# 연합 공간데이터베이스 시스템에서의 공간질의처리

황 정래, 오 지현, 이 기준
부산대학교 정보공학연구실
{jrhwang, jhoh}@quantos.cs.pusan.ac.kr, lik@hyowon.cc.pusan.ac.kr

# A Spatial Query Processing Method in Federated Spatial Database Systems

Jung-Rae Hwang, Ji-Hyeon Oh, and Ki-Joune Li
Information Systems and Engineering Lab. Pusan National University

**요 약**

분산 지리정보시스템을 구현하는데 가장 어려운 점 중의 하나는 공간데이터베이스의 불일치이다. 특히, 여러 데이터베이스 사이의 위치 정확도는 여러 가지 이유로 불일치하고 이에 따라 공간적 질의의 결과도 부정확하게 되는 문제점을 발생시킨다. 본 논문에서는 불일치하는 여러 개의 공간데이터베이스에 대하여 공간질의를 처리하는 방법을 제시한다. 본 논문에서 제시하는 방법은 불일치하는 각 사이트의 공간데이터베이스를 수정하지 않고, 공간질의를 처리할 수 있도록 한다. 본 논문에서 제안하는 방법은 탄성변형방법을 이용하면서, 각 질의의 위치를 동적으로 변환하는 방법에 기초하고 있다.

## 1. 서론

지리정보시스템의 이용이 활성화됨에 따라, 여러 분산 지리정보시스템에 대한 요구도 늘어나고 있다. 그러나, 분산 지리정보시스템이 본격적으로 사용될 수 있기 위해서는 몇 가지 문제가 해결되어야 한다. 그 중에서 중요한 것은 이질성의 극복이다. 공간데이터베이스의 이질성은 공간데이터베이스 관리시스템의 이질성에서부터, 공간데이터모델의 이질성까지 다양하다. 본 논문에서 주목하는 이질성은 공간데이터베이스의 위치에 대한 불일치에 관한 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 있을 때, 만일 두 개의 도면이 서로 독립적으로 구축되었다면, 서로 불일치되는 데이터가 발생된다. 만일 사용자가 두 개의 도면에 동시에 같은 공간질의를 주면, 그림1과 같이 처리가 된다. 그런데, 두 개의 도면은 서로 불일치되므로 같은 질의로 처리되는 것이 불가능하다. 예를 들어, "점 $p$에서 가장 가까운 객체를 찾아라"와 같은 질의를 주었을 때, 점 $p$의 위치는 각각의 도면에 따라 달리 해석되어야 한다. 그러나 일반적인 사용자는 불일치에 대한 정도나 자세한 정보는 알지 못하므로 불일치로 인한 질의의 처리과정은 시스템이 담당하여야 한다.

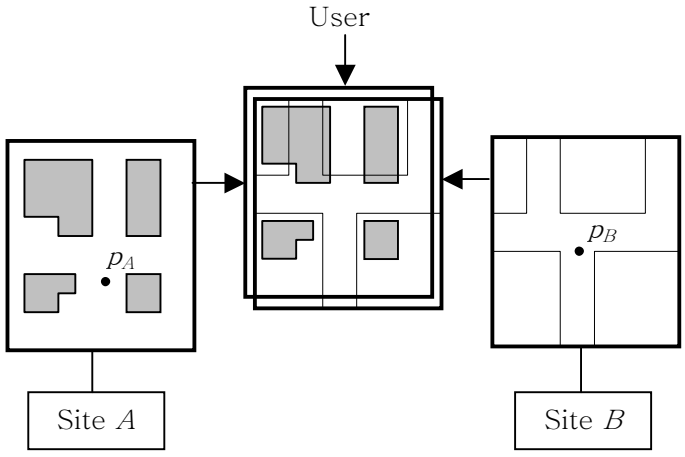



그림 1. 불일치되는 공간데이터베이스

지금까지 많은 노력이 공간데이터베이스의 모델이나 스키마, 그리고, 공간데이터베이스가 제공하는 기능에 대한 이질성 극복을 위하여 이루어졌다. ISO TC/211[11]이나 OGC[12]의 작업과 같이 표준화 작업은 이와 같은 이질성을 극복하는 대표적인 시도들이다. 그러나, 이와 같은 시도는 공간데이터의 내용에 대한 불일치 보다는 기능이나 형식적인 이질성에 초점이 맞추어져 있다. 연합된 공간데이터베이스를 사용하기 위해서는 기능이나 형식적인 이질성 뿐 아니라, 내용의 불일치도 함께 해결되어져야 한다.

본 논문에서는 이와 같이, 위치가 서로 불일치되는 연합 공간데이터베이스에서의 질의처리 방법을 제안한다. 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 이를 미들웨어에서 처리 할 수 있도록 시스템 구조를 제안하고 마지막으로 6장에서 결론을 맺는다.

