이동객체궤적에 대한 효율적인 범위질의

박영희 · 김규재 · 조우현*

Efficient Range Query on Moving Object Trajectories

Young-hee Park · Gyu-jae Kim · Woo-hyun Cho*

Department of Computer Engineering, Pukyong National University, Busan 608-737, Korea

요 약

최근 많은 응용프로그램에서 시공간 데이터와 멀티미디어 데이터 등이 사용되고 있다. 그래서 이러한 유형의 데이터들을 효율적으로 관리하고 분석할 수 있는 연구들이 많이 진행 중이다. 본 논문에서는 이동객체궤적에 대하여 단순화 기법을 사용하여 단순화한 후에 색인구조를 생성하고 이 색인구조를 이용하여 범위질의를 효율적으로 처리할 수 있는 알고리즘을 제안한다. 이동객체궤적의 단순화 기법으로는 Douglas-Peucker 알고리즘을 수정하여 이용한다. 제안된 방법과 기존의 최소 경계 사각형(MBR)을 이용한 색인 방법을 실험을 통하여 비교 및 분석한다. 실험 결과로 제안된 방법에서는 색인 데이터 량이 상대적으로 작아지고 색인 및 질의 처리방법이 간단하며 기존의 방법보다 시공간적으로 효율적임을 확인하였다.

ABSTRACT

The management and analysis of spatio-temporal and multimedia data is a hot issue in database research because such data types are handled in many applications. Querying databases of such a content is very important for these applications. This paper addresses algorithms that make index structure by using Douglas-Peucker Algorithm and process range query efficiently on moving objects trajectories. We compare and analyze our algorithms and MBR by experiments. Our algorithms make smaller size of index structure and process more efficiently.

키워드 : 색인구조, 이동객체궤적, 범위질의

Key word : Index structure, Moving Object Trajectories, Range Query

접수일자 : 2013. 11. 21 심사완료일자 : 2014. 01. 06 게재확정일자 : 2014. 01. 20

* Corresponding Author Woo-hyun Cho(whcho@pknu.ac.kr, Tel: 051-629-6251)

Department of Computer Engineering, Pukyong National University, Busan 608-737, Korea

Open Access <http://dx.doi.org/10.6109/jkice.2014.18.2.364>

print ISSN: 2234-4772 online ISSN: 2288-4165

©This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright © The Korea Institute of Information and Communication Engineering.

I. 서론

이동객체란 시간의 변화에 따라 공간적인 위치 및 모양이 연속적으로 변하는 시공간데이터를 말하며 이동객체체계는 이러한 데이터들이 시간에 따라 연결된 궤적을 말한다. 최근 다양한 응용프로그램에서 위치 기반 서비스의 사용이 증가하면서 이러한 데이터들을 관리하고 사용하는 이동객체 데이터베이스에 대한 관심이 증가하고 있다.

단말기와 이동 통신의 발달로 현대 사회에서는 대부분의 사람들이 노트북, 스마트폰, 태블릿 등 휴대용 단말기를 하나 혹은 여러 개를 동시에 들고 다닌다. 다량의 단말기들은 많은 양의 이동객체체계 데이터를 생성하며, 이를 통한 위치 기반 서비스는 대량의 데이터를 효율적으로 처리할 수 있는 적절한 색인구조가 필요하다. 이동객체체계 데이터에 대한 색인구조를 생성하는 방법은 다양하며, 현재까지 B^+ -트리, 최소경계사각형(MBR) 근사법을 이용한 R-트리, 다항식 근사법 또는 사분트리, 등의 여러 가지 색인 방법이 제시되었다[1-5].

본 논문에서는 Douglas-Peucker 알고리즘[6]을 이용하여 이동객체체계를 단순화하고 이를 이용하여 색인구조를 생성하고 이 색인구조에 대하여 범위질의를 처리하는 알고리즘을 고안한다. 지금까지 널리 연구된 MBR방식과 그 성능을 비교 및 분석하여 본 논문의 방법이 더 단순하고 적은 양의 데이터를 생성하며 범위질의 처리속도가 빠른 이점이 있음을 보인다.

