi Diagram 연구단
**한양대학교 Voronoi Diagram 연구단
***종신회원, 한양대학교 산업공학과
- 논문투고일: 2005. 06. 20
- 심사완료일: 2005. 11. 08

로노이 다이어그램에 대한 연구는 과학 및 공학의 다양하고도 중요한 응용분야가 잠재해 있다는 사실에 비해 그리 활발하지 않았다[1,6,13,16,18]. 이와 더불어 구들의 보로노이 다이어그램을 구하는 적절한 알고리듬과 안정적인 프로그래밍 라이브러리의 부재로 인하여 대부분의 응용문제에서는 구들의 보로노이 다이어그램을 대신하여 점들의 보로노이 다이어그램, 파워 다이어그램(power diagram) 등을 사용하고 있다.

본 논문에서는 최근 들어 신약 개발 등의 사회 경제적 가치로 인하여 그 중요성이 더욱 부각되는 생물학 및 생명 공학 분야의 응용 문제를 푸는데 핵심적 도구가 되는 삼차원 구들의 보로노이 다이어그램을 구하는 새로운 알고리듬인 모서리 추적법(Edge-Tracing)과 영역 확장법(Region-Expansion)을 제시한다. 모서리 추적법 및 영역 확장법은 모두 완전한 형태로 구현 되었으며 이들을 다양한 데이터로 실험한 결과를 제시한다. 그리고 계산된 구의 보로노이 다이어그램을 단백질의 기하 구조 분석에 적용한 예를 보여준다. 이를 통하여 본 논문에서는 CAD분야에 생물학과 관련된 새로운 응용 분야를 제시하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 삼차원 구의 보로노이 다이어그램에 대한 관련 연구를 알아보고, 3장에서는 보로노이 다이어그램에 대한 정의 및 특성에 대하여 설명을 한다. 4장에서는 보로노이 다이어그램을 구하는 알고리듬인 모서리 추적법에 대해 설명하고, 5장에서는 또 다른 새로운 알고리듬인 영역 확장법에 대하여 설명한다. 6장에서는 이들 알고리듬을 통하여 계산된 보로노이 다이어그램에 대한 적용 예를 보여주며, 7장에서는 추후 연구과제의 제시와 함께 본 논문의 결론을 맺는다.

2. 문헌 조사

다른 종류의 보로노이 다이어그램들과는 달리 삼차원에서 구들에 대한 보로노이 다이어그램의 계산 방법에 대해서는 매우 소수의 연구가 진행되어 왔다. 그 첫 번째로 Aurenhammer는 d 차원에서 구들의 보로노이 다이어그램과 $d+1$ 차원에서의 파워 다이어그램 사이의 연관관계에 대해 논하였다[2]. Will은 그의 박사학위논문에서 삼차원에서 구의 보로노이 다이어그램 전체가 아닌 하나의 보로노이 영역을 구하는 알고리듬을 제시하였고 이를 생물학의 응용 문제에 적용하였다. 또한 그는 하나의 삼차원 구와 보로노이 영역의 조합론적 복잡도(combinatorial complexity)가 n을 구의 개수라 할 때 $\Theta(n^2)$가 실제 가능함을 보였으며, 보로노이 영역 하나에 대하여 $O(n^2 \log n)$의 기대 계산량을 가지는 lower envelope 알고리듬을 제시하였다[19]. Luchnikov 등은 보로노이 다이어그램을 구하기 위한 간단하면서도 실용적인 방법인 모서리 추적법의 기본 개념을 제시하였다[13]. Boissonnat과 Karavelas는 반전(inversion)을 통하여 변환된 구들의 최소볼록집합(convex hull)을 구하고 이것의 위상정보를 이용하여 해당 보로노이 영역을 구하는 알고리듬을 제시하였다[3]. 최근 들어 Kim 등은 하나의 보로노이 영역뿐만 아니라 전체 구에 대한 보로노이 다이어그램을 구하는 모서리 추적 알고리듬의 상세한 설명을 하였으며, 이를 단백질의 구조 분석 등의 다양한 응용 문제에 적용하였다[8-11]. 또한 Kim 등은 m을 보로노이 모서리의 개수라 하고 n을 주어진 구의 개수라 할 때, 모서리 추적 알고리듬의 최악의 경우 시간 계산량(worst-case time complexity)이 $O(mn)$임을 밝혔다.

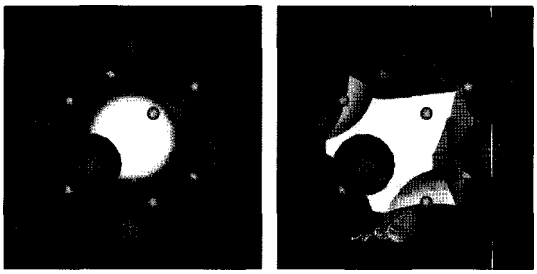
3. 삼차원 구에 대한 보로노이 다이어그램의 정의 및 특성

$S=\{s_1, s_2, \cdots, s_n\}$를 삼차원 공간에서의 구들의 집합이라 하고, 각각의 구 $s_i=(c_i, r_i)$는 중심점 $c_i=(x_i, y_i, z_i)$와 반지름 r_i로 구성되어 있다고 하자. 이때, 하나의 구가 다른 구에 완전히 포함되는 경우는 존재하지 않는다고 가정한다. 그러면 $\mathrm{dist}(a, b)$를 점 a와 점 b 사이의 유클리드 거리라 할 때, 각각의 구 s_i 마다 $VR_i=\{p|\mathrm{dist}(p, c_i)-r_i \le \mathrm{dist}(p, c_j)-r_j,\ i\neq j\}$로 정의되는 보로노이 영역이 존재하게 되며, 구집합 S에 대한 보로노이 다이어그램은 이러한 보로노이 영역의 집합인 $VD(S)=\{VR_1, VR_2, \cdots, VR_n\}$로 정의된다. 그리고 구집합 S에 의해 보로노이 다이어그램이 정의되므로, S의 각 원소를 제너레이터(generator)라 부른다.

