Journal of KIIT. Vol. 21, No. 1, pp. 79-90, Jan. 31, 2023. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2023.21.1.79

이동객체의 궤적 데이터에 대한 온라인 도로 네트워크 매칭 방법

정성모*, 송석일**

Online Road Network Matching Method for Trajectory Data of Moving Objects

Seongmo Jeong*, Seokil Song**

This work is supported by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport(Grant 22AMDP-C160501-02)

요 약

최근 궤적 데이터가 가진 공간적 특징을 이용하여 도로 속도 예측, 교통 혼잡 예측 등 다양한 위치 기반 응용들이 연구되고 있다. 본 논문에서는 상태 천이 모델과 그리드 색인 방법을 이용하는 궤적 데이터에 대한 도로 네트워크 매칭 방법을 제안한다. 제안하는 매칭 방법은 궤적 데이터의 연속적인 위치 데이터에 대해 도로 네트워크에 대한 링크매칭, 매칭유지, 매칭변경, 매칭오류 상태를 판단하는 상태 천이 모델을 통해 매칭 정확도를 향상 시킨다. 또한, 제안하는 매칭 방법은 궤적의 위치 데이터와 도로 네트워크 간의 비교 연산을 줄이기 위해 그리드 색인(Grid Index)을 이용한다. 이 논문에서는 제안하는 방법을 구현하고 기존의 매칭 방법과 비교 실험을 수행하여 기존 방법보다 높은 정확도와 빠른 동작시간을 증명하여 제안 방법의 우수성을 입증했다. 제안하는 방법을 통해 실시간 도로 속도 예측, 교통 혼잡 예측 등 다양한 연구에 도움이 될 것이다.

Abstract

In this paper, we propose a road network matching method for trajectory data using a state transition model and a grid index method. The proposed matching method improves the matching accuracy through a state transition model that determines the link matching, matching maintenance, matching change, and matching error status to the road network for continuous location data of the trajectory data. In addition, the proposed matching method uses a grid index to reduce the comparison operation between the location data of the trajectory and the road network. In this paper, the superiority of the proposed method is showed by implementing the proposed method and performing comparative experiments with the existing matching method.

Keywords
trajectory, matching, grid index, state transition

* 한국교통대학교 컴퓨터공학과 공학석사
- ORCID: https://orcid.org/0000-0002-9492-3900
** 한국교통대학교 컴퓨터공학과 교수(교신저자)
- ORCID: https://orcid.org/0000-0002-0110-7155

· Received: Nov. 25, 2022, Revised: Jan. 12, 2023, Accepted: Jan. 15, 2023
· Corresponding Author: Seokil Song
School of Computer, Korea National University of Transportation,
Daehakro 50, Chungju, Chungbuk 27469, Korea
Tel.: +82-43-841-5349, Email: sisong@ut.ac.kr

Ⅰ. 서 론

최근에 스마트폰, 네비게이션, 블랙박스 등 GPS를 탑재한 기기들의 사용이 증가하면서 궤적 데이터가 폭발적으로 증가해왔다. 궤적 데이터는 주로 GPS가 장착된 장치를 가지고 이동하는 객체에 의해 생성되는 연속적인 위치 데이터로 표현된다[1]. 궤적 데이터 저장 및 관리, 이를 이용한 도로 속도 예측, 교통 혼잡 예측, 이동패턴 분석 등 궤적 데이터에 대한 다양한 연구가 진행되어왔다[2]-[5].

도로상의 궤적 데이터를 분석하기 위해서는 먼저 궤적 데이터를 도로 네트워크에 매칭(Matching) 해야 한다[4]. 예를 들어서, 궤적 데이터를 기반으로 도로 속도를 예측하기 위해서는 궤적 데이터가 어떤 도로에 대한 것인지 알아야 가능하다. 대한민국의 경우 국가 표준 노드-링크(Node-link)[6]를 정의하여 도로 네트워크를 연속적인 노드 및 링크의 집합으로 표현한다. 표준 노드-링크에서 노드는 교차로, 고가도로 분기점 등 차량이 주행하면서 속도가 변화는 지점이다. 링크는 노드와 다른 노드를 연결하는 선으로 도로의 특정 구간에 해당된다. 각 링크는 링크 식별자(Link identification)로 구분되고 링크에 대한 간략한 정보와 링크를 구성하는 연속적인 위치 데이터를 포함한다.

기존 매칭 방법은 접근 방법에 따라 궤적과 도로 네트워크의 유사성을 측정하는 기하학적 방법, 도로 네트워크의 연결성, 연속성과 같은 특징을 활용하여 매칭하는 토폴로지 방법과 통계기법을 이용한 매칭 방법으로 분류될 수 있다. 또한, 사용 목적에 따라 온라인 매칭에 적합한 방법과 오프라인 매칭에 적합한 방법으로 구분할 수도 있다. 온라인 매칭은 전체 궤적 데이터를 이용하지 않고 실시간으로 입력되는 위치 데이터에 대해 매칭을 수행해야 한다. 일반적으로 온라인 매칭에 적합한 방법들은 정확도보다는 매칭 속도가 중요하다. 오프라인 매칭은 전체 궤적 데이터를 이용하여 수행되며 실시간성은 필요가 없으며 속도보다는 매칭 정확도가 더 중요하다.

이 논문에서는 온라인 매칭이 가능할 정도로 빠른 속도로 매칭을 하면서도 기존 오프라인 매칭 방법보다 정확도를 높이는 새로운 매칭 방법을 제안한다. 제안하는 방법은 기본적으로 토폴로지 방법 중 하나인 다중 가설 매칭 기술을 기반으로 하며 매칭 상태를 상태 천이 그래프로 표현하여 매칭의 정확도를 향상한다. 또한, 궤적 매칭 속도 개선을 위해서 궤적 데이터를 구성하는 위치 데이터와 거리가 가장 가까운 링크를 찾기 위해 그리드 색인(Grid index)[7] 기법을 적용한다.

제안하는 매칭 방법은 궤적 데이터의 연속적인 위치 데이터와 도로 네트워크의 노드, 링크를 비교하여 링크매칭(Link matching), 매칭유지(Matching maintenance), 매칭변경(Matching change), 매칭오류(Matching error)의 상태를 판단한다. 상태 판단은 연속적인 위치 데이터와 링크간의 방향(Heading) 유사성, 거리, 포함 여부를 기준으로 한다. 또한, 제안하는 방법은 궤적 데이터의 특정 위치 데이터의 초기 매칭 링크를 찾기 위해 그리드 색인을 이용하여 빠르게 후보 링크를 찾아 비교 연산을 줄여 매칭 속도를 개선한다. 이 논문에서는 제안하는 매칭방법을 실제로 구현하여 기존 방법인 FMM(Fast Map Matching) 알고리즘[8]과 성능을 비교하기 위한 실험을 진행한다. 구현한 매칭방법을 기반으로 실제 서비스가 가능한 매칭 시스템을 개발한다.