## 2. 기존 연구 및 배경

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

탄성변형에 대한 기존 연구들을 살펴보면 다음과 같다. [13]에서 제시된 수치지도 불일치 보정에 관한 연구에서는 불일치가 존재하는 기본도와 설비도의 전주에 대해 자동보정방법을 제안하였다. [13]에서 제시한 자동보정 처리과정을 간단히 살펴보면 다음과 같다.

단계1. 전처리 : 불일치 유형 분류
단계2. 보정 : 누락점 위치보정
단계3. 후처리 : 보정점 위치검증

단계1은 기본도와 설비도 사이의 불일치 유형을 분류한다. 들로니 삼각분할을 이용한 유사도를 비교하여 기본도와 설비도 사이에 대응되는 전주의 쌍을 기준점으로 설정한다. 단계2는 들로니 삼각분할을 이용한 삼각형 와핑방법을 사용하여 누락된 전주를 포함하고 있는 삼각형을 찾아 보정한다. 단계3은 누락점 위치보정 후 기본도내 다른 지형지물과 비교하여 위상적으로 잘못 위치 했을 경우에 전주의 위치를 다시 설정할 수 있도록 한다.

탄성변형에 대한 연구는 [1],[2],[3]에서도 제시되고 있다. [2]에서는 서로 다른 데이터베이스에서 다른 종류의 갱신을 수행 시 지리적 위치나 위상관계가 변하는 등의 발생하는 문제점을 해결하고자 하였다. 이 연구에서 제시하는 기법에 대해 간단히 살펴보면, 기존에 존재하는 지적 데이터베이스와 도로와 같은 지리적인 데이터베이스가 있다고 가정하자. 도로 데이터베이스에 새로운 도로가 추가된 후 두 데이터베이스를 합칠 경우, 새로 추가된 도로가 지적의 위치에 정확하게 위치하지 않을 수도 있거나 도로에 의해 지적이 분리될 수 있다. 이를 경우 [2]에서는 다음과 같은 알고리듬을 제시하고 있다.

**알고리듬**

```
지적에 있는 각 구역에 대해 {
- 구역의 경계와 도로의 경계 사이의 교차점을 조사
- 도로와 구역 윤곽에 대하여 새로운 점들을 생성
- 도로와 구역 윤곽을 재구축
- 나누어진 구역에 대해 새로운 정의
}
```

물론, 위의 알고리듬에는 탄성변형 기법을 사용하며, [2]에서는 점 좌표 일치, 각 선마다의 부분의 일치, 각도 일치 등의 방법을 제시하고 있다. 자세한 설명은 [2]를 참조한다. 이러한 종류의 기법은 [1]과 [3]에서도 나타난다. [3]은 [1]을 개선한 방법으로 동일한 지역을 가지는 두 지도 사이에서의 불일치 문제를 새로운 탄성변형 기법을 제시함으로써 해결하고자 하였다. 새로운 탄성변형 기법은 기존에 많이 사용했던 TIN 방법이 아닌 polygon을 분해하는 방법을 사용하였다. 이 방법은 두 지도의 유사성을 바탕으로 서로 대응되는 polygon의 유사성을 기존의 0.7에서 더 감소시켜 탄성변형 기법의 성능을 향상하고자 하였다. 특히, 이 연구에서는 새로운 탄성변형 기법을 제시함으로써 지도상의 불일치를 보정한 한 후에도 지리 객체 사이의 위상관계나 지리적 정확성 등이 보존된다는 것을 강조하고 있다. 자세한 내용은 [3]을 참조한다.

불일치 문제를 해결하는 또 다른 방법에는 Affine transformation 기법이 있다[7]. 이 기법은 3×3 행렬을 이용하여 입력되는 point, polygon, vector, circles 등을 변형시키는 것을 말한다. 이 기법 역시 불일치되는 부분들을 보정시키기 위하여 제시되었다. 이 기법에 의해 변형되어진 모양은 변형하기 전의 속성을 그대로 유지함으로써 불일치되는 부분들을 보정시키고자 하였다.

위에서 열거한 탄성변형에 의한 방법은 하나의 공간데이터베이스를 다른 공간데이터베이스에 맞게 전체적으로 수정하는 것이다. 그러나, 많은 경우 각각의 공간데이터베이스는 자신의 자치권(Autonomy)을 가지고 운영되기 때문에, 하나의 데이터베이스를 전체적으로 수정하는 것이 불가능하다. 따라서, 각각의 데이터베이스에 독립성을 부여하여 처리 할 수 있는 연합적 분산 공간데이터베

이스 시스템 (Federated Spatial Database System)적인 접근방법을 취할 수 밖에 없다.