본 논문의 구성은 다음과 같다. II장에서는 이동객체의 시공간 색인에 대한 관련 연구들을 살펴보고 III장에서는 색인구조를 생성하는 알고리즘과 이 색인구조에 대하여 범위질의를 처리하는 알고리즘을 제안한다. IV장에서는 제안하는 색인구조 알고리즘의 우수성을 입증하기 위해 기존 방법과의 성능 평가를 수행한다. 마지막으로 V장에서는 결론 및 향후 연구에 대해 기술한다.

II. 관련 연구

이동객체체계에 대한 질의는 범위질의, k-최근접 이웃검색[7]등이 있는데 이러한 질의들을 효율적으로 처리하기 위해서 색인구조를 생성하여 이용한다. 이러한

색인구조를 생성하기 위해서 어떠한 모양의 데이터 구조와 근사 방법을 사용하는지는 아주 중요한 요소가 되며 색인구조 검색 효율에 영향을 미친다.

주로 쓰이는 공간색인 방법으로는 이동객체체계 데이터가 포함된 공간을 분할하여 색인하는 사분트리와 해싱을 이용한 방법[3]과 이동객체체계 데이터를 다항식으로 근사하여 검색 공간을 줄이는 방법[4,5], 이동객체체계 데이터를 최소경계사각형(MBR)으로 분할하여 R-트리 기반의 색인구조를 생성하는 방법[1], B^+ -트리를 기반으로 하는 방법[2]과 같은 데이터 분할 색인이 있다. 이러한 방법들은 이동객체체계에 대한 색인 구성 과정에서 특별한 데이터 구조를 사용하기도 하고 데이터가 포함된 공간 자체를 분할하는 방법 또는 MBR과 같이 실제 데이터에 대한 근사치 영역을 이용해 이동객체체계 데이터 자체를 분할하는 방법을 사용한다.

본 논문에서 응용한 Douglas-Peucker 알고리즘[6]도 MBR과 마찬가지로 실제 위치 데이터에 대한 근사치 영역을 사용하며 실제 점과 직선으로 이루어진 데이터가 있을 때 단순화정도가 주어지면 그 값을 근사치 기준으로 하여 더 단순한 점과 직선들로 근사하는 알고리즘이다. 기존의 MBR을 기반으로 하는 방법과 색인을 생성하는 과정이 비슷하기 때문에 본 논문에서 제안하는 알고리즘의 성능을 분석하기 위해 MBR방법과 비교 및 분석을 해보기로 한다.

MBR기반의 색인구조는 여러 연구나 논문에서 사용되고 있으며 이동객체체계 데이터를 가장 작은 경계의 사각형들로 분할하여 색인구조를 생성하는 방법으로 아주 효율적으로 범위질의를 처리할 수 있다. 하지만 색인구조는 질의를 효율적으로 처리할 수 있을 뿐만 아니라 업데이트 또한 효율적으로 처리할 수 있어야 하는데 MBR기반의 R-트리의 경우 색인 노드들을 분할하는 과정에서 중복에 의한 오버헤드가 심하여 업데이트 비용이 아주 높다는 단점이 있다. 이는 계속해서 움직이는 이동객체와 같이 이동객체체계의 데이터 변화가 빈번한 응용프로그램에서 사용하기에는 한계가 있음을 나타낸다.

III. 색인구조 및 범위질의

이 장에서는 Douglas-Peucker 알고리즘을 응용하여

이동객체체적 데이터를 단순화시켜서 색인구조 데이터를 만드는 알고리즘과 이 색인구조에 대한 범위질의 처리 알고리즘을 제안한다. 먼저 색인구조생성 알고리즘을 단순하게 도식화하면 그림 1과 같다.

그림 1. 이동객체체적에 대한 색인구조 생성 도식화
Fig. 1 Diagram of Indexing algorithm on moving object trajectories

색인구조생성 알고리즘은 Douglas-Peucker 알고리즘 방식으로 이동객체체적 데이터를 단순화하고 단순화된 궤적이 원래의 궤적을 가리키는 식으로 색인구조를 생성한다.

