모바일 인터페이스를 이용한 차량 위치 추적 시스템 설계

오 준 석[†]·안 윤 애[†]·장 승 연[†]·이 봉 규[††]·류 근 호[†††]

요 약

무선 컴퓨팅 기술 및 이동 객체의 위치를 정확하게 추적할 수 있는 GPS 기술의 발달로 인하여 물류 차량 관리, 항공 교통 통제, 위치 기반 서비스 등과 같은 실시간 환경의 위치 정보 응용 시스템의 개발이 활발해지고 있다. 특히, 차량의 위치를 관제 센터에서 실시간으로 파악하는 차량 위치 추적 시스템에 관한 연구가 대표적인 응용 시스템으로 등장하였다. 그런데 기존의 차량 위치 추적 시스템은 데이터베이스에 저장되지 않은 특정 시간의 차량 위치 정보를 사용자에게 제공하지 못하는 문제점을 갖는다. 따라서 이 논문에서는 PDA와 같은 모바일 인터페이스를 통하여 실시간으로 차량의 위치 추적이 가능한 시스템을 설계한다. 제안 시스템은 차량 위치 검색 서버와 모바일 인터페이스로 구성되며, 이동 차량의 현재 위치뿐만 아니라 데이터베이스에 저장되지 않은 과거 및 미래 위치 정보까지 사용자에게 제공하는 장점을 갖는다.

Design of Vehicle Location Tracking System using Mobile Interface

Jun Seok Oh[†]·Yoon Ae Ahn[†]·Seung Youn Jang[†]
Bong Gyou Lee[††]·Keun Ho Ryu[†††]

ABSTRACT

Recent development in wireless computing and GPS technology cause the active development in the application system of location information in real-time environment such as transportation vehicle management, air traffic control and location based system. Especially, study about vehicle location tracking system, which monitors the vehicle's position in a control center, is appeared to be a representative application system. However, the current vehicle location tracking system can not provide vehicle position information that is not stored in a database at a specific time to users. We designed a vehicle location tracking system that could track vehicle location using mobile interface such as PDA. The proposed system consist of a vehicle location retrieving server and a mobile interface. It is provide not only the moving vehicle's current location but also the position at a past and future time which is not stored in database for users.

키워드 : 차량 추적 시스템(Vehicle Tracking System), 모바일 인터페이스(Mobile Interface), 이동 객체(Moving Object), 실시간 위치 추적(Real-time Location Tracking)

1. 서 론

무선 네트워크 기술과 모바일 장비의 발달은 기존의 정보 시스템들을 무선 통신 환경으로 확장이 가능하도록 하였다. 이러한 무선 환경과 GPS 기술의 발달로 인하여 실시간 위치 정보를 이용한 시스템 개발이 활발하게 진행되고 있으며 대표적인 예로 물류 차량 관리, 항공 교통 통제, 위치 기반 서비스 등을 들 수 있다. 특히 차량의 위치를 관제 센터에서 실시간으로 모니터링하는 차량 위치 추적 시스템[1-5]을 대표적인 응용 시스템으로 들 수 있다. 그러나 기존의 차량 위치 추적 시스템들은 주로 유선 통신망에서 차량의 위치 검색이 가능하고 기존의 상용 DBMS를 그대로 사용하기 때문에 이동 객체[6-10]의 특징이라 할 수 있는 이동 차량의 과거 및 미래의 불확실한 위치 정보를 제공하지 못하고 있다.

이 논문에서는 무선 네트워크 상에서 모바일 인터페이스를 이용하여 실시간으로 차량 위치 검색이 가능한 차량 위치 추적 시스템을 구현하였다. 구현 시스템은 차량의 위치 정보를 저장 및 관리하고 모바일 인터페이스에 차량 위치 검색 결과를 제공하는 차량 위치 추적 시스템 서버와 차량

※ 이 연구는 과학재단 RRC(청주대 정보통신 연구센터)의 지원으로 수행되었음.
† 준 회 원 : 충북대학교 대학원 전자계산학과
†† 정 회 원 : 한성대학교 정보전산학부 교수
††† 종신회원 : 충북대학교 전기전자및컴퓨터공학부 교수
논문접수 : 2002년 9월 30일, 심사완료 : 2002년 11월 14일

위치 관련 정보를 서버에 요청하고 서버로부터 제공된 위치 검색 결과를 출력하는 모바일 인터페이스로 구성된다. 먼저, 차량 위치 추적 시스템 서버 구현을 위하여 차량 위치 추적 시스템에 적용될 차량 위치 정보를 모델링 한다. 그리고 차량 위치 추적 시스템의 구조와 구성요소별 기능 및 처리 알고리즘을 제시한다. 특히 데이터베이스에 저장된 이력 위치 정보를 토대로 선형 보간법을 이용한 과거 및 미래의 위치 추정 방법을 제시한다. 또한 모바일 인터페이스를 위하여 서버 연결 모듈과 데이터 변환 및 송수신 모듈을 설계 구현함으로써 무선 환경에서 실시간으로 차량 위치 검색이 가능하도록 하였다. 구현된 시스템은 데이터베이스에 저장되지 않은 과거 및 미래의 위치 정보를 추정하고 모바일 인터페이스를 이용하여 데이터베이스에 저장된 차량의 위치와 관련된 질의를 처리한다.

제안 시스템은 '모든 차량의 전체 시간구간 위치 검색 질의', '모든 차량의 특정 시간구간 위치 검색 질의', '특정 차량의 전체 시간구간 위치 검색 질의', '특정 차량의 특정 시간구간 위치 검색 질의', '특정 시점에서의 차량 위치 검색 질의'등과 같은 차량 위치 검색 기능을 수행한다. 아울러 시스템 구현을 통하여 이와 같은 차량 위치 검색 기능이 적절히 수행됨을 확인하였다.

이 논문의 구성은 다음과 같다. 2장에서는 차량 위치 추적 시스템과 이동 객체 관련 연구를 검토한다. 3장에서는 차량 위치 추적 시스템의 구조와 주요 모듈의 알고리즘을 제시하고 이동 차량의 위치 정보를 모델링한다. 4장에서는 시스템 구현 환경 및 시스템 구현 결과를 살펴보고, 마지막으로 5장에서는 결론과 향후 연구 방향을 제시한다.

2. 관련 연구

기존의 차량 추적 시스템은 첨단 화물 운송 시스템(CVO)[1, 4], APTS의 차량 관리 시스템과 EuroBus의 차량배차 및 제어 시스템[2, 5, 11]등이 있다. 첨단 화물 운송 시스템은 화물 및 화물차량을 효율적으로 관리하여 물류비 절감, 안전사고 방지 및 돌발사고에 대한 응급처리 기능을 향상시켜주는 시스템이다. 이 시스템은 화물 및 화물 차량의 위치를 추적하여 효율적으로 차량 및 배차를 관리하는 화물 및 화물 차량 관리 시스템(FFMS)과 위험물 적재차량의 위치를 추적하여 차량을 특별 관리하고 돌발상황 시 신속하게 사고를 처리하는 위험물 차량관리 시스템(HMMS)으로 구성된다. APTS의 차량 관리 시스템은 미국의 첨단 대중교통 시스템에서 차량의 위치 및 승객과 관련된 정보를 제공해 주는 시스템이다. EuroBus의 차량 배차 및 제어 시스템은 버스에 부착된 수신기를 사용하여 모든 버스의 정보를 중앙 센터에 보내고 중앙 센터는 버스의 위치 및 관련 정보를 버스 사업자와 버스 정류장에 보내는 시스템이다.

그러나 이러한 시스템들은 상용 DBMS를 사용하기 때문에 연속적으로 변하는 차량의 이동 정보를 모두 저장하지 못한다. 따라서 시간의 변화에 따른 차량의 이동 정보를 이산적으로 저장하게 되며 저장되지 않은 정보를 사용자가 요구할 경우 이와 관련된 연산을 처리하지 못하는 문제점이 있다. 예를 들면 서울 시청에서 대학로까지 이동한 택배 차량의 위치 좌표를 오전 8시부터 5분 간격으로 저장 및 관리 할 경우 차량의 위치 좌표는 8시 5분, 10분, 15분 등의 시점에서 저장이 된다. 한편 사용자가 8시 6분, 7분, 13분 등의 시점에서 차량 위치 좌표를 요청하면 시스템은 이러한 시점에서의 차량 위치 검색에 대한 응답을 제시하지 못한다. 이와 같은 문제점 해결을 위하여 차량을 이동 점 객체로 정의하고 이동 객체 데이터베이스를 이용한 차량 위치 추적 시스템의 연구[12-14]가 수행되었다. 이러한 시스템에서 사용되고 있는 이동 객체는 시간에 따라 객체의 공간 정보가 연속적으로 변경되는 시공간 데이터로 이동 점과 이동 영역으로 구분된다. 이동 점은 시간에 따라 객체의 위치가 변하는 것으로 이동 점 객체에는 사람, 동물, 자동차, 비행기, 배 등이 있다. 이동 영역은 시간에 따라 객체의 위치뿐만 아니라 모양까지 변하는 것으로 한 국가의 행정 구역이나 폭풍의 영향권, 암세포의 상태 등[6-9]을 예로 들 수 있다.

