The Journal of GIS Association of Korea, Vol. 16, No. 1, pp. 113-127, April 2008

모바일 소프트웨어를 위한 효율적인 공간 인덱스[1)]

오 병 우*

Efficient Spatial Index for Mobile Software

Byoung-Woo Oh*

요 약

최근 모바일 환경의 급속한 발달로 이동 중인 사용자의 위치에 기반한 다양한 서비스가 가능하게 되면서 모바일 기기에서 지도를 처리하는 모바일 소프트웨어의 개발이 증대되고 있다. 본 논문은 지도를 처리하는 모바일 소프트웨어에서 성능을 결정짓는 가장 중요한 요소인 공간 인덱스를 효율적으로 구성하는 새로운 방법을 제시한다. 본 논문에서 제시하는 AR*-tree는 기존의 R*-tree를 변형하여 2차원 공간 데이터의 x축 및 y축에 면적 (a) 축을 추가하여 3차원 데이터를 저장한다. 그리고, 검색 시에도 면적 축에 대한 조건을 추가함으로써 작은 화면을 갖는 모바일 기기에서의 지도 가독성을 증대시키고 시간 효율성도 동시에 향상시킨다.

주요어 : 모바일 소프트웨어, 공간 인덱스, R*-tree, AR*-tree

ABSTRACT : This paper proposes an efficient spatial index, named AR*-tree(Area R*-tree) which is a variant of the R*-tree, for mobile software. A MBR(Minimum Bounding Rectangle) structure of the AR*-tree has additional min and max values of area axis as well as x and y axes. The value of area axis is used to determine the significance of a spatial data. If area of a spatial data is large, then it is significant when drawing a map. To reduce complexity of a map on a small screen of mobile device, only significant spatial data can be found by the AR*-tree. The result of a series of tests indicates that the AR*-tree provides a method for control of readability of a map and guarantees an efficient performance at the same time.

Keywords : mobile software, spatial index, R*-tree, AR*-tree

*국립금오공과대학교 컴퓨터공학부 컴퓨터공학전공 교수
1) 본 연구는 금오공과대학교 학술연구비에 의하여 연구된 논문임.

1. 서 론

최근 컴퓨터의 발달로 휴대용 모바일 장치가 보편화 되었고 모바일 환경을 위한 연구가 활발히 진행되고 있다(Gal and Toledo, 2005; Liu et al., 2005). 모바일 장치에서 동작하는 모바일 소프트웨어는 공간상에서 변화하는 사용자의 위치를 사용하기 위하여 대용량이고 복잡한 공간 데이터를 효율적으로 처리하여야 한다.

그러나, 일반적으로 모바일 장치는 낮은 해상도의 작은 화면을 가지고 있어서 지도 출력하기에 부족하며 성능이 낮은 CPU를 탑재하고 있어서 신속하게 공간 데이터를 처리하기 힘들다. 그러므로 모바일 장치에서 공간 데이터를 처리하기 위해서는 특성에 맞는 처리 방법이 필요하다.

본 논문에서는 모바일 장치의 특성을 반영하여 낮은 해상도의 작은 화면에서 지도 가독성을 높이면서도, 성능이 낮은 CPU 환경을 고려하여 효율적으로 공간 데이터를 처리할 수 있는 공간 인덱스를 제안한다. 매우 많은 수의 크기가 작은 공간 객체가 각각 화면상에서 형태를 인식할 수 없는 한 개의 픽셀 또는 매우 작은 영역으로 출력된다면 화면의 복잡도는 증가하고 지도의 가독성은 낮아진다. 이러한 문제를 해결하기 위하여 본 논문에서 제안하는 AR*-tree(Area R*-tree)는 화면상에 너무 작게 표시될 공간 객체를 질의 검색에서 미리 제외할 수 있도록 해준다.

AR*-tree는 기존 2차원(x, y)축에 면적축을 추가하여 3차원 인덱스를 구성함으로써 공간 객체의 면적에 대한 질의도 지원한다. 이를 통해 면적이 작은 공간 객체를 필터링할 수 있어서 지도 가독성을 높이고, 적은 수의 공간 객체를 처리하게 되므로 처리 시간이 줄어들어 시간 효율성도 동시에 향상된다. R*-tree로부터 수정하여야 할 부분도 적어서 쉽게 개발할 수 있다. 특히, 3차원 이상의 다차원 데이터 처리를 제공하는 R*-tree는 수정 없이 면적 값만을 사용하여 적용 가능하다.

공간 객체의 면적으로는 계산하기 복잡한 실제 면적이 아닌 MBR(Minimum Bounding Rectangle)의 면적을 사용하여 계산이 용이하고, 면(polygon) 객체뿐만 아니라 면적을 가지고 있지 않은 선(line) 객체도 처리할 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로서 모바일 소프트웨어의 특징에 대한 분석과 R*-tree에 대해 간략히 살펴본다. 3장에서는 본 논문에서 제안하는 공간 인덱스에 대해 자세히 설명하고, 4장에서는 지도 가독성과 시간 효율성에 대한 실험 및 결과를 기술한다. 마지막으로 5장에서는 결론 및 향후 연구 방향에 대해 언급한다.

2. 관련 연구

2.1 모바일 소프트웨어의 특징

컴퓨터의 발달로 모바일 장치도 가격이 낮아지면서 모바일 환경이 급격히 발전하고 있다. 텔레매틱스, 위치 기반 서비스 등과 같은 모바일 소프트웨어에서는 사용

자의 위치를 지도상에 표현하기 위하여 지도를 저장하고 접근하는 처리가 필요하다(오병우, 2006). 그리고, 최근에는 저장 기술의 발달에 따라 휴대용 초소형 메모리가 대용량화되면서 모바일 소프트웨어에서 필요한 데이터를 모두 저장하여 가지고 다닐 수 있어서 대용량인 공간 데이터도 저장이 가능하다.

모바일 소프트웨어는 다음과 같은 특징을 가지므로 공간 데이터를 모바일 장치에 저장하고 사용하는 응용 프로그램은 이러한 특성을 반영하여 개발하여야 한다.

- 작은 화면: 모바일 장치는 일반적으로 해상도가 낮고 작은 화면을 가지고 있으므로 복잡한 지도의 가독성을 위한 처리 필요
- 낮은 CPU 성능: 모바일 장치는 일반적으로 일반 PC보다 CPU의 성능이 낮으므로 더욱 효율적인 처리 필요

본 논문에서는 이러한 모바일 소프트웨어의 특성을 반영하여 공간 데이터의 면적을 활용하여 지도 가독성을 높이면서 동시에 시간 효율성도 우수한 공간 데이터 접근 방법을 제안한다.

2.2 R*-tree

R*-tree(Beckmann et al., 1990)는 효율성을 인정받아 R-tree와 함께 현재 거의 모든 상용 GIS에서 사용되고 있고, 다양한 확장 방법에 대한 연구도 활발히 진행되고 있다(Biveinis et al., 2007; Xia and Zhang 2005). 본 논문에서는 R*-tree 변형 방법 중 최초로 면적을 처리할 수 있도록 확장한다.