Ⅱ. 관련 연구

그리드 색인 기법은 대상 공간을 일정한 크기의 셀(Cell)들로 나누어 대상 공간을 색인하는 것이다. 특정 데이터와 같은 대상 공간에 포함된 데이터를 질의할 때, 먼저 어떤 셀에 데이터가 있는지 찾아야 한다. 특정 데이터는 셀을 나눈 기준값과 계산을 통해 특정 해당 셀을 찾아 나머지 셀은 가지치기를 수행하고 해당 셀에서 데이터를 찾는다. 기존 연구로 동적 공간-텍스트 데이터와 정적 공간-텍스트 데이터를 그리드 색인 기법을 통해 공간적인 질의나, 텍스트 대상의 질의를 수행하는 그리드 기반 공간-텍스트 색인 방법[7] 등이 있다.

기존에 제안된 궤적 데이터의 도로 네트워크 매칭 방법들은 앞서 설명한 것처럼 기하학적 방법, 토폴로지매칭방법, 통계기법 등 크게 3가지로 구분할 수 있다[9].

첫 번째, 기하학적 방법은 매칭하려는 궤적과 도로 네트워크의 각 링크 사이의 유사성을 측정해서

가장 유사한 도로 네트워크 링크로 궤적 데이터를 매칭한다. 각 구간 사이의 유사성은 점대점(Point-to-point), 점대곡선(Point-to-curve), 곡선대곡선(Curve-to-curve) 방식으로 측정한다. 점대점 방식은 궤적의 위치 데이터와 링크에 포함된 위치 데이터들 중 가까운 위치 데이터사이의 거리를 측정하는 방법이다. 점대곡선 방법은 궤적의 위치 데이터와 링크 사이의 거리를 측정한다. 마지막으로 곡선대곡선 방법은 연속적인 궤적의 위치데이터들과 링크사이의 거리를 측정하는 방법이다[11]. 기하학적 매칭의 대표적인 방법으로 [10]이 제안된 바 있다.

두 번째, 토폴로지 기반의 궤적 매칭 방법이다. 토폴로지 기반의 궤적 매칭 방법 중 다중 가설 매칭 기술은 궤적 데이터와 매칭이 가능한 다수의 후보 링크들로 가설을 세우고 가설을 지속적으로 변경(Hypothesis updating)하면서 매칭을 수행한다. 가설 변경은 가설 분기(Hypothesis branching)와 가설 가지치기(Hypothesis pruning)로 구성된다. 가설 분기는 교차로에서 여러 경로로 이동이 가능할 때 각 이동 가능한 경로를 가설로 설정하는 것이다. 가설 가지치기는 가설이 적합하지 않다고 판단할 때 해당 가설을 제거하는 것이다. 다중 가설 매칭 기술은 온라인 매칭에 적합하다.

마지막은 통계기법 기반의 궤적 매칭 방법이다. 대표적으로 히든 마코프 모델(Hidden Markov model)을 이용하는 방법들이 제안된 바 있다[8][12].

[12]에서는 스마트폰으로부터 생성된 궤적 데이터와 같이 낮은 샘플링 비율의 궤적 데이터를 매칭한다. FMM 방법[8]은 히든 마코프 모델을 이용하며 낮은 샘플링 비율로 만들어진 궤적 데이터에 대해서 모델을 적용하고 사전 계산을 결합하여 빠르게 매칭하는 방법이다.

기하학적 방법이나 다중 가설 매칭 방법은 단순 계산을 통해 매칭을 수행하므로 히든 마코프 모델보다 처리 속도는 빠르지만, 정확도는 낮다. 또한, 기하학적 방법이나 다중 가설 매칭 방법은 네비게이션이나 실시간 속도 제어나 경로 시각화와 같은 온라인 처리에 적합한 반면 경로 예측, 경로 분석 등과 같이 오프라인 처리에는 히든 마코프 모델 등의 통계기법이 주로 사용된다[11].

본 논문에서 제안하는 방법은 기본적으로 토폴로지 그룹의 다중 가설 매칭 기술을 기반으로 하며 매칭 상태를 상태 천이 그래프로 표현하여 매칭의 정확도를 향상한다. 또한, 속도 개선을 위해서 그리드 색인 기법을 적용한다. 이를 증명하기 위해서 최근의 히든 마코프 방법인 FMM 알고리즘[8]과 속도와 정확도 측면에서 비교한다. [13]은 이 논문에서 제안하는 방법과 동일한 접근방법을 가진다.

III. Map Matching 방법

이 논문에서 제안하는 매칭 방법의 구조는 그림 1과 같다. 제안하는 매칭 방법은 궤적 데이터를 입력으로 하며 궤적이 매칭된 링크 집합을 출력한다.

