# GPU를 이용한 예측 정지 구간 생성 알고리즘

이형석, 여은지, 임효상
연세대학교 원주캠퍼스 컴퓨터정보통신공학부
e-mail :{hyungseoklee, ejyeo, hyosang}@yonsei.ac.kr

# A GPU Accelerated Algorithm for Predicting Stop Intervals

Hyungseok Lee, Eunji Yeo, Hyo-Sang Lim
Computer and Telecommunications Engineering Division, Yonsei University

**요 약**

최근 위치기반서비스에 관심이 집중되면서 GPS 궤적에 관심 지점(POI: Point of Interest) 정보를 결합한 시맨틱 궤적(Semantic Trajectory)이 주목 받고 있다. 기존 연구에서는 GPS 궤적으로부터 속력을 계산하여 사용자가 정지했을 만한 예측 정지 구간(PSI: Predictive Stop Interval)과 실제로 방문했을 것이라 예상되는 POI 를 선정하여 시맨틱 궤적을 생성하였다. 그러나 CPU 에서는 대용량의 GPS 궤적에 대해서 PSI 를 구할 시 많은 연산 때문에 시간이 오래 걸리는 문제가 있다. 이에 본 논문에서는 GPU 의 병렬성을 이용하여 PSI 를 생성하는 알고리즘을 제안한다. 제안하는 GPU 를 이용한 PSI 생성 알고리즘은 기존의 CPU 를 사용한 PSI 알고리즘보다 최대 5 배 이상 속도 향상이 있으며, PSI 의 개수가 많을수록 성능상의 이득이 더 큰 장점을 가지고 있다.

## 1. 서론

최근 들어 스마트폰을 비롯한 모바일 기기의 사용이 확산되면서 모바일 기기의 센서 데이터를 통한 사용자의 위치정보 수집이 용이하게 되었고, 이를 이용하여 사용자의 이동 패턴 인식 및 위치 내역 추적이 가능하게 되었다. 이러한 환경으로 인해 위치기반서비스(Location Based Service)가 발달하게 되었고, 다양한 서비스를 제공하기 위해서 GPS 위성으로부터 수신 받은 사용자의 물리적인 GPS 궤적을 분석하여 내포된 의미나 새로운 정보를 발견하려는 연구들이 진행되어 왔다[1,2]. 시맨틱 궤적(Semantic Trajectory)이란 GPS 궤적(GPS Trajectory)에 의미정보를 더하여 생성한 궤적을 의미한다[3]. 의미정보로는 GPS 궤적에서 이동 객체가 정지했던 구간을 식별하는 어노테이션(annotation)정보, 도로망(road network)과의 연계성, 이동 수단, 머물렀던 관심 지점(POI: Point of Interest)의 업종정보(category)등이 있다. 이렇게 생성된 시맨틱 궤적을 분석하면 단순히 물리적인 위치를 나타내는 GPS 궤적으로부터는 알지 못했던 다양한 정보를 발견할 수 있다.

GPS 궤적과 POI 정보를 이용해서 시맨틱 궤적을 생성하려면 사용자가 방문했던 POI 를 추측할 수 있는 방법이 필요하다. 기존의 연구[4]에서는 GPS 궤적으로부터 속력을 계산하고, 계산된 속력으로 GPS 궤적에서 사용자가 정지했을 만한 예측 정지 구간(PSI: Predictive Stop Interval)을 판별하여 사용자가 실제로 방문했을 것이라 예상되는 POI 를 선정한다. 그러나 이 방법은 대용량의 GPS 궤적에 대해서 PSI 를 구할 시 많은 연산을 수행해야 하므로 시간이 오래 걸린다.