차량 관리를 위한 대표적인 응용 시스템 연구에는 DOMINO[10, 15-18], CHOROCHRONOS[19-23], Battlefield Analysis[24-26]가 있다. 먼저 DOMINO 프로젝트에서 개발된 프로토타입은 이동 차량의 현재 위치, 속도, 방향 정보를 이용하여 차량의 미래 위치를 예측하는 방법에 초점을 맞춘 시스템이다. 그러나 이 프로토타입은 이동 차량의 불확실한 미래 위치만 다루기 때문에 데이터베이스에 저장되지 않은 과거 위치 정보를 제공하지 못하는 문제점이 있다. CHOROCHRONOS에서는 이동 차량의 데이터 모델링 및 인덱싱에 관한 연구와 위치 및 궤적을 관리하는 차량 관리 시스템에 대한 연구가 이루어졌다. 특히 이 연구는 GPS 기반의 수송 관리 시스템과 멀티미디어 시스템에 적용한 응용 시나리오를 제시하였다. 이 연구 결과는 차량의 과거 위치 데이터만을 고려하기 때문에 차량의 미래 위치와 관련된 정보를 제공하지 못하며 DOMINO와 같은 프로토타입이 개발된 사례가 아직까지 없다. Battlefield Analysis는 이동 부대와 탱크를 이동 점 객체의 특성을 갖는 이동 차량처럼 정의하고 모의 전장에서 이들의 움직임을 예측하여 의사결정에 활용할 수 있도록 개발된 전장분석 프로토타입이다. 그러나 이 프로토타입은 실시간으로 제공되는 데이터를 처리하지 못하며 이로 인하여 모바일 환경에는 적용할 수 없는 단점이 있다.

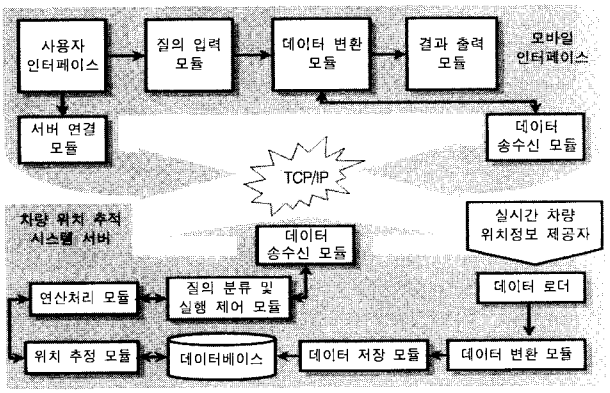
따라서, 이 논문에서는 데이터베이스에 저장되지 않은 이동 차량의 과거 및 미래 위치 추정이 가능하고 모바일 인터페이스를 사용하여 차량의 위치 관련 질의를 처리할 수 있는 차량 위치 추적 시스템을 제안한다. 또한 제안 시스템은 기존에 개발된 응용 시스템들을 확장시킬 수 있는 장점을 갖는다.

3. 모바일 인터페이스를 이용한 차량 위치 추적 시스템의 설계

이 장에서는 실시간 차량 모니터링 시스템의 구조와 각 모듈별 기능을 제시한다. 그리고 시스템에 적용될 이동 차량의 위치 정보를 모델링 한다. 모델링을 위하여 이동 점 객체 데이터 및 연산자를 정의하고 데이터베이스에 저장된 이동 객체의 이력 및 현재 정보의 형태를 제시한다. 마지막으로 시스템에서 사용된 데이터베이스의 구조와 각 모듈별 알고리즘을 제시한다.

3.1 시스템 구성 및 기능

제안 시스템은 차량 위치 추적 시스템 서버와 모바일 인터페이스로 구성되며 (그림 1)과 같은 구조를 갖는다. 차량 위치 추적 시스템 서버는 차량 위치 정보를 저장, 관리하고 모바일 인터페이스의 차량 위치 검색 질의 요청 결과를 반환하는 기능을 한다. 차량 위치 추적 시스템 서버의 구성은 데이터 로더, 데이터 변환 및 저장 모듈, 데이터 송수신 모듈, 질의 분류 및 실행 제어 모듈, 연산처리 모듈, 위치 추정 모듈, 데이터베이스로 되어있다. 모바일 인터페이스는 실시간으로 이동 차량의 위치 검색에 관한 질의를 요청하고 그 결과를 확인할 수 있다. 모바일 인터페이스의 구성은 사용자 인터페이스, 서버 연결 모듈, 질의 입력 모듈, 데이터 변환 모듈, 결과 출력 모듈, 데이터 송수신 모듈로 되어 있다.



(그림 1) 모바일 인터페이스를 이용한 차량 위치 추적 시스템 구성도

차량 위치 추적 시스템 서버의 각 모듈별 기능을 살펴보면 데이터 로더는 이동 차량의 실시간 위치 정보를 수신하고 수신된 차량 정보는 데이터 변환 모듈과 저장 모듈을 거쳐 데이터베이스에 저장된다. 이동 차량의 위치 정보는 일정한 시간 주기를 갖고 외부의 차량 데이터 제공 시스템으로부터 제공된다. 질의 분류 및 실행 제어모듈은 무선 네트워크를 통하여 모바일 인터페이스로부터 입력된 차량 위치 관련 질의를 처리하기 위하여 실행 연산자를 호출하고 제어하는 기능을 한다. 연산 처리모듈은 차량 위치 관련 질의 수행을 위한 연산을 처리한다. 모바일 인터페이스로부터 데이터베이스에 저장되지 않은 과거 및 미래 위치정보와 관련된 질의가 요청되면 위치 추정 모듈이 저장되지 않은 위치 정보를 계산한다. 위치 추정 모듈은 차량 위치를 계산하기 위하여 시간의 변화에 따라 변경되는 이동 차량의 위치 변화 함수를 생성한다. 위치 변화 함수에는 과거 위치 추정 함수와 미래 위치 추정 함수가 사용된다. 데이터베이스에는 이동 차량의 속성정보 및 위치정보가 저장된다.

3.2 이동 차량의 위치 정보 모델

이 논문에서는 이동 차량의 위치 정보 관리를 위하여 이동 점 객체의 형태를 갖는 이동 차량 데이터를 정의한다. 먼저, 이동 차량은 시간의 변화에 따라 객체의 위치 값만 변화되는 이동 객체를 말한다. 이동 차량 MV는 시간 속성, 공간 속성, 일반 속성을 가지며, $MV=\langle T_A, S_A, G_A\rangle$로 구성된다. MV의 시간 속성 $T_A=\langle vt_s, vt_e\rangle$로 구성되며 vt_s는 시작 시간, vt_e는 종료 시간을 나타낸다. 이 때 vt_s와 vt_e는 유효시간의 집합 S_{VT}의 원소가 된다. 유효시간은 실세계에서 발생된 시간을 나타내며, $S_{VT}=\{t_0, t_1, t_2, \cdots, t_k, \cdots, t_{now}\}$이고, 각 원소들은 $t_0 < t_1 < t_2 \cdots < t_k < \cdots < t_{now}$의 순서를 갖는다. $t_k=t_{k-1}+1$, $t_k=t_0+k$, $k\geq 0$인 정수로 정의된다. t_{now}는 현재 시간을 의미하는 시간 상수이다. 유효 시간의 도메인은 선형 시간(linear time), 이산 시간(discrete time), 절대 시간(absolute time)이며, 하나의 데이터베이스는 동일한 유효 시간의 주기를 갖는다. MV의 공간 속성 $S_A=\langle x, y\rangle$, $x, y\in R$, R은 실수이다.

