技術論文 DOI:http://dx.doi.org/10.5139/JKSAS.2012.40.9.818

접근물체 선별 알고리즘 계산 효율성 향상 연구

김형진*, 김해동**, 성재동*

A study on the Computational Efficiency Improvement for the Conjunction Screening Algorithm

Hyoung-Jin Kim*, Hae-Dong Kim** and Jae-Dong Seong*

ABSTRACT

In this paper, the improvement methods of the computational efficiency of the conjunction screening algorithm, which calculates the closest distance between primary satellite and space objects are presented. First method is to use GPU(Graphics Processing Unit) that has high computing power and handles quickly large amounts of data. Second method is to use Apogee/Perigee filter which excludes non-threatening objects that have low probability of collision and/or minimum distance rather than that of thresh hold. Third method is to combine first method with second method. As a result, the computational efficiency has been improved 34 times and 3 times for the first method only and second method only, respectively. On the contrary, the computational efficiency has been dramatically improved 163 times when two kinds of methods are combined.

초 록

본 논문에서는 우주파편 충돌위험 분석 과정에 1차적으로 필요한 접근물체 선별 알고리즘의 계산 효율성 향상 방법을 제시하였다. 첫 번째 방법은 높은 연산 능력을 바탕으로 대량의 데이터를 빠르게 처리할 수 있는 GPU(Graphics Processing Unit)를 이용하는 것이고, 두 번째 방법은 접근 가능성이 없는 물체들을 최소 근접거리 계산 과정에서 제외하여 계산 수행 시간을 단축할 수 있는 원/근지점 필터(Apogee/Perigee filter)를 이용하는 것이며, 세 번째 방법은 앞서 언급한 두 가지 방법을 결합하여 이용하는 것이다. GPU만 적용하였을 경우 평균 34 배 정도 계산 효율성이 향상되었고, 원/근지점 필터만 적용하였을 때는 평균 3 배 정도 계산 효율성이 향상되었다. 마지막으로 GPU와 원/근지점 필터를 함께 적용하였을 때는 약 163 배 정도 계산 효율성이 향상됨을 확인할 수 있었다.

Key Words : Space Debris(우주파편), Conjunction Screening(접근물체 선별), GPU(Graphics Processing Unit), Computational Efficiency(계산 효율성)

† 2012년 6월 29일 접수 ~ 2012년 8월 20일 심사완료
* 정회원, 과학기술연합대학원대학교 위성시스템 및 활용공학 전공
** 정회원, 한국항공우주연구원, 과학기술연합대학원대학교
교신저자, E-mail : haedkim@kari.re.kr
대전광역시 유성구 과학로 115

Ⅰ. 서 론

인류의 우주개발(Space Exploration)로 인해 지구 궤도상에는 수많은 우주물체(Space Objects)가 생성되었다. 우주물체에는 운영 중인 인공위성(Active Satellite)과 우주파편(Space Debris)이 포

함되는데, 이 중 제어가 되지 않는 우주파편의 경우 타 우주물체들과 충돌 위험성을 항상 내포하고 있다. 우주파편은 인류가 만들어낸 지구 궤도상에 있는 물체 중에서 작동하지 않는 것으로 정의되는데, 여기에는 로켓 발사 시에 분리되어 폐기된 로켓단(Stage)이나 파편, 발사체 혹은 위성으로부터 떨어져나간 부품, 폐기된 위성, 궤도상 위성 폭발로 인한 파편과 위성과 파편 혹은 위성과 위성끼리의 충돌로 인해 생성된 파편 등이 포함되며[1,2], 전체 우주파편의 약 84% 정도는 LEO(Low Earth Orbit) 상에 존재한다. 또한, 지구 주위 궤도상에 존재하는 우주파편은 크기가 10cm 이상인 것은 22,000개, 1cm ~ 10cm 사이인 것은 50만 개, 1mm ~ 1cm 사이인 것은 약 1억 개가 존재하는 것으로 추정되고 있다[3].

우주파편의 궤도상 속도는 저궤도의 경우 7~8 km/s, 정지궤도의 경우 3 km/s로, 운영 중인 인공위성이 지름 10 cm 이상의 우주파편과 충돌할 경우 완파되거나 기능이 정지될 수 있다.실제로, 지난 2009년 2월에 미국에서 운영 중인 IRIDIUM 33 위성과 운영이 멈춘 것으로 추정되는 러시아의 COSMOS 2251 위성이 궤도상에서 충돌하는 사건이 발생하였고, 미국은 운영 중인 인공위성을 잃어 경제적 손해를 입었으며 1,275개의 새로운 우주파편이 생성되어 우주환경이 악화되었다[4]. 이 사건을 계기로 우주 개발 선진국들은 우주파편 충돌위험 분석시스템을 개발하여 자국 인공위성의 안전과 우주환경 악화를 막고자 노력하고 있다. 현재 아리랑위성 2호, 3호, 천리안위성을 운영 중에 있으며, 아리랑위성 3A 및 5호를 발사해야 하는 우리나라에서도 항공우주연구원이 '우주파편 충돌위험 종합관리 시스템'을 '국가현안문제해결형과제(National Agenda Project, NAP)'의 협동과제로 개발 중에 있다[5].

우주파편 충돌위험 분석시스템의 첫 단계는 자국 인공위성과 NORAD 카탈로그 상에 있는 15,000 여개의 우주물체들 사이의 충돌확률 및 최소 근접거리를 구하는 것이다. 이 첫 단계를 접근물체 선별과정(Screening)이라고 하고, 충돌위험이 일정 범위를 초과하게 되면, 접근물체의 보다 정밀한 궤도 정보를 입수하거나 추적한 후 정밀궤도 결정 결과에 의존하여 상세 충돌위험 분석(Fine assessment)를 하게 된다. 이후 최종적으로는 최적화된 충돌회피기동 계획을 수립하여 실행하게 된다. 현재 개발 중인 시스템은 이러한 총괄적인 분석 작업 및 최적화된 회피기동계획을 제공하기 위해 개발되고 있다.

