모바일 응용을 위한 웹 맵 서비스 확장 인터페이스의 설계 및 구현†

Design and Implementation of Extended Web Map Service Interfaces for Mobile Applications

조대수*, 오병우**

Dae-Soo Jo, Byung-Woo Oh

요 약 최근 웹을 통해 지리정보를 공유하기 위해서 표준 인터페이스를 가지는 웹 맵 서비스(WMS, Web Map Service)가 개발되고 있다 특히, OGC(Open Geospatial Consortium)에서는 웹 맵 서비스를 위한 개방형 인터페이스 명세를 발표하고, ISO 표준으로 채택되기 위한 절차를 진행하는 등 가장 활발한 활동을 보이고 있다. 그러나 현재 OGC에서 정의하고 있는 WMS 명세는 모바일 환경에서의 응용 프로그램 작성에 몇 가지 문제점을 가지고 있다. 유선 인터넷 환경과 달리 모바일 환경에서의 응용에서 맵 서버는 클라이언트로 전송되는 데이터의 크기를 최소화할 필요가 있다. 왜냐하면, 데이터의 크기는 전송 비용뿐 아니라, 맵 클라이언트의 응답시간에 매우 중요한 요소이기 때문이다. 또한 모바일 응용에서는 모바일 디바이스의 화면 크기에 따라서 적절한 품질의 맵을 제공해야하며, 응용 분야에 특화된 질의를 지원할 필요가 있다. 이 논문에서는 OGC에서 제안한 표준 인터페이스를 수용하면서, 모바일 응용을 위한 요구조건을 반영하기 위해서 모바일 웹 맵 서비스(Mobile Web Map Service, M-WMS)를 위한 인터페이스를 제안한다. 또한 모바일 맵 서비스를 지원하기 위한 맵 서버를 설계하고, 구현하였다. 이 논문은 현재 관심이 집중되고 있는 LBS, 텔레매틱스 등 다양한 모바일 응용에서 적용 가능한 웹 맵 서버를 생성하고 사용하는데 기여하고 있다.

Abstract Recently, the web map services with standard interfaces have been developed to a high degree for the purpose of sharing of spatial data through the web. Among the various kinds of works related, the specification for the web map service released by Open Geospatial Consortium (OGC) is the most prominent, and is nearly adopted as international standard (ISO/DIS 19128) in developing the web map servers. The web map server of OGC, however, provides the insufficient capabilities for mobile applications. Unlike the wired applications, the mobile applications would require the map server to minimize of size of data transferred, because the size of data is very important factor in communication cost and the response time of map clients. And the mobile applications require different quality of map according to the screen size of mobile devices. The mobile application, also require some application specific queries. In this paper, We have proposed the interfaces for mobile web map services (M-WMS) which fully comply with the standard interfaces proposed in OGC. And we have designed and implemented the web map server for mobile map services. This paper has contributed to construction and practical use of the web map servers in mobile applications, such as LBS and telematics.

주요어 : 웹 GIS, 웹 맵 서비스, 웹 맵 서버, 개방형 GIS

KeyWords : Web GIS, Web Map Services, Web Map Server, Open GIS

1. 서 론

현재 GIS(Geographic Information System)는 공공 및 민간 부문의 광범위한 분야에서 널리 사용되고 있다. 최근 인터넷을 통한 정보시스템의 요구가 급증하고 있으며, 이에 따라 GIS 분야에서도 전통적인 클라이언트/서버 구조의 시스템에서 웹 브라우저를 활용한 웹 기반의 시스템으로 발전되고 있다[1][2][3]

* 동서대학교 인터넷공학부 전임강사 dscho@dongseo.ac.kr.

** 금오공과대학교 컴퓨터공학부 조교수 bwoh@kumoh.ac.kr.

[4][5][6]. 특히, 웹의 특징인 조작의 간편성, 뛰어난 접근성 등으로 인해 웹을 통해 지리정보를 공유, 활용하기 위한 웹 맵 서비스에 대한 요구가 증가하고 있다. 웹 맵 서비스(WMS, Web Map Services)란 기존의 GIS 전용 클라이언트 대신에 웹 브라우저를 사용하여 지리정보를 검색, 획득하기 위한 서비스를 의미한다. 이러한 WMS는 기존의 클라이언트/서버 구조의 GIS에서의 클라이언트를 웹 브라우저로 대체하여 사용한다는 것 이상의 의미를 지닌다. 왜냐하면, 사용이 간편한 웹 브라우저를 통한 맵 서비스는 GIS 비전문가들도 쉽게 지리정보를 획득할 수 있는 장점을 가지므로, GIS의 활용 분야가 획기적으로 넓어질 수 있기 때문이다.

GIS 분야 민간 컨소시엄인 OGC(Open Geospatial Consortium)에서는 WMS를 위한 개방형 인터페이스를 제안하기 위해서 'Web Map Service' 표준 명세[11]를 발표하고 있다. 개방형 인터페이스의 목적은 서로 다른 맵 서버간의 상호운용성(interoperability)을 지원함으로서, 지리정보의 공유와 활용을 극대화하는데 있다. 만약, 각각의 맵 서버가 자신만의 고유한 인터페이스(proprietary interface)를 갖는다면, 각각의 맵 서버에 대해서 서로 다른 맵 클라이언트를 요구하기 때문에, ActiveX, Plug-ins, Java Applet, Script 등의 도움 없이 웹 브라우저의 기능만으로는 맵 서비스를 제공받기 어렵다.

WMS는 사용하기 쉬운 웹 브라우저를 사용하기 때문에, GIS 분야의 초보자뿐 아니라, 전문가에 이르기까지 다양한 분야에서 널리 활용되고 있다. 그러나 현재 OGC에서 정의하고 있는 WMS 명세[11]는 모바일 환경에서의 응용 프로그램 작성에 몇 가지 문제점을 가지고 있다. 유선 인터넷 환경과 달리 모바일 환경에서의 응용에서 맵 서버는 클라이언트로 전송되는 데이터의 크기를 최소화할 필요가 있다. 왜냐하면, 데이터의 크기는 전송 비용뿐 아니라, 맵 클라이언트의 응답시간에 매우 중요한 요소이기 때문이다. 또한 모바일 응용에서는 모바일 디바이스의 화면 크기에 따라서 적절한 품질의 맵을 제공해하며, 응용 분야에 특화된 질의를 지원할 필요가 있다. 예를 들어, k-번째 근접한 POI(Point of Interest) 객체를 포함하는 맵을 요청하는 질의는 모바일 환경에서 주요 응용(Killer Application) 중의 하나인 LBS(Location-based Services)에서 매우 빈번히 사용되는 질의이다.

이 논문에서는 OGC에서 제안한 표준 인터페이스를 수용하면서, 모바일 응용을 위한 요구조건을 반영하기 위해서 웹 맵 서버를 위한 확장된 인터페이스를 제안한다. 이 논문의 구성은 다음과 같다. 2장에서는 OGC에서 제안하고 있는 WMS 명세를 살펴본다. 3장에서는 현재의 WMS 인터페이스의 문제점과 이에 대한 해결방안으로서 확장된 인터페이스를 제안한다. 4장에서는 이 논문에서 제안하는 확장된 인터페이스를 갖는 웹 맵 서버의 구현 내용을 기술하고, 5장에서 구현 결과를 보이다. 그리고 6장에서 결론과 향후 연구내용을 기술한다.

