스트림 환경에서 이동객체 궤적의 효율적 관리
(Efficient Management of Moving Object Trajectories in the Stream Environment)

이 원 철 † 문 양 세 †† 이 상 민 †††
(Won-Cheol Lee) (Yang-Sae Moon) (Sang-Min Rhee)

요 약 센서 네트워크, 위치 기반 서비스 등의 기술 발전에 따라, 최근의 이동객체 위치정보는 연속적이고 끊임없이 변경되는 스트림 데이타 형태를 가지게 되었다. 본 논문에서는 이와 같이 스트림 형태로 발생하는 이동객체의 위치정보를 제한된 메모리에 저장하고, 과거 위치를 추정하는 효율적인 방법을 제안한다. 이를 위하여, 우선 제한된 메모리 양으로 지속적으로 추가되는 이동객체의 과거 위치 이력을 저장하기 위한 위치정보의 **점진적 추출**(*incremental extraction*) 개념을 제시한다. 점진적 추출이란 새로운 위치정보가 추가될 때마다, 시스템이 관리해야 할 과거 위치정보를 기존 위치정보와 새로운 위치정보를 바탕으로 점진적으로 추출하는 방법을 의미한다. 그런 다음, 이러한 점진적 추출 개념을 적용하여 스트림 환경에서 위치정보를 저장 및 추정하는 전체적인 프레임워크를 제시한다. 그리고, 제안한 프레임워크 하에서 추정위치를 계산하는 방법으로 다항식을 이용한 직선기반과 곡선기반 방법을 제시한다. 다음으로, 점진적 추출 개념을 사용하여 과거 위치를 추출하는 방법으로 균등 간격 추출, 기울기 기반 추출, 그리고 최근 시점 강조 추출의 세 가지 방법을 제시한다. 실험 결과, 제안한 점진적 추출 방법은 적은 비율(0.1%)의 위치정보를 저장함에도 불구하고 과거 위치추정에 있어 비교적 높은 정확도(오차율 3% 이내)를 나타냈다. 특히, 곡선기반의 점진적 추출 방법은 전체 위치 데이타의 0.1% 만을 저장하면서도 오차율 1.5% 미만의 높은 정확도를 나타내었다. 이러한 결과로 볼 때, 제안한 방법은 스트림 환경에서 이동객체의 위치정보를 저장하고, 과거 위치를 추정하는 우수한 연구결과라 사료된다.

키워드 : 스트리밍 데이타, 이동객체, 궤적, 점진적 추출, 과거 위치 추정

Abstract Due to advances in position monitoring technologies such as global positioning systems and sensor networks, recent position information of moving objects has the form of streaming data which are updated continuously and rapidly. In this paper we propose an efficient trajectory maintenance method that stores the streaming position data of moving objects in the limited size of storage space and estimates past positions based on the stored data. For this, we first propose a new concept of incremental extraction of position information. The incremental extraction means that, whenever a new position is added into the system, we incrementally re-compute the new version of past position data maintained in the system using the current version of past position data and the newly added position. Next, based on the incremental extraction, we present an overall framework that stores position information and estimates past positions in the stream environment. We then propose two polynomial-based methods, line-based and curve-based methods, as the method of estimating the past positions on the framework. We also propose three incremental extraction methods: equi-width, slope-based, and recent-emphasis extraction methods. Experimental results show that the proposed incremental extraction provides the relatively high accuracy (error rate is less than 3%) even though we maintain only a little portion (only 0.1%) of past position information. In particular, the curve-based incremental extraction provides very low error rate of 1.5% even storing 0.1% of total position data. These results indicate that our incremental extraction methods provide an efficient framework

· 본 연구는 첨단정보기술연구센터를 통하여 과학기술부/한국과학재단의 지원을 받았음
† 학생회원 : 강원대학교 컴퓨터과학과
woncheol@kangwon.ac.kr
†† 정 회 원 : 강원대학교 컴퓨터과학과 교수
ysmoon@kangwon.ac.kr
††† 정 회 원 : 강원대학교 컴퓨터과학과 교수
smrhee@kangwon.ac.kr
논문접수 : 2007년 1월 22일
심사완료 : 2007년 5월 4일

for storing the position information of moving objects and estimating the past positions in the stream environment.

Key words : Streaming data, Moving objects, Trajectory, Incremental extraction, Past positions estimation

1. 서 론

시간의 흐름에 따라 연속적으로 위치가 변화하는 객체를 이동객체(moving object)라 한다. 이동통신, GPS (Global Positioning System) 등의 위치 측정 및 전달 기술의 발달에 따라, 이러한 이동객체에 대한 다양한 서비스 요구와 관심이 증가하고 있다. 이러한 이동객체 개념은 동물군 관찰 시스템, 기상 관측, 교통 통제 시스템, 비디오 데이타 계열 검색 등에서 널리 응용되며, 최근에는 위치 기반 서비스(LBS: Location-Based Service) 및 텔레매틱스 분야에서 활발한 연구가 진행되고 있다[1]. 그리고, 향후 센서 네트워크[2] 및 유비쿼터스(ubiquitious)[3] 환경에서는 이동객체에 대한 더욱 다양한 서비스 및 기술이 요구될 것으로 예상된다.

전통적인 이동객체 연구에서는 주로 객체의 현재 위치, 과거 위치, 미래 위치를 저장하거나 추정하는 내용을 다루었다[4-12]. 우선, 이동객체의 위치를 빠르게 검색하기 위한 색인 기법이 많이 연구되었다[4-8]. 다음으로, 이동객체의 위치를 추정하기 위한 방법으로는 과거 시점의 궤적을 추정하는 방법과 미래 시점의 위치를 예측하는 방법이 연구되었다[9-12]. 이중 과거 궤적을 추정하는 방법은 시스템에 저장된 두 시점 이상의 과거 위치를 사용하여, 저장되지 않은 시점의 이동객체 위치를 추정하는 방법이다. 그리고 미래 위치 예측 방법은 과거 시점의 위치정보, 이동 방향, 이동 속도 등을 사용하여 가까운 미래의 위치를 추정하는 방법이다.

본 논문에서는 메모리 제약이 따르는 스트림 환경[16,17]에서의 위치정보 저장 및 추정 방법을 제시한다. 센서 네트워크, LBS 등의 기술발전에 따라 최근의 이동객체는 자신의 위치를 연속적이고 끊임없이 보고하기에 이르렀다. 즉, 이동객체의 위치정보 자체가 스트림 형태로 발생하는 환경이 현실화되고 있다[1]. 그런데, 기존 연구로서는 이와 같이 스트림 형태로 발생하는 이동객체의 위치정보 저장 및 추정에 어려움이 있다. 이는 기존 연구 방법이 이동객체의 위치정보를 메모리 제약 없이 저장하고, 이를 기반으로 위치를 추정하기 때문이다. 반면에, 스트림 환경에서는 지속적으로 끊임없이 발생하는 데이타를 모두 메모리에 저장할 수 없게 된다. 따라서, 스트림 환경에서는 메모리에 저장할 수 있는 위치정보에 한계가 있음을 제약사항으로 하고, 이에 따른 위치정보 저장 및 추정 방법이 필요하다. 본 논문에서는 이와 같이 메모리 제약이 따르는 스트림 환경에서의 위치정보 저장 및 추정 방법을 제안한다.

본 논문에서는 다음과 같은 과정으로 스트림 환경에서의 위치정보 저장 및 추정 방법을 제안한다. 우선, 이동객체가 시스템에 보고하는 위치와 시스템에 의해 추정되는 이동객체의 위치를 구분하기 위하여, 실제위치와 추정위치의 개념을 정의한다. 그리고, 이를 기반으로 본 논문에서 해결하고자 하는 스트림 환경에서 이동객체의 과거 위치 추정 문제를 정형적으로 정의한다. 다음으로, 제한된 메모리 양으로 지속적으로 추가되는 위치정보의 과거 이력을 저장하기 위한 위치정보의 **점진적 추출** (*incremental extraction*) 개념을 제시한다. 점진적 추출이란 새로운 위치정보가 추가될 때마다, 시스템이 관리해야 할 과거 위치정보를 기존 위치정보와 새로운 위치정보를 바탕으로 점진적으로 추출하는 방법을 의미한다. 즉, 과거의 모든 위치정보를 저장할 수 없는 스트림 환경에서, 이미 추출하여 관리하는 기존의 과거 위치를 대상으로 새롭게 관리할 과거 위치를 추출하는 방법을 점진적 추출이라 정의한다.

다음으로, 점진적 추출 개념을 사용하여 스트림 환경에서 위치정보를 저장 및 추정하는 전체적인 동작 프레임워크를 제시한다. 제안한 프레임워크는 크게 저장된 과거 위치를 기반으로 추정위치를 계산하는 방법과 점진적 추출을 통해 새롭게 저장할 과거 위치를 추출하는 방법으로 구성된다. 먼저, 추정위치를 계산하는 방법으로는 다항식을 사용한 직선기반과 곡선기반의 두 가지 방법을 제시한다. 다음으로, 과거 위치 추출 방법으로는 균등 간격 데이타 추출, 기울기 기반 데이타 추출, 그리고 최근 시점 강조 데이타 추출의 세 가지 방법을 제시한다. 마지막으로, 다양한 실험을 통하여 제시한 방법들의 특징을 분석한다. 실험 결과, 제안한 점진적 추출 방법은 적은 비율(0.1%)의 위치정보를 저장함에도 불구하고 과거 위치 추정에 있어 비교적 높은 정확도(오차율 3% 이내)를 나타냈다. 이러한 결과는 제안한 방법이 스트림 환경에서 위치정보를 저장 및 추정하는 우수한 프레임워크를 제공함을 의미한다.