이동 차량의 위치 정보를 저장하는 데이터베이스는 이동 차량의 집합 $S_{MV}=\{mv_i\}_{i=0}^{n}$로 구성된다. S_{MV}로 구성된 데이터베이스의 이력 집합은 $H_{MV}=\{H_{mv_i}\}_{i=0}^{n}$이 된다. S_{MV}에 속하는 각각의 mv_i에 대한 모든 이력 집합은 $H_{mv_i}=\{mv_{i_k}\}_{k=0}^{n}$이고, $mv_{i_k}=\langle T_A(mv_{i_k}), S_A(mv_{i_k}), G_A(mv_{i_k})\rangle$이다. 여기에서 mv_{i_k}는 mv_i의 k번째 이력 정보를 의미하고, $T_A(mv_{i_k})$는 mv_i의 k번째 시간 속성, $S_A(mv_{i_k})$는 k번째 공간 속성, $G_A(mv_{i_k})$는 k번째 일반 속성을 의미한다.

이동 차량의 위치 정보 관리를 위하여 다음과 같은 연산자를 정의한다. 이동 차량의 일반 속성에 대한 검색은 기존의 상용 DBMS가 제공하는 기능을 그대로 사용하므로 별도로 정의하지 않는다. 이동 차량의 위치 정보 관련 연산자로 *SobjSTraj, FobjFTraj, TrajDistance, TrajNearest, PositionAtTime, MDistance*를 <표 1>과 같이 정의한다.

〈표 1〉 위치정보 처리 연산자

종 류	기 능
SobjSTraj	임의의 이동 차량에 대해 특정 시간 동안의 궤적 정보 추출
FobjFTraj	전체 이동시간 내의 모든 차량들의 궤적 정보 추출
TrajDistance	질의 차량들의 궤적간의 직선 거리 값 계산
TrajNearest	임의의 이동 차량의 궤적과 가장 가까운 궤적 정보 추출
PositionAtTime	특정 시간에서의 이동 차량의 위치 정보 추출
MDistance	임의의 두 이동 차량간의 거리 값 추출

<표 1>에서 정의한 연산자는 Erwig[6, 7]이 제시한 이동 점 객체 연산자 중 이 논문에서 차량 위치 검색을 위해 실제 구현한 연산자이다.

3.3 데이터베이스 구조

데이터 저장 구조는 관계 데이터베이스를 기반으로 하며 FleetObject, FleetCurrent, FleetHistory, 세 개의 릴레이션으로 구분하여 저장된다. FleetObject에는 이동 차량의 일반 속성정보가 저장된다.

〈표 2〉 FleetObject 릴레이션

CarId	Name	Type
356583455	fleet1	truck
356583466	fleet2	container

<표 2>에서 나타내는 FleetObject 릴레이션의 CarId는 객체 식별자로 키 값이 된다. 기타 일반 속성 정보는 차량의 이름을 나타내는 Name이나 차량의 유형을 나타내는 Type 등을 가질 수 있다. FleetCurrent에는 이동 차량의 현재의 시간 및 공간 속성정보가 저장된다.

〈표 3〉 FleetCurrent 릴레이션

CarId	Vs	Ve	X_Vs	Y_Vs	X_Ve	Y_Ve
356583455	2002-03-01 -08-20-00	now	202053.00	444755.69	null	null
356583466	2002-03-01 -09-00-00	now	201153.42	445201.23	null	null

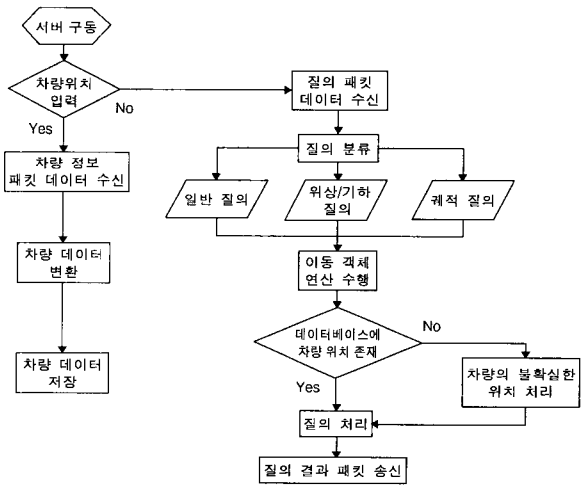
<표 3>의 FleetCurrent 릴레이션은 <표 4>의 FleetHistory 릴레이션과 구조가 동일하다. 가장 최근에 저장된 이동 차량의 위치 좌표만을 FleetCurrent 릴레이션에 저장하고 그 이전의 위치 정보는 모두 FleetHistory 릴레이션에 저장한다. FleetHistory 릴레이션은 데이터베이스에 저장된 모든 과거 시점의 이력 위치 정보를 저장한다. <표 4>는 5분 간격으로 차량의 위치정보를 저장한 FleetHistory 릴레이션의 예이다.

〈표 4〉 FleetHistory 릴레이션

CarId	Vs	Ve	X_Vs	Y_Vs	X_Ve	Y_Ve
356583455	2002-03-01 -07-50-00	2002-03-01 -07-55-00	200998.11	445124.01	201287.75	445238.44
356583455	2002-03-01 -07-55-00	2002-03-01 -08-00-00	201287.75	445238.44	201566.67	445345.72
356583455	2002-03-01 -08-00-00	2002-03-01 -08-05-00	201566.67	445345.72	201809.84	445410.08
356583455	2002-03-01 -08-05-00	2002-03-01 -08-10-00	201809.84	445410.08	201888.51	445188.38
356583455	2002-03-01 -08-10-00	2002-03-01 -08-15-00	201888.51	445188.38	201974.33	444973.82
356583455	2002-03-01 -08-15-00	2002-03-01 -08-20-00	201974.33	444973.82	202053.00	444755.69
356583466	2002-03-01 -08-50-00	2002-03-01 -08-55-00	201005.24	444345.34	201083.56	444980.21
356583466	2002-03-01 -08-55-00	2002-03-01 -09-00-00	201083.56	444980.21	201153.42	445201.23

3.4 차량 위치 추적 시스템 서버

차량 위치 추적 시스템 서버의 동작과정은 (그림 2)와 같다. 시스템의 통신을 위하여 우선 차량 위치 추적 시스템 서버가 실행되어 접속 대기 상태에 있어야 한다. 외부 데이터 제공자로부터 차량 위치 관련 데이터가 전송될 경우 차량 위치 정보 수신 및 저장 모듈을 통하여 데이터베이스에 차량 정보를 저장한다. 모바일 인터페이스로부터 질의가 입력 된 경우에는 적절한 질의 처리 모듈을 통하여 질의를 처리하고 결과를 다시 반환해준다.



(그림 2) 차량 위치 추적 시스템 서버의 동작 과정

3.4.1 연산 처리 모듈

연산 처리 모듈은 해당 질의의 실행 모듈을 호출한다. 실행 가능한 연산자의 종류는 3.2절의 <표 1>과 같다. 알고리즘에 사용되는 데이터베이스는 3.3절의 FleetObject, FleetCurrent, FleetHistory 릴레이션을 토대로 한다.

```
Algorithm SobjSTraj(name, t)
입력 => name : 임의의 이동 차량 mv의 이름, t : 검색하고 싶은 질의 시점
출력 => x : t 시점의 x 좌표 값, y : t 시점의 y 좌표 값
Begin
  FleetObject에서 입력된 name을 갖는 차량의 CarId를 검색
  If (검색 결과가 null이 아니면) Then
    current_time ← FleetCurrent에서 CarId를 갖는 차량의 time 속성 검색
    If (time = current_time) Then
      x ← FleetCurrent에서 CarId를 갖는 차량의 x 속성 검색
      y ← FleetCurrent에서 CarId를 갖는 차량의 y 속성 검색
    Else If (time < current_time) Then
        x ← FleetHistory에서 CarId와 time 속성을 갖는 튜플의 x값 검색
        y ← FleetHistory에서 CarId와 time 속성을 갖는 튜플의 y값 검색
        If (검색 결과가 null 이면) Then
          past_location(CarId, t)모듈 호출한 후 x, y 값 구함
              // past_location : 과거 시점의 불확실한 위치 추정 모듈
      Else future_location(CarId, t)모듈 호출한 후 x, y 값 구함
              // future_location : 미래 시점의 불확실한 위치 추정 모듈
  return x and y   // 이동 차량의 t 시점의 x, y 좌표 값 반환
End
```

(그림 3) 연산 처리 알고리즘

(그림 3)는 특정 시점에 대한 이동 차량의 위치를 검색하는 *SobjSTraj* 연산자의 실행 알고리즘이다. FleetObject에 존재하는 이동 차량을 대상으로, 입력된 시간 값을 비교하여 과거, 현재 또는 미래의 위치 값을 구한다. 각 연산 처리 모듈에서는 과거 및 미래의 시점에 대한 질의가 입력된 경우에 위치 추정 모듈을 호출하여 결과를 구한다.