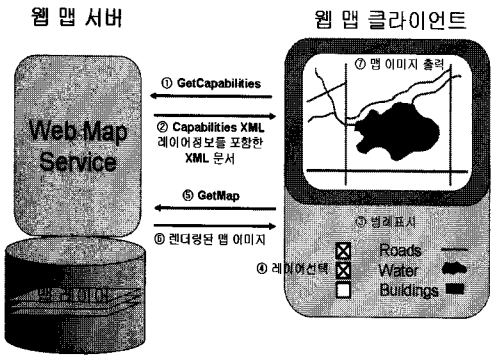
2. 관련 연구

OGC에서는 WMS를 위해서 다음과 같은 세 개의 연산자로 구성된 인터페이스를 정의하였다[11]. 현재 WMS 버전 1.3이 발표되었으며, 이 명세서는 ISO를 통해서 국제 표준으로 제정되기 위한 절차가 진행 중이다.

<표 1> WMS 인터페이스를 구성하는 세 가지 연산자

연산자	필수/선택	주요 내용
GetCapabilities	필수	해당 웹 맵 서버의 서비스 레벨의 메타데이터를 제공 웹 맵 서버가 제공하는 정보의 내용과 정보 요청 시 사용하는 매개변수들을 알려줌
GetMap	필수	맵 이미지(GIF, JPEG 등)를 제공 받음
GetFeatureInfo	선택	맵에 나타나는 특정 지형지물에 대한 정보 요청

WMS 명세에서는 맵을 요청(request)하기 위한 연산자에 대한 형식, 요청에 대한 응답(response)으로써 제공되어야 할 맵의 형식 또는 오류 발생 시 전달되는 문서의 형식에 대한 표준을 제시하고 있다. 웹 브라우저는 표준 인터페이스를 준수하는 모든 맵 서버에 대해서 동일한 방법으로 지리정보를 획득할 수 있으므로, 표준 인터페이스는 지리정보의 공유를 용이하게 한다. 클라이언트에서 맵을 요청할 경우에는 GetMap 연산자를 사용하여, 맵을 구성하는 레이어 리스트(Layers 매개변수), 스타일 리스트(Styles 매개변수), 공간참조체계(CRS 매개변수), 맵의 영역(BBOX 매개변수)을 맵 서버로 전달한다. 즉, 현재의 WMS 명세서에서는 맵을 구성하는 레이어와 영역(BBOX, Bounding Box)만이 맵을 검색하기 위한 수단으로 사용된다.



<그림 1> WMS 사용 시나리오

<그림 1>에서는 WMS를 사용하여 웹 맵 클라이언트에서 GetCapabilities 연산자와 GetMap 연산자를 사용해서 웹 맵 서버로부터 서비스 메타데이터 및 맵을 요청하는 시나리오를 보이고 있다.

① 웹 맵 클라이언트는 WMS의 GetCapabilities 연산자를 통해 해당 웹 맵 서버가 제공하는 서비스에 대한 메타 데이터를 요청한다.
② 웹 맵 서버는 레이어 정보를 포함한 서비스 메타데이터를 XML 문서로 웹 맵 클라이언트에게 전달한다.
③ 웹 맵 클라이언트는 서비스 메타 데이터를 분석해서 해당 웹 맵 서버가 제공할 수 있는 레이어 정보를 이용하여 범례를 표시한다.
④ 사용자는 웹 맵 클라이언트의 범례 정보를 통해 자신이 원하는 레이어를 선택한다. 단, 레이어에 대한 스타일 정보, 공간참조체계 정보, 영역 정보 등 맵을 요청할 때 필요한 정보들도 함께 선택한다.
⑤ 웹 맵 클라이언트는 사용자가 선택한 정보들을 WMS 인터페이스의 GetMap 연산자를 통해 웹 맵 서버로 전달한다.
⑥ 웹 맵 서버는 해당 GetMap 연산자를 통해 요청된 맵 이미지를 생성하여 웹 맵 클라이언트로 전달한다.
⑦ 웹 맵 클라이언트는 웹 맵 서버로부터 전달받은 맵 이미지를 화면에 출력한다.

3. 문제점 및 해결방안

이 장에서는 OGC에서 제안하고 있는 WMS에서 맵을 요청하는 방법이 모바일 응용을 개발할 때 발생하는 문제점을 기술하고, 이 논문에서 제안하는 해결 방안을 설명한다.

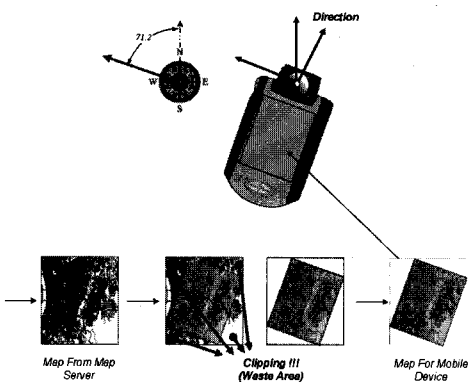
3.1 모바일 환경에서 대역폭을 고려한 WMS 검색 인터페이스 확장

OGC의 WMS 명세는 맵 서버로부터 전송되는 데이터의 크기를 고려하지 않는다. 전송되는 데이터의 크기는 통신비용 및 맵 클라이언트의 응답 시간과 매우 밀접한 관계가 있으며, 모바일 응용에서 가장 중요한 요구조건이다. 이 논문에서는 기존의 GetMap 요청에서 방향(DIRECTION 매개변수)과 크기의 한계(BOUND 매개변수)를 추가하였다.

DIRECTION 매개변수는 요청되는 맵의 각도를 기술한다. 현재의 WMS에서 맵 서버로부터 전송되는 모든 맵의 방향은 정북 방향으로 고정되어 있다. 그러나 PNS(Personal Navigation System)과 같은 모바일 응용에서는 정북 방향이 아닌 맵을 출력하는 경우가 빈번히 발생한다. 이러한 경우에 기존의 WMS 명세에 따르면, 사용자는 필요한 영역보다 더 넓은 영역에 대한 맵을 요청해야 하므로, 모바일 환경에서 대역폭의 낭비를 초래하게 되며, 정북 방향의 맵은 클라이언트에서 현재의 진행 방향에 따라 일정부분의 영역이 제거(clipping)되어 사용될 것이므로, 필요 없는 데이터 전송비용이 발생하는 문제점이 있다.

예를 들어 <그림 2>와 같이 사용자가 정북 방향이 아니라, 북동쪽 방향으로 진행하고 있을 경우에 맵 클라이언트는 자신이 실제로 필요한 영역을 포함하는 영역의 맵을 요청하게 된다. 왜냐하면, OGC의 WMS 인터페이스에서는 BBOX 매개변수만을 통해 정북방향의 사각형 영역의 맵만을 요청할 수 있기 때문이다. 이 경우에 불필요한 영역이 포함되어 전송되는 맵의 크기가 커지는 문제점이 있다. 또한 맵 클라이언트에서는 불필요한 영역을 제거(Clipping)해서 출력해야 하므로 자원의 낭비를 초래하는 문제점이 있다.

따라서 GetMap 연산자에 DIRECTION 매개변수를 추가하여, 클라이언트에서 실제로 사용되는 영역의 맵만을 요청하도록 기존의 인터페이스를 확장하였다. DIRECTION 매개변수의 값으로는 0~360까지가 가능하며, 정북방향을 기준으로 시계방향으로 자신의 진행 방향을 나타낸다.