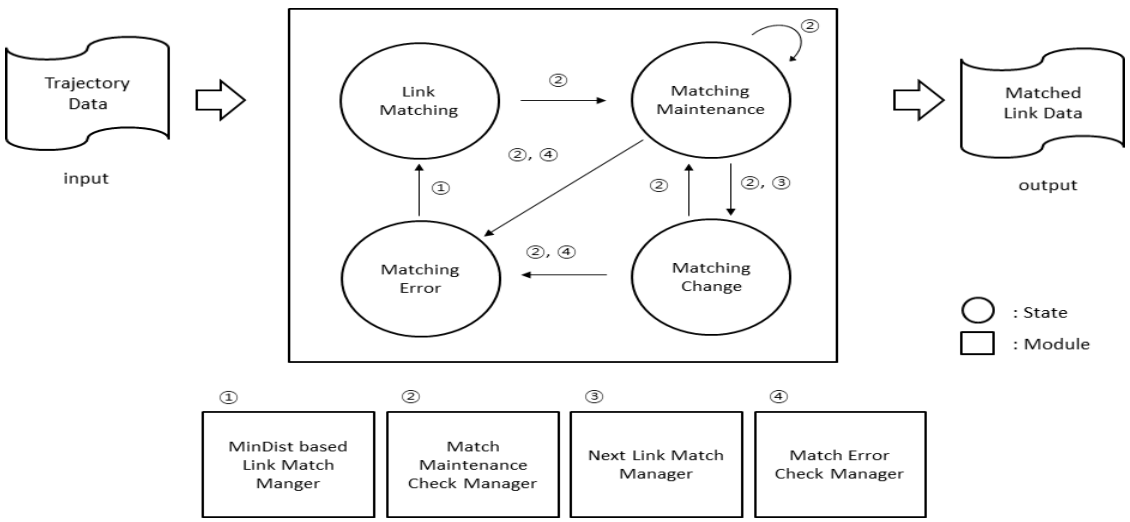


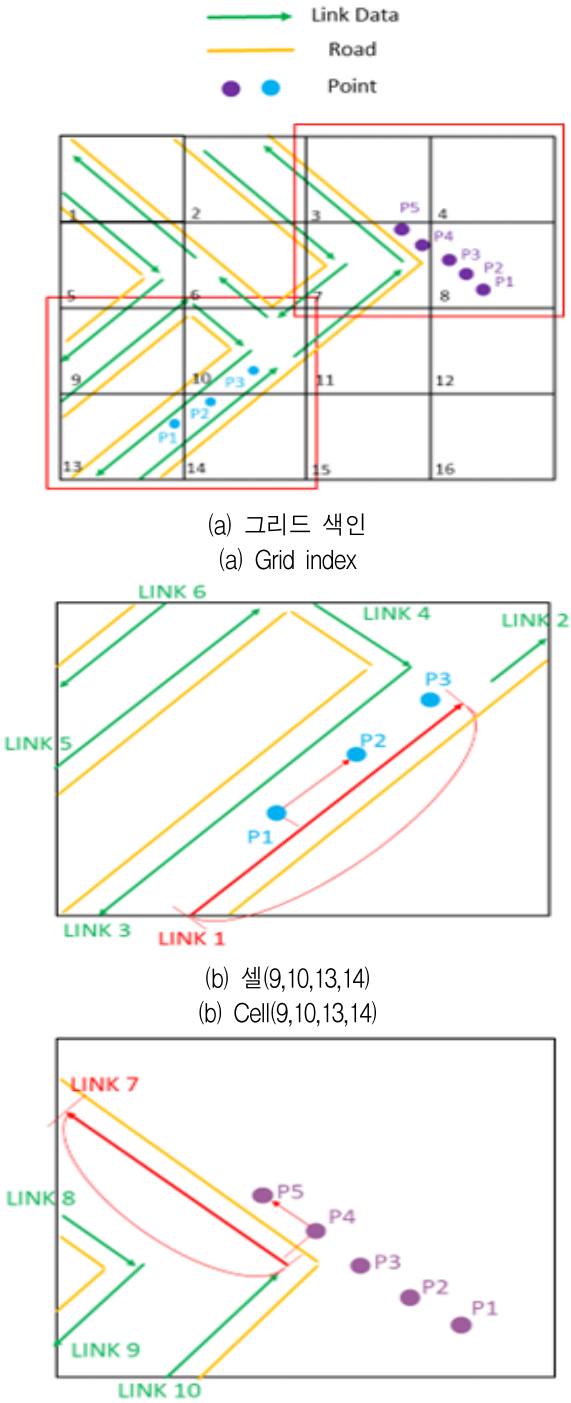
그림 1. 링크 매칭 방법의 구조
Fig. 1. Architecture of link matching method

출력 링크 집합의 각 링크는 매칭된 궤적의 위치 데이터를 포함한다. 제안하는 방법은 그림 1과 같이 링크매칭, 매칭유지, 매칭변경, 매칭오류의 4가지 상태를 가지며 MLM 관리자(MinDist based Link Match Manger), MMC 관리자(Match Maintenance Check Manager), NLM 관리자(Next Link Match Manager), MEC 관리자(Match Error Check Manager)를 통해 천이 된다. 그림에서 각 상태 사이에 화살표에 부여된 번호는 해당 상태 천이를 일으키는 특정 관리자를 의미한다. 예를들어 “Matching Error” 상태에서 “Link Matching” 상태로의 천이는 ① MLM 관리자에 의해 수행된다. n개의 위치 데이터를 갖는 궤적 T1을 (P0, P1, ... , Pn)라고 할 때, MLM 관리자는 P0로부터 시작하여 최초로 특정 링크에 매칭되는 위치 데이터 Pi를 찾는다. 이때 매칭된 링크를 Linkj라고 한다.

그림 2는 MLM 관리자가 궤적의 위치 데이터에 매칭 되는 링크 데이터를 빠르게 찾을 수 있도록 지원하는 그리드 색인 기법을 보여준다. 제안하는 방법에서는 각 셀에 포함되거나 걸쳐져 있는 링크들을 셀의 고유한 번호로 색인한다. 또한, 한번 색인한 데이터를 계속해서 사용함으로 매칭 시간을 단축 시킬 수 있다. 그림에서 (a)는 도로에 그리드 색인이 구축된 상황을 보여준다. 그림에서 (b)와 (c)는 그리드 색인의 9, 10, 13, 14번과 3, 4, 7, 8번 셀을 보다 확대해서 보여준다.

MLM 관리자의 링크 매칭 과정을 설명하기 위해서는 궤적을 구성하는 위치 데이터들과 링크 데이터들 간의 포함, 방향 유사도, 최소거리에 대한 설명이 필요하다. 그림 3에서 이를 설명한다. 그림에서 포함은 (a)와 같이 어떤 위치 데이터 P1이 링크의 시작과 종료지점 범위 안에 존재하는 것이다. P1과 Link1이 직교하는 점을 찾을 때, 이 점이 Link1위에 존재할 경우 포함으로 판단한다. 방향 유사성은 (b)에서 P1의 후속 위치 데이터와 P1을 연결했을 때의 벡터와 링크 벡터의 유사도를 판단한다. 최소거리는 (c)에서처럼 직선과 점사이의 거리를 측정하여 특정 한계값(Threshold) 이하면 매칭으로 판단한다.

그림 2의 (b)을 보면 P1 ~ P3는 어떤 궤적의 연속적인 위치 데이터이다. MLM 관리자는 시작 위치 데이터인 P1에 대해 P1을 포함하는 셀 13과 셀 13의 주변 셀들(9, 10, 14)을 선택한다. 선택된 셀들에 포함된 모든 링크들과 P1의 포함관계를 계산하여 후보 링크들을 선정한다. 후보 링크들은 셀 13과 셀 9의 Link1, Link3, Link5, Link6 이다.

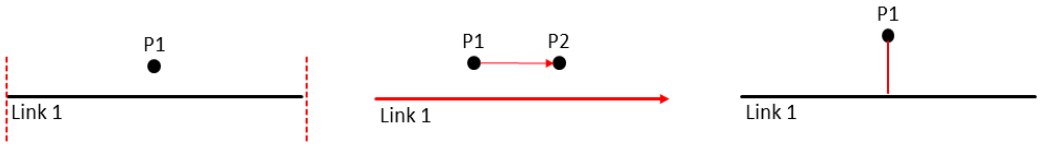


(a) 그리드 색인
(a) Grid index

(b) 셀(9,10,13,14)
(b) Cell(9,10,13,14)

(c) 셀(3,4,7,8)
(c) Cell(3,4,7,8)

그림 2. 링크 데이터 그리드 색인 방법
Fig. 2. Grid indexing method for link data



(a) 포함 (b) 방향 유사도 (c) 최소 거리
(a) Including (b) Heading similarity (c) Minimum distance

그림 3. 포인트와 링크 데이터에 대한 적합성 검사
Fig. 3. Checking the suitability of point and link data

다음으로 후보 링크들(Link1, Link3, Link5, Link6)과 P1과 P1의 후속 위치 데이터인 P2를 이용하여 P1의 방향 유사도를 계산하고 P1의 방향 유사도과 가장 유사한 링크들을 선택한다. 그림에서는 Link1과 Link5가 이에 해당한다. 마지막으로 최종 후보 링크들(Link1, Link5)에 대해 최소거리(Minimum distance)를 계산하여 가장 가까운 링크인 Link1을 매칭 링크로 선택한다.