Algorithm 1. 이동객체체적에 대한 색인 구조 생성
Input : 이동객체체적데이터($srcList$), 단순화범위($Epsilon$), 인덱스(s)
Output : 색인구조데이터($dstList$)

```

IndexCreation( $srcList$ ,  $Epsilon$ ,  $s$ )
1. FileWrite(궤적데이터파일명+".dug");
2. FirstPoint =  $srcList.getFirst()$ ;
3. endPoint =  $srcList.getLast()$ ;
4. length =  $srcList.length()$ ;
5.  $dmax = 0$ ;
6.  $index = -1$ ;
7. for( $int i=0; i<length; i++$ )
8.   distance = vertical( $srcList.get(i)$ , FirstPoint, endPoint);
9.   if(distance >  $dmax$ )
10.     $dmax = distance$ ;     $index = i$ ;
11. if( $dmax > Epsilon$ )
12.    $lList = IndexCreation(src, devidList(srcList, 0, index), Epsilon, s)$ ;
13.    $rList = IndexCreation(src, devidList(srcList, index, length), Epsilon, s+index)$ ;
14.   return (mergeList( $lList$ ,  $rList$ ));
15. else
16.   FileWrite.write(startPoint);
17.   FileWrite.write( $s$ );
18.   FileWrite.write(length);
19.   return (mergeList(startPoint, endPoint));
    
```

그림 2. 이동객체체적에 대한 색인구조 생성 알고리즘
Fig. 2 Indexing algorithm on moving object trajectories

데이터를 어느 정도로 단순화 할 것인지를 나타내는

단순화범위($Epsilon$)가 입력되어야 한다. 단순화 과정은 먼저 단계 2에서 단계 3까지 이동객체체적 데이터에서 첫 번째 데이터(firstPoint)와 마지막 데이터(endPoint)를 읽고, 단계 5에서 단계 10까지 이동객체체적의 모든 위치데이터들을 차례대로 읽으면서 첫 번째와 마지막 위치데이터로 이루어진 직선으로 수선을 그어서 그 거리를 계산하고, 수선들 중에서 길이가 가장 긴 최대수선의 길이($dmax$)와 그 값을 가지는 위치데이터의 인덱스(index)를 구한다.

마지막으로 단계 11에서 단계 19까지 구해진 최대수선의 길이가 단순화범위 안에 포함이 되면 단순화작업을 수행하여 색인구조결과를 출력하고, 최대수선의 길이가 단순화범위를 벗어나면 단순화 작업을 하지 않고 구해진 인덱스를 기준으로 이동객체체적의 위치데이터를 양쪽으로 나누어서 각각 알고리즘을 재귀호출로 반복적으로 수행한다. 더 이상 재귀호출이 발생하지 않고 모든 점들에 대한 단순화작업이 끝나면 알고리즘이 종료된다.

그림 3. 단순화된 색인구조 (a) 도식화 (b) 데이터 모형
Fig. 3 Simplified index structure (a) Diagram (b) Data model

그림 3은 제안된 색인구조 생성 알고리즘의 결과를 간단한 구조로 나타낸 것이다. 여러 개의 이동객체체적 데이터에 대해서 위 알고리즘의 시간 복잡도는 이동객체체적의 개수만큼 Douglas-Peucker 알고리즘을 수행

한 시간으로 나타낼 수 있다. 하나의 이동객체쿼적이 n 개의 위치데이터를 가질 때 Douglas-Peucker 알고리즘의 시간 복잡도는 $\Theta(n \cdot \log n)$ 이 되고, m 개의 이동객체쿼적들을 계산해야 하므로 알고리즘1의 시간복잡도는 $\Theta(m \cdot n \cdot \log n)$ 이 된다.

그림 4. 이동객체쿼제에 대한 색인구조 생성 도식화

Fig. 4 Diagram of Indexing algorithm on moving object trajectories

본 논문에서 제안하는 알고리즘으로 색인구조를 생성하고 이에 대한 범위질의를 처리하기 위해서는 크게 두 단계의 작업이 필요하다. 먼저 색인구조에서는 착오배제(False Drop)를 해결하기 위해서 범위질의의 크기를 확장하여 검사를 하고 확장된 범위질의에 포함이 되면 색인정보를 통해 원본데이터에 접근하여 원래 크기의 범위질의의 검사를 수행한다.

