Journal of Korea Multimedia Society Vol. 9, No. 4, April 2006(pp. 388-400)

LBS를 위한 위치 데이터 관리 시스템 설계 및 적용

안 윤 애†

요 약

LBS의 주요 기술 구성요소에는 무선측위기술 또는 위치결정기술, LBS 핵심기반 기술 및 LBS 응용기반 기술 등이 있다. 이 논문에서는 LBS의 주요기술 구성요소 중 핵심기반기술인 위치 데이터 관리 시스템을 설계한다. 제안 시스템은 LBS 응용인터페이스, 응용서비스 질의처리기, 이동객체 위치추정기, 위치정보 관리기, 실시간 데이터 수신기, 위치정보 데이터베이스로 구성된다. 데이터베이스 관리 기술을 활용하여 이동객체의 위치변화 정보를 효율적으로 관리하고 이와 관련된 유용한 정보를 LBS 사용자에게 제공하며, 기존의 상용 데이터베이스에서 처리할 수 없는 이동객체의 연속적인 위치정보를 처리하는 연산 및 위치추정 기능을 지원한다. 실시간 환경에서 발생되는 위치정보의 결손을 보완하는 위치정보 트리거링을 토대로 기존의 위치정보 관리 시스템이 갖는 문제점을 보완한다. 제안한 위치 데이터 관리 시스템의 응용을 위해 산불관리 응용서비스에 적용한 결과를 보인다.

Design and Application of Location Data Management System for LBS

Yoon-Ae Ahn†

ABSTRACT

There are wireless location acquisition technique, LBS platform technique, and LBS application technique in the important technical elements of the LBS. In this paper, we design a location data management system which is the core base technique of the important technical elements of the LBS. The proposed system consist of an application interface of LBS, a query processor of application service, a location estimator of the moving objects, a location information manager, a real-time data receiver, and a database of location data. This system manages efficiently the location change information of the moving objects using the database technique, suggests some useful inform to the users of LBS, and supports operation and facility of location estimation to process continuous location data of the moving objects. On the basis of location data triggering, this system supplements the problem of the related location data management systems to complement the loss of location data in the environment of real-time.

Key words: Location Based Service(위치기반서비스), Moving Object(이동객체), Location Data Management System(위치데이터관리시스템)

1. 서 론

LBS(Location Based Service:위치기반서비스)는 위치확인기술 및 유·무선 통신망을 통해 PDA, 이동전화 등 이동통신 단말기 소지자의 위치를 실시간으로 파악하여, 사용자가 필요로 하는 다양한 콘텐츠

※ 교신저자(Corresponding Author) : 안윤애, 주소 : 충북 충주시 이류면 검단리 123(380-702), 전화 : 043)820-5269, FAX : 043)820-5262, E-mail : ycahn@chungju.ac.kr

† 정회원, 충주대학교 전기전자 및 정보공학부 조교수

접수일 : 2005년 7월 20일, 완료일 : 2006년 1월 11일

및 응용서비스를 제공하는 것을 의미한다. LBS의 주요한 특징은 고객이 요구하는 정보를 이동통신 단말기를 통해 실시간으로 제공하는 즉시성, 전국 어디에서나 서비스 제공이 가능한 이동성, 개인의 현재 위치와 요구사항에 맞는 서비스 제공의 개인성 등을 들 수 있다. LBS는 위치를 이용한 날씨 서비스, 위치 찾기 서비스, 교통 정보 서비스, 모바일 광고 서비스, 엔터테인먼트 서비스, 보안 서비스, 소방방재시스템, 화물 추적 및 차량 운행 관리, 모바일 상거래, 의료, 환경, 물류관제시스템, 유통, 보험사 등에 응용될 수 있다. LBS 시장은 이동통신, 휴대 인터넷, 유·무선 통합 등의 새로운 통신기술과 접목되어 이동통신 서비스 시장의 큰 부분을 차지할 것으로 예상된다[1-3]. LBS를 가능하게 하는 주요 기술 구성요소에는 이동통신망에서 위치를 파악하는 무선측위기술 또는 위치결정기술(location determination technology), 위치 데이터 관리를 위한 LBS 핵심기반(platform) 기술 및 LBS 응용기반 기술 등이 있다.

이 논문에서는 LBS의 주요기술 구성요소 중 핵심 기반기술인 위치 데이터 관리 방법에 관한 연구로 위치 데이터 관리 시스템 구축에 관해 기술한다. LBS를 위한 위치 데이터 관리는 위치 데이터의 획득 방법에 따라 사건지향 시스템과 관측지향 시스템의 두 가지 형태로 분류된다. 사건지향 시스템은 이동객체의 속도나 방향의 변화를 자동으로 검출하는 것이 가능한 경우의 시스템을 말한다. 관측지향 시스템은 GPS와 같은 센서 시스템을 이용하여 정규적인 시간의 간격에서 정렬된 순서대로 객체의 위치를 획득하는 시스템을 말한다.

이 논문에서 설계하는 위치 데이터 관리 시스템은 관측지향 시스템을 기반으로 LBS 응용인터페이스, 응용서비스 질의처리기, 이동객체 위치추적기, 위치정보 관리기, 실시간 데이터 수신기, 위치정보 데이터베이스로 구성된다. 특히 데이터베이스 관리 기술을 활용하여 이동객체의 위치변화 정보를 효율적으로 관리하고 이와 관련된 유용한 정보를 LBS 사용자에게 제공하며, 기존의 상용 데이터베이스에서 처리할 수 없는 이동객체의 연속적인 위치정보를 처리하는 연산 및 위치추정 기능을 지원한다. 또한 이력 정보의 불확실성 값의 범위를 제공하며, 실시간 환경에서 발생되는 위치정보의 결손을 보완하는 위치정보 트리거링을 토대로 기존의 위치 데이터 관리 시스템이 갖는 문제점을 보완한다. 제안한 위치 데이터 관리 시스템의 적용을 위해 산불관리 응용서비스에 적용한 결과를 보인다.

이 논문의 전체 구성은 다음과 같다. 2절에서는 기존에 연구된 LBS를 위한 위치 데이터 관리 시스템의 관련연구를 기술한다. 3절에서는 LBS에 사용되는 이동객체의 위치 데이터 특성 및 실시간 위치 데이터 전송구조에 대해 기술한다. 4절에서는 이 논문에서 제안하는 LBS를 위한 위치 데이터 관리 시스템을 설계 및 적용한 LBS 응용인터페이스를 제시한 후 제안 시스템과 관련 시스템과의 특징을 비교한다. 마지막으로 5절에서는 결론을 맺는다.

2. 관련연구

LBS를 위한 이동객체의 위치 데이터 관리 시스템[4]은 자동차, 비행기, 배, PDA, 노트북 등과 같은 객체들의 연속적인 위치변화 과정을 저장 및 관리하며, 저장된 위치 데이터를 이용하여 사용자에게 다양한 질의 처리 기능을 제공하기 위한 연산 기능을 수행한다. 지금까지 연구된 대표적인 관련 프로토타입에는 DOMINO, CHOROS, DEDALE, STRBA, MOMS 등이 있다.

DOMINO[5-7]는 추측 항법의 위치추정 기법을 적용한 실시간 이동객체의 위치추적 프로토타입이다. 이동객체의 동적 속성을 위한 MOST 모델을 제안하였으며, FTL을 이용한 미래 시간에 대한 질의 표현 방법을 제시하였다. 이 시스템은 기존의 상용 DBMS 기술과 GIS를 이용한 인터페이스를 사용하였다. 그러나 이 프로토타입은 이동객체의 현재 위치, 속도, 방향 정보를 이용하여 미래의 위치를 예측하는 데 주로 초점을 맞추고 있으며 이동객체의 과거 위치정보를 데이터베이스에 전혀 저장하지 않는다. 따라서 과거 시점부터 현재 시점까지를 모두 포함하는 이동객체의 이동 궤적과 관련된 질의 처리 기능을 지원하지 못하는 제약사항이 있다.