<그림 2> DIRECTION 매개변수의 필요성: 맵 생성에서 낭비되는 영역의 최소화

BOUND 매개변수는 맵 서버로부터 전송되는 맵의 크기의 한계를 명시적으로 기술한다. 만약, 클라이언트가 BOUND 매개변수를 포함한 GetMap 요청을 전송할 경우에, 맵 서버는 이 매개변수의 값보다 작은 크기의 맵을 전송해야 한다. BOUND 매개변수의 값은 KByte 단위의 크기를 의미한다. 예를 들어, 클라이언트가 크기가 BOUND=30과 같이 30KByte 크기의 맵을 요청했다면, 맵 서버는 30KByte 보다 작은 맵을 전송해야만 한다. 맵 서버가 일정 크기 이하의 맵을 생성하는 방법에는 다음과 같이 두 가지가 있으며, 이 논문에서는 두 번째 방법을 사용함으로써, BOUND 매개변수를 처리한다. 즉, JPEG와 같은 손실 압축 알고리즘을 사용하기 때문에, 손실 압축을 지원하는 이미지 형식에 대해서만 BOUND 매개변수를 적용할 수 있다.

① 첫째, 맵 서버는 클라이언트가 요청한 모든 정보를 포함한 맵을 생성하지 않을 수도 있다. 즉, 맵 서버는 맵 이미지에 대해서 스타일 정보를 변경(예를 들어, 컬러 맵을 흑백 맵으로 변경)하거나, 레이어 리스트 상에서 우선순위가 낮은 레이어의 객체를 제거하거나, 지리 객체에 대해 맵 일반화 연산을 수행할 수 있다.

② 둘째, 맵 서버는 생성된 맵 이미지에 대해서 손실 압축 방법 등을 통해서 맵의 크기를 줄일 수 있다. 손실 압축 알고리즘은 원본 이미지에 대해서 품질이 저하되는 이미지가 생성되는 문제점이 있으나, 이미지의 품질보다 통신비용 및 응답시간이 보다 중요할 있는 모바일 환경에서는 적용될 수 있다.

3.2 모바일 디바이스의 제한된 화면 크기를 고려한 WMS 검색 인터페이스 확장

OGC의 WMS 명세는 모바일 디바이스의 작은 화면 크기를 고려하지 않고 있다. 맵 서버에서 생성된 매우 복잡한 맵은 작은 크기의 모바일 디바이스를 사용하는 사용자에게는 거의 인식되지 않는 문제가 있다. 따라서 이 논문에서는 모바일 디바이스의 제한된 화면 크기를 고려하는 방법으로, 맵 일반화(Map Generalization)를 고려하여 기존 WMS 인터페이스를 확장한다. 맵에 대한 일반화 방법론은 작은 화면의 모바일 디바이스에서 맵에 대한 가독성을 높일 수 있을 것으로 기대된다. 그러나 맵 일반화는 원래의 정보가 일부 사라지기 때문에, 모든 경우에 가독성이 높아지는 것은 아니다. 즉, 손실되는 정보로 인해서 오히려 가독성이 떨어지는 경우가 발생할 수 있으므로, 사용자가 요청한 맵이 매우 복잡해서 현재의 화면으로 판독이 어려울 경우에 맵 일반화를 요청해야 한다.

<표 2> 맵 일반화 연산

구분	연산자	내용
속성 변환	Classification	동일한 속성에 따라서 피처를 분류
기하 변환	Simplification	최소의 피처를 구성하는 모든 좌표 중에서 일부 좌표만으로 해당 지리객체를 표현
	Collapse	피처의 차원을 줄임
	Enhancement	피처의 특징을 잘 표현하기 위해서 기하정보를 변형
	Selection	전체 피처에서 일부만을 선택해서 표현
	Displacement	충돌되는 근접한 피처의 위치를 이동
	Aggregation	유사한 속성을 갖는 피처들을 하나로 합침

현재까지 맵 일반화를 위한 다양한 연구[12][13][14]들이 진행되어 왔으며, <표 2>와 같이 요약될 수 있다. 맵 일반화 방법은 크게 (1)속성 변환(Attribute Transformation)에 의한 일반화 방법과 (2)기하 변환(Geometry Transformation)에 의한 일반화 방법으로 구분될 수 있다. 속성 변환에 의한 일반화 방법은 맵을 구성하는 피처의 속성 정보에 따라 피처들을 분류

해야 하므로, WMS를 통해서 쉽게 적용하기 어렵다. 반면에, 기하 변환에 의한 일반화 방법은 속성 정보가 아닌 기하 정보를 기반으로 일반화 알고리즘을 적용하기 때문에, WMS에서 쉽게 적용할 수 있다. 단, 이 경우에 WMS 맵 서버는 각 피처의 기하 정보를 다룰 수 있는 경우라고 가정한다. 기하 변환에 의한 일반화 방법으로는 단순화(Simplification), 차원 축소(Collapse), 강조(Enhancement), 선택(Selection), 위치 이동(Displacement), 기하 통합(Aggregation) 등의 연산자가 존재한다.

이 논문에서는 단순화(Simplification), 차원축소(Collapse), 선택(Selection) 연산을 적용하여 맵 일반화를 수행할 수 있도록, 인터페이스를 확장하였다.

- 단순화: 1차원 객체(라인)를 구성하는 0차원 객체(포인트)의 수를 줄임 (GSIM 매개변수로 표현)
- 차원축소: 2차원 객체(폴리곤)를 0차원 객체(포인트)로 변경 (GCOL 매개변수로 표현)
- 선택: 검색된 모든 객체에 대해서 일부 객체만을 선택 (GSEL 매개변수로 표현)

GSIM 매개변수는 맵 서버로부터 전송되는 맵에 대해서 단순화 연산자를 적용시킬 레이어의 이름과 단순화에 사용되는 임계값을 명시적으로 기술한다. 여러 개의 레이어에 대해서 동시에 단순화를 요청할 수 있으므로, GSIM 매개변수의 값은 레이어이름과 임계값의 목록으로 표현된다. 예를 들어, 도로 레이어와 동경계 레이어에 대해서 각각 5, 10의 임계값으로 단순화를 요청하는 경우에 "GSIM=도로,5,동경계,10"과 같이 기술될 수 있다.

GCOL 매개변수는 맵 서버로부터 전송되는 맵에 대해서 차원축소 연산자를 적용시킬 레이어의 이름을 명시적으로 기술한다. 여러 개의 레이어에 대해서 동시에 차원축소를 요청할 수 있으므로, GCOL 매개변수의 값은 레이어 이름의 목록으로 표현된다. 예를 들어, 도로 레이어와 동경계 레이어에 대해서 차원축소를 요청하는 경우에 "GCOL=도로,동경계"와 같이 기술될 수 있다.

GSEL 매개변수는 맵 서버로부터 전송되는 맵에 대해서 선택 연산자를 적용시킬 레이어의 이름과 선택 연산에서 사용되는 선택 비율을 명시적으로 기술한다. 여러 개의 레이어에 대해서 동시에 선택 연산을 요청할 수 있으므로, GSEL 매개변수의 값은 레이어 이름과 선택비율 값의 목록으로 표현된다. 예를 들어, 도로 레이어와 건물 레이어에 대해서 각각 50%, 30%의 선택비율로 선택 연산을 요청하는 경우에 "GSEL=도로,50,건물,30"과 같이 기술될 수 있다.

