Delaunay Triangulation 을 이용한 공간데이타마이닝

강 인 수*, 이 기 준
부산대학교 전자계산학과

Spatial Data Mining Using Delaunay Triangulation

In-Soo Kang* and Ki-Joune Li
Department of Computer Science, Pusan National University
e-mail : {iskang, lik}@spatios.cs.pusan.ac.kr

요 약

지금까지 몇 가지 공간데이타마이닝 방법이 제안되었다. 이들 방법들은 몇 가지 문제점을 가지고 있다. 첫번째 문제점은 이들 방법이 객체 사이의 거리를 이용한 단순한 클러스터링 방법에 의존하고 있어 복잡한 형태를 가지는 분포에 대해서는 적용하기 힘들다는 것이다. 예를 들어, 호수를 둘러싸고 형성된 집들의 위치로 공간객체마이닝을 할 경우, 기존의 방법을 이용하여 이들 집들의 분포에 관한 정확한 패턴을 찾아내기 힘들다. 두 번째 문제점은 기존의 방법으로는 공간객체 분포의 계층적인 특성을 알아낼 수 없다는 것이다. 많은 경우, 공간객체의 분포는 계층적인 특성을 가지고 있어 이를 찾아 낼 수 있는 공간데이타마이닝 방법이 필요하다. 본 논문에서는 위의 두 가지 문제, 즉 복잡한 형태의 분포와 계층적인 분포를 가지는 공간객체를 위한 공간데이타마이닝 방법을 제안한다. 이 방법은 Delaunay Triangulation 을 이용하여 복잡한 공간객체의 분포도 패턴을 찾아주며 계층적인 구조도 함께 알아낸다. Delaunay Triangulation 을 이용할 경우 *O(nlogn)*의 시간에 데이타마이닝이 가능하며 삽입과 삭제에 대해 동적인 특성도 가질 수 있다. 또한 공간객체의 수가 많아서 디스크의 입력과 출력이 많은 경우에도, 공간색인방법을 쉽게 적용할 수 있어 성능을 크게 저하시키지 않는다.

1. 개요

최근 들어 지리정보시스템과 같이 방대한 공간데이타를 처리하는 응용분야가 떠오르고 있다. 이와 같은 응용분야를 위해서는 단순히 데이타베이스의 효과적인 관리뿐 아니라 주어진 데이타의 효과적인 분석도 매우 중요하다. 특히 많은 공간적인 데이타로부터 특성이나 패턴을 찾아내는 공간데이타마이닝은 공간데이타의 분석을 위해서 매우 중요한 기능이다. 이를 위하여 최근 들어 몇몇 연구가 공간데이타마이닝을 위해 이루어졌다.

공간데이타마이닝은 많은 공간객체의 데이타를 통해 숨겨져 있는 공간적 지식을 발견하는 것으로 다음과 같은 조건들을 만족시켜야 한다. 첫번째로, 공간데이타마이닝은 대량의 공간객체를 그 대상으로하여 공간적 지식을 찾아내는 것이므로 빠른 속도의 처리가 필요하다. 특히, 디스크의 입출력 시간을 가능한 최소로 줄여야 한다. 빠른 처리속도 조건에는 공간객체의 삽입과 삭제 등의 변동에 대해 동적인 특성도 포함된다. 두번째 조건은 공간데이타마이닝을 통해 자세하고 풍부한 공간적 지식을 얻을 수 있어야 한다는 것이다. 기존의 공간데이타마이닝 방법들은 주로 통계적인 방법을 통해 패턴을 찾아낸다. 그러나 이와 같은 방법을 통해서는 단순한 공간객체의 그룹을 분류하는 것 이상의 결과를 기대하기 곤란하다. 공간데이타마이닝은 그것 이상으로 공간객체의 분포에 관한 공간적 지식을 제공하여야 한다. 마지막 조건은 몇 개의 예외자를 효과적으로 처리할 수 있어야 한다. 예외자에 대해서 너무 민감하게 반응하는 것은 공간데이타마이닝에게는 이롭지 못하다. 경우에 따라서는 사용자가 예외자에 대한 반응의 민감도를 제어할 수 있는 방법을 제공해야 한다.

지금까지 제안된 공간데이타마이닝 방법은 주로 통계적인 방법 또는 클러스터링에 의한 방법을 이용하고 있다. 이들 방법은 대체로 위의 첫번째와 마지막 조건은 만족시키지만 두번째 조건을 만족시키는 공간데이타마이닝 방법은 아직 제안되거나 개발되지 않았다.