그런데, 최초 후보링크를 선택하기 위해 P1과 포함관계에 있는 링크들을 탐색할 때 포함관계가 있는 링크가 없을 경우도 있을 수 있다. 그림 2의 (c)가 이를 보여준다. 이 그림에서처럼 P1 ~ P3에 대해서는 포함관계에 있는 링크가 존재하지 않는다. 이 경우 제안하는 방법은 P1부터 순차적으로 포함관계를 계산하고 포함관계가 없을경우 계속해서 후속 위치 데이터로 이동하여 포함관계를 갖는 링크를 찾는다. 이 그림에서는 P4가 이에 해당한다. 이렇게 포함관계가 있는 링크들을 찾게 되면 앞서 기술한 바와 같이 이후의 방향 유사성, 최소거리를 이용해서 매칭되는 링크를 찾는다. 그렇게 링크를 찾게 되면 상태가 “Link Matching”으로 천이된다.

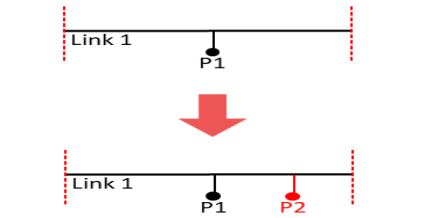
이미 기술한 바와 같이 MLM 관리자 에 의해 Linkj와 매칭된 위치 데이터를 Pi 라 한다. MMC 관리자는 위치데이터 Pi+1에 대해 Linkj와의 매칭 유지를 지속할지 판단한다. 판단결과 매칭유지를 지속 할 경우 상태를 “Matching Maintenance” 상태로 천이하고, 그렇지 않은 경우 “Matching Error” 상태나 “Matching Change” 상태로 천이한다. 매칭유지 판단은 MLM 관리자와 마찬가지로 포함, 방향 유사성, 최소거리를 통해 이루어진다. 그림 4는 MMC 관리자가 상태를 “Matching Maintenance”, “Matching Change”, “Matching Error”로 천이 시키는 예를 보여준다.

그림 4의 (a)는 P1의 후속 위치 데이터인 P2가 Link1에 포함되는 예이다. P2는 MMC 관리자에 의해 Link1과 포함관계, 방향 유사성, 최소거리들을 비교하여 Link1과 매칭을 지속할 것을 판단한다. 이 경우 상태를 “Matching Maintenance”로 천이한다.

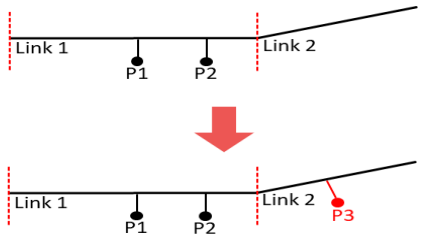
그림 4의 (b)는 P1, P2가 Link1과 “Matching Maintenance” 상태에 있을 때 Link1을 벗어난 P3은 Link2에 매칭 하는 예이다. P3을 입력 받으면 P1, P2가 매칭된 Link1과 포함관계를 통해 링크를 벗어났는지를 판단한다. MMC 관리자를 통해 링크를 벗어났다고 판단되면 다른 링크로 매칭을 변경해야 상황으로 “Matching Change” 상태로 천이한다. “Matching Change” 상태가 되면 현재 링크와 궤적의 위치 데이터 매칭 정보들을 저장한 후, NLM 관리자를 실행하여 P3는 Link2에 매칭 한다.

그림 4의 (c)는 교차로와 같이 두 링크 데이터(Link 1, Link 2)가 직교하는 도로에서 위치 데이터 P1 ~ P3로 구성되는 궤적의 매칭 변경 상황을 보여준다. (c)의 링크 변경 방법은 최초 위치 데이터 P1을 MLM 관리자를 통해 Link 1를 얻는다. 이후 MMC 관리자를 통해 매칭된 Link 1에 끝노드 (To Node)의 위치와 모든 위치 데이터들(P1, P2, P3) 간의 거리를 계산한다. 계산 결과 어떤 위치 데이터의 끝 노드와의 거리가 한계값 이하일 경우 “Matching Change” 상태로 천이하고 NLM 관리자를 실행한다. 거리가 점차 줄어들어 P3의 끝노드와의 거리가 한계 값보다 작아졌다는 것은 끝노드를 벗어났거나, 근접하여 다음 위치 데이터는 현재 위치 데이터의 매칭 되는 링크가 변경돼야 함을 의미한다. (Jeong, 2021)은 (c)에 대해 정상적으로 수행되지 않는다. 이 방법은 포함관계를 우선적으로 적용하므로 (c)에 대해서는 (d)와 유사하여 “Matching Error” 상태로 천이시킨다. 제안하는 방법은 이를 개선하여 정확도를 향상한다.

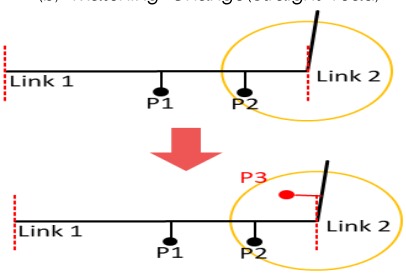
그림 4의 (d)는 위치 데이터 P1이 매칭된 Link1으로부터 후속 위치 데이터 P2가 벗어나는 경우를 보여주는 예이다. P2를 입력 받으면 MMC 관리자로부터 Link1과의 최소거리 비교에서 한계값을 벗어났으므로 매칭 오류를 판단하여 상태를 "Matching Error"로 천이한다. 하지만, MMC 관리자가 "Matching Error" 상태로 판단한경우 일시적인 이상치 발생 여부를 추가로 확인해야 한다. MEC 관리자가 이를 수행한다.