본 논문의 구성은 다음과 같다. 제2장에서는 본 연구와 관련된 기존 연구를 소개한다. 제3장에서는 본 논문에서 다루는 문제를 정의한다. 제4장에서는 과거 위치의 점진적 추출 개념을 제시하고, 위치정보 저장 및 추정을 위한 전체적인 동작 프레임워크를 제시한다. 그리고, 이

에 기반한 여러가지 저장 및 추정 방법을 제안한다. 제5장에서는 성능 평가 결과를 제시하고, 마지막으로 제6장에서는 결론을 맺는다.

2. 관련 연구

본 논문에서는 스트림 형태로 발생하는 이동객체의 위치에 대해 과거 시점의 위치를 추정하는 방법을 다룬다. 이와 관련된 연구로는 이동객체의 저장과 관리방법, 궤적 추정 방법, 시계열 데이타베이스의 유사 시퀀스 매칭 연구 등이 있다.

전통적인 이동객체 연구에서는 주로 객체의 현재 위치, 과거 위치(궤적), 미래 위치를 저장하거나 추정하는 내용을 다루었다[4-12]. 특히, 객체의 위치를 빠르게 검색하기 위한 색인 기법이 많이 연구되었으며, 이들 색인 방법은 다시 현재 위치를 검색하기 위한 방법과 미래 위치를 예측하기 위한 방법으로 분류할 수 있다. 현재 위치를 검색하기 위한 색인으로는 격자파일(grid file)[4], 사분트리(quad-tree)[5]와 같은 공간 분할 색인과 R-트리[6]와 같은 데이타 분할 색인 이 주로 사용되었다. 반면에 미래 위치를 예측하기 위한 색인 방법에는 TPR-트리[10]와 같이 기존 트리에 속도 파라미터 값을 추가하여 객체의 미래 위치를 예측하는 방법이 연구되었다. 또한, 과거에서 현재까지의 위치 변화를 추적(trace)하는 궤적 검색을 위한 색인으로 HR-트리[7], TB-트리[8] 등이 연구되었다. 본 연구에서 다루는 위치 추정 방법은 과거 위치를 추정한다는 측면에서 궤적 검색을 위한 색인과 유사한 측면이 있다. 그러나 궤적 검색을 위한 색인은 객체의 과거 위치를 모두 저장하는 환경을 가정한 반면에, 본 연구의 추정 기법은 과거 위치를 모두 저장하지 못하는 스트림 환경을 대상으로 한다는 점에서 차이가 있다.

이동객체의 위치를 추정하는 방법으로는 과거 시점의 궤적을 추정하는 방법과 미래 시점의 위치를 예측하는 방법이 연구되었다. 과거 궤적을 추정하는 방법은 시스템에 저장된 두 개 이상의 과거 위치를 사용하여, 저장되지 않은 시점의 위치(궤적)를 추정하는 방법이다. 즉, 시간 t_1과 t_2에 저장된 위치정보를 사용하여 t_1과 t_2 사이에 있는 시간 t_k ($t_1 < t_k < t_2$)의 위치를 추정하는 방법이다[8,9,11]. 이러한 과거 궤적 추정 방법은 다시 직선(line) 기반의 추정 방법과 곡선(curve) 기반의 추정 방법으로 구분할 수 있다. 직선기반의 궤적 추정방법은 이동객체가 갖는 임의의 움직임에 따른 복잡성을 피할 수 있고, 움직임의 형태를 단순하게 추정할 수 있는 장점이 있다[8]. 그러나 실세계에서 이동객체는 직선이라기 보다는 복잡하고 다양한 이동 패턴을 가지기 때문에, 직선기반의 추정 방법은 실세계의 문제를 제대로 해결할 수 없다. 이를 해결하기 위하여 제안된 방법이 곡선기반의 궤적 추정 방법이다[9,11]. 다음으로, 이동객체의 과거 위치정보를 사용하여 미래의 위치를 예측하는 방법이 연구되었다. 이 방법 또한 과거 궤적 추정 방법과 같이 직선기반의 예측 방법[10]과 곡선기반의 예측 방법[12]으로 구분할 수 있다. 그러나 이들 과거 궤적 및 미래 위치 추정 방법은 과거 시점의 객체 위치를 모두 저장할 수 있는 환경을 가정하고 있다. 따라서 이들 연구는 본 논문에서 다루는 스트림 환경에는 직접 적용이 어려운 문제점이 있다.

시계열 데이타베이스의 유사 시퀀스 매칭[13,14,18]에서 연구된 저차원으로 변환(lower-dimensional transformation)도 잦은 위치 변경이 일어나는 이동객체의 과거 혹은 미래 위치 추정에 활용할 수 있다. 즉, 이동객체의 위치를 시계열 데이타 형태로 나타내고, 이를 저차원 변환하여 저장, 관리, 예측에 활용하는 것이다. 저차원 변환 방법으로는 이산 퓨리에 변환(DFT: discrete fourier transform)[13], 이산 웨이블릿 변환(DWT: discrete wavelet transform)[14], APCA(adaptive piecewise constant approximation)[15] 등이 사용되었다. 이러한 저차원 변환은 제4장에서 제안하는 위치 추정 프레임워크에서 저장 및 관리해야 하는 위치정보를 줄이기 위한 한 가지 방법으로 해석할 수 있다. 그러나, 본 논문에서는 위치정보를 줄이기 위한 방법으로 직선 및 곡선의 다항식 방법을 주로 다루며, 저차원 변환을 이용한 방법은 향후 연구로 남겨 둔다.

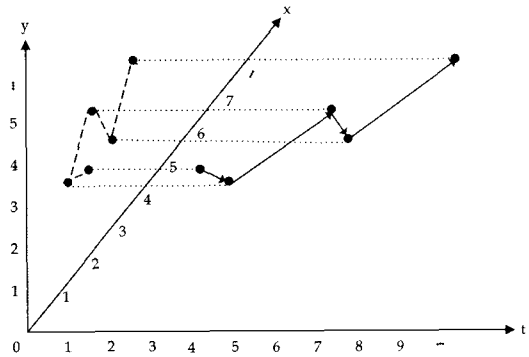
3. 문제 정의

스트림 환경에서는 이동객체의 위치가 지속적으로 갱신되고, 이러한 이동객체의 특정 시점 위치는 다차원 데이타로 표현할 수 있다. 예를 들어, 기차, 이동전화, 축구선수 등의 이동에 따른 위치 데이타들은 수직, 수평의 좌표 값인 2차원 데이타로 표현할 수 있고, 비행기 시뮬레이션 데이타는 2차원 데이타에 고도 정보를 추가하여 3차원 데이타로 나타낼 수 있다. 또한, 기상 관련 데이타 중 태풍(풍속, 풍향, 이동경로)은 4차원 이상의 데이타로 표현된다.

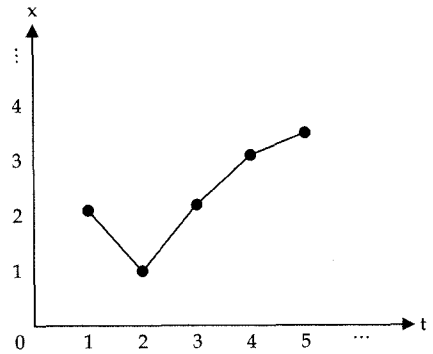
시간의 흐름에 따라 지속적으로 변화하는 이동객체의 위치를 특정 차원을 중심으로 표현하면, (시점, 해당 차원의 위치 값)을 쌍으로 하는 이차원 데이타로 나타낼 수 있다. 이를 설명하기 위해 이동객체를 O라 하고, O의 2차원 위치정보를 (x,y)로 표현한다. 그리고 객체 O는 시간의 흐름에 따라 이동하므로, 시간 정보를 추가하여 시점 t에서의 객체 O의 위치정보를 (t,x,y)로

표 1 이동객체 O의 위치 데이타 예제

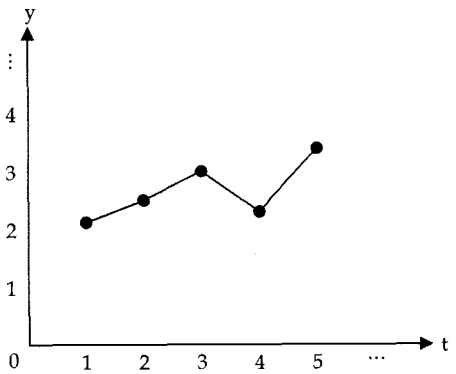
구분	위치 데이타
(t,x,y)	(1, 2.1, 2.2), (2, 1.0, 2.5), (3, 2.2, 3.0), (4, 3.1, 2.3), (5, 3.5, 3.4)
(t,x)	(1, 2.1), (2, 1.0), (3, 2.2), (4, 3.1), (5, 3.5)
(t,y)	(1, 2.2), (2, 2.5), (3, 3.0), (4, 2.3), (5, 3.4)



(a) (t,x,y)의 좌표 표현



(b) (t,x)의 좌표 표현



(c) (t,y)의 좌표 표현

그림 1 이동객체 O의 위치 데이타 표현 예제

나타낸다. 그러면 이동객체 O의 x축 및 y축 위치정보를 시간 축으로 구분하여, x축의 경우 (t,x), y축의 경우 (t,y)로 나타낼 수 있다. 표 1은 이러한 표기법에 따른 객체 O의 위치 변화의 예제를 나타낸 것이고, 그림 1은 표 1의 위치 데이타를 좌표에 표현한 것이다. 그림 1(a)는 (t,x,y) 형태의 위치정보를 3차원 좌표에 표현한 것이고, 그림 1(b)와 그림 1(c)는 x축 및 y축을 각각 시간축으로 구분하여 나타낸 것이다. 이와 같이 모든 다차원 위치 정보는 시간 축을 기준으로 하는 이차원 데이타로 표현할 수 있고, 따라서 본 논문에서는 이동객체의 위치가 (시간 값, 위치 값)의 쌍, 특히 (t,x)와 같이 2차원으로 나타난다고 가정한다.