CHOROS[8-10]는 시공간 데이터베이스 시스템의 설계 및 구현에서 포함된 문제들을 연구하여 시공간 데이터베이스 시스템의 구조를 제안하고 부분적으로 구현하고 있다. 또한, GPS 기반의 수송 관리 시스템과 멀티미디어 시스템에 적용한 응용 시나리오를 제시하였다. 그러나 아직 이동객체 데이터베이

스를 활용한 응용 시스템의 모델 및 개발 사례는 제시되지 않고 있으며, 개발 중인 질의 처리 시스템에서는 이동객체의 불확실한 과거 및 미래의 위치정보 추정에 관한 구체적인 방법이 제시되지 않고 있다.

. DEDALE[11,12]은 제약사항 데이터베이스 모델을 이용하여 시공간 데이터를 모델링하고 질의 처리를 하기 위해 개발된 프로토타입이다. DEDALE은 기존의 시공간 데이터의 모델뿐만 아니라 이동객체의 궤적 등과 같은 데이터 모델 및 질의 표현도 제공한다. 그러나 DEDALE 프로토타입은 데이터베이스에 시간의 변화에 따른 이동객체의 위치정보가 직접 저장되지 않고, 특정 구간의 궤적을 표현하는 선형 제약사항의 공식이 저장되므로 실시간 위치 모니터링을 위한 응용 시스템의 질의 처리 기능을 지원하지 못한다.

STRBA[13-16]는 전장분석을 위한 시공간 추론 시스템 프로토타입으로 모의 전장에서 이동하는 부대 및 탱크들의 움직임을 예측하여 이를 의사결정에 활용할 수 있도록 개발되었다. 전장분석 프로토타입은 이동객체 관리기와 추론 엔진을 접목시키고자 하는 데 초점이 맞추어졌다. 특히, 시공간 이동객체의 연산 결과를 추론 엔진에서 활용하는 새로운 이동객체 추론 모델을 제시하였다. MOMS[17]는 물류 차량 관리를 위한 이동객체 관리 엔진이다. 이 프로토타입은 기존의 차량 추적 시스템의 기능을 제공함은 물론 이동 차량의 과거 및 현재의 위치정보를 제공한다. 아울러, 기존의 GPS, Beacon, ITS 등의 서로 다른 차량 추적 시스템의 정보를 통합하여 하나의 시스템에서 관리하려는 시도를 하였다.

3. LBS에 사용되는 위치 데이터

자동차, 비행기, 사람, PDA, 휴대폰, 노트북 등과 같이 실세계에서 시간의 흐름에 따라 위치를 변화하면서 이동하는 사물을 이동객체[18-20]라 한다. 이와 같은 이동객체의 모든 유형은 LBS의 응용 대상이 될 수 있으며 LBS의 기반기술인 위치 데이터 관리 시스템의 위치정보 유형이 된다. 이동객체의 이동 형태는 자유 궤적과 제약 궤적으로 분류할 수 있다. 이동객체의 위치 데이터를 획득하는 방법에 따라 사건지향(event driven) 시스템과 관측지향(observation driven) 시스템의 두 가지 형태로 분류할 수 있다.

이 논문에서는 다음과 같은 몇 가지의 가정을 두고 이동객체의 위치 데이터를 관리한다. 첫째, 이동객체는 2차원 공간에서 이동하는 점 객체를 대상으로 한다. 둘째, 이동객체의 위치변화 과정을 자유 궤적으로 간주한다. 셋째, 이동객체의 위치 데이터 관리 방법은 GPS, Beacon 등을 이용하는 관측지향 시스템을 기반으로 한다. 넷째, 이동객체의 위치변화 관측은 모든 객체에 대해 독립적으로 발생시킬 수 있다. 그러나 하나의 객체에 대한 관측은 정규적인 시간 간격에 따라 획득되며 정렬 순서대로 입력된다.

GPS와 같은 센서 시스템으로부터 획득할 수 있는 이동객체의 위치 데이터는 세 가지로 구성되며, 세 쌍의 집합으로 표현되는 상태들의 정렬된 순서로 데이터를 전송받을 수 있다. 이 때 세 개의 원소는 (Oid, t, v)이며, Oid는 점으로 표현되는 객체의 식별자이고, t는 유효 시간을 나타내는 타임스탬프이며, v는 객체의 위치에 해당되는 (x, y) 좌표 값을 나타낸다. 따라서 센서 시스템으로부터 획득하는 이동객체의 위치 데이터는 (객체의 식별자, 타임 스탬프, 위치 좌표 값)으로 구성된다. 표 1은 센서 시스템으로부터 획득한 위치 데이터의 예이다.

이동객체의 실시간 위치 데이터는 위치 제공 서버로부터 일정한 시간의 주기마다 위치 데이터 관리 시스템에 전송된다. 위치 제공 서버는 각 이동객체의 위치 데이터를 관측하여 패킷 형태로 변환한 후 위치 데이터 관리 시스템에 전송한다. 위치 제공 서버로부터 전송되는 패킷의 구조는 응용 시스템의 특성에 따라 서로 다른 구조를 가진다. 표 2는 이동객체의 위치 데이터 패킷의 예이다.

표 2는 위치 데이터 관리 시스템에 저장될 위치 데이터만을 구성 요소로 하여 작성한 패킷 구조이다. 만약 이동객체의 다른 정보들을 필요로 할 경우에는 추가적인 정보를 패킷의 구성 요소에 넣을 수 있다. 그러나 통신비용 및 데이터 송수신 속도를 고려하여 최소한의 위치 데이터만을 이용하여 패킷을 구성한다.

표 1. 센서 시스템에서 획득한 위치 데이터

oid	*t*	*x*	*y*
356583455	07-50	200998.11	445124.01
356583455	07-55	201287.75	445238.44
356583455	08-00	201566.67	445345.72
356583455	08-05	201809.84	445410.08

표 2. 이동객체의 실시간 위치 데이터 패킷

구성요소	크기 (바이트)	설명
Header	1	데이터 전송 헤더
Size	4	Size 이후의 데이터 길이
Code	1	메시지 구분 코드
Oid	4	이동객체 ID
X	4	이동객체의 현재 위치 X 좌표
Y	4	이동객체의 현재 위치 Y 좌표
Time	3	현재 시간(시/분/초)
Validity	1	데이터 유용성 여부
Dummy	9	미결정 및 추가 데이터 입력 부분

표 3. 실시간 위치 데이터 패킷

```
7e 00 1d 11 15 41 08 2a 01 3d 11 06 02 b1 cb d5 11
20 38 00 00 00 41 00 00 00 00 00  00 00 00 00
7e 00 1d 11 15 41 04 11 01 26 de d5 02 af  25 f5 11
20 38 00 00 00 41 00 00 00 00 00  00 00 00 00
7e 00 1d 11 15 41 03 39 01 2d b5 d2 02 af 48 a1 11
20 38 00 00 00 41 00 00 00 00 00 00 00 00 00
```

네트워크를 통해 송수신되는 실시간 위치 데이터 패킷은 표 3과 같은 형태로 일정한 시간 주기에 따라 위치 제공 서버로부터 위치 데이터 관리 시스템에 전송된다.

4. LBS를 위한 위치 데이터 관리 시스템 설계 및 응용

LBS를 위한 이동객체의 위치 데이터 관리 시스템은 데이터베이스 관리 기술을 활용하여 이동객체의 위치변화 과정을 효율적으로 관리하고 이와 관련된 정보를 LBS 사용자에게 제공한다. 위치 데이터 관리 시스템은 그림 1과 같이 LBS 응용인터페이스, 응용서비스 질의처리기, 이동객체 위치추적기, 위치정보 관리기, 실시간 데이터 수신기, 위치정보 데이터베이스로 구성된다.

