분산 공간데이터베이스의 위치상 불일치를 위한 공간질의영역 변형

Transformation of Spatial Query Region for Resolving of Mismatching Distributed Spatial Databases

요약

분산 지리정보시스템을 구현하는데 가장 어려운 점 중의 하나는 공간데이터베이스의 불일치이다. 특히, 여러 가지 이유로 발생하는 분산 공간데이터베이스 사이의 위치 불일치는 공간 질의의 결과를 부정확하게 만든다. 하나의 간단한 해결책은 가장 중요한 사이트에 따라 각 사이트의 공간데이터베이스에 있는 위치 데이터를 보정하는 것이다. 그러나, 이러한 방법은 각 사이트의 자율성이 중시되어야 하는 경우에는 실용적이지 않다. 본 논문에서는 위치 데이터가 불일치하는 여러 개의 공간데이터베이스에 대하여 공간질의를 처리하는 방법을 제시한다. 본 논문에서 제시하는 방법은 각 지역 공간데이터베이스를 수정하지 않고 자율성을 보장하는 방식으로 공간질의를 처리할 수 있도록 한다. 본 논문에서 제안하는 방법은 탄성변형방법을 이용하면서, 각 질의의 위치를 동적으로 변환하는 방법에 기초하고 있다. 이 방법의 정확성은 수학적으로 증명이 되었으며, 실험으로도 확인하였다. 더욱이, 이 방법의 유용성 검증을 위하여 상용 데이터베이스시스템을 이용하여 구현하였다.

국문 키워드

탄성변형, 분산 공간데이터베이스, 들로니 삼각분할

Abstract One of the most difficult problems in building a distributed GIS lies in the heterogeneity of spatial databases. In particular, positional mismatches between spatial databases, which arise due to several reasons, may incur incorrect query results. They result in unreliable outputs of query processing. One simple solution is to correct positional data in spatial databases at each site, according to the most accurate one. This solution is however not practical in cases where the autonomy of each database should be respected. In this paper, we propose a spatial query processing method without correcting positional data in each spatial database. Instead of correcting positional data, we dynamically transform a given query region or position onto each space where spatial objects of each site are located. Our proposed method is based on an elastic transformation method by using delaunay triangulation. Accuracy of this method is proved mathematically, and is confirmed by an experiment. Moreover, we implemented using common use database system for usefulness verification of this method.

영문 키워드

Elastic transformation, Distributed spatial database, Delaunay triangulation

1. 서론

지리정보시스템의 이용이 활성화됨에 따라, 여러 분산 지리정보시스템에 대한 요구도 늘어나고 있다. 그러나, 분산 지리정보시스템이 본격적으로 사용될 수 있기 위하여서는 몇 가지 문제가 해결되어야 한다. 그 중에서 중요한 것은 이질성의 극복이다. 공간데이터베이스의 이질성은 공간데이터베이스 관리시스템의 이질성에서부터, 공간데이터모델의 이질성까지 다양하다. 본 논문에서 다루는 이질성은 공간데이터베이스의 위치 데이터 사이의 불일치에 따른 이질성이다.

공간데이터베이스의 위치에 대한 불일치는 여러 가지 이유에서 비롯된다. 구축하는 기관이 서로 다르면 불일치가 발생할 수 있고, 축척이 다르면 불일치가 발생한다. 예를 들어, 그림 1과 같이 두 개의 공간데이터베이스를 이용하는 지리정보시스템에서, 하나의 공간데이터베이스는 건물을 나타내는 지형도이고, 다른 하나의 같은 지역에 대한 교통도가 있다고 가정하자. 만일 두 개의 도면이 서로 독립적으로 구축되었다면 서로 불일치하는 데이터가 발생된다. 또한 사용자가 두 개의 도면에 동시에 같은 공간질의를 주면 그림1과 같이 처리가 되는데, 두 개의 도면은 서로 불일치하므로 질의처리의 결과는 서로 일관성을 보장할 수 없다. 예를 들어, '점 p 에서 가장 가까운 객체를 찾아라' 와 같은 질의를 주었을 때, 점 p 는 각각의 도면 위의 다른 위치로 대응된다. 결과적으로 각 도면에서 찾아지는 가장 가까운 객체는 서로 일관성이 없을 수 있다.

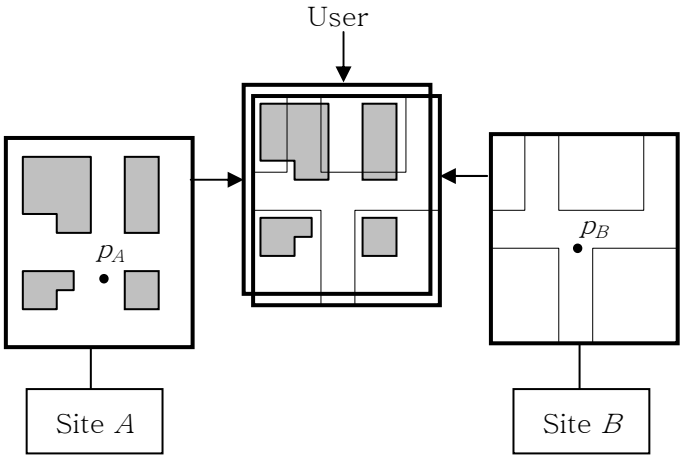


그림1. 불일치되는 공간데이타베이스

이 불일치의 문제를 가장 쉽게 해결하려면 사용자가 점 p 의 위치를 각각의 도면에 맞게 전환하여 질의를 주어야 한다. 그러나 일반적인 사용자는 불일치에 대한 자세한 정보는 알지 못하므로 불일치로 인한 질의의 처리과정은 시스템이 담당하여야 한다.

공간데이터베이스의 모델이나 스키마, 그리고 공간데이터베이스가 제공하는 기능에 대한 이질성 극복을 위하여 지금까지 많은 연구가 이루어졌다. ISO TC/211[1,2]이나 OGC[3]의 작업과 같이 표준화 작업은 이와 같은 이질성을 극복하는 대표적인 시도들이다. 그러나, 이와 같은 시도는 공간데이터의 내용에 대한 불일치보다는 기능이나 형식적인 이질성에 초점이 맞추어져 있다. 연합된 공간데이터베이스를 사용하기 위해서는 기능이나 형식적인 이질성뿐 아니라, 내용의 불일치도 함께 해결되어져야 한다. 본 논문에서는 이와 같이, 위치가 서로 불일치되는 연합 공간데이터베이스에서의 질의처리 방법을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 우선 공간데이터베이스의 위치 불일치의 문제에 대하여 지금까지의 연구를 살펴본다. 3장에서는 본 논문에서 사용되고 있는 들로니 삼각분할을 바탕으로 탄성변형을 이용한 불일치 보정방법을 설명하고,

4장에서는 연합 공간데이터베이스에서의 질의 처리 방법을 제안한다. 5장에서는 4장에서 제시한 알고리듬의 정확성을 알아보기로 한다. 6장에서는 본 논문에서 제시한 방법을 바탕으로 실험한 결과를 제시한다. 7장에서는 본 논문에서 제안하는 방법이 분산환경에서 처리되는 과정을 설명하고, 마지막으로 8장에서 결론을 맺는다.

2. 기존 연구 및 배경

다른 종류의 데이터베이스와 마찬가지로 공간데이터베이스에서도 여러 데이터베이스를 통합하여 사용하고, 이 경우 여러 가지 이질성의 문제가 발생한다. 현재까지 주로 다루어진 공간데이터베이스 사이의 이질성 문제는 크게 데이터모델의 불일치와 데이터베이스시스템의 기능적 이질성의 문제이었다. 그러나 본 논문에서 다루고 있는 문제는 공간데이터베이스의 객체의 위치 불일치 문제이다. 이 문제는 도면 불일치문제로 알려져 있다[4]. 지금까지의 도면 불일치 문제의 해결방법은 주로 탄성변형(Elastic Transformation, 또는 Rubber Sheeting)을 통하여 불일치되는 데이터를 보정하는 방식을 취하고 있다.

탄성변형에 대한 연구는 [5],[6-9],[10],[11]에서도 제시되고 있다. 가장 일반적으로 사용되는 탄성변형은 [10]에서는 불일치되는 부분을 보정하는 방법으로 Affine Transformation 방법을 제시하고 있다. 이 방법은 2차원 공간에서 주어지는 $T_{2.0} = T_{2.1} = 0$ 과, $T_{2.2} \neq 0$ 인 3×3 행렬을 이용하여 입력되는 점, 다각형 벡터, 원 등을 변형시키는 것이다. 예를 들어, (p_x, p_y, p_w)의 좌표를 가진 점 p를 변형할 경우 행렬은 다음과 같이 표현된다.

$$\begin{bmatrix} w & 0 & x \\ 0 & w & y \\ 0 & 0 & w \end{bmatrix} \xrightarrow{\text{(x/w, y/w)에 의해 변형}} \begin{bmatrix} a & -b & 0 \\ b & a & 0 \\ 0 & 0 & w \end{bmatrix}$$

이 방법에 의해 변형되어진 모양은 변형되기 전의 속성을 그대로 유지함으로써 불일치되는 부분들을 보정시키고자 하였다.