점집합의 보로노이 다이어그램과 같이 구집합의 보로노이 다이어그램에서도 최소볼록집합의 경계를 이루는 구들은 무한대(unbounded)의 보로노이 영역을 가지며, 이 외의 구들은 보로노이 면들로 둘러싸인(bounded) 보로노이 영역을 가지게 된다. 구들의 보로노이 다이어그램의 경우 보로노이 곡면은 두 개의 구에 대한 이등분면(bisector)으로 정의되며, 이엽쌍곡면(hyperboloid of two sheets)의 일부분을 이루게 된다. 또한, 하나의 보로노이 곡면은 다른 보로노이 곡면과 교차하여 보로노이 모서리를 형성한다. 이러한 보로노이 모서리들이 교차하여 보로노이 꼭지점을 형성한다. 삼차원에서 구의 보로노이 다이어그램은 보로노이 영역들의 집합으로 구성되어 있는 셀 구조를 가지

게 되는데, 이는 CAD분야에서 널리 쓰이는 비다양체(non-manifold) 모델을 위한 자료구조[23,25] 또는 약간 변형된 형태의 자료구조로 표현이 가능하다. 본 논문에서는 삼차원 공간상의 구들은 일반 위치(general position)에 놓여있는 것으로 가정한다.

Fig. 1(a)에서는 서로 다른 반지름을 가지는 15개의 구들을 보여주고 있고, Fig. 1(b)에서는 중앙에 있는 가장 큰 구의 보로노이 영역을 주어진 구와 함께 보여주고 있다. 그림에서 보이는 것과 같이 보로노이 면과 모서리는 각각 곡면과 곡선의 형태를 띠게 된다. 그렇기 때문에, 점들에 대한 보로노이 다이어그램에서는 보로노이 영역이 볼록 집합이지만, 삼차원 구들에 대한 보로노이 영역은 해당 구의 중심에 대하여 별모양(star-shaped)이다.



(a) (b)

Fig. 1. 15 spheres with 3 different types of radii and the Voronoi region of the biggest ball in the center.

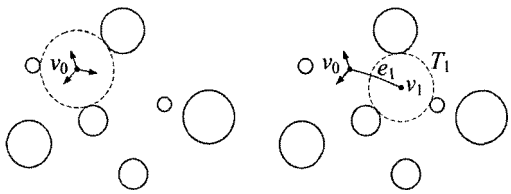
4. 모서리 추적법

모서리 추적법의 기본 개념은 다음과 같다. 우선 하나의 보로노이 꼭지점 v_0을 찾는다. 여기에서 꼭지점 v_0로부터 뻗어 나가는 네 개의 보로노이 모서리가 존재한다는 것을 쉽게 알 수 있으며, 이들 모서리들을 모서리스택(edge-stack)이라 부르는 스택 자료구조에 저장한다. 그리하여 이들 모서리들은 시작 꼭지점으로 v_0를 가지게 된다. 참고로 모서리스택에 저장되어 있는 모서리들은 하나의 꼭지점만 알고 있을 뿐 나머지 하나의 꼭지점은 아직 성해지지 않았다.

모서리스택으로부터 하나씩 모서리들을 가지고 온 후 해당 모서리의 나머지 꼭지점을 찾음으로써 모서리의 정보를 완성하게 되며, 본 논문에서는 이러한 과정을 모서리 추적(edge-tracing)이라 부른다.

Fig. 2에서는 모서리 추적법의 과정을 평면에서의 예를 들어 보여주고 있다. Fig. 2(a)에서는 초기의 꼭지점 v_0를 보여주고 있으며, v_0로부터 뻗어나가는 모서리들을 화살표로 표시하고 있으며 이들은 모서리스택에 저장된다. Fig. 2(b)에서는 모서리스택의 모서리 하나인 e_1의 나머지 끝 꼭지점 v_1을 찾음으로써 모서리의 정보를 완성하는 모서리 추적 과정의 예를 보여주고 있다. 그리고 v_1을 찾은 이후 v_1을 시작 꼭지점으로 하는 모서리 두 개를 모서리스택에 저장하게 된다.

3D에서의 경우, 모서리 추적 과정 중 끝 꼭지점은 해당 모서리를 정의하는 세 개의 구와 나머지 $n-3$개의 구들 중 하나로 정의되는 내부가 비어있는 접구(empty tangent sphere)들 중에서 결정된다. 그리고 이때 해당 꼭지점이 처음으로 방문한 꼭지점이라면, 이 점을 시작 꼭지점으로 하는 나머지 세 개의 모서리들을 구성하여 모서리스택에 넣는다. 이러한 작업을 모서리스택이 비워지게 될 때까지 진행함으로써 보로노이 다이어그램의 모서리 그래프를 완성할 수 있다. 이어지는 4.1과 4.2 두 절에서는 모서리 추적법의 두 가지 핵심 계산 요소인 유효한 꼭지점의 계산법과 기존 꼭지점 탐색법에 대하여 자세히 설명하고 있다.



(a) (b)

Fig. 2. 2D analogy of the edge-tracing algorithm: (a) initial ture vertex v_0 and three edges (denoted by the arrows) incident to the vertex, and (b) edge-tracing process for edge e_1.

4.1 유효한 꼭지점의 계산

모서리스택에서 가지고 온 모서리의 끝 꼭지점을 계산하는 것은 모서리를 정의하는 세 개의 구와 함께 해당 꼭지점을 정의하는 하나의 구를 찾는 과정으로 볼 수 있다. 그리하여, 끝 꼭지점을 구하기 위하여 네 개의 구를 통해 접구를 계산한다. 여기에서 구한 접구가 나머지 구들과 교차하지 않을 때에만 해당 접구의 중심이 보로노이 꼭지점이 될 수 있다.

보로노이 모서리 하나에 대한 끝 꼭지점을 구하기 위해 쉽게 생각할 수 있는 방법으로 다음을 들 수 있다. 이는 $n-3$개의 모든 후보 구들에 대하여 접구를 구한 다음 그 구가 $n-4$개의 구들에 대하여 모두 비어있는지를 테스트하는 것이다. 그러므로 이러한 방법은 $O(n^5)$의 계산시간이 소요되는데, 이 계산량은 몇 가지 고찰을 통하여 다음과 같은 방법으로 개선될 수

있다.