이동객체의 위치가 보고되는 시점을 $t_i(1 \le i \le n)$라 하자, 즉, 이동객체의 위치정보는 각 시점 t_i마다 지속적으로 갱신된다고 하자. 본 논문에서는 이동객체의 위치를 객체가 실제 이동했던 위치인지, 아니면 추정된 위치인지의 여부에 따라 실제위치 또는 추정위치로 구분한다. 이들을 다음과 같이 정형적으로 정의한다.

정의 1. 이동객체 O에 대해 보고된 위치 데이타 집합이 $\mathbb{S} = \{(t_1,x_1),(t_2,x_2),\cdots,(t_n,x_n)\}$이라 할 때, 시점 $t_k(\in \mathbb{S})$에서의 **실제위치**(*actual position*)란, 이동객체 O가 시점 t_k에 실제로 존재했던 위치인 x_k를 의미한다. □

정의 2. 이동객체 O에 대해 보고된 위치 데이타 집합이 $\mathbb{S} = \{(t_1,x_1),(t_2,x_2),\cdots,(t_n,x_n)\}$이라 할 때, 시점 $t_k(\in \mathbb{S})$에서의 **추정위치**(*estimated position*)란, 오직 $m << n$개의 위치 데이타 집합 $\mathbb{S}' = \{(t_j,x_j) \mid ((t_j,x_j) \notin \mathbb{S}) \vee ((t_j,x_j) \in \mathbb{S})\}$를 저장한 상태에서 추정한 시점 t_k에서의 이동객체 O의 위치이다. □

스트림 환경에서는 이동객체의 위치를 저장할 수 있는 메모리 양에 한계가 있다. 따라서, 본 논문에서는 메모리 양이 제한된 스트림 환경에서 정의 2의 추정위치를 얻는 문제를 다룬다.

정의 3. 스트림 환경에서 이동객체의 과거 위치추정이란, 이동객체에 대해 위치정보를 저장할 수 있는 메모리 양이 제한되어 있을 때, 지속적으로 변화하는 객체의 위치 데이타를 반영하면서 임의 시점에서의 추정위치를 계산하는 방법이다. □

본 논문의 제4장에서는 다항식을 사용하여 정의 3의 스트림 환경에서 이동객체의 추정위치를 계산하는 방법을 제안한다.

4. 스트림 환경에서 과거 위치 추정 방법

본 장에서는 스트림 환경에서 이동객체의 추정위치를 계산하기 위하여, 점진적 추출 개념을 이용하는 동작 프레임워크를 제4.1절에서 제시하고, 추정위치 계산 방법과 위치정보 추출 방법은 제4.2절에서와 제4.3절에서 각각 제안한다.

4.1 동작 프레임워크

동작 프레임워크는 크게 두 가지 과정으로 구성된다. 첫째는 제한된 메모리 양으로, 지속적으로 변화하는 객체의 위치 데이타를 저장하는 과정이다. 둘째는 저장된 위치 데이타를 사용하여 과거 임의 시점의 추정위치를 계산하는 과정이다.

그림 2는 스트림 환경에서 추정위치를 계산하는 전체 프레임워크를 나타낸다. 그림을 보면, 우선 위치정보 저장소에는 각 이동객체에 대해 m 개의 제한된 정보를 저장, 관리한다. 그리고, 위치정보 반영 알고리즘은 이동객체의 위치가 변경되었을 때, 해당 객체에 대해 관리하는 m 개의 정보를 점진적 추출을 통해 새롭게 변경하는 기능을 수행한다. 즉, 객체가 이동하여 새로운 위치가 보고되었을 때, 이를 저장된 m 개의 정보에 반영하기 위한 알고리즘으로, 이 과정에서 위치정보의 점진적 추출 개념을 사용한다. 다음으로, 과거 위치추정 알고리즘은 사용자가 제시한 과거 임의 시점에 대한 추정위치를 계산한다. 이 알고리즘에서는 위치정보 저장소에 저장된 m 개의 정보를 사용하여 사용자가 제시한 시점의 추정위치를 계산하여 반환한다.

그림 2의 프레임워크를 사용하기 위해서는 1) 저장된 정보로 어떻게 추정위치를 계산할지와 2) 위치정보 저장소의 제한된 메모리에 어떤 정보를 저장할지를 결정해야 한다. 따라서, 다음의 제4.2절에서는 우선 주어진 m 개의 정보가 있다고 가정할 때 과거 임의 시점의 추정위치를 계산하는 방법을 제시한다. 다음으로, 제4.3절에서는 위치정보 저장소에 관리할 m 개의 제한된 정보를 어떻게 결정하는지의 방법을 제안한다.

4.2 추정위치 계산 방법

이동객체 O에 대해 저장된 위치 데이타 집합이 $\mathbb{S}=\{(t_1,x_1),(t_2,x_2),\cdots,(t_n,x_n)\}$ 이라 할 때, 추정위치를 계산하는 방법은 집합 $\mathbb{S}$를 기반으로 임의의 시점 t_k에 대한 객체 O의 위치 x_k를 계산하는 것이다. 본 논문에서는 이러한 추정위치를 계산하는 방법으로 다항식[9,11,19]을 사용한다. 특히 주어진 시점과 인접한 두 개의 위치를 사용하는 직선기반 추정 방법과 세 개 이상의 위치를 사용하는 곡선기반 추정 방법을 사용한다. 그림 3은 위치를 추정하는 알고리즘이다. 라인 (1)은 질의 시점 t와 가장 인접한 시점에 있는 데이타 집합을 찾는다. 다음으로, 라인 (2)는 검색된 데이타 집합을 지나는 다항식 $f(t)$를 계산한다. 마지막으로, 라인 (3)은 계산된 다항식 $f(t)$에 질의 시점 t를 대입하여 추정 위치 x를 계산한다.

Algorithm Compute_Position(Query time t)
(1) Find S which is the nearest data set at t; // S is subset of $\mathbb{S}$
(2) Make a polynomial function $f(t)$ using S;
(3) Compute a position x at t using $f(t)$;
(4) Return x;

그림 3 과거 위치추정 알고리즘

직선기반 추정 방법

직선기반 추정 방법[8]은 임의의 시점 t_k에 대해 인접한 두 개의 시점 t_i와 t_{i+1} $(t_i \le t_k \le t_{i+1})$에 저장된 위치정보 x_i와 x_{i+1}을 사용하는 방법이다. 우선, 두 점

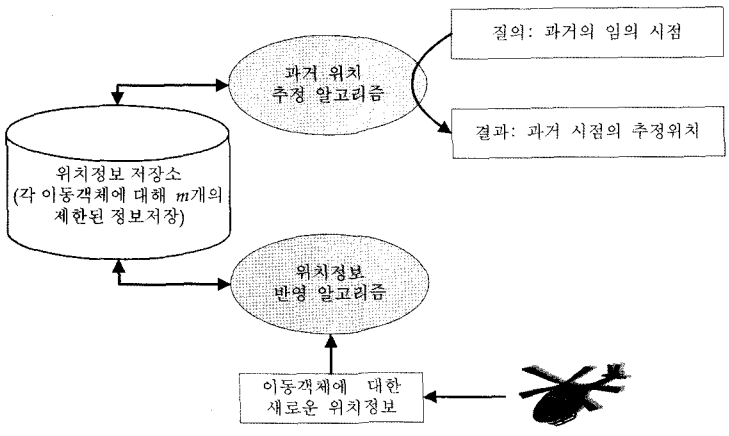


그림 2 스트림 환경에서 과거 위치 추정 프레임워크

(t_i, x_i)와 (t_{i+1}, x_{i+1})을 2차원 좌표상에서 직선으로 연결한 방정식 $f(t)$을 계산한다. 그런 다음, 방정식 $f(t)$에 질의 시점 t_k를 대입하여 추정위치 x_k를 $f(t_k)$로 계산하는 것이다. 2차원 좌표의 두 지점 (t_i, x_i)와 (t_{i+1}, x_{i+1})이 주어졌을 때, 이 두 좌표 값을 지나는 직선의 방정식은 다음 공식 (1)과 같이 쉽게 구할 수 있다.

$$f(t) = \frac{(t_{i+1} - t) \cdot x_i + (t - t_i) \cdot x_{i+1}}{t_{i+1} - t_i} \quad (1)$$

곡선기반 추정 방법

곡선기반 추정 방법[9,11]은 두 개의 위치 보다 많은 즉, 세 개 이상의 위치정보를 사용한다. 문제를 간단히 하기 위해서, 본 논문에서는 질의 시점에 인접한 네 개(앞과 뒤로 각 두 개)의 위치 데이타를 이용한다. 즉, 임의의 시점 t_k $(t_{i+1} \le t_k \le t_{i+2})$에 대해 인접한 네 개의 시점 t_i, t_{i+1}, t_{i+2}, t_{i+3}에 저장된 위치정보 x_i, x_{i+1}, x_{i+2}, x_{i+3}를 사용하는 방법이다. 이를 위해, 네 점 (t_i, x_i), (t_{i+1}, x_{i+1}), (t_{i+2}, x_{i+2}), (t_{i+3}, x_{i+3})를 지나는 3차 다항식 $f(t)$를 구하고, 이 다항식 $f(t)$에 질의 시점 t_k를 대입하여 추정위치 x_k를 $f(t_k)$로 계산한다.

곡선기반 추정방법에서 계산에 사용되는 위치를 네 개로 제한한 이유는 다음과 같다. 첫째, 많은 위치정보를 사용할 경우 차수가 높아져 계산 비용이 많이 들기 때문이다. 둘째, 차수가 높아지면 불필요한 진동으로 인해 오히려 오차가 커지기 때문이다[19]. 이에 따라 기본적으로 네 개의 위치정보를 사용하여 3차 다항식을 구성하고, 이를 기반으로 추정위치를 보간한다. 단, 질의 시점 t_k가 처음 입력 시점인 t_1과 t_2사이에 있거나 마지막 입력 시점인 t_m과 t_{m-1}사이에 있을 경우는 네 개의 정보를 구할 수 없으므로, 예외처리하여 인접한 세 개의 위치정보만을 사용한다. 예를 들어, t_k가 t_1과 t_2 사이에 있을 때는 인접한 세 개의 위치정보로 (t_1, x_1), (t_2, x_2), (t_3, x_3)를 사용하며, t_{m-1}과 t_m사이에 있을 때에는 인접한 세 개의 위치정보로 (t_{m-2}, x_{m-2}), (t_{m-1}, x_{m-1}), (t_m, x_m)를 사용한다. 그리고 이들 예외처리에서는 3차 다항식 대신 2차 다항식을 사용한다.