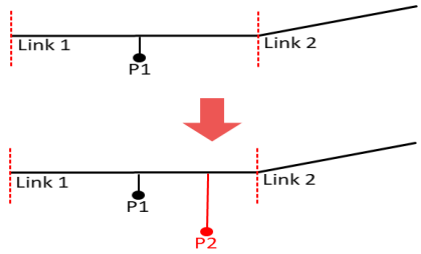
(a) 매칭 유지
(a) Matching Maintenance



(b) 매칭 변경(직선 도로)
(b) Matching Change(straight road)



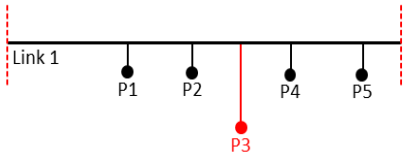
(c) 매칭 변경(수직 도로)
(c) Matching Change(vertical road)



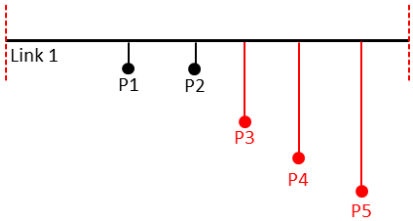
(d) 매칭 오류
(d) Matching Error

그림 4. MMC 관리자의 상태 천이 예
Fig. 4. Example of state transition by MMC manager

그림 5는 이상치로 인한 "Matching Error" 상태와 실제 "Matching Error" 상태에 대한 예를 보여준다. 그림 5의 (a)는 P3가 이상치를 갖는 경우로, MEC 관리자는 이상치가 발생한 이후 후속 위치 데이터 P4, P5를 검사하여 "Matching Error" 상태로 천이하지 않고 일시적인 오류로 판단하여 매칭 상태를 유지하는 경우를 보여준다. 그림 5의 (b)는 P3 ~ P5가 점차 링크와의 거리가 커져서 한계 값을 벗어나는 상황으로 "Matching Error" 상태로 천이해야 하는 경우이다. 이 경우는 보통 도심지에서 고가도로나, 도로들이 겹쳐 있는 상황에서 겹쳐있는 도로들이 서로 멀어질 때 발생한다. 이를 통해 소수의 이상치로 인한 불필요한 상태 천이를 방지한다. MEC 관리자에 의해 "Matching Error" 상태로 판명된 위치 데이터에 대해 다시 MLM 관리자을 수행하여 매칭을 시도한다.



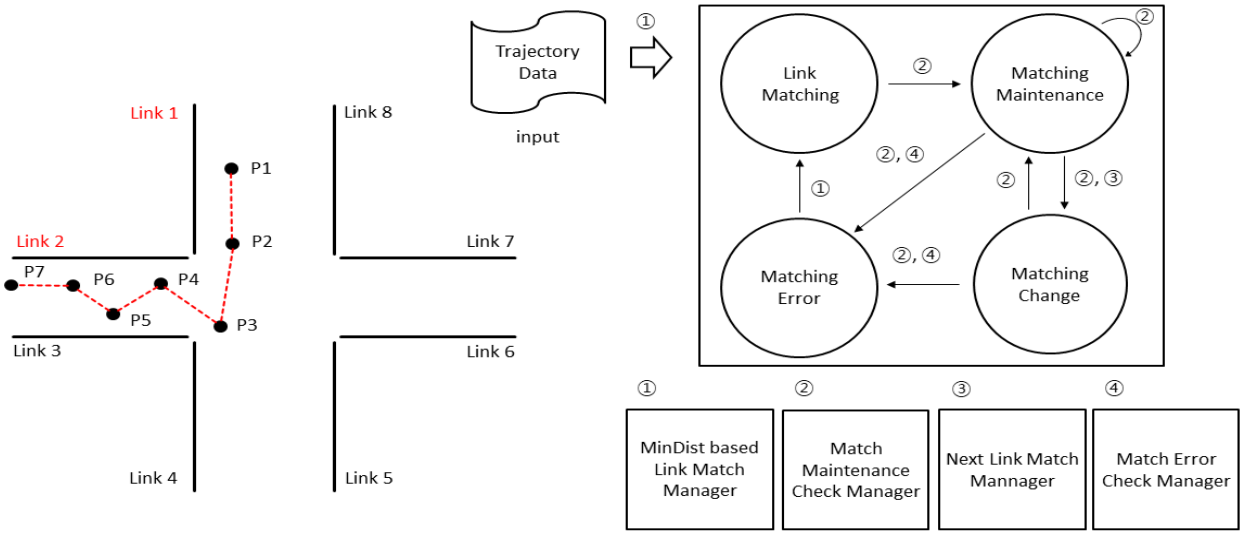
(a) 이상치 정의
(a) Definition of outlier



(b) 매칭 오류 정의
(b) Definition of matching error

그림 5. MEC 관리자
Fig. 5. MEC manager

제안하는 링크 매칭 방법은 궤적의 위치 데이터에 대해 연속적으로 수행한다. 궤적의 모든 위치 데이터에 대해 매칭을 수행하면 매칭 된 링크 집합을 얻는다. 이때 각 링크에는 매칭된 위치 데이터가 포함된다. 그림 6은 위치 데이터 P1 ~ P7로 구성되는 궤적 데이터 T1에 대한 매칭과정을 보여준다. 그림 6의 (a)는 교차로에 Link1에서 Link2로 이어지는 도로상의 궤적 데이터를 보여준다. 그림 6의 (b)는 해당 궤적 데이터에 대한 매칭 과정을 각 관리자의 실행 순서와 상태의 천이 순서를 통해 보여준다.



(a) 매칭 예 (b) 제안 방법 동작과정

(a) Example (b) Proposed method

그림 6. 궤적 데이터 매칭 과정

Fig. 6. Trajectory data matching process

매칭은 T1의 P1 입력과 함께 시작된다. MLM 관리자는 앞서 설명한 방식에 의해 P1에 대해 Link1을 매칭한다. 동시에 상태를 “Link Matching”으로 천이한다. P2가 입력되면 “Link Matching” 상태이므로 MMC 관리자에 의해 Link1과 매칭이 유지가 되면 “Matching Maintenance” 상태로 천이시킨다. P3가 입력되면 “Matching Maintenance” 상태이므로 다시 MMC 관리자가 동작하여 “Matching Maintenance” 상태를 유지할지 판단한다. P3의 경우 Link1의 범위를 벗어나므로 NLM 관리자를 통해 Link1과 연결된 Link2를 찾고 “Matching Change” 상태로 천이한다.

또한 “Matching Change” 상태로 천이 되면서 P1, P2는 Link1의 매칭 위치 데이터로 표시한다. 하지만 P3의 경우 Link2와의 거리가 멀어져 MEC 관리자가 이상치로 판단하고 상태 천이 없이 계속해서 매칭을 진행한다. P4가 입력되면 MMC 관리자에 의해 이 전에 매칭된 Link2와 매칭을 유지하면서 “Matching Maintenance” 상태로 천이 된다. P5, P6, P7은 모두 매칭 유지 상태로 상태 천이 없이 Link2에 매칭 유지되면서 매칭 작업이 종료 된다. 최종적으로 Link1에는 P1, P2와, Link2에는 P3, P4, P5, P6, P7가 매칭 된다.