본 논문에서는 공간객체의 분포에 관한 지식을 발견하는 공간데이타마이닝의 방법을 제안한다. 일반적으로 대부분의 경우 공간객체는 균등분포를 이루기보다는 편중된 분포를 이룬다. 따라서, 많은 공간객체들 집합에 숨겨져 있는 중요한 지식 중의 하나가 우리의 공간데이타마이닝 방법에 발견하려고 하는 분포에 관한 것이다. 우리의 방법은 앞에서 서술한 공간데이타마이닝 중 첫번째와 세번째의 조건은 물론 두번째 조건도 잘 만족시키는 방법을 제안하고자 한다. 이 방법은 기존의 방법과 같이 클러스터링에 의존하고 있다. 그러나 단순히 거리에 따라 공간객체의 그룹을 찾는 것이 아니라 공간객체가 클러스터링되어 있는 공간적 패턴에 관한 여러 가지 지식을 찾는다.

우리의 방법이 제공하는 공간적 지식은 크게 두 가지로 나누어 진다. 첫번째는 복잡한 분포에 대한 정보이다. 예를 들어 그림 1 과 같이 복잡하게 분포되어 있는 공간객체들에 대한 공간적 패턴을 제공한다. 그림 1 에서 점들은 공간객체들을 나타내는데, 공간객체들은 여러 그룹으로 형성하여 집중적으로 분포하고 있으며, 각 그룹들은 공간객체가 희박한 부분을 둘러싸는 원의 형태를 이루고 있다. 본 논문에서 제안하는 방법의 두번째 특징은 공간객체의 계층적인 분포에 관한 지식을 제공한다는 것이다. 많은 경우, 공간객체의 분포는 그림 2 에서와 같이 계층적으로 되어 있다. 우리의 방법은 이 계층적인 분포에 관한 정보를 제공하여 준다.

우리의 방법은 Delaunay Triangulation 방법에 기초하고 있다. 이 접근의 장점 중의 하나는 비교적 빠른 시간 내에 공간데이타마이닝이 가능하다는 것이다. 시간 복잡도는 공간객체의 수가 n 일 때 *O(nlogn)*이다. 또 한가지 이 접근 방법의 장점은 공간색인방법과 쉽게 결합할 수 있다는 것이다. 따라서 만일 공간데이타마이닝하려고 하는 공간객체들에 대한 공간색인이 만들어져 있다면, 이를 이용하여 디스크입출력 횟수

를 최소한으로 줄일 수 있다는 것이다. 마찬가지로 공간객체의 삽입/삭제에 대해 부분적인 자료만을 이용하고 수정하여 빠른 시간내에 처리될 수 있다.

본 논문은 다음과 같이 구성되어 있다. 먼저 2 장에서는 지금까지 제안된 공간데이타마이닝방법에 대해 살펴본다. 3 장에서는 본 논문에서 제안하는 방법을 소개하고 4 장과 5 장에서는 이 방법이 발견하는 분포에 관한 공간적 지식과 그러한 분포를 표현하는 방법에 대해서 알아본다. 그리고 6 장에서 결론과 향후 연구방향에 대해 살펴본다.

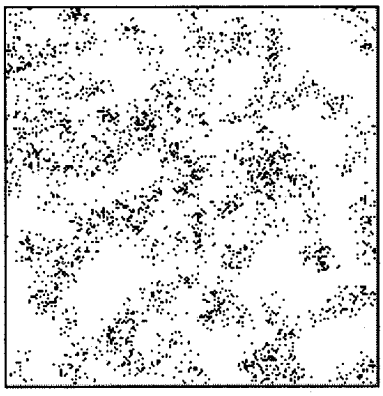

그림 1 복잡한 공간객체의 분포

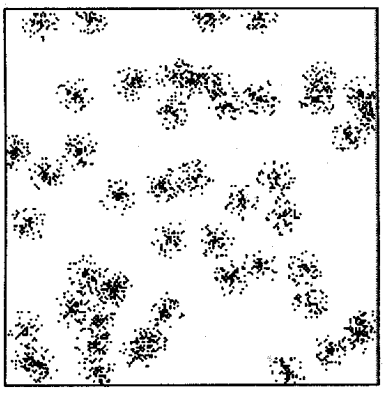

그림 2 계층적인 공간객체의 분포

2. 관련 연구

최근 들어 몇가지의 공간데이타마이닝방법이 제안되었다. 이 방법들은 주로 데이타 클러스터링방법에 의존하고 있다. 이들은 크게 공간색인방법을 위한 방법과 기하학적 클러스터링으로 분류될 수 있다. 이 두가지 방법은 모두 가까운 공간객체를 같은 그룹으로 나눈다는 것에서 동일하나, 목적에 있어서 서로 다르다.