본 논문에서는 우주파편 충돌위험 분석 과정에 1차적으로 필요한 접근물체 선별 알고리즘의 계산 효율성을 높이기 위한 방법을 제시하였다. 첫 번째 방법은 높은 연산 능력을 바탕으로 대량의 데이터를 빠르게 처리할 수 있는 GPU를 이용하는 것이고, 두 번째 방법은 접근 가능성이 없는 물체들을 최소 근접거리 계산 과정에서 제외하여 계산 수행 시간을 단축할 수 있는 원/근지점 필터를 이용하는 것이며, 세 번째 방법은 앞서 언급한 두 가지 방법을 결합하여 이용하는 것이다. 각각의 방법을 이용하여 접근물체 선별 알고리즘의 계산 효율성을 얼마만큼 향상시킬 수 있는지에 대해 기술하였으며, 기술에 필요한 접근물체 선별 알고리즘의 계산 수행 시간은 MATLAB 상에서 측정된 값을 이용하였다.

제안된 접근물체 선별 알고리즘의 최종 결과값(사용자가 설정하는 제한 조건, 예) 최소 근접거리 <5km 를 만족하는 접근물체의 정보)은 상용 프로그램인 STK®의 Advanced Close Approach(CAT) 결과값과 비교하여 검증하였다.

II. 계산 효율성 향상 방법

2.1 GPU(Graphics Processing Unit) 이용

2.1.1 GPU

GPU는 그래픽 카드에 장착된 3D 그래픽 연산 전용 프로세서로써, 3D 게임 산업의 급속한 성장으로 등장하게 되었다. 3D 게임은 원활한 진행을 위해 비디오 프레임 당 엄청난 수의 부동 소수점 연산을 수행하여야 한다. 이 조건을 만족하기 위해 GPU에 다수의 프로세서를 장착하여 연산 능력을 증대시켰다. 초기 그래픽 카드의 경우 프로그램이 불가능하여, 높은 연산 능력을 가지고 있음에도 불구하고 다른 분야에서는 사용할 수 없는 상태였다. 하지만, 2007년 NVIDIA 사에서 프로그램이 가능한 그래픽 카드 G80을 출시하면서 여러 분야에서 GPU를 이용할 수 있게 되었고, 개인용 컴퓨터에서 수퍼 컴퓨터에 버금가는 연산 능력을 보유할 수 있게 되었다[6].

Fig. 1은 프로세서 기술 발전 방법에 따른 프로세서 성능을 비교한 것으로, 프로세서 동작 주파수를 증가시키는 기존의 방법 보다 프로세서의 수를 증가시키는 방법이 더 유용하다는 것을 나타내고 있다[7]. 이 결과를 바탕으로 프로세서의 성능 향상 기술이 프로세서의 수를 증가하는 방향으로 전환되었다. Fig. 2는 프로세서 성능 향상 기술 전환에 따른 CPU의 성능 향상을 나타낸 그림이다[8]. 현재 듀얼, 쿼드, 옥타 코어 CP

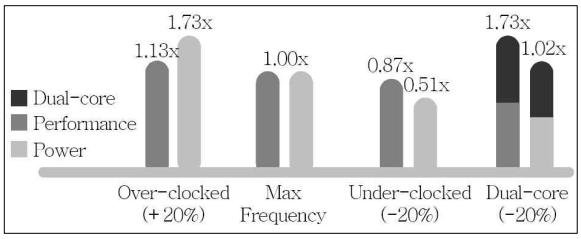

Fig. 1. 프로세서 기술 발전 방법에 따른 성능 향상 비교[7]

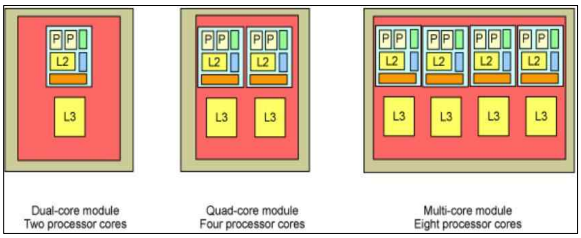

Fig. 2. CPU의 성능 향상[8]

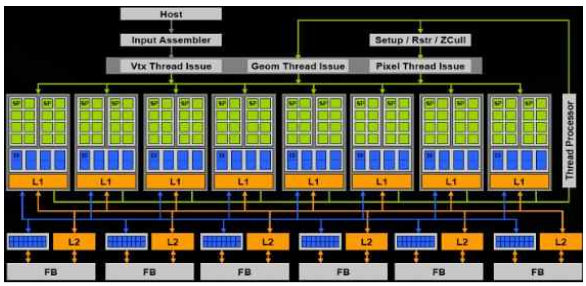

Fig. 3. GeForce GTX 8800 내부 구조[9]

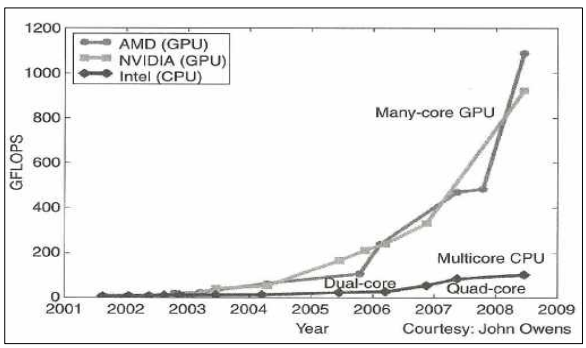

Fig. 4. 부동 소수점 연산 속도 비[10]