그림 7은 제안하는 방법의 알고리즘이다. 입력으로 궤적의 위치 데이터들을 받으며 출력으로는 링크에 매칭된 포인트들의 집합이다. 알고리즘은 처음에 MLM 관리자를 통해 궤적의 첫 번째 위치 데이터가 위치한 링크를 찾아 상태를 “Link Matching”로 천이한다.

```
MapMatching ( )
Input: PT(P_0~P_n) // trajectory data
Output: LT(L_0~L_m),L_i (P_0~P_k) // matched links and points
Init Distance, Include, Heading, ELD (End Link Distance)
1. L_j, P_j=MLMManager(PT)
2. for P_j ~P_n in PT
3.  if 15 < Distance(P_i,P_(i+1)) => continue
4.   if Distance(P_i,L_j)
5.    if Include(P_i,L_j)
6.     if Heading(P_i,P_(i+1),L_j)
7.      if Distance(P_i,L_j=End Point)
8.       Distance= value, Include= T or F
9.       Heading= T or F, ELD= value
10.      end if
11.    end if
12.   end if
13.  end if
14.  if (Not Include) | (ELD< 15)
15.   LT+= (L_j,P_(j~i))
16.   L_(j+1)= NLM Manager(L_j,P_i)
17.  else if(Include & Distance > 30) |
          (!Include & Distance > 30)
18.   if MEC Manager(L_j,P_i)
19.    L_(i+1)= MLMManager(P_i)
20.   end if
21. end if
22.end for
```

```
MLM ( )
Input: PT(P_0~P_n) // trajectory data
Output: LT(L_i ),  PT(P_k) // set of matched Links and
point
1. ⟦Per⟧_x cell=(38.763189-32.950424)/(589∗2)
2. ⟦Per⟧_y cell=(131.563393-124.773835)/(647∗2)
3. ⟦grid⟧_x=(P_i.lat-32.950424)/⟦Per⟧_x cell
4. ⟦grid⟧_y=(P_i.lng-124.773835)/⟦Per⟧_y cell
5. selectedCell=(⟦grid⟧_x∗589∗2)+⟦grid⟧_y
6. Cell+=selectedCell~selectedCell+8
7. P_i  ~ P_n  in PT
8.     candidate=0
9.     L_0  ~ L_n  in Cells
10.       if(Include_Check(P_i, L_(o~n) )  &
            Heading_Check(P_i, P_(i+1),L_(o~n) )  &
            Distance_Check(P_i,  L_(o~n) ))
11.          candidate+=L_k
12.       end if
13.    end for
14.    if candidate==1
15.       return candidate
16. end for

NLM ( )
Input: PT(P_i~P_n) // trajectory data
       LT(L_i) // matched current Link
Output: LT(L_i ) // set of matched Links and point
                 // L_i (P_0~P_k)
Init Distance, Include, Heading, End Link Distance(ELD)
1.  Links=next_link(LT)
2.  P_i  ~ P_n  in PT
3.     candidate=0
4.     L_0  ~ L_n  in Cells
5.       if(Include_Check(P_i, L_(o~n) )  &
            Heading_Check(P_i, P_(i+1),L_(o~n) )  &
            Distance_Check(P_i,  L_(o~n) ))
6.          candidate+=L_k
7.       end if
8.     end for
9.   if candidate==1
10.    return candidate
11. end for

MEC ( )
Input: errorCheck
Output: True or False
1.  if errorCheck==3
2.     return True
3.  else
4.     errorCheck+=1
5.     return False
```

그림 7. 맵 매칭 알고리즘
Fig. 7. Algorithm of map matching

이후 위치 데이터들에 대해서 링크간의 방향 유사성, 거리, 포함 여부, 끝 노드와의 거리 계산을 통해 링크와 거리 값(Distance), 포함 유무(Include), 방향 유사(Heading), 끝노드와 위치 데이터의 값(ELD)을 얻는다. 위에서 설명한 MMC 관리자의 동작 과정처럼 Include가 False거나, ELD가 15m보다 작을 때 링크 변경이 발생하여 상태를 “Matching Change”로 천이한 후, NLM 관리자가 동작한다. Include가 True이고 Distance가 30m보다 클 경우나, Include가 False이고 Distanse가 30m보다 클 경우, 매칭 오류로 판단하여 “Matching Error” 상태로 천이 후, MLM 관리자가 동작한다. 이외 경우는 “Matching Maintenance”로 다음 위치 데이터를 비교한다.

Ⅳ. 성능평가

본 논문에서는 제안하는 매칭 방법을 기반으로 실제 궤적데이터에 대한 도로 링크 매칭이 가능한 시스템을 구현하였다. 그림 8에서 본 논문에서 구현한 궤적 매칭 시스템의 구조를 보여준다. Flask 프레임워크[15]를 기반으로 REST API를 이용해서 원격의 클라이언트로부터 궤적데이터를 수신하고 궤적데이터에 매칭되는 링크를 실시간으로 응답할 수 있다. 다수의 클라이언트의 요청에 대해서 비동기 방식으로 동시에 처리하도록 하였다. 구현한 매칭 시스템을 이용하여 제안하는 매칭방법의 매칭 속도 및 매칭 정확도를 평가하였다.

성능평가에 사용된 실험 환경은 표 1과 같다. Ubuntu 운영체제가 설치된 서버에서 실험을 수행하였으며 주기억장치는 32GByte, 저장장치는 NVMe SSD 1TB를 사용하였다. 실험을 위해서 사용한 데이터는 ㈜WeDrive[16] 에서 수집하는 궤적 데이터를 이용하였다.

표 1. 실험 환경
Table 1. Experimental environments

Category	Contents
OS	Ubuntu 20.04.1 LTS
HW	INTEL(R) Core(TM) i7-9700 CPU 3.00GHz, RAM 32G, 1TB NVMe SSD
Dataset	100,000(Wedrive trajectory data)