공간색인방법에서 이루어지는 데이타클러스터링은 디스크입출력 횟수를 최소로 줄이기위해 인접한 공간객체를 같은 디스크블럭에 저장하는 것이 그 목적이다. 따라서 각 분류된 그룹의 공간객체는 하나의 디스크에 들어가도록 그 수를 제한해야 한다는 제약조건이 있다. 또한 일반적으로 계층적인 구조를 가지고 클러스터링된다. 이 방법의 대표적인 것이 Kriegel 이 제안한 R*-tree[6]를 이용하는 것이다. 이 R*-tree 는 공간객체를 색인하는 방법으로 최소경계사각형을 이용하는데, 인접한 공간객체들을 같은 부모 최소경계사각형내에 저장하며, 또한 각 최소경계사각형들은 다른 이웃하는 최소경계사각형들과 같은 부모 최소경계사각형에 저장된다. 그래서 서로 이웃하는 공간객체들은 하나의 최소경계사각형내에 모이게 되므로 어느정도 클러스터링이 되지만, 최소경계사각형의 경계영역에 존재하고 서로 다른 부모 최소경계사각형을 가지는 두 공간객체는 서로 인접해 있음에도 불구하고, 따로 떨어진 공간객체로 분류가 되며, 다른 디스크블럭에 저장된다. 그리고, 실제로 공간객체를 엑세스할 때에도 최소경계사각형을 기반으로하여 접근하기 때문에, 공간적인 인접성을 살릴 수 가 없다. 또한, 이 방법은 공간색인 자체가 그 목적이므로 이 방법만으로는 클러스터링을 이용하여 공간데이타마이닝을 하는 것은 거의 불가능하다.

반면에, 기하학적인 클러스터링은 공간색인방법이 하는 클러스터링과 같은 제약조건은 없고 공간객체의 밀집정도를 가장 잘 나타내는 클러스터링을 찾는 것이다. 따라서, 이 방법은 공간데이타마이닝에 보다 적절하게 사용될 수 있다. 대표적인 방법으로는 Han 이 제안한 CLARANS[2]과 Zhang 이 제안한 BIRCH[1]가 있다. CLARANS 는 Randomized 알고리듬을 기반으로 하여 k-medoid 를 이용하여 클러스터링을 하는 방법이다. 이 방법은 먼저 임의의 노드를 선택하여 현재 노드로 한 뒤에, 이 노드와 임의의 이웃노드를 선택하여 비교한 뒤에 이 두 노드 중 보다 나은 노드를 현재 노드로 선택한다. 그리고, 이웃노드를 선택하는 횟수를 최대이웃노드수를 넘지 않도록 하면서 이 과정을 반복한다. 이 과정이 끝나면 현재 노드를 부분적 최소값으로 하여 이 부분적 최소값이 지역노드수 만큼 되었을 때 이들 부분적 최소값에서 가장 좋은 노드를 찾아낸다. CLARANS 방법은 다음의 몇 가지 문제가 있다. 우선 데이타마이닝의 속도가 Randomized 알고리듬을 이용하여 상대적으로 느리다는 것이다. 특히, 디스크의 입출력시간을 전혀 고려하지 않아, 데이타의 갯수가 많은 경우에는 상당한 속도의 저하가 예상된다. 두번째 문제점은 최상의 클러스터링을 찾는 것이 보장되어 있지 않다는 것이다. 이는 Randomized 알고리듬의 제약때문이다. 세번째 문제점은 K-Mediod 를 이용하였기 때문에 공간객체가 복잡한 패턴을 가지고 분포되어 있을 경우에는 클러스터링의 결과가 공간데이타마이닝에 크게 도움이 되지 않는다. 특히, 공간객체가 계층적인 패턴을 가지고 분포되어 있을 경우에는 계층적 구조에 대한 아무런 단서를 제공해주지 못한다.

