韓國地形空間情報學會論文集 第4卷 第1號 1996年 6月 pp. 103-114

이동체 통제 시스템을 위한 데이타 관리자

A Data Manager for a Vehicle Control System

한재준* 한기준**
Han, Jae-Joon Han, Ki-Joon

요 약

이동체 통제 시스템은 자동차 항법 시스템(Car Navigation System)과 GIS(Geographical Information System), 통신망 등을 결합하여 중앙통제실에서 동적으로 이동체들의 위치를 감지하고, 이를 종합적으로 관리하는 시스템이다. 따라서, 이동체 통제 시스템은 신속한 순찰차의 이동이나 화물 수송 등을 실시간에 통제할 수 있다.

본 논문에서는 이동체 통제 시스템과 같이 도로 데이타, 이동체 데이타, 이동체의 위치 데이타 등을 다루는 응용 시스템의 개발을 위한 데이타 관리자를 설계하고 구현하였다. 본 논문에서 제안하는 데이타 관리자는 크게 시스템 관리 모듈, 도로 데이타 관리 모듈, 이동체 데이타 관리 모듈, GPS 데이타 관리 모듈, 부가정보 데이타 관리 모듈로 구성되어 있다. 그리고, 저장 시스템으로는 미국 위스콘신 대학에서 개발하고 있는 SHORE(Scalable Heterogeneous Object REpository) 베타 0.9.3 버젼의 저장 관리자를 사용하였다.

Abstract

A vehicle Control System that is a system combined by a Car Navigation System, a Geograhical Information System, and a Communication Network usually permit a central control room to detect vehicle's locations dynamically and manage these informations synthetically. Therefore, a vehicle control system can be used to control the rapid movement of patrol cars, freight transportation, and so on.

In this paper, we designed and implemented a data manager for an application system, such as a vehicle control system, that usually manages road data, vehicle data, vehicle's location data, etc. The data manager that is implemented in this paper consists of system management module, road data management module, vehicle data management module, GPS(Global Positioning System) data management module, and additional data management module. Especially, we use the SHORE(Scalable Heterogeneous Object REpository) system as a storage system of the data manager.

1. 서론

최근들어 국내외에서 이동체 통제 시스템에 관한 연구가 활발히 진행되고 있다[5),13),23)]. 이동체 통제 시스템은 자동차 항법 시스템(Car Navigation System), 지리 정보 시스템 그리고 통신망 등을 결합하여 중앙 통제실에서 이동체의 위치를 감지하고 이를 종합적으로 관리할 수 있는 시스템이다. 이동체 통제 시스템의 일환인 자동차 항법 시스템은 항법용 위성에서 송출하는 전파를 수신하여 위치 정보를 구하고, 또한 이를 사용하여 운전자가 지정한 목적지까지의 최적 주행 경로를 제공하는 기능을 수행한다[21)].

자동차 항법 시스템의 가장 초보적인 형태는 시스템의 저장 매체에 저장된 정적인 정보를 제공하는 정도이지만, 자동차 항법 시스템이 차세대 교통 관리 시스템과 연계된다면 운전자에게 동적인 교통 정보를 실시간에 제공할 수 있다. 차세대 교통 관리 시스템은 자동차 항법 시스템을 장착한 자동차로부터 제공받은 각각의 자동차의 위치 정보 및 교통 상황에 관한 정보, 교통 정보 서비스 센터로부터 제공받은 도로 상황

* 건국대학교 전자계산학과 석사과정
** 건국대학교 전자계산학과 교수

정보 및 기상정보 등을 종합하여 자동차 항법 시스템을 장착한 차량이 가고자 하는 목적지까지 가장 빠르고 안전하게 도달할 수 있도록 경로 및 그 밖의 필요한 정보를 제공한다[7),23)].

현재 지리 데이타를 처리하는 응용 시스템의 개발을 위해 다양한 도구들(예를 들면, 지리 정보 시스템)들이 개발되어 있다. 그러나, 대부분 모든 도구들이 범용의 기능을 갖고 있기 때문에 가격이 비싸며, 또한 이들을 사용하여 특정 분야를 위한 응용 시스템을 개발하고자 할 때는 불필요한 기능들이 포함되어 있을 수 있기 때문에 시스템 오버헤드만을 야기할 수 있다. 그리고, 진짜 필요한 기능이 빠져있을 수 있으므로 이러한 도구들을 사용하여 시스템을 개발하고자 할 때는 시간과 비용에서 많은 낭비를 초래할 수 있다.

따라서, 본 논문에서는 차세대 교통 관리 시스템의 일환인 이동체 통제 시스템과 같이 도로 데이타, 이동체 데이타, 이동체의 위치 데이타 등을 다루는 응용 시스템의 개발을 쉽게 하기 위한 데이타 관리자를 설계하고 구현하였다. 이 데이타 관리자는 크게 시스템 관리 모듈, 도로 데이타 관리 모듈, 이동체 데이타 관리 모듈, GPS 데이타 관리 모듈, 부가정보 데이타 관리 모듈로 구성된다.

시스템 관리 모듈은 저장 시스템인 SHORE[16), 17)] 저장 관리자를 운용하는 함수들과 다른 서버와의 연결을 수행하는 함수들로 구성되며, 도로 데이타 관리 모듈은 최단 경로 추출 혹은 화면 출력을 위한 도로 데이타를 처리하는 함수들로 구성되며, 이동체 데이타 관리 모듈은 응용 시스템에서 관리하는 이동체에 관한 정보를 처리하는 함수들로 구성된다. 그리고, GPS 데이타 관리 모듈은 응용 시스템에서 관리하는 이동체들의 위치 데이타를 수신하여 관리하는 함수들로 구성된다. 마지막으로 부가정보 데이타 관리 모듈은 자동차 항법 시스템이나 이동체 통제 시스템과 같은 응용에서 여행지에 관한 정보나 기상 정보와 같은 도로 정보와는 다른 정보들을 처리하는 함수들로 구성된다.

본 논문의 구성은 다음과 같다. 제 2 장은 관련연구로서 지리 정보 시스템과 Global Positioning System(GPS), 자동차 항법 시스템, SHORE, 최단 경로 생성 알고리즘인 Dijkstra 알고리즘 등에 대해 살펴본다. 제 3 장에서는 본 논문에서 제안한 데이타 관리자의 설계에 대해 언급하고, 제 4 장에서는 실제 구현을 어떻게 했는지에 관해 언급한다. 제 5 장에서는 결론 및 이후의 연구과제에 대해 살펴본다.