모서리를 정의하는 세 개의 구와 임의의 후보 구인 s_i로부터 접구 T_i를 계산한다. 그리고 나서 후보구들의 집합에서 또 다른 후보인 s_j를 가지고 와서 접구 T_j를 계산한다. 만약 T_j가 현재의 해인 T_i 보다 시작 꼭지점에 가까이 있으면 T_j를 현재의 해로 대체시킨다. 만약 그렇지 않다면 현재의 해로 T_i를 그대로 가진다. 이러한 작업을 모든 후보 구에 대하여 수행하면 최종적으로 남아있는 접구가 끝 꼭지점을 정의하게 된다.

시작 꼭지점에 더 가까운 접구를 찾기 위해서는 angle distance라 불리는 새로운 거리 척도를 정의할 필요가 있다. 이 거리는 시작 꼭지점에서 0의 값을 가지며, 모서리를 따라서 진행할 때에 단조 증가하여야 한다. 이러한 성질을 만족하는 거리 척도는 모서리의 타입에 따라서 다르게 정의될 수 있다. 만약 모서리의 타입이 직선일 경우 시작 꼭지점에서 해당 접구의 중심까지의 실제 유클리드 거리가 이러한 성질을 만족시킨다는 것을 쉽게 알 수 있다. 모서리의 타입이 곡선의 형태를 띠게 되면 실제 유클리드 거리로는 위의 성질을 만족시키지 못하는 경우가 발생한다. 이러한 경우에는 시작 꼭지점과 해당 모서리 곡선의 초점과 계산된 후보 꼭지점과의 각도로써 거리를 정의할 수 있으며 이때의 거리를 본 논문에서는 angle distance라 부른다.

끝 꼭지점을 찾는 방법은 모든 후보 구들로부터 만들어지는 접구들 중에서 최소의 angle distance를 가지는 접구를 찾는 것으로 정리할 수 있다. 그러므로 이러한 방법은 모든 후보 구를 한번씩만 탐색하기 때문에 정확한 끝 꼭지점을 찾는데 $O(n)$의 시간이 소요된다.

4.2 기존의 꼭지점 탐색

모서리 추적 과정 중에 끝 꼭지점을 찾았다고 가정하자. 만약 그 꼭지점이 이전 단계에서 이미 찾은 꼭지점이 아니라면, 그것을 해당 모서리의 끝 꼭지점으로 안전하게 사용할 수 있다. 그러나 만약 이미 찾아놓은 꼭지점이 존재할 경우에는 기존의 꼭지점에 해당 모서리의 정보를 추가하는 형태로 모서리 추적을 진행하여야 한다. 참고로 모서리 추적이 끝나게 되면 각 꼭지점마다 네 개의 모서리가 할당되게 된다. 이러한 이유로 새로이 찾은 꼭지점을 정의하는 네 개의 구들이 이전 단계에서 이미 꼭지점을 정의했었는지를 판단할 필요가 있다.

이러한 판단의 효율성을 위하여 VIDIC(Vertex Index Dictionary)이라는 자료 구조를 고안하였다. 이는 사전(dictionary) 자료 구조의 형태를 띠며, VIDIC에서 하나의 요소는 꼭지점을 정의하는 네 개의 구에 대한 인덱스와 해당 꼭지점을 가리키는 하나의 포인터로 구성되어있다. 그래서 기존의 꼭지점이 존재하는지의 여부는 VIDIC의 데이터를 검색함으로써 알아낼 수 있다. 여기에서 VIDIC 내부의 데이터 개수는 최대 값으로 모로노이 모서리의 수만큼 가질 수 있는데, 이는 최악의 경우 $O(n^2)$가 될 수 있다. 그리고 VIDIC 내부의 데이터들이 일정한 기준으로 정렬되어있다면 이분탐색(binary search) 등의 방법으로 최악의 경우 $O(\log n)$의 시간에 기존 꼭지점의 존재 여부를 판단할 수 있다. 하지만 적절한 해싱(hashing) 기법을 사용하면 평균적으로 $O(1)$의 시간에 검색이 가능하며 본 연구에서 제안하는 알고리듬은 이 방법에 기반하였다.

4.3 알고리듬

제안된 모서리 추적 알고리듬은 다음과 같은 단계들로 정리할 수 있다.

Algorithm: Edge-Tracing

1. Find an initial true vertex and generate four edges emanating from the vertex and push these edges into an Edge-stack.
2. Pop an edge from the Edge-stack and find empty tangent sphere to define the end vertex.
3. Check if the vertex found in Step 2 is a valid one or not by using VIDIC.
4. If the vertex is already computed one, finalize the popped edge in Step 2. If the vertex is a new one, generate three more edges and push them in the Edge-stack.
5. Repeat Step 2 through 4 until the Edge-stack is empty.

4.4 시간 계산량

제안하는 알고리듬은 초기의 꼭지점이 주어졌을 경우 모서리의 개수인 m회 반복하며 하나의 모서리를 추적할 때 $O(n)$ 시간이 걸리므로 최악의 경우 $O(mn)$의 시간 계산량을 가진다. 참고로 m은 최악의 경우 $O(n^2)$가 될 수 있다.

그리고 새로운 꼭지점을 찾았을 때 이것이 기존에 이미 찾아놓은 것인지의 판단이 필요한데, 본 알고리듬에서는 VIDIC이라는 자료구조를 이용을 하며 이를 통해 평균적으로 $O(1)$의 시간에 판단이 가능하다. 그런데 이 연산은 끝 꼭지점 하나를 찾은 다음에 수행되

기 때문에 모서리 하나를 추적할 때의 전체 계산 시간은 $O(n)$이 된다.