R*-tree는 R-tree(Guttman, 1984)의 분할 방식을 영역의 포함, 겹침 관계를 최소화하도록 수정하고, 강제 재삽입 방법을 사용한다. R-tree 및 R*-tree는 다차원 인덱스로서 현재 응용 프로그램에서는 주로 2차원의 공간 데이터를 위하여 사용된다. 본 논문에서는 2차원 공간 데이터를 처리하는 R*-tree를 확장하여 3번째 차원인 면적축을 고려하도록 확장한다.

3. AR*-tree

3.1 AR*-tree의 특징

3.1.1 MBR 면적을 공간 데이터의 중요도로 활용

공간 데이터를 화면에 출력함에 있어서 일반적으로는 자세한 정보를 제공하는 것이 사용자의 욕구를 충족시킬 수 있다. 그러나, 모바일 환경에서 사용하는 장치는 주로 작은 화면을 가지고 있으므로 너무 많은 공간 데이터를 출력하면 공간 데이터를 구분할 수 없게 되어 가독성이 오히려 떨어지게 된다.

이를 해결하기 위하여 일반적으로 객체 또는 레이어별 중요도인 LOD(Level of Detail) 데이터를 추가로 입력하여 사용한다(Han and Bertolotto, 2004). 그러나, 추가적으로 LOD 데이터를 입력하는 과정은 수작업으로 지정하여야 하므로 공간 데이터 구축 비용이 증가된다. 본 논문에서는

공간 데이터 자체의 특성인 면적을 이용하여 자동으로 처리하면서도 가독성을 증가시킬 수 있는 방법을 지원한다.

지도를 구성하는 공간 객체들은 크기가 큰 객체와 작은 객체가 혼합되어 있다. 선 객체는 길이를 가지고 있고, 면 객체는 면적을 가지고 있다. 점 객체는 면적이 0이며 주로 POI(Point-of-Interest)로 사용되므로 본 논문에서는 고려하지 않는다.

본 논문에서는 지도를 출력할 때 공간 객체의 중요도를 MBR의 면적으로 평가한다. MBR의 면적이 큰 공간 객체일수록 지도에 출력할 때 화면의 넓은 부분을 차지하기 때문이다. 객체 자체의 면적을 사용하려면 복잡한 계산이 요구될 뿐만 아니라 선 객체의 경우는 면적이 없고 길이만을 가지므로 넓은 부분을 차지하는 중요한 객체이면서도 평가 절하될 수 있으므로 객체의 면적보다는 MBR의 면적이 중요도를 산출하는데 있어서 적합하다.

3.1.2 MBR에 면적 축 추가

본 논문에서 제안하는 AR*-tree는 공간 객체의 중요도를 MBR의 면적으로 계산하고 기존의 R*-tree에 면적 축을 추가하여 처리한다. R*-tree는 R-tree의 엔트리 구조를 사용하는데 트리를 구성하는 단말 노드는(I, *tuple-identifier*)이고, 중간 노드는 (I, *child-pointer*)의 구조이다. 이때 I는 다음과 같이 MBR을 의미한다.

$I = (I_0, I_1, \ldots I_{n-1})$

n은 차원의 개수, I_i는 닫힌 구간[a, b]

일반적인 2차원 공간 좌표를 다루는 경우는 다음과 같이 표현할 수 있다.

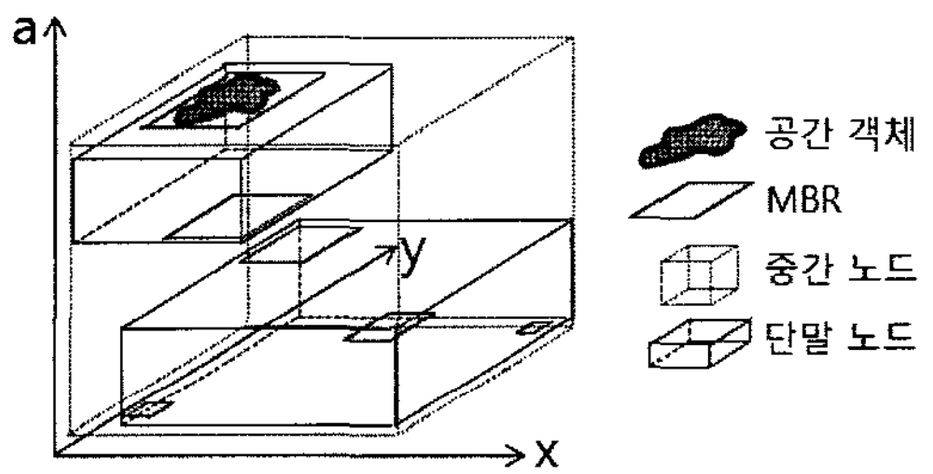
$I = (I_0, I_1) = ([x1, x2], [y1, y2]), n = 2$

본 논문에서는 기존의 MBR에 면적 축을 추가하여 다음과 같은 구조를 사용한다.

$I = (I_0, I_1, I_2)$
$= ([x1, x2], [y1, y2], [a1, a2]), n=3$

a1은 최소 면적이고 a2는 최대 면적이다. 공간 객체의 MBR의 면적은 범위가 아니라 단일 값($\Delta x * \Delta y$)이므로 단말 노드의 엔트리에서는 a1과 a2가 동일하다.

[그림 1]은 면적 축을 추가한 AR*-tree의 노드 구성 예제를 보여준다. MBR의 면적이 큰 객체는 a축의 상위 부분에 위치하고 면적이 작은 객체는 a축의 하위 부분에 위치한다.



[그림 1] AR*-tree의 노드 구성 예제

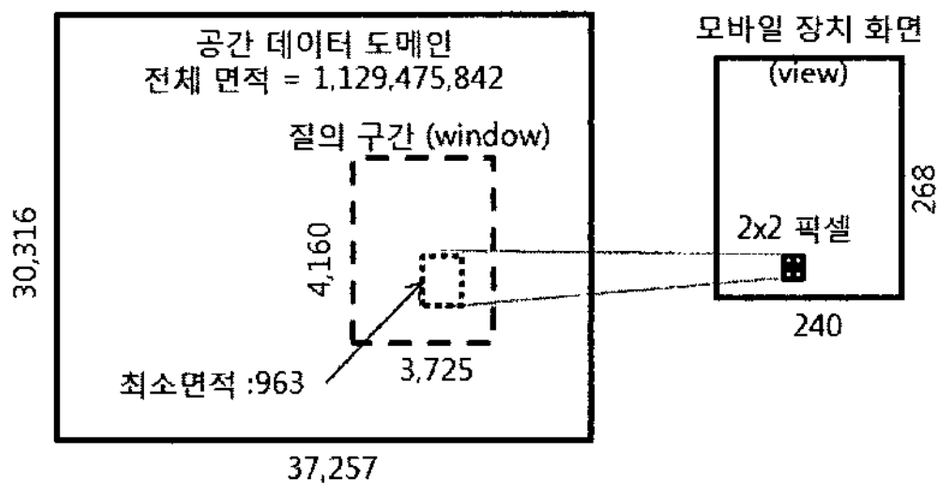
3.1.3 검색을 위한 면적 파라미터 활용

공간 데이터 전체 도메인으로부터 질의 구간에 대한 공간 객체들을 검색하여 화면에 출력할 때 작은 공간 객체는 화면상

에서 1픽셀의 점이나 또는 2×2 픽셀 등과 같이 매우 작게 표시될 수 있다. 이러한 객체는 본래의 기하학적인 형태는 표현하지 못하면서 지도의 복잡성을 증가시키므로 검색에서 제외하는 것이 지도 가독성 증대를 위하여 바람직하다.