3.3 LBS 응용을 위한 인터페이스 확장

OGC의 WMS 명세는 한 가지 유형의 질의만을 지원할 수 있다. 즉, 맵 클라이언트는 GetMap 요청 시 특정한 사각 영역을 기술(BBOX 매개변수)함으로써 맵을 요청할 수 있다. 이 경우에 맵 서버는 해당 사각 영역 내에 포함된 모든 피처를 포함한 맵을 생성한다. LBS와 같은 모바일 응용에서 특정한 위치로부터 k-번째 근접한 POI 객체를 검색하고, 이 객체를 포함하는 맵을 요청하는 것은 매우 빈번히 발생된다. OGC의 WMS 명세를 따를 경우에, 맵 클라이언트는 해당 POI 객체를 포함하는 사각 영역을 모르기 때문에, 이 객체를 포함하는 맵을 요청할 수 없다. 이 논문에서는 k-번째 근접 POI를 검색하고, 이를 포함하는 맵을 요청하는 질의를 지원하기 위해서 GetMap 요청에 대해서 NLOC 매개변수, NK 매개변수, NPOI 매개변수를 추가였다.

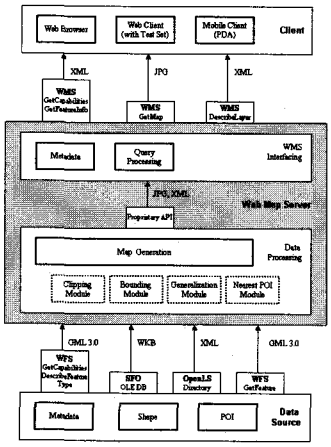
- NLOC(Nearest LOCation): 근접 POI 검색을 위한 기준 위치
- NK(Nearest K-th): 몇 번째 근접 POI를 검색해야 하는지를 지정
- NPOI(Nearest POI): 검색 대상이 되는 POI 객체의 유형

4. 시스템 설계

전체 시스템 구조를 기술하기 전에 먼저 이 논문에서 제안하는 M-WMS 인터페이스의 GetMap 요청을 위해 확장된 매개변수를 요약하면 <표 3>과 같다. 이 논문에서 제안하는 M-WMS 인터페이스를 구현한 맵 서버가 기존의 WMS 맵 서버와 연동되기 위해서는, GetCapabilities 요청에 대한 응답으로 제공되는 서비스 메타 데이터에 대한 확장도 필요하지만, 이 논문에서는 서비스 메타 데이터에 대한 부분은 생략하고 기술한다.

요청 매개변수	내용
DIRECTION = degree	요청되는 맵의 방향에 대한 각도를 기술 (0~360까지 가능)
BOUND = upper_bound	요청되는 맵의 크기의 한계 (단위: KByte)
GSIM=sim_condition_list	하나 또는 그 이상의 쉼표로 분리된 맵 레이어(layer_name)와 임계 값(threshold_value)의 목록
GCOL=col_condition_list	하나 또는 그 이상의 쉼표로 분리된 맵 레이어의 목록
GSEL=sel_condition_list	하나 또는 그 이상의 쉼표로 분리된 맵 레이어(layer_name)와 선택비율 값(selection_ratio)의 목록
NLOC=x,y	근접 POI 검색을 위한 기준 위치
NK=k	몇 번째 근접 POI를 검색해야 하는지를 지정
NPOI=layer_name	검색 대상이 되는 POI 객체의 유형

이 논문에서 제안하는 WMS 확장 인터페이스를 구현하기 위한 맵 서버의 프로토타입 시스템 구조는 <그림 3>과 같다. 웹 맵 서버는 WMS 인터페이스를 처리하기 위한 WMS Interfacing 컴포넌트와 실제 맵을 생성하는 Data Processing 컴포넌트로 구분된다. 맵 생성을 위해 필요한 원시 데이터(피처 데이터)는 OGC의 WFS[8] 또는 SFO[9] 인터페이스를 통해서 획득된다. 웹 맵 서버에서는 클라이언트로부터의 맵 요청에 대해서 WFS, SFO를 통해 피처 데이터를 획득하고, 렌더링을 통해서 사용자가 요구한 맵을 생성하는 역할을 담당한다.

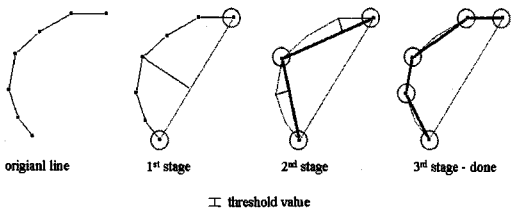


<그림 3> WMS 확장 인터페이스를 지원하는 웹 맵 서버의 시스템 구조

이 논문에서 추가된 확장 인터페이스를 처리하기 위해서 Data Processing 컴포넌트에 4가지 모듈을 설계하였다. 첫 번째, Clipping 모듈은 사용자가 요청한 BBOX 매개변수에 대해서 생성된 맵에 대해서, DIRECTION 매개변수의 내용을 기반으로 실제 사용자에게 필요하지 않은 부분을 제거하는 역할을 담당한다.

둘째, Bounding 모듈은 생성된 맵의 크기가 BOUND 매개변수를 통해 요청된 값보다 작아지도록 맵을 변환한다. 이 논문에서는 손실압축방법인 JPEG 포맷의 압축률을 조정함으로서, 맵에 대한 크기 제한을 지원한다. 따라서 원본 맵에 대해서 품질의 저하가 발생하며, 사용자가 정의한 스타일 정보가 정확히 표현되지 않을 수 있다.

셋째, Generalization 모듈은 GSIM 매개변수, GCOL 매개변수, GSEL 매개변수를 처리한다. GSIM 매개변수는 (레이어명, 임계값) 쌍에 대한 목록으로 표현함으로서, 단순화가 요구되는 레이어를 기술한다. 단, 해당 레이어는 1차원 기하로 표현되어야 하며, 임계값은 픽셀 단위로 표현한다. 픽셀로 표현된 크기는 BBOX, WIDTH, HEIGHT 매개변수에 의해서 실제 크기로 변환될 수 있다. 이 논문에서는 단순화 기능을 제공하기 위해서 Douglas-Peucker 알고리즘을 구현하였다. <그림 4>에서는 Douglas-Peucker 알고리즘에 대한 예를 보이고 있다. 이 예제에서는 최초에 7개의 포인트로 구성된 라인 객체가 3번째 단계에 거쳐 5개의 포인트로 단순화됨을 알 수 있다. GCOL 매개변수는 (레이어)에 대한 목록으로 표현된다. 단, 해당 레이어의 객체는 2차원 기하로 표현되어야 한다. GSIM 매개변수는 (레이어, 선택율) 쌍에 대한 목록으로 표현함으로서, 맵 생성 시 해당 레이어에 포함된 객체 중에서 선택율 만큼의 객체만을 선택한다.



<그림 4> Douglas-Peucker 알고리즘의 예

넷째, Nearest POI 모듈은 특정 위치에 대한 k-번째 근접한 POI를 검색하는 역할을 담당한다. NLOC, NK, NPOI 매개변수에 대해서 실제 k-번째 근접한 POI를

검색하는 것은 OpenLS[16]의 Directory 서비스를 활용한다. Directory 서비스는 POI 검색을 위해 포괄적인 인터페이스를 제공하고 있으므로, 이 논문에서 확장한 매개변수는 근접(Proximity) Directory 서비스로 사상되어 표현될 수 있다.