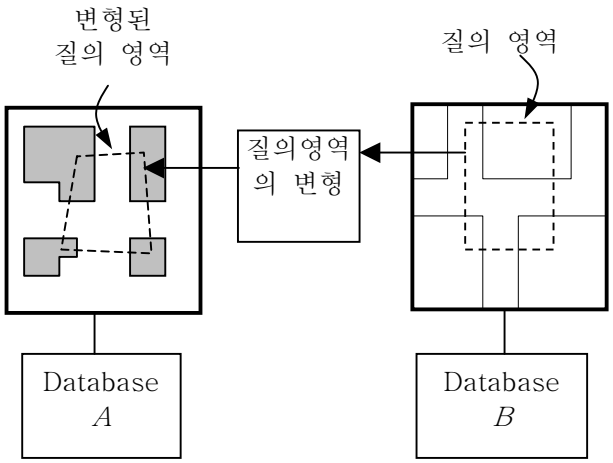



그림 2. 공간 질의의 변형

연합적 분산 공간데이터베이스 시스템에의 질의처리는 위에서 서술한 이유에서 다른 접근방법이 필요하다. 예를 들어, 사용자가 하나의 공간데이터베이스의 공간객체의 위치를 참조하여 질의를 주었을 경우, 다른 공간데이터베이스의 보정없이 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. 그림 2와 같은 방법으로 질의를 처리하면 각각의 공간데이터베이스는 독립적으로 자율권을 확보하게 된다.

따라서, 본 논문에서는 여러 개의 공간데이터베이스가 독립적으로 운영되고, 위치가 서로 불일치되는 경우, 공간적 질의를 처리하는 방법을 제안한다.

## 3. 들로니 삼각분할을 이용한 탄성변형

본 논문에서 제안하는 방법의 기본적 방법론적인 토대는 들로니 삼각분할을 이용한 탄성변형에 기초하고 있다. 따라서, 본 장에서는 들로니 삼각분할을 이용한 탄성변형 방법을 간단하게 소개한다[10]. 여기서 탄성변형이라고 하는 것은 공간객체사이의 위상적 관계가 변하지 않는 범위 내에서 이동, 휘어짐, 크기 조절 등으로 공간의 위치를 변형하는 것을 의미한다[1][2][3].

들로니 삼각분할을 이용한 탄성변형의 기본적 개념은 각각의 공간데이터베이스에 주어진 기준점을 이용하여 들로니 삼각분할을 수행한다. 삼각분할을 통하여 얻어지는 대응 삼각형의 변형정도를 고려하여 탄성변형을 하는 것이다. 이 과정을 알고리듬으로 나타내면 다음과 같다.

**알고리듬 들로니 삼각분할을 이용한 탄성변형**

**입력** $M_S$ : 변형의 기준이 되는 공간데이터베이스의 꼭지점 집합
$M_T$ : 변형이 실제로 일어나는 공간데이터베이스의 꼭지점 집합
$G = \{ (g_s, g_t) \mid g_s$와 $g_t$는 $M_S, M_T$ 에 각각 주어진 대응되는 기준점 $\}$
**출력** $M_T$ : 꼭지점 위치가 보정된 $M_T$

**알고리듬 시작**
$T_S \leftarrow$ DelaunayTriangulation($G_S$);
// $G_S = \{g_s \mid g_s$ 는 $M_S$ 의 기준점 $\}$,
// $T_S$ 는 들로니 삼각분할로 만들어진 삼각형의 집합
$T_T \leftarrow$ MappingTriangles($T_S$);
// $T_S$ 에 대응되는 $M_T$ 의 삼각형 집합
각각의 $p_T \in M_T$ 에 대하여 {
  $t_T \leftarrow$ FindTriangleContaining($T_T, p_T$);
  // $p_T$를 포함하는 삼각형 탐색
  $t_S \leftarrow$ MappingTriangles($t_T$);
  // $t_T$ 에 대응되는 $T_S$ 의 삼각형 탐색
  RubberSheeting($t_s, t_T, p_T$);
  // $t_s$ 와 $t_T$를 이용하여 $p_T$ 의 위치보정
}
**알고리듬 끝**

위의 알고리듬이 수행되는 과정에서 가장 중요한 부분은 RubberSheeting($t_s$, $t_T$, $p_T$)에 해당하는 삼각형을 이용한 탄성변형 부분이다. 삼각형을 이용한 탄성변형은 다음과 같이 수행된다. $M_T$상의 점 $p_T(x_T, y_T)$ 에 대하여 이를 포함하는 삼각형을 $t_T = (p_{T1}, p_{T2}, p_{T3})$ 라고 하고 이에 대응되는 $M_S$상의 삼각형을 $t_S = (p_{S1}, p_{S2}, p_{S3})$ 라고 하자. 이 때, $p_T$ 의 점이 탄성변형된 공간에서의 좌표 $p_S(x_S, y_S)$는 다음과 같이 계산된다. 우선, $p_T$ 는 다음과 같이 표현된다.