본 논문에서 제안하는 AR*-tree는 MBR의 면적을 추가하여 R*-tree를 구성하고 있으므로, 질의 구간을 지정할 때에도 기존의 *I = ([x1, x2], [y1, y2])*에 면적 축에 대한 조건을 추가하여 *I = ([x1, x2], [y1, y2], [a1, a2])*로 지정한다. 즉, 최소 면적과 최대 면적을 지정하여 공간 객체의 중요도에 따른 검색을 지원한다. 최대 면적은 공간 데이터 전체 도메인의 면적으로 지정하고, 최소 면적은 화면 출력에서 제외할 최소 픽셀 면적에 해당하는 공간 도메인의 면적을 계산하여 질의의 파라미터로 지정한다.

예를 들어, [그림 2]와 같이 PDA에서 가장 많이 사용되는 화면 크기인 240×320(QVGA)을 갖는 모바일 장치에서 10% 질의 구간에 대해 화면의 2×2 픽셀을 최소 면적으로 지정하는 질의는 다음과 같이 처리된다. 전체 도메인은 너비 37,257 높이 30,316으로 총 면적은 1,129,475,842이다. 10% 질의 구간은 너비의 1/10인 3,726과 응용 프로그램 화면의 크기인 240×268의 비율을 적용(240 : 268 = 3,726 : height)하여 높이 4,160을 갖는다. 화면의 너비 2 픽셀은 축소 비율을 적용(240 : 3,726 = 2 : min_width)하여 원본 데이터 도메인의 31.05에 해당하고, 같은 방법으로 화면의 높이 2픽셀은 축소 비율을 적용(268 : 4,160 = 2 : min_height)하면 원본 데이터 도메인에서 31.04에 해당한다. 그러므로, 질의에 사용할 면적의 파라미터는 최소 면적 963 (= 31.05×31.04), 최대 면적 1,129,475,842이 된다.



[그림 2] 화면의 픽셀에 대한 공간 데이터 도메인에서의 최소 면적 예제

3.2 AR*-tree의 구현

앞서 설명한 AR*-tree를 구현하기 위해서는 R*-tree의 구현으로부터 다음과 같이 세 부분만을 간단히 수정하면 된다.

3.2.1 MBR 처리

- 멤버 변수 추가: 노드의 엔트리 구조에서 MBR 구조체 또는 클래스에 면적 축의 최소값 및 최대값을 저장하기 위한 a1과 a2 멤버 변수 추가
- Overlap(): MBR간에 중첩(overlap) 되는지 여부를 검사하는 함수에서 x, y축뿐만 아니라 a축(면적 축)에 대해서도 같은 요령으로 두 MBR의 최소값들 중에서 큰 값이 MBR의 최대값들 중에서 작은 값보다 작은지를 검사하는 코드 추가
- Intersect(): MBR간의 교집합을 구하는

함수에서 x, y축뿐만 아니라 a축(면적 축)에 대해서도 같은 요령으로 두 MBR의 최소값들 중에서 큰 값을 최소값으로 지정하고, MBR의 최대값들 중에서 작은 값을 최대값으로 지정하는 코드 추가

- Union(): MBR간의 합집합을 구하는 함수에서 x, y축뿐만 아니라 a축에 대해서도 같은 요령으로 두 MBR의 최소값들 중에서 작은 값을 최소값으로 지정하고, 최대값들 중에서 큰 값을 최대값으로 지정하는 코드 추가
- Margin(): MBR의 둘레 길이(margin)를 계산하는 함수는 기존에는 *2 * ((x2 - x1) + (y2 - y1))*으로 계산하였으나 3차원으로 확장되어 모든 에지(edge) 길이의 합인 *4 * ((x2 - x1) + (y2 - y1) + (a2 - a1))*으로 계산하도록 수정
- Area(): MBR의 면적을 구하기 위한 함수는 3차원으로 확장되어 x, y, a축 모두에 대한 부피를 계산하도록 수정. 만약 단말 노드(leaf node)에 존재하는 엔트리의 MBR이어서 a1과 a2가 같다면 기존과 같이 x, y축의 면적만을 사용

3.2.2 삽입 함수 호출을 위한 MBR 구조체 값 지정

삽입 알고리즘은 R*-tree와 동일하므로 수정할 필요가 없다. 다만, 삽입을 위한 함수를 호출하기 위해서 MBR 구조체인 *I = ([x1, x2], [y1, y2], [a1, a2])*의 a1과 a2의 값을 지정할 때는 다음 수식과 같이 면적을 계산하여 동일하게 지정한다.

$$a1 = a2 = (x2 - x1) * (y2 - y1)$$

3.2.3 검색 함수 호출을 위한 MBR 구조체 값 지정

검색 알고리즘은 앞서 이미 수정한 Ovelap() 함수를 사용하므로 기존의 R*-tree와 동일하다. 질의 구간에 대한 MBR 구조체의 면적 축 최소값인 a1을 위해서는 먼저 질의 구간(window) 대비 화면(view) 비율인 스케일(scale)을 계산하고, 화면에서 몇 픽셀(*view_min_pixel*)보다 작은 공간 객체를 제외할 것인지를 결정하여 다음 수식과 같이 최소 면적(*window_min_area*)을 계산한 값을 a1에 지정한다.

*window_min_area = view_min_pixel * view_min_pixel / scale / scale*

면적 축 최대값인 a2는 공간 데이터 전체 도메인의 면적으로 지정한다. 필요에 따라서는 최소값 및 최대값을 모두 임의의 값으로 지정할 수도 있다.

4. 성능 평가

4.1 실험 환경 및 방법

본 논문에서 제안한 AR*-tree의 성능을 평가하기 위해서 두 가지 방법의 실험을 수행하였다. 첫 번째는 작은 화면을 가지고 있는 모바일 장치에서 지도 가독성을 높이기 위한 실험으로서, 공간 데이터베이스에 저장된 모든 데이터를 출력하면

너무 복잡하게 되어 가독성이 떨어지는 문제를 본 논문에서 제안한 방법으로 해결할 수 있는지에 대한 실험이다. 두 번째는 일반적으로 CPU의 성능이 낮아서 성능의 효율성이 요구되는 모바일 장치에서 본 논문에서 제안한 방법이 얼마나 효율적인지를 검증하기 위한 실험이다.

실험은 <표 1>과 같이 3가지 종류의 모바일 장치에서 수행하였다. 메모리를 사용하는 2 종류의 PDA와 하드디스크를 사용하는 PMP를 사용하였다. 가장 많이 사용되는 QVGA 및 VGA 해상도와 고해상도(800×480)를 실험 대상으로 하였다.

공간 데이터로는 서울시의 행정경계, 수계, 도로, 건물 레이어를 통합하여 18.0MB의 WKB(Well-Known Binary by Open Geospatial Consortium) 데이터를 생성하고 실험에 사용하였다. 원본 데이터의 KATECH 좌표계(TM128 투영 좌표)를 정수형(integer) 타입으로 변환하지 않고 원본 실수형(double) 타입 데이터를 그대로 사용하였다. WKB 및 AR*-tree의 구축은 모바일 장치가 아닌 성능이 우수한 PC에서 수행되며 모바일 장치에서는 생성된 데이터 및 인덱스를 검색에만 사용하므로 생성에 소요되는 시간 측정은 실험에서 제외하였다.