비록 보로노이 꼭지점, 모서리, 면의 개수가 최악의 경우 $O(n^2)$가 가능하지만, 평균 개수는 $O(n)$인 것으로 알려지 있다. 그리고, 만약 구들이 공간상에 비교적 균등하게 분포되어있고 버킷(bucket) 자료구조 등과 같은 여러 가지 가속화 기법을 적절히 이용한다면, 모서리 하나의 추적 시 끝 꼭지점의 계산이 평균적으로 $O(1)$의 시간에 가능해진다. 참고로 여기서 버킷 자료구조는 보로노이 다이어그램의 계산 전에 한번만 구성하면 되며, 이때 $O(n)$의 시간이 소요 된다. 그리하여 평균적인 시간 계산량은 주어진 구의 개수에 다해 선형의 값을 가질 것이라고 판단한다.

Fig. 3에서는 PDB(Protein Data Bank)[20]에서 다운로드한 단백질 데이터들을 이용하여 가속화 기법을 적용하기 진과 후의 실행 시간을 보여주고 있다. 모서리 추적 알고리듬은 MS Visual C++로 구현되었으며, 실험환경은 Pentium 4 2.4GHz CPU, 1GB RAM, MS Windows XP OS에서 이루어졌다. 그림에서와 같이 가속화기법의 적용 전에는 수행 시간이 제곱의 형태로 증가하고 있으며, 가속화기법의 적용 후에는 매우 많은 수행 시간의 단축이 이루어졌으며 구의 개수가 늘어남에 따라 선형의 형태로 증가하고 있음을 알 수 있다.

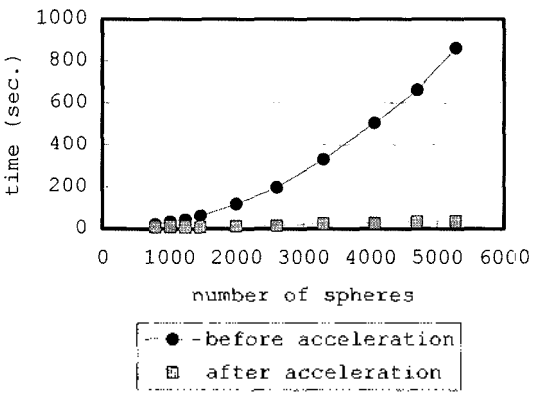


Fig. 3. Run-time behavior of the edge-tracing algorithm before and after applying acceleration techniques.

5. 영역 확장법

영역 확장법의 기본 개념은 구들의 중심점들의 보로노이 다이어그램이 주어져 있을 때, 각각의 보로노이 영역을 확장하여 전체 구들에 대한 보로노이 다이어그램을 구하는 것이다. 이때 영역의 확장이란 해당 구의 중심점의 보로노이 영역이 주어진 반지름을 가지는 구에 대한 보로노이 영역을 가지도록 일련의 위상연산을 통하여 변환하는 것을 말한다. 만약 하나의 점 제너레이터(generator)를 선택하고 해당 보로노이 영역을 확장하게 되면 나머지(일부는 구로 확장이 끝난) 제너레이터들과 선택한 구로 정의되는 보로노이 다이어그램이 완성된다. 이렇게 모든 구에 대해서 영역 확장을 하게 되면 우리가 원하는 삼차원에서 구들의 보로노이 다이어그램을 구할 수 있다.

설명의 편의를 위하여 다음과 같이 용어를 정의한다. 영역 확장이 진행되고 있는 보로노이 영역을 확장 영역이라 하겠다. 확장이 진행중인 확장 영역의 구를 확장 구라 한다. 확장 영역의 경계를 구성하는 보로노이 꼭지점, 모서리, 면을 각각 on-vertex, on-edge, on-face라 하겠다. 그리고 확장 영역의 경계를 구성하지는 않지만 그것에 인접한 보로노이 모서리 및 면을 각각 radiating-edge, radiating-face라 하겠다. 또 확장 영역과는 떨어져 있는 꼭지점, 모서리, 면은 off-vertex, off-edge, off-face라 하겠다.

5.1 영역 확장과 위상의 변화

이제 하나의 제너레이터에 대한 영역 확장을 좀 더 자세히 살펴보도록 하겠다. 하나의 구의 반지름을 연속적으로 늘리게 되면 해당 보로노이 영역도 연속적으로 확장되게 된다. 이때 늘어난 보로노이 영역은 이전의 보로노이 영역을 항상 포함하게 된다. 그리고 확장 영역의 각 on-vertex는 해당 꼭지점에 인접해있는 radiating-edge를 따라서 확장 영역의 바깥쪽으로 이동하게 된다. 이와 비슷하게 각각의 on-edge들은 인접해 있는 radiating-face를 따라서 바깥쪽으로 이동하게 된다. 여기서 충분히 작은 규모의 확장은 보로노이 다이어그램의 위상 정보를 변경시키지는 않으며 확장 영역의 경계를 이루는 보로노이 꼭지점의 위치, 모서리의 곡선식, 면의 곡면식 등의 기하 정보만을 변화시킨다.

그러나 영역 확장이 계속 진행되다 보면 일정한 시점에 위상 정보에 대한 변화가 발생하게 된다. 예를 들어, 하나의 on-vertex가 인접해있는 radiating-edge를 따라서 이동하다가 반대편의 off-vertex에 이르게 되면 해당 radiating-edge는 순간적으로 점으로 축소되었다가 이후로는 사라지게 되어서 주변의 위상 정보에 변화를 가져오게 된다. 본 논문에서는 이와 같은 위상 정보에 변화를 가져오는 상황들을 이벤트라 부르겠다.

보로노이 영역은 해당 구의 중심점에 대해 언제나

star-shape인 것으로 알려져 있기 때문에 영역 확장 중에 보로노이 면의 내부에서 서로 교차하는 현상은 발생하지 않고 오직 경계인 on-edge들에서만 일어난다. 이는 영역 확장 과정에서 새로운 위상 변화를 감지할 때 보로노이 꼭지점과 모서리의 변화만을 고려하여도 충분하다는 것을 의미한다. 게다가 on-vertex들과 on-edge들은 각각 해당 radiating-edge 및 radiating-face를 따라서 이동한다. 그리고 이벤트란 on-vertex들과 on-edge들이 radiating-edge 및 radiating-face를 따라서 이동하다가 off-vertex 및 off-edge를 만나게 될 때 발생하게 되므로 현재 시점에서 가장 가까운 다음 이벤트는 항상 radiating-face를 둘러싸는 모서리들 중 하나에 의해 발생하게 된다. 그러므로 새로운 위상변화를 감지하기 위해서 전체 모서리가 아닌 radiating-face를 둘러싸는 모서리들만을 검사하여도 충분하다는 결론에 이르게 된다.