Zhang 이 제안한 BIRCH 는 위의 두가지 방법을 상당히 개선하였다. BIRCH 는 CF(Clustering Feature) Tree[1]를 이용하여 클러스터링을 하는 방법으로서, CF 는 하나의 벡터인데, 이 벡터를 이용하면 두 공간객체간의 거리관계를 알 수 있는데, 이를 이용하여 CF 트리를 구성하는 것이다. CF 트리는 공간객체에 대한 클러스터링 정보를 포함하고 있으며, 또한 하나의 노드의 크기가 작아서, 메모리에 생성이 가능하므로, 디스크 입출력시간에 대하여 고려할 필요가 없는 장점을 지닌다. 또한 클러스터링을 하는 수행 시간 측면이나 클러스터링의 결과에 대하여 앞서의 방법보다 뛰어나다. 그러나 이 방법도 앞에서 언급한 세번째 문제점을 효과적으로 해결하지 못하였다.

따라서 본 논문에서는 위의 방법들의 문제점을 보완한 새롭고 효과적인 공간데이타마이닝 방법을 제안한다. 이 방법의 특징은 공간객체 분포의 패턴을 가능한 자세하게 찾아낸다는 것이다. 기존의 방법들과 같이 단순하게 점을 중심으로 공간객체의 클러스터를 찾는 것이 아니라, 분포자체의 형태를 찾아낸다. 뿐만아니라 공간객체의 계층적 분포구조에 관한 정보도 함께 제공한다. 이는 응용분야에 따라 유용하게 이용될 수 있는 정보이다. 마지막으로 본 방법의 특성은 디스크 입출력이나 부분적 갱신때문에 발생하는 성능의 감소에 탄력적으로 대응할 수 있다는 장점을 가지고 있다. 그러면 다음 장에서 우리 방법에 대해서 자세히 살펴보기로 한다.

3. Delaunay Triangulation 를 이용한 클러스터링

본 논문에서는 공간객체를 Delaunay Triangulation 과정을 통하여 TIN(Triangulated Irregular Network)데이터로 재구성한 뒤에 이를 이용하여 클러스터링을 하는 방법인 MUTin (Mining Using Triangulation)을 제시한다. 본 논문에서 공간객체는 포인트 데이타인 것으로 한정할 것이다.

먼저 공간객체를 기존에 제안된 TIN을 구성하는 알고리듬 중에서 J.R. Shewchuk 가 제안한 Delaunay Triangulation 이라는 알고리듬을 이용하여 TIN을 구성한다. 이 TIN 데이타는 삼각형들의 집합으로 구성되어 있는데, 클러스터링을 하는 것은 각 공간 객체들이 얼마나 인접해 있는가를 고려하는 것이므로, 이들 삼각형의 공간적인 관계를 이용한다. 각 삼각형의 각 edge 중에서 길이가 가장 긴 edge(즉, 삼각형의 세 꼭지점간의 각각의 거리중의 최대 거리)를 적당한 threshold, T와 비교를 통하여 T보다 작거나 같으면 그것을 하나의 클러스터로 만드는 것이다. 이 방법의 이점은 하나의 공간객체에 대하여 나머지의 공간객체들과의 인접성을 비교할 때에 비교 대상이 되는 공간객체의 수가 매우 적어진다. 따라서, 보다 적은 비교회수로 공간객체를 클러스터링할 수 있다.

위에서 언급한 threshold T는 두 공간객체간의 거리가 어느정도 멀고 가까운지에 따라 클러스터로 묶을 것인지 아닌지를 결정하는 척도로써, 이 값의 크기에 따라 클러스터링의 정도가 달라지게 된다. 여기서는 좌표체계를 [0.0,1.0]-[0.0,1.0]의 2차원 실수 좌표계를 사용하며 T값의 범위는 [0.0,1.0]이다. 이 방법의 기본적인 알고리듬은 다음과 같다.