Algorithm 2. 범위질의에 대한 이동객체쿼제 검색
Input : 색인데이터(dug), 질의범위(qx, qy, width, height)
Output : 질의범위에 포함된 이동객체쿼제들(result)

```

DouglasQuery(dug, qx, qy, width, height)
1. FileRead(dug);
2. Epsilon = FileRead.readInt();
3. eqx = qx-Epsilon;          eqy = qy-Epsilon;
4. ewidth = width*(2+Epsilon); eheight = height*(2+Epsilon);
5. point1 = FileRead.readPoint();
6. while(FileReader.available() > 0)
7.   startIndex = FileRead.readInt();
8.   countPoint = FileRead.readInt();
9.   point2 = FileRead.readPoint();
10.  if(queryIntersect(eqx, eqy, ewidth, eheight, point1, point2))
11.    if(queryInSrcFile(dug, qx, qy, width, height, startIndex, countPoint))
12.      result = dug;
13.      break;
14.  point1 = point2;
15. return result;
  
```

그림 5. 이동객체쿼제에 대한 범위질의 처리 알고리즘

Fig. 5 Range query algorithm on moving object trajectories

그림 5는 범위질의의 처리 알고리즘으로, 생성된 색인구조 데이터에 대하여 범위질의를 수행하면 먼저 색인구조 데이터는 단순화된 쿼제 데이터이기 때문에 검사하기 전에 단계 2에서 단계 4까지 착오배제(False Drop)를 해결하기 위해서 범위질의의 크기를 색인구조 데이터의 단순화범위(Epsilon)만큼 확장한다.

그림 6. 범위질의의 크기 확장

Fig. 6 Extending size for range query

그 다음 단계 5에서 10까지 색인구조 데이터로부터 차례대로 데이터를 읽으면서 확장된 범위질의에 속하는지 검사한다. 만약 확장된 범위질의에 포함된다면, 해당 색인구조 데이터의 이동객체쿼제는 후보 이동객체쿼제에 대하여 원래 크기의 범위질의에 대한 상세검사(알고리즘3)를 수행하여 최종 결과를 출력한다. 알고리즘2의 시간복잡도는 크게 두 단계로 n 개의 위치데이터를 가지는 m 개의 이동객체쿼제를 압축률 C 로 색인구조 데이터를 만들어 확장된 범위질의를 처리하고, k 개의 결과 후보이동객체쿼제에 대하여 원래 크기의 범위질의를 계산해야 하므로 $\Theta(n \cdot (C \cdot m + k))$ 이 된다.

Algorithm 3. 후보 이동객체쿼제에 대한 상세검사

Input : 후보 이동객체 쿼제데이터(candidate), 질의범위(qx, qy, width, height), 색인정보(startIndex, count)
Output : 상세검사 결과(true 혹은 false)

```

queryInSrcFile(candidate, qx, qy, width, height, startIndex, count)
1. FileRead(candidate.getName()+"_src");
2. FileRead.skip(startIndex);
3. point1 = FileRead.readPoint();
4. for(int i=0; i<count-1; i++)
5.   point2 = FileRead.readPoint();
6.   if(queryIntersect(qx, qy, width, height, point1, point2))
7.     return true;
8.   point1 = point2;
9. return false;
  
```

그림 7. 범위질의에서 후보이동객체쿼제 상세검사

Fig. 7 Refinement check on candidate moving object trajectories in range query algorithm

상세검사 과정은 후보 이동객체 궤적 데이터에서의 입력된 색인정보가 가리키는 위치와 범위의 데이터들을 차례대로 범위질의에 대한 검사를 수행하고 만약 데이터가 범위질의에 포함되면 TRUE를 결과로 출력하고, 검사받는 모든 데이터가 범위질의에 포함되지 않으면 FALSE를 결과로 출력한다.

IV. 실험 및 비교분석

본 논문에서 제안하는 알고리즘의 성능을 분석하기 위해서 기존의 이동객체 궤적 색인방법인 MBR방법과 비교 실험을 해보았다. MBR방법은 앞서 말했다시피 실제 위치 데이터에 대한 근사치 영역을 이용한 방법이다. 본 논문의 방법도 실제 이동객체 궤적의 데이터를 더 단순한 궤적데이터로 만들어서 색인구조를 생성하므로 실제 위치 데이터에 대한 근사치 영역을 사용하며 기존의 MBR방법과 아주 유사한 방법으로 색인구조를 생성한다.

이 비교 실험은 Intel(R) Core i5-2400 CPU 3.10GHz 프로세서와 메모리 3.24Gbyte, Windows 운영체제를 사용하는 시스템 상에서 수행되었으며, 알고리즘 구현을 위해서 Java 언어를 사용하였다.