실험 방법은 <표 2>와 같이 각각의 모바일 장치에서 질의 영역을 전체 너비 기준으로 10%, 25% 50%, 75%, 100%로 변경하면서 랜덤으로 100개의 질의 구간을 생성하여 R*-tree와 본 논문에서 제안한 AR*-tree의 평균 응답 시간을 측정하였다. 이때 랜덤에 의한 R*-tree 및 AR*-tree의 응답 시간 차이를 없애기 위하여 랜덤 발생 시드(seed number)를 동일하게 지정하여 두 트리에 대한 질의가 동일하도록 하였

<표 1> 실험 환경

제품 명칭	CPU	운영체제	화면 해상도	저장매체
iPAQ rx3715	삼성 S3C2440 400MHz	Windows Mobile 2003 SE	240×320 (QVGA)	64MB RAM
iPAQ hx4700	Intel PXA270 624MHz	Windows Mobile 2003 SE	480×640 (VGA)	64MB RAM
iStation NetForce	AMD Alchemy MIPS32-Au1200 500MHz	WinCE.NET 5.0	800×480 (고해상도)	30GB HDD

<표 2> 실험 방법

모바일 장치	화면 해상도	공간 인덱스	최소 면적 범위 (화면 픽셀)	질의 영역 크기	질의 회수	시간 측정
iPAQ rx3715	240×320 QVGA	R*-tree	-	원본 데이터 너비의 10%, 25%, 50%, 75%, 100% (높이는 장치의 화면 해상도에 따라 계산)	각각의 경우에 대해 100번씩 질의 수행	인덱스 검색 시간, 데이터 접근 시간, 화면 출력 시간
		AR*-tree	2×2, 5×5, 10×10			
iPAQ hx4700	480×640 VGA	R*-tree	-			
		AR*-tree	2×2, 5×5, 10×10			
iStation NetForce	800×480	R*-tree	-			
		AR*-tree	2×2, 5×5, 10×10			

다. 그리고, AR*-tree는 검색을 위해서 면적의 파라미터를 추가로 가지므로, 공간 데이터가 출력될 모바일 장치의 화면을 기준으로 2×2, 5×5, 10×10 픽셀에 해당하는 원본 공간 데이터에서의 면적을 계산하여 검색 조건의 최소 면적 파라미터로 사용하였다.

실험에서 수행한 총 질의 회수는 3(모바일 장치 종류) * 5(질의 영역 크기 10%, 25%, 50%, 75%, 100%) * 4(R-tree 및 AR*-tree의 화면 픽셀에 따른 최소 면적 범위 2×2, 5×5, 10×10) * 100(랜덤에 의한 질의 구간 생성) = 6,000번이다.

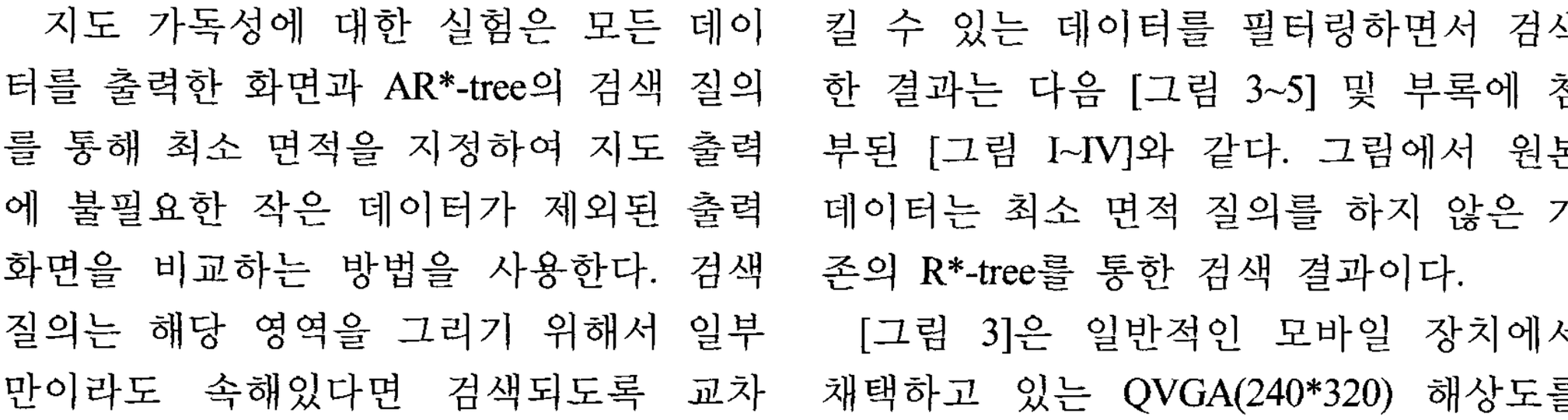
4.2 지도 가독성

지도 가독성에 대한 실험은 모든 데이터를 출력한 화면과 AR*-tree의 검색 질의를 통해 최소 면적을 지정하여 지도 출력에 불필요한 작은 데이터가 제외된 출력 화면을 비교하는 방법을 사용한다. 검색 질의는 해당 영역을 그리기 위해서 일부만이라도 속해있다면 검색되도록 교차(intersect) 연산을 사용한다.

지도 가독성에 대해서는 정량적인 평가가 불가능하므로 질의 결과 화면을 수록한다. 지면 관계상 모든 질의 구간에 대한 화면을 수록하기가 어려우므로 100번의 질의 결과 화면 중에서 한 개의 동일한 질의에 대해 R*-tree를 통해 검색한 결과와 AR*-tree의 최소 면적을 변경하면서 검색한 결과 데이터를 출력한 화면을 비교한다. 10% 질의 구간에 대한 결과 화면은 3가지 종류의 모바일 장치에 대해 모두 수록하고, 나머지 25%, 50%, 75%, 100%의 질의 구간에 대한 결과 화면은 지면 관계상 각각 한 종류의 모바일 장치에 대한 결과를 부록에 수록한다.

지도를 출력하기 위한 공간 데이터의 검색에서 R*-tree를 통해 검색한 결과와 본 논문에서 제안한 AR*-tree를 통해 화면상에서 작게 표시되어 가독성을 저하 시킬 수 있는 데이터를 필터링하면서 검색한 결과는 다음 [그림 3~5] 및 부록에 첨부된 [그림 I~IV]와 같다. 그림에서 원본 데이터는 최소 면적 질의를 하지 않은 기존의 R*-tree를 통한 검색 결과이다.

[그림 3]은 일반적인 모바일 장치에서 채택하고 있는 QVGA(240*320) 해상도를 갖는 rx3715에서 10% 질의 결과를 출력한 화면을 보여준다. 화면 영역에서 실제 응용 프로그램에서 사용할 수 있는 영역은



(a) 원본 데이터 (b) 2x2 픽셀 (c) 5x5 픽셀 (d) 10x10 픽셀

[그림 3] rx3715(240x268)에서의 너비 10% 질의 결과 화면 비교

240×268이다. 그림에서 보는 바와 같이 R*-tree를 통해 검색한 결과를 보여주는 "(a) 원본 데이터"는 작은 화면에 많은 데이터가 출력되어 복잡한 구역에서 지도의 가독성이 매우 낮다. "(b) 2×2 픽셀"은 공간 데이터를 검색할 때 임의의 객체가 질의 영역에는 속하지만 해당 객체의 MBR이 화면상의 픽셀 면적 4(=2×2)보다 작다면 결과에서 제외하는 방법을 통해 얻어진 출력 결과이다. 이와 동일하게 (c) 및 (d)는 각각 화면상의 면적이 25, 100보다 작은 객체를 제외한 결과이다.