그림 1에서 LBS 응용인터페이스는 응용서비스와 관련된 질의를 입력하고 실행 결과를 제공받는다. 응용서비스 질의처리기는 응용인터페이스에서 입력된 질의 처리를 위해 질의 수행 계획을 수립하여 연산을 수행한 후 결과를 반환한다. 이동객체 위치추적기는 위치정보 데이터베이스에 직접 저장되지 않은 이동객체의 과거 및 미래 위치를 추정한다. 위치정보 관리기는 응용서비스 관리자가 위치정보 데이터베이스에 새로운 테이블을 생성하고, 일반 속성 정보를 관리할 수 있는 기능을 제공한다. 실시간 데이터 수신기는 이동객체의 실시간 위치정보를 수신하고 이를 위치정보 데이터베이스에 저장한다. 위치정보 데이터베이스는 이동객체의 속성 정보 및 실시간 위치정보가 저장된다.

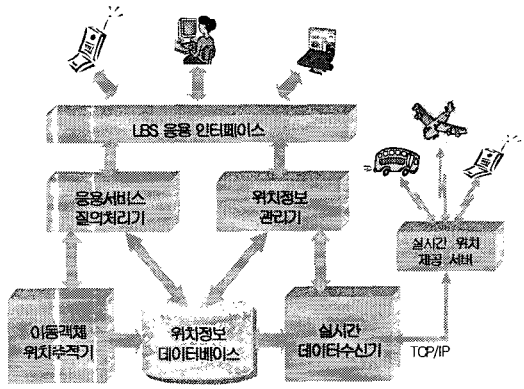

그림 1. LBS를 위한 위치 데이터 관리 시스템

4.1 위치정보 데이터베이스

위치정보를 저장하기 위해 관계형 데이터베이스 관리 시스템을 기반으로 하는 저장 구조를 사용한다. 첫 번째 릴레이션은 이동객체의 일반 속성 정보를 저장하는 MovingObject 릴레이션이다. MovingObject 릴레이션의 구조는 표 4와 같다. mo_id는 이동객체의 식별자로서 키 값이 된다. 기타 일반 속성 정보는 이동객체의 이름을 나타내는 name, 관리자를 나타내는 manager, 객체의 유형을 나타내는 type 등을 가질 수 있다. MovingObject 릴레이션의 속성 정보는 위치정보 관리기에서 관리자가 임의로 관리할 수 있는 정보이다. 그러나 마지막 속성인 tag는 반드시 입력되어야 하는 정보이다. tag는 이동객체의 이동궤적의 유형을 구분하는 구분자이다. tag의 값이 '1'이면 선형 이동 궤적, '2'이면 곡선 이동 궤적을 나타낸다.

두 번째 릴레이션은 이동객체의 이력 위치정보를 저장하기 위한 릴레이션으로 표 5와 같다. MovingHistory

표 4. MovingObject 릴레이션

mo_id	name	type	tag
char(10)	char(10)	char(10)	int

표 5. MovingHistory 릴레이션

mo_id	t_start	t_end	x_start	y_start	x_end	y_end	u_id
char (10)	char (20)	char (20)	float	float	float	float	char (10)

릴레이션에는 위치정보 데이터베이스에 저장된 과거 시점의 이력 위치정보가 저장된다. mo_id는 이동객체의 식별자, t_start는 유효 시간의 시작 시점, t_end는 유효 시간의 종료 시점, x_start와 y_start는 t_start 시점에서의 위치 좌표, x_end와 y_end는 t_end 시점에서의 위치 좌표를 의미한다. u_id는 외래 키로서 UncertainHistory 릴레이션으로부터 유효 시간 [t_start, t_end] 사이의 위치정보에 대한 불확실성 영역 값을 검색하는 데 사용된다.

세 번째 릴레이션은 MovingHistory 릴레이션에 저장된 이동객체의 이력 위치정보의 불확실성 영역 값을 저장하기 위한 UncertainHistory 릴레이션으로 표 6과 같다. u_id는 이 릴레이션의 키가 된다. center_x와 center_y는 불확실성 영역의 중심 좌표를 나타내고, radius는 (center_x, center_y)로부터의 반지름을 나타낸다. UncertainHistory 릴레이션에 저장된 정보는 사용자에게 불확실성 값의 범위를 제공하기 위해 보존된다.

표 6. UncertainHistory 릴레이션

u_id	center_x	center_y	radius
char(10)	float	float	float

4.2 응용서비스 질의처리기

응용서비스 질의처리기는 이동객체의 위치정보와 관련된 연산 처리를 통해서 LBS에서 사용자가 원하는 이동객체의 위치 값을 제공한다. 응용서비스 질의처리기의 동작과정은 그림 2와 같다.

이동객체의 위치정보 연산 처리를 위한 함수는 표 7과 같이 구성되며, 이 함수는 [18,19]에서 제시된 이동객체 연산들을 참조하여 정의한 것이다.

표 7의 입출력에서 mid , mid_A, mid_B는 이동객체의 식별자이고, $[t_s;t_e]$는 유효 시간의 간격이며 $t_s \le t_e$의 관계가 항상 성립한다. *real*은 실수 값, *line*은 선으로 표현되는 공간 속성, *location*은 이동객체의 $(x;y)$ 좌표

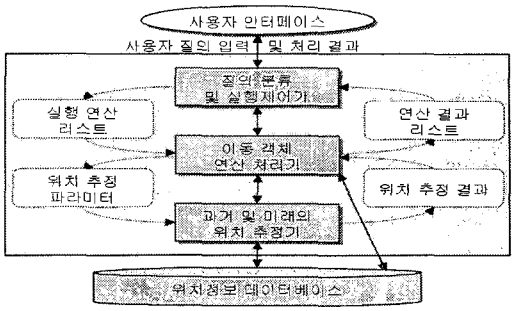

그림 2. 응용서비스 질의처리기 동작

표 7. 이동객체 연산 처리 함수

함수	입력	출력
mdistance	$mid_A \times mid_B \times [t_s, t_e]$	real
trajectory	$mid \times [t_s, t_e]$	line
length	$mid \times [t_s, t_e]$	real
velocity	$mid \times [t_s, t_e]$	real
attime	$mid \times [t_s, t_e]$	location
mnearest	$mid \times [t_s, t_e]$	location
mfarthest	$mid \times [t_s, t_e]$	location
minvalue	$mid \times [t_s, t_e]$	location
maxvalue	$mid \times [t_s, t_e]$	location
uncertainty	$mid \times [t_s, t_e]$	real

값이다. *mdistance*는 임의의 두 이동객체 mid_A와 mid_B 간의 유효 시간 간격 $T=[t_s;t_e]$ 동안의 거리를 계산한다. $distance_{t_i}$는 t_i 시점에서 mid_A와 mid_B의 거리이다. *trajectory*는 T 동안 mid 가 이동한 위치 좌표를 모두 검색하여 궤적을 표현한다. *length*는 *trajectory* 연산 결과로 생성된 이동 경로의 총 거리를 계산한다. *velocity*는 T에서 mid의 평균 이동 속도를 구한다. *attime*은 T 동안 각 시점 t_i에서 mid의 위치 좌표 값을 모두 검색한다. *mnearest*는 T 동안 mid와 가장 가까운 곳에 위치한 객체의 위치 좌표를 구한다. *mfarthest*는 *mnearest*와 반대이다. *minvalue*와 *maxvalue*는 T 동안 존재하는 이동객체의 모든 위치 좌표 중 최소 또는 최대가 되는 값을 추출한다. *uncertainty*는 T 동안 이동객체 mid의 이동 궤적에 관한 불확실성 영역 값을 제공한다. 각 연산자의 연산식은 다음과 같다.