2. 관련 연구

2.1 지리 정보 시스템과 GPS

지리 정보 시스템은 지표 상의 다양한 사물들에 관한 복잡한 데이타를 수집, 저장하고 이를 분석, 가공하여 각종 지리 관련 응용 분야 즉, 도시 계획, 경로 최적화, 지도 제작, 인구조사, 항법, 천연 자원 관리, 교통, 행정, 군사 등에 이용되는 시스템을 말한다. 지리 정보 시스템을 사용하면 종이로 된 지도상에 표현되는 데이타는 물론, 해당 지형 객체에 관한 여러가지 정보를 보다 종합적이고 효율적으로 처리할 수 있다[12),22)].

GPS는 지구 궤도를 선회하면서 항법 정보를 지구상을 향해 24시간 계속해서 발송해 주는 위성으로서 이를 이용하면 지표상의 절대 위치를 측정할 수 있다[19)]. GPS를 사용하면 위치 데이타의 연속성을 보장받을 수 있지만 가로수, 건물군에 의한 전파 방해때문에 자료의 정확성이 떨어진다는 단점이 있다. 이러한 자료의 부정확성을 보정하기 위한 방법으로는 현재 DGPS(Differential GPS)가 널리 쓰이고 있다. DGPS는 GPS 위치 측정시 정확한 위치가 확보된 하나 이상의 기준국의 위치 데이타를 새로운 위치를 측정할 때 이용하는 방법으로서 타 방법에 비해 높은 정확성을 제공할 수 있는 방법이다[1)].

GPS는 정밀 3차원 측지, 민간 및 군사 지도 제작, 지리 정보 시스템, LIS(Land Information System), 지형도 작성, 해양 측량, 해양 자원 탐사, 석유 시추선의 위치, 해저 지도 작성, 준설 작업, 전파 강도 관측도 작성, 자동차 및 선박, 비행기의 항법 시스템, 도로 지도 작성, 토지 및 수자원 관리 등에 사용될 수 있다.

2.2 자동차 항법 시스템

자동차 항법 시스템이란 기본적으로 지리 정보 시스템과 GPS를 결합하여 운전자가 지정한 위치까지의 최적의 경로를 제공하는 시스템이다. 이러한 자동차 항법 시스템에 관한 연구는 실시간에 교통 정보를 운전자에게 제공하여 정체 지역을 피해갈 수 있도록 하는 방향으로 진행되고 있다[18]. 일본과 미국, 유럽 등지에서는 이미 자동차 항법 시스템에 관한 연구가 상당히 진척되어 실용화 단계에 있다. 특히 일본의 경우에는 일반 사용자들에게 적합한 가격 조건을 갖추었기 때문에 현재 약 80만명 이상의 사용자들이 자동차 항법 시스템을 이용하고 있는 실정이다.

일본에서 일반 운전자들이 사용하고 있는 자동차 항법 시스템은 개발 순서에 따라 크게 4가지로 분류된다. 첫번째 부류는 가장 초보적인 시스템으로 인공위성을 이용해 교통 상황과 현재 위치를 알려주는 시스템이다. 이것은 지역에 따라서 전파 수신 사각 지대가 있다는 점과 지역에 따른 오차가 문제가 된다. 두번째 부류는 GPS에 자이로(gyro)라는 지자기 센서를 부착하여 전파 수신 사각 지대에서 일어나는 문제점을 완화시킨 시스템이다.

세번째 부류는 RGS(Root Guidance System)라는 시스템인데 지난 1988년 소개되어서 현재 일본 운전자들에게 보편화되어 있다. 이는 각종 교통 상황이 중앙통제실에 입력되어 관리되고, 운전자는 목적지만 선택하면 안내 지도가 화면에 나타나서 운전자가 목적지까지 도달하는 것을 도와준다. 네번째 부류는 VICS(Vehicle Information Communication System)라는 현재 일본에서 개발중인 형태이다. 여기에서는 도로 상에 설치되어 차량의 통과 여부를 감지할 수 있는 비콘(beacon)등의 장치를 통하여 일정 구간을 통과한 차량의 숫자 등과 같은 실시간의 교통 정보가 운전자에게 전달되기 때문에 정체 지역을 보다 효율적으로 피해갈 수 있다.

2.3 SHORE

SHORE는 위스콘신 대학에서 객체 지향 데이타베이스와 화일 시스템 기술을 접목시킨 새로운 지속 객체 시스템(persistent object system)이다. SHORE는 여러 개의 데이타 서버들이 서로 연동하는 구조로 되어 있으며 대칭적인 peer-to-peer 구조를 지원하고 있다[16],[17]. 이 구조에서 SHORE의 모든 클라이언트 프로세스는 데이타 저장 볼륨을 가지고 있지 않고 지역 서버가 SHORE 저장 관리자의 저장 볼륨에 저장되어 있는 데이타를 가져온다. 또한, SHORE에서 클라이언트 프로세스는 지역 서버뿐만 아니라 원격 서버의 데이타를 가지고 올 수 있고, 지역 서버는 다른 원격 서버의 데이타를 사용할 수 있다. 그림 2.1은 SHORE의 프로세스 구조이다.

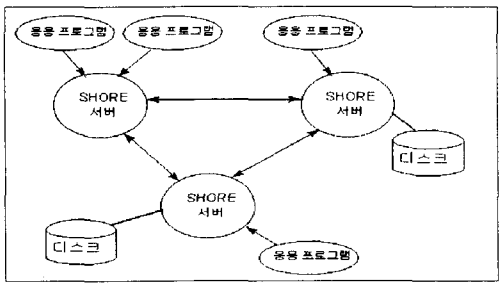

그림 2.1 : SHORE의 프로세스 구조

SHORE에서는 value-added 서버라는 개념이 사용되고 있다. 이것은 사용자가 기존의 SHORE 서버 소스 코드를 수정하여 응용 시스템에 적합한 서버를 구축하여 사용할 수 있도록 한다.