참고로 on-edge를 제외한 각각의 모서리가 하나의 이벤트와 관련되어있다. 하지만 일반적으로 해당 구의 크기가 유한하기 때문에 모든 모서리에 대하여 이벤트가 발생하지는 않고 일부분인 해당 구 주변의 모서리들에 대해서만 이벤트가 발생하게 된다. 주변의 모서리들이란 해당 구의 radiating face의 경계를 이루는 모서리들임을 설명하였으며, 이벤트가 발생함에 따라서 새로운 radiating-face가 생겨나게 되는 경우 해당 면의 경계를 이루는 새로운 모서리들을 통하여 추후 발생 가능성이 있는 이벤트들을 파악할 수 있다.

5.2 이벤트의 종류

모서리 하나에 대하여 가능한 위상의 변화는 다음과 같이 세 가지의 형태로 나타날 수 있음을 밝혔다.

a. end-event란 모서리의 한쪽 끝 꼭지점에서부터 모서리의 축소가 모서리를 따라서 계속하여 일어나다가, 나머지 한쪽 끝 꼭지점에 이르게 되면서 해당 모서리가 사라지게 되는 경우를 말한다.

b. mid-event란 모서리의 양 끝 꼭지점이 각각의 반대편 꼭지점 방향으로 축소가 일어나다가, 모서리의 가운데에서 만나서 해당 모서리가 사라지게 되는 경우를 말한다.

c. split-event란 이동하던 on-edge가 해당 모서리와 끝 부분이 아닌 해당 모서리의 가운데에서 만나게 되면서, 해당 모서리가 두 개의 모서리로 나뉘게 되는 경우를 말한다.

정확하고 일관성을 가지도록 위상 정보를 유지하기 위해서는 이벤트의 종류에 따른 적절한 위상 연산을 취하여야 한다.

5.3 이벤트의 처리

하나의 모서리에 의해 하나의 이벤트가 발생되지만, 전체 위상구조의 변화라는 관점에서는 하나가 아닌 여러 개의 모서리가 동시에 하나의 이벤트를 발생시키는 경우가 존재한다. 예를 들어 end-event를 가지는 두 개의 모서리가 동일한 꼭지점에서 일어나는 경우가 있을 수 있는데, 이 때에는 하나의 모서리에서만 end-event를 가지는 경우와는 다르게 처리해야 한다. 그래서 end-event의 경우, 전체 위상 정보의 변화라는 관점에서 one-end-event와 two-end-event로 구분한다. 나머지 mid-event와 split-event는 각각 같은 곳에서 동시에 발생할 수 없기 때문에 전체 위상정보의 입장에서 또다시 세분화 될 필요가 없다. 그러므로 영역 확장에서는 다음과 같이 총 네 개의 서로 다른 이벤트가 존재한다: a. one-end-event, b. two-end-event, c. mid-event, d. split-event.

Fig. 4에서는 이러한 네 종류의 이벤트를 설명하고 있는데, 각 그림의 왼쪽은 이벤트가 일어나기 전의 상태를 보여주고 있고, 오른쪽은 이벤트가 일어난 후의 상태를 보여주고 있다. 그리고 왼쪽 그림에서 굵게 그려진 모서리는 이벤트 모서리를 나타내며, 오른쪽 그림의 굵게 그려진 모서리는 이벤트가 일어난 후 새로이 생겨난 모서리를 나타낸다.

Fig. 4(a)에서와 같이 one-end-event일 경우에는, 이벤트 모서리는 사라지고 세 개의 새로운 on-edge가 생겨서 하나의 새로운 on-face를 생성하게 된다. Two-end-event의 경우에는 두 개의 해당 이벤트 모서리들은 사라지게 되고, 이벤트 모서리와 하나의 on-edge로 싸여있는 radiating-face가 사라져서 하나의 on-edge를 새로이 생성한다. mid-event의 경우에는 이벤트 모서리와 하나의 on-edge 및 그들로 싸여있는 radiating-

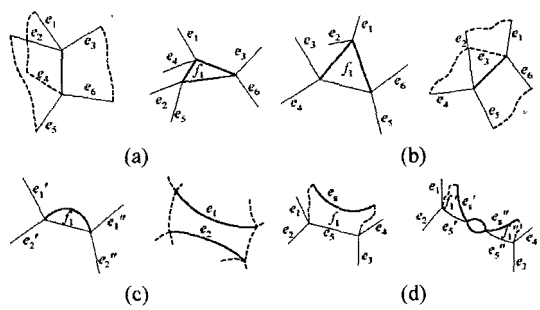


Fig. 4. Handling of events: (a) one-end-event, (b) two-end-event, (c) mid-event, and (d) split-event.

face가 사라지게 되고, 두 개의 on-face들이 하나로 합쳐지게 된다. 마지막으로 split-event의 경우에는 이벤트 모서리와 해당 radiating-face가 둘로 나뉘고, 가운데 두 개의 새로운 on-edge들로 둘러싸인 하나의 on-face가 생기게 된다. 이 경우 새로이 나뉘어진 모서리들은 각각 새로운 이벤트를 가지게 된다.

각 이벤트에 대하여 해당 이벤트가 발생하는 시점을 파악하여 순서에 맞게 처리해야 할 필요가 있다. 이벤트의 발생시점은 해당 이벤트가 일어나는 순간 해당 구의 반지름으로 정의된다. 우선 end-event의 경우는 on-vertex가 해당 off-vertex를 만나는 순간 발생하므로, off-vertex에 해당하는 집구에 확장 구가 접하는 순간으로 계산된다. 그러므로 집구와 확장 구의 중심 사이의 거리로 이벤트 시점이 비교적 쉽게 계산된다. Mid-event와 split-event의 정확한 이벤트 시점을 구하기 위해서는 4차 다항식의 근을 구하여야 하는 것으로 판명 되었다. 이는 해당 모서리의 위의 점을 중심으로 하고 그 모서리를 정의하는 구에 접하는 구들로 정의되는 swept-volume에 확장 구가 접하는 시점으로 계산이 된다.