[8]에서는 복소수($z = x + iy$)와 가중치 함수를 사용함으로써 탄성변형의 정확한 방법을 제시하고 있다. 그러나, 이 방법은 직사각형과 병행성 같은 조건들을 통합하지 못한다. 따라서, [9]에서는 가중치 함수를 수정함으로써 [8]에서 제시한 방법을 좀더 향상시켰다. 이러한 방법들은 일반적인 탄성변형 방법으로써 사용되어졌다. 그러나, 이러한 방법들의 정확성은 불투명하며, 그것들을 만족시키기 위해서는 어려운 제약 조건들이 따른다. 예를 들어, 앞에서 제시한 탄성변형 방법에 의해 변형된 후에도 다각형의 면적이 변하지 않아야 한다는 것이다. 따라서, 본 논문에서는 이러한 제약 조건들을 만족시키기 위해서 들로니 삼각분할을 바탕으로 탄성변형을 이용한 불일치 보정방법을 제시하였다.

[5]에서는 디지털 지도간의 불일치를 해결하기 위한 탄성변형 방법을 제시하고 있다. 이 방법은 두 지도의 유사성을 고려한 것으로, 단순한 다각형을 대상으로 형태 알고리듬을 기반으로 디자인된 것이다. 그리고, [5]에서 제시하고 있는 형태 알고리듬은 변형을 하는 동안 지도의 위상 속성과 기하적인 정확성을 보존한다. 하지만 [5]에서는 서로 대응하는 다각형의 모서리 수가 같으면 성립이 가능하나, 그 모서리의 수가 서로 다르면 [5]에서 제시하고 있는 방법으로는 객체의 위치가 보존된다는 것을 보장할 수 없게 된다. 따라서, [7]에서는 두 지도의 유사성을 바탕으로 서로 대응되는 다각형의 유사성 비율을 [5]에서 사용한 0.7에서 그 값을 더 감소시켜 탄성변형 방법의 성능을 향상시키고자 하였다. 특히, [7]에서는 기존의 연구들에서 제시되었던 탄성변형 기법과 다른 새로운 탄성변형

기법을 제시함으로써 지도상의 불일치를 보존한 후에도 지리 객체 사이의 위상 관계나 지리적 정확성 등이 보존된다는 것을 강조하고 있다.

[11]에서는 들로니 삼각분할을 이용한 탄성변형 방법을 제시하고 있다. [11]에서 제시한 방법은 수치지도 불일치 보정에 관한 연구로 불일치가 발생하는 기본도와 설비도의 시설물에 대해 들로니 삼각화를 이용한 유사도 비교를 통한 자동보정 방법을 제안하였다. 본 논문에서는 [11]에서 제시하고 있는 들로니 삼각분할을 이용한 탄성변형 방법을 사용한다. 자세한 내용은 다음 장에서 살펴보기로 한다.

위에서 열거한 탄성변형에 의한 방법은 하나의 공간데이터베이스를 다른 공간데이터베이스에 맞게 전체적으로 수정하는 것이다. 그러나, 많은 경우 각각의 공간데이터베이스는 자신의 자율성(Autonomy)을 가지고 운영되기 때문에, 하나의 데이터베이스를 전체적으로 수정하는 것이 불가능하다. 따라서, 각각의 데이터베이스에 독립성을 부여하여 처리 할 수 있는 연합적 분산 공간데이터베이스 시스템 (Federated Spatial Database System)적인 접근방법을 취할 수 밖에 없고, 위에서 제안된 방법들은 이와 같은 환경에서는 사용되기 어렵다.

따라서, 연합적 분산 공간데이터베이스 시스템에서의 질의처리는 다른 접근방법이 필요하다. 예를 들어, 사용자가 하나의 공간데이터베이스의 공간객체의 위치를 참조하여 질의를 주었을 경우, 다른 공간데이터베이스의 보정 없이 질의의 영역이나 위치를 다른 공간데이터베이스에 맞추어 변형하여 처리하는 것이 바람직하다. 즉, 그림 2와 같이, 하나의 도면의 위치를 기준으로 하여 질의를 주었을 때, 이 질의의 영역이나 위치를 다른 데이터베이스에 맞게 변형하여 전달하는 과정이 필요하다.

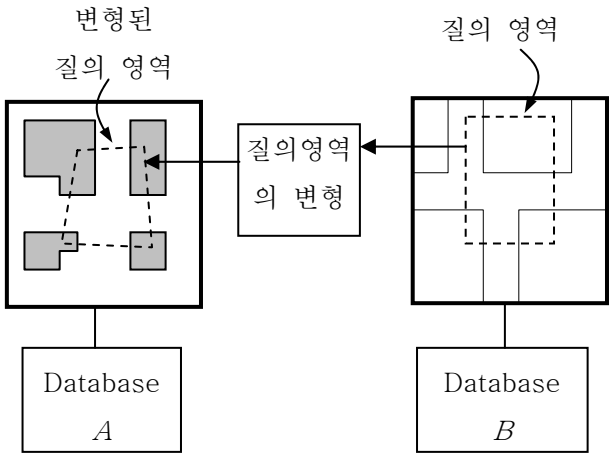


그림2. 공간질의의 변형

이와 같은 방법으로 질의를 처리하면 각각의 공간데이터베이스는 독립적으로 자율권을 확보하게 된다. 따라서, 본 논문에서는 여러 개의 공간데이터베이스가 독립적으로 운영되고, 위치가 서로 불일치하는 경우 공간적 질의를 처리하는 방법을 제안한다.

3. 들로니 삼각분할을 이용한 탄성변형

본 논문에서 제안하는 기본적 방법론은 들로니 삼각분할을 이용한 탄성변형에 기초하고 있다. 따라서, 본 장에서는 들로니 삼각분할을 이용한 탄성변형 방법을 간단하게 소개한다[11]. 여기서 탄성변형이라고 하는 것은 공간객체 사이의 위상관계가 변하지 않는 범위 내에서 이동, 휘어짐, 크기 조절 등으로 공간의 위치를 변형하는 것을 의미한다.

들로니 삼각분할을 이용한 탄성변형의 기본적 개념은 각각의 공간데이터베이스에 주어진 기준점을 이용하여 들로니 삼각분할을 수행한다는

것이다. 즉, 삼각분할을 통하여 얻은 다음, 대응되는 두 삼각형의 변형정도를 고려하여 탄성변형을 한다. 이 과정을 알고리듬으로 나타내면 다음과 같다.

알고리듬 1. 들로니 삼각분할을 이용한 탄성변형

입력 S_A : 변형의 기준이 되는 공간데이터베이스의 꼭지점 집합
S_B : 변형이 일어나는 공간데이터베이스의 꼭지점 집합
$G = \{ (g_A, g_B) \mid g_A$ 와 g_B 는 G_A, G_B 에 각각 주어진 대응되는 기준점 $\}$
P_B : 변형되는 점의 집합
출력 P_A : G_A 에 맞게 위치가 보정된 P_B

알고리듬 시작

$T_A \leftarrow$ DelaunayTriangulation(G_A);
// $G_A = \{g_A \mid g_A$ 는 S_A 의 기준점 $\}$, T_A 는 들로니 삼각분할로 만들어진
// 삼각형의 집합
$T_B \leftarrow$ MappingTriangles(T_A); // T_A 에 대응되는 S_B 의 삼각형 집합
$P_A \leftarrow \{\}$; // P_A 를 공집합으로 초기화
각각의 $p_B \in P_B$ 에 대하여 {
$t_B \leftarrow$ FindContainingTriangle (T_B, p_B); // p_T 를 포함하는 삼각형 탐색
$t_A \leftarrow$ MappingTriangle(t_A); // t_B 에 대응되는 T_A 의 삼각형 탐색
$p_A \leftarrow$ TriangularTransformation (t_B, t_A, p_B);
// t_s 와 t_T 를 이용하여 p_T 의 위치보정
$P_A \leftarrow P_A \cup \{ p_A \}$
}

알고리듬 끝

알고리듬 1 이 수행되는 과정에서 가장 중요한 부분은 삼각형 탄성변형에 해당하는 TriangularTransformation (t_B, t_A, p_B) 이다. 삼각형을 이용한 탄성변형은 다음과 같이 수행된다. 먼저 S_A 상의 점 $p_A(x_A, y_A)$ 에 대하여 이를 포함하는 삼각형을 t_A (p_{A1}, p_{A2}, p_{A3}) 라고 하고, 이에 대응되는 S_B 상의 삼각형을 t_B = (p_{B1}, p_{B2}, p_{B3}) 라고 하자. 이 때, p_B 의 점이 탄성변형된 공간에서의 좌표 p_A (x_A, y_A)는

다음과 같은 과정으로 계산된다. 우선, p_A 는 다음과 같이 표현된다.