본 논문에서는 다항식 계산을 위해 라그랑제 기초 보간함수(Lagrange interpolation basis function)를 사용한다. 이는 추정에 사용되는 위치정보 개수가 네 개로 제한되어 있어 네빌레(Nevile)의 반복 보간법이나 뉴튼(Newton)의 분할 차분 보간법과 같은 복잡한 기법을 사용할 필요가 없기 때문이다. 또한 저장된 좌표 값들이 등간격이라는 보장이 없기 때문에 Chebyshev 다항식 등의 사용이 어렵기 때문이다[9]. 따라서, 본 논문에서는 고정된 데이타 수에 적합하고, 등간격이 아닌 경우에도 다항식 계산이 가능한 라그랑제 기초 보간함수를 선택하였다. 좌표상에서 m개의 위치정보 $\{(t_1, x_1), (t_2, x_2), \cdots, (t_m, x_m)\}$과 질의 시점 t_k $(t_{i+1} \le t_k \le t_{i+2})$가 주어졌을 때, t_k에 인접한 네 좌표 (t_i, x_i), (t_{i+1}, x_{i+1}), (t_{i+2}, x_{i+2}), (t_{i+3}, x_{i+3})을 지나는 곡선의 라그랑제 방정식은 다음 공식 (2)와 같이 구할 수 있다[19].

$$f(t) = \sum_{c=i}^{i+3} x_c \prod_{j=i, j \ne c}^{i+3} \frac{(t_k - t_j)}{(t_c - t_j)} \quad (2)$$

4.3 위치정보의 점진적 추출

본 절에서는 지속적인 위치 변경이 일어나는 스트림 환경에서, 제한된 m개의 정보를 점진적으로 추출하는 방법을 제안한다. 우선, 위치정보의 점진적 추출을 다음과 같이 정의한다.

정의 4. 위치정보의 **점진적 추출**(*incremental extraction*)이란 m개의 제한된 메모리에 저장된 위치정보 $\{(t_1, x_1), (t_2, x_2), \ldots, (t_m, x_m)\}$에 새로운 위치정보 (t_{m+1}, x_{m+1})이 입력되었을 때, (t_{m+1}, x_{m+1})을 반영한 m개의 새로운 위치정보 $\{(t'_1, x'_1), (t'_2, x'_2), \ldots, (t'_m, x'_m)\}$을 추출하는 방법이다. □

위치정보의 점진적 추출로는 시간 축을 기준으로 균등간격으로 위치정보를 추출하는 방법과 균등하지 않은 간격으로 위치정보를 추출하는 방법을 제시한다. 또한, 균등하지 않은 경우는 다시 기울기 기반 추출과 최근 시점 강조 추출로 나누어 제안한다.

그림 4는 새로운 위치정보가 보고되었을 때, 제한된 저장공간에 위치정보를 점진적으로 추출하는 알고리즘이다. 라인 (1)과 (2)는 시스템에 저장 가능한 데이타 개수를 넘지 않는 경우로, 보고되는 위치정보는 모두 저장된다. 라인 (3)에서 (8)은 저장될 수 있는 데이타의 개수를 초과하는 경우이다. 라인 (4)는 입력된 첫번째 시점과 마지막으로 입력된 시점을 이용하여 새로운 시간간격 δ를 계산한다. 다음으로, 라인(5)는 새로운 시간간격 δ를 이용하여 새로운 시간 t'을 각각 계산한다. 마지막으로 라인 (6)에서 (8)까지는 각각의 계산된 시간 t'에 대한 추정위치 x'을 계산하여 저장하는 과정으로, 그림 3의 위치추정 알고리즘을 사용한다.

균등간격 위치정보 추출

균등간격 위치정보 추출에서는 m개의 새로운 위치정보를 추출할 때, 시간 축을 기준으로 동일한 시간 간격으로 위치정보를 추출한다. 즉, 위치정보 추출 시점

Algorithm Incremental_Extraction(time t, position x)

(1) **IF** count <= MAX **then** //count: number of data objects which are reported

(2) Store (t, x); //MAX: maximum number of data objects to be stored in the system

(3) **Else**

(4) Compute new Interval δ using t_1 and t_{MAX+1};

(5) Compute new time points t' using t_1 and δ;

(6) **For each** t'

(7) x' = Cpmpute_postion(t')

(8) Save (t', x');

그림 4 위치정보 반영 알고리즘

t'_{i-1}과 시점 t'_i의 간격이 시점 t'_i와 t'_{i+1}의 간격과 동일하다. 균등간격 추출에 대해 직선기반 방법과 곡선기반 방법을 구분하여 설명한다.

직선기반 방법은 새로운 위치정보 (t_{m+1}, x_{m+1})을 포함한 $m+1$개 위치정보에서 균등한 시간 간격을 계산하고, 이를 공식 (1)에 적용하여 새로운 m개의 위치정보를 계산하는 방법이다. 이를 위해, 새로운 위치정보 (t_{m+1}, x_{m+1})가 입력되었을 때, 마지막 입력시간 t_{m+1}과 처음 입력시간 t_1의 차이를 m개 시점 $(t'_1, \cdots, t'_m)$의 간격 수 $m-1$로 나누어 시간간격 $\delta\ (=\frac{t_{m+1}-t_1}{m-1})$를 계산한다. 그리고, 시간간격 δ를 기준으로 각각의 새로운 시점 $t'_i(=t_1+(i-1)\cdot\delta)$를 계산한다. 다음으로, 각 시점 t'_i의 위치 x'_i를 제4.2절의 직선기반 추정 방법의 공식 (1)로 계산한다. 그러나 x'_1과 x'_m은 예외처리하여 계산하지 않는다. 이는 처음 입력시점 t_1이 새로운 시점 t'_1과 동일하고, 마지막 입력시점 t_{m+1}도 새로운 시점 t'_m과 동일하기 때문이다. 따라서 시점 t'_1과 t'_m에 대한 x좌표 값 x'_1과 x'_m은 별도의 계산없이 각각 x_1과 x_{m+1}로 설정한다.

예제 1. 저장 가능한 위치정보 개수가 여덟 개로 제한된 시스템이 있다고 하자. 이 시스템에 위치정보가 {(1, 2.1), (2, 1.0), (3, 2.2), (4, 3.1), (5, 3.5), (6, 4.4), (7, 5.0), (8, 5.8)}로 저장되어 있고, 새로운 위치정보 (9, 6.0)이 입력된다고 하면 점진적 추출을 사용해 새로운 여덟 개의 위치정보를 다음과 같이 계산할 수 있다. 먼저 새로운 시간간격 δ는 $1.14(\approx(t_{m+1}-t_1)/(m-1)=(9-1)/7)$가 된다. 계산된 δ를 기준으로 각 시점 $t'_i=1+(i-1)\cdot 1.14$로 구할 수 있다. 그런 다음, 각 시점에 대해 x'_i를 계산한다. 예를 들어, $t'_2=1+(2-1)\cdot 1.14=2.14$로 구할 수 있고, 새로운 시점 2.14의 x좌표 값을 구하기 위해, 이에 인접한 시간 값 2와 3에 대한 직선의 방정식을 공식 (1)로 계산하면 다음과 같다.

$$f(t) = \frac{(3-t)\cdot 1.0+(t-2)\cdot 2.2}{3-2} = 1.2t-1.4$$

그리고, 계산된 $f(t)$에 시점 2.14를 대입하여 x좌표 값 1.17을 구한다. 나머지 시점 3.29, 4.43 등에 대해서도 같은 방법으로 x'_i을 구할 수 있다. 여기에 처음 시점과 마지막 시점의 위치정보는 기존 값을 그대로 사용한다. 그 결과 새로운 여덟 개의 위치정보는 {(1.00, 2.10), (2.14, 1.17), (3.29, 2.46), (4.43, 3.27), (5.57, 4.01), (6.71, 4.83), (7.86, 5.69), (9.00, 6.00)}이 된다. □

곡선기반 방법은 직선의 방정식 대신 라그랑제 다항식을 사용하고, 두 점 대신 네 점을 사용하는 것을 제외하고는 직선기반 방법과 동일하다. 곡선기반 방법의 절차는 다음과 같다. 먼저 직선기반 방법과 동일한 방식으로 시간 간격 δ와 새로운 시점 $t'_i(=t_1+(i-1)\cdot\delta)$를 계산한다. 다음으로, 각 시점 t'_i의 위치 x'_i를 계산하기 위하여, t'_i와 인접한 시간 쌍 t_{j+1}와 $t_{j+2}\ (t_{j+1}\le t'_i\le t_{j+2})$를 찾아 (t_j, x_j), (t_{j+1}, x_{j+1}), (t_{j+2}, x_{j+2}), (t_{j+3}, x_{j+3})을 지나는 곡선의 방정식 $f(t)$를 공식 (2)로 계산한다. 단, 직선기반 방법과 마찬가지로 x'_1과 x'_m은 예외처리하여 계산하지 않는다.