5.4 알고리듬

하나의 보로노이 영역에 대한 영역 확장 알고리듬은 다음과 같이 정리할 수 있다. 전체 구에 대한 보로노이 다이어그램을 계산하기 위해서는 우선 구들의 중심점에 대한 보로노이 다이어그램을 구한 다음 아래의 영역 확장 알고리듬을 전체 구에 대해 하나씩 적용하여야 한다.

Algorithm: Region-Expansion(s_i)

1. Find all the edges defining radiating-faces and insert into a set E.
2. For each edge $e \in E \backslash E_{on}$, determine its event time t_e and event type. If $t_e < r_i$, insert the edge e into the event queue Q which implements a priority queue.
3. Pop an edge from Q and check its event type. If the event type is end-event, test if it causes to one-end-event or two-end-events by checking the next edge in Q.
4. Perform an appropriate action for the detected event. Then, find new edges bounding new radiating-faces and insert them into a set E^*. Perform Step 2 for all edges in the edge set E^*.
5. Repeat Step 3 and 4 until Q is empty.

5.5 시간 계산량

보로노이 영역 하나를 확장하는 데에는 최악의 경우 $O(m \log n)$의 시간이 걸리는 것으로 판명되었다. 왜냐하면 m을 보로노이 모서리의 개수라 할 때, 최악의 경우 모든 모서리에 대하여 이벤트가 발생할 수 있기 때문이다. 그리고 발생하는 이벤트의 수를 m이라고 할 때, m은 최악의 경우 $O(n^2)$가 될 수 있으므로, 총 계산시간은 최악의 경우 $O(n^2 \log n)$이 된다. 참고로 삼차원 구들에 대한 보로노이 다이어그램은 최악의 경우 $O(n^2)$개의 보로노이 모서리들로 구성될 수 있다[19].

그런데 이벤트가 발생하는 시점에서의 위상 정보를 이용하면 각 모서리와 그것에 이웃한 모서리들에 할당되어있는 이벤트들 사이의 우선순위를 이용하여 DAG(directed acyclic graph)를 구성할 수 있고, DAG에서의 위상 정렬(topological ordering)을 통하여 이벤트의 처리 순서를 정할 수 있다. 그리고 DAG에서의 위상 정렬은 DAG의 내부 노드 개수에 대해 선형의 시간이 요구되는 것으로 잘 알려져 있다[7]. 그러므로 하나의 보로노이 영역 확장 시에 최악의 경우 $O(m)$의 시간이 요구된다.

그러므로 전체 구들의 보로노이 다이어그램을 구하기 위해서 이러한 영역 확장을 구들의 개수인 n번 반복 수행해야 하기 때문에 전체 구들에 대해서는 $O(nm)$의 계산 시간이 요구된다.

그러나 본 저자는 전체 보로노이 다이어그램을 계산하는데 걸리는 평균 시간은 모서리의 개수인 $O(m)$으로 줄어들 수 있을 것이라 기대하고 있다. 그리고 영역 확장 전에 모든 구들을 가장 작은 구의 반지름만큼 줄여놓고 시작하면 영역 확장 시의 계산을 줄일 수 있을 것이다.

영역 확장 알고리듬 또한 MS Visual C++로 구현되었으며, PDB로부터 다운로드 받은 단백질 데이터들을 대상으로 하여 실험을 하였다. Fig. 5에서는 영역 확장 알고리듬과 가속화 기법 적용 후의 모서리 추적 알고리듬의 실제 수행 시간을 함께 보여주고 있다. 그림에서 보이는 것과 같이 영역 확장 알고리듬의 수행 시간은 주어진 구의 개수가 늘어남에 따라 선형의 형태로 증가함을 알 수 있다. 이는 단백질 데이터에 대해서는 원자들의 분포가 비교적 균등하게 되어있기 때문에, 보로노이 영역 하나가 확장할 때 평균적으로 상수개수의 이벤트가 발생하였기 때문인 것으로 유추할 수 있다. 또한 두 알고리듬의 실제 수행 시간은 PDB데이터에 대해서는 매우 비슷한 패턴을 보임을 확인할 수 있다.

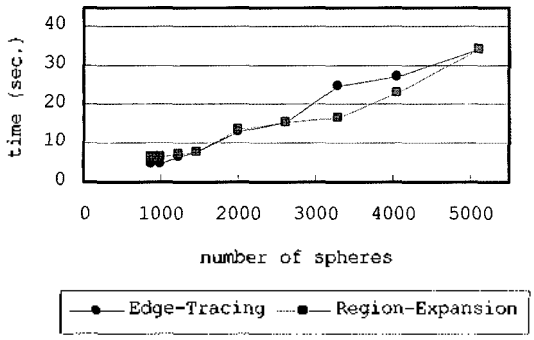


Fig. 5. Run-time behavior of the region-expansion algorithm and edge-tracing algorithm.

6. 응용 예

구들의 보로노이 다이어그램을 구하는 두 가지 알고리듬은 모두 구현이 되었고 단백질 데이터에 대해서 실행을 해보았다. Fig. 6에서는 단백질을 구성하는 원자들에 대한 보로노이 다이어그램과 그것을 이용한 응용 예들을 보여주고 있다. Fig. 6(a)에는 PDB (protein data bank)[20]에서 다운로드 받은 833개의 원자로 구성되어 있는 단백질 데이터(PDB code: 1fkf)를 보여주고 있으며, Fig. 6(b)에는 계산된 보로노이 다이어그램을 보여주고 있다.