$$x_A = \lambda_1 x_{A1} + \lambda_2 x_{A2} + \lambda_3 x_{A3} \quad - (1)$$

$$y_A = \lambda_1 y_{A1} + \lambda_2 y_{A2} + \lambda_3 y_{A3} \quad - (2)$$

여기서 p_A 는 삼각형 내의 점이므로 $\lambda_1 + \lambda_2 + \lambda_3 = 1$ 이며, $0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1$ 이 된다. 여기에서 $\lambda_1, \lambda_2, \lambda_3$ 의 실제적인 의미는 삼각형 $t_A = (p_{A1}, p_{A2}, p_{A3})$이 점 p_A (x_A, y_A)에 의하여 세 개의 작은 삼각형으로 분할될 때, 원래 삼각형에 대한 각 삼각형의 넓이 비율이 된다. 그러면, 변형될 점을 포함하는 삼각형 $t_B = (p_{B1}, p_{B2}, p_{B3})$을 같은 넓이의 비율로 분할하는 변형된 공간상의 대응되는 점 p_B 도 마찬가지로 아래와 같이 계산된다.

$$x_B = \lambda_1 x_{B1} + \lambda_2 x_{B2} + \lambda_3 x_{B3} \quad - (3)$$

$$y_B = \lambda_1 y_{B1} + \lambda_2 y_{B2} + \lambda_3 y_{B3} \quad - (4)$$

여기서 $\lambda_1, \lambda_2, \lambda_3$ 은 식 (3), (4)와 $\lambda_1 + \lambda_2 + \lambda_3 = 1$ 을 이용하면 다음과 같이 계산된다.

$$\lambda_1 = \omega \{ (x_B - 1)(y_{B2} - y_{B3}) - (y_B - 1)(x_{B2} - x_{B3}) \} \quad - (5)$$

$$\lambda_2 = \omega \{ (x_B - 1)(y_{B3} - y_{B1}) - (y_B - 1)(x_{B3} - x_{B1}) \} \quad - (6)$$

$$\lambda_3 = \omega \{ (x_B - 1)(y_{B1} - y_{B2}) - (y_B - 1)(x_{B1} - x_{B2}) \} \quad - (7)$$

여기서 $\omega = 1/(x_{B1} y_{B1} + x_{B2} y_{B2} + x_{B3} y_{B3} - x_{B1} y_{B3} - x_{B2} y_{B1} - x_{B3} y_{B2})$ 이다. 결국 $\lambda_1, \lambda_2, \lambda_3$ 을 식 (1)과 (2)에 대입하면 원하는 p_A (x_A, y_A)의 값을 구할 수 있다.

위의 알고리듬이 수행되는 과정을 간단하게 그림으로 나타내면 그림 3과 같다.

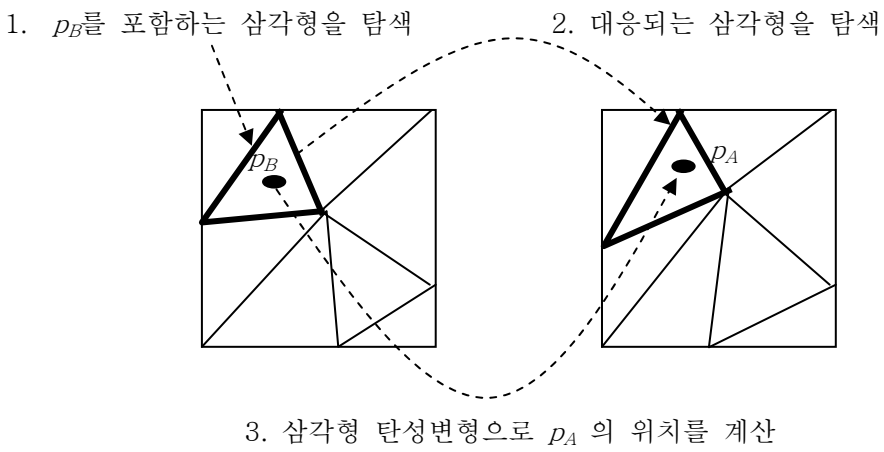


그림 3. 들로니 삼각분할을 이용한 탄성변형 과정

위의 알고리듬은 몇 가지 중요한 특성을 가지고 있다. 우선, 탄성변형이므로 대부분의 위상 관계가 변하지 않는다는 것이다. 예를 들어, 두 개의 공간객체가 포함이라는 위상 관계를 가지고 있을 때, 탄성변형을 한 후에도 두 개의 객체는 계속 포함이라는 위상 관계를 가지고 있게 된다. 이는 탄성변형의 기본적 요건이기도 하지만 실제 응용분야에 적용하기 위하여서는 매우 중요한 특성이다.

두 번째 특성은 정확도에 관한 것이다. 두 가지 공간데이터베이스가 모두 완벽하게 정확하지 않은 상황에서 하나의 공간 데이터베이스를 다른 공간데이터베이스로 전환한다는 것은 경우에 따라 오차를 더욱 유발할 수도 있다. 따라서, 변형이 일어나지 않은 공간데이터베이스와 변형이 일어난 공간데이터베이스 사이의 차이가 얼마인가를 알아내는 것은 매우 중요하다. 이 차이는 다음과 같은 정리로 설명이 가능하다.

정리. 들로니 삼각분할을 이용한 탄성변형의 최대 이동값

변형이 일어나는 공간데이터베이스의 기준점의 집합과 변형의 기준이 되는 공간데이터베이스의 기준점의 집합을 각각 $G_A = \{ g_{A1}, g_{A2}, \cdots, g_{An} \}$, $G_B = \{ g_{B1}, g_{B2}, \cdots, g_{Bn} \}$ 이라고 하면, 변형전의 임의의 점 p_A 와 변형된 점 p_B 는 다음의 조건을 만족한다.

$$0 \le \mathrm{dist}(p_A, p_B) \le \mathrm{Max}(\mathrm{dist}(g_{Ak}, g_{Bk}), 0 \le k \le n)$$

위의 정리에 대한 증명은 본 논문에서 생략한다. 위의 정리가 의미하는 것은 탄성변형이 적용될 때, 대응되는 기준점의 쌍의 최대 거리 이하로 이동이 일어난다는 것을 의미한다.

4. 들로니 삼각분할을 이용한 공간 질의 영역의 탄성변형

본 장에서는 위치가 일치하지 않는 공간데이터베이스 시스템들이 있을 때, 질의 처리하는 방법을 제시한다. 특히, 앞의 방법에서 제시한 탄성변형방법과 달리, 본 논문에서 제시하는 방법의 기본적 개념은 각각의 공간데이터베이스를 수정하지 않고 주어진 질의 영역이나 위치를 보정하여 각 시스템에서 공간질의를 수행하는 것이다. 질의 영역을 보정하는 방법은 앞 절에서 설명한 들로니 삼각분할의 탄성변형을 이용한다. 특히, 질의영역 탄성변형 방법을 최근접 질의 처리방법과 포함질의 처리방법에 적용하는 것을 제시한다. 다른 공간질의의 처리 방법도 이 두 가지 처리방법과 유사하므로 본 논문에서는 자세한 언급은 생략한다.

위치가 일치하지 않는 공간데이터베이스에게 공간질의를 요청할 때는 항상

질의위치의 기준이 되는 공간데이터베이스가 존재한다. 예를 들어, 시설도와 교통도를 나타내는 공간데이터베이스 시스템이 각각 있을 경우, 사용자는 하나의 공간데이터베이스를 선택하여 질의를 요청한다. 사용자는 시설도 위의 하나의 점을 명시하고, 이에 가장 가까운 시설도와 교통도의 객체를 찾으라는 질의를 요청한다. 이 경우, 시설도가 기준이 되는 공간데이터베이스가 된다. 결국, 공간질의의 탄성변형은 기준이 되는 공간데이터베이스에 주어진 공간질의의 영역이나 위치를 다른 공간데이터베이스상의 위치로 변환하는 과정을 의미하게 된다. 여기서 기준이 되는 공간데이터베이스로부터 다른 공간데이터베이스로 변환이 일어나는 방향이 앞 절에서 서술한 공간데이터베이스 전체를 변형하는 방법의 방향과 반대로 된다는 것에 주목할 필요가 있다. 이 과정을 간단하게 그림으로 나타내면 그림 4와 같다.