3.4.2 위치 추정 모듈

위치 추정 모듈은 데이터베이스에 저장되지 않은 특정한 시간에 대한 차량의 위치 좌표를 계산하여 결과를 반환하는 기능을 갖는다. 위치 추정 모듈에서는 과거 및 미래의 위치 정보 추정을 위해 선형 보간법을 이용한다. 이력 데이터베이스에 저장되는 시점이 $\{t_i\}_{i=0}^{now}$ 일 때, $t_i < t_p < t_{i+1}$의 조건을 만족하는 임의의 과거 시점 t_p에 대해 (t_i, x_{t_i})와 $(t_{i+1}, x_{t_{i+1}})$의 위치 좌표 쌍을 이용하여 $x(t_p)$ 함수를 구하고 (t_i, y_{t_i})와 $(t_{i+1}, y_{t_{i+1}})$의 위치좌표 쌍을 이용하여 $y(t_p)$ 함수를 구한다. 이 때 함수 $x(t_p)$ 와 $y(t_p)$ 를 이용하여 과거의 시점 t_p의 위치 값을 추정한다.

```
Algorithm past_location(CarId, t_p)
입력 => CarId : 임의의 이동 차량 mv의 식별자, t_p : 과거의 특정 시점
출력 => x_{t_p} : 시점 t_p시 mv의 x 좌표 값, y_{t_p} : 시점 t_p시 mv의 y좌표 값
Begin
  FleetObject에서 입력된 CarId를 갖는 차량 검색
  If (검색 결과가 null이 아니면) Then
    FleetHistory에서 t_i < t_p < t_{i+1}의 조건을 만족하는 위치 좌표 쌍
        (t_i, x_{t_i}), (t_i, y_{t_i})와 (t_{i+1}, x_{t_{i+1}}), (t_{i+1}, y_{t_{i+1}})을 검색
    If (검색 결과가 null이 아니면) Then
        x_{t_p} ← ((x_{t_{i+1}} − x_{t_i}) / (t_{i+1} − t_i)) (t_p − t_i) + x_{t_i}

        y_{t_p} ← ((y_{t_{i+1}} − y_{t_i}) / (t_{i+1} − t_i)) (t_p − t_i) + y_{t_i}

  Else x_{t_p} ← 오류 값, y_{t_p} ← 오류 값
  Return x_{t_p} and y_{t_p} // 과거의 t_p 시점의 (x, y) 좌표 값 반환
End
```

(그림 4) 과거 위치 추정 알고리즘

(그림 4)의 *past_location*알고리즘은 이동 객체 *mv*의 *Car Id*와 과거의 임의의 시점 t_p를 입력 값으로 받는다. 이 입력 값을 사용하여 차량의 속성 정보가 저장된 FleetObject 릴레이션에서 입력된 *CarId*를 갖는 차량을 검색한다. 검색된 차량이 존재할 경우, 이력 정보가 저장된 FleetHistory 릴레이션에서 t_p와 가장 인접한 이전의 시점 t_i의 (x, y) 좌표 쌍 $(t_i, x_{t_i}), (t_i, y_{t_i})$를 검색한다. 그리고 t_p와 가장 인접한 이후의 시점 t_{i+1}의 (x, y) 좌표 쌍 $(t_{i+1}, x_{t_{i+1}}), (t_{i+1}, y_{t_{i+1}})$를 검색한다. 검색된 위치 좌표를 이용하여 x_{t_p}와 y_{t_p}값을 구하여 결과를 반환한다.

$t_{now} < t_f$의 조건을 만족하는 임의의 미래 시점 t_f에 대하여 $(t_{now-1}, x_{t_{now-1}})$와 $(t_{now}, x_{t_{now}})$의 위치 좌표 쌍을 이용하여 $x(t_f)$ 함수를 구하고 $(t_{now-1}, y_{t_{now-1}})$와 $(t_{now}, y_{t_{now}})$의 위치 좌표 쌍을 이용하여 $y(t_f)$ 함수를 구한다. 이 때 함수 $x(t_f)$ 와 $y(t_f)$를 이용하여 미래의 시점 t_f의 위치 값을 추정한다.

```
Algorithm future_location(CarId, t_f)
입력 => CarId : 임의의 이동 차량 mv의 식별자, t_f : 미래의 특정 시점
출력 => x_{t_f} : 시점 t_f 시 mv의 x 좌표 값, y_{t_f} : 시점 t_f 시 mv의 y좌표 값
Begin
  FleetObject에서 입력된 CarId를 가지는 차량 검색
  If (검색 결과가 null이 아니면) Then
    FleetHistory에서 (t_{now-1}, x_{t_{now-1}}), (t_{now-1}, y_{t_{now-1}})와
                (t_{now}, x_{t_{now}}), (t_{now}, y_{t_{now}}) 좌표 검색
    If (검색 결과가 null이 아니면) Then
```

$$x_{t_f} \leftarrow \frac{x_{t_{now}} - x_{t_{now-1}}}{t_{now} - t_{now-1}} (t_f - t_{now}) + x_{t_{now}}$$

$$y_{t_f} \leftarrow \frac{y_{t_{now}} - y_{t_{now-1}}}{t_{now} - t_{now-1}} (t_f - t_{now}) + y_{t_{now}}$$

Else x_{t_f} ← 오류 값, y_{t_f} ← 오류 값

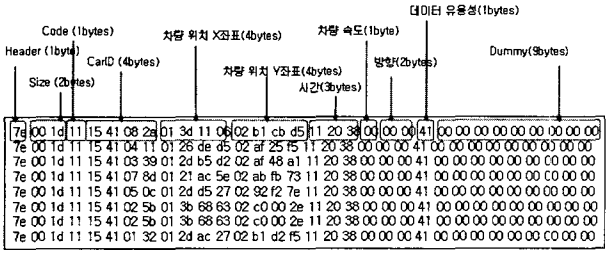
Return x_{t_f} and y_{t_f} // 미래의 t_f 시점의 (x, y)좌표 값 반환

End

(그림 5) 미래 위치 추정 알고리즘

3.4.3 실시간 위치정보 수신 및 저장 모듈

실시간 위치 제공자로부터 전송되는 이동 차량의 위치 정보는 <표 5>와 같은 형태의 패킷으로 구성된다. 이 논문에서 설계한 데이터베이스에 저장될 위치 정보만을 구성요소로 하여 작성한 패킷 구조는 (그림 6)과 같다.



(그림 6) 실시간 차량 위치 정보 패킷

이동 차량의 실시간 이동 정보는 (그림 6)과 같은 구조로 일정한 시간 주기에 따라 데이터 로더에 전송된다. 데이터 로더는 수신된 위치 정보를 (그림 7)과 같은 데이터 변환 및 저장 알고리즘을 통하여 데이터베이스에 저장한다.

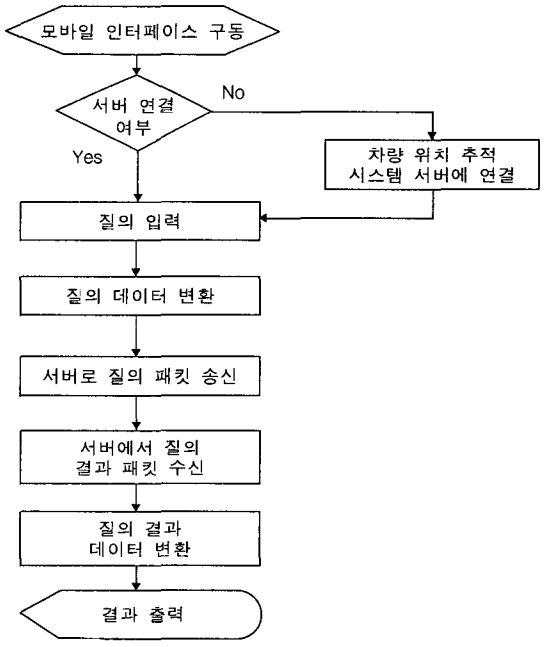
Algorithm data_translator (packet)
입력 => *packet* : 버퍼 형태로 구성된 패킷 데이터 배열
Begin
  CarId ← *packet* 배열 중 *CarId* 부분인 4 번째 값을 정수로 변환
  x ← *packet* 배열 중 x 좌표 값을 나타내는 5 번째 값을 실수로 변환
  y ← *packet* 배열 중 y 좌표 값을 나타내는 6 번째 값을 실수로 변환
  t ← *packet* 배열 중 t 값을 나타내는 7 번째 값을 문자형으로 변환
  data_store(*CarId*, t, x, y) 모듈을 호출하여 실행
End
Algorithm data_store(*CarId*, t, x, y)
입력 => *CarId* : 차량 식별자, t : 시간 값, x : x좌표 값, y : y좌표 값
Begin
  FleetHistory에 *CarId*, t, x, y를 가지는 새로운 튜플 삽입
  FleetCurrent에서 *CarId*를 가지는 객체 검색
    If (검색 결과가 null이 아니면) Then
      *CarId*를 가지는 튜플의 t, x, y 속성 값을 갱신
    Else
      FleetCurrent에 *CarId*, t, x, y를 가지는 새로운 튜플 삽입
End