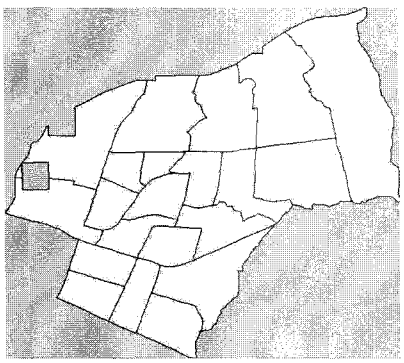
5. 구현 결과

이 논문에서 구현한 프로토타입 시스템의 개발환경은 Visual Studio 6.0에서 C++ 언어를 사용하였으며, XML 문서를 파싱하기 위해서 MSXML 파서를 활용하고 있다. 데이터 소스는 별도의 공간 데이터베이스를 활용하지 않고, 공개 포맷인 Shape 파일을 직접 접근하여 사용하였다. 테스트를 위해 사용한 데이터 소스는 서울시 강동구 데이터로서, 총 4개의 레이어(구경계, 동경계, 도로, 건물)로 구성되어 있다.

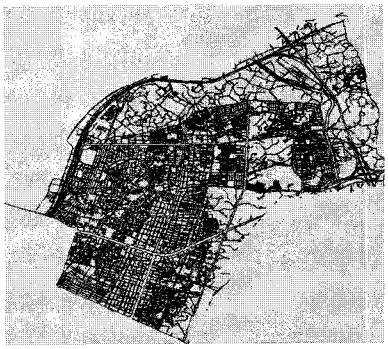
테스트를 위한 GetMap 요청 URL은 다음과 같이 표현될 수 있다.

```
GetMap 요청 URL:
http://localhost/wms/wms.exe?VERSION=1.3.0&REQUEST=GetMap&LAYERS=Layer_Names&STYLES=&CRS=EPSG:4301&BBOX=209622.740232,446083.031232,216092.040232,453227.940232&WIDTH=600&HEIGHT=530&FORMAT=JPEG&BGCOLOR=0xEEEEEE
```

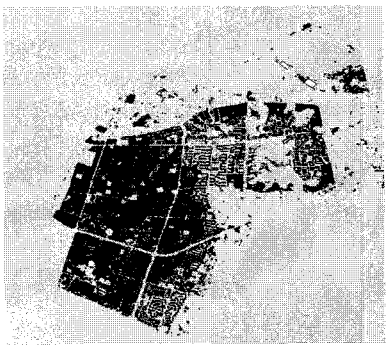
이와 같은 GetMap 요청 URL에서 Layer_Names에는 실제 레이어 이름의 목록이 기술되어야 하며, 서버에서는 레이어 이름에 따라서 <그림 5>와 같은 JPEG 이미지가 생성된다.



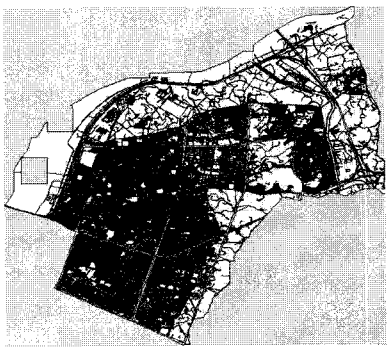
(a) LAYERS=구경계,동경계



(b) LAYERS=도로



(c) LAYERS=건물



(d) LAYERS=구경계,동경계,도로,건물

<그림 5> LAYERS 매개변수에 따른 GetMap 요청 결과

<표 4>에서는 GetMap 연산 수행 결과에 대해서 요약하고 있다. 표에서 Time1은 메모리 DC 상에 기하 객체를 그리는 시간이고, Time2는 JPEG 파일까지 생성하는데 걸리는 시간을 의미한다. 시간은 Pentium IV 프로세서 2.6GHz, 235MByte 메모리상에서 Visual Studio 6.0 Debug 모드에서 측정된 결과이며, 절대적인 수치의 비교보다 상대적인 수치의 비교를 위해서 기술하였다.

<표 4> GetMap 연산 수행 결과 요약

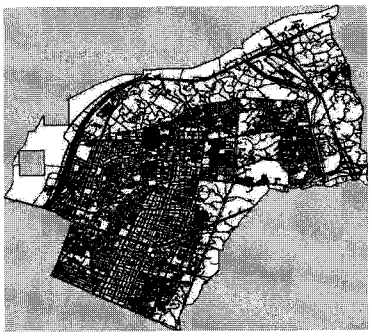
	경계	도로	건물	모두
기하 유형	Polygon	Line	Polygon	
기하객체 개수	24	8161	24,774	32,959
포인트 개수	2,633	138,738	218,890	360,261
Time1 (초)	0.0	0.2	0.5	0.7
Time2 (초)	0.8	1.0	1.3	1.4
결과	그림 5 (a)	그림 5 (b)	그림 5 (c)	그림 5 (d)

5.1. GetMap 확장 구현 결과: 단순화

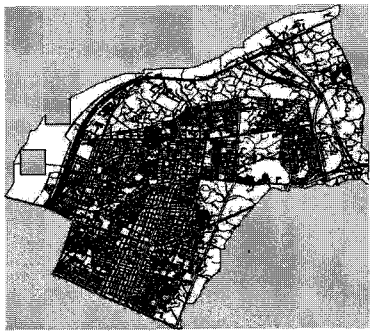
단순화 기능을 추가한 GetMap 요청 URL은 다음과 같이 표현될 수 있다. 이 논문에서 단순화 연산은 1차원 기하 객체에 대해서 수행되는 것으로 한정하였으므로, 구현 결과에서는 "도로" 객체를 단순화하여 보인다.

GetMap 요청 URL:
http://localhost/wms/wms.exe?VERSION=1.3.0&REQUEST=GetMap&LAYERS=구경계,동경계,도로&STYLES=&CRS=EPSG:4301&BBOX=209622.740232,446083.031232,216092.040232,453227.940232&WIDTH=600&HEIGHT=530&FORMAT=JPEG&BGCOLOR=0xEEEEEE&*GSIM=Simplification_Condition*

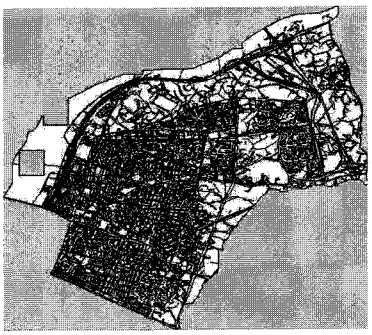
이와 같은 GetMap 요청 URL에서 Simplification_Condition에는 실제 레이어 이름, 임계값의 목록이 기술되어야 하며, 서버에서는 단순화 조건에 따라서 <그림 6>과 같은 JPEG 이미지가 생성된다. 그림에서 (a)는 단순화 조건을 주지 않은 경우이며, (b), (c), (d)는 각각 도로 레이어에 대해서 임계값을 10, 20, 30으로 변경해서 적용한 경우이다. 임계값이 커짐에 따라서 도로가 보다 단순화 되어 표현됨을 알 수 있다.
<표 5>에서는 GetMap 요청시 단순화 연산 수행 결과에 대해서 요약하고 있다. 단순화를 적용하지 않은 경우, 전체 기하객체를 구성하는 포인트 객체의 수는 141,371개였으며, 단순화 연산에 따라 포인트의 개수가 감소함을 알 수 있다. 특히, 임계값을 크게 함에 따라 제거되는 포인트의 수가 증가함을 알 수 있다. 또한, 메모리 DC 상에 기하 객체를 그리는 시간인 Time1에서 알 수 있듯이 단순화 알고리즘의 적용으로 인해 시간 오버헤드가 증가함을 알 수 있다.