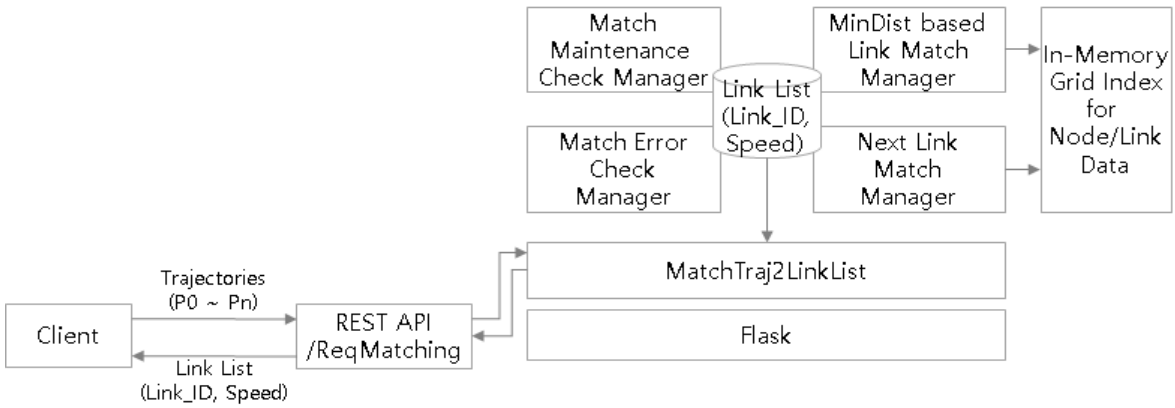


그림 8. 매칭 서비스 구현
Fig. 8. Implementation of matching service

```
accuracy,id,lat,lng,speed,time,types
15.92,91873,37.52687,126.849495,0.0,1607567607048,START
12.893,91874,37.52695,126.84951,0.018970493,1607567610000,NORMAL
10.334,91875,37.52694,126.84954,0.2629434,1607567611000,NORMAL
7.329,91876,37.52693,126.84958,1.4897686,1607567613000,NORMAL
10.318,91877,37.52693,126.849556,1.4897145,1607567613545,NORMAL
8.493,91878,37.52694,126.849556,1.6634067,1607567617000,NORMAL
8.466,91879,37.52694,126.84957,0.72405565,1607567619000,NORMAL
3.863,91880,37.52694,126.84958,0.83795345,1607567621000,NORMAL
3.872,91881,37.526966,126.84957,2.5430417,1607567623000,NORMAL
3.9,91882,37.52701,126.84955,1.739564,1607567625000,NORMAL
3.9,91883,37.527035,126.84951,3.1659758,1607567627000,NORMAL
3.9,91884,37.527096,126.849464,5.5427227,1607567629000,NORMAL
3.9,91885,37.527225,126.849434,7.746535,1607567631000,NORMAL
```

그림 9. 궤적 데이터(WeDrive) 예
Fig. 9. Example of trajectory data(WeDrive)

그림 9는 대한민국에서 WeDrive가 사용자들로부터 수집한 궤적 데이터의 형태이다. 그림과 같이 궤적 데이터는 accuracy, id, lat, lng, speed, time types로 구성되어 있으며, 높은 샘플링 주기로 생성되었다. 이 궤적 데이터의 위치 데이터(lat, lng)를 제안하는 방법의 입력으로 사용한다. 실험에 사용된 궤적 데이터는 총 100,000건이었다. WeDrive 수집 데이터를 이용하여 제안하는 매칭방법과 FMM에서 제안하는 방법간의 매칭 정확도 및 매칭 속도를 비교하였다.

표 2와 표 3은 기존 제안된 FMM(Yang, 2018)에서 제안하는 방법과 본 논문에서 제안하는 방법의 매칭 정확도와 매칭 속도를 단일 쓰레드에서 실험을 수행하여 측정한 결과이다.

표 2는 2,525개의 궤적 데이터에 대해서 궤적 매칭을 수행하고, 하나의 궤적에 대한 매칭 평균 시간을 보여준다. 표 2에서처럼 본 논문에서 제안하는 방법이 실행속도 측면에서는 FMM에 비해 약 20배 정도 성능이 향상된 것을 볼 수 있다. FMM은 모든 포인트에 대해서 학습을 통해 매칭 과정을 수행하므로 처리 속도가 느리다. 본 논문에서 제안하는 방법은 그리드 색인 기법 및 상태에 따라 계산법을 다르게 적용하는 등의 접근방법을 이용하여 속도를 개선한 결과로 보인다.

표 3은 제안하는 방법과 기존 방법들간의 매칭 정확도를 실제 값(Ground truth)과 매칭된 값(Matched result)의 중복 비율의 평균 측면에서 비교한 것으로, 아래와 같이 정상 매칭된 경우(True positive), 잘못 매칭된 경우(False positive)와 실제 값을 이용하여 매칭 정확도로 측정한다. GT는 실제 값, MR은 매칭된 값(정상 매칭 + 잘못 매칭), n은 궤적 개수이다.

표 2. 평균 매칭 수행 시간
Table 2. Average time of matching execution

Method	Average execution time per trajectory
FMM[8]	0.409 sec
Proposed method	0.021 sec

표 3. 매칭 정확도
Table 3. Matching accuracy

Method	TF	FP	ACC
FMM[8]	140	65	69.7%
Proposed method	135	3	88.3%

$$Acc = \frac{1}{n}\sum_{i}^{n}\frac{|GT[i] \cap MR[i]|}{|GT[i] \cup MR[i]|} \quad (1)$$

실험에 사용된 궤적 데이터가 정상적으로 매칭된다면 총 149개의 링크와 매칭이 되어야 한다. 그러나, 표 3에서와 같이 본 논문에서 제안하는 방법은 135개의 링크와 정상 매칭되었고, FMM은 140개 링크와 정상 매칭이 되었다. 본 논문에서 제안하는 방법의 경우 잘못 매칭된 링크의 수가 2개로 매우 작은 반면 FMM은 65개로 매우 많았다.

기존 방법은 모든 포인트를 임계 값과 같은 기준을 통해 매칭 하는 것이 아닌 Hidden Markov 방법 기반으로 학습하여 매칭을 수행하기 때문에 잘못 매칭되는 경우가 많다. 잘못 매칭되는 경우가 크게 2가지로 첫 번째는 도로들이 겹쳐 있는 경우이다. 제안하는 방법은 이후 위치 데이터를 고려하여 링크 변경시에 매칭을 수행하지만, 기존 방법은 현재 위치 데이터와 더 확률이 높은 링크에 매칭시킨다. 특히, 도심지역의 고가도로와 같이 도로들이 겹쳐 있는 경우 잘못 매칭되는 경우가 많다.

두 번째는 링크가 없는 부분에서는 매칭이 되지 않아야 하는데, 매칭을 수행하여 잘못 매칭되는 경우가 많다. 그러므로 기존 방법은 정상 매칭이 더 많았지만, 잘못 매칭이 많아 매칭 정확도가 떨어진다. 매칭은 궤적 데이터를 도로상 네트워크에 표현하는 것으로 잘못 매칭되는 경우를 줄이는 것도 중요한 과제이다.

표 4는 실험 결과를 각 성능지표로 보여주고 있다. 궤적 데이터를 도로 네트워크에 매칭 방법은 잘못 매칭보다 정상 매칭되는 확률이 더욱 중요하여 precision 값이 중요하다. 궤적 데이터를 도로 네트워크의 매칭되면 매칭된 링크는 궤적 데이터의 정보들이 매칭 한다. 하지만, 잘못 매칭되면 전혀 다른 데이터가 링크 데이터에 매칭되므로 향후 도로 네트워크를 감시하거나, 활용할 때 문제가 될 수 있다.