(그림 7) 위치정보 변환 및 저장 알고리즘

(그림 7)에서 *data_translator*는 데이터 로더를 통해 수신된 위치정보 패킷 중에서 데이터베이스에 실제 저장될 *CarID*, *X*, *Y*, *Time*부분의 값을 해당 자료형으로 변환한다. 그리고 변환된 데이터는 *data_store* 모듈을 통하여 데이터베이스에 저장된다. *data_store* 에서는 FleetHistory 릴레이션에 새로운 *CarID*, *X*, *Y*, *Time*값을 갖는 튜플을 삽입한다. 만약, FleetCurrent 릴레이션에 *CarId*를 갖는 튜플이 존재하면 *X*, *Y*, *Time* 속성을 갱신하고 그렇지 않으면 *CarID*, *X*, *Y*, *Time* 값을 갖는 새로운 튜플을 삽입한다.

3.5 모바일 인터페이스

모바일 인터페이스의 동작과정은 (그림 8)과 같은 순서로 이루어진다. 구현된 차량 위치 추적 시스템의 작동을 위하여 우선 모바일 인터페이스와 서버가 소켓 통신이 가능해져야 한다. 서버가 작동중일 경우 모바일 인터페이스의 서버 연결 모듈을 실행하면 접속 대기중인 서버에 연결되어 TCP/IP를 사용한 소켓 통신이 가능해진다. 그 후 사용자가 질의 입력 모듈을 사용하여 질의를 입력하면 입력된 질의 데이터는 데이터 변환 모듈에서 패킷 데이터로 변환되고 서버로 전달된다. 서버에서 처리된 질의 결과는 모바일 인터페이스의 데이터 수신 모듈에 패킷 형태로 수신된다. 데이터 변환 모듈은 수신된 결과 데이터를 결과 출력 모듈 화면에 출력 가능한 형태로 변환하고 사용자는 변환된 질의 결과를 텍스트 형태로 확인할 수 있다.



(그림 8) 모바일 인터페이스의 동작 과정

3.5.1 서버 연결 모듈

서버 연결 모듈은 모바일 인터페이스에서 차량 위치 추

적 시스템 서버에 질의를 송신하기 위해 가장 처음 수행되는 부분이다.

```
Algorithm ServerConnect()
Begin
  If ( Socket이 Null이 아니면) Then
    화면에 접속 에러 메시지를 표시한다.
  Else
    Create( ) 함수를 호출하여 소켓을 생성한다.
    Connect(ServerIP, PortNmber) 함수를 사용하여 서버에 연결한다.
  if (서버 연결에 실패한 경우) Then
    Close( ) 함수를 호출하여 소켓 생성을 해제시킨다.
    접속 실패 메시지를 화면에 표시한다.
End
```

(그림 9) 서버 연결 알고리즘

(그림 9)는 서버 연결 알고리즘을 나타내며 서버 모듈이 작동중일 때 소켓을 이용하여 서버에 연결하고 이미 연결이 되어 있는 경우나 서버 연결에 실패한 경우에는 에러 메시지를 보여준다. 모바일 인터페이스는 Microsoft의 Embedded Visual C++ 4.0을 사용하였기 때문에 Microsoft Foundation Class를 기반으로 하여 구현되어 있다. 서버 연결 알고리즘에서 사용되는 소켓과 관련된 함수들 또한 MFC에서 제공하는 소켓 클래스를 사용하였다. 소켓 클래스가 Null이 아닌 경우 소켓이 이미 열려 있으나 서버에 재접속을 시도하는 경우이므로 에러 메시지를 표시해 준다. 서버에 접속이 되어 있는 경우에는 *Create*()함수를 호출하여 소켓을 생성하고 서버의 IP 번호와 Port 번호가 입력된 *Connect*()함수를 사용하여 서버에 연결한다. 서버 연결에 실패한 경우는 *Close*() 함수를 호출하여 소켓을 해제시키고 화면에 에러 메시지를 표시한다.

3.5.2 데이터 송수신 모듈

모바일 인터페이스의 데이터 송수신 모듈은 질의를 전송하는 부분과 결과를 수신하는 부분으로 구성된다. (그림 10)은 데이터 변환모듈에서 전송된 문자열 형태의 데이터를 소켓을 이용하여 서버로 전송하는 데이터 송신 부분과 서버의 질의 처리 결과를 전송 받아 결과 출력 모듈로 질의 결과를 전송하는 데이터 수신 부분의 알고리즘을 보여주고 있다.

```
Algorithm Packet (Packet)
입력 => Packet : 패킷 형태로 변환된 데이터
Begin
if (질의 전송 일 경우) Then
   if (Server에 연결되지 않았을 경우) Then
     접속 에러 메시지 표시
   Else
    Send(packet)함수 호출하여 서버에 질의 패킷을 전송한다.
Else (질의 결과 수신 일 경우)
   Recieve(packet)함수를 호출하여 서버로부터 결과 패킷을 전송 받음
End
```

(그림 10) 데이터 송수신 알고리즘

모바일 인터페이스의 데이터 송수신 모듈은 데이터 변환 모듈에서 패킷 형태로 변환된 질의 데이터와 차량 위치 추적 시스템 서버에서 패킷 형태로 전달된 질의 결과를 입력값으로 받는다. 그리고 현재의 패킷을 질의 전송 패킷과 질의 결과 수신 패킷으로 구분한다. 질의 전송의 경우, 우선 서버에 연결되지 않고 질의를 전송하려는 오류를 제거하기 위하여 서버에 연결되어 있는지를 검토한다. 서버에 연결이 되어 있다면 *Send*()함수를 사용하여 패킷 데이터를 서버로 보낸다. 질의 결과 수신일 경우에는 *Recieve*()함수를 호출하여 결과 패킷을 수신한다.

3.5.3 데이터 변환 모듈

데이터 변환 모듈은 모바일 인터페이스의 사용자 인터페이스에서 입력된 질의 데이터를 차량 위치 추적 시스템 서버로 전송하기 위한 패킷 형태의 데이터로 변환한다. 그리고 서버에서 전송 받은 결과 패킷 데이터를 사용자 화면에 출력하기 위한 데이터 형태로 변환하는 기능을 수행한다. (그림 11)은 데이터 입력 모듈에서 전송된 데이터들을 서버의 데이터 송수신 모듈에서 수신하기 알맞은 형태로 변환하는 알고리즘과 서버에서 전달된 결과 패킷 데이터를 사용자 인터페이스화면에 출력하기 위한 형태로 변환하는 알고리즘을 보여준다.

```
Algorithm TransSendData ( Tname, CarId, Ts, Te)
입력 => Tname : 테이블 명, CarId : 차량 번호
        Ts : 유효 시작 시간, Te : 유효 종료 시간
Begin
     (String) Packet 변수에 Tname값과 구분자 할당
     (String) Packet 변수에 CarId값과 구분자 추가 할당
     (String) Packet 변수에 Ts값과 구분자 및 Te값과 구분자 추가
              할당
     (String) Packet 변수에 질의 타입인 Type값을 할당
     Packet (Packet) 함수를 호출하여 변환된 데이터 전송
end
Algorithm TransRecieveData (Packet)
입력 => Packet : 서버에서 전송된 질의 결과 패킷 데이터
Begin
  for (Packet의 길이)
    if (Packet이 구분자인 경우)
      ListBox에 출력하지 않고 통과
    Else
      ListBox에 식별자를 붙여 출력.
End
```

(그림 11) 데이터 변환 알고리즘

TransSendData() 함수는 우선 데이터 입력 모듈로부터

각 *Tname, CarId, Ts, Te* 데이터를 전달받고 *Type* 데이터와 결합하여 패킷 형태로 변환한다. *Type* 데이터는 서버에서 처리될 각 질의 종류를 구분하기 위한 질의 타입이다. 또한 이러한 데이터들은 서버에서 다시 분리되어야 하기 때문에 각 데이터 사이에 구분자를 추가한다. *TransRecieveData*() 함수는 서버에서 전송된 패킷 형태의 질의 결과 데이터를 모바일 인터페이스 화면에 출력하기 위한 형태로 변환한다. 서버에서 전달된 데이터는 구분자를 중심으로 분리되고 각 데이터별 식별자를 붙여 화면에 출력된다.