- $mdistance = \bigcup_{i=1}^{n} distance_{t_i}$
- $distance_{t_i} = \sqrt{(By_{t_i} - Ay_{t_i})^2 + (Bx_{t_i} - Ax_{t_i})^2}$

- $trajectory = \bigcup_{i=1}^{n} Traj_{t_i}$
- $Traj_{t_i} = \frac{y_{t_{i-1}} - y_{t_i}}{x_{t_{i-1}} - x_{t_i}}(x - x_{t_{i-1}}) + y_{t_{i-1}}$
- $length = \sum_{i=1}^{n} distance_{t_i}$
- $velocity = \frac{length}{t_e - t_s}$
- $\%ime = \bigcup_{i=1}^{n} (x_{t_i}; y_{t_i}), t_i \in [t_s, t_e]$
- $mnearest = loc(\min\{mdistance(mid, candidate_i)\}_{i=1}^{n})$
- $mfarthest = loc(\max\{mdistance(mid, candidate_i)\}_{i=1}^{n})$
- $\min value = loc(\min\{(x_i, y_i)\}_{i=1}^{n})$
- $\max value = loc(\max\{(x_i, y_i)\}_{i=1}^{n})$
- $uncertainty = \bigcup_{i=1}^{n} area_{t_i}$
- $area_{t_i} = < center_{x_i}; center_{y_i}; radius_i >$

4.3 이동객체 위치추적기

관측지향 위치 데이터 관리 시스템에서 가장 많이 사용되는 위치 결정 도구는 위성 신호를 이용하는 GPS 단말기이다. 그러나 위성 신호는 지하에는 도달하지 못할 뿐만 아니라, 높은 빌딩 또는 밀집된 나무숲에 의해 방해를 받는다[21]. 이와 같은 지역에서는 GPS에 의해 관측된 위치 데이터를 제공받지 못하게 되고, 데이터베이스에 위치 데이터를 저장할 수 없는 문제점이 발생된다. 이 논문에서는 수신되지 않은 위치정보를 추정하고, 그 결과를 데이터베이스에 저장하는 위치정보 트리거를 사용한다.

그림 3은 위치정보 트리거를 포함하는 위치추적 모듈 및 관계를 나타낸 것이다.

가. 위치정보 트리거

위치추적 모듈은 4.2절의 응용서비스 질의처리기로부터 위치추정에 필요한 파라미터를 넘겨받은 후 궤적 분류기를 통해 선형 또는 곡선 궤적으로 분류된 후 그에 적합한 위치추정 연산이 수행된다. 그림 4는 위치정보 트리거의 처리과정을 표현한 것이다.

알고리즘 1의 LocationTrigger 모듈은 패킷 변환기로부터 생성된 위치정보인 ParsedPacket과 MovingObject 릴레이션에 저장된 tag 정보를 입력

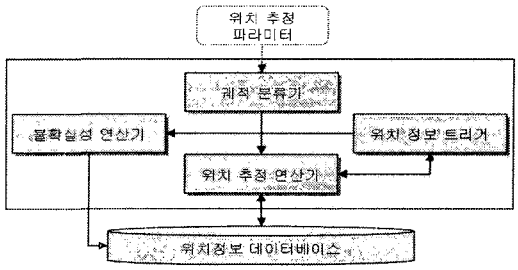

그림 3. 이동객체 위치추적 모듈의 구성

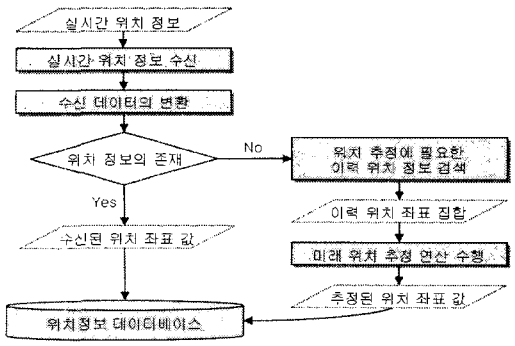

그림 4. 위치정보 트리거의 처리과정

받아 데이터베이스에 저장될 최종 위치정보가 저장된 *InputData*를 반환한다. 이 정보는 스키마 관리기의 입력 정보로 사용된다. *Schema Manager*는 *LocationTrigger*로부터 *InputData*를 입력받은 후 이 정보를 해당되는 이력 릴레이션에 추가시킨다.

알고리즘 1. LBS를 위한 위치정보 트리거링

```
Algorithm LocationTrigger(ParsedPacket, Tag)
 Input: ParsedPacket(파싱된 패킷 정보);
        Tag(이동 궤적 유형);
 Output: InputData(DB의 최종 위치정보 저장배열);
 Begin
  InputData ← ParsedPacket;
  mo_id ← ParsedPacket[0];
  temp_x ← ParsedPacket[1];
  temp_y ← ParsedPacket[2];
  t_now ← ParsedPacket[3];
  If (temp_x == 'null' or temp_y == 'null') Then
    If (Tag == 1) Then
      future_location ← FutureLinear(mo_id, t_now);
      InputData[1] ← future_location[0];
      InputData[2] ← future_location[1];
    If (Tag == 2) Then
      future_location ← FutureSpline(mo_id, t_now);
      InputData[1] ← future_location[0];
      InputData[2] ← future_location[1];
    Else
```

```
    InputData[1] ← temp_x; InputData[2] ← temp_y;
  Return InputData;
 End
Algorithm SchemaManager(InputData)
 Input: InputData(MovingHistory에 저장될 객체 이동정보)
 Begin
  mo_id ← InputData[0];
  table ← MovingObject에서 mo_id가 저장된 테이블 검색;
  target ← "MovingHistory_"+table;
  time ← InputData[3];
  x ← InputData[1]; y ← InputData[2];
  target 릴레이션에 새로운 위치정보를 추가;
 End
```

나. 위치추정 연산 알고리즘

위치정보 데이터베이스에 저장되는 이동객체의 위치추정을 위해 이동 궤적에 관련된 필드를 두어 2가지로 구분하였다. 이동 궤적에 관한 정보 입력은 위치정보 관리기를 통해 초기 릴레이션을 생성할 때 관리자가 선택적으로 입력할 수 있도록 하였다. 선형 궤적은 직선 함수를 이용하게 되고, 곡선 궤적은 3차 스플라인 보간 다항식을 이용한다.