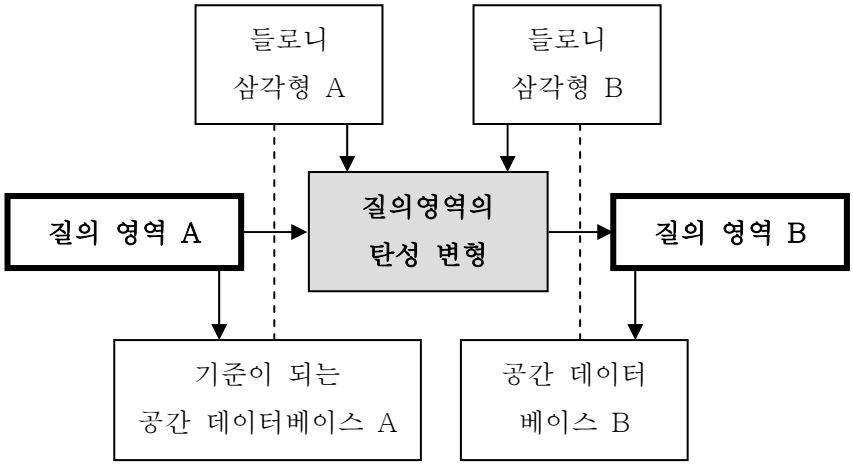


그림 4. 공간 질의 영역의 탄성 변형과정

그러면 공간질의영역의 탄성변형을 이용하여 최근접 질의와 영역질의를 처리하는 방법을 살펴보기로 한다.

4.1 들로니 삼각분할에 따른 탄성변형을 이용한 최근접 질의의 처리 방법

최근접 질의를 위한 질의영역은 하나의 점으로 주어진다. 따라서, 질의영역의 탄성변형은 질의가 주어지는 공간데이터베이스의 질의 위치를 다른 공간데이터베이스의 질의 위치로 변환하여 질의를 처리하게 된다. 이 과정은 알고리듬으로 간단하게 나타내면 다음과 같다.

알고리듬 2 에서 RubberSheeting(t_A, t_B, p_A) 의 함수는 앞 절에서 설명한 삼각형 탄성변형에 의한 보정 방법을 이용한다. 또한 각 공간데이터베이스에 대한 최근접 질의처리 PerformQuery(Nearest Neighbour, S_A, p_A) 는 기존의 여러 가지 방법[12][13][14] 중 한가지를 선택하여 처리할 수 있다.

알고리듬 2. 들로니 탄성변형을 이용한 최근접 질의처리

입력 S_A : 질의가 주어진 기준이 되는 공간데이터베이스
S_B : 다른 공간데이터베이스
$G = \{ (g_A, g_B) \mid g_A$ 와 g_B 는 S_A, S_B 에 각각 주어진 대응되는 기준점 $\}$
p_A : 데이터베이스 S_A 에 주어진 질의의 위치
출력 R_A , R_B : S_A, S_B 에 대하여 처리한 질의 결과

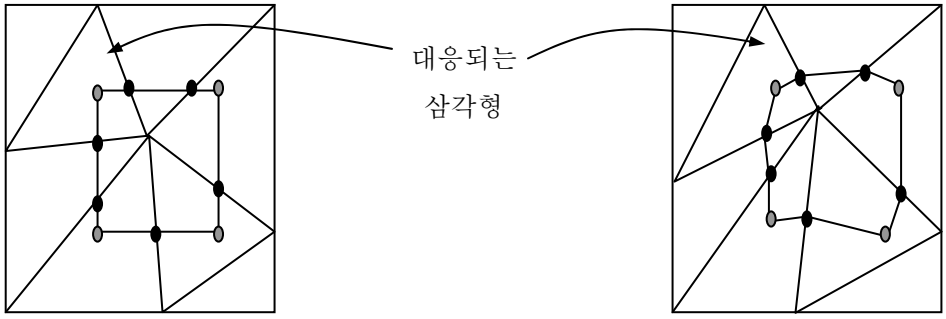
알고리듬 시작

$T_A \leftarrow$ DelaunayTriangulation(G_A); // $G_S = \{g_s \mid g_s$ 는 S_S의 기준점 $\}$
$T_B \leftarrow$ MappingTriangles(T_A); // T_S 에 대응되는 M_T 의 삼각형 집합
$t_A \leftarrow$ FindTriangleContaining(T_A, p_A); // p_S 를 포함하는 삼각형 탐색
$t_B \leftarrow$ MappingTriangle(t_A); // t_A 에 대응되는 T_B 의 삼각형 탐색
$p_B \leftarrow$ RubberSheeting(t_A, t_B, p_A); // t_A 와 t_B 를 이용하여 p_B 의 위치보정
$R_A \leftarrow$ PerformQuery("Nearest Neighbour", S_A, p_A);
$R_B \leftarrow$ PerformQuery("Nearest Neighbour", S_B, p_B);

알고리듬 끝

4.2 들로니 삼각분할에 따른 탄성변형을 이용한 포함질의 처리

질의영역의 탄성변형을 이용하여 포함질의를 처리하는 과정은 최근접 질의를 처리하는 것보다 약간 더 복잡하다. 그 이유는 최근접 질의의 경우, 질의 위치는 하나의 점으로 주어지지만, 포함질의는 질의영역이 다각형으로 주어지기 때문이다. 다각형을 탄성변형하는 과정은 점을 탄성변형하여 위치를 찾는 과정보다 복잡하다. 들로니 삼각분할을 이용하여 다각형을 탄성변형하는 과정은 그림 5에서 설명된다.



a. 질의가 주어진 공간 데이터베이스의 질의 영역 b. 탄성변형에 의하여 변형되는 질의영역

그림 5. 포함질의의 처리과정

그림 5에서 검은 점으로 표현된 것은 원래 질의영역을 나타내는 다각형의 꼭지점이 된다. 그러나, 그림에 나타난 것과 같이 질의영역이 들로니 삼각분할 방법으로 탄성변형될 때 고려되는 점은 단순히 원래의 꼭지점뿐만 아니라, 질의영역의 변과 들로니 삼각형의 변이 교차되는 점들도 포함한다. 그림 6에서 회색 점으로 나타나 있는 점들을 고려한 질의영역이다.

따라서 탄성변형을 이용한 영역 질의처리 알고리듬은 아래와 같다. 알고리듬 3 은 앞에서 설명한 알고리듬 1을 사용하며, 알고리듬 2와 유사한 형태로

수행된다. 그리고, 알고리듬 3에서 사용하는 포함질의 PerformQuery(Contain , S_A, a_A)는 일반적인 포함 공간연산을 이용하여 수행된다. 공간 색인을 이용하면 더욱 좋은 성능을 기대할 수 있다. 공간 색인 방법에는 여러 가지 방법이 제시되어 있다[15][16][17]. 본 논문에서는 [16]을 토대로 보다 효과적인 클러스터링을 지원하기 위해 [15]에서 제안된 공간 색인 방법을 사용하여 공간 연산을 수행하도록 하였다.

알고리듬 3. 들로니 탄성변형을 이용한 포함질의 처리

입력 S_A : 질의가 주어진 기준이 되는 공간데이터베이스
S_B : 다른 공간데이터베이스
$G = \{ (g_A, g_B) \mid g_A$ 와 g_B 는 S_A, S_B 에 각각 주어진 대응되는 기준점 $\}$
$a_A = (p_{A1}, p_{A2}, p_{A3}, \cdots, p_{An})$: 데이터베이스 S_S 에 주어진 질의의 영역
출력 R_A , R_B : S_A, S_B 에 대하여 처리한 질의 결과

알고리듬 시작
$T_A \leftarrow$ DelaunayTriangulation(G_A); // $G_A = \{g_A \mid g_A$ 는 S_A 의 기준점 $\}$
$T_B \leftarrow$ MappingTriangles(T_B); // T_A 에 대응되는 S_B 의 삼각형 집합
$b_A \leftarrow$ MakeNewPolygonWithIntersectingPoints(a_A , T_A);
// a_A 와 삼각형의 변과 교차하는 점으로 질의영역 생성
각 b_A 의 꼭지점 $q_{Ak} \in b_A$ 에 대하여 {
$t_A \leftarrow$ FindTriangleContaining(T_A, q_{Ak}); // q_{Sk} 를 포함하는 삼각형 탐색
$t_B \leftarrow$ MappingTriangle(t_A); // t_S 에 대응되는 T_T 의 삼각형 탐색
$q_{Bk} \leftarrow$ ElasticTransformation(t_A, t_B, q_{Ak});
// t_A 와 t_B 를 이용하여 p_B 의 위치보정
}
$a_B \leftarrow (q_{B1}, q_{B2}, q_{B3}, \cdots, q_{Bm})$;
$R_A \leftarrow$ PerformQuery("Contain", S_A, a_A);
$R_B \leftarrow$ PerformQuery("Contain", S_B, a_B);
알고리듬 끝

위의 두 질의처리 방법의 성능은 각각의 공간데이터베이스 시스템의 질의처리 성능에 따라 결정된다. 물론, 질의영역을 변형하는 시간이 추가되지만 이는 주로 계산 시간이므로 각 데이터베이스의 질의처리 시간에 비하여 무시할 만큼 작은 시간이다. 따라서, 본 논문에서는 성능적인 측면은 고려하지 않고, 원래의 포함질의와 변형된 포함질의가 각각 처리된 결과가 동일하면 공간질의 보정의 정확성을 입증한 것으로 만족한다.