본 논문에서는 이러한 기존 연구의 PSI 생성시 발생하는 성능 문제를 GPU 의 병렬성을 이용하여 해결하는 알고리즘을 제시한다. GPU 는 대량의 데이터를 병렬적으로 연산하기에 적합한 구조를 갖고 있다. 즉, CPU 보다 더 많은 코어(core)를 갖고 있어 이를 병렬적으로 수행하여 더 높은 처리량을 얻을 수 있다. 본 논문의 목적은 GPU 의 병렬성을 이용하여 대용량의 GPS 궤적에 대해 PSI 생성시간을 단축시키는 것이다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존 연구에서의 PSI 생성 알고리즘과 GPU 에 대해서 서술한다. 3 장에서는 제안하는 기법인 GPU 를 이용한 PSI 생성 알고리즘에 대해서 기술하며 4 장에서는 실험을 통해 CPU 와 GPU 에서의 PSI 생성의 성능을 비교한다. 마지막으로 5 장에서는 결론을 제시한다.

## 2. 관련 내용 소개

본 장은 기존 연구[4]에서의 CPU 를 사용하여 PSI 를 생성하는 알고리즘과 GPU 에 대한 개요를 설명한다.

### 2.1 기존의 CPU 를 사용한 PSI 생성 알고리즘[4]

사용자의 이동경로를 나타내는 GPS 궤적은 시간 순서를 갖는 각 지점들의 시퀀스이다. GPS 궤적 즉 $GT$는 $GT = \{ P_1, \cdots, P_i, \cdots, P_n \}$로 표현된다. 궤적의 한 지점(position) $P_i$는 $P_i = (id, (x, y), t)$ 로 나타내며, $id$는 지점의 식별 정보이고, $(x, y)$는 각 지점의 좌표를 나타낸다. 마지막으로 $t$ 는 GPS 정보를 수신 받은 시점을 나타낸다.

· 이 논문은 2014 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 일반연구자지원사업 지원을 받아 수행된 것임(2012R1A1A1042875).

예측 정지 구간 즉 PSI 는 *GT* 에서 사용자가 정지했을 것이라고 예측되는 구간들이며 이는 *GT* 의 서브시퀀스이다. 직관적으로 생각했을 때, 사용자의 속력이 0 이면 사용자가 정지했다고 볼 수 있다. 하지만 GPS 의 수신 특성상 실제 사용자가 가만히 멈춰있더라도 속력이 0 이 되는 경우는 많지 않다[5]. [4]에서는 어떠한 구간이 사람이 걷는 속력(1.3m/s)보다 낮은 속력일 때 그 구간을 정지한 구간이라 간주한다. 따라서 PSI 를 생성시 *GT* 의 $P_i$ 에서 $P_{i+l}$ 까지의 평균속력이 1.3m/s · $a$ 이하인 구간을 PSI 구간으로 선정하게 된다. 여기서 α 는 $0< a \leq 1$ 인 실수로 응용 환경에 따라 걷는 속도를 다르게 조정하기 위한 목적으로 사용되었다. 다음의 식(1)을 통해 임의의 $k$ 번째 PSI 를 구할 수 있다. 임의의 $k$ 번째 PSI 는 $PSI_k = (P_i, P_{i+1}, \cdots, P_{i+l})$ 로 표현되고 *GT* 에서 생성된 $PSI_k$ 의 집합은 $PSIset = \{ PSI_1, \cdots, PSI_m \}$ 으로 표현된다.

$$\frac{\sqrt{(P_{i+l}\cdot x - P_i \cdot x)^2 + (P_{i+l}\cdot y - P_i \cdot y)^2}}{(P_{i+l}\cdot t - P_i \cdot t)} \leq 1.3m/s \cdot \alpha \quad (1)$$

2.2 GPU

GPU 는 그래픽 연산을 수행하기 위한 연산장치이다[6]. 기존에는 GPU 가 그래픽을 위한 목적으로만 이용되었으나 최근 NVIDIA 사의 CUDA(Compute Unified Device Architecture)와 같이 GPU 를 이용하여 범용적인 연산을 수행할 수 있도록 하는 라이브러리들이 제공되고 있다. 또한 GPU 는 CPU 보다 많은 코어, 공유메모리, 그리고 레지스터들로 구성되어 있다. GPU 의 수많은 코어에서 각각의 쓰레드(thread)들이 동시에 연산을 수행하기 때문에 데이터를 병렬적으로 처리하여 CPU 보다 성능을 높일 수 있다.