$$x_T = \lambda_1 x_{T1} + \lambda_2 x_{T2} + \lambda_3 x_{T3} \quad -(1)$$
$$y_T = \lambda_1 y_{T1} + \lambda_2 y_{T2} + \lambda_3 y_{T3} \quad -(2)$$

여기서 $p_T$ 는 삼각형 내의 점이므로 $\lambda_1 + \lambda_2 + \lambda_3 = 1$ 이며, $0 \le \lambda_1, \lambda_2, \lambda_3 \le 1$ 이 된다. 또한 변형된 공간상의 대응되는 점 $p_S$도 마찬가지로 아래와 같이 계산된다.

$$x_S = \lambda_1 x_{S1} + \lambda_2 x_{S2} + \lambda_3 x_{S3} \quad -(3)$$
$$y_S = \lambda_1 y_{S1} + \lambda_2 y_{S2} + \lambda_3 y_{S3} \quad -(4)$$

여기서 $\lambda_1, \lambda_2, \lambda_3$ 은 식 (1), (2)와 $\lambda_1 + \lambda_2 + \lambda_3 = 1$을 이용하면 다음과 같이 표현된다.

$$\lambda_1 = \omega\{ (x_T - 1)(y_{T2} - y_{T3}) - (y_T - 1)(x_{T2} - x_{T3}) \} \quad -(5)$$
$$\lambda_2 = \omega\{ (x_T - 1)(y_{T3} - y_{T1}) - (y_T - 1)(x_{T3} - x_{T1}) \} \quad -(6)$$
$$\lambda_3 = \omega\{ (x_T - 1)(y_{T1} - y_{T2}) - (y_T - 1)(x_{T1} - x_{T2}) \} \quad -(7)$$

이때 $\omega = 1/(x_{T1} y_{T1} + x_{T2} y_{T2} + x_{T3} y_{T3} - x_{T1} y_{T3} - x_{T2} y_{T1} - x_{T3} y_{T2})$ 이다. 결국 식 (3) – (7)을 이용하면 원하는 $p_S$ ($x_S$, $y_S$)의 값을 구할 수 있다. 위의 알고리듬이 수행되는 과정을 간단하게 그림으로 나타내면 그림 3과 같다.

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

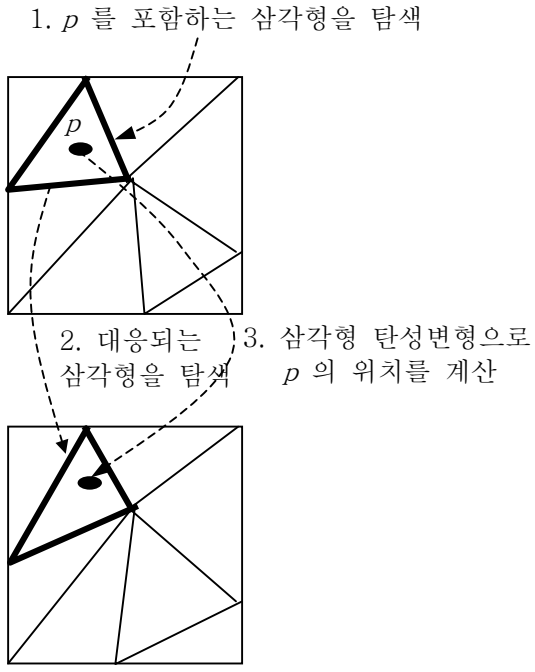



그림 3. 들로니 삼각분할을 이용한 탄성변형 과정

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 최대 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 $G_S = \{ g_{S1}, g_{S2}, \ldots, g_{Sn} \}$, $G_T = \{ g_{T1}, g_{T2}, \ldots, g_{Tn} \}$ 이라고 하면, 변형전의 임의의 점 $p_S$ 와 변형된 점 $p_T$ 는 다음의 조건을 만족한다.

$$0 \le \mathrm{dist}(p_T, p_S) \le \mathrm{Max}( \mathrm{dist}(g_{Tk}, g_{Sk}), 0 \le k \le n )$$

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 질의처리의 탄성변형

본 장에서는 위치가 일치하지 않는 공간데이터베이스 시스템들이 있을 때, 질의처리하는 방법을 제시한다. 본 논문에서 제시하는 방법의 기본적 개념은 각각의 공간데이터베이스를 수정하지 않고 주어진 질의의 영역이나 위치를 보정하여 각 시스템에서 공간질의를 수행하는 것이다. 질의의 영역을 보정하는 방법은 앞 절에서 설명한 들로니 삼각분할의 탄성변형을 이용한다. 본 논문에서는 최근접 질의 처리방법과 포함질의 처리방법을 제시한다. 다른 공간질의의 처리 방법도 이 두 가지 처리방법과 유사하므로 본 논문에서는 자세한 언급은 생략한다.