5. 정확성

본 장에서는 4장에서 제시한 알고리듬의 정확성을 알아보기로 한다. 제시된 알고리듬의 정확성을 증명하기 위해서는 주어진 질의의 정확한 답을 알아야 한다. 즉, 주어진 질의의 정확한 답을 알면 그것과 본 논문에서 제시하는 질의처리방법의 결과와 비교하여 정확성을 증명할 수 있다. 그러나 모든 공간데이터베이스에서 위치 데이터는 오차를 포함하고 있고, 더욱이 불일치 하는 공간 데이터베이스가 있을 경우, 어느 것이 정확한지 알 수 없으므로, 주어진 질의의 정확한 답을 알 수 없다. 그러므로 본 논문에서는 알고리듬의 정확성을 증명하는 대신에 3장에서 설명한 개별 공간데이터베이스의 변형에 의한 기존의 방법과 본 논문에서 제시하는 질의영역의 변형에 의한 질의 결과가 동일하다는 것을 증명한다. 서로 다른 사이트에 있는 공간데이터베이스에서 하나의 공간데이터베이스는 건물을 나타내는 지형도이고, 다른 공간데이터베이스에는 교통도가 있다고 가정하자. 만일 각각의 공간데이터베이스에서 두 개의 도면이 서로 불일치 한다면, 사용자가 두 개의 도면에 공간질의를 주면 질의처리 결과는 서로 일관성을 보장할 수 없다. 해결 방법은 어느 한쪽으로 공간데이터베이스를 변환하는 것이다. 만약 어느 한쪽으로 공간데이터베이스를 변환한다면, 각

사이트의 자율성이 중시되어야 하는 경우에는 실용적이지 않으며, 공간데이터베이스에 있는 모든 데이터를 변환해야 하므로 그에 따른 비용이 매우 크다. 따라서 본 논문에서는 각 사이트의 공간데이터베이스를 변환하지 않고 공간질의를 변환하여 처리하였다. 공간질의를 변환함으로써 각 사이트의 공간데이터베이스는 자율성을 보장할 수 있으며, 공간데이터베이스를 변환하는 것보다 훨씬 적은 비용이 따른다. 따라서, 공간데이터베이스를 변환하는 것과 공간질의를 변환하는 것과는 서로 다르다는 것을 알 수 있다.

우선 3장에서 기존의 방법과 본 논문에서 제시하는 방법과의 동등 관계를 직관적으로 설명하여 보기로 한다. 먼저, 삼각형 단위로 이루어지는 각각의 탄성변형은 위상적인 특성을 보존한다. 이에 따라, 삼각형 변형의 조합으로 이루어진 전체적인 탄성변형도 위상적인 특성을 보존하게 된다. 즉, 삼각형 단위로 이루어지는 탄성변형을 적용할 경우 전체적인 위상관계도 유지된다. 이것은 탄성변형이 이루어지기 전에 주어진 질의영역에 포함되어 있었던 객체는 탄성변형 후에도 포함된다는 것을 의미한다. 아래의 그림 6과 같이 두개의 공간데이터베이스 S_A와 S_B가 있을 때, S_A에 질의 Q_A 가 주어졌다고 가정하자.

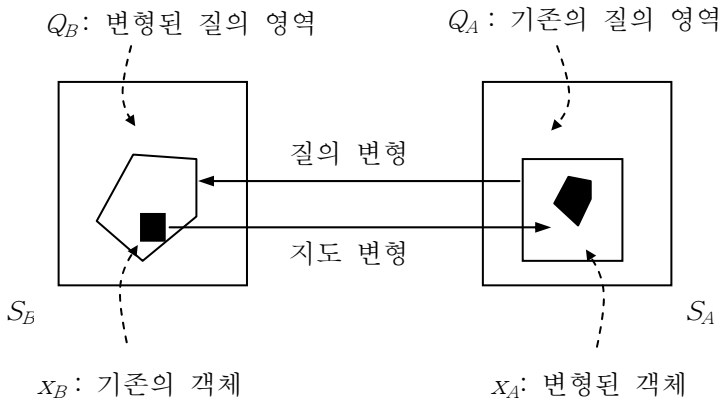


그림 6. 지도 변형과 질의 변형 사이에서의 동등 관계

이 때, 지도 변형에 따라 변형된 공간 객체 x_A 는 Q_A 에 포함된다. 그리고, 공간 객체 x_B 역시 질의 변형에 따라 Q_B 에 포함되어진다는 것을 보여준다. 다시 말해서, 그림 6은 변형 후 위상이 변하지 않기 때문에 Q도 여전히 x를 포함하고 있다. 이것은 포함 위상에 대해서 지도 변형 방법과 질의 변형 방법에 의한 질의처리 결과가 동일하다는 것을 의미한다. 포함 질의에 대한 상세한 증명은 다음과 같은 정리들로 설명되어진다.

보조정리 1

아래의 그림 7과 같이 점 p 가 삼각형 T (p_1, p_2, p_3) 내에 있을 때, 이 점으로부터 변형된 점 p' 역시 변형된 삼각형 T'(p'_1, p'_2, p'_3) 내에 있다.

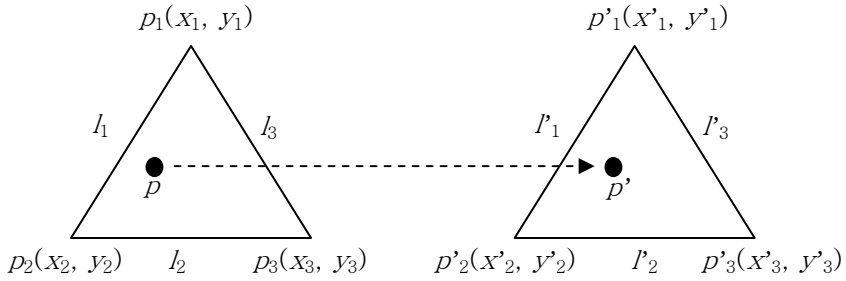


그림 7. 삼각형의 탄성변형 전과 후의 위상관계1

증명

방정식 (1)과 (2)에 의하면, 변형된 위치 p'는 아래의 식에 위해 계산되어진다.

$$p' = \lambda_1 p'_1 + \lambda_2 p'_2 + \lambda_3 p'_3 \quad (\text{여기서 } \lambda_1 + \lambda_2 + \lambda_3 = 1)$$

이 때 $0 \le \lambda_1, \lambda_2, \lambda_3 \le 1$ 이므로, 점 p'가 삼각형 T '(p'_1, p'_2, p'_3) 내에 있다는 것을 알 수 있다.

보조정리 2

만약 점 p가 삼각형 T 를 통과하는 다각선 pl (v_1, v_2, …, v_k)의 오른쪽에 있다면, 변형된 점 p' 역시 변형된 다각선 pl'의 오른쪽에 존재한다.

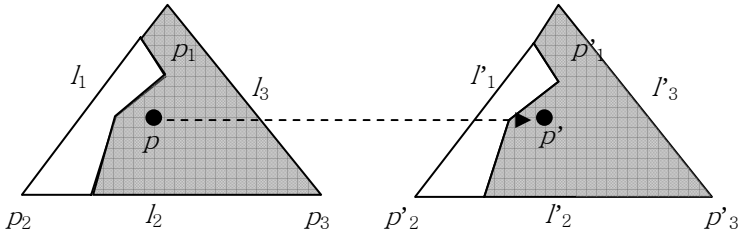


그림 8. 삼각형의 탄성변형 전과 후의 위상관계2

증명

pl 의 하나의 선분을 l_i (v_i, v_{i+1}) 이라 하고, l'_i 를 그것의 변형된 선분이라고 하자. 이 때 [보조정리 1]에 의해, l_i 가 T 내에 있다면 l'_i 도 T 로부터 변형된 삼각형 T' 내에 있다는 것을 알 수 있다. 그리고 v_i 와 v_{i+1} 을 각각 (x_a, y_a)와 (x_a, y_b)라고 하자. p (x, y)가 l_i 의 아래쪽에 있다고 가정할 때 다음과 같이 표현된다.

$$(x - x_a)(y_b - y_a) - (y - y_a)(x - x_a) > 0 \quad -(8)$$

l'_i 의 끝점들을 (x'_a, y'_a)와 (x'_b, y'_b)라고 하고 변형된 점 p'를 (x', y')라고 하자. 이 때 3장의 방정식 (3) – (7)에 의해 다음과 같이 변형된 선분과 점에 대한 부등식 (8)의 왼쪽 부분과 상응하는 수식을 이끌어 낼 수 있다.

$$(x'-x'_a)(y'_b-y'_a)-(y'-y'_a)(x'_b-x'_a)$$

$$=4\{(x-x_a)(y_b-y_a)-(y-y_a)(x_b-x_a)\}A(T)A(T')$$

여기서 $A(T)$는 삼각형 T 의 면적이다.

부등식 (8)에 의하여 $A(T)A(T') > 0$ 과 $\{(x-x_a)(y_b-y_a)-(y-y_a)(x_b-x_a)\} > 0$ 이라는 것을 알 수 있다. 따라서 다음과 같은 결론을 내릴 수 있다.

$$(x'-x'_a)(y'_b-y'_a)-(y'-y'_a)(x'_b-x'_a) > 0$$

이것은 p'가 l'_i 의 아래쪽에 있다는 것을 의미한다. 다시 말해서 p' 는 다각선 pl 에 해당하는 변형된 모든 선분의 똑 같은 쪽에 존재한다는 것이다. 결과적으로, p'가 pl 의 똑 같은 쪽에 존재한다.