CUDA 프로그램은 호스트 코드(host code)와 디바이스 코드(device code)로 구성되어 있다. 호스트 코드는 CPU 에서 실행되며 병렬성이 필요 없는 부분을 구현할 때 사용된다. 그리고 디바이스 코드는 GPU 에서 실행되며 병렬성을 필요로 하는 부분을 구현할 때 사용된다. 디바이스 코드는 커널이라 불리는 디바이스 함수로 구현되는데, CUDA 프로그래밍은 CPU 의 호스트 코드에서 커널 함수를 호출하여 GPU 에서 커널 함수를 수행하는 방식이다. Off-chip 인 GPU 를 이용하기 위해서는 CPU 의 메모리를 GPU 의 메모리로 복사한 뒤에 커널 함수를 호출하여 GPU 에서 커널 함수를 수행해야 한다.

## 3. GPU 를 이용한 PSI 생성 알고리즘

본 장은 제안하는 기법인 GPU 를 이용한 PSI 생성 알고리즘에 대해 설명한다. 제 3.1 절에서는 제안하는 알고리즘의 개요로 병렬처리의 내용을 설명하고, 제 3.2 절에서는 제안하는 알고리즘의 전체적인 흐름을 설명한다.

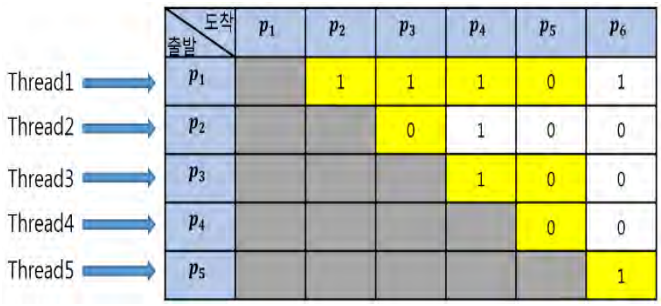
3.1 개요

기존 CPU 에서 PSI 를 생성하는 방법은 *GT* 의 첫 출발지점부터 마지막 도착지점까지 식(1)을 순차적으로 수행하여 식(1)이 만족되는 구간들을 PSI 로 생성하는 방법이다. 먼저 출발지점을 고정시킨 뒤에 순차적으로 도착지점을 증가시켜가면서 식(1)이 만족하지 않을 때까지 계산을 수행한다. 그 뒤에 식(1)이 만족되지 않은 지점을 새로운 출발지점으로 고정시킨 뒤에 마지막 도착지점까지 계산을 반복적으로 수행한다. 예를 들어, 그림 1 에서 테이블 안의 각 cell 의 값은 출발지점($P_i$)와 도착지점($P_{i+l}$)에 대해서 식(1)을 수행한 결과이다. 값이 1 인 cell 은 두 지점 간의 식(1)이 만족됨을 나타내고, 값이 0 인 cell 은 두 지점 간의 식(1)이 만족되지 않음을 나타낸다. 그림 1 과 같은 상황에서 CPU 에서 PSI 를 생성하기 위해서는 노란색 영역의 계산을 순차적으로 수행한다. 그리고 PSI 생성 조건에 알맞은 구간 즉, 출발지점부터 연속적으로 식(1)이 만족되는 구간을 PSI 로 생성한다. 따라서 그림 1 에서는 노란색 영역 중 $P_1 \sim P_4$ 와 $P_5 \sim P_6$ 구간이 PSI 로 생성된다.