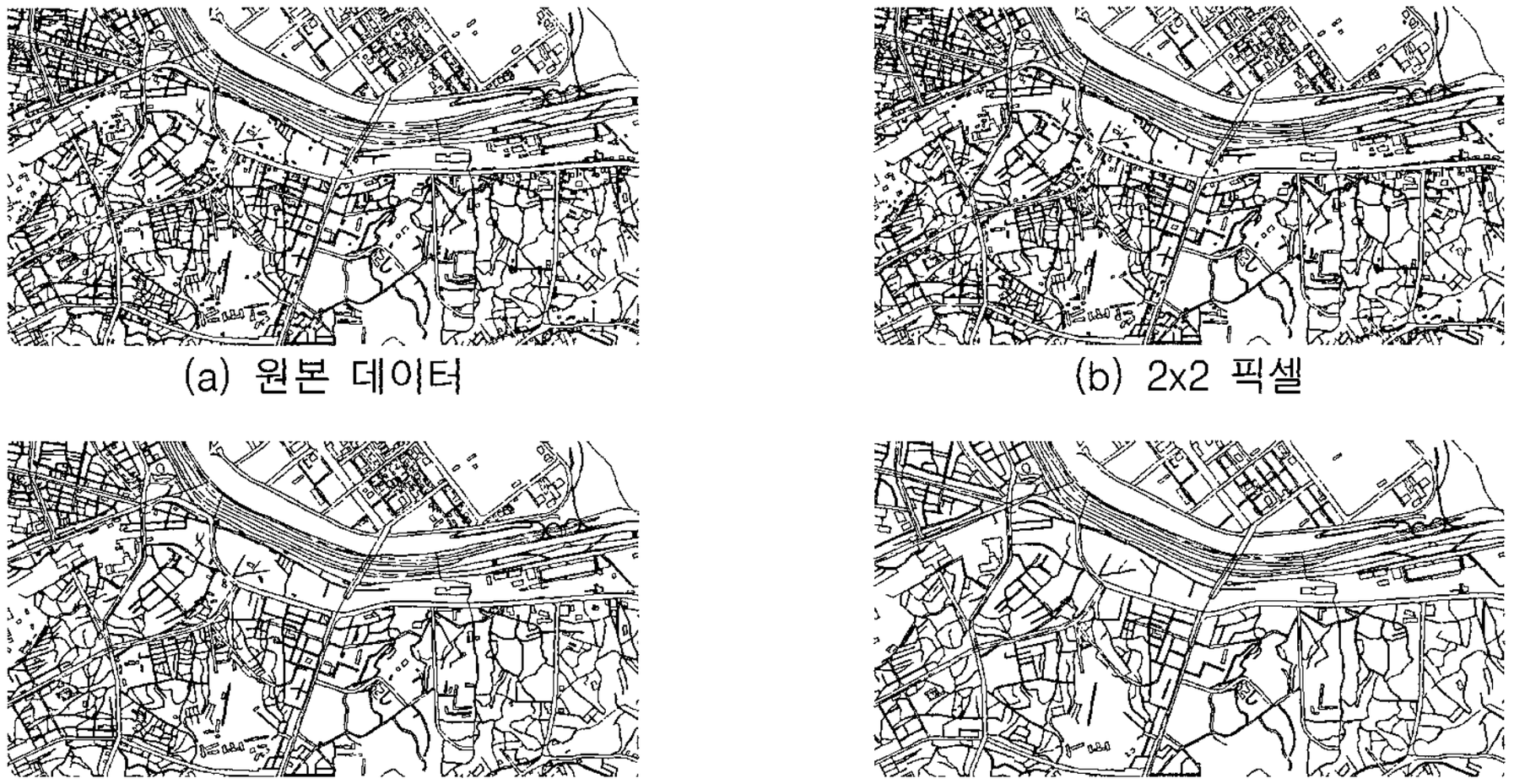
(b)의 경우는 (a)와 유사한 결과를 보여주며 (c)는 도로가 명확히 표현되어 (a)에 비해 가독성이 증가하였음을 알 수 있다. (d)는 작은 화면의 낮은 해상도에서 전체 윤곽을 보기에는 적절하다.

[그림 4, 5]는 고해상도를 지원하는 모바일 장치에서의 결과 화면을 보여준다. 고해상도이므로 저해상도 화면보다 일반적으로 가독성이 뛰어남을 알 수 있다.

[그림 4, 5]의 (b)는 (a)의 원본 데이터와

(a) 원본 데이터 (b) 2x2 픽셀 (c) 5x5 픽셀 (d) 10x10 픽셀

[그림 4] hx4700 VGA모드(480x562)에서의 너비 10% 질의 결과 화면 비교



(a) 원본 데이터 (b) 2x2 픽셀

(c) 5x5 픽셀 (d) 10x10 픽셀

[그림 5] NetForce(796x562)에서의 너비 10% 질의 결과 화면 비교

차이를 발견하기 힘들고, (c)도 여전히 복잡하며, (d)에서는 작은 객체들이 제외되어 가독성이 증가하였음을 알 수 있다. 해상도가 높아질수록 픽셀은 전체 화면에서 차지하는 면적이 작기 때문이다.

주관적인 차이가 존재하므로 특정 결과가 가장 지도 가독성이 우수하다고 판정할 수는 없지만, 본 논문에서 제안한 AR*-tree를 실제 응용 프로그램에 적용할 때는 사용자로부터 파라미터를 입력받아 자유롭게 지정함으로써 주관적인 취향에 따라 지도 가독성을 높일 수 있다는 장점이 있다. 그러므로 고정적인 R*-tree 보다는 사용자 정의를 통한 지도 가독성을 지원하는 AR*-tree가 지도 가독성 측면에서 유연성이 우수하다.

4.3 시간 효율성

시간 효율성에 대한 실험은 R*-tree를 사용하여 모든 데이터를 출력하는 시간과 AR*-tree를 통해 면적이 아주 작지 않은 (충분히 큰) 데이터만을 출력하는 시간을 비교하는 방법을 사용한다. 각 방법에 대해 100번의 랜덤 질의를 수행하여 시간을 측정하고 평균 시간을 계산한다. 각각의 질의에 대해서는 트리를 순환하면서 조건 검색 및 결과 리스트를 작성하는데 소요되는 인덱스 검색 시간, 검색 결과 데이터를 접근하여 공간 데이터를 읽는데 소요되는 데이터 접근 시간, 공간 데이터의 좌표를 해석하고 그래픽 함수를 호출하는데 소요되는 화면 출력 시간을 분리하여 측정하고 합계를 질의 처리 시간으로 계산한다.

<표 3, 4>는 메모리를 사용하여 데이터를 저장하는 모바일 장치인 rx3715 및 hx4700과 하드 디스크를 사용하여 데이터를 저장하는 모바일 장치인 NetForce에서의 실험 결과를 보여준다. 표의 수치는 임의의 질의에 대한 평균 시간으로서 밀리세컨드 단위(ms)이다.