U가 시중에 판매되고 있으며, 도데카 코어, 헥사데카 코어가 개발되고 있다. Fig. 3은 NVIDIA사가 출시한 프로그램이 가능한 G80 계열의 GeForce GTX 8800의 내부 구조를 나타낸 것이다[9]. CPU 상의 Processor를 GPU 상에서는 SP(Streaming Processor)라고 명하고 있다. GeForce GTX 8800의 경우 총 128개의 SP를 보유하고 있으며, 그래픽 카드의 성능이 향상 될수록 보유할 수 있는 SP의 수가 증가하게 된다. Fig. 4는 CPU와 GPU의 부동 소수점 연산 능력을 비교한 것이다[10]. 부동 소수점 연산은 멀티미디어나 과학적 계산에 이용되는데, CPU와 GPU의 구조적 차이로 인해 부동 소수점 연산 능력이 크게 차이가 나는 것을 알 수 있다.

2.1.2 GPU 활용 사례

컴퓨터가 등장하면서부터 다양한 분야에서 컴퓨터를 이용해 데이터를 처리하고 있는데, 처리해야할 데이터의 양이 날이 갈수록 증가하고 있다. 데이터 양 증가는 곧 연산량 증가를 초래하며, 사용자가 결과값을 얻는데 오랜 시간이 걸리게 된다. 이러한 상황 속에서 프로그램이 가능한 그래픽카드가 출시되면서 여러 분야, 특히 처리해야 할 데이터의 양이 많은 분야에서 GPU를 이용하여 원하는 결과값을 빠른 시간 안에 얻을 수 있게 되었다. Fig. 5는 각 분야 별로 GPU를 이용하여 얻은 속도 향상 정도를 나타낸 것이다. 아래 그림 중 GIS(Geographic Information Services)의 경우 엄청난 양의 데이터로부터 필요한 정보를 정확하고 빠르게 찾아내기 위해 GPU를 이용하였고, CPU를 이용하였을 때 보다 34배 정도 빠른 연산결과를 얻을 수 있게 되었다. 다른 많은 분야에서도 대량 데이터를 빠르게 처리할 수 있는 능력을 보유한 GPU를 이용해 수 배 ~ 수백 배의 속도 향상을 얻었다[11]. 이러한 기존 결과를 바탕으로 본 논문에서 제시한 방법의 계산 효율성 정도가 기대 수준을 만족하는지를 가늠할 수 있다.

2.1.3 접근물체 선별 알고리즘에 GPU 적용

접근물체 선별 알고리즘은 우주파편 충돌위험 분석 과정에 필요한 최소 근접거리 값을 계산한다. Fig. 6은 접근물체 선별 알고리즘의 순서도이다. 알고리즘은 NORAD 카탈로그 데이터를 입

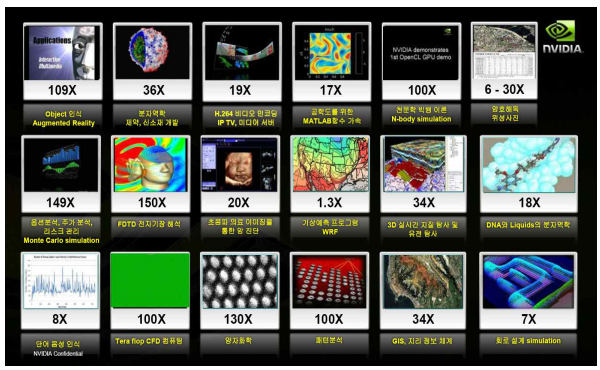

Fig. 5. GPU 활용 사례 및 속도 향상[11]

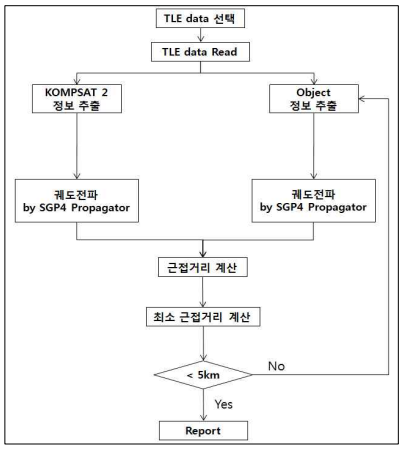

Fig. 6. 접근물체 선별 알고리즘

력값으로 받아 자국 인공위성과 접근물체 사이의 최소 근접거리를 계산하고, 사용자가 설정한 제한 조건(<5km)을 만족하는 접근물체의 정보(SSC Number, 최소 근접거리 및 최소 근접 시점)를 txt 파일 형태로 출력하도록 설정되어있다.

최소 근접거리를 계산하기 위해서는 두 물체의 궤도상 거리 성분이 필요하며, 이는 SGP4(Simplified General Perturbations No.4) 궤도전파기를 이용해 얻을 수 있다. SGP4 궤도전파기는 입력받은 시간 항목과 NORAD 카탈로그상에 있는 궤도정보를 이용해 섭동력을 계산한 뒤, 입력받은 시점에서의 거리 및 속도 성분을 반환해 준다. 이때, 입력할 시간 항목 및 적분 간격 단위가 작아질수록 계산양이 크게 증가하여 계산 수행 시간이 늘어나게 된다.

