이동객체 데이타베이스를 위한 시공간 데이타 생성기의 설계 및 구현

정종환[O] 전세길 나연묵
단국대학교 전자.컴퓨터공학과
{jhjeong[O], sgjeon}@dankook.ac.kr, ymnah@dku.edu

Design & Implementation of a Spatio-Temporal Data Generator for Moving object Databases

Jonghwan Jeong[O], Segil Jeon, Yunmook Nah
Dept. of Electronics & Computer Engineering, Dankook University

요 약

위치 기반 서비스 시스템은 이동객체에 대한 효율적인 검색을 가능하게 해주는 시스템으로 이동객체의 현재와 과거 위치에 대한 정보를 유지한다. 이러한 시스템을 평가하기 위해서는 이동객체들의 사실적인 움직임을 기술한 데이타 집합이 필요하다. 이러한 데이타 집합을 생성하는 대표적인 생성기로는 GSTD가 있다. 하지만 GSTD도 몇 가지 문제점을 지니고 있다. GSTD에 의해서 생성되는 작업공간과 데이타 집합은 실생활 객체의 움직임을 묘사하는데 부적절한 면이 있다. 또한 GSTD가 제공하는 매개변수 안에 사용자가 객체의 이동성과 시간성을 명확히 기술 해줄 수 있는 부분이 제시 되어 있지 않다.

본 논문에서는 GSTD의 여러 매개변수들과 함수들을 변환하거나 추가하여 좀 더 현실적인 객체 움직임을 나타내는 데이타 집합을 만들고, 사용자가 객체의 특성을 세밀히 지정해 줄 수 있는 확장된 GSTD를 구현한다. 또한 확장된 GSTD 기반 LBS 시뮬레이터의 설계와 구현에 관한 원리를 포함한 주요 특징을 기술한다. 확장된 GSTD는 다양한 LBS 응용 시스템의 성능 측정에 활용될 수 있을 것이다.

1. 서 론

현재 엄청난 수의 가입자를 보유한 이동 통신 시장은 모바일폰을 이용하여 다양한 위치 기반 서비스를 제공하고 있다. 예를 들어, 현재 나의 위치로부터 등록된 가장 가까운 친구를 찾는 경우 또는 어떤 상점에서 특정 시간에 근 거리 내에 있는 잠재적인 고객들에게 할일 쿠폰을 발송하는 서비스를 들 수 있다. 또한 모바일폰 사용자가 현재 위치한 지역의 일기예보를 알고자 할 때 이러한 서비스를 제공하는 서비스 제공자는 고객의 현재 위치를 파악하고 있어야 한다.

모바일폰 가입자의 시간에 따라 변화하는 위치를 저장대상으로 보고 저장, 인덱싱하여 고객에게 관련된 서비스를 제공하는 시스템을 위치 기반 서비스(Location Based Service: LBS)시스템이라 한다. 이러한 위치 기반 서비스 시스템을 구현하고 평가하기 위해서는 실제 시간, 공간상에서 연속적으로 위치가 변화하는 데이타 집합이 필요하다. 이러한 데이타 집합은 실제 이동객체의 위치를 추적, 저장함으로써 생성하거나 시뮬레이터를 이용하여 생성할 수 있다[1,6].

대표적인 시뮬레이터로서 GSTD(Generate Spatio Temp oral Data)를 들 수 있다. 기존의 GSTD는 통계학적 분포를 이용하여 시간의 변화에 따라 이동객체의 위치를 변화 시킨다. 하지만 실생활의 이동객체들은 이러한 통계학적 분포를 따라 움직이지 않으며, 특정한 이동 패턴을 가지는 개별 객체들의 움직임을 생성하는데 있어서 기존의 GSTD는 한계를 가지고 있다. 이러한 한계점을 극복하고 좀 더 사실적인 데이타 집합을 생성하기 위해서 본 논문에서는 기존의 GSTD 알고리즘을 확장하여, 모바일폰 가입자를 서비스 대상으로 하는 위치 기반 서비스의 실험 데이타 집합을 생성하기 위한 시뮬레이터를 설계 및 구현한다[2,3,4,5].