예제 2. 예제 1에서와 같이 메모리 량이 여덟 개로 제한된 시스템에 위치정보가 {(1, 2.1), (2, 1.0), (3, 2.2), (4, 3.1), (5, 3.5), (6, 4.4), (7, 5.0), (8,5.8)}로 저장되어 있고, 새로운 위치정보 (9, 6.0)이 입력된다고 하자. 그러면 점진적 추출을 통해 새로운 여덟 개의 위치정보를 다음과 같이 계산할 수 있다. 먼저 새로운 시간간격 δ를 직선기반 방법과 동일하게 계산한다. 그런 다음, 각 시점 t'_i에 대해 x'_i를 계산한다. 예를 들어, 새로운 시점 $t'_2=1+(2-1)\cdot 1.14=2.14$로 구할 수 있고, 시점 2.14

의 x 좌표 값을 구하기 위해 이에 인접한 시간 값 1, 2, 3, 4에 대한 곡선의 방정식을 공식 (2)로 계산하면 다음과 같다.

$$f(t) = x_0 L_0(t) + x_1 L_2(t) + x_2 L_2(t) + x_3 L_3(t)$$

함수 $f(t)$를 계산하기 위해 각각의 $L_i(t)$를 다음과 같이 구한다.

$$L_0(t) = \frac{(t-2)(t-3)(t-4)}{(1-2)(1-3)(1-4)}, \cdots, \quad L_3(t) = \frac{(t-1)(t-2)(t-3)}{(4-1)(4-2)(4-3)}$$

최종 계산된 식 $f(t)$에 2.14를 대입하여 x 좌표 값으로 1.09를 구한다.

$$f(2.14) =$$

$$2.1 \cdot L_0(2.14) + 1.0 \cdot L_1(2.14) + 2.2 \cdot L_2(2.14) + 3.1 \cdot L_3(2.14) \approx 1.09$$

나머지 시점 3.29, 4.43 등에 대한 x 좌표 값도 같은 방식으로 구할 수 있다. 단, 처음 시점 t'_1과 마지막 시점 t'_m의 위치정보는 직선기반 방법과 같이 예외처리하여 계산하지 않는다. 그 결과 새로운 여덟 개의 위치정보는 {(1.00, 2.10), (2.14, 1.09), (3.29, 2.50), (4.43, 3.27), (5.57, 4.00), (6.71, 4.83), (7.86, 5.70), (9.00, 6.00)}이 된다. □

기울기 기반 위치정보 추출

기울기 기반 추출 방법은 비균등 간격 위치정보 추출 방법으로서, 기울기 변화가 심한 구간에서는 많은 수의 위치정보를 추출하고 그렇지 않은 구간에서는 적은 수의 위치정보를 추출한다. 즉, 기울기 변화가 심하지 않은 구간은 적은 수의 정보로 간략히 표현하고, 변화가 심한 구간에서는 많은 수의 정보로 자세히 표현하는 것이다.

이 방법은 직선기반 방법에 적용이 가능하며, 그 절차는 다음과 같다. 새로운 위치정보 (t_{m+1}, x_{m+1})가 입력되었을 때, 각각의 인접한 위치정보 (t_i, x_i)와 (t_{i+1}, x_{i+1})에 대한 기울기 값 $\alpha_i = (x_{i+1} - x_i)/(t_{i+1} - t_i)$를 계산한다. 그런 다음, 인접한 두 기울기 α_i와 α_{i+1}의 차이 $|\alpha_i - \alpha_{i+1}|$을 계산하여, 그 값이 가장 작은 구간을 찾아 이를 계산하는데 포함된 (t_{i+1}, x_{i+1})을 위치정보에서 삭제한다. 단, 기울기 차이가 가장 작은 구간이 여러 개 발생할 경우, 그 구간들 중에서 가장 과거 구간에 포함된 위치정보를 삭제한다. 그 결과로 남은 나머지 m 개의 위치정보를 새로운 위치정보로 다시 저장한다. 다음 예제 3은 기울기 기반 데이타 추출방법의 예를 나타낸다.

예제 3. 예제 1에서와 같이 여덟 개의 위치정보가 {(1, 2.1), (2, 1.0), (3, 2.2), (4, 3.1), (5, 3.5), (6, 4.4), (7, 5.0), (8, 5.8)}로 저장되어 있고, 새로운 위치정보 (9, 6.0)이 입력된다고 하자. 그러면 점진적 추출을 통해 새로운 여덟 개의 위치정보를 다음과 같이 계산할 수 있다. 먼저 저장된 위치정보의 인접한 쌍에 대하여 기울기 $\alpha_1 = -1.1 (=(1.0-2.1)/(2-1))$로, $\alpha_2 = 1.2$, $\alpha_3 = 0.9$, $\cdots$, $\alpha_8 = 0.2$로 계산한다. 그리고 인접한 기울기 차이 $|\alpha_i - \alpha_{i+1}|$를 각각 계산하면 2.3, 0.3, 0.5, 0.5, 0.3, 0.2, 0.6이 된다. 이때, 가장 작은 값이 0.2이므로, 점 (7, 5.0)을 삭제하고 남은 여덟 개 {(1, 2.1), (2, 1.0), (3, 2.2), (4, 3.1), (5, 3.5), (6, 4.4), (8, 5.8), (9, 6)}만을 새로운 위치정보로 저장하여 관리한다. □

최근 시점 강조 위치정보 추출

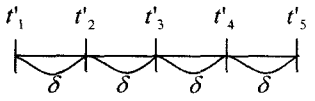
이 방법은 이동객체 데이타베이스 응용에서 최근의 위치정보는 자주 이용하지만, 과거의 위치정보는 자주 이용하지 않는 특성을 고려한 방법이다. 즉, 최근의 위치정보는 많은 수를 추출하고, 과거의 위치정보는 적은 수를 추출하는 방법이다. 따라서 최근 시점 강조 데이타 추출 방법은 비균등 위치정보 추출 방법으로, 과거 시점에 대해 다른 비율로 위치정보를 추출한다.

비균등 간격 데이타 추출을 위해서는 위치정보를 추출하는 시점을 달리해야 한다. 그림 3은 균등 간격과 비균등 간격 정보 추출에 따른 시점 변화를 나타낸다. 그림 5(a)는 균등 간격으로 시점간의 시간 간격이 δ로 동일하다. 반면에, 그림 5(b)는 비균등 간격으로 시점간의 시간 간격 δ_i가 동일하지 않다. 이러한 비균등 간격 정보추출을 위해서는 각 시점 간의 간격인 δ_i를 구해야 한다. 시간 간격 δ_i가 일정한 규칙을 가져야 정형적인 모델링이 가능하므로, δ_i가 $f(t)$와 같이 함수로 나타난다고 가정한다. 즉, $\delta_i = f(t)$라 했을 때, 다음 공식 (3)에 의해 새로운 시점 t'_i를 구할 수 있다.

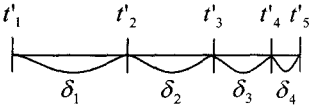
$$t'_i = \begin{cases} t_1 & \\ t_1 + \sum_{k=1}^{i-1} \delta_k & if \quad 1 < i < m; \\ t_{m+1} & if \quad i = m; \end{cases} \qquad (3)$$

공식 (3)을 사용하여 비균등 간격 정보 추출을 위해서는 함수 $f(t)$를 결정해야 한다. 함수 $f(t)$로는 그림 5(b)와 같이 $f(t) = (t_{m-1} - t_1)/2^{t-1}$로 하여 간격을 절반씩 줄여 나가거나, $f(t) = (m-k)\theta$ $(\theta = (t_{m+1} - t_1)/((m(m-1)/2))$로 하여 각 간격을 일정 간격$(\theta)$의 배수로 하는 등의 방법을 사용할 수 있다. 또한, 공식 (3)의 관점에서 보면, 그림 5(a)의 균등 간격 추출방법은 $f(t) = (t_{m+1} - t_1)/(m-1)$로 하는 비균등 간격의 특수한 형태로 해석할 수 있다.

앞서 제시한 비균등 간격 정보추출 방법은 균등 간격



(a) 균등 간격 정보 추출



(b) 비균등 간격 정보추출

그림 5 간격 설정에 따른 정보 추출 시점의 변화

과 마찬가지로 직선기반 및 곡선기반 방법에 모두 적용될 수 있다. 즉, 비균등 간격으로 구한 시간 t'_i를 직선기반의 경우 공식 (1)에 적용하고, 곡선기반의 경우 공식 (2)에 적용하면 된다. 다음 예제 4는 직선기반을, 예제 5는 곡선기반을 사용한 비균등 간격 정보추출의 예이다.

예제 4. 예제 1에서와 같이 여덟 개의 위치정보 {(1, 2.1), (2, 1.0), (3, 2.2), (4, 3.1), (5, 3.5), (6, 4.4), (7, 5.0), (8, 5.8)}와 새로운 위치정보 (9, 6.0)이 주어졌다고 가정하자. 그러면 점진적 추출을 통해 여덟 개의 새로운 위치정보를 다음과 같이 계산할 수 있다. 이때 시간 간격 δ_i를 계산하는 함수로는 각 간격을 일정간격 $\theta=(t_9-t_1)/((8(8-1)/2)$의 배수로 하는 함수 $f(t)=(8-k)\theta$를 사용한다. 그러면 공식 (3)을 이용하여 새로운 시점 t'_i을 계산할 수 있다. 예를 들어, 새로운 시점 t'_2는 $3.0\ (=t_1+\sum_{k=1}^{i-1} f(t)=1+2)$으로 계산된다. 새로운 시점을 모두 구한 후에는, 각 시점에 대한 새로운 위치정보를 균등 간격 정보추출의 직선기반 방법과 동일한 과정으로 구할 수 있다. 새로운 시점 t'_2 (=3.0)의 x 좌표 값을 구하기 위해 인접한 시간 값 t_3 (=3)와 t_4 (=4)에 대한 직선의 방정식을 공식 (1)로 계산하면 다음과 같다.

$$f(t) = \frac{(t_4-t)\cdot 2.2+(t-t_3)\cdot 3.1}{t_4-t_3} = \frac{(4-t)\cdot 2.2+(t-3)\cdot 3.1}{4-3} = 0.9t-0.5$$