초속 7~8 km로 움직이는 우주물체들 간의 최소 근접거리를 계산하기 위해서는 적분간격을 기본적으로 초 단위로 설정한다. 초 단위로 설정하면 하루 동안 궤도전파 시키는데 86,400번의 계산이 수행되며, 이러한 과정이 약 15,000 여개의 우주물체들을 대상으로 진행되므로 총 계산 수행 시간이 급격하게 증가하게 된다. 또한, 계산된 86,400 개의 위치 성분들을 이용하여 자국 위성과 나머지 우주물체 간의 근접거리를 계산하는데도 많은 시간이 소요된다. Table 1은 STK®소프트웨어에 내장된 SGP4와 MATLAB 코드로 작성된 SGP4 궤도전파기[12]의 초 단위 궤도전파 계산 수행 시간을 비교한 것이다. MATLAB 코드로 작성된 SGP4 궤도전파기의 경우 STK SGP4 궤도전파기보다 평균 6배 정도 계산 시간이 오래 걸리는 것을 알 수 있는데 이는 MATLAB 컴파일 시간이 상대적으로 길기 때문이다.

본 논문에서는 GPU를 활용한 계산 시간 효율성 향상 방법을 제안하고 분석하기 위해 상대적으로 계산시간이 많이 소요되는 MATLAB 코드를 이용하였다.

한편, MATLAB 상에서 GPU를 이용하기 위해서는 MATLAB 버전이 2010b 이상이여야 하며, Parallel Computing Toolbox®와 GPU의 Compute Capability가 1.3 이상인 그래픽 카드가 설치되어 있어야 한다[13].

Table 2는 비교 실험에 사용된 프로세서 제원을 나타낸 것이다. NVIDIA GeForce GTX 460의 경우 Compute Capability가 2.1이며, 계산을 담당하는 SP(Streaming Processor)을 336개 보유하고 있다.

Fig. 7은 GPU의 데이터 처리 방식을 나타낸 것이다[14]. GPU 상에서 계산을 수행하기 위해서는 CPU 메모리상에 있는 데이터를 GPU 메모리로 복사하고, GPU에 장착된 다수의 프로세서에 계산량을 할당하여 계산을 수행하도록 하며, GPU 메모리상에 저장된 계산 결과값을 다시 CPU 메모리로 불러와야 한다. 이 같은 절차를 수행하기 위해 Parallel Computing Toolbox®을 이용하였다.

Table 1. STK SGP4 vs. MATLAB SGP4 수행 시간 비교

궤도전파 기간	수행 시간(sec)	
	STK SGP4	Matlab SGP4
1 일	0.9012	5.3073
5 일	4.2718	26.0799
6 일	5.0955	31.2187
7 일	5.8865	36.6430
10 일	8.3460	51.9202
20 일	16.4502	104.1003
30 일	24.8229	155.7641

Table 2. 비교 실험에 사용된 프로세서 제원

	Processor	Number of Processor
CPU	Intel Pentium Dual-Core	2
GPU	NVIDIA GeForce GTX 460	336

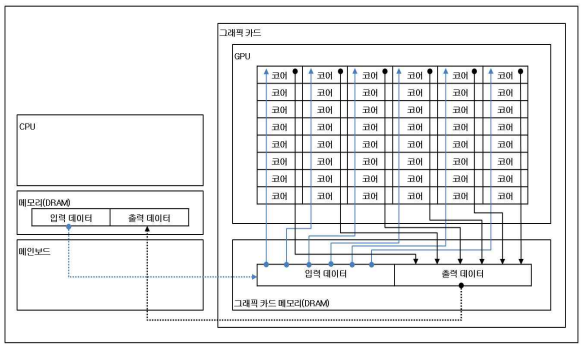

Fig. 7. GPU 데이터 처리 방식[14]

Table 3. 궤도전파 부분 연산 수행 시간 비교 (초 단위 궤도전파 기준)

궤도전파 기간	수행 시간(sec)		속도 향상
	CPU	GPU	
1 일	5.3073	0.1737	30.55배
5 일	26.0799	0.2542	102.59배
6 일	31.2187	0.2751	113.48배
7 일	36.6430	0.2941	124.59배
10 일	51.9202	0.3550	146.25배
20 일	104.1003	0.5608	185.62배
30 일	155.7641	0.7541	206.55배

Table 4. 근접거리 계산 부분 연산 수행 시간 비교

궤도전파 기간	수행 시간(sec)		속도 향상
	CPU	GPU	
1 일	1.0893	0.0161	67.65배
5 일	5.4514	0.0609	89.51배
6 일	6.5301	0.0728	89.69배
7 일	7.5835	0.0838	90.49배
10 일	10.8387	0.1189	91.15배
20 일	21.7455	0.2385	91.17배
30 일	32.6005	0.3525	92.48배

궤도전파 부분과 근접거리 계산 부분을 GPU 상에서 동작 되도록 코드를 수정한 결과, 정상적으로 동작함을 확인하였다. Table 3은 궤도전파 부분의 연산 수행 시간을 비교한 것으로, GPU를 이용하게 되면 최대 200배 정도 속도 향상을 얻을 수 있음을 확인하였다. Table 4는 근접거리 계산 부분의 연산 수행 시간을 비교한 것이다. 궤도전파 기간이 약 30일인 경우 최대 92배 정도의 속도 향상을 얻었다. 궤도전파 경우보다 속도 향상 정도가 낮은 이유는 근접거리 계산 부분의 경우 계산식이 상대적으로 단순하기 때문이다.

위 결과를 바탕으로 GPU 상에서 동작하는 궤도전파 부분과 근접거리 계산 부분을 접근물체 선별 알고리즘에 접목하여 알고리즘의 계산 효율성 향상 정도를 측정해 보았다.