<표 3>의 결과를 살펴보면 메모리를 사용하는 모바일 장치에서는 공통적으로 인덱스 검색 시간은 R*-tree가 100% 질의 영역을 제외하고는 모두 빠르지만 결과 객체의 수가 많아서 데이터 접근 시간 및 화면 출력 시간이 증가함을 알 수 있다. 앞 절에서 살펴본 바와 같이 AR*-tree를 통해 2×2 픽셀보다 작은 객체를 필터링한 경우에는 R*-tree를 통해 검색한 결과와 크게 차이가 없으면서도 성능에서는 가장 차이가 없는 경우(10% 질의 영역 크기, rx3715)에 77.9%의 시간으로 처리가 가능하였으며, 최대로는 100% 질의 영역 크기의 경우(rx3715)에 11.3%의 시간만으로도 처리가 가능하여 매우 우수한 성능을 입증하였다. 특히, 지도 가독성도 높이면서 동시에 시간 효율성도 뛰어난 5×5 픽셀 이하 객체를 필터링할 때 가장 효율성이 우수한 경우(100% 질의 영역 크기, rx3715)에는 R*-tree로 처리하는데 필요한 시간의 4.9%의 시간만으로 처리할 수 있어 매우 우수한 성능을 증명하였다. 10×10 픽셀보다 작은 객체를 필터링한 경우에는 R*-tree를 통해 모든 데이터를 출력할 때보다 가장 빠르게는 2.5%만의 시간으로 데이터를 출력할 수 있는 매우 우수한 성능을 나타낸다.

<표 3> 메모리를 사용하는 모바일 장치에서의 시간 측정 실험 결과

질의 영역 크기	공간 인덱스		최소 면적	질의 결과 객체 수	인덱스 검색 시간	데이터 접근 시간	화면 출력 시간	합계
iPAQ rx3715								
10%	R*-tree		-	1,407.8	34.3	470.1	389.4	893.8
	AR*-tree	2×2	963	657.5	155.2	285.9	255.1	696.2
		5×5	6,024	282.3	117.2	156.0	171.2	444.4
		10×10	24,098	135.0	89.0	90.7	124.1	303.8
25%	R*-tree		-	9,763.8	162.8	3,180.5	2,333.5	5,676.8
	AR*-tree	2×2	6,024	1,839.1	455.3	953.7	859.8	2,268.7
		5×5	37,654	616.4	266.6	400.8	482.4	1,149.7
		10×10	150,616	264.6	159.9	201.4	313.0	674.3
50%	R*-tree		-	37,375.3	556.8	12,268.3	8,799.9	21,625.1
	AR*-tree	2×2	24,098	3,099.0	892.1	1,903.6	2,106.9	4,902.6
		5×5	150,616	982.0	411.7	738.1	1,157.4	2,307.2
		10×10	602,466	409.1	253.6	360.6	684.0	1,298.2
75%	R*-tree		-	64,888.6	907.8	21,103.0	14,816.2	36,827.1
	AR*-tree	2×2	54,222	3,029.3	929.4	2,042.3	2,497.9	5,469.6
		5×5	338,887	882.7	419.5	741.7	1,254.2	2,415.3
		10×10	1,355,550	378.1	245.9	358.1	691.3	1,295.3
100%	R*-tree		-	70,351.0	969.2	22,659.3	15,745.2	39,373.7
	AR*-tree	2×2	96,394	2,240.0	764.8	1,589.7	2,105.7	4,460.2
		5×5	602,466	649.0	361.7	569.1	1,003.7	1,934.5
		10×10	2,409,866	280.0	184.1	275.8	541.9	1,001.9
iPAQ hx4700								
10%	R*-tree		-	1,486.3	25.3	491.8	428.0	945.1
	AR*-tree	2×2	240	634.5	93.6	296.4	260.1	650.2
		5×5	1,506	301.4	78.3	157.4	139.9	375.7
		10×10	6,024	139.8	63.3	80.2	73.1	216.6
25%	R*-tree		-	10,111.6	121.2	3,260.5	2,617.5	5,999.2
	AR*-tree	2×2	1,506	2,596.4	330.2	1,335.7	1,145.6	2,811.4
		5×5	9,413	878.4	214.8	516.6	475.8	1,207.2
		10×10	37,654	339.2	144.0	218.0	230.5	592.5
50%	R*-tree		-	40,080.0	431.8	12,810.1	9,991.8	23,233.7
	AR*-tree	2×2	6,024	5,318.4	773.2	3,065.9	2,873.3	6,712.4
		5×5	37,654	1,731.5	422.9	1,133.3	1,274.2	2,830.4
		10×10	150,616	666.7	235.8	474.7	686.4	1,396.9
75%	R*-tree		-	64,788.5	687.0	20,610.0	15,806.0	37,103.0
	AR*-tree	2×2	13,555	6,536.7	1,037.9	4,045.1	4,026.1	9,109.1
		5×5	84,721	2,034.7	510.2	1,410.7	1,784.1	3,704.9
		10×10	338,887	757.6	273.8	588.3	940.0	1,802.1
100%	R*-tree		-	70,351.0	793.9	22,177.3	16,729.7	39,701.0
	AR*-tree	2×2	24,098	5,525.0	938.8	3,568.3	3,763.9	8,271.0
		5×5	150,616	1,618.0	412.6	1,195.4	1,656.9	3,264.9
		10×10	602,466	649.0	243.7	527.9	889.0	1,660.6