위치가 일치하지 않는 공간데이터베이스들에게 공간질의를 요청할 때는 항상 기준이 되는 공간데이터베이스가 존재한다. 만일, 시설도와 교통도를 각각 나타내는 공간데이터베이스 시스템이 있을 경우, 사용자는 하나의 공간데이터베이스를 선택하여 질의를 요청한다. 예를 들어, 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.

즉 기준이 되는 공간데이터베이스로부터 다른 공간데이터베이스로 변환이 일어나는 방향이 앞 절에서 서술한 공간데이터베이스 전체를 변형하는 방법의 방향과 반대로 된다는 것을 주목할 필요가 있다. 이 과정을 간단하게 그림으로 나타내면 그림 4와 같다.

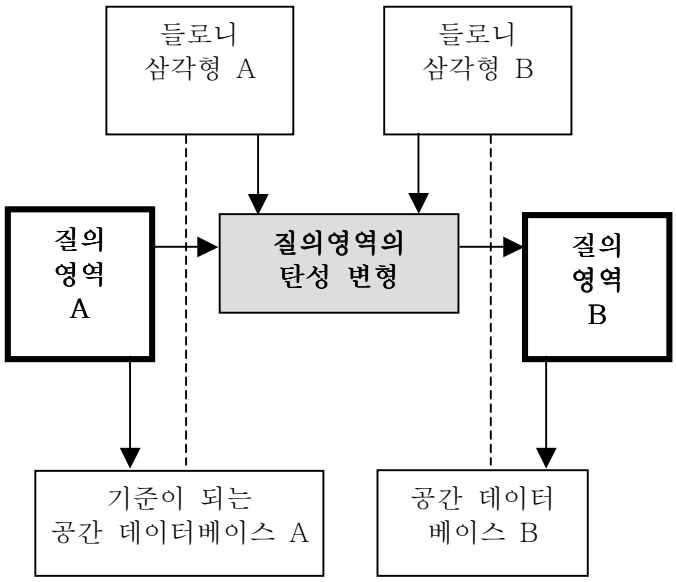



그림 4. 공간 질의 영역의 탄성 변형과정

## 4-1. 들로니 삼각분할에 따른 탄성변형을 이용한 최근접 질의의 처리 방법

최근접 질의를 위한 질의영역은 하나의 점으로 주어진다. 따라서, 질의영역의 탄성변형은 질의가 주어지는 공간데이터베이스의 질의위치를 다른 공간데이터베이스의 질의위치로 변환하여 질의를 처리하게 된다. 이 과정을 다음과 같은 알고리듬으로 간단하게 나타내 보았다. 다음 알고리듬에서 RubberSheeting($t_s$, $t_T$, $p_S$)의 함수는 앞 절에서 설명한 삼각형에 의한 보정의 방법을 이용한다. 또한 각 공간데이터베이스에 대한 최근접 질의처리 PerformQuery("Nearest Neighbour", $S_S$, $p_S$) 는 기존의 여러 가지 방법[8][9][14] 중 한가지를 선택하여 처리하면 된다.

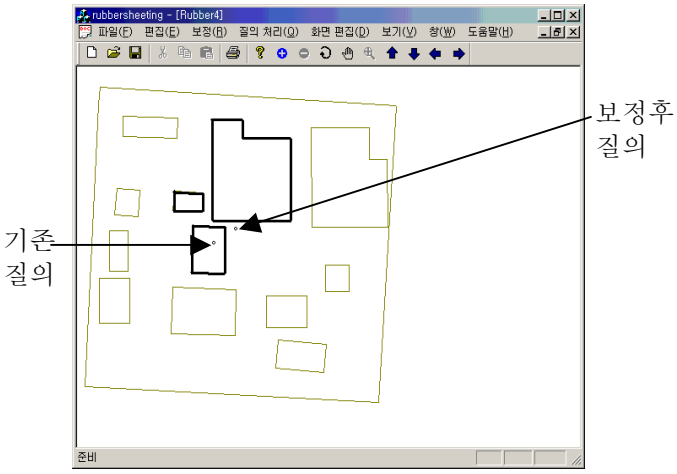



그림 5. 최근접 질의처리 결과 ($k$ = 3)

**알고리듬 들로니 탄성변형을 이용한 최근접 질의처리**

**입력** $S_S$ : 질의가 주어진 기준이 되는 공간데이터베이스
$S_T$ : 다른 공간데이터베이스
$G = \{ (g_s, g_t) \mid g_s$와 $g_t$는 $S_S$, $S_T$ 에 각각 주어진 대응되는 기준점 $\}$
$p_S$ : 데이터베이스 $S_S$ 에 주어진 질의의 위치
**출력** $R_S$ , $R_T$ : $S_S$, $S_T$ 에 대하여 처리한 질의 결과