제안하는 방법의 문제점으로, 제안하는 방법은 실시간으로 빠르게 매칭을 수행하는데 최적화 되어 있지만, 낮은 주기로 샘플링된 궤적 데이터에 대해서는 매칭이 어렵다. 또한, 매칭 방법들의 고질적인 문제점으로 고가도로와 같이 도로가 겹치는 곳에서 매칭 정확도가 떨어진다.

표 4. 매칭 성능(정확도, 정밀도, 재현율, F1 점수)
Table 4. Matching performance(Accuracy, precision, recall, F1-score)

Method	Accuracy	precision	recall	F1-score
FMM[8]	68%	68%	100%	80%
Proposed method	90%	97%	92%	94%

V. 결 론

최근 궤적 데이터가 증가하면서 이를 활용한 연구들이 많아졌다. 교통 분야에서 도로 속도와 교통사고 예측을 궤적 데이터를 도로 네트워크에 매칭하는 기술이 필요하다. 논문에서는 상태 천이 모델과 그리드 색인 방법을 이용하는 궤적 데이터에 대한 도로 네트워크 매칭 방법을 제안하였다. 제안하는 매칭 방법은 온라인 매칭이 가능할 정도로 빠른 속도로 매칭을 하면서도 기존 오프라인 매칭 방법보다 정확도를 높인다. 속도를 향상하기 위해 그리드 색인을 이용한 후보 링크 선택 방법과, 상태에 따른 불필요한 계산량 감소하는 방법을 적용하였다. 또한, 정확도를 향상하기 위해서 매칭 상태를 4개로 구분하여 지속적으로 상태 천이를 수행하는 방법을 제안하였다.

마지막으로, 이 논문에서는 제안하는 방법을 구현하고 기존에 제안된 FMM 매칭 방법과 비교실험을 수행하였다. 실험 결과 속도는 약 20배 향상되었고, 매칭 정확도 측면에서는 TP와 FP를 통해 매칭 정확도를 비교했을 때, 더 우수한 성능을 보였다. 향후 제안하는 방법을 통해 실시간 교통상황이나, 궤적 데이터를 이용하는 관련된 연구들에 전처리 과정이 될 것이다.

References

[1] X. Kong, M. Li, K. Ma, K. Tian, M. Wang, Z. Ning, and F. Xia, "Big trajectory data: A survey of applications and services", IEEE Access, Vol. 6, pp. 58295-58306, Oct. 2018. https://doi.org/10.1109/ACCESS.2018.2873779.

[2] Z. Feng and Y. Zhu, "A survey on trajectory data

mining: Techniques and applications", IEEE Access, Vol. 4, pp. 2056-2067, Apr. 2016. https://doi.org/10.1109/ACCESS.2016.2553681.

[3] T. Bogaerts, A. D. Masegosa, J. S. Angarita-Zapata, E. Onieva, and P. Hellinckx, "A graph CNN-LSTM neural network for short and long-term traffic forecasting based on trajectory data", Transportation Research Part C: Emerging Technologies, Vol. 112, pp. 62-77, Mar. 2020. https://doi.org/10.1016/j.trc.2020.01.010.

[4] X. Qian, T. Lei, J. Xue, Z. Lei, and S. V. Ukkusuri, "Impact of transportation network companies on urban congestion: Evidence from large-scale trajectory data", Sustainable Cities and Society, Vol. 55, pp. 1-12, Apr. 2020. https://doi.org/10.1016/j.scs.2020.102053.

[5] S. Sun, J. Chen, and J. Sun, "Traffic congestion prediction based on GPS trajectory data", International Journal of Distributed Sensor Networks, Vol. 15, No. 5, pp. 1-18, May 2019. https://doi.org/10.1177/155014771984744.

[6] National Transport Information Center, https://www.its.go.kr/nodelink/intro [accessed: Dec. 20, 2022]

[7] S. D. Park, N. Ariunerden, D. S. Go, and S. I. Song, "Grid based Spatio-Textual Indexing Method using Bloom Filter", Journal of Platform Technology, Vol. 7, No. 4, pp. 42-48, Dec. 2019.

[8] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden Markov model with precomputation", International Journal of Geographical Information Science, Vol. 32, No. 3, pp. 547-570, Nov. 2017. https://doi.org/10.1080/13658816.2017.1400548.

[9] M. Kubicka, A. Cela, H. Mounier, and S. I. Niculescu, "Comparative study and application-oriented classification of vehicular map-matching methods", IEEE Intelligent Transportation Systems Magazine, Vol. 10, No. 2, pp. 150-166, Apr. 2018. https://doi.org/10.1109/ MITS.2018.2806630.

[10] H. Alt and L. J. Guibas, "Discrete geometric shapes: Matching, interpolation, and approximation", Handbook of computational geometry, pp. 121-153, https://doi.org/10.1016/B978-044482537-7/50004-8.

[11] P. Chao, Y. Xu, W. Hua, and X. Zhou, "A Survey on Map-Matching Algorithms", Australasian Database Conference 2020: Databases Theory and Applications, pp. 121–133, Jan. 2000. https://doi.org/10.1007/978-3-030-39469-1_10.

[12] A. Luo, S. Chen, and B. Xv, "Enhanced map-matching algorithm with a hidden Markov model for mobile phone positioning", Vol. 6, No. 11, Oct. 2017. https://doi.org/10.3390/ijgi6110327.

[13] S. M. Jeong, G. D. Lee, and S. I. Song, "Road Network Matching Method for Trajectory Data Using Grid Index and State Transition Model", Proc. of the Korean Society of Information Sciences Conference, pp. 229 – 231, Jun. 2021.

[14] W. Cho and E. Choi, "A GPS trajectory map-matching mechanism with DTG big data on the HBase system", Proc. of the 2015 International Conference on Big Data Applications and Services, pp. 22–29, Oct. 2015. https://doi.org/10.1145/2837060.2837062.

[15] Flask's Documentation Website, https://flask. pallets projects.com/en/2.0.x [accessed: Dec. 20, 2022]

[16] WeDrive Website, https://www.wedrive.kr/#/data [accessed: Dec. 20, 2022]

저자소개

정 성 모 (Seongmo Jeong)

2020년 8월 : 한국교통대학교 컴퓨터공학과(공학사)
2022년 8월 : 한국교통대학교 컴퓨터공학과(공학석사)
관심분야 : 내용기반 영상검색, ML, DL, GNN

송 석 일 (Seokil Song)

1998년 2월 : 충북대학교 정보통신공학과(공학사)
2000년 2월 : 충북대학교 정보통신공학과(공학석사)
2003년 2월 : 충북대학교 정보통신공학과(공학박사)
2003년 7월 ~ 현재 : 한국교통대학교 컴퓨터공학과 교수
관심분야 : 데이터베이스, 센서 네트워크, 스토리지 시스템