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구에 대해서 기술하고, 3절에서는 확장된 GSTD에 대해서 설명한다. 4절에서는 시뮬레이터 구현환경과 인터페이스에 대해서 설명하고 5절에서는 결론 및 향후 연구에 대해서 기술한다.

2. 관련 연구

2.1 위치 기반 서비스

그림 1은 기본적인 LBS 시스템의 전체 구조를 나타내고 있다. 위치 시스템(Positioning System)은 이동객체 의 현재 위치를 측정해서 구하는데 사용한다.

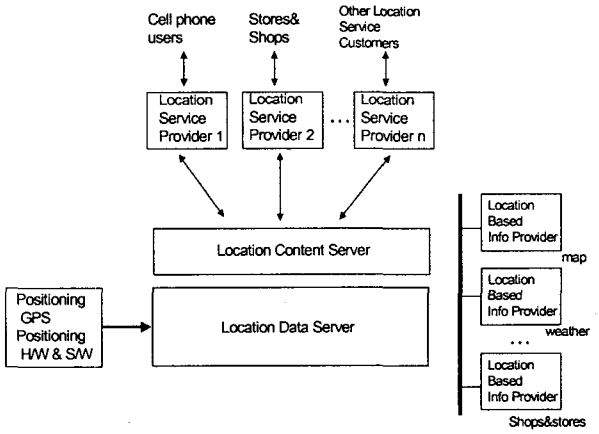

그림 1. 기본적인 LBS 시스템 구조

위치 데이타 서버(Location Data Server: LDS)는 메인 메모리와 디스크 기반 데이타베이스를 이용하여 이동객체의 현재, 과거 위치를 저장하고 인덱싱한다. LCS(Location Content Server)는 LDS에서 제공하는 사용자의 위치 정보와 LBIPs에서 제공하는 정보를 조합하고 실시간으로 LSP에게 정보를 제공한다. 위치 기반 정보 제공자(Location-Based Information Providers: LBIPs)는 지도, 날씨등의 개별적인 정보를 제공한다. LSP(Location Service Provider)는 사용자의 요청을 받아 LCP에 접속하여 정보를 제공하는 독립적인 공급자이다[6].

LBS 시스템을 사용하는 예를 들어보면 어떤 모바일폰 사용자가 현재 자기로부터 가장 가까운 친구를 찾는 경우 LSP에게 정보를 요청하게 되고, LSP는 LBIB와 LDS의 정보를 조합하여 저장한 LCP 이용하여 사용자에게 해당 정보를 제공한다.

2.2 GSTD

시공간 접근 방법(Spatio-Temporal access methods: STAMs)를 효과적으로 평가하기 위해서는 데이타 집합을 생성-저장하고, 평가할 STAMs들을 수집하고, 이 STAMs을 가지고 생성된 데이타 집합에 대해 실행한 결과를 보여주는 종합적인 모듈이 필요하다[3].

이 모듈에서 데이타 생성부분에만 관심을 집중하면, 실생활 객체들의 움직임을 보여주는 실제 데이타 집합의 수집, 또는 이에 상응하는 데이타 집합을 사용자의 요구사항에 맞춰서 생성할 수 있는 알고리즘이 필요하다. 과거에 몇 몇 알고리즘들이 구현되었는데 이 알고리즘들은 정적인 공간 데이타를 생성했고, 작업공간 안에서 미리 정해진 분포만 나타낼 수 있었다. 그러나 STAMs를 평가하기 위해서는 객체들의 시간적인 전개와 그에 따른 객체의 위치변화가 이루어져야 하므로 정적인 공간 데이타 집합은 평가에 적절하지 않다[2,4].

본 논문에서 소개하는 GSTD 알고리즘은 시공간 데이타 집합 생성에서 고려되어야 할 여러 가지 매개변수와 알고리즘을 제시한다. 시간에 따라 전개되는 이동객체들을 제어하는 매개변수 집합의 정의는 GSTD 알고리즘의 가장 기본적인 문제이다. 매개변수 집합은 그 기능에 따라 다음의 세 가지 분류로 나눠 볼 수 있다.