Paradise(Parallel Data Information System)는 EOSDIS(Earth Observing System Data and Information System)를 위한 병렬 정보 시스템으로 개발되고 있다. Paradise는 기본적인 SHORE 서버에 카탈로그 관리자(catalog manager), 익스텐트 관리자(extent manager), 튜플 관리자(tuple manager), 질의 최적기(query optimizer), 실행 엔진(execution engine), 그리고 point, polyline, polygon과 레스터 ADT 등을 추가한 value-added 서버이다[3],[6].

SHORE를 사용하면 Paradise를 개발할 때 처럼 기존의 SHORE 서버에 필요한 기능을 추가할 수도 있지만 반대로 기존의 SHORE 서버의 기능에 비해 간

단한 기능만을 수행하는 작은 SHORE value-added 서버도 만들 수도 있다. 예를 들어, 단순히 한줄의 문자를 SHORE 저장 관리자에 저장했다가 다시 출력하는 기능을 수행하는 서버도 SHORE value-added 서버라고 할 수 있다[17].

2.4 최단 경로 생성 알고리즘

이동체 통제 시스템이나 자동차 항법 시스템에서 어떤 이동체를 위한 최단 경로를 얻어내기 위해 최단 경로 생성 알고리즘이 필요하다. 이러한 응용에 많이 쓰이는 알고리즘이 Dijkstra 알고리즘[4]이다.

Dijkstra 알고리즘은 그래프상에서 두 정점을 연결하는 경로중에서 변의 cost를 합한 총계가 가장 적은 경로를 구하는 최단 경로 알고리즘의 일종이다. Dijkstra 알고리즘은 출발점과 각 정점 사이의 최단 경로를 출발점에서 가까운 것에서부터 하나씩 확정해 나간다. 이 알고리즘으로 최단 경로를 구하는 순서는 다음과 같다.

① 시작 노드에서 다른 모든 노드까지의 cost를 계산한다.
② 시작 노드에서 도달 가능한 모든 노드 중에서 도달하는데 걸리는 cost가 가장 적은 노드를 선택한다.
③ ②에서 선택된 노드에서 다른 모든 노드까지의 cost를 계산한다.
④ 시작 노드에서 다른 노드로 바로 가는 cost와 가장 싼 경로를 통해서 가는데 드는 비용을 비교한다.
⑤ 다른 노드로 가는 cost가 가장 싼 경로를 선택한다.
⑥ 이상의 단계들을 반복한다.

3. 데이타 관리자의 설계

3.1 전체적인 구조

이동체 통제 시스템을 위한 데이타 관리자는 크게 두 계층 즉, 데이타 관리 계층과 데이타 저장 계층으로 구성되며 그 전체적인 구조는 그림 3.1과 같다.

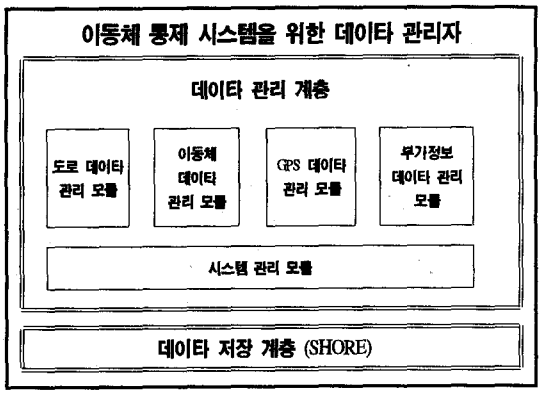

그림 3.1 : 데이타 관리자의 구조

데이타 관리 계층은 크게 도로 데이타 관리 모듈, 이동체 데이타 관리 모듈, GPS 데이타 관리 모듈, 부가정보 데이타 관리 모듈, 시스템 관리 모듈로 구성된다. 이들 모듈은 이동체 통제 시스템이나 자동차 항법 시스템에서 이동체의 현재 위치를 화면에 나타내거나 이동체의 최단 경로를 추출하는 등의 기능을 구현할 때 필요한 도로 데이타, 이동체 데이타, GPS 데이타, 부가정보 데이타 등을 다루기 위한 함수들과 하부 데이타 저장 계층인 SHORE 저장 관리자를 초기화하거나 종료시키는 함수들로 구성된다.

본 논문에서 제안한 데이타 관리자는 API (Application Programming Interface)로써 호출가능한 함수들을 사용하였다.

3.2 시스템 관리 모듈의 설계

시스템 관리 모듈에 포함된 함수들은 데이타 관리자의 하위 계층인 SHORE 저장 관리자를 초기하고 응용 프로그램이 종료될 때 SHORE 저장 관리자를 종료하는 기능을 수행한다.

본 논문에서 사용하는 SHORE 저장 관리자의 초

기화와 종료시에 가장 핵심적인 부분은 ss_m이라는 클라스이다. ss_m 클라스의 한 인스턴스를 생성하는 것이 SHORE 저장 관리자를 구동시키는 역할을 한다. 그리고, ss_m 클라스의 인스턴스를 제거하면 SHORE 저장 관리자는 종료된다.

초기화 함수는 SHORE 저장 관리자를 구동시키기 위해서 그림 3.2와 같은 선언을 포함하고 있다.

```
ssm = new ss_m( );
```

그림 3.2 클라스 ss_m의 인스턴스 생성

그림 3.2는 클라스 ss_m의 한 인스턴스 ssm을 생성하는 선언문이고, 이 선언문으로 인해 SHORE 저장 관리자가 구동된다. 그러면 SHORE 시스템은 구성 옵션(configuration option)들을 처리하고, 마지막으로 저장 디바이스를 포맷하고 마운트한다.

본 논문의 시스템 초기화 함수에서는 위에서 열거한 저장 관리자 구동, 구성 옵션 처리등의 작업을 사용자가 하나의 함수를 호출해서 모두 수행할 수 있도록 하였다.