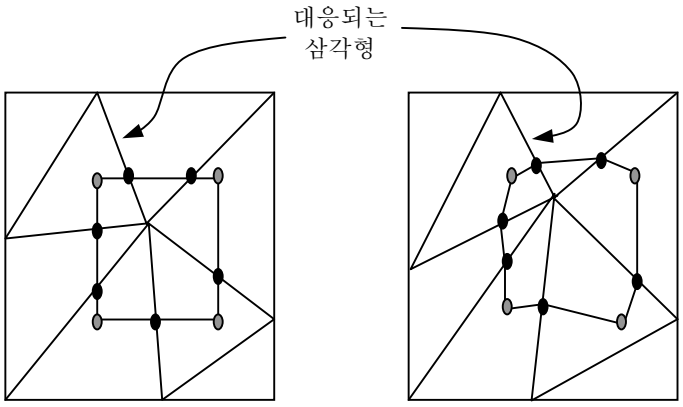
**알고리듬 시작**
$T_S \leftarrow$ DelaunayTriangulation($G_S$);
// $G_S = \{g_s \mid g_s$ 는 $S_S$ 의 기준점 $\}$
$T_T \leftarrow$ MappingTriangles($T_S$);
// $T_S$ 에 대응되는 $M_T$ 의 삼각형 집합
$t_S \leftarrow$ FindTriangleContaining($T_S$, $p_S$);
// $p_S$를 포함하는 삼각형 탐색
$t_T \leftarrow$ MappingTriangle($t_S$);
// $t_S$ 에 대응되는 $T_T$ 의 삼각형 탐색
$p_T \leftarrow$ RubberSheeting($t_s$, $t_T$, $p_S$);
// $t_s$ 와 $t_T$를 이용하여 $p_T$ 의 위치보정
$R_S \leftarrow$ PerformQuery("Nearest Neighbour", $S_S$, $p_S$);
$R_T \leftarrow$ PerformQuery("Nearest Neighbour", $S_T$, $p_T$);
**알고리듬 끝**

그림 5는 위의 알고리듬을 이용하여 지적도와 지형도의 두 개의 공간데이터베이스에 질의를 처리한 결과이다.

## 4-2. 들로니 삼각분할에 따른 탄성변형을 이용한 포함질의 처리

포함질의를 처리하는 과정은 최근접 질의를 처리하는 것보다 복잡하다. 그 이유는 최근접 질의의 경우, 질의영역은 하나의 점으로 주어지지만 포함질의는 질의영역이 다각형으로 주어지기 때문이다. 다각형을 탄성변형하는 과정은 점을 탄성변형하여 위치를 찾는 과정보다 복잡하다. 들로니 삼각분할을 이용하여 다각형을 탄성변형하는 과정은 그림 6에 의하여 설명된다.

그림 6에서 회색 점으로 표현된 것은 원래의 질의영역의 꼭지점이 된다. 그러나, 그림에 나타난 것과 같이 질의영역이 들로니 삼각분할 방법으로 탄성변형될 때 고려되는 점은 단순히 원래의 꼭지점 뿐 아니라, 질의영역의 변과 들로니 삼각형의 변이 교차되는 점들도 포함한다. 그림에서 회색점으로 나타나 있는 이와 같은 점들을 고려한 질의영역의 탄성변형 알고리듬은 다음과 같다.




a. 질의가 참조한 기준이 되는 공간 데이터베이스의 질의영역

b. 탄성변형되어 질의가 처리되어야하는 공간데이타베이스의 질의영역

그림 6. 포함질의의 처리과정

**알고리듬 들로니 탄성변형을 이용한 포함질의 처리**

**입력** $S_S$ : 질의가 주어진 기준이 되는 공간데이터베이스
$S_T$ : 다른 공간데이터베이스
$G = \{ (g_s, g_t) \mid g_s$와 $g_t$는 $S_S, S_T$ 에 각각 주어진 대응되는 기준점 $\}$
$a_S = (p_{S1}, p_{S2}, p_{S3}, \ldots, p_{Sn})$ : 데이터베이스 $S_S$ 에 주어진 질의의 영역