- 객체 인스턴스의 시간 간격(Duration)
- 객체의 이동(Shift)
- 객체의 크기 변화(Resizing)

객체의 이동은 시간 간격(duration)을 가지며, 각 시간마다 공간 위치의 변화(shift)를 수반한다. GSTD 알고리즘은 점과 사각형, 두 가지 형식의 객체를 생성할 수 있는데 사각형 객체는 시간의 흐름에 따라 그 위치뿐만 아니라 크기도 사용자가 지정한 수치 내에서 변화한다. 그림 2는 GSTD 알고리즘에 의해 생성된 점 객체의 움직임을 도식화 한 것이다[5].

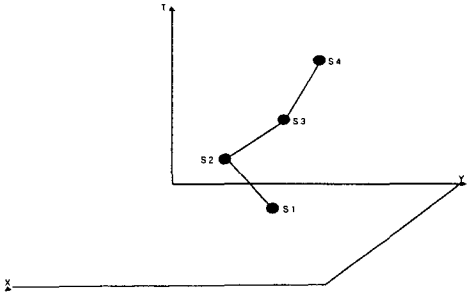

그림 2. 시간 전개에 따른 객체의 이동.

3. GSTD의 확장

3.1 GSTD의 문제점

2절에서 살펴본 GSTD는 STAMs를 평가하기 위한 데이타 집합을 생성하는데 있어서 가장 기본적이고 효율적인 알고리즘이다. 하지만 GSTD도 몇 가지 문제점을 가지고 있다.

- 실생활의 여러 이동객체들의 움직임은 GSTD에서 제시하는 통계학적 분포(Uniform, Gaussian, Skewed)만을 따르지는 않는다.
- 객체들의 이동에 있어서 지형 지물등(blocking object)과 같은 객체의 이동성을 방해하는 여러 장애 요소들이 고려되지 않고 있다.
- 작업공간과 시간축의 한계값(1.0)이 작아 GSTD 데이타 집합으로 객체들의 실제 움직임을 시뮬레이션 하기엔 부족한 면이 있다.
- 객체의 이동속도를 정교하게 제어하는 GSTD 매개변수의 부족으로 객체의 이동 속도를 사용자가 지정할 수 없다. 또한 특정 시간마다 객체들의 위치를 갱신하거나 평균속도를 가지는 시스템을 벤치마크 하기 위해선 부적절하다.

이러한 단점을 보완하기 위해 우리는 GSTD의 시간 전개 알고리즘을 수정하고 객체의 움직임에 있어서도 블럭킹객체(blocking object)의 영향을 받는 확장된 알고리즘을 3.2절에서 제안한다.

3.2 GSTD의 개선

위에서 제시한 문제점을 해결하기 위해서 우리는 실생활 움직임과 유사한 객체 움직임을 만들기 위해 작업공간 안에 고정된 블럭킹객체를 생성하여 객체의 움직임에 영향을 줄 수 있도록 알고리즘을 수정하였다. 블럭킹객체 생성에서 사용자는 생성될 객체의 분포와 객체의 수를 지정할 수 있을 뿐만 아니라 객체의 시간 전개와 움직임에 대해서도 사용자가 더 세밀히 조정할 수 있도록 매개변수와 함수를 추가하였다.

이동객체는 그림 3에서 보이는 것과 같이 생성되는 위치 또는 이동한 위치에 따라 활성/비활성 상태로 나누어지게 된다. 활성상태는 객체가 활발한 움직임을 보이는 상태로 도보 또는 이동수단을 타고 이동하고 있는 모든 객체들의 움직임을 묘사할 수 있다. 비활성 상태는 객체가 건물 안에서 활동성에 제약을 받고 있는 상태로서 활동력이 활성상태보다 떨어진다. 객체 인스턴스는 상태에 따라 사용자가 지정한 시간 범위 내에서 전개될 것이며, 객체의 이동성도 사용자가 지정한 범위에 따라 다른 움직임 패턴을 보일 것이다. 객체의 상태는 객체 인스턴스의 좌표를 블럭킹객체의 위치와 계산하여 결정한다.