종료 함수는 초기화 함수에서 SHORE 저장 관리자를 구동시키기 위해서 선언한 ss_m이라는 클라스의 인스턴스인 ssm을 제거한다. 그림 3.3은 ss_m 클라스의 인스턴스인 ssm을 제거하는 코드이다.

```
delete ssm;
```

그림 3.3 인스턴스 ssm의 제거

3.3 도로 데이타 관리 모듈의 설계

도로 데이타 관리 모듈은 도로 데이타를 관리하는 함수들로 구성된다. 도로 데이타 관리 함수는 SHORE 저장 관리자에 접근하여 도로 데이타의 삽입, 삭제, 갱신, 검색 등의 연산을 수행하도록 설계되었다. 그림 3.4는 본 논문에서 정의한 도로 데이타인 wdata_t의 자료 구조이다.

```
typedef struct {
    int w_id;
    int x1, y1, x2, y2;
    int node1, node2, cost;
    int *linkedwayid;
} wdata_t;
```

그림 3.4 : 도로 데이타의 자료 구조

도로 데이타의 자료 구조에서 w_id는 도로의 고유 ID를 의미한다. 그리고, x1, y1, x2, y2는 각각 도로의 시작 좌표와 끝 좌표를 의미하고, node1과 node2는 도로의 시작 노드와 끝 노드이다. cost는 해당 도로가 갖는 비용을 의미하고, linkedwayid는 해당 도로에 연결된 도로의 ID를 가지도록 설계하였다. 이것을 첨가한 이유는 한 도로에 연결된 도로가 n개가 될 수 있기 때문이다.

3.4 이동체 데이타 관리 모듈의 설계

이동체 데이타 관리 모듈은 이동체 데이타를 관리하는 함수들로 구성된다. 이동체 데이타를 관리하는 함수들은 응용 시스템에서 관리하는 이동체 데이타의 삽입, 삭제, 갱신, 검색 등의 연산을 수행하도록 설계되었다. 이동체 데이타 관리 모듈에서 사용하는 이동체 데이타는 vdata_t라는 구조체형 자료 구조로 설계하였다. vdata_t의 자료 구조는 그림 3.5와 같다.

```
typedef struct {
    int  v_id;
    char name[20];
    char owner[10];
    char year[4];
    char etc[40];
} vdata t;
```

그림 3.5 : 이동체 데이타의 자료 구조

이동체 데이타의 자료 구조에서 v_id는 이동체의

고유한 ID이고, name은 이동체의 이름을 나타낸다. owner는 이동체를 소유한 사람의 이름을 나타내고, year는 이동체의 생산년도를 의미하며, etc는 이동체에 관한 기타 정보를 갖는다.

3.5 GPS 데이타 관리 모듈의 설계

GPS 데이타 관리 모듈은 각 이동체로부터 위치 정보가 들어오면 이 정보를 시간별로 체계적으로 관리하는 함수들로 구성된다. 이동체에서는 시스템으로부터 실시간으로 위치정보가 들어온다고 가정하였지만 이동체 통제 시스템에서는 이러한 자료의 저장과 검색이 필요하므로 GPS 데이타 관리 모듈이 본 논문에서 포함된 것이다. GPS 데이타 관리 모듈은 이동체의 위치 데이타를 저장하고 필요한 경우 검색이 가능하도록 하여 도로 데이타 관리 모듈에서 최단 경로를 생성할 때 참조할 수 있도록 하였다. 이 GPS 데이타는 gdata_t라는 자료 구조로 그림 3.6과 같이 설계하였다.

```
typedef struct {
    int sec;
    int min;
    int hour;
    int day;
    int mon;
    int year;
    int xpos;
    int ypos;
    int vel;
    int height;
} gdata t;
```

그림 3.6 : GPS 데이타의 자료 구조

GPS 데이타의 자료 구조에서 sec, min, hour는 각각 데이타가 들어온 시간을 나타내기 위한 것으로 각각 초, 분, 시를 나타낸다. day는 날짜를 나타내고, mon은 몇월인지를 나타내며, year는 년도를 나타낸다. 또한, xpos와 ypos는 현재 위치를 나타내고, vel은 속도를 나타내고, height는 현재 고도를 나타낸다.

3.6 부가정보 데이타 관리 모듈의 설계

부가정보 데이타 관리 모듈은 이동체 데이타 관리 모듈이나 도로 데이타 관리 모듈에서 처리하지 않는 나머지 모든 데이타를 관리한다. 이 모듈에서는 여러 가지 자료 구조의 데이타를 다루기 때문에 데이타 타입이 다른 모듈보다 더 일반적으로 설계되었다. 그림 3.7은 부가정보 데이타를 저장하는 idata_t의 자료 구조이다.

```
typedef struct {
    int infosortid;
    int infoid;
    char key[20];
    char *infotext;
} idata t;
```

그림 3.7 : 부가정보 데이타의 자료 구조

부가정보 데이타의 자료 구조에서 infosortid는 정보의 종류마다 고유한 ID이다. 본 논문에서 사용되는 모든 정보가 각각의 세부적인 정보들을 가질 수 있으므로 infoid는 그 세부적인 정보들을 구별하기 위한 ID이다. 그리고, key는 정보의 검색을 위한 키값인데 이것을 사용하여 부가정보 데이타 관리 모듈은 부가정보의 삽입이나 삭제가 이루어진 경우에 새로 B-Tree 인덱싱을 하고, 특정 정보를 찾아야 할 때마다 이 키값을 기준으로 검색을 하게 된다. 마지막으로 infotext는 정보가 저장되는 부분이다.

4. 데이타 관리자의 구현

본 논문에서 제안하는 이동체 통제 시스템을 위한

데이타 관리자는 API로써 호출가능한 함수들로 구현되어 있기 때문에 호스트 언어인 C++에서 쉽게 불러 사용할 수 있다.

함수들의 이름은 다음의 규칙을 가진다. 모든 함수의 첫자는 그 함수가 어떤 정보를 처리하기 위한 함수인지를 나타낸다. 즉, 이동체 데이타 관리 함수는 함수는 'V'로 시작해서 vehicle에 관련된 함수임을 나타냈고, 도로 데이타를 처리하는 함수는 'W'로, 부가정보를 처리하는 함수는 'I'로 시작해서 각각 도로(way), 부가정보(information)에 관련된 함수임을 나타냈다. 그리하여, 본 논문에서 구현한 데이타 관리자는 함수의 이름만 보아도 어떤 모듈에 속한 함수인지를 알 수 있도록 하였다.