<표 4> 하드 디스크를 사용하는 모바일 장치에서의 시간 측정 실험 결과

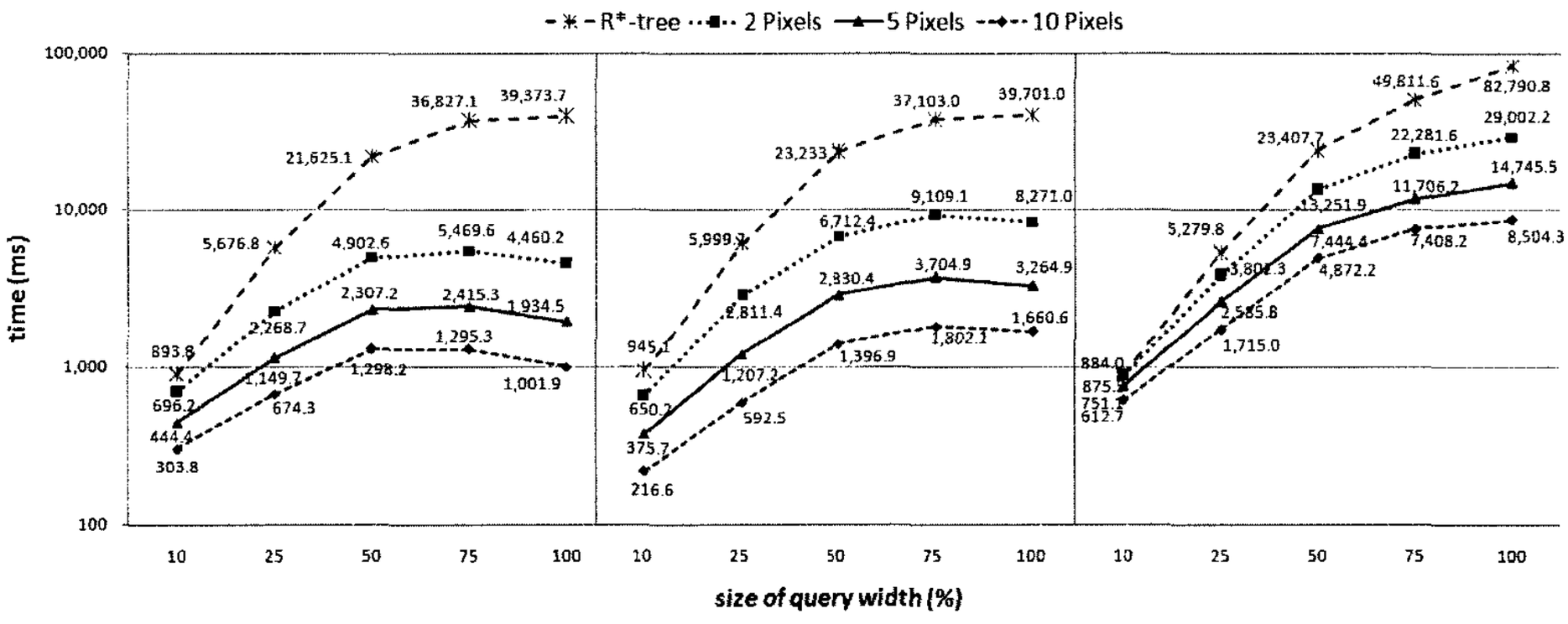
질의 영역 크기	공간 인덱스		최소 면적	질의 결과 객체 수	인덱스 검색 시간	데이터 접근 시간	화면 출력 시간	합계
iStation NetForce								
10%	R*-tree		-	559.7	90.2	679.9	113.9	884.0
	AR*-tree	2×2	87	528.6	167.4	596.9	111.2	875.5
		5×5	547	336.6	162.3	501.5	87.3	751.1
		10×10	2,190	197.8	151.8	393.5	67.4	612.7
25%	R*-tree		-	4,453.3	230.5	4,346.1	703.3	5,279.8
	AR*-tree	2×2	547	2,462.4	403.8	2,921.3	477.2	3,802.3
		5×5	3,423	1,089.4	366.0	1,930.5	289.3	2,585.8
		10×10	13,692	489.6	324.7	1,205.6	184.8	1,715.0
50%	R*-tree		-	18,984.8	635.1	19,793.6	2,979.0	23,407.7
	AR*-tree	2×2	2,190	6,163.7	1,079.2	10,724.1	1,448.6	13,251.9
		5×5	13,692	2,222.0	918.6	5,729.2	796.6	7,444.4
		10×10	54,768	994.2	771.0	3,590.9	510.3	4,872.2
75%	R*-tree		-	40,440.4	1,131.0	42,463.2	6,217.5	49,811.6
	AR*-tree	2×2	4,929	8,669.6	1,662.4	18,308.7	2,310.5	22,281.6
		5×5	30,807	2,935.7	1,317.1	9,167.2	1,221.9	11,706.2
		10×10	123,228	1,216.3	1,118.8	5,540.0	749.4	7,408.2
100%	R*-tree		-	70,351.0	1,722.3	70,954.7	10,113.9	82,790.8
	AR*-tree	2×2	8,762	10,189.0	2,141.0	23,971.0	2,890.2	29,002.2
		5×5	54,768	3,251.0	1,589.7	11,681.0	1,474.8	14,745.5
		10×10	219,073	1,220.0	1,313.3	6,343.6	847.5	8,504.3

<표 4>에서는 하드 디스크를 사용하는 모바일 장치의 처리 시간이 메모리를 사용하는 장치보다 오래 걸리고, 디스크를 사용할 때에도 R*-tree보다 AR*-tree를 통해 지도 가독성을 방해하는 충분히 작은 공간 객체를 필터링하는 것이 시간 효율성이 높음을 보여준다. 가장 성능이 차이나는 경우(100%, 10x10 픽셀)는 10.3%의 시간만으로 지도 가독성도 더욱 우수한 화면을 출력할 수 있는 성능을 보여준다.

[그림 6]은 <표 3, 4>의 질의 처리 평균 시간을 로그 축을 기준으로 도식화한 그래프를 보여준다. 앞서 설명한 것과 같이 R*-tree에 비해 AR*-tree는 시간 효율성 측면에서도 월등한 성능을 가지고 있음을 확인할 수 있다.

5. 결 론

본 논문에서는 모바일 소프트웨어를 위해 면적을 사용한 AR*-tree 공간 인덱스를 제안하였다. AR*-tree는 다음과 같은 장점을 갖는다. 첫째, 지도 가독성을 개선한다. 작은 모바일 장치의 화면에 크기가 작은 공간 데이터를 출력하면 점으로 표시



(a) rx3715 (b) hx4700 (c) NetForce

[그림 6] 질의 처리 평균 시간

되어 지도가 복잡해지면서 지도 가독성이 떨어지므로, 본 논문에서 제안하는 AR*-tree는 검색시에 2차원 질의 영역이외에 추가로 최소 및 최대 면적 파라미터를 입력받아서 화면에 출력하기에 너무 작은 공간 객체를 결과에서 제외할 수 있도록 지원하여 낮은 해상도에서도 지도 가독성이 우수한 지도를 출력할 수 있도록 하였다. 특히, 사용자의 주관적인 선호도를 반영할 수 있다는 장점이 있다. 사용자가 너무 큰 객체까지 제외한다면 지도의 표현력이 저하될 수 있으므로 구현시에는 최소 면적의 최대값을 제한하는 것도 고려해 볼 수 있다.

둘째, 시간 효율성이 우수하다. AR*-tree는 최소 면적 파라미터를 사용하여 화면에 출력하기에는 너무 작은 공간 객체들을 검색 과정에서 제외하여 데이터 접근 시간 및 화면 출력 시간이 줄어들게 되므로 R*-tree에 비해 지도 출력 시간 효율성이 우수하다.

제안한 공간 인덱스의 성능을 평가하기 위해서는 모바일 장치에서 파리미터를 변경해 가면서 공간 질의를 수행하여 결과 화면을 저장하고 시간을 측정하였다. 결과 화면을 통해 지도 가독성의 증가를 확인할 수 있었다. 또한, 측정된 시간을 통해서는 2×2 픽셀 만큼의 면적 필터링을 수행한 경우에는 기존 방법의 시간보다 11.3%의 시간만으로도 비슷한 화면을 출력할 수 있었다. 가장 시간 효율성이 우수한 경우에는 기존 방법의 2.5%만의 시간으로 지도 가독성도 더욱 높은 지도를 출력하여 본 논문에서 제안하는 AR*-tree는 매우 우수한 성능을 가진 것으로 측정되었다.

향후 연구 방향으로는 AR*-tree를 플래시 메모리에 최적화된 페이징 기법과 클러스터링 기법을 통해 효율성을 더욱 향상시키는 것이 있다.

참고문헌

Beckmann, Norbert, Hans-Peter Kriegel, Ralf Schneider, Bernhard Seeger, 1990, "The R*-

tree: An Efficient and Robust Access Method for Points and Rectangles", Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp. 322-331.