구현된 보로노이 다이어그램의 응용 예로 molecular surface의 계산을 들 수 있다. Molecular surface란 분자를 구성하는 원자들이 삼차원 공간상에 주어져 있을 때 일정한 크기를 가지는 구로 만들어지는 분자의 블렌딩 곡면으로 정의된다. 구의 보로노이 다이어그램이 구해져 있다면 그것의 위상정보를 이용하여 블렌딩 곡면이 생성되는 위치를 정확하고 매우 효율적으로 찾아낼 수 있다[22]. Fig. 6(c)에서는 물 분자를 구로 근사하였을 때의 반지름인 1.4Å의 구로 molecular surface를 생성한 예이다. 그러므로, 본 예에서의 molecular surface는 용매인 물과 해당 단백질 사이의 경계를 의미한다.

또 다른 한가지 응용 예로 단백질에서 움푹 들어가 있는 영역으로 정의되는 포켓을 찾는 문제를 들 수 있다. 이는 신약 개발 시에 매우 근본적이면서도 중요한 문제로 잘 알려져 있다. 이 문제의 경우 구의 보로노이 다이어그램이 단백질을 구성하는 원자들 사이의 근접 정보를 매우 잘 표현하고 있기 때문에 공간상에서 주위 원자들 사이의 관계를 적절히 파악하여 포켓을 찾는 데에 매우 효과적으로 적용할 수 있었다[24]. Fig. 6(d)에서는 보로노이 다이어그램을 이용하여 찾아낸 포켓을 붉은색 영역으로 보여주고 있다.

구의 보로노이 다이어그램은 단백질 사이의 상호작용을 파악하는데 적용할 수 있다. 단백질간의 상호작용을 통하여 대부분의 생명현상이 일어나기 때문에 많은 생물학자들은 이러한 상호작용을 파악하는 데에 많은 노력을 기울이고 있다. 만약 어떠한 단백질이 서로 다른 몇 개의 그룹으로 구성되어 있다면, 서로 다른 그룹 사이의 보로노이 면의 집합으로 상호작용 인터페이스를 매우 설득력 있게 정의할 수 있다는 관찰을 하였다[21]. Fig. 6(e)에서는 1,074개의 원자로 이루어진 단백질(PDB code: 1bh8)를 보여주고 있는데, 서로 다른 두 개의 그룹으로 구성되어 있으며 서로 다른 색으로 표현하였다. 그리고 두 그룹 사이에는 보로노이 면의 부분으로 정의한 상호작용 인터페이스를 함께 나타내었다. Fig. 6(f)에서는 6(c)의 상호작용 인터페이스만 따로 가시화 하였다.

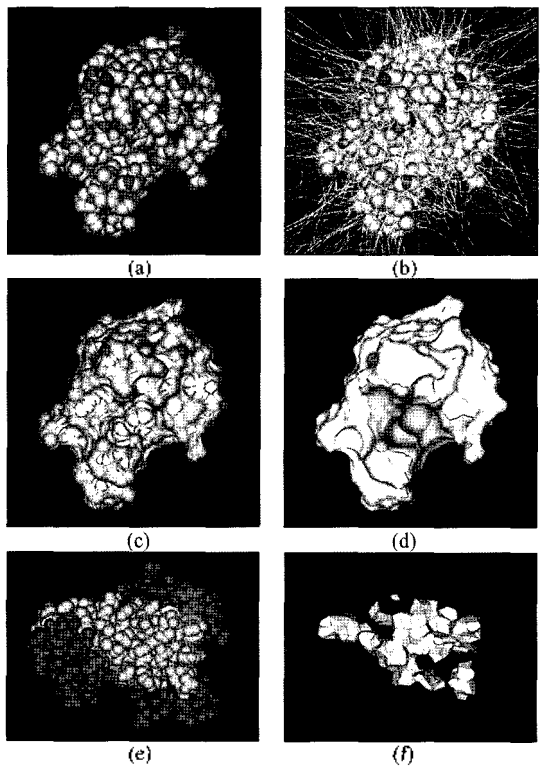


Fig. 6. Voronoi diagram and its applications to protein structure analysis.

7. 결 론

본 논문에서는 삼차원 구들에 대한 보로노이 다이

어그램을 구하는 알고리듬인 모시리 추적법과 영역 확장법을 제시하였다. 모서리 추적법 및 영역 확장법은 모두 $O(mn)$의 최악 경우 시간 계산량을 가진다.

하지만 단백질 데이터의 경우에는 한정된 종류의 원자들로 구성되어 있고, 원자들이 비교적 균등하게 분포되어있기 때문에 기대되는 시간 계산량은 훨씬 줄어들 것으로 판단하고 있다. 그리고 다양한 가속 기법들을 적용함으로써 실제 계산 시간을 더욱 개선할 수 있을 것이라 기대하고 있다.

그리고 본 연구에서 제안한 알고리듬을 바탕으로 하는 구들의 보로노이 다이어그램 다양한 응용 분야에 활발히 적용시키기 위해서는 수치오차에 대하 더욱 강건하게 만들 필요가 있는데, 이를 위하여 exact arithmetic 기법 및 topology-oriented 기법 등 다양한 exact computation 기법들을 적용하는 방안에 대한 연구가 계속하여 진행되어야 할 것이다.

감사의 글

본 연구는 과학기술부 창의적연구진흥사업의 지원으로 이루어졌으며 이에 감사드립니다.