알고리즘 2. 선형 및 곡선 위치추정 알고리즘

```
Algorithm FutureLinear(mo_id, t_f)
 Input: mo_id(객체 식별자); t_f(미래의 특정 시점)
 Output: future_location(시점 t_f에서의 (x, y) 좌표 값);
 Begin
  MovingObject에서 입력된 mo_id를 가지는 객체 검색;
  If (검색 결과가 null이 아니면) Then
   t_pre ← MovingHistory에 저장된 (마지막 시점-1)값;
   x_pre ← t_pre 시점의 x 좌표 값;
   y_pre ← t_pre 시점의 y 좌표 값;
   t_last ← MovingHistory에 저장된 마지막 시점 값;
   x_last ← t_last 시점의 x 좌표 값;
   y_last ← t_last 시점의 y 좌표 값;
   x_t_f ← (x_last-x_pre)/(t_last-t_pre)*(t_f-t_last)+x_last;
   y_t_f ← (y_last-y_pre)/(t_last-t_pre)*(t_f-t_last)+y_last;
  Else
   x_t_f ← error;  y_t_f ← error;
   future_location[0] ← x_t_f; future_location[1] ← y_t_f;
  Return future_location;//시점 t_f에서 (x, y) 값 반환
 End
Algorithm PastLinear(mo_id, t_p)
 Input: mo_id(객체 식별자); t_p(과거의 특정 시점)
 Output: past_location(시점 t_p에서의 (x, y) 좌표 값);
 Begin
  MovingObject에서 mo_id를 가지는 객체 검색;
  If (검색 결과가 null이 아니면) Then
   t_before ← t_p 바로 이전 시점 값;
   x_before ← t_before 시점의 x 좌표 값;
   y_before ← t_before 시점의 y 좌표 값;
   t_after ← t_p 바로 이후 시점 값;
   x_after ← t_after 시점의 x 좌표 값;
   y_after ← t_after 시점의 y 좌표 값;
   x_t_p ← (x_after-x_before)/
           (t_after-t_before) *(t_p-t_before)+x_after;
   y_t_p ← (y_after-y_before)/
           (t_after-t_before)*(t_p-t_before)+y_after;
  Else
   x_t_p ← error;  y_t_p ← error;
   past_location[0] ← x_t_p; past_location[1] ← y_t_p;
  Return past_location; //t_p에서 (x,y) 값 반환
 End
Algorithm FutureSpline(mo_id, t_f)
 Input: mo_id(객체 식별자); t_f(미래의 임의의 시점)
 Output: future_location(시점 t_f에서의 (x, y) 좌표 값)
 Begin
  MovingObject에서 mo_id를 가지는 객체 검색
  If (검색 결과가 null이 아니면) Then
   t[0]← t_{n-3}; t[1]← t_{n-2}; t[2]← t_{n-1}; t[3]← t_n;
   x[0]← x_{n-3}; x[1]← x_{n-2}; x[2]← x_{n-1}; x[3]← x_n;
   y[0]← y_{n-3}; y[1]← y_{n-2}; y[2]← y_{n-1}; y[3]← y_n;
   x_t_f ← FutureSplineInterpolation(t, x, t_f);
   y_t_f ← FutureSplineInterpolation(t, y, t_f);
  Else
   x_t_f ← error;  y_t_f ← error;
   future_location[0] ← x_t_f; future_location[1] ← y_t_f;
  Return future_location;//시점 t_f에서의 (x, y) 값 반환
 End
Algorithm PastSpline(mo_id, t_p)
 Input: mo_id(객체 식별자); t_p(과거의 임의의 시점)
 Output: past_location(시점 t_p에서의 (x, y) 좌표 값);
 Begin
  MovingObject에서 mo_id를 가지는 객체 검색;
  If (검색 결과가 null이 아니면) Then
   t[0]← t_{i-1}; t[1]← t_i; t[2]← t_{i+1}; t[3]← t_{i+2};
   x[0]← x_{i-1}; x[1]← x_i; x[2]← x_{i+1}; x[3]← x_{i+2};
   y[0]← y_{i-1}; y[1]← y_i; y[2]← y_{i+1}; y[3]← y_{i+2};
   x_t_p ← PastSplineInterpolation(t, x, t_p);
   y_t_p ← PastSplineInterpolation(t, y, t_p);
  Else
   x_t_p ← error;  y_t_p ← error;
   past_location[0] ← x_t_p;  past_location[1] ← y_t_p;
  Return past_location;//시점 t_p에서의 (x, y) 좌표 값 반환
 End
```

알고리즘 2에서 선형 이동 궤적의 위치추정은 1차 선형 함수를 이용하여 FutureLinear와 PastLinear 함수로 처리한다. 곡선 이동 궤적의 위치추정은 3차 스플라인 보간법을 이용하며 *FutureSpline*과 *PastSpline* 함수로 처리한다.

다. 불확실성 영역 처리

제안 시스템에서는 다음과 같은 방법으로 위치추

정 결과에 대한 불확실성 값의 범위를 정량화하여 사용자의 질의 결과로서 제공한다. 스키마 관리기가 실행될 때 위치추적기의 불확실성 연산 모듈이 함께 수행된다. 이 불확실성 연산 모듈을 통해 이력 위치 정보의 불확실성 영역 값이 생성된다. 그림 5는 이동 객체의 궤적에서 발생되는 불확실성 영역을 그림으로 표현한 것이다.

그림 5에서 유효시간 [t_start, t_end]에서 이동객체의 위치 좌표가 (x_start, y_start, x_end, y_end)일 때, 불확실성 범위의 중심 좌표는 (center_x, center_y) 가 되고, 이 때 (center_x, center_y)로부터 하나의 원을 이루기 위한 반지름은 r이 된다. 이력 위치 데이터의 불확실성 값 생성 과정은 알고리즘 3과 같다. Uncertain History는 MovingHistory에 저장된 각 튜플의 불확실성 영역 값을 저장하며, UncertainHistory에 저장되는 정보는 실시간 위치정보가 MovingHistory에 저장되는 이력 위치정보의 수만큼 생성된다.

알고리즘 3. 이력 위치정보의 불확실성 연산

```
Algorithm UncertaintyCreator(HistoryTuple,
                             u_id, table)
 Input: HistoryTuple(가장 최근의 이력 튜플);
        u_id(불확실성 값의 식별자); table(이력 릴레이션명);
 Begin
  x_start ← HistoryTuple[0];  y_start ← HistoryTuple[1];
  x_end ← HistoryTuple[2];  y_end ← HistoryTuple[3];
  table ← 데이터베이스에 저장된 MovingObject 릴레이션
          에서 mo_id를 가지는 릴레이션 이름 검색;
  center_x ← x_start + (x_end - x_start)/2;
  center_y ← y_start + (y_end - y_start)/2;
  radius ← SQRT((x_end-x_start)^2+(y_end-y_start)^2)/2;
  target ← "UncertainHistory_"+table;
  target 릴레이션에 새로운 불확실성 값 추가;
 End
```

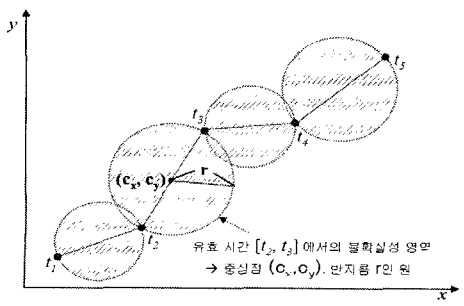

그림 5. 이동 궤적의 불확실성 영역

4.4 위치정보 관리기

위치정보 데이터베이스의 릴레이션 생성 및 속성 정보 입력은 관리자 메뉴를 통해 처리된다. 관리자 메뉴를 통해 새로운 테이블 이름을 입력하면 데이터베이스에 새 테이블을 추가로 생성하여 관리할 수 있다.

그림 6은 위치정보 관리기의 동작과정을 나타낸 것이다. 예를 들어, 테이블 이름으로 'Fleet'을 입력하면 'MovingObjectFleet', 'MovingHistoryFleet', 'UncertainHistoryFleet'의 세 개의 새로운 테이블이 생성된다. 새롭게 생성된 테이블에 저장될 이동객체의 일반 속성 정보는 관리자 메뉴의 속성 정보 입력을 통해서 저장된다. 관리자 메뉴를 이용한 테이블 생성 및 속성 정보 입력 과정은 다음의 알고리즘 4와 같다.