| 출발 \ 도착 | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ |
|---|---|---|---|---|---|---|
| $p_1$ | | 1 | 1 | 1 | 0 | 1 |
| $p_2$ | | | 0 | 1 | 0 | 0 |
| $p_3$ | | | | 1 | 0 | 0 |
| $p_4$ | | | | | 0 | 0 |
| $p_5$ | | | | | | 1 |

(그림 1) CPU 에서의 PSI 생성 테이블

CPU 를 이용한 PSI 생성 방법은 한 번에 하나의 cell 의 계산을 순차적으로 수행해야 하지만 제안하는 방법은 GPU 의 병렬성을 이용하여 한 번에 여러 개의 cell 의 계산을 병렬적으로 수행한다. 예를 들어, 그림 2 와 같이 출발지점마다 하나의 GPU 쓰레드가 할당이 되어 해당 cell 이 식(1)을 만족하는지 계산을 병렬적으로 수행하게 된다. 각각의 쓰레드는 식(1)을 만족하지 않는 cell 을 만날 때까지 계속해서 cell 의 계산을 수행한다. 그림 2 과 같은 상황에서 GPU 에서 PSI 를 생성하기 위해서는 노란색 영역의 계산을 병렬적으로 수행하게 된다. 계산을 수행한 뒤에는 CPU 에서의 PSI 생성조건과 같이 출발지점부터 연속적으로 식(1)이 만족되는 구간을 PSI 로 생성한다. 이때에 GPU 에서는 CPU 와는 다르게 모든 출발지점에 대하여 계산을 수행하므로 PSI 들끼리 겹치는 영역이 생길 수 있다. 이러한 겹치는 PSI 들은 GPU 에서의 수행이 모두 끝난 뒤에 CPU 에서 출발지점이 가장 빠른 PSI 를 제외하고 나머지는 제거하는 후처리 과정을 거쳐서 최종 PSI 를 얻는다.

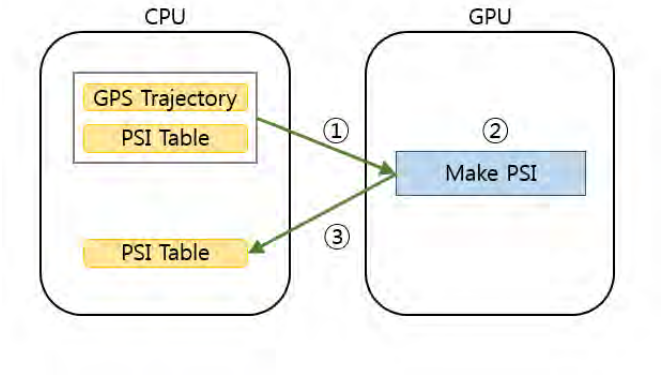



| 출발 \ 도착 | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ |
|---|---|---|---|---|---|---|
| $p_1$ | | 1 | 1 | 1 | 0 | 1 |
| $p_2$ | | | 0 | 1 | 0 | 0 |
| $p_3$ | | | | 1 | 0 | 0 |
| $p_4$ | | | | | 0 | 0 |
| $p_5$ | | | | | | 1 |

(그림 2) GPU 에서의 PSI 생성 테이블

### 3.2 GPU 에서의 PSI 생성 알고리즘

제안하는 GPU 를 이용한 PSI 생성 알고리즘의 전체 흐름은 그림 3 과 같이 총 3 단계로 이루어 진다. 첫 번째 단계(그림 3 의 ①)에서는 GPS 궤적 데이터 즉 *GT* 와 결과 PSI 를 저장 할 *PSI Table* 을 GPU 의 메모리 공간에 복사하고 GPU 커널 함수를 호출한다. 두 번째 단계(그림 3 의 ②)는 실제 GPU 의 병렬성을 이용하는 부분이다. 이 단계에서는 GPU 커널 함수를 수행하여 *PSI Table* 에 *GT* 에서 생성된 PSI 들을 저장한다. 세 번째 단계(그림 3 의 ③)에서는 PSI 가 저장된 *PSI Table* 을 CPU 메모리 공간으로 복사해온다. 그 후 *PSI Table* 에서 겹치는 PSI 들을 제거하는 후처리 과정을 수행한다.




