한국컴퓨터정보학회 하계학술대회 논문집 제26권 제2호 (2018. 7)

소나 기반 해저 시뮬레이션의 성능 향상을 위한 병렬처리 적용 방법 연구

백승재*, 이건표**, 하옥균O*
O*경운대학교 항공소프트웨어공학과
**경상대학교 정보과학과
e-mail: qorxjaosx@naver.com, gnurvy2@gnu.ac.kr, okha@ikw.ac.kr

A Study on Application Method of Parallel Processing for Performance Improvement of Sonar-based Undersea Simulation

Seoung-Jea Back*, Keon-Pyo Lee**, Ok-Kyoon Ha1O*
O*Dept. of Aeronautics & Software Engineering, Kyungwoon University
**Dept. of Informatics, Gyeongsang National University

● 요 약 ●

해상 선박의 안전을 위해 해저의 객체 및 장애물의 정확한 탐지를 위해 해저환경에서 감쇠현상이 비교적 적은 음파 기반의 소나가 널리 활용된다. 그러나 기존의 소나 영상 시뮬레이션은 고해상도의 영상, 잡음 처리, 해저지형과 객체 데이터 등의 방대한 데이터 처리로 인해 물체 탐지 및 식별을 위한 처리속도와 비용이 크게 증가한다. 이러한 문제를 최소화하기 위해서 해저지형, 객체 생성과 잡음 처리 모델을 Multi-Threading, SIMD 등 병렬처리를 적용하여 처리속도를 최적화 한다. 본 논문에서는 혼합된 병렬처리 방법을 적용하여 소나를 기반으로 해저 환경 시뮬레이션을 위한 모의 신호를 생성하는 성능을 향상시킨다. 병렬처리로 인해 개선된 성능을 순차처리에 따른 속도와 실험적으로 비교한다.

키워드: Sonar image simulation, SIMD(Single Instruction Multiple Data), Multi-threading, Parallel processing

I. Introduction

소나 영상 시뮬레이션을 위해서는 센서 데이터로부터의 영상 합성, 물체 인식, 장치나 어군의 시뮬레이션과 같은 작업에서 방대한 연산이 수행되며, 이러한 작업들은 실시간 처리를 위해서 병렬처리를 사용해 처리 속도를 증대시키고 있다. 그러나 각 작업들의 처리속도를 효율적으로 향상시키기 위해서는 작업의 특성을 파악하고 적절한 연산장치와 병렬처리 기법을 선택하여 사용해야 한다.

본 논문에서는 병렬처리 기법을 복합적으로 적용하여 실시간 소나 영상 시뮬레이션의 성능을 향상시켜 실제 소나 영상 시뮬레이션에 실용적으로 적용할 수 있게 개선한다.

II. Background

1. 소나 영상 시뮬레이션

사이드 스캔 신호의 영상화는 해저의 지형, 부유하는 객체를 영상화하기 위해서는 해저 정보 뿐만 아니라 소나 영상을 생성하기 위한 추가적인 신호의 연산이 필요하여 물체 탐지 및 식별 처리속도가 크게 증가한다. 이러한 문제는 보통 Multi-Threading, SIMD와 같은 병렬처리 기법을 적용하여 최적화 또는 실시간성을 확보한다[1-2].

2. Multi-threading

Multi-threading은 하나의 프로세서 내에서 둘 이상의 thread를 사용하여 하나의 수행업무를 동시에 처리함으로써 컴퓨터의 처리속도를 증가시키는 병렬처리 기법이다. 다수의 thread는 stack을 제외한 Code, Data, Heap영역을 공유하기 때문에 메모리에 대한 효율성을 가질 수 있다.

3. SIMD(Single Instruction Multiple Data)

여러 개의 처리기가 하나의 제어 처리기에 의해 제어되는 구조이며, 모든 처리기는 제어 장치로부터 같은 명령어를 수행하도록 제어되나, 각각 다른 데이터대상으로 처리한다. 또한 128bit 또는 256bit 레지스터를 활용하여 기존의 연산보다 빠른 속도로 연산이 가능하다.