정리

만약 다각형 m 이 질의영역 R 에 의해 포함되어진다면, 이 때 변형된 다각형 m' 역시 변형된 질의영역 R' 에 의해 포함되어진다.

증명

다각형 m 과 질의영역 R 이 그림에서와 같이 삼각형의 집합에 의해 나누어진다.

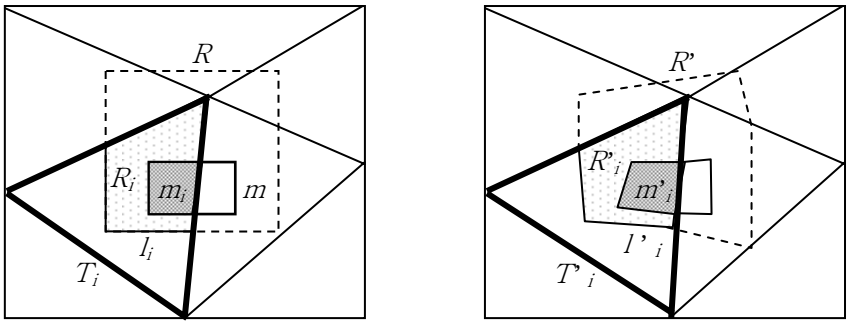


그림 9. 포함 질의의 탄성변형 전과 후

삼각형 T_i 내에 있는 m 의 어떤 조각 m_i ($=m \cap T_i$)가 R_i 에 포함되어지는 것에 대하여 변형된 조각 m_i 역시 변형된 조각 R_i 에 의해 포함되어진다는 것을 그림으로부터 설명되어진다. 만약 m_i 에 있는 점이 짧게 잘린 질의경계선 l_i 의 오른쪽 또는 왼쪽에 있다면, 그림에서와 같이 이 점 역시 삼각형 T'_i 에 있고 l'_i 의 똑 같은 쪽에 있다는 것을 보조정리 1과 보조정리 2에 의해 알 수 있다. 이것은 m'_i 에 있는 모든 점이 R'_i 에 의해 포함되어진다는 것을 의미한다. 따라서 m'_i 는 R'_i 에 포함되어진다.

6. 실험

본 몬문에서 제시한 방법의 유용성 검증을 위하여 5장에서 설명한 정확성을 바탕으로 실험을 하였다. 실험 환경은 펜티엄 4-2.4G와 512 메모리를 가진 윈도우 2000 환경이었으며, 실험은 Microsoft Visual C++을 사용하여 구현되었다. 특히, 공간데이터베이스를 주로 사용하며, 클라이언트/서버 구조의 다수 사용자 환경에 적합한 상용 DBMS인 ZEUS를 이용하여 구현하였다. 즉, 서로 다른 공간데이터베이스에 지형도와 지적도를 구축하여 공간질의를 처리하는

실험을 ZEUS 기반하에서 구현하였다. 물론, 공간질의에 대한 처리 결과는 변형 전의 공간질의와 변형 후의 공간질의가 동일하게 나타났다. 이 실험은 상용 데이터베이스 시스템을 사용함으로써 실질적으로도 사용이 가능하다는 것을 말해주고 있다. 다음 그림은 본 논문에서 사용한 실험 데이터로 지형도와 지적도의 일부분을 보여주고 있다.

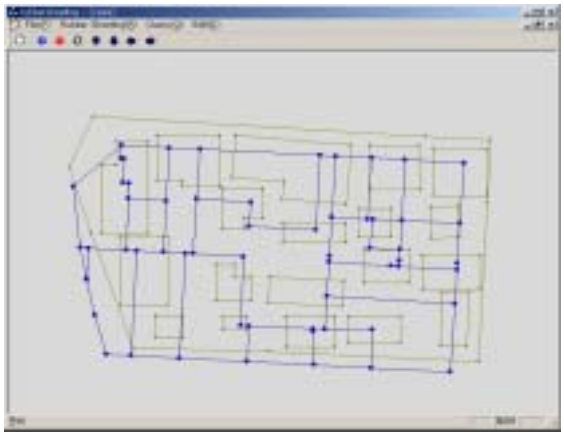


그림 10 지형도와 지적도의 차이

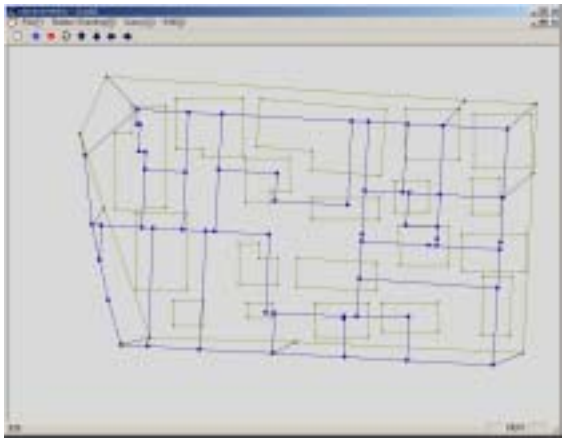


그림 11. 지형도의 변형 과정

그림 10은 동일한 지역임에도 불구하고 지형도와 지적도가 불일치가 일어나고 있음을 보여주고 있다. 그리고, 그림 11은 지적도를 지형도로 보정하는 과정을 보여주고 있다. 즉, 지적도의 경계에 있는 점들을 지형도의 경계에 있는 점들로 일치시키는 것이다.

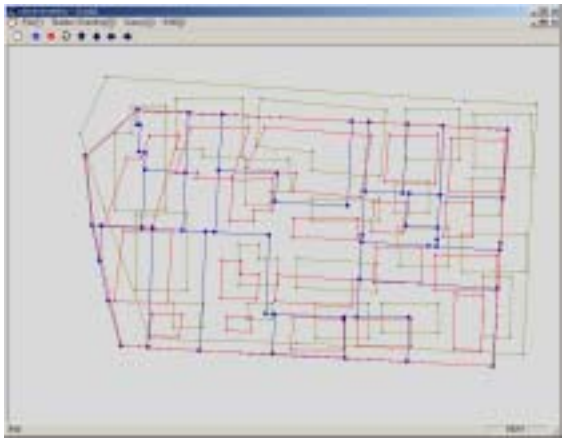


그림 12. 변형된 지형도의 모습

그림 12는 지적도를 지형도로 보정한 결과로 나타난 지형도를 빨간색으로 표현되고 있다. 이와 같이 본 논문에서는 지형도나 지적도 전체를 변형시키는 과정을 생략하고, 공간데이터베이스에 주어지는 공간질의만을 변형시켜 그 결과를 정확하게 얻고자 하는 것이다. 그림 13은 지적도와 지형도의 두 개의 공간데이터베이스에 근접질의를 처리한 결과이다.

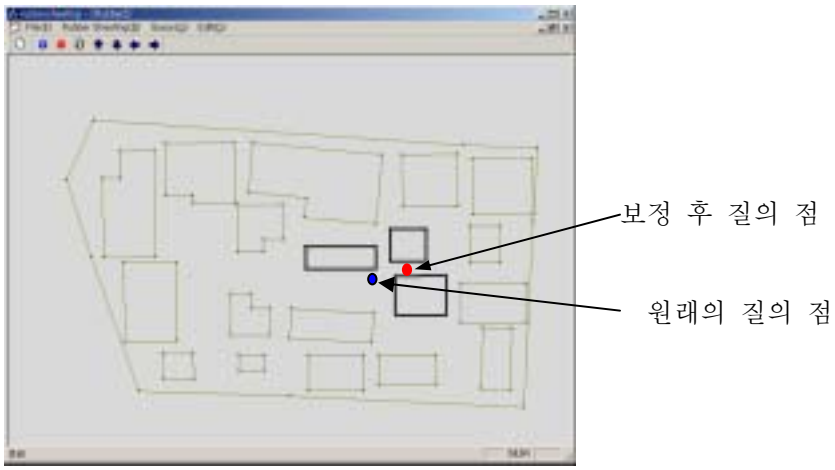


그림 13. k-근접 질의처리 결과 (k = 3)

이 실험은 수치상으로 표현이 불가능며, 수행시간은 큰 의미가 없다. 따라서, 원래의 공간질의와 변형된 공간질의가 각각 처리된 결과가 동일하면 공간질의 보정의 정확성을 입증한 것이다. 즉, 공간질의를 무작위로 주었을 때, 원래의 공간질의 결과와 변형된 공간질의의 결과가 모두 동일하게 처리된 것을 보여주고 있다. 그림 14는 영역질의를 지적도와 지형도의 공간데이터베이스에 대하여 실행한 결과의 예이다. 점선은 주어진 포함 질의영역이 탄성변형 후의 영역을 나타내고 있다. 굵은 색은 네 개의 객체가 포함되고 있음을 나타내고 있다.