접근물체 선별 알고리즘 수행 조건은 최소 근접거리 분석 기간(궤도전파 기간)을 1일로 설정하고, 아리랑 위성 2호와 15,000 개 우주물체 사이의 최소 근접거리를 계산하도록 설정하였다. Table 5는 CPU와 GPU 상에서의 접근물체 선별 알고리즘의 계산 수행 시간을 비교한 것이다. GPU를 이용할 경우 평균 34배 정도 계산 효율성이 향상됨을 알 수 있다.

Table 5. 계산 수행 시간 비교

TLE DB 생성시각	수행 시간(hour)		
	CPU	GPU	속도 향상
2011.07.04 pm	17.09	0.42	40.69배
2011.07.05 am	18.80	0.55	34.18배
2011.07.05 pm	18.94	0.55	34.43배
2011.07.06 am	18.89	0.56	33.73배
2011.07.07 am	17.09	0.56	30.51배
2011.07.10 pm	18.88	0.55	34.32배
2011.07.12 pm	18.06	0.56	32.25배

2.2 원/근지점 필터 (Apogee/Perigee Filter (A.P.F.) 이용

2.2.1 원/근지점 필터

원/근지점 필터는 최소 근접거리 결과를 얻는 과정에서 자국 인공위성에 접근 가능성이 없는 물체들을 계산과정에서 제외시키는 역할을 담당하고 있으며, 이를 통해 계산 수행 시간을 감소시킬 수 있다. 원/근지점 필터는 자국 인공위성과 접근물체의 원지점(Apogee), 근지점(Perigee) 값을 아래의 공식에 대입하여 설정 한계 값을 넘어서는지를 판단하여, 계산 과정에서의 제외 여부를 결정하게 된다.

$$q - Q > D \quad (1)$$

q 는 자국 인공위성과 접근물체의 근지점 값 중 큰 값을 나타내며, Q는 두 물체의 원지점 값 중 작은 값을 나타낸다. D는 설정 한계 값으로 본 연구에서는 30 km로 설정하였다. Fig. 8은 원/근지점 필터의 개념도를 나타낸 것이다[17,18]. 자국 인공위성(Primary)이 원 궤도를 돌고 있을 때, 자국 인공위성과 동일한 원 궤도이나 고도가 더 높은 접근물체(Secondary)의 경우 접근 가능성이 없으므로 계산과정에서 제외시키고, 타원 궤도를 돌고 있는 접근물체의 경우 접근 가능성이 있으므로 계산과정에 포함시킨다.

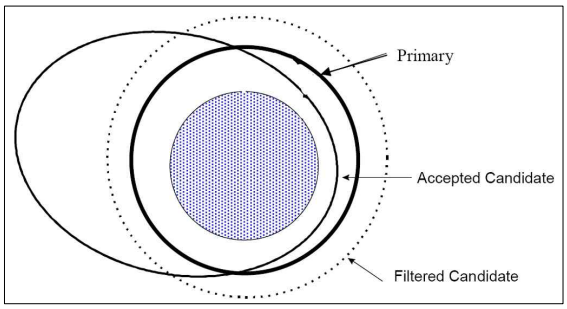

Fig. 8. 원/근지점 필터 개념도[18]

2.2.2 접근물체 선별 알고리즘에 원/근지점 필터 적용

원/근지점 필터 공식에 대입 할 원지점, 근지점 값은 TLE DB 상에 있는 평균궤도요소(Mean orbit elements)를 이용하거나 거리 및 속도 성분으로부터 얻은 접촉궤도요소(Osculating orbit elements)를 이용해 구할 수 있는데, 일반적으로 최소 근접거리 분석 기간의 시작점에서 계산하게 된다. 그 이유는 분석 기간 동안 근지점 값은 일정하게 유지되지만 원지점 값은 시간이 지날수록 감소하는 경향을 보이기 때문이다[18].

최소 근접거리 분석 기간의 시작점에서 원지점, 근지점 값을 계산하기 위해서는 SGP4 궤도 전파기를 이용해 얻은 분석 기간 시작점에서의 거리 및 속도 성분으로부터 계산된 접촉궤도요소를 이용하여야 하는데, 일부 접근물체의 경우에는 접근 가능성이 있음에도 불구하고 원지점/근지점 필터에 의해 제거되는 경우가 발생하게 된다. 이러한 경우에는 평균궤도요소를 이용하는 것이 더 적절한 것으로 알려져 있다[18].

따라서, 본 논문에서는 원/근지점 필터 공식에 대입 할 원지점, 근지점 값을 평균궤도요소를 이용하는 경우, 접촉궤도요소를 이용하는 경우에 따라 필터링 후 남은 물체의 수 측정하였고 STK®의 CAT 결과값과 비교 하였다. 실제 자국 인공위성에 근접 가능성이 있는 물체들을 정확하게 파악하는 것은 매우 어려운 일이므로, 검증된 상용 툴인 STK®의 CAT 결과값(필터링 후 남은 물체 수 & 최종 검출 물체 수)을 자국 인공위성에 접근하는 물체 수의 참값으로 설정하였다.

또한, 접근물체 선별 알고리즘에 원/근지점 필터를 적용했을 때와 적용하지 않았을 때의 계산 수행 시간을 측정하여 비교하였다.

2.2.2.1 평균궤도요소를 이용한 경우

평균궤도요소를 이용하는 것은 TLE DB 상에 있는 평균궤도 정보 중 원하는 궤도 정보(Eccentricity, Mean Motion)를 얻어 원지점, 근지점 값을 계산하는 경우를 말한다.