알고리즘 4. 테이블 생성 및 속성 정보 입력

```
AlgorithmTableCreator(Tname, AttrList, Tag)
 Input: Tname(생성 테이블 이름);
        AttrList(객체 속성정보 배열); Tag(이동궤적 유형);
 Begin
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  Con ← DriverManager.getConnection(url, id, pswd);
  Stmt ← Con.createStatement();
  Query ← "CREATE TABLE MovingObject_"+Tname+
          "(mo_id char(10), name char(10), manager
           char(10), type char(10), tag int)";
  Stmt.executeQuery(Query);
  Query ← "CREATE TABLE MovingHistory_"+Tname+
          "(mo_id char(10), t_start char(20), t_end
           char(20), x_start float, y_start float, x_end
           float, y_end float, u_id char(10))";
  Stmt.executeQuery(Query);
  Query ← "CREATE TABLE UncertainHistory_"+Tname
          + "(u_id char(10), center_x float, center_y
           float, radius float)";
  Stmt.executeQuery(Query);
  Query ← "INSERT INTO MovingObject_"+Tname
       +"(mo_id, name, manager, type) VALUES (?,?,?,?)";
  Pstmt ← Con.prepareStatement(Query);
  Pstmt.setString(1, AttrList[0]);
  Pstmt.setString(2, AttrList[1]);
  Pstmt.setString(3, AttrList[2]);
  Pstmt.setString(4, AttrList[3]);
  Pstmt.executeUpdate();
  Stmt.close();  Pstmt.close();  Con.close();
 End
```

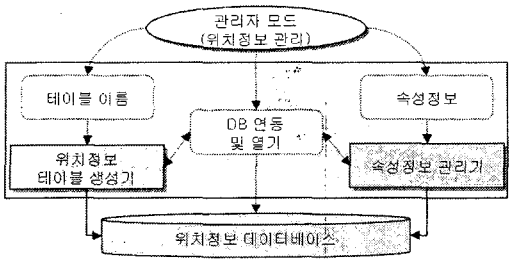

그림 6. 위치정보 관리기의 동작과정

알고리즘 4의 TableCreator 모듈은 관리자가 입력한 테이블 이름과 속성 정보를 입력으로 받아 데이터베이스에 새로운 테이블을 생성하고 속성 정보를 추가한다.

4.5 실시간 데이터 수신기

이동객체의 위치 데이터는 외부 위치 제공 서버로부터 데이터 로더를 통해 수신된다. 데이터 로더에서는 실시간으로 전송되는 이동객체의 위치 데이터를 TCP/IP를 이용하여 패킷 형태로 읽어 들인다. 데이터 로더를 통해 수신된 위치 데이터 패킷은 16진수 형태로 구성되어 있다. 수신된 값은 패킷 변환기를 통해 텍스트 형태의 문서로 변환된 후 스키마 관리기를 거쳐서 위치정보 데이터베이스에 저장된다. 그림 7은 실시간 데이터 수신기의 동작과정이다.

실시간 위치 데이터 패킷의 처리 과정은 다음 알고리즘 5와 같다.

알고리즘 5. 실시간 위치 데이터 패킷 수신 및 변환

```
Algorithm DataLoader(ServerIp)
 Input: ServerIp(실시간 위치 제공 서버의 주소)
 Output: LoadPacket(수신된 위치정보 패킷 배열);
 Begin
  socket ← new Socket(ServerIp, 포트번호);
  byte[] LoadPacket ← new byte[32];
  in←new BufferedInputStream(socket.getInputStream());
  For i = 0, 31
    ReadOneByte = (byte) in.read();
    LoadPacket[i] = ReadOneByte;
  Return LoadPacket;
 End
Algorithm PacketTranslator(LoadPacket)
 Input: LoadPacket(바이트 형태의 패킷 데이터 배열)
 Output: ParsedPacket(DB 저장 형태로 변환된 패킷정보)
 Begin
  mo_id ← ((LoadPacket[4]*255+LoadPacket[5])*255+
          LoadPacket[6])*255+LoadPacket[7];
  x ← ((LoadPacket[8]*255+LoadPacket[9])*255+
       LoadPacket[10])*255+LoadPacket[11]/100;
  y ← ((LoadPacket[12]*255+LoadPacket[13])*255+
       LoadPacket[14])*255+LoadPacket[15]/100;
  time ← ""+LoadPacket[16]+"-"+LoadPacket[17]+"-"+
         LoadPacket[18];
  ParsedPacket[0] ← mo_id; ParsedPacket[1] ← x;
  ParsedPacket[2] ← y; ParsedPacket[3] ← time;
 Return ParsedPacket ;
End
```

알고리즘 5의 DataLoader는 실시간 위치 제공 서버의 URL을 입력 값으로 받은 후 수신된 위치 데이터 패킷 배열을 출력 값으로 제공한다. *PacketTranslator*는 데이터 로더를 통해 수신된 위치 데이터 패킷 중에서 위치정보 데이터베이스에 실제 저장될 *mo_id*, *x*, *y*, *time* 부분의 값만을 추출하여, 16진수 형태로 입력된 각각의 값들을 10진수 형태로 변환한 후 해당되는 데이터 타입으로 변경하여 *ParsedPacket*에 저장한 후 반환한다.

4.6 LBS 응용 인터페이스

제안한 위치 데이터 관리시스템의 적용을 위해 산불관리 응용서비스에 적용하였다. 적용한 응용 프로그램은 Windows2000 운영체제에서 JDK1.4와 MS-SQL Server를 이용하여 구현하였다. 실험 데이터는 2002년 2월 28일 10시부터 2002년 2월 28일 13시 20분까지의 유효시간 구간 동안에 2분과 20분을 주기로 소방차 3대와 하나의 화재 지역을 대상으로 샘플링이 이루어졌다고 가정하고 임의로 생성하였다.

이동객체의 위치정보 연산이 산불 관리 응용에서 적용된 사례를 질의 수행 화면을 통해서 결과로서 제시한다. 첫 번째 질의는 "소방차 NFireT113이 목

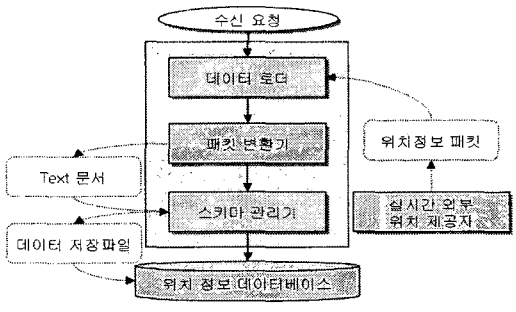

그림 7. 실시간 데이터 수신기의 처리과정

적지에 도달한 2002년 2월 28일 13시 21분에 다른 소방차 ChFireT102와의 거리는 얼마인가?"이다. 이 질의를 수행한 화면은 그림 8과 같다. 이것은 두 이동 객체간의 거리 값을 반환하는 mdistance 연산자가 사용된 예제이다. mdistance 연산자는 질의 처리 내용이 질의 입력창을 통해 입력되면 먼저 attime 연산자를 통해 질의 시점에서 두 질의 객체의 공간 정보를 획득한다. 질의 처리에 사용되는 연산은 4.2절의 표 7에서 설명한 이동객체 연산 처리 함수의 연산식에 의해 처리된다. 획득된 공간 정보에 거리 연산을 적용함으로써 두 질의 객체간의 거리 값을 구한다. 그림 8은 질의 시점 값으로 2002년 2월 28일 13시 21분을 그리고 질의 객체로 NFireT113과 ChFireT102 값을 입력하여 mdistance 연산을 수행한 결과이다.

두 번째 질의는 "소방차 ChFireT102의 전체 이동 거리는 얼마인가?"이다. 이 질의를 수행하기 위해서는 질의 시간 구간 동안 이동객체의 이동 거리 값을 반환하는 length 연산자가 필요하다.