그리고, 계산된 $f(t)$에 시점 $3(=t'_2)$를 대입하여 x 좌표 값 2.2을 구한다. 나머지 시점 t'_i에 대해서도 같은 방법으로 x'_i을 구할 수 있다. 그 결과 새로운 여덟 개의 위치정보는 {(1.00, 2.10), (3.00, 2.20), (4.71, 3.39), (6.14, 4.49), (7.29, 5.23), (8.14, 5.83), (8.71,5.94), (9.00, 6.00)}이 된다. □

예제 5. 예제 4와 동일한 조건이되, 직선 대신 곡선기반 방식을 사용하면 다음과 같다. 먼저, 예제 4의 비균등 시점 추출과 동일한 방법으로 새로운 시점 $\{t'_i\}$= {1.00, 3.00, 4.71, 6.14, 7.29, 8.14, 8.71, 9.00}를 계산한다. 그런 다음, 각 시점 t'_i에 대한 새로운 위치정보 x'_i를 곡선기반 방법과 동일하게 구할 수 있다. 단, t_7보다 큰 새로운 시점 8.14와 8.71은 시점 9(t_9)보다 큰 값이 없으므로 인접한 네 점으로 계산할 수 없다. 따라서 세 개의 위치정보 (7, 5.0), (8, 5.8), (9, 6.0)만을 사용하여 계산한다. 그 결과, 새로운 위치정보는 {(1.00, 2.10), (3.00, 2.20), (4.71, 3.38), (6.14, 4.49), (7.29, 5.24), (8.14, 5.87), (8.71, 6.00)(9.00, 6.00)}이 된다. □

5. 성능 평가

본 장에서는 제안한 위치정보 추출 방법에 대한 성능평가 결과를 설명한다. 제5.1절에서는 성능평가를 수행한 실험 데이타와 실험 환경을 설명하고, 제5.2절에서는 실험 결과를 제시한다.

5.1 실험 데이타 및 환경

제안한 방법의 우수성을 입증하기 위하여 추정위치에 대한 오차 개념을 사용한다. 본 논문에서는 임의의 시점에서 실제위치와 추정위치 간 차이의 상대적 비율을 오차로 정의하였다. 다음 공식 (4)는 시점 t에서의 실제위치에 대한 추정위치의 오차(t)를 나타낸 것이다. 수식 (4)에서 실제위치(t)와 추정위치(t)는 시점 t에서의 실제위치와 추정위치를 각각 나타낸다. 그리고, 실제위치 최대값과 최소값은 지금까지 발생한 이동객체의 실제위치 최대값과 최소값을 의미하며, 이는 상대적 비율을 계산하기 위해 사용한다.

$$\text{오차}(t) = \frac{|\text{실제위치}(t)-\text{추정위치}(t)|}{\text{실제위치 최대값}-\text{실제위치 최소값}} \quad (4)$$

실험 데이타 집합은 기존 연구[8]에서 사용된 시공간 데이타 생성의 대표 도구인 GSTD(General Spatio-Temporal Data) 생성기[20]를 사용하여 생성하였다. GSTD 생성기 사용에 있어서, 이동객체는 랜덤하게 이동하고, 임의의 시간 간격으로 위치가 갱신되도록 설정하였다. 이동객체에 대한 위치는 첫번째 위치에서 시작하여 지속적으로 갱신되어 100,000번 변경되도록 하였다. 데이타 크기의 변화에 따른 오차의 변화를 측정하기 위하여, 데이타 크기는 1,000개, 10,000개, 100,000로 변화시키며 실험하였다. 실험에서는 갱신된 모든 위치 데이타를 실제위치로 사용하고, 메모리 저장 용량을 100개

로 제한한 후 추정위치를 계산하였다.

추정위치에 대한 오차를 계산하기 위해서는, 실제위치에서 5%의 샘플을 임의로 추출하여 사용하였다. 즉, 5% 샘플에 대해서 각 방법에 의한 추정위치를 구하고 이를 실제위치와 비교하여 오차를 계산하였다. 그리고, 각 방법에 대한 이들 오차의 평균 값을 최종 실험결과로 하였다. 또한, 시점에 따른 오차의 변화를 측정하기 위하여, 데이타 샘플 추출 시 세 가지 방법을 사용하였다. 첫째는 전체 시간 범위에서 일정한 간격으로 샘플을 추출하는 방법(**균등 샘플**이라 한다)이고, 둘째는 최근 50% 이내 구간에서 샘플을 추출하는 방법(**50% 샘플**이라 한다)이며, 셋째는 최근 20% 이내의 구간에서 샘플을 추출하는 방법(**20% 샘플**이라 한다)이다. 50% 샘플과 20% 샘플을 실험한 것은 오래된 시점보다 최근 시점에 대해 위치 요구가 많을 것으로 생각되기 때문이다.

본 논문에서 제안한 추정방법들은 메모리 제약이 없는 환경에서 직선 혹은 곡선 기반 방법을 사용하여 임의 시간의 위치를 추정하는 방법들에 비해서는 오차가 큰 것으로 나타났다. 실제 실험결과, 이러한 메모리 무제약 조건에서는 추정오차가 0.1% 미만으로 매우 작은 것으로 나타났다. 이는 메모리 무제약 환경인 경우 무제한의 메모리에 과거의 모든 위치 데이타를 저장함으로써 제안한 방법들보다 매우 정확하게 위치를 추정함을 의미한다. 그러나, 이러한 메모리 제약 조건은 실제 스트림 환경에서는 실용적이지 못한 가정이다. 따라서, 본 논문에서는 메모리 제약조건에서의 제안한 방법들만을 대상으로 위치 추정 정확도와 수행시간에 대한 실험을 수행한다.

실험에서는 직선기반으로 균등간격 추출방법, 기울기 기반 추출방법, 최근시점 강조 추출방법을, 곡선기반으로는 균등간격 추출방법과 최근시점 추출방법을 대상으로 오차를 측정하였다. 실험 수행을 위한 하드웨어 플랫폼은 Intel Pentium III 552 MHz CPU, 256MB RAM을 장착한 PC이며, 소프트웨어 플랫폼은 GNU/Linux Version 2.4.20-8 운영체제이다. 그리고, 모든 점진적 추출방법은 C언어로 구현하였다.

5.2 실험 결과

본 절에서는 제안한 점진적 추출 방법의 실험 결과를 설명한다. 먼저, 실험 1)에서는 추출방법에 따른 오차 변화의 실험 결과를 설명한다. 다음으로, 실험 2)에서는 제안한 각각의 추출 방법에 대하여, 추정 시점에 따른 오차 변화의 실험 결과를 설명한다. 마지막으로 실험 3)에서는 각 추출방법에 대한 수행시간의 실험 결과를 설명한다.

실험 1) 추출 방법에 따른 오차 변화의 실험 결과

그림 6은 균등 샘플에 대한 다섯 가지 추출 방법의 실험 결과를 나타낸다. 실험 결과를 보면, 다섯 가지 방법 모두 데이타의 개수가 많아질수록 오차가 커짐을 확인할 수 있다. 이는 발생한 데이타의 수는 1,000개, 10,000개, 100,000개로 증가한 반면에, 시스템에서 저장할 수 있는 데이타의 개수는 100개로 제한되었기 때문이다. 즉, 데이타의 개수가 많아질수록, 고정된 데이타 수를 사용하여 추정해야 하는 시간 구간이 커지게 되어, 자연스럽게 추정위치에 대한 오차가 증가하기 때문이다. 그러나, 저장하는 데이타의 비율이 0.1%(=100/100,000)로 가장 작은 경우에도, 추정위치의 오차가 1.34%~2.85%로 매우 적음을 알 수 있다. 이는 제안한 점진적 추출 방법이 비교적 정확하게 위치를 추출하고 추정함을 의미한다.

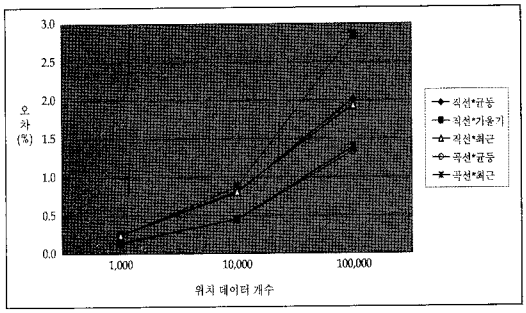
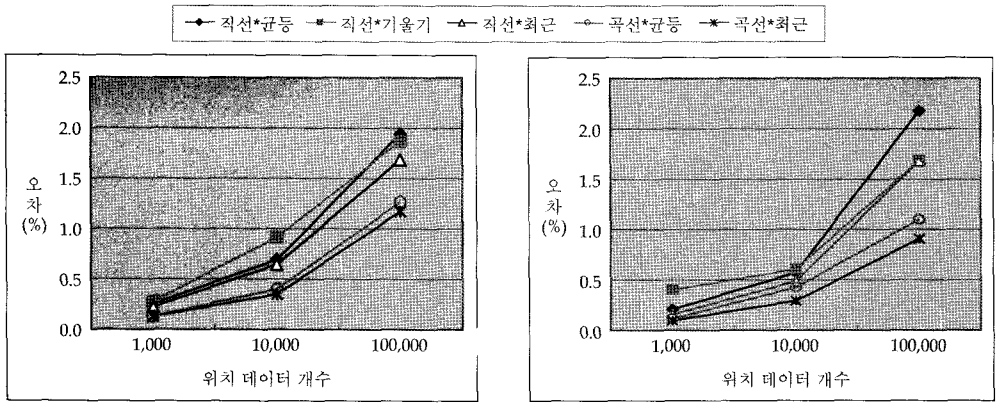


그림 6 추출 방법에 따른 오차변화(균등 샘플)

그림 6을 보면, 직선기반의 세 가지 방법에 비해 곡선기반의 두 가지 방법의 오차가 작음을 알 수 있다. 이는 직선기반 추출 방법들은 두 점만을 사용하여 위치를 보간하는 반면에, 곡선기반 방법들은 네 점을 사용하여 위치를 보간하기 때문이다. 또한, 그림의 결과를 보면, 직선기반 기울기 방법이 다른 방법에 비해 상대적으로 오차가 크게 나타남을 알 수 있다. 이는 직선기반 기울기 방법이 점진적 추출 시에 특정 구간 자체를 없애기 때문이다. 즉, 다른 방법들은 시점의 위치가 조금씩 변화하는 반면에, 직선기반 기울기 방법은 기울기가 작은 구간 전체를 없애기 때문에, 해당 구간에서의 오차가 크게 발생하는 것으로 분석되었다.