Table 6은 자국 인공위성과 접근물체 모두 평균궤도요소를 이용해 원지점, 근지점을 계산하여 필터 공식에 대입한 결과를 나타낸 것이다. STK®보다 대략 400 ~ 600개의 물체를 더 제거함을 볼 수 있다. 이는 분석 기간 시작점(자국 인공위성의 기산일)에서 원지점, 근지점 값을 계산하여야 하는데, 접근 물체의 경우 접근 물체의 기산일(Epoch time)을 기준으로 원지점, 근지점 값을 계산하였기 때문에 분석 기간 시작점에서 계산된 각각의 값들이 서로 차이가 나기 때문이다. Table 7은 필터 미 적용 시와 적용 시의 알고리즘 계산 수행 시간을 비교한 것이다. CPU 상에서 필터를 적용하지 않는 경우보다 필터를 적용하여 근접물체를 선별하는 작업 소요시간이 약 3.3 배 정도 향상된 것을 알 수 있다.

본 연구에서 평균궤도요소를 이용하여 원지점, 근지점을 계산하는 필터의 구현 결과, 비교 대상인 STK®보다 근접물체를 더 제거하였다. 필터 상호 간의 차이는 STK®가 기본적으로 평균궤도요소와 접촉궤도요소를 특정 상황에서 일정 규칙을 가지고 번갈아 적용하기 때문으로 사료되나, 이 규칙에 대한 상세한 내용은 공개된 바 없어 더 이상의 정확한 비교는 어렵다. 그럼에도 불구하고, 최종적으로는 최소 근접거리 제한 조건(<5 km)을 만족하는 접근물체의 개수는 STK®와 동일하게 검출된 것을 다음 2.2.2.4절에서 기술한 바와 같이 확인하였으므로 구현 필터의 타당성은 검증된 것으로 사료된다.

Table 6. 구현 필터의 필터링 결과

TLE DB 생성시각	필터링 후 남은 물체		
	STK	구현필터	차이
2011.07.04 pm	4875	4326	549
2011.07.05 am	4892	4321	571
2011.07.05 pm	4876	4325	551
2011.07.06 am	4845	4326	519
2011.07.07 am	4849	4328	521
2011.07.10 pm	4808	4343	465
2011.07.12 pm	4783	4352	431

Table 7. 필터 미 적용 시와 적용 시의 계산 수행 시간 비교

TLE DB 생성시각	수행 시간(hour)		
	필터미적용 (CPU)	필터적용 (CPU)	속도 향상
2011.07.04 pm	17.09	5.52	3.09배
2011.07.05 am	18.80	5.17	3.63배
2011.07.05 pm	18.94	5.17	3.66배
2011.07.06 am	18.89	5.60	3.37배
2011.07.07 am	17.09	5.26	3.24배
2011.07.10 pm	18.88	6.33	2.98배
2011.07.12 pm	18.06	5.78	3.12배

2.2.2.2 접촉궤도요소를 이용한 경우

접촉궤도요소를 이용하는 것은 SGP4 궤도전파기를 이용해 얻은 분석 기간 시작점에서의 거리 및 속도성분으로부터 계산된 접촉궤도요소 중 원하는 궤도 정보(Eccentricity, Mean Motion)를 얻어 원지점, 근지점 값을 계산하는 경우를 말한다.

Table 8은 자국 인공위성과 접근물체 모두 접촉궤도요소를 이용해 원지점, 근지점을 계산하여 필터 공식에 대입한 결과를 나타낸 것이다. 접촉궤도요소를 이용하게 되면 자국 인공위성과 접근물체 모두 최소 근접거리 분석 기간의 시작점에서 원지점, 근지점 값을 계산하기 때문에 평균궤도요소를 이용하였을 때 보다 물체를 덜 제거하였지만, STK®보다는 대략 130 ~ 200개의 물체를 더 제거하는 것을 알 수 있다. 이는 앞서 언급했던 것처럼 일부 접근 물체의 경우에는 평균궤도요소를 이용하는 것이 더 적합하기 때문이다. Table 9는 필터 미 적용 시와 적용 시의 알고리즘 계산 수행 시간을 비교한 것이다. 필터를 적용했을 때 약 2.98 배 계산시간 감소가 있음을 알 수 있었다. 이 경우에도 최소 근접거리 제한 조건(<5 km)을 만족하는 최종 접근물체의 개수는 STK®와 동일하였다.

2.2.2.3 두 가지 궤도요소를 결합하여 이용한 경우

최소 근접거리 분석 기간의 시작점은 자국 인공위성의 기산일에 맞추어져 있기 때문에, 자국 인공위성의 경우에는 평균궤도요소와 접촉궤도요소를 이용해 근지점, 원지점 값을 계산할 수 있다. 접촉궤도요소를 이용한 것은 2.2.2.2절에서 기술하였으므로 여기서는 자국 인공위성은 평균궤도요소를 이용하고 접근물체는 접촉궤도요소를 이용했을 때 어떤 결과를 얻을 수 있는지 알아보았다.

Table 10은 자국 인공위성은 평균궤도요소를, 접근물체는 접촉궤도요소를 이용해 원지점, 근지점을 계산하여 필터 공식에 대입한 결과를 나타낸 것이다. STK®보다 대략 300개 ~ 500개 정도 물체를 더 제거함을 알 수 있으며, 자국 인공위성(아리랑위성 2호)의 경우 접촉궤도요소를 이용하는 것이 STK®와의 차이를 더 줄일 수 있음을 알 수 있다. Table 11은 필터 미 적용 시와 적용 시의 알고리즘 계산 수행 시간을 비교한 것이다. 약 3.3 배 계산시간 감소가 있음을 알 수 있었다. 이 경우에도 최소 근접거리 제한 조건(<5 km)을 만족하는 최종 접근물체의 개수는 STK®와 동일하였다.