(그림 3) 제안하는 기법의 전체 흐름

GPU 에서의 PSI 생성 알고리즘은 GPS 의 궤적 데이터인 *GT* 와 결과 PSI 를 저장할 *PSI Table* 을 입력으로 받아서 GPU 커널 함수를 수행하여 얻은 PSI 가 저장된 *PSI Table* 을 출력한다. PSI 를 생성할 때에 GPU 의 병렬성을 이용하기 위해서 각각의 출발지점에서의 식(1)을 계산하는 과정은 각각의 GPU 쓰레드에서 수행되어 병렬적으로 PSI 를 생성한다. 즉, *GT* 의 마지막 지점을 제외한 모든 지점을 출발지점으로 하는 GPU 쓰레드를 할당하여 PSI 생성 조건을 계산하는 작업을 수행한다. 먼저 각 GPU 쓰레드는 자신에게 할당된 지점을 출발지점($P_i$)으로 하고 바로 다음 지점($P_{i+1}$)을 도착지점으로 하여 해당 지점들이 식(1)을 만족하는지 계산한다. 식(1)을 만족한다면 현재 도착지점의 다음 지점($P_{i+2}$)을 도착지점으로 하여 식(1)의 만족여부를 판단하는 계산을 수행한다. 이렇게 순차적으로 기존 도착지점의 다음 지점을 도착지점으로 하여 식(1)의 만족여부를 판단하는 계산을 계속해서 수행한다. 이러한 계산을 수행하던 중에 식(1)이 처음으로 만족되지 않는 도착지점( $P_{i+l}$) 이 발생하면 해당 GPU 쓰레드는 식(1)의 만족여부를 판단하는 계산을 정지한다. GPU 에서의 PSI 생성조건은 CPU 에서의 PSI 생성조건과 같이 출발지점부터 연속적으로 식(1)이 만족되는 구간이므로 각각의 GPU 쓰레드가 계산을 중지한 도착지점 전까지의 구간을 PSI 로 생성한다. 예를 들어, 그림 2 에서 $P_1$ 을 출발지점으로 할당받은 Thread1 은 도착지점 $P_2$~$P_4$ 구간은 식(1)이 만족되어 순차적으로 연산을 수행하고 도착지점 $P_5$ 와는 식(1)이 처음으로 만족되지 않아서 $P_1$~$P_4$ 구간을 PSI 로 선정하고 작업을 중지한다. 즉, 각각의 GPU 쓰레드는 자신이 계산을 중지한 도착지점의 바로 전 지점의 *id* 를 *PSI Table* 에 저장하여 PSI 를 생성한다. GPU 에서 생성된 PSI 는 GT 의 마지막 지점을 제외한 모든 지점을 출발지점으로 하여 생성되므로 CPU 에서 생성된 PSI 와는 다르게 겹치는 PSI 가 존재한다. 이러한 겹치는 PSI 는 CPU 에서 후처리 과정을 통해 하나의 PSI 로 출발지점이 가장 빠른 PSI 를 제외한 나머지 PSI 를 전부 제거하는 과정을 거친다.

## 4. 실험

본 장은 제안하는 GPU 를 이용한 PSI 생성 알고리즘의 성능 평가 결과를 제시한다. 제 4.1 절에서는 실험 데이터와 실험 환경에 대해서 설명하고, 제 4.2 절에서는 실험 결과를 설명한다.