알고리듬 MUTin

1. 공간객체를 입력으로 받아서 Delaunay Triangulation 알고리듬을 이용하여 TIN을 구성한다.
2. 구성된 TIN 내의 각 삼각형에 대하여 삼각형의 가장 긴 변의 길이 L에 threshold T를 적용하여 L이 T보다 작거나 같은 삼각형을 구하여 초기 클러스터 집합을 생성한다.
3. 초기 클러스터 집합에 대하여 다음 과정을 반복한다.
4. 초기 클러스터 집합에서 임의의 삼각형 하나를 꺼낸 뒤, 이 삼각형을 원소로하는 클러스터를 하나 생성한다.
5. 이 클러스터와 인접한 모든 삼각형을 위의 집합에서 찾아내어 클러스터에 추가한다.
6. 더이상 인접한 삼각형이 없는 경우 위에서 생성한 클러스터를 클러스터 집합에 추가하고, 삼각형 집합내에 원소가 없을 때까지 4 번부터 다시 수행한다.

입력된 공간객체의 개수를 n, TIN 내의 삼각형의 개수를 m ($m<=2n$), 초기 클러스터집합내의 삼각형 개수를 t ($t<=m$), 생성된 클러스터의 개수를 c, 클러스터당 평균 삼각형의 개수를 s라고 할 때, TIN을 구성하는데 필요한 시간은 $O(nlogn)$이고, 초기 클러스터 집합을 생성하는데는 $O(m)$의 시간이 소요된다. 그리고, 실제 클러스터 집합을 생성하는 과정에 소요되는 시간 CGT(Cluster Generating Time)는 $CGT = t + (t-s) + (t-2s) + ... + (t-cs)$이다. 그런데, $cs=t$이므로, $CGT = (c+1)\,t - (c+1)\,t/2 = (c+1)\,t/2$이다. 따라서, 실제 클러스터 집합을 생성하는 데에는 $O(ct)$또는 $O(c^2s)$의 시간이 소요된다. 여기서, t는 T에 비례하며, 총 소요되는 시간은 클러스터의 개수 c와 클러스터당 평균 삼각형의 개수 s에 비례한다는 사실을 알 수 있다.

그림 3의 예는 초기 공간객체의 개수가 1000 개이고 threshold T가 0.05로 주어진 것이다. 이 공간객체를 TIN 으로 구성을 한 뒤에 생성되는 삼각형 데이타(TIN)의 개수는 초기 공간객체 개수보다 양 2 배정도 많은 약 1977 개이다. 이는 초기 공간객체의 분포에 따라 조금의 오차가 발생하지만 실험결과 모든 데이타들에 대하여 거의 2 배 정도가 생성된다. 이 삼각형 데이타를 이용하여 클러스터링을 한 결과, threshold 조건을 만족하는 삼각형의 개수는 1581 개 이며, 생성된 클러스터의 개수는 17 개이다. 그리고, 각 클러스터당 평균 데이타 개수는 93 개이다. 이 데이타에 대한 클러스터링 결과는 그림 4 에 나타나 있다.

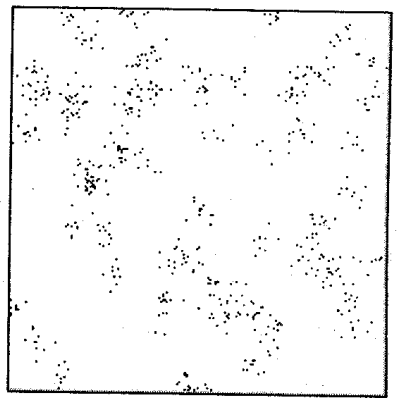

그림 3 공간객체의 분포 상태

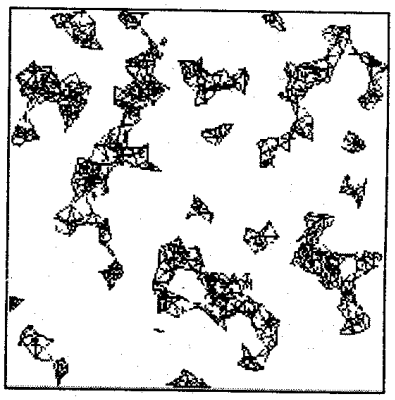

그림 4 TIN 구성 후 클러스터링 결과

4. 복잡한 형태의 클러스터링

3 장에서 우리는 Delaunay Triangulation 을 이용해 Clustering 하는 방법에 대해서 살펴보았다. 이 방법의 가장 중요한 특징 중의 하나는 복잡한 형태를 띠는 분포에 대한 공간데이타마이닝에 대해서도 효율적이라는 것이다.