III. Design

1. parallel processing

소나 영상 시뮬레이션의 처리속도를 최적화하기 위해 소프트웨어의

데이터 의존성 및 모듈별 자료구조를 분석하여 메모리 효율성을 증가시키는 Multi- threading과 동시 처리 속도가 개선되는 SIMD 중 적절한 병렬처리 방법을 혼합하도록 설계한다.

1.1 Multi-threading

Fig 1은 소나 영상 시뮬레이션 소프트웨어에 Multi-threading 방법을 적용한 예로 지형과 물체에 반사되어 돌아오는 신호의 시간을 기준으로 각 각도의 펄스파형을 중첩하여 1ping의 신호를 생성할 때, 다중 스레드를 이용하여 thread당 1ping 신호를 연산하여 동시에 여러 개의 ping을 생성할 수게 한다.

```
        omp_set_num_threads(concurentThreadsSupported);
#pragma omp parallel for
        for (int ty = y; ty < y + concurentThreadsSupported; ty++)
```

Fig. 1. An example of codes using Multi-threading Method

1.2 SIMD

Fig 2는 소나 영상 시뮬레이션 소프트웨어에 OpenMP API 중 SIMD 병렬처리가 가능하도록 구현된 예를 나타낸다. 제시된 예는 표적에 반사되어 돌아오는 시간차이로 인해 중첩된 펄스를 수신하고, 수신 시간별로 서로 다른 펄스를 계산함으로써 중첩된 신호를 생성한다. 이때 중첩신호들의 각 펄스를 병합할 때 SIMD연산을 이용하여 병렬처리 함으로써 연산 및 처리 속도를 개선할 수 있게 한다.

```
#pragma omp simd
for (int i = 0; i < ied - ist; i++)
{
        sigi[i] += arg[i] * amp * (lev - levb);
}
```

Fig. 2. An example of codes using SIMD Method

IV. Implementation and Test

설계된 복합 병렬처리는 Intel i7 4-Core CPU, Windows 10 OS 환경에서 구현하고 실험하였다. 실험을 위해 해저지형 DTED 파일 2종에 드럼, 타이어 등 3D객체 모델링 5종을 합성하여 테스트 하였다. Table 1의 결과와 같이 기존의 소나 영상 시뮬레이션은 각 3D 객체 모델링을 하는데 평균 약260초의 시간이 걸린 반면, 복합 병렬처리를 적용하여 평균 약 72초의 시간이 소요되었다. 복합 병렬처리를 통해 평균 72%의 시간이 감소되어 소나 영상 시뮬레이션의 처리 속도의 최적화를 통해 실시간성을 개선한다.

Table 1. Measured results of undersea simulation

DTED / Objects	N42.DT2		N37.DT2	
	순차처리	병렬처리	순차처리	병렬처리
Drum	258	71	284	71
Fishing Banks	259	72	257	72
Sea Mine	266	71	293	71
Submarine	263	74	306	74
Tire	251	72	251	74
Average	259.4	72.0	278.2	72.4

V. Conclusions

본 연구에서는 소나 영상 시뮬레이션의 물체 탐지 및 식별 처리속도를 최적화하기 위해 Multi-threading과 SIMD 방법을 복합적으로 적용하여 병렬처리를 통한 성능향상을 제시하였다. 제시된 병렬처리를 적용한 소나 영상 시뮬레이션은 약 72%의 수행 시간 감소를 보여 개선된 성능을 제공한다. 향후 GPU 기반 병렬처리를 추가하여 실시간성 제공이 가능하도록 성능을 향상할 예정이다.

REFERENCES

[1] M. S. Rasmussen, M. B. Stuart, and S. Karlsson, “Parallelism and Scalability in an Image Processing Application,” International journal of parallel programming, Vol. 37, pp. 306-323, June 2009.

[2] M. Park, O. Ha, S. Ha, and Y. Jun “Real-time 3D Simulation for the Trawl Fishing Gear based on Parallel Processing of Sonar Sensor Data," Int’l Journal of Distributed Sensor Networks, 2014.