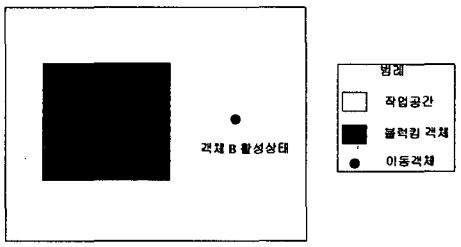

그림 3. 객체 상태

작업공간과 시간축의 크기는 사용자가 지정할 수 있도록 조정되었고 이동객체 인스턴스는 시간 한계값에 이를 때까지 계속 전개된다. 불특정한 시간간격을 가지는 GSTD의 문제점을

해결하기 위해 사용자가 직접 시간 전개 범위를 지정할 수 있도록 매개변수를 추가하였다. 그림 4는 확장된 GSTD 매개변수들과 그 입력 값의 예를 들어 보았다.

<n>은 생성될 이동객체의 수를 지정하고 <starting_id> 생성될 객체의 첫 번째 객체 번호를 지정한다. 이를 이용하여 여러 특성들의 객체를 생성하여 하나의 데이타베이스 테이블에 저장함으로서 종합적인 데이타 집합을 얻을 수 있다.

<init_adjust>는 이동객체의 초기 분포 생성 시 작업공간 안에 미리 생성되어 있는 블럭킹객체를 고려할 것인지, 고려한다면 어떤 식으로 이동객체에 영향을 줄 것인지 결정하는 변수이다. <distr_initial>는 이동객체의 초기분포를 결정하는 변수이고 <distr_c>는 이동객체 초기화 후 시간 전개 시 객체 움직임을 결정해 준다. <time_limit>는 이동객체의 시간 전개 한계값을 설정해주고, <in_t_min>~<out_t_max>는 객체 상태에 따라 시간 전개 범위를 결정해 준다. <in_min_c[0]>~<in_max_c[1]>는 객체가 비 활성상태일 때, <out_min_c[0]>~<out_max_c[1]>은 객체가 활성상태 일 때 작업공간에서 객체의 x, y축 이동 범위를 제한하는 값이다.

```
gstd <filename> <n> <starting_id> <seed> <init_adjust>
     <distr_initial> <distr_init_mean> <distr_init_sigma>
     <distr_c> <distr_c_mean> <distr_c_sigma>
     <time_limit> <in_t_min> <in_t_max> <out_t_min> <out_t_max>
     <in_min_c[0]> <in_min_c[1]> <in_max_c[0]> <in_max_c[1]>
     <out_min_c[0]> < out_min_c[1]> <out_max_c[0]> <out_max_c[1]>

gstd result.txt 2 1 100 0
    0 0.1 0.02
    0.0.1 0.02
    2.0 0.0001 0.0010 0.005 0.007
  -0.01 -0.01 0.0001 0.0001
  -0.02 -0.02 0.0002 0.0002
```

그림 4. 확장된 GSTD의 매개변수들.

4. 시뮬레이터 설계 및 구현

기존 GSTD 프로그램을 확장하면서 마이크로소프트사의 Visual C++ 프로그램을 이용하여 변수와 함수들을 클래스화 하였다. 그림 5는 새로운 GSTD 프로그램의 객체 클래스로서, 클래스의 멤버변수와 멤버함수를 보여주고 있다.

```
class object
{
        public :
        int id;
        int state;
        int state_counter;
        float T_T;
        float c[2];

        public :
        object();
        ~object();

        int get_block_number(float x, float y, struct BLOCK *pBlock);
        int get_state(float x, float y, struct BLOCK *pBlock);

        void block_out_adjust(class object *p, struct BLOCK *pBlock);
        void my_output(FILE *fp1, class object *p);
        void copy(class object *old, class object *new_p);

        float get_random(int distr, float min, float max, float mean, float sigma);
        float uniform(float x1, float x2);
        float zipf(float x1, float x2,float p);
        float gaussian (float mean, float sigma, float min, float max);
};
```