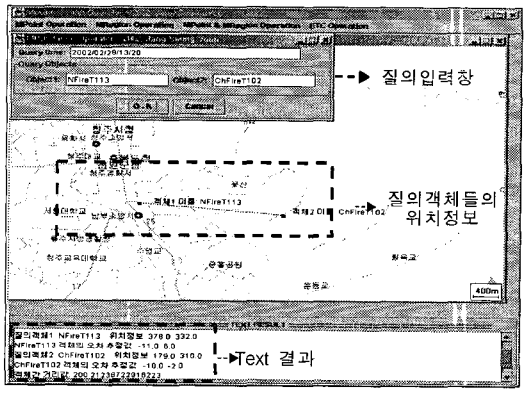

그림 8. 이동객체간의 거리 연산

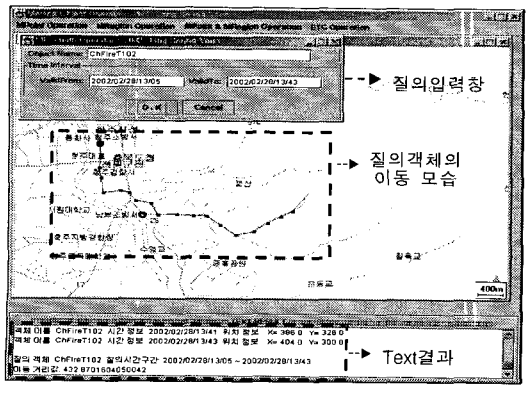

그림 9. 이동객체의 이동거리 연산

그림 9는 질의 객체 ChFireT102와 전체 이동 시간 구간 값 2002년 2월 28일 13시 5분에서 2002년 2월 28일 13시 43분을 입력하여 length 연산을 수행한 것이다. 화면 결과는 질의 객체가 질의 시간 구간 동안 이동한 모습을 보여준다. 연산 수행 결과인 이동 거리 값은 텍스트로 출력된다.

세 번째 질의는 "3분후에 소방차 ChFire107이 도달할 것이라 기대되는 위치는 어디인가?"이다. 현재 시간은 2002년 2월 28일 10시 20분으로 가정한다. 이 질의를 수행하기 위해서는 과거, 현재, 가까운 미래 시점에서 객체의 위치정보를 반환해 주는 attime 연산자가 사용된다.

그림 10은 3분 후의 질의 시점 2002년 2월 28일 10시 23분 값과 질의 객체 이름 ChFire107를 입력값으로 받아 attime 연산자를 수행한 것이다. 화면 결과는 3분후에 ChFire107이 도달할 것이라 기대되는 위치를 보여주며, 좌표 정보는 텍스트 결과 창에 출력된다. 이때 추출된 결과 값은 미래 위치추정 함수를 통해 계산 된 것이다.

네 번째 질의는 "Fire135의 현재 화재 상태를 보여라."이다. 현재 시간은 2002년 2월 28일 13시 50분으로 가정한다. 이 질의를 수행하기 위해서 질의 시점에서의 객체 위치정보를 반환하는 attime 연산자가 사용된다.

그림 11은 질의 시점 2002년 2월 28일 13시 50분 값과 이동 영역 객체 Fire135값을 입력 값으로 받아서 attime 연산자를 수행한 것이다. 이때 추출된 결과는 이동 영역의 모양 변화 처리 및 위치추정을 통해 연산 된다. 이 질의에 대한 연산 결과 해당 시점에서의 폴리곤 정보 즉, 객체의 위치정보가 화면과 텍스트 결과 창에 출력된다.

4.7 제안 시스템의 특성

이 논문에서 제안한 위치 데이터 관리 시스템과 관련 시스템들과의 특성을 비교 분석하였다. 비교 항목의 기준은 위치 데이터 모델과 위치추정 모델로 선정하였다. 먼저 위치 데이터 모델의 특성을 비교하기 위해 각 시스템에서 사용한 기본 데이터 모델의 유효 시간에 따른 위치정보 처리 능력 및 질의 처리 기능을 과거 위치정보 처리, 미래 위치정보 처리, 과거 질의 연산, 미래 질의 연산으로 구분하여 비교하였으며 그 결과는 표 8과 같다.

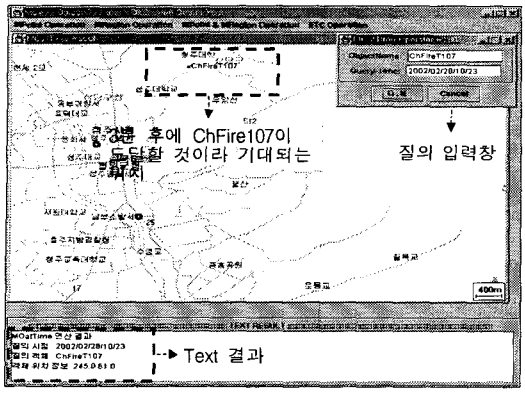

그림 10. 이동객체의 위치추정 연산

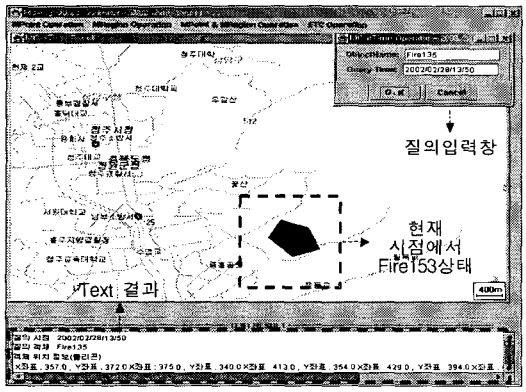

그림 11. 이동 영역에 대한 위치추정 연산

표 8. 위치 데이터 모델의 특성 비교

평가 항목 \ 비교 시스템	DOMINO	CHOROS	DEDALE	STRBA	MOMS	OUR
과거 위치정보	×	○	○	○	○	○
미래 위치정보	○	×	×	△	△	○
과거 질의연산	×	○	○	○	○	○
미래 질의연산	○	×	×	△	△	○

지원함(○), 지원하지 못함(×), 방법론만 제시됨(△)

표 8에서와 같이 각 시스템에서 지원하는 데이터 모델의 특성을 검토하면 먼저 DOMINO [5,6]에서는 미래 위치정보 처리 기능과 미래 질의 처리 연산을 제공한다. 그러나 과거 위치정보 처리 기능과 과거 질의 처리를 위한 연산은 제시하지 않고 있다. CHOROS[18-20]에서는 DOMINO와 정 반대로 과거 시간에 대한 위치정보 처리 및 연산 기능만을 제공하며, DEDALE[11,12]에서도 역시 CHOROS와 마찬가지로 과거 정보에 대한 처리 기능만을 제공한다. STRBA[13,14]와 MOMS[17]에서는 과거 정보에 대한 처리 기능을 제공하며, 미래 정보에 대한 처리 기능은 방법론만 언급하고 있다. 이 논문에서 제안한 데이터 모델에서는 과거 위치정보 처리, 미래 위치정보 처리, 과거 질의 연산, 미래 질의 연산 기능을 모두 제공하는 특징을 가진다.

두 번째로 각 시스템의 위치추정 모델의 특성을 비교하기 위해 과거 위치추정, 미래 위치추정, 실시간 불확실성 처리, 불확실성 값의 정량화 기능의 지원여부를 비교하였으며 그 결과는 표 9와 같다.

표 9의 위치추정 기능을 검토하면 먼저 DOMINO [5,6]에서는 미래 위치추정과 실시간 불확실성 처리 기능을 지원하며, 불확실성 값의 정량화 방안을 제시하고 있다. CHOROS[18-20]에서는 과거 위치추정 기능을 지원하고, 곡선 위치추정 및 불확실성 값의 정량화 방법을 제시하고 있다. DEDALE[11,12]은 과거 위치추정 기능만을 지원한다. STRBA[13,14]는 과거 위치추정 및 미래 위치추정 기능을 지원하며, MOMS[17]는 과거 위치추정 기능만을 지원한다.