그림 1 에서 보는 바와 같이 공간객체들은 어느 한 중심을 기준으로 모여 있는 것이 아니라, 임의의 중심 둘레로 분포되어 있음을 알수 있다. 이 경우에, 이 공간객체의 분포가 어떠한 형태로 이루어져 있는지를 알아내기가 쉽지 않다. 즉, 공간객체가 중심이 비어 있는 도우너스 형태의 분포를 이루는 경우 제대로된 클러스터링을 하기에는 기존의 방법들은 부적절 하다. 왜냐하면, 기존의 방법들은 단순히 공간객체간의 거리만을 클러스터링의 기준으로 삼고 있기 때문이다. 하지만 MUTin 은 공간객체간의 거리뿐만아니라, 클러스터링에 사용되는 데이타인 TIN 의 특성을 충분히 활용하여 위와 같은 형태의 공간객체 분포도 쉽게 찾아낼 수 있다. 클러스터를 이루는 TIN 내에서, 클러스터내의 가운데에 구멍이 있는 경우 이 구멍은 TIN 의 일부가 아닌 임의의 다각형이 된다. 따라서 이러한 다각형을 찾아내면 된다. 이러한 다각형을 찾아내는 방법은 아주 간단하다. 이는 TIN 이라는 특성이 있기 때문인데, 위의 다각형을 구성하는 edge 들은 TIN 내의 각 삼각형에 대하여 단 한번만 참조되기 때문이다. 따라서, TIN 내의 각 edge 에 대하여, 참조회수가 한번 뿐인 edge 들을 찾아내어 이

들을 다각형으로 구성하기만 하면 된다. 이 다각형을 contour 라고 부르는데, 이 contour 에는 두가지 종류가 있다. 클러스터의 전체 모양을 나타내는 outer contour 와 클러스터내의 구멍인 inner contour 이다. Outer contour 의 경우 이 클러스터의 분포형태를 파악하는데 중요한 역할을 하며, inner contour 의 경우에는 클러스터내의 각 공간객체들이 어떠한 상태로 분포를 이루어 나가는지에 대한 중요한 참고정보가 된다. 그림 1 에 대한 contour 는 그림 5 에서 보여지고 있다. Contour 를 생성하는 알고리듬은 다음과 같다.

알고리듬 Contour

1. 하나의 클러스터내에서 클러스터를 구성하는 각 edge 들에 대하여 edge E 의 참조횟수가 1 인 E 들을 원소로하는 초기 contour 집합 initC 를 구성한다.
2. initC 에서 하나의 edge E 를 선택한 뒤, contour C 를 하나 생성하고 여기에 E 를 추가한다.
3. 이 C 와 연결된 edge 들을 찾아서 C 에 추가한다.
4. 더이상 연결된 edge 가 없다면, C 를 Cset 에 추가한다.
5. initC 에 edge 가 남아 있을때 까지 위 과정을 반복한다.
6. 생성된 Cset 에 대하여 다른 contour 를 포함하는 contour 를 선택한다.
7. 6 번에서 찾아진 contour 가 outer contour 이며, contour 집합에 있는 나머지 contour 들은 모두 inner contour 들이다.

그림 5 복잡한 공간객체의 Contour

5. 계층적 클러스터링

클러스터링의 두번째 특성으로는 공간객체들이 계층적으로 구성되어 있다는 것이다. 공간객체들이 계층적으로 구성되어 있다는 것은 공간객체들이 하나의 커다란 클러스터를 구성하고, 그 클러스터 안에서 각 공간객체들이 또다시 작은 여러개의 클러스터로 구성되어 있다는 것이다. 이러한 특성은 공간객체의 분포 및 특성을 이해하는 데 또 하나의 유용한 정보가 될 수 있다. 본 장에서는 이러한 특성을 어떻게 알아내어 클러스터링을 할 것인가를 알아볼 것이다.