그림 5. 객체 클래스 구성

각 객체는 정수형의 객체 번호(id)를 가진다. state 변수는 객체의 현재 상태를 저장하는 변수이고, state_counter 변수는 객체의 상태 변화 횟수를 저장하는 변수로서, 객체 상태를 활성/비활성 두 가지 상태 이외에 추가적인 객체 상태를 나타 낼 수 있으며 그에 따라 여러 가지 제어를 가할 수 있다. T_T변수는 객체 인스턴스가 전개되는 시간이며, C[2]변수는 작업공간 안의(X, Y) 좌표를 나타낸다. 객체 클래스의 모든 멤버 변수들은 객체 인스턴스가 갱신 될 때마다 SQL server 2000 데이타베이스에 그림 6과 같이 저장된다.

	oid	Transaction_Time	x	y	state	state_counter
1	1	.00000000	.10417730	.43600360	0	0
2	2	.00000000	.37334440	.06045230	0	0
3	3	.00000000	.47698040	.44348570	0	0
4	4	.00000000	.02059985	.43323360	0	0
5	5	.00000000	.40085330	.10051560	0	0
6	6	.00000000	.36140470	.21500690	0	0
7	7	.00000000	.28906850	.43403220	0	0
8	8	.00000000	.58399640	.42636630	0	0
9	9	.00000000	.39235750	.27059380	0	0
10	10	.00000000	.35621500	.11242360	0	0
11	1	.02000000	.20417730	.53600360	1	1
12	1	.02300000	.30417730	.63600360	1	1
13	1	.02600000	.40417730	.73600360	1	1
14	1	.02900000	.50417730	.83600370	1	1
15	1	.03200000	.60417730	.93600370	1	1
16	1	.03500000	.70417730	1.03600400	1	1
17	1	.03800000	.80417730	1.13600400	1	1
18	1	.04100000	.90417740	1.23600400	1	1
19	1	.04400000	1.00417700	1.33600400	1	1
20	1	.04699999	1.10417700	1.43600400	1	1
21	1	.04999999	1.20417700	1.53600400	1	1
22	1	.05299999	1.30417700	1.63600400	1	1

그림 6. 데이타베이스에 저장된 이동객체 레코드들.

5. 결론

본 논문에서는 시공간 데이터 집합 생성기로서 가장 대표적인 GSTD의 문제점을 제시하고 수정하여, 좀 더 현실적인 데이터 집합을 생성하였고, 실생활 객체의 여러 특성들을 사용자가 직접 지정할 수 있도록 하였다. 또한 이 수정된 GSTD를 기반으로 한 LBS 시뮬레이터를 설계 및 구현하였다. 이 시뮬레이터는 다양한 LBS 응용 시스템의 성능 측정에 활용될 수 있을 것이다. 향후 연구 과제로서 실생활 이동객체의 데이타 집합과 시뮬레이터를 이용하여 만든 데이타 집합과의 유사도 비교 연구가 필요하다.

참고문헌

[1] Forlizzi, L., Guting, R. H., Nardelli, E. and Schneider, M., "A *Data Model and Data Structures for Moving Object* Databases," in Proc. ACM SIGMOD, 2000

[2] Y. Theodoridis and M.A. Nascimento. "Generating spatiotemporal datasets on the WWW," ACM SIGMOD Record. Vol. 29(3):39-43, 2000.

[3] M. A. Nascimento, J. R. O. Silva, Y. Theodoridis, "Access Structures for Moving Points", TIMECENTER Technical Report TR-33, August 1998.

[4] Yannis Theodoridis, Jeffersion R.O. Silva and Mario A. Nascimento, "On the Generation of Spatiotemproal Datasets", TIMECENTER Technical Report TR-40, January 1999.

[5] Dieter Pfoser, Yannis Theodori0ts, "Generating Semantics-Based Trajectories of Moving Objects", International Workshop on Emerging Technologies for Geo-based Applications, Ascona, Switwerland, 2000.

[6] Nah, Y.,Wang, T., Kim, K.H., Kim. M.H., and Yang Y.K., "TMO-structured Cluster-based Real-time Management of Location Data on Massive Volume fo Moving Items." in Proc. STFES 2003, IEEE Press, Hakodate, Japan, May 2003.