참고문헌

1. Angelov, B., Sadoc, J.-F., Jullien, R., Soyer, A., Mornon, J.-P. and Chomilier, J., "Nonatomic Solvent-driven Voronoi Tessellation of Proteins: An Open Tool to Analyze Protein Folds", *Proteins: Structure, Function, and Genetics*, Vol. 49, No. 4, pp. 446-456, 2002.
2. Aurenhammer, F., "Power Diagrams: Properties, Algorithms and Applications", *SIAM Journal of Computing*, Vol. 16, pp. 78-96, 1987.
3. Boissonnat, J. D. and Karavelas, M. I., "On the Combinatorial Complexity of Euclidean Voronoi Cells and Convex Hulls of d-dimensional Spheres", in *Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 305-312, 2003.
4. Gavrilova, M., *Proximity and Applications in General Metrics*. Ph.D. thesis: The University of Calgary, Dept. of Computer Science, Calgary, AB, Canada; 1998.
5. Gavrilova, M. and Rokne, J., "Updating the Topology of the Dynamic Voronoi Diagram for Spheres in Euclidean d-dimensional Space," *Computer Aided Geometric Design*, Vol. 20, No. 4, pp. 231-242, 2003.
6. Goede, A., Preissner, R. and Frömmel, C., "Voronoi Cell: New Method for Allocation of Space Among Atoms: Elimination of Avoidable Errors in Calculation of Atomic Volume and Density", *Journal of Computational Chemistry*, Vol. 18, No. 9, pp. 1113-1123, 1997.
7. Goodrich, M. T. and Tamassia, R., Data Structures and Algorithms in Java. 2nd ed. New York: John Wiley & Sons, 2001.
8. Kim, D.-S., Cho, Y. and Kim, D., "Edge-tracing Algorithm for Euclidean Voronoi Diagram of 3D Spheres", in *Proc. of the 16th Canadian Conference on Computational Geometry*, pp. 176-179, 2004.
9. Kim, D.-S., Cho, Y., Kim, D., Kim, S., Bhak, J. and Lee, S.-H. "Euclidean Voronoi Diagrams of 3D Spheres and Applications to Protein Structure Analysis", In *Proc. of the International Symposium on Voronoi Diagrams in Science and Engineering*, pp. 137-144, 2004.
10. Kim, D.-S., Cho, Y., Kim, D., Kim, S., Bhak, J. and Lee, S.-H. "Euclidean Voronoi Diagrams of 3D Spheres and Applications to Protein Structure Analysis", *Japan Journal of Industrial and Applied Mathematics*, Vol. 22, No. 2, pp. 251-265, 2005.
11. Kim, D.-S., Cho, Y. and Kim, D., "Euclidean Voronoi Diagram of 3D Balls and Its Computation via Tracing Edges", *Computer-Aided Design*, Vol. 37, No. 13, pp. 1412-1424, 2005.
12. Luchnikov, V. A., Medvedev, N. N., Oger, L. and Troadec, J.-P. "Voronoi-Delaunay Analysis of Voids in Systems of Nonspherical Particles", *Physical Review E*, Vol. 59, No. 6, pp. 7205-7212, 1999.
13. Montoro, J. C. G. and Abascal, J. L. F., "The Voronoi Polyhedra as Tools for Structure Determination in simple Disordered Systems", *The Journal of Physical Chemistry*, Vol. 97, No. 16, pp. 4211-4215, 1993.
14. Naberukhin, Y. I., Voloshin, V. P. and Medvedev, N. N., "Geometrical Analysis of the Structure of simple Liquids: Percolation Approach", *Molecular Physics*, Vol. 73, pp. 917-936, 1991.
15. Okabe, A., Boots, B., Sugihara, K. and Chiu, S. N., *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. 2nd ed. Chichester: John Wiley & Sons, 1999.
16. Richards, F. M., "The Interpretation of Protein Structures: Total Volume, Group Volume Distributions and Packing Density", *Journal of Molecular Biology*, Vol. 82, pp. 1-14, 1974.
17. Sastry, S., Corti, D. S., Debenedetti, P. G. and Stillinger, F. H., "Statistical Geometry of Particle Packings. I. Algorithm for Exact Determination of Connectivity, Volume, and Surface Areas of Void Space in Monodisperse and Polydisperse Sphere Packings", *Physical Review E*, Vol. 56, pp. 5524-5532, 1997.
18. Voloshin, V. P., Beaufils, S. and Medvedev, N. N., "Void Space Analysis of the Structure of Liquids", *Journal of Molecular Liquids*, Vol. 96-97, pp. 101-112, 2002.
19. Will, H.-M., *Computation of Additively Weighted*

Voronoi Cells for Applications in Molecular Biology. Ph.D. thesis, Swiss Federal Institute of Technology, Zurich, 1999.
20. RCSB Protein Data Bank Homepage. http://www.rcsb.org/pdb/.
21. 김정민, 조영송, 이병훈, 서정연, 박상민, 원성인, 김동욱, 김덕수, "단백질간의 상호작용 인터페이스: 보로노이 다이어그램을 이용한 접근법", 한국 CAD/CAM학회 학술발표회 논문집, pp. 936-941, 2005.
22. 류중현, 박노훈, 이병훈, 조영송, 김동욱, 김덕수, "단백질의 Molecular Surface 계산: 보로노이 다이어그램과 볼 블렌딩을 이용한 접근법", 한국 CAD/CAM학회 학술발표회 논문집, pp. 931-935, 2005.
23. 이상현, 이건우, "비다양체 형상 모델링을 위한 간결한 경계 표현 및 확장된 오일러 작업자", 한국 CAD/CAM학회 논문집, Vol. 1, No. 1, pp. 1-19, 1996.
24. 조철형, 조영송, 김동욱, 김덕수, "단백질의 포켓인식: 보로노이 다이어그램과 convex hull을 이용한 접근법", 한국 CAD/CAM학회 학술발표회 논문집, pp. 685-688, 2005.
25. 최국헌, 한순흥, 이현찬, "선택 저장을 이용한 복합 다양체 자료구조", 한국 CAD/CAM학회 논문집, Vol. 2, No. 3, pp. 150-160, 1997.

김 동 욱

1999년 한양대학교 산업공학과 학사
2001년 한양대학교 산업공학과 석사
2004년 한양대학교 산업공학과 박사
2004년~현재 한양대학교 Voronoi Diagram 연구단 전임연구원
관심분야: Voronoi diagrams, computational geometry, biological applications

조 영 송

1995년 한양대학교 산업공학과 학사
1997년 한양대학교 산업공학과 석사
2003년 한양대학교 산업공학과 박사
2003년~현재 한양대학교 Voronoi Diagram 연구단 전임연구원
관심분야: geometric modeling, computational geometry

김 덕 수

1982년 한양대학교 산업공학과 학사
1985년 New Jersey Institute of Technology 산업공학과 석사
1990년 The University of Michigan 산업공학과 박사
1989년~1991년 Schlumberger Technology CAD/CAM Co. Senior Software Engineer
1991년~1995년 삼성 종합 기술원 선임 연구원
1995년~현재 한양대학교 산업공학과 교수
관심분야: Voronoi diagrams, geometric modeling, computational geometry