또한 모든 함수의 반환 값은 항상 'int'형으로 하였고, 함수 수행후 발생한 에러 값을 반환하도록 하였다. 본 논문에서 에러 코드는 각각의 함수를 설명할 때 언급된다. 만약 아무런 에러가 발생하지 않은 경우에는 mOK가 반환된다.

4.1 구현 환경

본 논문에서는 SunSparc Station을 사용하고 운영체제로는 SunOS 4.1.3을 사용하였다. 또한, 호스트 언어인 C++ 컴파일러로는 GNU에서 개발한 GCC 2.6.3 버젼을 사용하였다. GCC 2.6.3 버젼을 사용한 이유는 위스콘신 대학에서 SHORE를 개발할 때 사용한 버젼이기 때문이다.

현재 SHORE 0.9.3 베타 버젼에는 하나의 디바이스에 하나의 볼륨(volume)만 쓸 수 있다는 제한 사항이 있다. 그러나, 본 논문의 구현시에는 디바이스, 볼륨, 화일, 레코드의 순서로 구성된 SHORE의 계층적인 저장 구조에서 하나의 볼륨에 여러 개의 화일을 생성할 수 있는 SHORE 저장 관리자의 기능을 사용하여 도로 데이타, 이동체 데이타, 부가정보 데이타에 대해 각각 하나씩의 화일을 생성하였다. 그러므로, 하나의 디바이스에 하나의 볼륨만을 가질 수 있다는 제한 사항은 전혀 문제가 되지 않는다.

본 논문에서는 API를 사용하기 쉽게 하기 위하여 포인터의 사용은 가급적 피하고 참조를 이용하여 사용자가 직관적으로 사용할 수 있도록 하였다.

4.2 시스템 관리 모듈의 구현

시스템 관리 모듈은 SHORE 저장 관리자를 초기화하고 종료하는 기능을 수행하는 함수들을 포함하고 있다. 그림 4.1은 시스템 관리 모듈에 포함된 함수들이다.

```
int SStart(int argc, char **argv,
        bool dev_init, int dev_size);
int SFinish(void);
```

그림 4.1 : 시스템 관리 함수

SStart( ) 함수는 SHORE 저장 관리자를 초기화하는 기능을 수행한다. 이 함수의 첫번째 두인자는 응용 프로그램의 main 함수에서 사용된 argc, argv 두 인자를 받아들이는데 이것을 사용해서 응용 프로그램의 실행 프로그램 이름을 알아낸다. 이렇게 쓰는 이유는 SHORE에서 옵션을 구성할 때 응용 프로그램이 실행되는 화일의 이름을 지정해 주어야 하기 때문이다. 그리고 세번째 인자인 dev_init는 응용 프로그램을 실행할 때마다 저장 디바이스를 새로 포맷(format)할 것인지를 지정한다. 그리고 네번째 인자인 dev_size는 응용 프로그램에서 사용할 디바이스의 크기를 지정한다.

SFinish( ) 함수는 SHORE 저장 관리자를 종료하는 기능을 수행하는데 인자없이 사용된다.

4.3 도로 데이타 관리 모듈의 구현

도로 데이타 관리 모듈은 도로 데이타를 관리하는 함수들을 포함하고 있다. 그림 4.2는 도로 데이타 관리 모듈에 포함된 함수들이다.

WDatafileCreate( ) 함수는 도로 데이타를 저장하기 위한 화일을 생성하는 함수로서 인자없이 사용된다. 사용자가 도로 데이타를 관리하는 응용 프로그램을 작성한다면 이 함수는 반드시 한번은 실행되어야 한다. 이 함수를 실행할 때 남아 있는 저장 공간이 하나의 도로 데이타도 삽입할 수 없을 만큼 작다면 이

```
int WDatafileCreate(void)
int WDataInsert(wdata_t wdata, int& w_id);
int WDataDelete(int w_id);
int WDataUpdate(int w_id, wdata_t wdata);
int WDataGet(int w_id, wdata_t& wdata);
int WPathGet(int sw_id, int ew_id,
             wpath_t wpath);
int WUserPathInsert(int v_id, int sw_id,
                    int ew_id, wpath_t wpath);
int WUserPathGet(int v_id, wpath_t& wpath);
```

그림 4.2 : 도로 데이타 관리 함수

함수는 mNODATAVOL로 정의된 에러 코드 값을 반환한다.

WDataInsert( ) 함수는 도로 데이타를 SHORE 저장 관리자에 삽입하는 함수이다. 이 함수의 첫번째 인자는 wdata_t형의 도로 데이타이고, 두번째 인자는 int형의 도로 ID이다. 따라서 이 함수를 사용하여 도로 데이타를 SHORE 저장 관리자에 저장하면, 그 도로의 ID를 반환받게 된다. 삽입하려는 도로 데이타를 삽입할 수 없을 경우에 이 함수는 mNODATAVOL이라는 에러 코드를 반환한다.

WDataDelete( ) 함수는 도로 데이타를 SHORE 저장 관리자에서 삭제하는 함수이다. 따라서 이 함수를 사용하여 도로의 ID를 주면 그 ID에 해당하는 도로를 SHORE 저장 관리자에서 삭제할 수 있다. 지정한 도로 데이타의 ID가 없을 경우에 이 함수는 mNOWAY라고 정의된 에러 코드를 반환한다.

WDataUpdate( ) 함수는 SHORE 저장 관리자에 저장되어 있는 도로 데이타를 갱신하는 함수이다. 이 함수의 첫번째 인자는 도로 ID이고 두번째 인자는 갱신할 새로운 도로 데이타이다. 따라서 이 함수를 사용하면 원하는 ID를 가진 도로에 관련된 데이타를 새로운 값으로 갱신할 수 있다. 이 함수는 지정한 도로 데이타의 ID가 없을 경우에는 mNOWAY라는 에러 코드를 반환하고, 저장 공간 부족으로 인하여 갱신하려는 새로운 데이타를 저장할 수 없을 경우에는 mNODATAVOL이라는 에러 코드를 반환한다.

WDataGet( ) 함수는 SHORE 저장 관리자에 저장되어 있는 도로 데이타를 검색하는 기능을 수행한다. 따라서 사용자는 이 함수를 이용하여 SHORE에 저장되어 있는 데이타를 검색할 수 있다. 이 함수의 첫번째 인자인 w_id는 도로의 ID를 의미하고, wdata는 wdata_t형인 도로 데이타를 나타낸다.