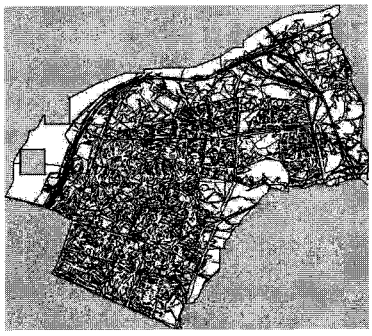
(a) GSIM 매개변수 없는 경우



(b) GSIM=도로,10



(c) GSIM=도로,20



(d) GSIM=도로,30

<그림 6> GSIM 매개변수에 따른 GetMap 요청 결과

<표 5> 단순화 연산 수행 결과 요약

	단순화 적용 안함	도로, 10	도로, 20	도로, 30
기하객체 개수	8,185	8,185	8,185	8,185
포인트 개수	141,371	32,253	24,281	21,518
Time1 (초)	0.2	0.5	0.4	0.3
Time2 (초)	1.0	1.3	1.2	1.1
결과	그림 6 (a)	그림 6 (b)	그림 6 (c)	그림 6 (d)

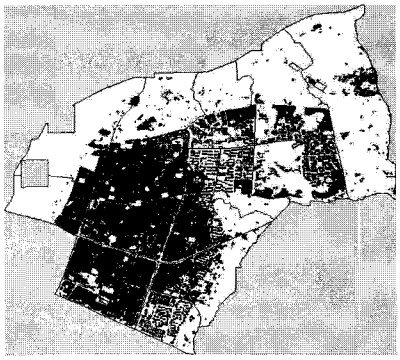
5.2. GetMap 확장 구현 결과: 차원축소

차원축소 기능을 추가한 GetMap 요청 URL은 다음과 같이 표현될 수 있다. 이 논문에서 차원축소 연산은 2차원 기하 객체에 대해서 수행되는 것으로 한정하였으므로, 구현 결과에서는 "건물" 객체를 단순화하여 보인다.

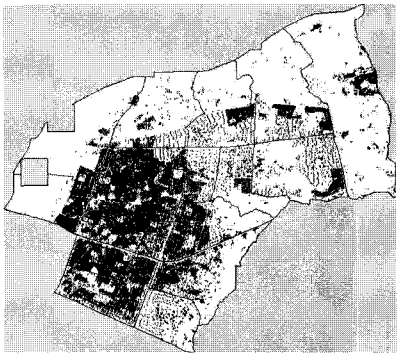
GetMap 요청 URL:
http://localhost/wms/wms.exe?VERSION=1.3.0&REQUEST=GetMap&LAYERS=구경계,동경계,도로&STYLES=&CRS=EPSG:4301&BBOX=209622.740232,446083.031232,216092.040232,453227.940232&WIDTH=600&HEIGHT=530&FORMAT=JPEG&BGCOLOR=0xEEEEEE&*GCOL=Collapse_Condition*

이와 같은 GetMap 요청 URL에서 Collapse_Condition에는 실제 레이어 이름의 목록이 기술되어야 하며, 서버에서는 차원축소 조건에 따라서 <그림 7>과 같은 JPEG 이미지가 생성된다. 그림에서 (a)는 차원축소 조건을 주지 않은 경우이며, (b)는 건물 레이어에 대해서 차원축소를 적용한 경우이다.

<표 6>에서는 GetMap 요청시 차원축소 연산 수행 결과에 대해서 요약하고 있다. 차원축소를 적용하지 않은 경우, 전체 기하객체를 구성하는 포인트 객체의 수는 221,523개였으며, 차원축소 연산에 따라 2차원 기하객체가 0차원 기하객체로 변환되어 포인트의 개수가 급격히 감소함을 알 수 있다. 또한, 메모리 DC 상에 기하 객체를 그리는 시간인 Time1에서 알 수 있듯이 차원축소로 인해 그리는 시간이 줄어듦을 알 수 있다.



(a) GCOL 매개변수 없는 경우



(b) GCOL=건물

<그림 7> GCOL 매개변수에 따른 GetMap 요청 결과

<표 6> 차원축소 연산 수행 결과 요약

	단순화 적용 안함	도로, 10
기하객체 개수	24,798	24,798
포인트 개수	221,523	27,407
Time1 (초)	0.5	0.4
Time2 (초)	1.3	1.2
결과	그림 7 (a)	그림 7 (b)

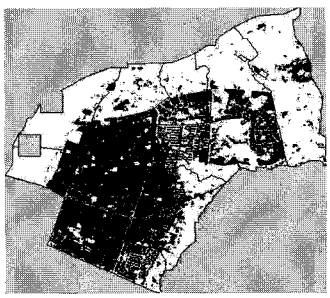
5.3. GetMap 확장 구현 결과: 선택

선택 기능을 추가한 GetMap 요청 URL은 다음과 같이 표현될 수 있다. 이 논문에서 선택 연산은 모든 차원의 기하 객체에 대해서 수행될 수 있으나, 구현 결과에서는 건물 객체를 선택하여 보인다.

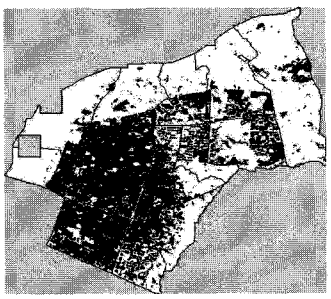
GetMap 요청 URL:
http://localhost/wms/wms.exe?VERSION=1.3.0&REQUEST=GetMap&LAYERS=구경계,동경계,도로&STYLES=&CRS=EPSG:4301&BBOX=209622.740232,446083.031232,216092.040232,453227.940232&WIDTH=600&HEIGHT=530&FORMAT=JPEG&BGCOLOR=0xEEEEEE&*GSEL=Selection_Condition*

이와 같은 GetMap 요청 URL에서 Selection_Condition에는 실제 레이어 이름, 선택비율 값의 목록이 기술되어야 하며, 서버에서는 선택 조건에 따라서 <그림 8>과 같은 JPEG 이미지가 생성된다. 그림에서 (a)는 선택 조건을 주지 않은 경우이며, (b), (c), (d)는 각각 건물 레이어에 대해서 선택비율을 75%, 50%, 25%로 변경해서 적용한 경우이다. 선택비율이 작아짐에 따라서 건물 레이어의 전체 객체 중에서 출력되는 객체 수가 줄어듦을 알 수 있다.

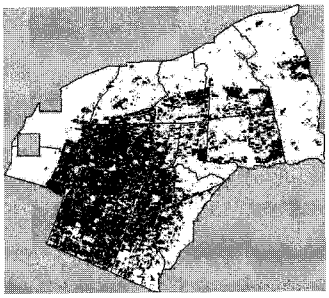
<표 7>에서는 GetMap 요청시 선택 연산 수행 결과에 대해서 요약하고 있다. 선택을 적용하지 않은 경우, 전체 기하객체 수는 24,798개였으며, 선택 연산에 따라 출력되는 객체 수가 선택비율에 따라 감소함을 알 수 있다. 또한, 출력되는 객체 수가 줄어듦에 따라서 메모리 DC 상에 기하 객체를 그리는 시간인 Time1이 감소함을 알 수 있다.