**출력** $R_S$ , $R_T$ : $S_S, S_T$ 에 대하여 처리한 질의 결과

**알고리듬 시작**

$T_S \leftarrow$ DelaunayTriangulation($G_S$);
// $G_S = \{g_s \mid g_s$ 는 $S_S$ 의 기준점 $\}$,
$T_T \leftarrow$ MappingTriangles($T_S$);
// $T_S$ 에 대응되는 $M_T$ 의 삼각형 집합
$b_S \leftarrow$MakeNewPolygonWithIntersectingPoints (MappingTriangles($a_S$ , $T_S$);
// $a_S$ 와 삼각형의 변과 교차하는 점으로 질의영역 생성
각 $b_S$ 의 꼭지점 $q_{Sk} \in b_S$ 에 대하여 {
  $t_S \leftarrow$ FindTriangleContaining($T_S$, $q_{Sk}$);
  // $q_{Sk}$ 를 포함하는 삼각형 탐색
  $t_T \leftarrow$ MappingTriangle($t_S$);
  // $t_S$ 에 대응되는 $T_T$ 의 삼각형 탐색
  $q_{Tk} \leftarrow$ RubberSheeting($t_s$, $t_T$, $q_{Sk}$);
  // $t_s$ 와 $t_T$ 를 이용하여 $p_T$ 의 위치보정
}
$a_T \leftarrow (q_{T1}, q_{T2}, q_{T3}, \ldots, q_{Tm})$;
$R_S \leftarrow$ PerformQuery("Contain", $S_S$, $a_S$);
$R_T \leftarrow$ PerformQuery("Contain", $S_T$, $a_T$);

**알고리듬 끝**

위의 알고리듬에서 사용하는 포함질의PerformQuery ("Contain", $S_S$, $a_S$) 는 일반적인 포함공간연산을 이용하여 수행된다. 공간 색인을 이용하면 더욱 좋은 성능을 기대할 수 있다. 공간 색인에는 R-tree[5], R*-tree[4] 그리고 Grid Files[6] 등 여러 가지의 공간 색인 방법이 제안되어 왔다.

본 논문에서는 R-tree를 토대로 보다 효과적인 클러스터링을 지원하기 위하여 제안된 R*-tree 색인 구조를 사용하여 공간연산을 수행하도록 제시하였다.

그림 7은 위의 알고리듬을 이용하여 포함질의를 지적도와 지형도의 공간데이터베이스에 대하여 실행한 결과이다. 점선은 주어진 포함 질의영역이 보정 후에 변형된 질의영역을 나타내고 있다. 굵은 색은 네 개의 객체가 포함되고 있음을 나타내고 있다.

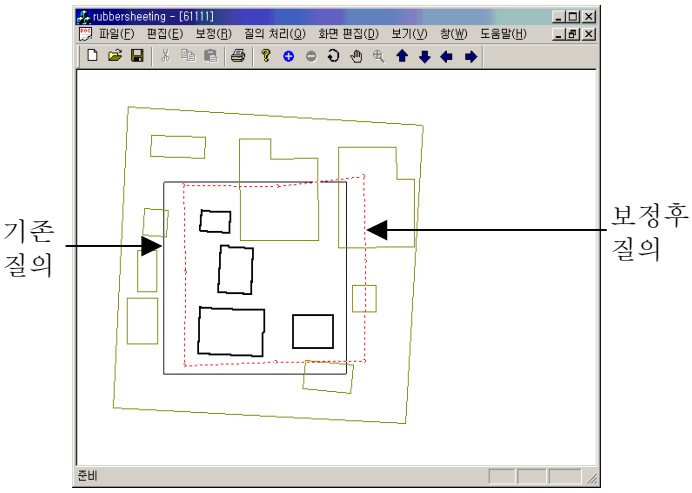



그림 7. 포함 질의처리 결과

위의 두 질의처리 방법의 성능은 각각의 공간데이터베이스 시스템의 질의처리 성능에 따라 결정된다. 물론, 질의영역을 변형하는 시간이 추가되지만 이는 주로 계산시간이므로 각 데이터베이스의 질의처리 시간에 비하여 무시할 만큼 작은 시간이 걸린다. 따라서, 본 논문에서는 성능적인 측면은 고려하지 않는다.

## 5. 들로니 삼각분할의 탄성 변형을 이용한 공간질의처리 시스템의 구조

본 장에서는 앞 장에서 제시한 질의처리기능을 구현하기 위하여 필요로 하는 시스템의 구조에 대하여 알아본다. 본 논문에서 제시하는 질의처리 방법의 가장 중요한 개념은 들로니 삼각분할을 이용한 질의영역의 변형이다. 따라서, 각 공간데이터베이스 시스템에서는 서로 대응되는 기준점에 관한 정보를 저장하고 있어야 한다. 그런데, 이 정보는 각각의 공간데이터베이스 시스템이 가지고 있는 것이 아니라, 일종의 미들웨어에서 가지고 있는 것이 더욱 효과적이다. 이를 그림으로 나타내면 그림 8과 같다.