그림 7(a)와 (b)는 각각 50% 샘플과 20% 샘플에 대한 다섯 가지 방법의 실험 결과를 나타낸다. 그림을 보면, 데이타 개수가 증가에 따라 오차가 늘어나는 전체적인 경향은 그림 6과 유사함을 알 수 있다. 그러나, 최근 시점에 인접한 샘플일수록 최근시점 강조 추출방법이 균등간격 추출방법에 비해 우수한 성능(작은 오차)을 보임을 알 수 있다. 이는 균등간격 추출 방법은 전 구간에서 동일한 시간 간격으로 데이타를 추출하는 반면에, 최근시점 강조 추출방법은 현재시점에 가까울수록 더 좁



(a) 50% 샘플 (b) 20% 샘플

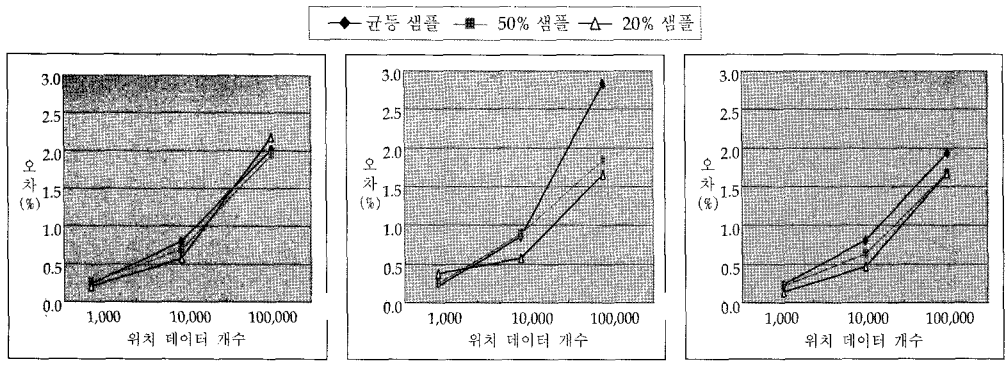
그림 7 추출 방법에 따른 오차변화(50% 및 20% 샘플)

은 시간 간격으로 데이타를 추출하기 때문이다. 또한, 그림 결과를 보면, 직선기반 기울기 추출방법의 오차 패턴은 다른 방법과는 다르게 나타났다. 이는 직선기반 기울기 방법에서 점진적 추출을 통해 변화되는(제거되는) 시간 간격이 일정하지 않기 때문이다.

실험 2) 추정 시점에 따른 오차 변화의 실험 결과

그림 8은 직선기반 추출 방법에 대한 세 가지 추정 시점 샘플의 실험 결과를 나타낸다. 그림 8(a)는 직선기반의 균등간격 추출 방법이고, 그림 8(b)는 기울기 기반 추출 방법을 나타내며, 그림 8(c)는 직선기반의 최근 시점 강조 추출 방법을 나타낸다. 그림 8(a)를 보면, 데이타의 개수가 1,000개와 10,000개일 때는 50% 샘플과 20% 샘플이 균등 샘플에 비해 오차가 작게 나타났다. 이는 최근 시점 정보가 많이 반영되는 점진적 추출의 특성에 따라 오래된 과거 시점 보다는 최근 시점에 대한 추정이 보다 정확하게 이뤄지기 때문이다. 그러나 데이타가 100,000개 일 때는 세 가지 샘플의 오차에 큰 차이가 없다. 그 이유는 데이타 개수가 많아짐에 따라 점진적 추출에 따른 최근 시점 정보 반영 효과가 약해지기 때문이다. 즉, 데이타 개수가 많아지면 시스템에 저장하는 하나의 위치정보가 대표하는 구간이 넓어져서 추출 시점에 따른 변화가 작게 되는 것이다.

그림 8(b)와 8(c)의 전체적인 경향은 그림 8(a)와 비슷하다. 그러나 그림 8(b)는 데이타가 1,000개일 경우 균등 샘플에서 오차가 가장 작지만, 데이타가 100,000개일 경우는 20% 샘플에서 오차가 가장 작다. 이는 기울기 기반 추출에서는 데이타 개수가 적을 때는 오차의 크기가 추출 시점의 영향을 받지 않으나, 데이타 개수가 많아질수록 오차의 크기가 추출 시점의 영향을 받기 때문으로 해석된다. 다음으로 그림 8(c)는 그림 8(a)와 (b)에 비해 샘플 추출시점이 현재에 가까울수록, 즉 20% 샘플에서 가장 오차가 작게 나타난다. 이는 최근 시점에 인접할수록 더 좁은 간격으로 데이타를 추출하는 최근 시점 강조 추출의 특징 때문이다.



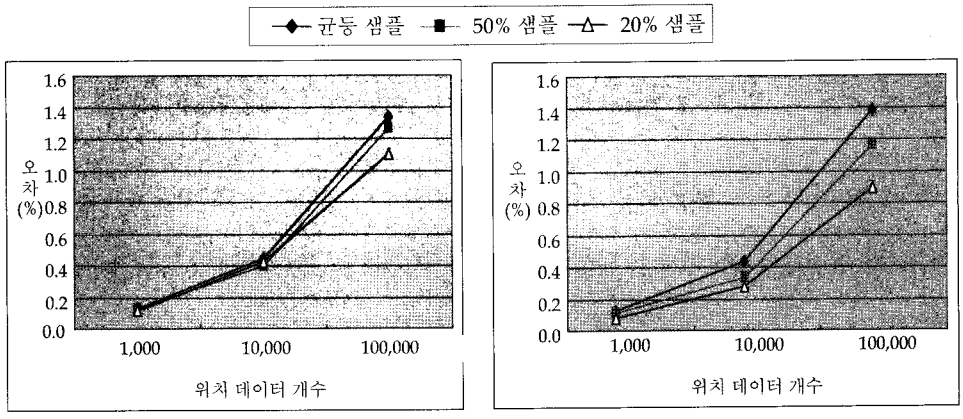
(a) 균등간격 추출 (b) 기울기 기반 추출 (c) 최근 시점 강조 추출

그림 8 추정 시점 따른 오차의 변화(직선기반 방법)

그림 9는 곡선기반 추출 방법에 대한 두 가지 샘플의 실험 결과를 나타낸다. 그림 9(a)는 곡선기반의 균등간격 추출 방법이고, 그림 9(b)는 곡선기반의 최근 시점 강조 추출 방법을 나타낸다. 그림 9(a)의 전체적인 경향은 그림 8(a)와 유사함을 알 수 있다. 그러나, 그림 9(a)의 곡선기반 방법이 그림 8(a)의 직선기반 방법에 비해 오차를 크게 줄였음을 알 수 있다. 또한, 그림 9(b)의 경향은 그림 8(c)의 경향과 유사하나, 그림 9(b)가 그림 8(c)에 비해 오차를 크게 줄였음을 알 수 있다. 앞서 설명한 바와 같이, 이는 곡선기반 방법에서 보다 많은 점을 사용하여 추정의 정확도를 높이기 때문이다. 그림 9(a)는 그림 8(a)에서와 같은 교차 구간이 발생하지 않고, 데이타의 개수가 커질수록 샘플 데이타의 추출 시점에 따른 오차 차이가 크게 발생한다. 이는 곡선기반 방법이 직선기반 방법에 비해, 추정시점의 영향을 더 받는 것으로 해석되기 때문이다. 또한 그림 9(a) 보다는 그림 9(b)에서 샘플 데이타의 추출 시점에 따른 오차 차이가 크게 발생한다. 이는 그림 9(b)에서는 현재에 가까운 시점일수록 과거 시점에 비해 더 좁은 시간 간격으로 점진적 위치 추출을 수행했기 때문이다.

실험 3) 각 추출 방법에 따른 수행 기간의 측정 결과

그림 10은 추출 방법에 따른 실제 수행 시간을 비교한 실험 결과이다. 실험에서는 각 방법에 대해 위치 갱신 횟수를 100,000번으로 하고 점진적 추출에 드는 실제 시간을 측정하였다. 실험 결과를 보면 직선기반 기울기 방법의 수행 시간이 가장 짧음을 알 수 있다. 이는 직선기반 기울기 방법이 새로운 위치정보 추출시 가장 간단한 방법으로 데이타를 추출하기 때문이다. 즉, 다른 방법은 모든 시점에 대한 위치를 다시 계산하는 반면에, 기울기 기반 방법은 기울기 변화가 가장 작은 구간에 있는 위치 데이타를 삭제함으로써 위치 추출이 완료되기 때문이다. 직선기반의 균등 방법과 최근 시점 강조 방법은 기울기를 기반한 방법보다는 시간이 오래 걸리나, 곡선기반 방법들 보다는 짧게 걸리는 것으로 나타났다. 기울기 기반 방법보다 시간이 많이 걸리는 이유는 두 방법 모두 모든 시점에 대해서 위치 계산을 다시 해야 하기 때문이다. 반면에, 곡선기반 방법보다 적은 시간이 걸리는 이유는 계산이 간단하기 때문이다. 즉, 직



(a) 균등간격 추출 (b) 최근 시점 강조 추출

그림 9 추정 시점에 따른 오차 변화(곡선기반 방법)

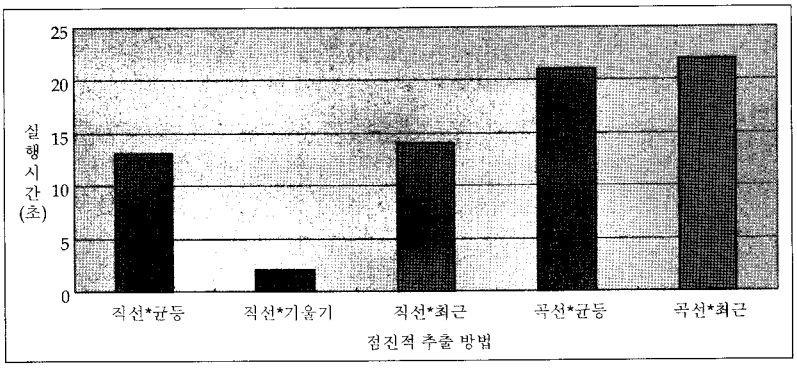


그림 10 점진적 추출 방법들의 성능비교

선기반 방법들은 네 점을 고려하는 곡선기반 방법들과 달리 두 점만 고려하여 위치 추출을 수행하기 때문이다.