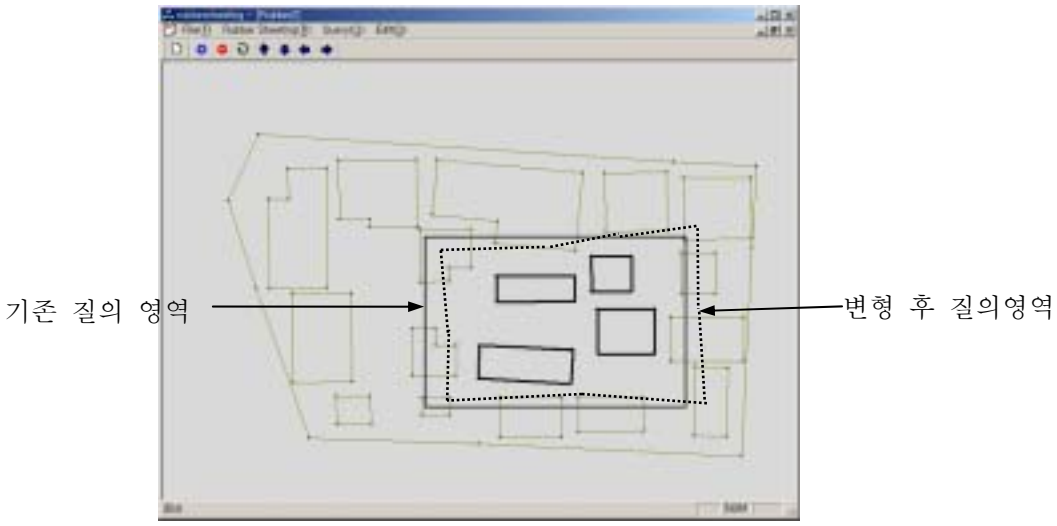


그림 14. 포함 질의처리 결과

본 실험에서는 그림 13과 14에서 실행한 방법으로, 그림 15의 데이터를 상용 DBMS에 적용하여 공간질의의 변형 전과 변형 후를 비교하는 실험을 하였다. 그림 15는 부산의 한 지역을 대상으로 한 데이터이다.

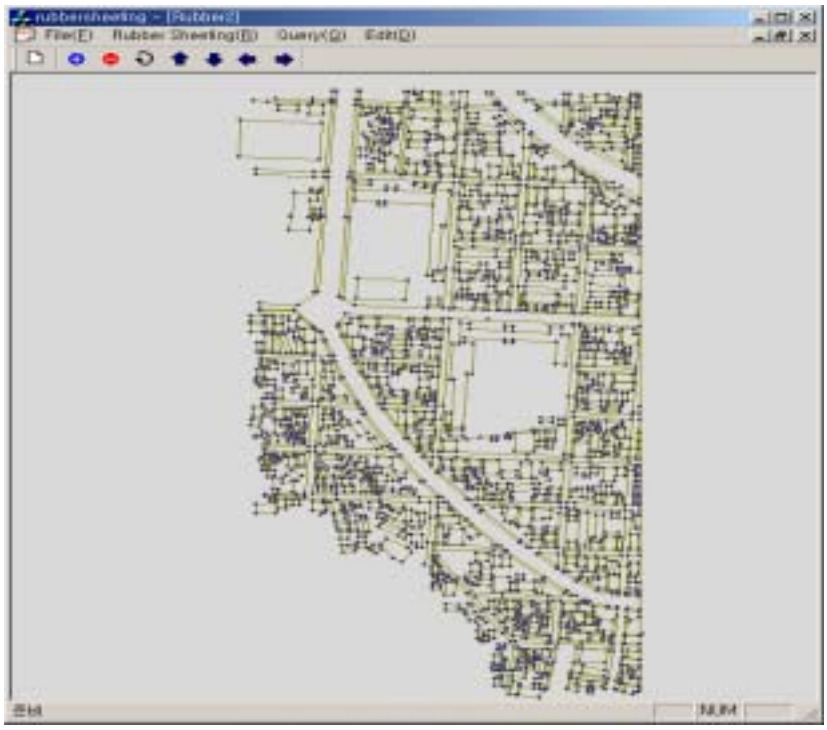


그림 15. 상용 DBMS에 적용한 실험 데이터

이 실험은 점 질의와 영역 질의를 그림 15에 각각 1000개씩 주어 공간질의의 변형 전과 변형 후를 비교한 실험이다. 여기서, 영역 질의는 질의의 크기에 상관없이 적용되었다.

표 1. 상용 DBMS에 적용한 결과

	변형 전 질의	변형 후 질의	일치율
점 질의	1000	999	99.9%
영역 질의	1000	998	99.8%

표 1은 위에서 설명한 상용 DBMS에 적용하여 나타난 실험의 결과이다. 표에서 알 수 있듯이 공간질의의 변형 전과 변형 후의 결과가 100%에 가깝도록 동일하게 나타났다. 따라서, 본 논문에서 제시한 방법이 실질적으로도 충분히 사용이 가능하다는 것을 입증해 준다.

7. 들로니 삼각분할의 탄성 변형을 이용한 공간질의처리 시스템의 구조

본 논문에서 제안하는 방법을 일반화시켜 이용하기 위해서는 본 논문의 방법에 필요한 들로니 삼각형에 관한 정보를 관리해야 한다. 따라서, 본 장에서는 공간질의 처리 기능을 구현하기 위하여 필요한 들로니 삼각형에 관한 정보를 관리할 수 있는 공간질의처리 시스템의 구조에 대해 살펴본다.

본 논문에서 제시하는 질의처리 방법의 가장 중요한 개념은 들로니 삼각분할을 이용한 질의영역의 변형이다. 따라서, 질의 영역을 변형하기 위해서는 기준점의 쌍의 집합이 유지되어야 하며, 그러기 위해서는 기준점 테이블이라는 것이 필요하다. 그림 16은 각 사이트가 기준점 테이블을 유지하고 있고, 이러한 테이블은 질의를 변형시키기 위해 사용되고 있음을 보여주고 있다.

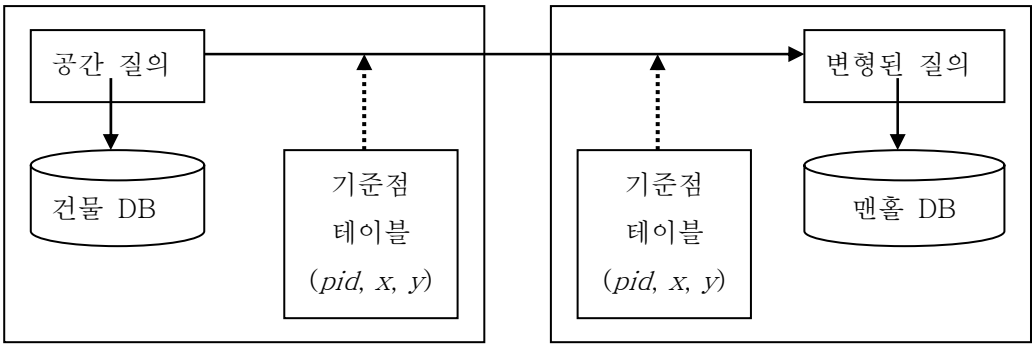


그림 16. 기준점 테이블

그리고, 이것은 오직 분산 데이터베이스에서만 가능하다는 것을 전제로 하고 있다.