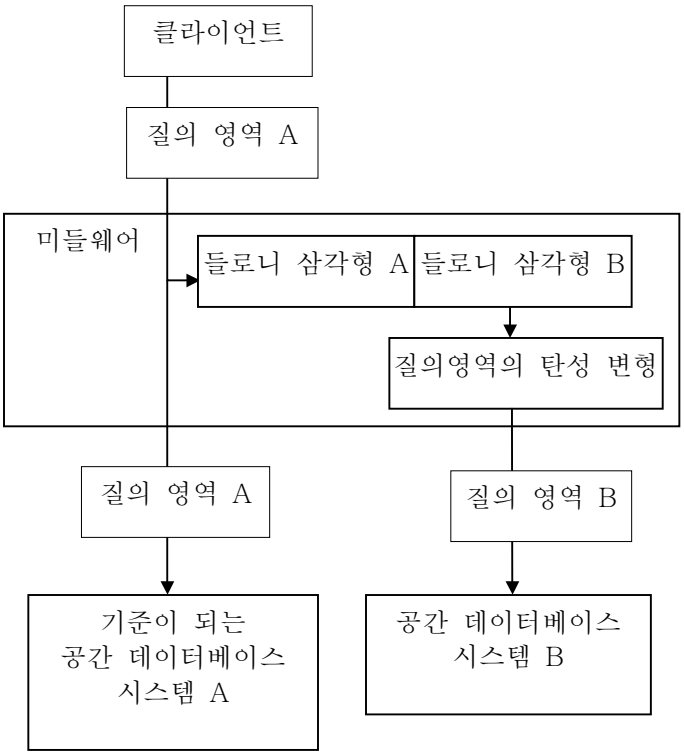



그림 8. 들로니 삼각분할의 탄성 변형을 이용한 공간질의처리 시스템의 구조

그림에서와 같이 미들웨어는 들로니 삼각분할을 위한 삼각형들의 집합을 가지고 있다. 경우에 따라서는 단순히 대응되는 기준점들의 집합을 가지고 있어도 동적으로 들로니 삼각형들을 구할 수 있다. 그러나 기준점의 수가 많아지면, 삼각분할의 시간이 많이 소모되므로 대응되는 삼각형들의 집합을 가지고 있는 것이 효과적이다. 사용자가 클라이언트에 질의를 요청하면 이는 미들웨어에 전달된다. 미들웨어는 전달받은 질의를 해당되는 공간데이터베이스 시스템에 다시 전달한다. 이 때, 각 공간데이터베이스 시스템으로 질의를 전달하기 전, 미들웨어가 관리하는 들로니 삼각형의 정보를 이용하여, 질의의 영역을 해당 공간데이터베이스에 맞게 변형한다. 변형된 질의영역과 함께 주어지는 공간질의를 각 공간데이터베이스 시스템이 받아 처리하고, 결과를 미들웨어를 통하여 해당 클라이언트에 전달한다.

미들웨어가 들로니 삼각형에 관한 정보를 관리하기 위하여서는 각 공간데이터베이스의 구축 시 미들웨어에게 들로니 삼각형에 관한 정보를 등록하여야 한다. 이 과정에 대하여서는 더욱 많은 연구와 작업이 필요하다.

## 6. 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 위하여서는 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.

## 7. 참고문헌

[1] M. G. Cho, K. J. Li and H. G. Cho, “A Rubber Sheeting Method With Polygon Morpging”, The 7th International Symposium on Spatial Data Handling Vol. 1, pp. 7A 31-42, 1996

[2] Sylvie SERVIGNE, Robert LAURINI, “Updating Geographic Databases Using Multi-Source Information”, ACM-GIS 96’

[3] Mi-Gyung Cho and Hwan-Gue Cho, “Resolving Mismatches and Measure Functions for Evaluating its Validity”, SDH 2000’

[4] 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

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

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

[7] Kurt Mehlhorn, Stefan Naher, Michael Seel, Christian Uhrig, “The LEDA user Manual”, Version 4.2

[8] K. Cheung, and A. Fu, “Enhanced Nearest Neighbor Search on the R-tree”, ACM SIGMOD

Record 27(3), pp.16-21, 1998
[9] Roussopoulos N., Kelley S., Vincent F, “Nearest Neighbor Queries”, Proc. ACM SIGMOD Int. Conf. On Management of Data, San Jose, CA, 1995, pp.71-79
[10] J. O’Rourke, Computational Geometry in C, Camgridge University Press, 1994
[11] ISO/TC211, Geographic information/ Geomatics, 1999
[12] The OpenGIS Abstract Specification, Open GIS Consortium, version 4
[13] 박상미, 정규상, 손은정, 이기준, 지리 정보 시스템용 수치 지도 자동 보정 기법, 한국정보과학회 97 가을 학술발표논문집, 제 24권 2호, 1997
[14] 장인성, 이기준, 밀도를 이용한 k-최근접 탐색 방법, 한국정보학회 2000 가을 학술발표논문집, 제 27권 2호, 2000