이 논문에서 제안한 위치추정 모델에서는 과거 위치추정, 미래 위치추정, 실시간 불확실성 처리, 불확실성 값의 정량화 기능을 모두 제공한다. 기존에 제안된 모바일 객체 관리 시스템과 이 논문에서 제안한 시스템을 비교 분석한 결과 제안 시스템에서 지원하는 데이터 모델 및 위치추정 모델이 다른 시스템에서 지원하지 못하는 기능을 추가적으로 제공함을 알 수 있다.

표 9. 위치정보 추정 기능의 비교

평가 항목 \ 비교 시스템	DOMINO	CHOROS	DEDALE	STRBA	MOMS	OUR
과거 위치추정	×	○	○	○	○	○
미래 위치추정	○	×	×	○	×	○
실시간 불확실성처리	○	×	×	×	×	○
불확실성값의 정량화	△	△	×	×	×	○

지원함(○), 지원하지 못함(×), 방법론만 제시됨(△)

5. 결 론

LBS를 가능하게 하는 주요 기술 구성요소에는 이동통신망에서 위치를 파악하는 무선측위기술 또는 위치결정기술, 위치 데이터 관리를 위한 LBS 핵심기

반기술 및 LBS 응용기반기술 등이 있다. 이 논문에서는 LBS의 주요기술 구성요소 중 핵심기반기술인 위치 데이터 관리 시스템을 설계하고 이를 응용서비스에 적용한 사례를 제시하였다. 설계한 시스템은 LBS 응용인터페이스, 응용서비스 질의처리기, 이동객체 위치추적기, 위치정보 관리기, 실시간 데이터 수신기, 위치정보 데이터베이스로 구성된다. 이 시스템은 데이터베이스 관리 기술을 활용하여 이동객체의 위치변화 정보를 효율적으로 관리하고 이와 관련된 유용한 정보를 LBS 사용자에게 제공하며, 기존의 상용 데이터베이스에서 처리할 수 없는 이동객체의 연속적인 위치정보를 처리하는 연산 및 위치추정 기능을 지원한다. 아울러 실시간 환경에서 발생되는 위치정보의 결손을 보완하는 위치정보 트리거링 기법을 지원하도록 하였다. 제안한 위치 데이터 관리 시스템의 적용을 위해 산불관리 응용서비스에 적용한 LBS 응용인터페이스 및 질의처리 결과를 보임으로서 위치 데이터 관리의 실증적인 사례를 제시하였다.

제안한 위치 데이터 관리 시스템은 다른 시스템에서 통합적으로 지원하지 못하는 과거 위치추정, 미래 위치추정, 실시간 불확실성 처리, 불확실성 값의 정량화 기능을 모두 제공하는 특성을 가진다. 아직 관련 시스템들의 이동객체 처리 결과에 대한 신뢰성에 대한 연구가 진행되지 않고 있다. 향후에는 이 논문에서 제안한 기능에 대한 정확성 검증을 통해 보다 정교한 데이터 관리가 가능하도록 할 것이다. 아울러 이 논문에서 설계한 시스템 구조를 좀 더 확장하여 화물 운송 정보 관리, 네비게이션, 물류관제 시스템 등의 응용에서 실제 데이터를 활용한 실험이 이루어질것이다.

참 고 문 헌

[1] 최재경, 정지영, 윤원정, "LBS(위치기반서비스)관련 기술 및 시장동향," *주간기술동향 제1067호*, 한국전자통신연구소, pp. 1-14, 2002. 10.

[2] 문형돈, "LBS 기술 및 시장동향," *주간기술동향 제1080호*, 한국전자통신연구소, pp. 25-36, 2003. 1.

[3] 박용우, "위치기반서비스(Location Based Service)의 기술동향 및 활성화 전망," 정보통신정책연구원, *KISDI IT FOCUS*, pp. 79-83, 2001. 7.

[4] O. Wolfson, "Moving Objects Databases: Issues and Possible Solutions," Keynote Address, *Proceedings of the Mobile Data Management*, 2001.

[5] P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects," *Proceedings of the 13th International Conference on Data Engineering (ICDE13)*, 1997.

[6] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, "Moving Objects Databases: Issues and Solutions," *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pp. 111-122, 1998.

[7] O. Wolfson, P. Sistla, B. Xu, J. Zhou, S. Chamberlain, N. Rishe, and Y. Yesha, "Tracking Moving Objects Using Database Technology in DOMINO," *Proceedings of the 4th Workshop on Next Generation Information Technologies and Systems*, pp. 112-119, 1999.

[8] S. Dieker and R. H. Guting, "Plug and Play with Query Algebras: SECONDO A Generic DBMS Development Environment," *Proceedings of the International Databases Engineering and Applications Symposium*, pp. 380-390, 2000.

[9] J. A. C. Lema, L. Forlizzi, R. H. Guting, E. Nardelli, and M. Schneider, "Algorithms for Moving Objects Databases," Fern University, Hagen, Informatik-Report 289, 2001.

[10] R. H. Guting, S. Dieker, C. Freundorfer, L. Becker, and H. Schenk, "Secondo/QP: Implementation of a Generic Query Processor," *Proceedings of the 10th Internatioal Conference on Database and Expert System Applications*, pp. 66-87, 1999.

[11] S. Grumbach, P. Rigaux, and L. Segoufin, "Spatio-Temporal Data Handling with Constraints," *ACM GIS*, 1998.

[12] S. Grumbach, P. Rigaux, M. Scholl, and L. Segoufin, "The Design and Implementation of DEDALE," 1999.

[13] S. S. Park, Y. A. Ahn, and K. H. Ryu, "Moving Objects Spatiotemporal Reasoning Model for Battlefield Analysis," *Proceedings of Military, Government and Aerospace Simulation part of ASTC2001*, pp. 108-113, 2001.

[14] K. H. Ryu and Y. A. Ahn, "Application of Moving Objects and Spatiotemporal Reasoning," *A TimeCenter Technical Report*, TR-58, 2001.

[15] 안윤애, 조동래, 류근호, "전장분석을 위한 이동 객체의 위치 예측 시스템," *정보과학회논문지*, 제8권 제6호, pp. 765-777, 2002.

[16] 안윤애, "Moving Object Management System for Battlefield Simulation," *한국데이터정보과학회지*, 제15권 3호, pp. 663-675, 2004.

[17] D. H. Kim, J. S. Kim, Y. A. Ahn, and K. H. Ryu, "Moving Objects Relational Model and Design for e-Logistics Applications," *Proceedings of International Conference*, ICITA2002, 2002.

[18] M. Erwig, R. H. Guting, M. Schneider, and M. Vazirgiannis, "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases," *GeoInformatica* Vol. 3, No. 3, pp. 269-296, 1999.

[19] L. Forlizzi, R. H. Guting, E. Nardelli, and M. Schneider, "A Data Model and Data Structures for Moving Objects Databases," *Proceedings of the ACM SIGMOD Conference*, pp. 319-330, 2000.

[20] R. H. Guting and et. al, "A Foundation for Representing and Querying Moving Objects," *ACM Transactions on Database Systems*, Vol. 25, No. 1, pp. 1-42, 2000.

[21] R. Casey, "Automatic Vehicle Location Successful Transit Applications," A Cross-Cutting Study, in Intelligent Transportation Systems Electronic Document Library, Document No. 11487, U.S DOT Publication No. FHWA-JPO-99-022, 2000.

안 윤 애

1993년 2월 한남대학교 전자계산공학과 공학사
1996년 2월 충북대학교 전자계산학과 이학석사
2003년 2월 충북대학교 전자계산학과 이학박사
2003년 3월~2006년 2월 청주과학대학 컴퓨터과학과 조교수
2006년 3월~현재 충주대학교 전기전자 및 정보공학부 조교수
관심분야 : 모바일 S/W, LBS, 시공간 데이터베이스, 임베디드 응용 등