Biveinis, Laurynas, Simonas Šaltenis, Christian S. Jensen, 2007, "Main-memory operation buffering for efficient R-tree update", Proc. of the 33rd int. conf. on VLDB '07, pp. 591-602.

Gal, Eran, Sivan Toledo, 2005, "Algorithms and data structures for flash memories", ACM Computing Surveys(CSUR), 37(2), pp. 138-163.

Guttman, A., 1984, "R-trees: a dynamic index structure for spatial searching", Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp. 47-57.

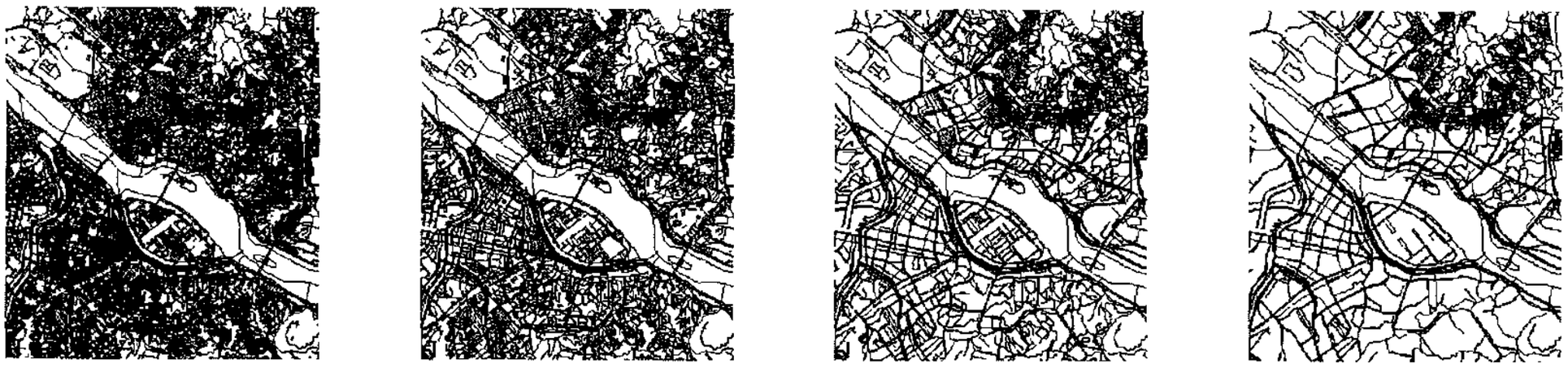
Han, Qiang, Michela Bertolotto, 2004, "A multi-level data structure for vector maps", Proc. of the 12th annual ACM int. workshop on Geographic information systems, pp. 214-221.

Liu, Bin, Wang-Chien Lee, Dik Lun Lee, 2005, "Distributed caching of multi-dimensional data in mobile environments", Proc. of the 6th int. conf. on Mobile data management, pp. 229-233.

Xia, Tian, Donghui Zhang, 2005, "Improving the R*-tree with outlier handling techniques", Proc. of the 13th annual ACM int. workshop on Geographic information systems, pp. 125-134.

오병우, 2006, "모바일 서비스를 위한 메인 메모리 기반 공간 데이터 관리자", 한국공간정보시스템학회 논문지 제8권 제1호, pp. 77-92.

[부록] 25%, 50 %, 75%, 100% 질의 구간에 대한 결과 예제

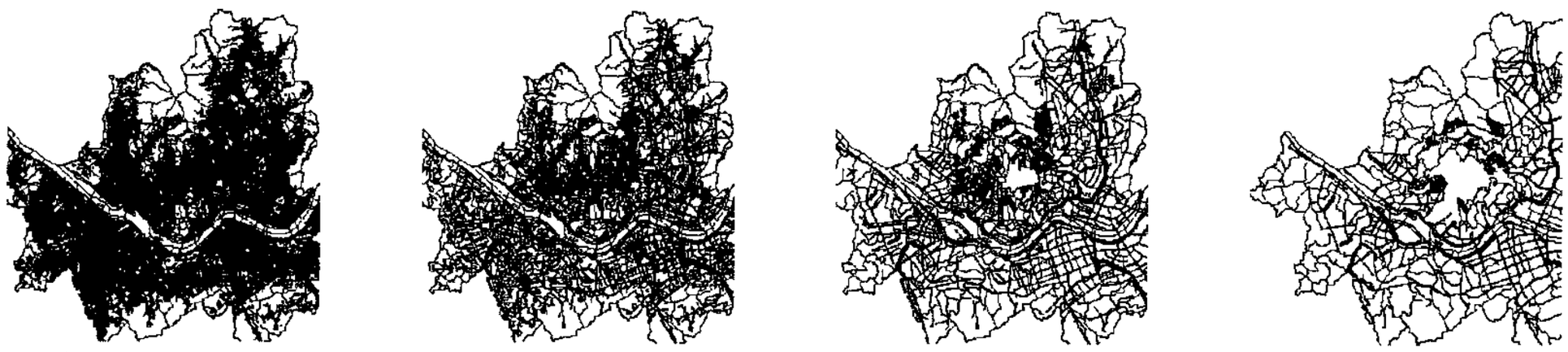


(a) R*-tree (b) 2x2 픽셀 (c) 5x5 픽셀 (d) 10x10 픽셀

[그림 I] rx3715(240x268)에서의 너비 25% 질의 결과 화면 비교

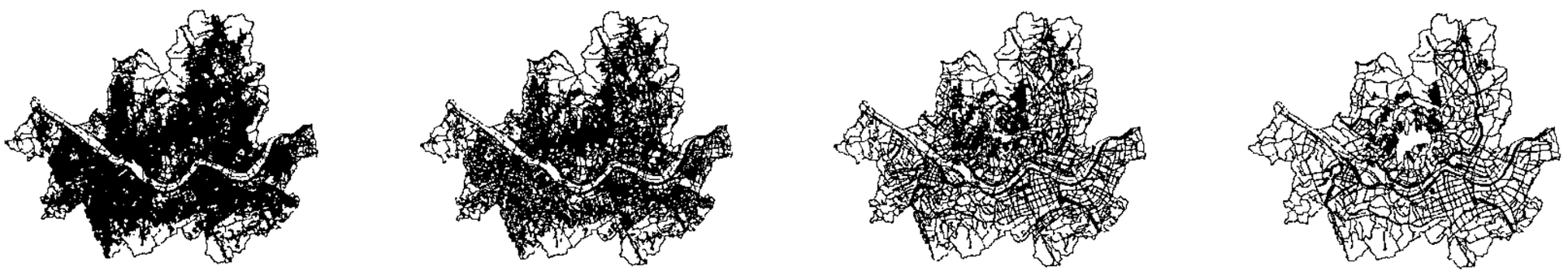
(a) R*-tree (b) 2x2 픽셀 (c) 5x5 픽셀 (d) 10x10 픽셀

[그림 II] NetForce(796x562)에서의 너비 50% 질의 결과 화면 비교



(a) R*-tree (b) 2x2 픽셀 (c) 5x5 픽셀 (d) 10x10 픽셀

[그림 III] rx3715(240x268)에서의 너비 75% 질의 결과 화면 비교



(a) R*-tree (b) 2x2 픽셀 (c) 5x5 픽셀 (d) 10x10 픽셀

[그림 IV] hx4700 VGA모드(480x562)에서의 너비 100% 질의 결과 화면 비교