원래 궤적의 위치 정보는 2차원 좌표로 표현되고, 시간을 고려하여 3차원으로 궤적을 나타낸다. 그러나 본 논문에서는 간단히 하기 위해 위치정보를 1차원 좌표로 표현하고 시간정보를 고려하여 궤적을 2차원 공간으로 나타낸다. 각 실험은 총 120개의 일정한 시간에 따른 위치데이터를 가지는 100개의 동일한 파일들을 MBR방법과 본 논문의 알고리즘 방법으로 색인화 하였고, 압축 비율은 1/10로 하여 색인구조 데이터에서 12개의 위치데이터를 가지도록 하였다.

비교 항목으로는 범위질의에 대하여 각 방법으로 질의를 처리할 때, 알고리즘의 정확도 그리고 질의를 처리하는데 걸리는 시간을 비교하였다. 여기서 정확도는 질의 검사를 하는데 있어 최종 결과 이동객체 궤적과 후보이동객체 궤적의 비율을 말한다. 즉, 결과를 도출하는데 있어 얼마만큼의 후보가 추출되었는지를 나타내며, 예를 들면 범위질의의 검사에 대하여 후보 이동객체 궤적이 100개가 추출되고, 결과로서 이동객체 궤적이 80개가 질의에 포함되면 80/100으로 80%의 정확

도를 가진다.

아래 그림 8과 그림 9는 범위질의의 크기에 따라서 두 방법의 수행속도와 정확도가 어떻게 달라지는지를 실험한 결과이다. 각 실험에서의 정확도 값과 수행시간 값은 한 번의 범위질의에 대하여 100개의 모든 파일을 검사할 때 걸리는 시간과 그에 따른 정확도이며, 500여 차례 이상의 반복실험을 통하여 평균을 내었다.

그림 8. 범위질의의 크기에 따른 알고리즘 수행시간
Fig. 8 Algorithm execution times for various query sizes

그림 9. 범위질의의 크기에 따른 알고리즘 정확도
Fig. 9 Algorithm accuracy for various query sizes

정확도 부분에서는 MBR방법이 본 논문의 알고리즘 방법보다 대체적으로 높은 평균값이 나왔지만, 알고리즘 수행시간 그래프를 보면 MBR방법 보다 제안한 알고리즘의 방법이 수행시간이 더 적게 걸린다는 것을 알 수 있다. 범위질의의 크기가 증가할수록 두 방법 모두 수행시간이 줄어들었으나 두 방법 간의 수행시간의 차이 변화는 없었고 모든 질의 크기에서 본 논문에서의 방법이 더 적은 시간 안에 범위질을 처리하였다.

이는 본 논문의 방법이 더 많은 위치데이터를 검사하지만 전체 색인구조 데이터의 크기가 상대적으로 MBR방법보다 작아서 시간이 적게 걸린다는 것을 알 수 있다.

그림 10. 범위질의 크기에 따른 알고리즘 수행시간(압축비율: 1/5)
Fig. 10 Algorithm execution times for various query sizes (compression ratio: 1/5)

그림 11. 범위질의 크기에 따른 알고리즘 정확도(압축비율: 1/5)
Fig. 11 Algorithm accuracy for various query sizes (compression ratio : 1/5)

그림 10과 그림 11의 그래프는 이전 실험과 동일한 데이터를 가지고 압축 비율을 조정하여 실험한 결과로서 마찬가지로 제안한 방법이 범위질의를 처리하는데 걸리는 시간은 기존의 방법보다 더 적게 걸리는 것을 볼 수 있다.

위의 실험들 이외에도 시간에 따른 위치 변화도를 조정해보거나 데이터의 압축 비율 등을 조정해보면서 여러 방면으로 실험을 해보았고 실험 결과로는 모두 비슷한 결과가 나왔다.

본 논문에서 제안한 알고리즘의 방법은 범위질의를 처리하는데 있어서 질의의 크기를 확장하기 때문에 알고리즘의 정확도는 기존의 MBR방법보다 부족하지만, 직선형 데이터를 사용하기 때문에 사각형 데이터를 사용하는 기존의 색인구조 방법보다 좀 더 가볍고 단순한 구조의 색인구조를 생성하고 데이터의 크기 또한 줄어들어서 범위질의를 처리하는데 있어 전체적인 처리 시간을 절약할 수 있다.