WPathGet( ) 함수는 도로 데이타 관리 모듈에 의해 저장된 모든 도로 데이타와 GPS 데이타 관리 함수를 통해 특정 이동체의 위치 데이타를 참조하여 시작 도로를 나타내는 sw_id에서 끝 도로를 나타내는 ew_id까지의 최단 경로를 wpath를 통하여 반환한다.

WUserPathInsert( ) 함수는 추후에 움직였던 데이타가 필요한 경우에 이동체가 움직인 경로를 저장하는 함수이다. 사용자는 이 함수를 이용하여 이동체가 자주 가는 길을 저장하여 경로 안내시 이동체가 자주 가는 경로에 대한 안내를 할 수 있다.

WUserPathGet( ) 함수는 SHORE 저장 관리자에 저장되어 있는 이동체의 이동 경로를 검색한다. 사용자는 이 함수를 이용하여 쉽게 이동체의 경로 데이타를 참조할 수 있다.

4.4 이동체 데이타 관리 모듈의 구현

이동체 데이타 관리 모듈은 본 논문에서 정의된 vdata_t 형태의 이동체 데이타를 쉽게 SHORE 저장 관리자에 저장할 수 있도록 구현되었다. 이동체 데이타 관리 모듈의 함수들을 그림 4.3에 나타나 있다. 그림 4.3에서 v_id는 이동체의 고유한 ID를 나타내고, vdata는 이동체 데이타를 의미한다.

```
int VDatafileCreate(void);
int VDataInsert(vdata_t vdata, int& v_id);
int VDataGet(int v_id, vdata_t& vdata);
int VDataUpdate(int v_id, vdata_t vdata);
int VDataDelete(int v_id);
int VPosfileCreate(void);
int VPosInsert(int v_id, int x, int y);
int VPosGet(int v_id,. int& x, int& y);
int VPathGet(int v_id, vpath_t& vpath);
int VPathReset(int vid);
```

그림 4.3 : 이동체 데이타 관리 함수

VDatafileCreate( ) 함수는 처음 이동체 데이타를 저장하기 전에 이동체 데이타를 위한 화일을 생성하는 함수로 인자없이 사용된다. 사용자가 이동체 데이타를 관리하는 응용 프로그램을 작성한다면 이 함수는 반드시 한번은 실행되어야 한다. 이 함수를 실행할 때 저장 공간이 하나의 데이타 레코드도 삽입할 수 없을 정도로 작다면 이 함수는 mNODATAVOL로 정의된 에러 코드 값을 반환한다.

VDataInsert( ) 함수는 이동체 데이타를 저장 관리자에 저장할 때 사용되는 함수이다. 이 함수는 vdata와 v_id를 인자로 가지는데 vdata는 입력되는 인자로서 이동체 데이타를 나타내고 v_id는 반환되는 값으로 이동체의 고유한 ID이다. 이 함수가 실행될 때 저장 공간이 없어서 삽입한 데이타를 저장할 수 없다면 이 함수는 mNODATAVOL라는 정의된 에러 코드를 반환한다.

VDataGet( ) 함수는 v_id와 vdata를 인자로 가지는데 v_id는 입력되는 인자이고 vdata는 반환되는 인자이다. 따라서 이동체의 ID를 첫번째 인자로 주면 두번째 인자를 통해 원하는 이동체 데이타를 얻을 수 있다. 지정한 v_id 값이 없다면 mNOVEHICLE라고 정의된 에러 코드를 반환한다.

VDataUpdate( )는 이미 저장 관리자에 저장된 이동체 데이타를 갱신할 때 사용되는 함수이다. 이 함수는 v_id와 vdata를 인자로 가지는데 자료를 갱신할 이동체 ID를 첫번째 인자로 주고 vdata를 두번째 인자로 주면 원하는 이동체 데이타를 갱신할 수 있다. 이 함수는 저장 공간이 부족하여 새로 갱신될 데이타를 저장할 수 없을 경우에는 mNODATAVOL이라는 에러를 반환하고, 지정한 이동체 ID가 없는 경우에는 mNOVEHICLE라고 정의된 에러 코드를 반환한다.

VDataDelete( )는 저장되어 있는 이동체에 관한 데이타를 삭제하는 함수이다. 이 함수는 이동체 ID를 인자로 가지므로 사용자가 특정 이동체에 관한 데이타를 삭제하기 위해서는 그 이동체의 ID를 이 함수의 인자로 주면 된다. 만약, 지정한 v_id 값이 없다면 mNOVEHICLE라고 정의된 에러 코드가 반환된다.

VPosfileCreate( )는 이동체 데이타 관리 모듈에서 이동체의 위치 데이타를 관리하기 위한 별도의 화일을 생성하는 함수이다. 이 함수는 VPosInsert( )을 사용하기 전에는 반드시 수행되어야 하며, 만일 이 함수가 수행되기 전에 VPosGet( )함수가 수행되면 mNOVPOSFILE로 정의된 에러 코드가 반환된다.

VPosInsert( )는 자동차 항법 시스템으로부터 얻은 이동체의 위치를 저장하는 함수이다. 자동차 항법 장치에서 얻은 데이타는 수신기에 따라 서로 다른 데이타 포맷으로 되어 있을 수 있으므로 본 논문에서는 사용자가 직관적으로 알기 쉬운 x, y좌표로 저장한다. 만일 VPosfileCreate( )가 수행되기 이전에 이 함수가 수행되면 이 함수는 mNOVPOSFILE로 정의된 에러 코드를 반환한다.

VPosGet( )는 이동체의 현재 위치를 얻는 함수이다. 이것은 실시간으로 갱신되는 GPS의 위치 정보를 사용자에게 편리한 형태인 x, y좌표로 변환하여 반환한다. 만일 잘못된 이동체의 ID가 입력되면 mNOVEHICLE로 정의된 에러 코드가 반환된다.

VPathGet( )는 이동체가 이동한 경로를 반환하는 함수이다. 이때 특정 이동체의 v_id를 지정할 수 있으며 자료의 반환은 vpath_t 형의 변수를 통해서 이루어진다. 만일, 잘못된 이동체 ID가 입력되면 이 함수는 mNOVEHICLE로 정의된 에러 코드를 반환한다.