Table 8. 구현 필터의 필터링 결과

TLE DB 생성시각	필터링 후 남은 물체		
	STK	구현필터	차이
2011.07.04 pm	4875	4713	162
2011.07.05 am	4892	4704	188
2011.07.05 pm	4876	4695	181
2011.07.06 am	4845	4696	149
2011.07.07 am	4849	4665	184
2011.07.10 pm	4808	4658	150
2011.07.12 pm	4783	4644	139

Table 9. 필터 미 적용 시와 적용 시의 계산 수행 시간 비교

TLE DB 생성시각	수행 시간(hour)		
	필터미적용 (CPU)	필터적용 (CPU)	속도 향상
2011.07.04 pm	17.09	5.63	3.03배
2011.07.05 am	18.80	6.77	2.77배
2011.07.05 pm	18.94	6.74	2.81배
2011.07.06 am	18.89	5.98	3.15배
2011.07.07 am	17.09	5.93	2.88배
2011.07.10 pm	18.88	5.95	3.17배
2011.07.12 pm	18.06	5.94	3.04배

Table 10. 구현 필터의 필터링 결과

TLE DB 생성시각	필터링 후 남은 물체		
	STK	구현필터	차이
2011.07.04 pm	4875	4414	461
2011.07.05 am	4892	4401	491
2011.07.05 pm	4876	4420	456
2011.07.06 am	4845	4416	429
2011.07.07 am	4849	4410	439
2011.07.10 pm	4808	4444	364
2011.07.12 pm	4783	4441	342

Table 11. 필터 미 적용 시와 적용 시의 계산 수행 시간 비교

TLE DB 생성시각	수행 시간(hour)		
	필터미적용 (CPU)	필터적용 (CPU)	속도 향상
2011.07.04 pm	17.09	5.56	3.07배
2011.07.05 am	18.80	5.14	3.65배
2011.07.05 pm	18.94	5.60	3.38배
2011.07.06 am	18.89	5.66	3.33배
2011.07.07 am	17.09	5.12	3.33배
2011.07.10 pm	18.88	5.65	3.34배
2011.07.12 pm	18.06	5.64	3.20배

2.2.2.4 원/근지점 필터 적용 후 최종결과 검증

본 절에서는 원/근지점 필터를 이용하여 1차적으로 30 km 이내 접근하는 물체를 선별한 후 최종적으로 5 km 이내로 최소 근접거리가 발생하는 물체가 정확하게 확인되는지 여부를 알아보았다.

TLE DB의 경우 15,000개의 우주물체 정보를 담고 있다. 각각의 방법을 이용하였을 경우 대략 10,000개의 우주물체들을 제거하였고, 평균 3배 정도 계산 효율성을 향상 시켰음을 앞 절에서 기술하였다. 이때, 본 연구에서 구현된 원/근지점 필터 적용 시 최종적으로 선별되어야 할 물체가 필터링 과정에서 잘못 제거가 되지는 않았는지 확인해 볼 필요가 있다.

Table 12는 테스트에 사용한 2011.07.04 pm에 생성된 TLE DB를 이용하여, 최종 최소 근접거리가 5 km 이내인 물체를 상호 비교한 결과이며, 상호 간의 최소 근접거리 계산 차이는 10 cm 이내로 거의 동일함을 확인하였다. Table 13은 STK®와 접근물체 선별 알고리즘의 최소 근접거리 제한 조건을 만족하는 검출 물체의 최종 개수를 비교한 것이다. 7개의 TLE DB를 사용한 결과 최소 근접거리 5 km 이내에 접근하는 물체가 총 15개 검출되었으며, 검출된 물체들은 모두 STK®에서 얻은 물체의 정보(식별번호, 궤도정보)와 동일함을 확인하였다.

Table 12. 최소 근접거리 비교

물체 정보 (식별번호)	최소 근접거리 (km)		차이 (km)
	STK®	구현 알고리즘	
SSC_No 14780	2.286847	2.286856	0.000009
SSC_No 17316	2.329142	2.329145	0.000003

Table 13. 최종 검출 물체 개수 비교 (최소 근접거리 5 km 미만 물체)

TLE DB 생성시각	최종 검출 물체 개수	
	STK®	구현 알고리즘
2011.07.04 pm	2개	2개
2011.07.05 am	4개	4개
2011.07.05 pm	3개	3개
2011.07.06 am	1개	1개
2011.07.07 am	3개	3개
2011.07.10 pm	2개	2개
2011.07.12 pm	0개	0개

2.3 GPU와 원/근지점 필터를 함께 이용

접근물체 선별 알고리즘에 GPU만 이용하였을 경우에는 약 34배, 원/근지점 필터만 사용하였을 때는 약 3배 정도 계산 효율성이 향상되었다. 이에 두 가지 방법을 결합하였을 경우에는 어느 정도 계산 효율성이 향상되는지를 알아보았다.

Table 14는 GPU와 원지점/근지점 필터를 접근물체 선별 알고리즘에 동시에 적용한 결과를 나타낸 것이다. 이때, 필터 공식에 사용된 원지점, 근지점 값은 접촉궤도요소를 이용하였다. GPU와 원/근지점 필터를 결합한 경우 표에서 확인하는 바와 같이 계산 효율성이 평균 163배 정도 향상됨을 확인하였다.

이는 이론적인 예상 성능 향상 수치(34×3=102배)에 비해 59% 정도 추가 성능 향상이 된 것인데, 이는 Table 5의 GPU 수행 시간과도 비슷한 결과(이론치: 0.79hour, 실제치: 0.42hour, 32% 추가 성능 향상됨)이다.