### 4.1 실험 데이터 및 실험 환경

실험 데이터로는 임의로 생성된 GPS 궤적 데이터를 사용하였다. GPS 궤적 데이터 생성 방법은 *GT* 의 사이즈($n$)와 PSI 의 개수($l$)를 정해놓고 PSI 안 지점의 개수를 *GT* 의 사이즈와 PSI 의 개수를 고려하여 모든 PSI 안의 지점의 개수를 동일하게 생성하였다. 실험 환경은 표 1 와 같다.

<표 1> 실험 환경

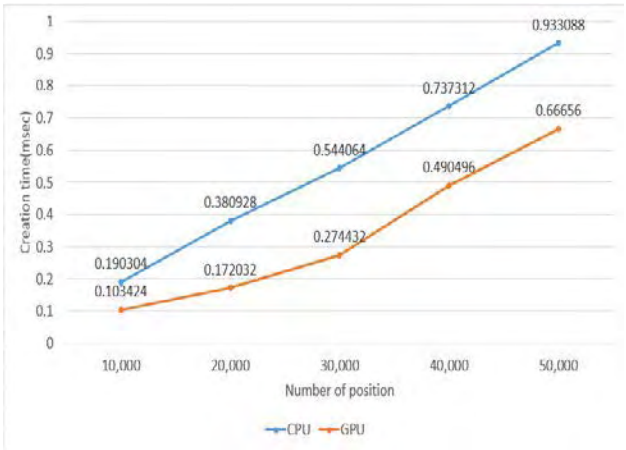
| | |
|---|---|
| CPU | Intel®Core™ i7-4790K CPU 4.00GHZ |
| Memory | 8GB |
| OS | Ubuntu 14.04.1 LTS |
| GPU | NVIDIA GeForce GTX 980 |

### 4.2 실험 결과

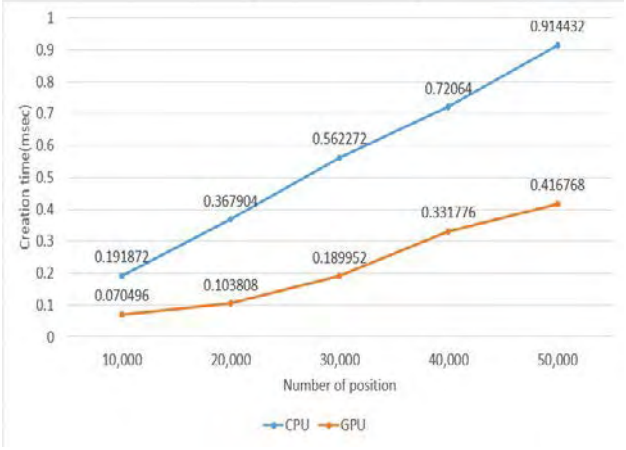
실험에서는 PSI 의 개수를 고정시켜 놓고 GT 의 사이즈를 변경하며 CPU 와 GPU 에서의 수행시간을 비교하여 성능차이를 보인다. 그림 4, 그림 5 와 그림 6 은 각각 PSI 의 개수를 100, 200, 300 으로 고정시켜 놓고 GT 의 사이즈를 10,000 에서 50,000 까지 증가시켜 실험한 결과들이다. 실험 결과들을 보면 GPU 가 CPU

보다 수행 시간이 빠른 것 확인 할 수 있다. 그림 4 를 보면 CPU 보다 GPU 가 최소 1.5 배에서 최대 2.2 배정도 수행 시간이 빠르고 그림 5 를 보면 최소 2.1 배에서 최대 3.5 배정도 GPU 의 수행시간이 CPU 의 수행시간 보다 빠르다. 마지막으로 그림 6 을 보면 GPU 의 수행시간이 CPU 의 수행시간보다 최소 3.0 배에서 5.7 배정도 빠르다. 이를 통해 계산해야 할 *GT* 의 사이즈가 커질수록 성능이 더 향상되는 것을 알 수 있다. 이는 *GT* 크기가 클수록 GPU 를 사용할 때 더 많은 쓰레드를 사용할 수 있어서 병렬성이 더 높아지기 때문으로 분석된다.