VPathReset( )은 이전까지 저장된 이동체의 이동한 경로를 삭제하고 새로운 이동 경로를 저장하는 함수이다. 따라서 이 함수를 사용하면 특정 이동체의 위치를 필요한 시간부터 추적할 수 있다.

4.5 GPS 데이타 관리 모듈의 구현

GPS 데이타 관리 모듈에는 실시간으로 들어오는 각 이동체의 위치 데이타를 종합적으로 저장 관리하기 위한 함수들이 포함되어 있다. 그림 4.4는 GPS 데이타 관리 모듈의 함수들을 보여준다.

GDatafileCreate( ) 함수는 GPS 데이타를 관리하기 위한 화일을 생성하는 함수로서 인자인 gf_id를 통해 이 함수에 의해 새로 생성된 GPS 데이타의 화일의 ID를 반환한다. 응용 프로그램이 GPS 데이타를 처리해야 한다면 이 함수를 반드시 한번은 실행시켜야 한다. 다른 모듈의 화일 생성 함수와는 달리 gf_id를 반

```
int GDatafileCreate(int& gf_id);
int GDatafileDelete(int gf_id);
int GDateInsert(int gf_id, gdata_t gdata,
                int& g_id);
int GDataDelete(int gf_id, int g_id);
int GDataGet(int gf_id,
             gtime_t sgtime, gtime_t etime,
             gdatalist_t gdatalist);
```

그림 4.4 : GPS 데이타 관리 함수

환하도록 한 이유는 GPS 데이타는 실시간으로 들어오는 것이기 때문에 일정 간격으로 삭제하지 않을 경우 방대한 양의 저장 볼륨이 필요하기 때문이다. 그러므로, 필요할 경우에는 여러 개의 화일로 나누어 관리할 수 있도록 하였다.

GDatafileDelete( ) 함수는 GPS 데이타의 화일을 삭제하기 위한 함수로서 인자는 삭제하려는 화일의 ID인 gf_id이다.

GDataInsert( ) 함수는 GPS 데이타를 GPS 데이타 화일에 삽입하는 기능을 한다. 이 함수의 첫번째 인자는 gf_id인데 여러 개의 GPS 데이타 화일이 열려 있을 경우에 어느 GPS 화일에 데이타를 삽입할 것인지를 지정할 수 있도록 하였다. 이 함수에 의해 첫번째 생성된 GPS 데이타 화일의 ID는 1이다. 따라서 모든 GPS 데이타 화일 ID는 1보다 크다. 만약 사용자가 GPS 데이타 화일 ID를 0으로 지정하였다면 이 함수는 가장 최근에 만들어진 GPS 데이타 화일에 GPS 데이타를 삽입한다. 또한 지정한 GPS 데이타 화일이 없다면 mNOGPSFILE로 정의된 에러 코드 값이 반환된다.

GDataDelete( ) 함수는 특정한 GPS 데이타 화일에서 하나의 GPS 데이타를 삭제하는 함수이다. 이 함수의 첫번째 인자인 gf_id는 여러개의 GPS 데이타 화일중에서 삭제할 GPS 데이타가 포함된 화일을 지정하는 것이고, 두번째 인자인 g_id는 특정한 GPS 데이타 화일에 포함된 하나의 GPS 데이타의 ID를 나타낸다. 이 함수를 사용할 때 gf_id 값이 존재하는 화일에 대한 것이 아닌 경우에 mNOGPSFILE로 정의된 에러 코드 값이 반환된다.

GDataGet( ) 함수는 특정한 GPS 데이타 화일에서 일정 시간 동안의 GPS 데이타를 검색하는 함수이다.

이 함수의 첫번째 인자인 gf_id는 여러개의 GPS 데이타 화일중에서 검색할 GPS 데이타 화일을 지정하기 위해서 사용되는 인자이다. 그리고, 뒤에 나온 gtime_t형의 sgtime과 egtime은 GPS 데이타를 검색할 시간 구간을 지정해주는 인자들이고, 마지막에 나온 gdatalist_t형의 gdatalist는 GPS 데이타를 리스트의 형태로 반환받기 위한 인자이다.

```
int IClassCreate(char *classname, int& ic_id);
int IClassDelete(int ic_id);
int IDataInsert(int ic_id, idata_t idata, int& i_id);
int IDataDelete(int ic_id, int i_id);
int IDataUpdate(int ic_id, int i_id, idata_t idata);
int IDataGet(int ic_id, int i_id, idata_t& idata);
```

그림 4.5 : 부가정보 데이타 관리 함수

4.6 부가정보 데이타 관리 모듈의 구현

부가정보 데이타 관리 모듈은 이동체 통제 시스템이나 자동차 항법 시스템과 같은 응용 시스템에서 별도의 정보를 제공하기 위한 함수들이 포함되어 있다.

그림 4.5는 부가정보 데이타 관리 모듈의 함수들이다.

IClassCreate( ) 함수는 사용자가 필요한 정보의 클라스를 생성할 때 사용되는 함수이다. 이 함수는 첫번째 인자로 클라스의 이름을 제공하면 두번째 인자로 클라스의 ID를 반환한다.

IClassDelete( ) 함수는 생성되어 있는 클라스를 삭제하는 함수이다. 이 함수에 포함된 인자는 이미 생성되어 있는 클라스의 ID를 가지는데 만일 존재하지 않는 ic_id가 입력되면 이 함수는 mNOSUCHCLASS로 정의된 에러 코드를 반환한다.

IDataInsert( ) 함수는 특정 클라스에 데이타를 삽입하고 데이타의 ID를 반환받는 함수이다. 이 함수의 첫번째 인자는 ic_id인데 이 값으로 데이타를 저장할 클라스를 지정한다. 두번째 인자는 idata인데 저장할 부가정보 데이타를 이 인자에 넣어 주면 지정된 클라스에 저장된다. 세번째 인자는 i_id인데 이 인자는 부

가정보 데이타를 입력한 후 정보의 ID를 반환받기 위하여 필요한 인자이다.