위와 같은 추가 성능 향상은 MATLAB 상에서 for문을 수행할 때, MATLAB 내부적으로 연산속도를 높이는 알고리즘이 적용되는 것으로 생각되며, for문으로 수행해야할 데이터의 양이 적을수록 추가 속도 향상의 정도가 더 큰 것을 알 수 있다.(Table 5에서는 15,000개 데이터, Table 14에서는 5,000개 데이터 수행)

추후 우주물체의 개수가 급격하게 증가하여 접근물체 선별 알고리즘 적용 시 계산 시간 증가가 불가피할 경우 GPU와 원/근지점 필터를 함께 적용하면 계산 효율성을 대폭 향상 시킬 수 있음을 알 수 있다.

Table 14. CPU 기반 필터 미 적용 시와 GPU 기반 필터 적용 시의 계산 수행 시간 비교

TLE DB 생성시각	수행 시간(hour)		
	CPU 기반미필터	GPU 기반필터	속도 향상
2011.07.04 pm	17.09	0.11	155.36배
2011.07.05 am	18.80	0.11	170.90배
2011.07.05 pm	18.94	0.11	172.18배
2011.07.06 am	18.89	0.12	157.41배
2011.07.07 am	17.09	0.11	155.36배
2011.07.10 pm	18.88	0.11	171.63배
2011.07.12 pm	18.06	0.11	164.18배

Ⅲ. 결 론

본 논문에서는 우주파편 충돌위험 분석 과정에 1차 단계로써 필요한 최소 근접거리 값을 구하기 위해 구현된 접근물체 선별 알고리즘의 계산 효율성 향상 방법에 대해 기술하였다. 첫 번째 방법은 높은 연산 능력을 보유한 GPU를 이용하는 것이고, 두 번째 방법은 접근 가능성이 없는 물체들을 최소 근접거리 계산 과정에서 제외하여 계산 수행 시간을 단축할 수 있는 원/근지점 필터를 이용하는 것이며, 세 번째 방법은 앞서 언급한 두 가지 방법을 결합하여 이용하는 것이다. 결과적으로 세 번째 방법을 이용하게 될 경우, 접근물체 선별 알고리즘의 계산 효율성을 대폭 증대시킴을 알 수 있었다.

계산 효율성 증대의 가장 큰 중점은 순차적으로 데이터를 처리하던 for문을 GPU를 이용해 병렬적으로, 그리고 원/근지점 필터를 이용해 for문에서 처리해야할 데이터의 양을 줄이는 것이다. 이를 통해 기존 CPU만 이용할 경우와 비교해 계산 효율성을 크게 향상 시킬 수 있었다.

우주쓰레기를 포함한 우주파편의 개수는 지속적으로 증가하고 있는 추세이므로 향후 보다 많은 우주물체와의 선별 작업이 필요할 것으로 예상되며, 제안된 방법은 이 작업의 계산 효율성 향상에 크게 기여할 수 있을 것으로 사료된다.

후 기

본 연구는 기초기술연구회 'NAP 우주물체 전자광학 감시체계 기술개발'의 협동연구과제('우주파편 충돌위험 종합관리시스템 개발 및 우주파편 제거시스템 연구')의 일부로써 수행되었으며, 이에 기초기술연구회와 한국항공우주연구원의 지원에 감사드립니다.

참고문헌

1) "Technical Report on Space Debris", UNITED NATIONS, New York, 1999, pp. 2.

2) "Space Debris", Parliamentary Office of Science and Technology, postnote No. 355, 2010.

3) Liou. J.-C., "Orbital Debris and Future Environment Remediation", Presentations at the KARI, 2011.

4) http://celestrak.com/events/collision

5) 김해동, 김은혁, 엄위섭, 김은규, 김학정, "우주파편 충돌위험 종합관리시스템 개념설계", 한국항공우주학회 추계 학술대회, 2011, pp. 543~546.

6) 이주석, 류현곤, "GPU 병렬 컴퓨팅 기술을 이용한 개인용 수퍼 컴퓨터 현황과 전망", 전자공학회지 제 36권 제 5호, 2009, pp. 562~571.

7) 정무경, 박성모, 엄낙웅, "병렬 프로세서 기술 및 동향 ", 전자통신동향분석 제 24권 제 6호, 2009, pp. 86~93.

8) www.goggle.co.kr/image

9) Thanh-Nghi Do, Van-Hoa Nguyen, "A Novel Speed-up SVM Algorithm for Massive Classification Tasks", RIVF, 2008, pp. 215~220.

10) 데이비드 B. 커크, 원메이 W.후, 하순회, 김크리스, 이영민, "대규모 병렬 프로세서 프로그래밍:CUDA를 이용한 실용적 접근", 비제이퍼블릭, 2011, pp. 20~21.

11) www.3dskoreaforum.com/pdf/day1_6_NVIDIA.pdf

12) www.centerforspace.com

13) www.mathworks.co.kr/discovery/matlab-gpu.html

14) 정영훈, "CUDA 병령 프로그래밍", 프리렉, 2011, pp. 13~36.

15) www.mathworks.co.kr/help/toolbox/distcomp/ arrayfun.html

16) http://blogs.mathworks.com/loren/2011/07/ 18/a-mandelbrot-set-on-the-gpu/

17) Hoots, F. R., Crawford, L.L., Roehrich, R.L., " An analytic method to determine Future close between satellites", celestial mechanics, Vol. 33, 1984, pp. 143~158.

18) James Woodbrun, Vincent Copploa, Frank Stoner, " A description of filters for minimizing the time required for orbital conjunction computations", AAS/AIAA Astrodynamics Specialist Conference, 2009, pp. AAS 09-372.