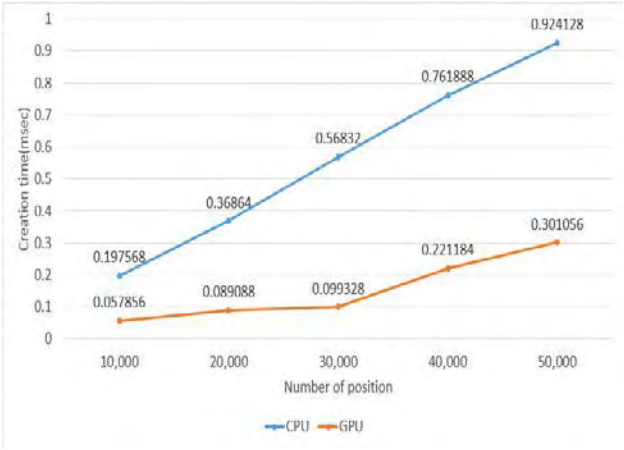
그림 4, 그림 5 와 그림 6 을 보면 CPU 에서의 수행시간은 PSI 의 개수를 증가시켜도 각 *GT* 의 사이즈마다 비슷한 시간을 가진다. 이러한 이유는 CPU 에서의 연산은 GT 에 저장된 처음 지점($P_0$)부터 마지막 지점까지($P_n$) 순차적으로 식(1)을 수행하기 때문이다. 하지만 GPU 에서의 수행시간은 PSI 의 개수가 많아질수록 수행시간이 빨라지는 경향을 볼 수 있다. CPU 와는 달리 GPU 에서는 병렬적으로 연산을 수행하기 때문이다. 각 지점마다 쓰레드가 할당되어 각 쓰레드마다 PSI 를 생성하면 작업을 중지하기 때문에 수행시간이 더 빠르다. 즉 PSI 의 개수가 많아질수록 각각의 PSI 의 크기는 작아지기 때문에 쓰레드마다 작업하는 양이 적어지게 되어 수행시간이 빨라지게 된다.




(그림 4) CPU 와 GPU 의 수행시간(PSI 개수 = 100)




(그림 5) CPU 와 GPU 의 수행시간(PSI 개수 = 200)




(그림 6) CPU 와 GPU 의 수행시간(PSI 개수 = 300)

## 5. 결론

본 논문에서는 GPU 를 이용한 PSI 생성 알고리즘을 제안하였다. 제안하는 방법은 PSI 생성시 필요한 연산을 GPU 를 이용하여 병렬적으로 수행하였다. GPU 의 병렬성을 활용하면 대용량의 GPS 궤적에 대해서 순차적 연산을 수행하는 CPU 에서 보다 빠른 시간에 PSI 를 생성 할 수 있는 것을 실험을 통하여 알 수 있었다. 또한 PSI 의 개수가 늘어날수록 수행시간이 빨라지는 유용한 성질이 있음을 확인 할 수 있었다.

## 참고문헌

[1] Giannotti, Fosca, et al. "Trajectory pattern mining." Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007.

[2] Li, Quannan, et al. "Mining user similarity based on location history."Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems. ACM, 2008.

[3] Parent, Christine, et al. "Semantic trajectories modeling and analysis." ACM Computing Surveys (CSUR) 45.4 (2013): 42.

[4] Jang,Yuhee, et al.” A Technique for Generating Semantic Trajectories by Using GPS Moving Trajectories and POI Information” 한국정보처리학회 2015 년 춘계학술발표대회, VOL 22 NO. 01 PP. 0722 ~ 0725 2015. 04

[5] Parent, Christine, et al. "Semantic trajectories modeling and analysis." ACM Computing Surveys (CSUR) 45.4 (2013): 42.

[6] Owens, John D., et al. "GPU computing." Proceedings of the IEEE 96.5 (2008): 879-899.