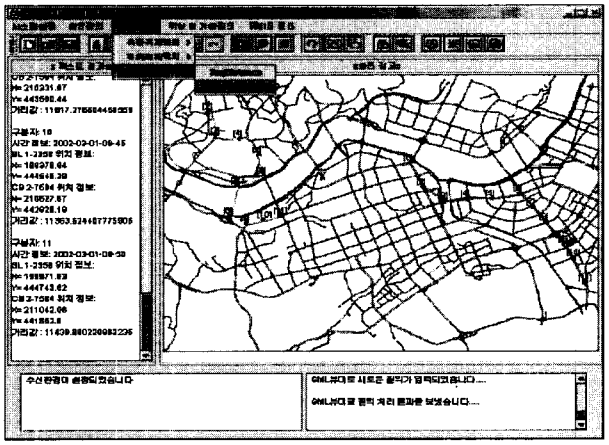
4. 구 현

차량 위치 검색 시스템은 Pocket PC(EVC) 및 JDK 1.3(Java)을 사용하여 구현하였으며 DBMS로는 Microsoft SQL Server 2000를 사용하였다. 모바일 인터페이스는 Microsoft Windows CE 2002(Pocket PC) 운영체제를 사용하는 PDA로 구현하였다. Pocket PC 기반 애플리케이션들은 Microsoft Embedded Visual Basic(EVB) 및 Microsoft Embedded Visual C++(EVC) 등으로 구현 할 수 있다. 이 논문에서 구현한 모바일 인터페이스는 EVC 4.0를 사용하여 구현하였다. 모바일 인터페이스의 테스트는 Pocket PC 운영체제를 내장한 PDA 및 Desktop Pocket PC Emulation에서 실시하였다. 차량 위치 검색 서버는 플랫폼에 독립적으로 운영될 수 있는 JDK에서 사용될 수 있도록 JAVA를 사용하여 구현하였다. 서버 모듈의 테스트는 Windows 2000 Server 운영체제, JDK 1.3 환경에서 실시하였다.

4.1 차량 위치 추적 시스템 서버의 연산 처리

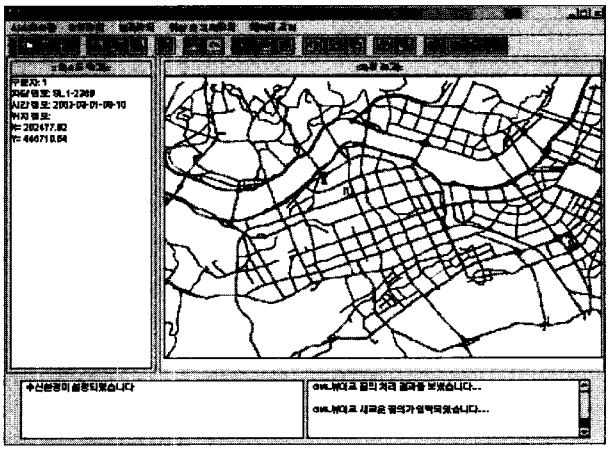
이 절에서는 3.2절에서 정의한 연산자 중 *TrajNearest, PositionAtTime, MDistance* 연산 결과를 질의 예를 통하여 보여준다. 차량 위치 추적 시스템 서버는 모바일 인터페이스와 별도로 질의처리가 가능한 프로그램을 갖고 있다. 서버 프로그램에서는 모바일 인터페이스에서 요청할 수 있는 모든 질의를 요청할 수 있다. (그림 12)는 '임의 이동 차량의 궤적과 가장 가까운 차량의 궤적'을 구하는 *TrajNearest* 연산 결과를 보여준다. 이 연산에 사용된 질의 예는 '2002년 3월 1일 오전 08시 00분부터 2002년 3월 1일 오전 08시 50분 사이에서 차량 SL 1-2358과 가장 가까운 거리에 있는 차량들의 궤적을 구하여라.'이다.

차량 위치 추적 시스템 서버 프로그램의 사용자 인터페이스를 사용하여 테이블명, 차량번호, 유효 시작 시간, 유효 종료 시간을 입력한다. 입력된 질의 데이터는 서버 프로그램의 질의 분류 모듈을 거쳐 연산 처리 모듈에 전달된다. 연산 처리 모듈은 질의 데이터를 사용하여 데이터베이스의 차량 위치 관련 정보 검색한다. 검색된 차량 정보는 (그림 12)와 같이 서버 프로그램의 사용자 인터페이스에서 텍스트 및 그래픽 형태로 출력된다. 텍스트 형태의 출력 데이터는 시간 정보, SL 1-2358 차량의 위치 정보, 이 차량과 가장 가까운 차량들의 같은 시간에서의 위치 정보, 두 차량간의 거리값이다. 그래픽 형태의 출력은 SL 1-2358 차량의 시간별 이동한 궤적과 이 차량과 가장 가까운 거리에 위치한 차량의 궤적을 보여준다.



(그림 12) TrajNearest 연산 수행 결과

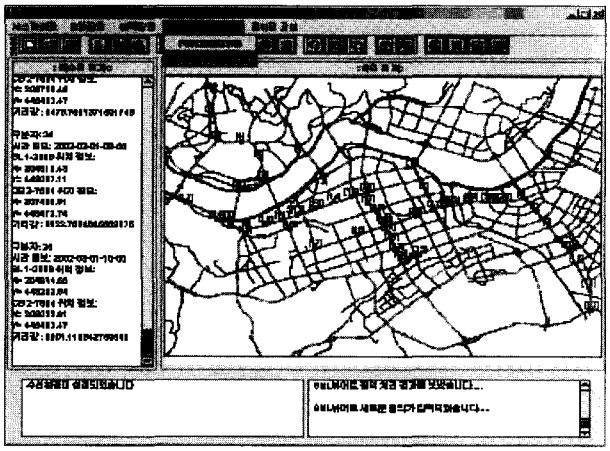
*PositionAtTime*은 '특정 시간에서의 이동 차량의 위치 정보'를 구하는 연산으로 사용된 질의 예는 '2002년 3월 1일 09시 10분에 차량 SL-2358의 위치를 검색하라.'이고 (그림 13)에서 연산 결과를 보여준다.



(그림 13) *PositionAtTime* 연산 수행 결과

TrajNearest 연산과 같은 방법으로 서버 프로그램의 질의 입력 인터페이스를 사용하여 테이블명, 차량 번호, 특정 시간을 입력한다. 입력된 데이터는 프로그램의 연산처리를 통하여 사용자 인터페이스에 출력된다. (그림 13)과 같이 검색된 차량 번호, 검색 시간, 위치 좌표가 텍스트 형태로 출력되고 차량의 도로상의 위치가 그래픽 형태로 출력된다.

마지막으로 (그림 14)에서는 '임의의 두 이동 차량간의 거리 값'을 구하는 *MDsitance* 연산 결과를 나타낸다. 사용된 질의 예는 '2002년 3월 1일 오전 08시 00분부터 2002년 3월 1일 오전 10시 00분까지 차량번호가 SL 1-2358이고 차량번호가 CB 2-7584인 차량간의 거리를 검색하라.'이다.