IDataDelete( ) 함수는 부가정보 데이타를 삭제하기 위하여 사용되는 함수이다. 이 함수의 첫번째 인자는 부가정보의 클라스의 식별자인 ic_id이고, 두번째 인자는 부가정보 데이타의 식별자인 i_id이다. 이렇게 구별한 이유는 클라스마다 다른 클라스와는 무관한 부가정보 데이타의 식별자가 존재하기 때문이다.

IDataUpdate( ) 함수는 이미 저장되어 있는 부가정보 데이타를 새로운 데이타로 갱신하고자 할 때 사용되는 함수이다. 이 함수의 첫번째 인자는 갱신하려는 데이타가 포함된 클라스의 ID이고, 두번째 인자는 부가정보의 ID이며, 세번째 인자는 새로운 부가정보 데이타이다.

IDataGet( ) 함수는 저장 관리자에 저장되어 있는 부가정보를 응용 프로그램에서 검색하기 위해서 사용되는 함수이다. 이 함수의 첫번째 인자는 클라스의 ID, 두번째 인자는 부가정보 데이타의 식별자, 세번째 인자는 부가정보 데이타이다. 이 함수를 사용하기 위해서는 첫번째, 두번째 인자에 각각 부가정보 데이타가 포함되어 있는 클라스의 ID와 부가정보 데이타의 ID를 지정하고, 세번째 있는 인자에 idata_t형으로 정의된 변수를 넣어주면 이 변수에 부가정보 데이타가 반환된다.

5. 결론

본 논문에서는 이동체 통제 시스템을 위한 데이타 관리자를 설계하고 구현하였다. 본 논문은 하부 구조에 SHORE 저장 관리자를 사용하고 있는데, SHORE는 앞서 설명한 것처럼 peer-to-peer 통신을 지원하고, 또한 사용자가 자신의 목적에 적합한 value-added 서버의 구현을 가능하게 한다.

본 논문에서 설계하고 구현한 데이타 관리자는 SHORE 저장 관리자에 이동체 데이타, 도로 데이타, GPS 데이타, 부가정보 데이타 등을 저장할 수 있는 API(Application Programming Interface)를 제공한다. 따라서 본 논문에서 구현한 데이타 관리자를 내부적으로 사용하면 이동체 통제 시스템을 쉽게 구현할 수 있으며, SHORE 저장 관리자를 사용하기 때문에 데이타를 효과적으로 관리할 수 있다.

그러나, 본 논문에서는 서버간의 통신을 위한 일반적인 API를 제공하지 않기 때문에 본 논문을 사용하여 응용 시스템을 작성할 때 peer-to-peer 통신을 가능하게 하기 위해서는 해야 될 일이 많다는 단점이 있다.

앞으로의 연구과제는 이동체 통제 시스템을 위한 데이타 관리에 관해 계속적으로 연구하고2),14), 또한 본 논문에서 제안한 데이타 관리자에 더 많은 기능을 추가하여 API를 확장하면서 실제 이동체 통제 시스템을 구현하는 것이다.

6. 참고문헌

1. E.G. Blackwell, "Overview of Differential GPS Methods," Global Positioning System, Vol. 3, 1996, pp. 89-100.
2. R.A. Cass, "Building Navigable Database for the Real World," Proc. of Vehicle Navigation and Information System Conf., 1992, pp. 585-589.
3. D.J. Dewitt, N. Kabra, J. Luo, J.M. Patel, and J. Yu, "Client-Server Paradise," Proc. of 20th VLDB Conf., Sep. 1994, pp. 558-569.
4. E. Dijkstra, "A Note on Two Problems in Connexion with Graphs," Numeric Mathematics, No. 1, 1959, pp. 269-171.
5. M.J. Egenhofer, "What's Special about Spatial? Database Requirements for Vehicle Navigation in Geographic Space," Proc. of ACM SIGMOD, Jun. 1993, pp. 398-402.
6. ESS Project, Paradise - A Parallel Information System for EOSDIS, 1995.
7. F. Iwaki, M. Kakihara and M. Sasaki, "Recognition of Vehicle's Location for

Navigation," Proc. of Vehicle Navigation and Information Systems Conf., 1989, pp. 131-138.
8. R.K. Jurgen "Smart Cars and Highways Go Global," IEEE Spectrum, 1991, pp. 635-643.
9. R. Laurini and D. Thompson, Fundamentals of Spatial Information System, Academic Press, 1992.
10. T. Logsdon, The Navstar Global Positioning System, Van Nostrand Reinhold, 1992.
11. T. Logsdon, Mobile Communication Satellites, McGraw-Hill, 1995.
12. C.B. Medeiros and Pires, F., "Database for GIS," SIGMOD RECORD, Vol. 23, No.1, Mar. 1994, pp. 107-115.
13. N. Ness and M. Herbert, "A Prototype Low Cost In-Vehicle Navigation System," Proc. of Vehicle Navigation and Information Systems Conf., 1993, pp. 56-59.
14. H. Ohnishi, I. Ogawa and F. Morise, "Map Database Generation for In-Vehicle Navigation System," Proc. of Vehicle Navigation and Information Systems Conf., 1994, pp. 607-612.
15. J.H. Rillings and J. W. Lewis, "TravTek," Proc. of IEEE Vehicle Naviagation and Information Systems Conf., 1992, pp. 729-737.
16. The Shore Project Group, An Overview of Shore, May 1995.17. The Shore Project Group, Writing Value-Added Servers with the Shore Storage Manager, May 1995.
18. A. Stevens and D.K. Martell, "Development and Evaluation of the Trafficmaster Driver Information System," Proc. of Vehicle Navigation and Information Systems Conf., 1993, pp. 251-258.
19. D. Wells, Guide to GPS Positioning, Canadian GPS Associates, 1987.
20. R. Whelan, Smart Highways, Smart Cars, Arttech House, 1995.
21. 성태경, "자동차 항법 시스템," Proc. of 2th GPS Workshop, Nov. 1995, pp. 263-271.
22. 오병우, 한기준, "지리 정보 시스템을 위한 사용자 인터페이스," 한국정보과학회, 제13권 3호, 1995, pp. 18-29.
23. 이종훈, 강태호, 김진서, "GPS를 이용한 자동차용 주행안내 시스템 프로토타입 개발," 한국지형공간정보학회, GPS, CNS, 지하매설물에 관한 워크샵, 1995, pp. 34-40.