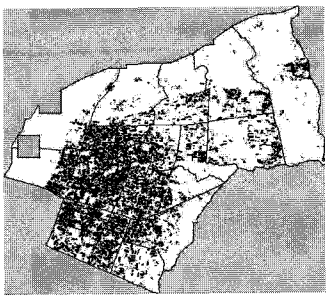
(a) GSEL 매개변수 없는 경우



(b) GSEL=건물,75



(c) GSEL=건물,50



(d) GSEL=건물,25

<그림 8> GSEL 매개변수에 따른 GetMap 요청 결과

<표 7> 선택 연산 수행 결과 요약

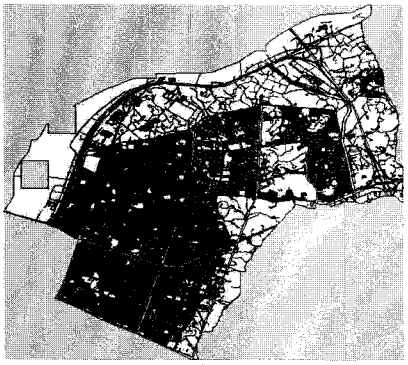
	선택 적용 안함	건물, 75%	건물, 50%	건물, 25%
기하객체 개수	24,798	18,953	12,700	6,513
포인트 개수	221,523	170,251	114,270	59,473
Time1 (초)	0.5	0.4	0.3	0.3
Time2 (초)	1.3	1.2	1.1	1.0
결과	그림 8 (a)	그림 8 (b)	그림 8 (c)	그림 8 (d)

5.4. GetMap 확장 구현 결과: 일반화 모두 적용

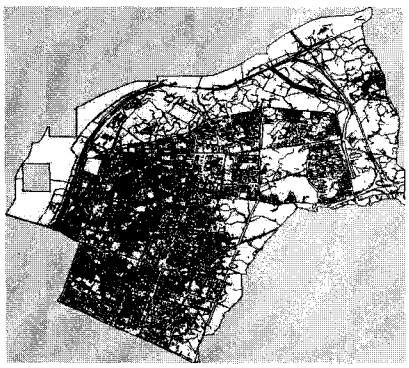
이 절에서는 일반화를 모두 적용한 경우에 구현 결과를 기술한다. 서버에서는 클라이언트로부터의 일반화 요청에 따라서 <그림 9>와 같은 JPEG 이미지가 생성된다. <그림 9>의 (a)는 <그림 5>의 (d)와 동일하며, 구경계, 동경계, 도로, 건물 레이어에 대해서 GetMap을 요청한 경우이다. <그림 9>의 (b), (c), (d)는 각각 다음과 같은 일반화 조건을 적용하였다.

- 일반화 조건1: GSIM=도로,10&GCOL=건물&GSEL=건물,50,도로,75
- 일반화 조건2: GSIM=도로,50&GCOL=건물&GSEL=건물,50,도로,75
- 일반화 조건3: GCOL=건물&GSEL=건물,50,도로,75

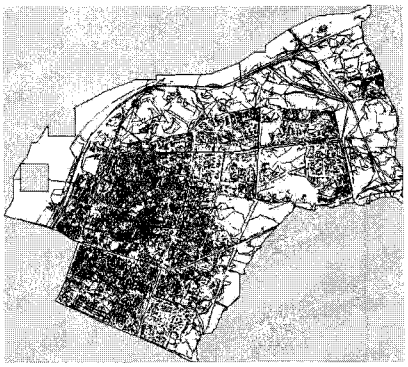
<표 8>에서는 GetMap 요청시 일반화 연산 수행 결과에 대해서 요약하고 있다. 일반화 조건1, 일반화 조건2의 경우에서는 도로 객체에 단순화를 적용하였기 때문에, 전체 기하객체를 구성하는 포인트 수가 각각 38,011개, 29,901개로 매우 줄었으나, 단순화 연산의 시간 오버헤드로 인해 시간이 많이 걸림을 알 수 있다. 반면에, 일반화 조건3에서는 단순화 연산을 적용하지 않고, 차원축소 및 선택 연산만을 적용함으로서 빠른 시간 내에 일반화 조건 1과 시각적으로 유사한 맵을 생성하였다. 따라서 처리 시간이 많이 소요되는 단순화 연산을 적용하는 것보다는 차원축소 및 선택 연산을 적절히 활용하는 것이 응답 시간을 줄일 수 있을 것으로 판단된다.



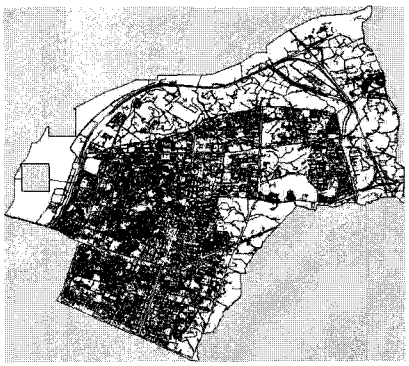
(a) 일반화 적용하지 않은 경우



(b) 일반화 조건1



(c) 일반화 조건2



(d) 일반화 조건3

<그림 9> 다양한 일반화 조건에 따른 GetMap 요청 결과

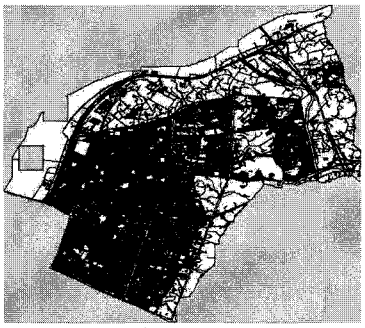
<표 8> 일반화 연산 수행 결과 요약

	일반화 적용 안함	일반화 조건1	일반화 조건2	일반화 조건3
기하객체 개수	32,959	18,871	19,118	18,947
포인트 개수	360,261	38,011	29,901	121,620
Time1 (초)	0.7	0.8	0.6	0.5
Time2 (초)	1.4	1.5	1.4	1.3
결과	그림 9 (a)	그림 9 (b)	그림 9 (c)	그림 9 (d)

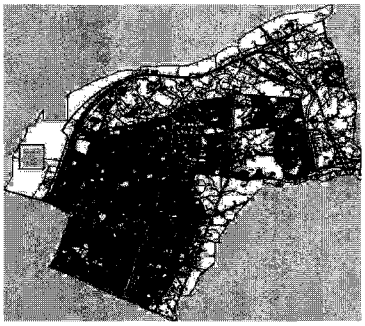
5.5. GetMap 확장 구현 결과: BOUND 적용

서버에서는 클라이언트로부터의 BOUND 요청에 따라서 <그림 10>과 같은 JPEG 이미지가 생성된다. <그림 10>의 (a)는 <그림 5>의 (d)와 동일하며, 구경

계, 동경계, 도로, 건물 레이어에 대해서 GetMap을 요청한 경우이다. <그림 10>의 (b)는 생성되는 JPEG 파일의 크기를 30으로 제한하였다. 이 경우에 (a)에서 63.1KB의 크기로 생성된 JPEG 파일은 (b)에서 16.1KB의 크기로 줄어듦을 알 수 있다. 반면에, 손실 압축 방법에 의해서 이미지의 품질은 저하되었다.