(그림 14) MDistance 연산 수행 결과

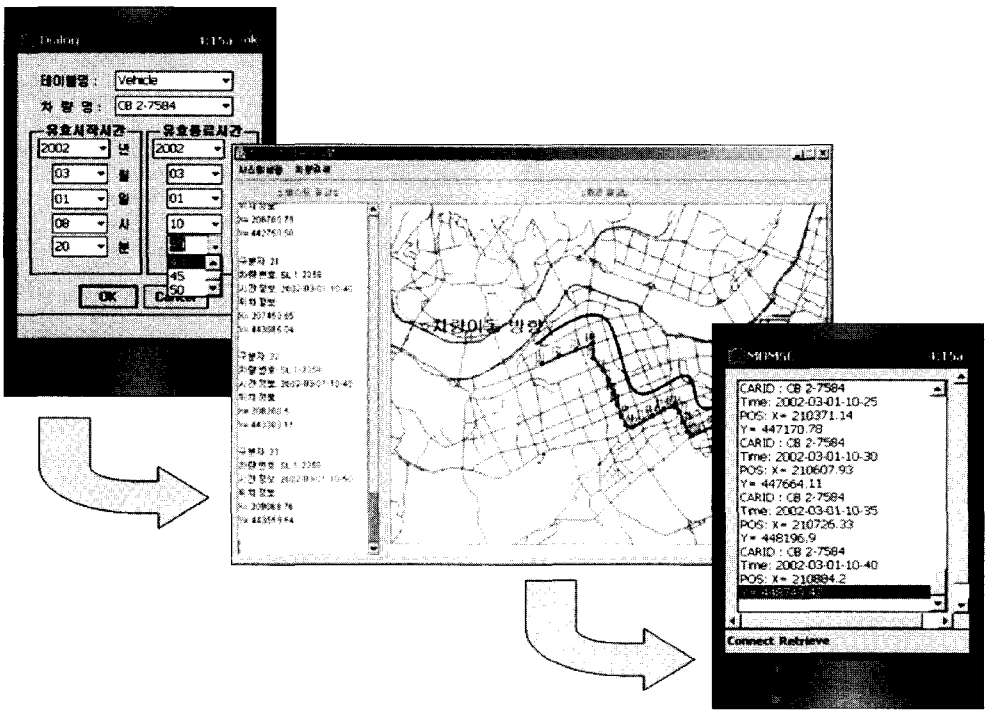
*MDsitance*질의 입력 데이터는 테이블명, 거리를 알고싶은 두 차량의 차량 번호, 유효 시작 시간과 끝 시간이다. 입력 데이터를 사용하여 질의를 처리한 결과는 (그림 14)와 같다. 텍스트 형태의 결과 데이터는 시간 정보, 두 차량의 위치 정보, 두 차량간의 거리 값이다. 또한 두 차량의 위치가 지도상에 그래픽 형태로 출력된다.

4.2 모바일 인터페이스의 차량 위치 검색

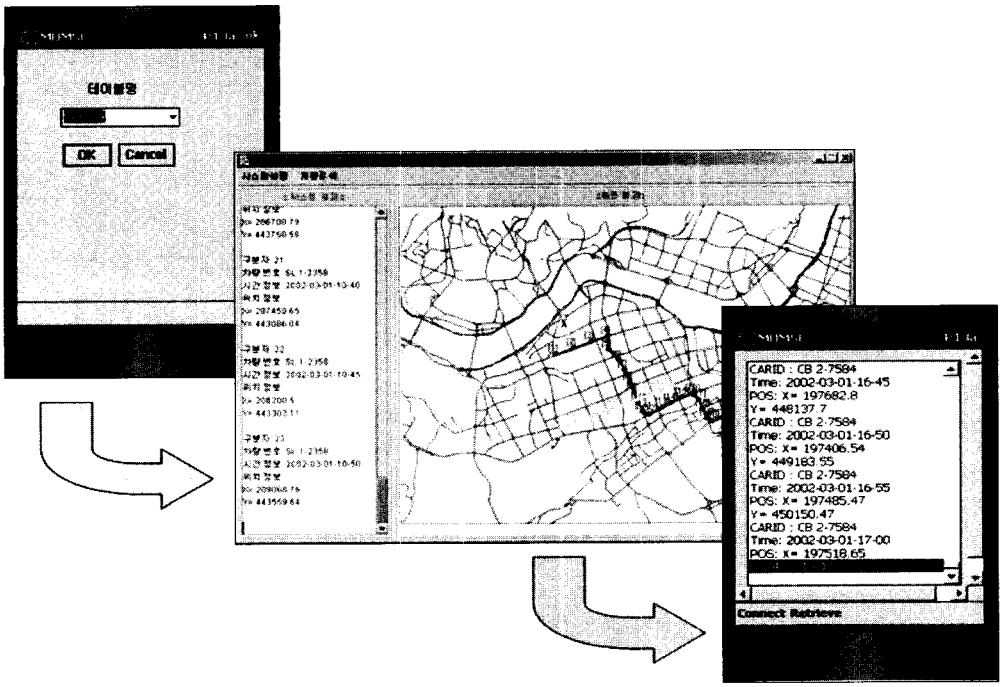
모바일 인터페이스에서 차량 위치 검색을 실시하기 위해서는 우선 위치 검색 서버를 실행시킨다. 그리고 모바일 인터페이스를 실행시키고 서버에 접속한다. 모바일 인터페이스는 질의 메뉴를 통하여 질의를 입력하고 차량 위치 검색 서버에 전송한다. 서버에서 수행된 질의 결과는 클라이언트의 리스트 박스에서 텍스트 형태로 볼 수 있다. 이 논문에서 구현된 모바일 인터페이스의 차량 위치 검색 기능 중에서 '특정 차량의 특정 시간구간 위치 검색 질의'와 '모든 차량의 전체 시간구간 위치 검색 질의' 수행 과정을 예시한다. (그림 15)는 특정 차량의 특정 시간구간 위치 검색 질의 수행과정이다. 질의 예는 '2002년 3월 1일 오전 8시 20분부터 2002년 3월 1일 오전 10시 40분까지의 CB 2-3584 차량의 위치를 검색하라'이다.

모바일 인터페이스의 질의 메뉴에서 질의를 선택하고 테이블명, 차량명, 유효시작 및 종료시간을 입력하면 서버로 질의 데이터가 전송된다. 서버에서는 질의를 처리하고 (그림 15)의 서버 인터페이스와 같이 텍스트 및 그래픽 형태로 결과를 보여주고 클라이언트로 결과를 보낸다. 마지막으로 모바일 인터페이스는 결과를 전송 받아 리스트박스에 텍스트 형태로 결과를 보여준다.

(그림 16)은 모든 차량의 전체 시간구간 위치 검색 질의 수행과정이다. 질의 예는 '데이터베이스에 존재하는 모든 차량의 위치를 검색하라'이다.



(그림 15) 특정 차량의 특정 시간구간 위치 검색 질의 수행



(그림 16) 모든 차량의 전체 시간구간 위치 검색 질의 수행

특정 차량의 특정 시간구간 위치 검색 질의 입력과 같은 방법으로 모바일 인터페이스에서 질의를 선택하고 테이블명을 입력하면 서버에서 질의를 처리한다. 질의 결과는 차량 위치 검색 서버에서는 텍스트 형태 및 궤적 형태로 확인 할 수 있고 모바일 인터페이스에서는 텍스트 형태로 확인할 수 있다.

5. 결 론

기존의 차량 위치 검색 시스템은 데이터베이스에 저장되지 않은 이동 차량의 불확실한 과거 및 미래의 위치 추정 방법을 제시하지 못하고 있다. 또한 주로 유선 네트워크 상에서 차량의 위치 검색 서비스를 제공하기 때문에 무선 통신을 이용하는 모바일 클라이언트에서의 실시간 위치 검색 기능을 수행할 수 없다. 따라서 이 논문에서는 무선 네트워크 상에서 PDA와 같은 모바일 인터페이스를 이용하여 실시간으로 차량의 위치를 검색할 수 있는 차량 위치 추적 시스템을 설계하였다. 제안 시스템은 차량 위치 추적 시스템 서버와 모바일 인터페이스로 구성된다. 차량 위치 추적 시스템 서버는 차량의 과거 및 미래 위치 정보를 저장, 관리하고 모바일 인터페이스에 차량 위치 검색 결과를 제공하도록 하였다. 모바일 인터페이스는 실시간으로 차량 위치 관련 정보를 서버에 요청하고 서버로부터 제공된 위치 검색 결과를 출력하는 역할을 한다.