6. 결 론

이동객체의 과거 위치를 저장하고 추정하는 다양한 연구가 진행되었다. 그러나 이들 기존연구는 위치정보가 끝임없이 발생하는 스트림 환경에는 적용하기 어렵다. 이는 기존 방법들이 이동객체의 위치정보를 메모리 제약없이 저장하고, 이를 기반으로 위치를 추정하였기 때문이다. 즉, 스트림 환경에서는 끊임없이 발생하는 위치정보를 모두 메모리에 저장할 수 없기 때문이다. 따라서 본 논문에서는 스트림 형태로 발생하는 위치정보를 제한된 메모리에 점진적으로 저장하고, 이를 기반으로 과거 위치를 추정하는 방법을 제안하였다.

본 논문의 공헌은 다음과 같이 요약할 수 있다. 첫째, 제한된 메모리 양으로 스트림 형태의 과거 위치 이력을 저장하기 위한 위치정보의 점진적 추출 개념을 제안하였다. 둘째, 점진적 추출 개념에 기반하여 위치정보를 저장 및 추정하는 전체적인 동작 프레임워크를 제안하였다. 셋째, 이동 객체가 시스템에 보고하는 위치와 시스템에 의해 추정된 위치를 구분하여 실제위치와 추정위치로 정의하고, 추정위치를 계산하는 방법으로 다항식을 사용한 직선기반과 곡선기반의 두 가지 방법을 제안하였다. 넷째, 과거 위치를 점진적으로 추출하는 방법으로 균등간격 데이타 추출, 기울기 기반 데이타 추출, 그리고 최근 시점 강조 데이타 추출의 세 가지 방법을 제안하였다. 다섯째, 다양한 실험을 통해 제안한 점진적 추출 방법이 스트림 환경에서 위치 정보를 저장 및 추정하는 우수한 방법임을 확인하였다. 그러나 본 논문에서 제안한 방법은 다음과 같은 제한점이 있다. 첫째, 저장할 수 있는 데이타가 제한되어 있기 때문에 시간이 흐를수록 오차의 크기가 커진다. 둘째, 새로운 위치 데이타가 보고될 때마다 저장된 데이타를 전체적으로 변경되어야하기 때문에 갱신 비용이 많이 발생한다. 이러한 제한점을 고려하여 일정시간 단위로 데이타를 저장하는 방법이나, 일정 주기로 갱신을 수행하는 방법을 향후 연구로 진행할 예정이다.

이 같은 결과로 볼 때, 제안한 방법은 스트림 환경에서 위치정보를 저장하고 추정하는 우수한 방법이라 사료된다. 특히, 라그랑제 다항식을 사용한 곡선기반 추정 방법과 위치정보의 점진적 추출 방법은 기존의 연구와는 차별되는 우수한 연구결과이다.

참 고 문 헌

[1] Hu, H., Xu, J., and Lee, D. L., "A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects," In *Proc. Int'l Conf. on Management of Data* ACM SIGMOD, Baltimore, Md, pp. 479-490, June 2005.

[2] Wang, G., Cao, G., Porta, T., and Zhang, W., "Sensor Relocation in Mobile Sensor Networks," *IEEE, INFOCOM*, Vol. 4, pp. 2302-2312, Mar. 2004.

[3] Brugnoli, M. C., Hamard, J., and Rukzio, E., "User Expectations for Simple Mobile Ubiquitous Computing Environments," In *Proc. the 2nd workshop on Mobile Commerce and Services*, Munich, Germany, pp. 2-10, July 2005.

[4] Nievergelt, J. and Hinterberger, H., "The Grid File: An Adaptable, Symmetric Multikey File Structure," *ACM Transactions on Database Systems*, Vol. 9, No. 1, pp. 38-71, Mar. 1984.

[5] Tzouramanis, T., Vassilakopoulos, M., and Manolopoulos, Y., "Overlapping Linear Quadtrees: A Spatio-Temporal Access Method," In *Proc. the 6th Int'l Symp. on Advances in Geographic Information Systems*, Washington D.C., pp. 1-7, Nov. 1998.

[6] Guttman, A., "R-trees: A Dynamic Index Structure for Spatial Searching," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Boston, MA, pp. 47-57, June 1984.

[7] Nascimento, M. and Silva, J., "Towards Historical R-trees," In *Proc. of ACM Symp. on Applied Computing*, ACM SAC, Atlanta, Georgia, pp. 235-240, Feb. 1998.

[8] Pfoser, D., Jensen, C., and Theodoridis, Y., "Novel Approaches to the indexing of Moving Object Trajectories," In *Proc. Int'l Conf. on Very Large Data Bases*(VLDB), Cairo, Egypt, pp. 395-406, Sept. 2000.

[9] Cai, Y. and Ng, R., "Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Paris, France, pp. 599-610, June, 2004.

[10] Saltenis, S., Jensen, C. S., Leutenegger, S. T., and Lopez, M. A., "Indexing the Positions of Continuously Moving Objects," In *Proc. Int'l. Conf. on Management of Data*, ACM SIGMOD, Dallas, TX, pp. 331-342, June 2000.

[11] Yu, B.-G., Kim, S.-H., Bailey, T., and Gamboa, R., "Curve-Based Representation of Moving Object Trajectories," In *Proc. Int'l Conf. on Database Engineering and Applications Symposium*, IEEE, Dijon, France, pp. 419-425, Apr. 2004.

[12] Tao, Y., Faloutsos, C., Papadias, D., and Liu, B., "Prediction and Indexing of Moving Objects with Unknown Motion Patterns," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Paris, France, pp. 611-622, June 2004.

[13] Faloutsos, C., Ranganathan, M., and Manolopulos, Y., "Fast Subsequence Matching in Time-Series

Database," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Minneapolis, MN, pp. 419-429, May 1994.
[14] Chan, K.-P. and Fu, W.-C., "Efficient Times Series Matching by Wavelets," In *Int'l Conf. on Data Engineering*, IEEE, Sydney, Australia, pp. 126-133, Mar. 1999.
[15] Keogh, J. E., Chakrabarti, K., Mehrotra, S., and Pazzani, M., "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases," In *Int'l Conf. on Management of Data*, ACM SIGMOD, Santa Barbara, CA, pp. 151-162, May 2001.
[16] Babcock, B. et al., "Models and Issues in Data Stream Systems," In *Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symp. On Principles of Databases Systems (PODS)*, Madison, Wis, pp. 1-16, June 2002.
[17] Lim, H.-S., Lee, J.-G., Lee, M.-J., Whang, K.-Y., and Song, I.-Y., "Continuous Query Processing in Data Streams Using Duality of Data and Queries," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Chicago, Ill, pp. 313-324, June 2006.
[18] Keogh, E. J. et al., "LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures," In *Proc. Int'l Conf. on Very Large Data Bases (VLDB)*, Seoul, Korea, pp. 882-893, Sept. 2006.
[19] Rao, S. S., Applied Numerical Methods for Engineers and Scientists, Prentice Hall, 2002.
[20] Theodoridis,Y., Silva, J. R. O., and Nacimento, M. A., "On the Generation of Spatiotemporal Datasets," In *Proc. Symp. on Advances in Spatial Databases*, Hong Kong, China, pp. 147-164, July 1999.
[21] Palpanas, T., Vlachos, M., Keogh, E., Gunopulos D., and Truppel, W., "Online Amnesic Approximation of Streaming Time Series," In *Proc. Int'l Conf. on Data Engineering*, IEEE, Boston, MA, pp. 338-349, Mar. 2004.
[22] Keogh, J. E., and Pazzani, J. M., "An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering and Relevance Feedback," In *Proc. Int'l Conf. on Knowledge Discovery and Data Mining*, New York, NY, pp. 239-243, Aug. 1998.

이 원 철
2000년 2월 강원대학교 행정학과 학사 2002년 8월 강원대학교 컴퓨터과학과 석사. 2007년 2월~현재 강원대학교 컴퓨터과학과 박사과정. 관심분야는 Moving Objects, Similarity Search on XML, Data Mining, Multimedia Systems

문 양 세
정보과학회논문지 : 데이타베이스
제 34 권 제 1 호 참조

이 상 민
1976년 2월 경북대학교 전자공학과 학사 1979년 8월 경북대학교 전자공학과 석사(전산공학 전공). 1993년 8월 경북대학교 전자공학과 박사(전산공학 전공). 1989년~1995년 강원대학교 전산소장. 1993년~1995년 교육연구전산망 추진위원. 1991년~2000년 전국대학정보전산기관협의회 상임이사. 1984년 2월~현재 강원대학교 컴퓨터학부 컴퓨터과학전공 정교수 관심분야는 Computer Architecture, Digital System Design, Design Automation, Multimedia Systems