먼저 공간객체들이 계층적으로 구성되어 있으므로, 이들을 제대로 표현하기 위해서는 트리구조가 필요한데, 이를 ClusterTree 라고 한다. 이 ClusterTree 는 leaf node 의 height 가 동일하지 않고, 각 클러스터마다 다르게 나타난다. 이는 각 클러스터의 공간객체의 분포가 조밀할 수도 있고 희박할 수도 있는데, 이러한 특성이 클러스터의 계층성에 영향을 주기 때문이다. 이 ClusterTree 는 ClusterNode 라는 구조로 구성되는데, 이 ClusterNode 는 몇 개의 ClusterEntry 로 구성된다. 이 ClusterTree 는 공간객체들의 계층성과 공간적인 인접성을 표현하는데 아주 적합한데, 왜냐하면, 이 트리는 클러스터링된 공간객체들로 이루어진 각 클러스터들에 대하여 독립적으로 다시 클러스터링을 하기 때문이다. 이 ClusterTree 는 3 장에서 소개한 알고리듬을 이용하면 구성이 가능하다. 3 장에서 소개된 알고리듬은 공간객체들에 대하여 하나의 threshold 만을 적용하여 클러스터링을 하는 것인데, 이 방법의 결과로 나오는 각 클러스터들에 대하여 threshold 를 적당한 값만큼 줄인 뒤에 위의 알고리듬을 다시 적용하면 된다. 이 ClusterTree 를 구성하는 방법에 대한 알고리듬은 다음과 같다.

알고리듬 ClusterTree

1. 공간객체를 입력으로 받아서 클러스터트리의 루트노드를 생성한다.
2. 루트 노드에 초기 threshold startT 를 적용하여 알고리듬 ClusterNode 를 Call.

알고리듬 ClusterNode

1. 현재 T 가 stopT 보다 작을 경우 Halt.
2. 노드내의 TIN 을 입력으로하고 알고리듬 MUTin 을 Call.
3. 생성된 각 클러스터에 대하여 ClusterEntry 를 생성한다.
4. 각각의 ClusterEntry 에 대하여 threshold T 를 적용하여 알고리듬 ClusterNode 를 Call.

6. 결론

본 논문에서 제시하는 방법은 지금까지 연구된 클러스터링 방법들이 가지는 단점들에 대한 보완과 아울러, 클러스터링의 성능을 보다 향상시키고 보다 많은 공간정보의 획득에 있다. 즉, 복잡한 패턴의 공간객체 분포나, 계층적인 분포구조를 가지는 공간객체에 대한 클러스터링을 제공하여 보다 광범위한 공간객체에 대한 정보의 제공이 가능하다.

그러나, 본 논문에서 제안한 MUTin 은 현재 공간색인기법을 사용하지 않았기 때문에 실제 공간객체에 대한 적용에는 무리가 따르게 된다. 그리고, 공간객체에 대한 클러스터링을 batch 에 의해서 수행하는데, 이때 공간객체의 생성과 소멸이 이루어져야 하는 경우, 클러스터링은 언제나 전체 공간객체에 대하여 다시 해야하는 문제점이 있다.

따라서, 향후 연구해야할 방향은 다음의 두가지로 나누어 볼 수 있다. 첫째, 공간색인방법인 R*-tree 를 MUTin 에 결합하여 방대한 양의 공간객체들에 대한 클러스터링을 효과적으로 수행할 수 있도록 한다. 둘째, 임의의 공간객체의 동적인 삽입과 삭제에 대한 알고리듬에 대한 연구와 아울러 구현이 이루어져야 한다.

7. 참고문헌

[1] Tian Zhang, Raghu Ramakrishnan, and Miron Livny, "BIRCH : An Effieicent Data Clustering Method for Very Large Databases," Proc. ACM SIGMOD , 1996

[2] Raymond T. Ng, and Jiawei Han, "Effieicent and Effective Clustering Methods for Spatial Data Mining," Proc. VLDB Conference, Santiago, Chile, 1994

[3] Thomas Brinkhoff, and Hans-Peter Kriegel, "The Impact of Global Clustering on Spatial Database Systems," Proc VLDB Conference, Santiago, Chile, 1994

[4] Leonard Kaufman, and Peter J. Rousseeuw, "Finding Groups in Data - An Introduction to Cluster Analysis," Wiley Series in Probability and Mathematical Statistics, 1990

[5] Tian Zhang, Raghu Ramakrishnan, and Miron Livny, "BIRCH : An Effieicent Data Clustering Method for Very Large Databases, Technical Report," Computer Science Dept., Univ. of Widconsin-Madison, 1995

[6] Beckmann N., Kriegel H.-P., Schneider R., and Seeger B., "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," Proc. ACM SIGMOD, 1990

[7] F. Murtagh, "A Survey of Recent Advances in Hierarchical Clustering Algorithms", The Computer Journal, 1983

[8] Samet H., "The Design and Analysis of Spatial Data Structures," Addison-Wesley, 1990