이러한 차량 위치 추적 시스템 구현을 위하여 시스템에 적용할 차량 위치 정보를 모델링 하였으며 시스템 구성 및 처리 알고리즘을 제시하였다. 구현 시스템에서 제공하는 차량 위치 검색 기능은 '모든 차량의 전체 시간구간 위치 검색 질의', '모든 차량의 특정 시간구간 위치 검색 질의', '특정 차량의 전체 시간구간 위치 검색 질의', '특정 차량의 특정 시간구간 위치 검색 질의', '특정 시점에서의 차량 위치 검색 질의'등이다. 이 논문에서는 이 질의들을 만족하는 차량 위치 검색 기능이 잘 수행됨을 확인하였다. 아울러 제안 시스템의 실행 모델 및 가상 시나리오를 적용한 구현 결과를 보임으로써 모바일 환경에서 이동 클라이언트들의 과거 및 미래 차량 위치 검색 질의를 실시간으로 처리할 수 있음을 보였다.

제안 시스템은 현재 우정 물류 분야의 e-Logistics 시스템의 일부로 개발 중에 있다. 이 논문에서 설계한 모바일 인터페이스는 텍스트 형태로 차량의 위치 정보 결과를 출력하기 때문에 차량의 위치 검색 결과를 시각적으로 표현하지 못하고 있다. 따라서 향후에는 모바일 인터페이스의 차량 위치 추적 결과 출력 모듈에 전자 지도를 추가적으로 구성할 수 있도록 시스템을 확장 및 구현하는 연구를 진행할 것이다.

참 고 문 헌

[1] 교통개발연구원, 한국통신, "첨단 화물운송시스템(CVO) 기본설계", 1997.

[2] 김종혁, "첨단 교통관리 시스템", 정보과학회지, 제16권 제6호, pp.5-13, 1998.

[3] 김형욱, 김성수, "물류 정보서비스를 위한 통신 매체와 이를 이용한 교통 물류관제 시스템의 전반적인 이해와 적용", 한국

통신 Technical Memo, 1997.
[4] 안승범, "안전도향상을 위한 첨단 화물운송 시스템(CVO)의 서비스와 기술", 정보과학회지, 제16권 제6호, pp.30-35, 1998.
[5] 이승룡, 홍영래, 김형일, 배수강, 최대순, "첨단 대중교통 시스템", 정보과학회지, 제16권 제6호, pp.23-29, 1998.
[6] M. Erwig, R. H. Guting, M. Schneider, and M. Vazirgiannis, "Abstract and Discrete Modeling of Spatio-Temporal Data Types," Chorochronos Technical Report, CH-98-14, 1998.
[7] M. Erwig, R. H. Güting, 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.
[8] L. Forlizzi, R. H. Güting, E. Nardelli, and M. Schneider, "A Data Model and Data Structures for Moving Objects Databases," In Proc. of the ACM SIGMOD Conf., pp.319-330, 2000.
[9] 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.
[10] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, "Moving Objects Databases : Issues and Solutions," Proc. of the 10th International Conference on Scientific and Statistical Database Management, SSDBM'98, Capri, Italy, pp.111-122, 1998.
[11] Federal Transit Administration, "Advanced Public Transportation Systems : The State of the Art Update '96," U.S. Department of Transportation FTA-MA-26-7007-96-1, Jan., 1996.
[12] 류근호, 안윤애, 이준욱, 이용준, "이동객체 데이터베이스와 위치기반 서비스의 적용", 데이터베이스연구회지, 제17권 제3호, pp.57-74, 2001.
[13] 장승연, 정영진, 양은주, 안윤애, 류근호. "관계 데이터베이스를 이용한 이동 객체 처리기의 구현", 한국정보처리학회 지식 및 데이터공학연구회, pp.205-211, 2001.
[14] 안윤애, 류근호, 김동호, "차량 위치 추적을 위한 이동 객체 관리 시스템의 설계", 정보처리학회논문지D, 제9-D권 제5호, 2002.
[15] P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects," Proc. of the 13th International Conference on Data Engineering, ICDE'97, Birmingham, UK, Apr., 1997.
[16] P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Querying the Uncertain Position of Moving Objects," Springer Verlag Lecture Notes in Computer Science number 1399, pp.310-337, 1998.
[17] O. Wolfson, P. Sistla, B. Xu, J. Zhou, S. Chamberlain, N. Rishe, and Y. Yesha, "Tracking Moving Objects Using Database Technology in DOMINO," Proc. of NGITS'99, The 4th Workshop on Next Generation Information Technologies and Systems, Zikhron-Yaakov, Israel, pp.112-119, 1999.
[18] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez, "Cost and Imprecision in Modeling the Position of Moving Objects," Proc. of the 14th International Conference on Data Engineering, ICDE'98, Orlando, FL, Feb., 1998.
[19] S. Saltenis, C. S. Jensen, S. Leutenegger, and M. Lopez, "Indexing the Positions of Continuously Moving Objects," Proc. of the ACM SIGMOD Conference, pp.331-342, 2000.
[20] D. Pfoser, Y. Theodoridis, and C. S. Jensen, "Indexing Trajectories of Moving Point Objects," Chorochronos Technical Report, CH-99-3, Oct., 1999.
[21] D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel Approaches in Query Processing for Moving Objects," Proc. of the VLDB Conference, pp.395-406, 2000.
[22] D. Pfoser and C. S. Jensen, "Capturing the Uncertainty of Moving Object Representations," Proc. of Advances in Spatial Databases, 6th International Symposium, SSD'99, pp.20-23, 1999.
[23] D. Pfoser and N. Tryfona, "Fuzziness and Uncertainty in Spatiotemporal Applications," Chorochronos Technical Report, CH-00-4, Feb., 2000.
[24] I. B. Oh, Y. A. Ahn, E. J. Lee, K. H. Ryu, and H. G. Kim, "Prediction of Uncertain Moving Object Location," Proc. of International Conference on East-Asian Language Processing and Internet Information Technology, EALPIIT'02, pp.51-58, Jan., 2002.
[25] S. S. Park, Y. A. Ahn, and K. H. Ryu, "Moving Objects Spatiotemporal Reasoning Model for Battlefield Analysis," Proc. of Military, Government and Aerospace Simulation part of ASTC'01, pp.108-113 2001.
[26] K. H. Ryu and Y. A. Ahn, "Application of Moving Objects and Spatiotemporal Reasoning," TimeCenter TR-58, 2001.

오 준 석

e-mail : seokoh@dblab.chungbuk.ac.kr
2002년 한성대학교 정보공학과 졸업(학사)
2002년~현재 충북대학교 대학원 전자계산학과 석사과정
관심분야 : Mobile database, Mobile GIS, LBS, ITS 등

안 윤 애

e-mail : yeahn@dblab.chungbuk.ac.kr
1993년 한남대학교 전자계산공학과 졸업(학사)
1996년 충북대학교 대학원 전자계산학과 (이학석사)
1999년~현재 충북대학교 대학원 전자계산학과 박사수료
관심분야 : 시공간 데이터베이스, 모바일 데이터베이스, 지리정보 시스템, 지식기반 정보검색 시스템 등

장 승 연

e-mail : syjang@dblab.chungbuk.ac.kr

2001년 충북대학교 경영정보학과(경영학사), 멀티미디어공학(공학사)

2001년~현재 충북대학교 대학원 전자계산학과 석사과정

관심분야 : 시공간 데이터베이스, 이동 객체 데이터베이스, 지리정보 시스템

이 봉 규

e-mail : bong97@hansung.ac.kr

1988년 연세대학교 졸업(학사)

1992년 Cornell University Dept. of CRP 졸업(석사)

1994년 Cornell University Dept. of CRP 졸업(박사)

1993년~1997년 Cornell University Dept. of CRP 조교수

1997년~현재 한성대학교 정보전산학부 부교수

관심분야 : Mobile Computing, XML, GML, GIS, ITS 등

류 근 호

e-mail : khryu@dblab.chungbuk.ac.kr

1976년 숭실대학교 전산학과 졸업

1980년 연세대학교 공학대학원 전산전공 (공학석사)

1988년 연세대학교 대학원 전산전공 (공학박사)

1976년~1986년 육군군수지원사전산실(ROTC 장교), 한국전자통신연구소(연구원), 한국방송통신대, 전산학과(조교수) 근무

1989년~1991년 Univ. of Arizona Research Staff (TempIS 연구원, Temporal DB)

1986년~현재 충북대학교 전기전자 및 컴퓨터공학부 교수

관심분야 : 시간 데이터베이스, 시공간 데이터베이스, Temporal GIS, 객체 및 지식베이스 시스템, 지식기반 정보검색 시스템, 데이터 마이닝, 데이터베이스 보안 및 Bio-Informatics 등