V. 결 론

본 논문에서는 이동객체체제에 대한 색인구조생성과 범위질의를 처리하는 알고리즘을 제안하였다. 제안하는 알고리즘들은 Douglas-Peucker 알고리즘을 응용하여 단순한 색인구조를 만들고 이를 이용하여 범위질의를 처리한다. 기존의 MBR방법과의 성능 비교 실험 결과 제안하는 알고리즘의 방법이 더 적은 데이터양의 색인구조를 생성하고 범위질의를 처리하는데 걸리는 시간이 더 적게 걸리는 결과가 나왔다. 이는 제안하는 알고리즘을 이용하여 이동객체체제 데이터를 색인구조로 만들어 두면 범위질의를 처리하는데 있어 처리하는 데이터의 양이 감소되어 성능 향상을 이룰 수가 있다는 것을 알 수 있다. 또한 여러 유형의 데이터를 검사하여 본 논문의 알고리즘의 정확도를 좀 더 개선할 수 있다면 기존의 MBR방법보다 아주 효과적인 색인구조 시스템을 만들 수 있다.

본 논문의 내용은 단순히 두 방법의 알고리즘만을 비교한 내용으로 향후 더욱 정확한 비용모델을 제시하기 위해서 이동객체체제에 대하여 좀 더 구체화된 색인구조 구현을 통한 실험이 필요하다.

감사의 글

이 논문은 부경대학교 자율창의학술연구비 (2013년)에 의하여 연구되었습니다.

REFERENCES

- [1] Marios Hadjieleftheriou, George Kollios, Dimitrios Gunopulos, and Vassilis J. Tsotras. "Efficient Indexing of Spatio temporal Objects," in *Proceeding of the 8th International Conference on Extending Database Technology*, London, pp. 251-268, 2002.
- [2] Christian S. Jensen, Dan Lin, and Beng Chin Ooi. "Query and Update Efficient B+-Tree Based Indexing of Moving Objects," in *Proceeding of the 30th VLDB Conference*, Toronto, Canada, 2004.
- [3] Dong Li, Yu-hui Peng, and Jiang-long Yin. "Quadtree and

- Hash Table Based Index Structure for Indexing the Past, Present and Future Positions of Moving Objects,” *Computer Science and its Applications*. CSA '08. International Symposium on, Hobard, pp. 17-21, 2008.
- [4] Elena Braynova. “Indexing Spatio-Temporal Trajectories with Orthogonal Polynomials,” in *Conference on Data Mining : DMIN*, pp. 343-348, 2006.
- [5] Jinfeng Ni and Chynya V. Ravishankar, Senior Member, IEEE. “Indexing Spatio-Temporal Trajectories with Efficient Polynomial Approximations,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 5, pp. 663-678, May. 2007.
- [6] D. Douglas and T. Peucker. “Algorithms for the reduction of the number of points required to represent a digitized line or its caricature,” *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112-122, December. 1973.
- [7] Hans-Peter Kriegel, Peer Kroger, and Matthias Renz. “Techniques for Efficiently Searching in Spatial, Temporal, Spatio-temporal, and Multimedia Database,” in *Proceeding of the 14th ICDE, DASFAA'09*, Brisbane. Australia, pp. 780-783, 2009.

박영희(Young-hee Park)

2002년 한국방송통신대학교 컴퓨터정보학과 학사
2006년 부경대학교 산업대학원 석사
2009년~현재 부경대학교 컴퓨터공학과 박사과정
※관심분야 : 데이터베이스, 디지털 포렌식

김규재(Gyu-jae Kim)

2013년 부경대학교 컴퓨터멀티미디어공학과 학사
2013년~현재 부경대학교 컴퓨터공학과 석사과정
※관심분야 : 데이터베이스, 센서 네트워크

조우현(Woo-Hyun Cho)

1985년 경북대학교 전자공학과 전산공학전공(공학사)
1988년 경북대학교 대학원 전자공학과 (공학석사)
1998년 경북대학교 대학원 전자공학과 전산공학전공(공학박사)
1989년~현재 부경대학교 공과대학교 컴퓨터공학부 교수
※관심분야 : 지능형 데이터베이스, 멀티미디어 인덱싱, 객체 데이터베이스 관리 기술