(a) BOUND 매개변수 없는 경우
(파일크기: 63.1KB)

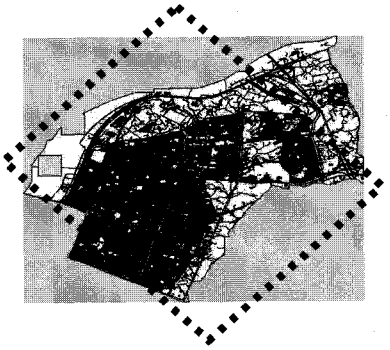


(b) BOUND=30
(파일크기: 16.1KB)

<그림 10> BOUND 매개변수에 따른 GetMap 요청 결과

5.6. GetMap 확장 구현 결과: DIRECTION 적용

서버에서는 클라이언트로부터의 DIRECTION 요청에 따라서 <그림 11>와 같은 JPEG 이미지가 생성된다. <그림 11>의 (a)는 <그림 5>의 (d)와 동일하며, 구경계, 동경계, 도로, 건물 레이어에 대해서 GetMap을 요청한 경우이다. <그림 11>의 (b), (c), (d)는 각각 DIRECTION 매개변수의 값을 통해 진행방향이 정북 대비 45도, 90도, 180도 변경해서 적용한 경우이다. 그림에서 (b)는 사용자가 45도 방향으로 이동하는 경우에 (a)의 맵에서 점선 사각형의 영역에 대한 맵을 의미한다.



(a) DIRECTION 적용하지 않은 경우

(b) DIRECTION=45

(c) DIRECTION=90

(d) DIRECTION=180

<그림 11> DIRECTION 매개변수에 따른 GetMap 요청 결과

6. 결론 및 향후 연구

OGC에 의해 제안된 WMS 명세는 다양한 응용 분야에서 공간 데이터의 공유와 활용을 증진시키고 있다. 따라서 자신의 공간 데이터를 소유하지 않은 클라이언트 측 응용 개발자들도 공간 데이터를 다루는 응용을 구현할 수 있게 되었다. 즉, 클라이언트는 GIS의 복잡한 기능을 이해할 필요 없이 맵 서버의 단순한 인터페이스를 통해 다양한 맵 서버에게 맵을 요청할 수 있다.

그러나 OGC 명세에 따라 개발된 맵 서버는 모바일 응용 분야에는 충분한 기능을 제공하지 못하는 단점이 있다. 이 논문에서는 OGC의 표준 인터페이스를 준수하며, 모바일 환경에서 요구되는 몇 가지 유용한 확장 기능을 제공하기 위한 웹 맵 서버 인터페이스를 제안하였다. 확장 인터페이스를 지원하는 맵 서버를 구현하여 실험한 결과 기존의 OGC의 맵 서버에 비해서 보다 가독성이 높은 맵을 빠른 응답 시간 내에 생성할 수 있음을 보였다. 따라서 제안된 확장 인터페이스를 구현한 맵 서버를 통해 모바일 환경의 응용에서도 WMS가 널리 활용될 수 있을 것으로 기대한다.

현재 확장된 인터페이스는 모바일 디바이스의 화면 크기에 따른 맵 가독성 제고 및 빠른 응답시간을 제공할 수 있는 맵 서버의 가능성을 제시하였다. 즉, 맵 서버가 가지고 있는 원본 맵에 대해서 일부 불필요한 정보를 제거함으로써 맵의 가독성을 높일 수 있음을 보였다. 그러나 손실되는 정보의 양이 많아질 경우 오히려 맵의 가독성이 저해될 가능성도 배제할 수 없다. 향후 모바일 디바이스의 화면 크기, 맵의 축척, 대역폭을 고려한 서비스 품질 등 상황에 따라 최적화된 맵을 생성하기 위한 연구가 필요하다. 또한 맵 클라이언트의 능력 정보를 사용함으로써, 맵 서버에서 해당 클라이언트에 최적화된 맵을 생성하는 방법에 대한 연구도 필요하다.

참고문헌

[1] Steinke, A. P., "Developing Geographic Services on the World Wide Web," Asia-Pacific World Wide Web '95 Conference, 1995.

[2] Plewe, B., "A Primer on Creating Geographic Services," GIS World Magazine, 1996, pp. 56-58

[3] Crossley, D., Boston, T., "A Generic Map Interface to Query Geographic Information Using the World Wide Web," Fourth International World Wide Web conference, 1995.

[4] Foote, K. E., Kirvan, A. P., "WebGIS," NCGIA Core Curriculum in GIScience, http://www.ncgia.ucsb.edu/giscc/units/u133/u133.html, posted July 13, 1998.

[5] Fonseca, F. T., Davis Jr., C. A., "Using the Internet to Access Geographic Information: An Open GIS Interface Prototype," Conference and Workshop on Interoperating Geographic Information Systems, 1997.

[6] Alexander, J. P., Warwick, V. J., "Writing GIS Applications for the WWW," Proceedings of the 1997 ESRI User Conference, 1997.

[7] Open Geospatial Consortium Inc., "The OpenGIS Abstract Specification Model Version 3," 1998.

[8] Open Geospatial Consortium Inc., "OpenGIS Discussion Paper#01-023: Web Feature Service Draft Candidate Implementation Specification 0.0.12," 2001.

[9] Open Geospatial Consortium Inc., "The OpenGIS Simple Feature Specification for OLE/COM Revision 1.1," 1999.

[10] Open Geospatial Consortium Inc., "Styled Layer Description Implementation Specification 1.0.0," 2002.

[11] Open Geospatial Consortium Inc., "Web Map Service 1.3," 2004.

[12] Davis, C. A., Laender, A. H. F., "Multiple Representations in GIS: Materialization Through Map Generalization. Geometric, and Spatial Analysis Operations," ACM-GIS 1999, 1999, pp. 60-65.

[13] Bjørke, J. T., Myklebust, I., "Map generalization

: Information theoretic approach to feature elimination," ScanGIS 2001, 2001, pp. 203-211.

[14] Ware, J. M., Jones, C. B., Thomas, N., "Automated map generalization with multiple operators: a simulated annealing approach," International Journal of Geographical Information Science 17(8), 2003, pp. 743-769.

[15] Lehto, L., "GiMODig system architecture," http://gimodig.fgi.fi/deliverables.php.

[16] Open Geospatial Consortium Inc., "OpenGIS Location Services (OpenLS): Core Services 1.0," 2004.

조대수

1995년 부산대학교 컴퓨터공학과 졸업 (공학사)

1997년 부산대학교 컴퓨터공학과 졸업 (공학석사)

2001년 부산대학교 컴퓨터공학과 졸업(공학박사)

2001년~2004년 한국전자통신연구원 텔레매틱스연구단 선임연구원

2004년 ~ 현재 동서대학교 인터넷공학부 전임강사

관심분야 : GIS, 공간데이터베이스, LBS, 이동체 데이터베이스 등

오병우

1993년 건국대학교 졸업(학사)

1995년 건국대학교 전자계산학과 졸업 (석사)

1999년 건국대학교 전자계산학과 졸업 (박사)

1999년~2004년 한국전자통신연구원 텔레매틱스연구단 선임연구원

2004년 ~ 현재 금오공과대학교 컴퓨터공학부 조교수

관심분야 : 데이터베이스, GIS, Mobile GIS, 텔레매틱스 등