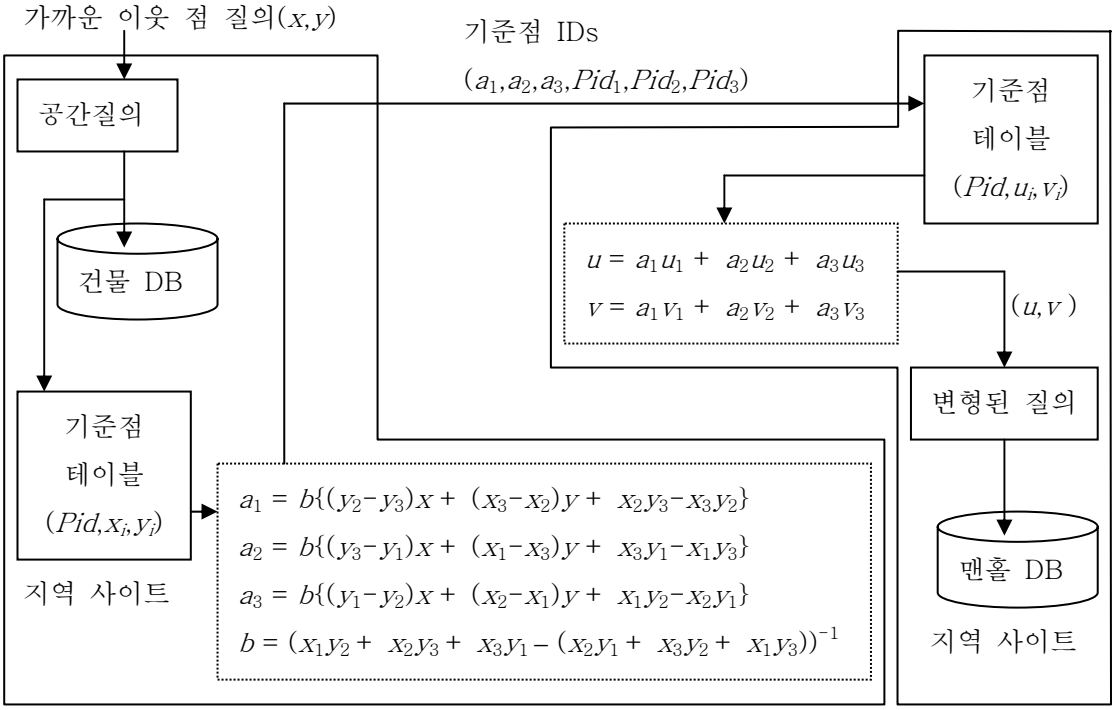


그림 17. 질의변형 과정

그림 17 은 그림 16 를 바탕으로 하여 각 사이트에 분포해 있는 기준점 테이블을 가지고 상세하게 질의변형의 과정을 보여주고 있다. 질의 영역의 좌표가 주어졌을 때, 그림 17 에서는 변형에 사용되어지는 계수 a_1, a_2, a_3 를 계산한다. 그리고, 이러한 삼각형 ID 를 가진 계수는 다른 사이트에 전달되어진다. 그 때 기준점 테이블로부터 새로운 좌표를 계산할 수 있고, 변형된 질의 역시 얻을 수 있다. 변형된 질의는 계수가 전달되어진 이 사이트에서 실행되어지게 된다.

그림 18 은 기준점 테이블을 바탕으로 하여 이전에 존재하는 기준점으로부터 새로 운 기준점의 집합을 만드는 과정을 나타낸 것이다.

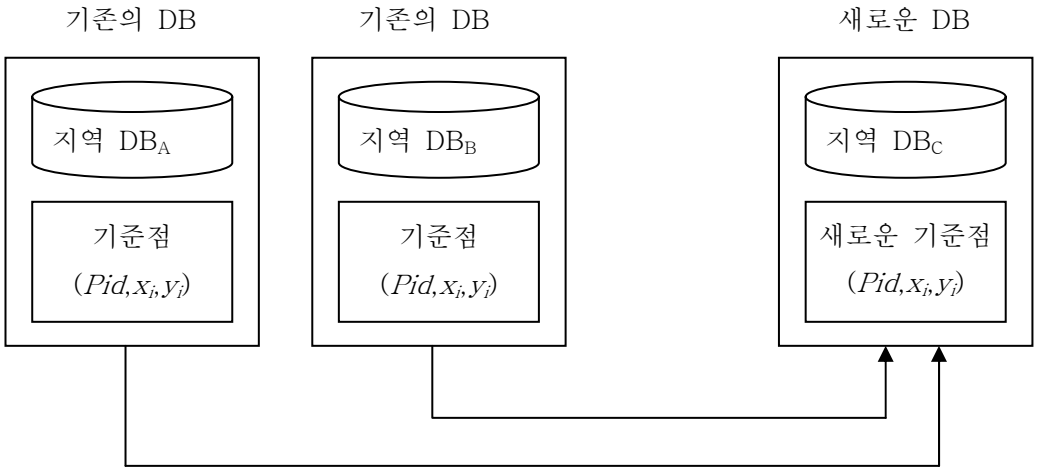


그림 18. 기준점 테이블의 관리

기준점을 관리하기 위해서는 각 사이트의 *pid*, *x*, 그리고 *y* 좌표를 가진 일관된 기준점 테이블을 유지하여야 한다는 것을 그림 18 에서 보여주고 있다. 그리고, 새로운 사이트가 생성되어질 때 다른 사이트들의 이전에 존재하는 기준점으로부터 새로 운 기준점의 집합을 생성할 수 있다는 것을 말해 주고 있다.

위의 설명들을 정리하면, 본 논문에서 제시하는 공간질의처리 시스템의 구조는 서로 다른 각 지역 사이트에 데이터베이스와 공간질의를 변형시키기 위한 기준점 테이블을 가지고 있다. 어떤 지역 사이트에 공간질의를 처리할 경우 공간질의는 기 준점 테이블에 의해 공간질의가 변형이 된다. 그리고, 변형된 질의는 다른 사이트에 있는 데이터베이스로 적용이 되어 공간질의가 실행되어진다.

8. 결론

분산 지리정보시스템을 구현하는데 가장 어려운 점 중의 하나는 공간데이터베이스의 불일치이다. 특히, 여러 데이터베이스 사이의 위치 정확도는 여러 가지 이유로 불일치 하여, 정확하지 않는 질의 결과를 만들게 된다. 불일치 되는 공간데이터베이스의 문제를 해결하기 위한 가장 간단한 방법은 불일치가 일어나는 공간적 데이터를 보정하여 일관성 있게 만드는 것이다. 그러나, 이 방법은 각각의 공간데이터베이스의 자율성을 해친다는 점에서 제한적으로만 적용될 수 있다.

본 논문에서는 불일치 되는 공간데이터베이스 분산환경에서 주어진 질의를 올바르게 처리할 수 있도록 하는 방법을 제안하였다. 이 방법의 특징은 각각의 공간데이터베이스의 데이터를 바꾸지 않고 자율권을 보장하면서, 주어진 질의를 해당 데이터베이스에 맞게 변환하는 방식을 취하고 있다. 즉, 질의영역을 들로니 삼각분할에 의한 탄성변형 방법을 이용하여 각각의 데이터베이스에 맞게 변형하는 방법을 제안하였다.

본 논문에서 제안하는 방법을 일반화시켜 이용하기 위해서는 본 논문의 방법에 필요한 들로니 삼각형에 관한 정보를 관리하는 방안을 만들어야 한다. 특히, 들로니 삼각형의 정보를 미들웨어가 관리하기 위한 여러 가지 표준이나 프로토콜등이 정의되어야 한다. 이 작업은 앞으로 연구되어야 할 분야이기도 하다.

9. 참고문헌

[1] ISO/TC211/WG2 Report N298 ISO WD15046-20, *Geographic information-Spatial Operators*, 1996.

[2] ISO/TC211/WG2 Report N298, *Geospatial Data Models and Operators*, 1996.

[3] Open GIS Consortium, *The OpenGIS Abstract Specification*, version 4, 1999.

[4] R. Laurini and D. Thompson, *Fundamentals of Spatial Information Systems*, 2nd edition, Academic Press, 1992.

[5] M. G. Cho, K. J. Li and H. G. Cho, "A Rubber Sheeting Method With Polygon Morpging", SDH, pp. 7A 31-42, 1996.

[6] S. Servigne, R. Laurini, "Updating Geographic Databases Using Multi-Source Information", ACM-GIS, 1996.

[7] M. G. Cho and H. G. Cho, "Resolving Mismatches and Measure Functions for Evaluating its Validity", SDH, 2000.

[8] P. Langlois, *Une transformation élastique du plan basée sur un modèle d'interaction spatiale, Applications à la géomatique*. Technical Paper in French, MTG, University of Rouen, 1994.

[9] P. Dufour, *Les bases de données géographiques fédérées : continuitiés géométriques et topologiques*, Mémoire de DEA in French, INSA de Lyon, June 1995.

[10] Kurt Mehlhorn, Stefan Naher, Michael Seel, Christian Uhrig, The LEDA user Manual, Version 4.2.

[11] 박상미, 정규상, 손은정, 이기준, “지리 정보 시스템용 수치 지도 자동 보정 기법”, 한국정보과학회 97 가을 학술발표논문집, 제 24권 2호, 1997.

[12] K. Cheung, and A. Fu, “Enhanced Nearest Neighbor Search on the R-tree”, Proc. ACM SIGMOD Record 27(3), pp.16-21, 1998.

[13] Roussopoulos N., Kelley S., Vincent F, “Nearest Neighbor Queries”, Proc. ACM SIGMOD, pp.71-79, 1995.

[14] 장인성, 이기준, “밀도를 이용한 k-최근접 탐색 방법”, 한국정보학회 2000 가을 학술발표논문집, 제 27권 2호, 2000.

[15] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger, “The R*-Tree, An Efficient and Robust Access Method for Points and Rectangles”, Proc. ACM SIGMOD, pp.322-331, 1990.

[16] A. Guttman, “R-Tree, A dynamic Index Structure for Spatial Searching”, Proc. ACM SIGMOD, pp.47-57, 1984.

[17] J. Nievergelt, Hinterberger and K.C.Sevick, “The Grid File: An Adaptable, Symmetric Multikeys File Structure”, ACM Trans. Database Systems, 9(1), pp.38-71, 1984.

[18] J. ORourke, *Computational Geometry in C*, Camgridge University Press, 1994, 1995.