2009년도 대한전기학회 하계학술대회 논문집 2009. 7. 14 - 17

GPU를 이용한 고속 카메라 모션 추적 시스템

유동현*, 김도윤*, 김재헌**, 유정재**, 김혜미**
위드로봇(주)*, 한국전자통신연구원**

High Speed Camera Motion Tracking System using GPU

Dong-Hyun Yoo*, Do-Yoon Kim*, Jae-Heon Kim** , Jung-Jae Yoo**, Hye-Mi Kim**
Withrobot*, ETRI**

Abstract - 영상처리시스템은 대량의 데이터를 고속으로 처리해야하기 때문에 고성능의 프로세서를 요구한다. 카메라의 성능은 점차 해상도가 높아져서 데이터가 많아지고 있는 반면 프로세서의 성능은 물리적인 한계로 인해서 단일 프로세서로는 속도 향상에 한계에 부딪히고 있다. 최근 CPU업계에서의 추세는 단일코어의 성능향상 한계로 인해 점차 코어의 개수를 늘리는 방v향으로 개발이 진행되고 있는데 이와 같이 병렬 프로세싱을 이용해서 영상처리시스템을 개발하는 연구가 최근 진행되고 있다. 병렬처리프로세싱 방법의 하나로 그래픽카드의 프로세서인 GPU를 사용하는 방법이 많이 시도되고 있다. 본 연구에서는 GPU를 이용하여 카메라의 모션을 추적하는 시스템을 실시간 시스템으로 개발하는 방법을 소개하고자 한다.

1. 서 론

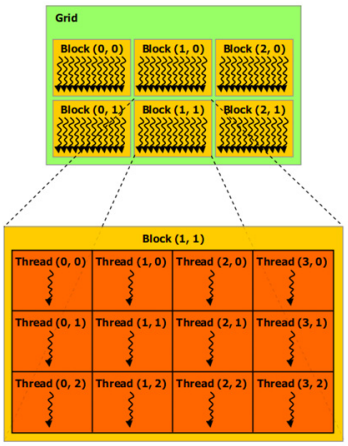
우리말로 중앙연산처리장치로 번역이 되는 CPU(Central Processing Unit)는 말 그대로 시스템 내에서 연산을 처리하는 중추 역할을 담당한다. CPU는 잘 알려진 무어의 법칙에 따라 지속적으로 성능은 발전하고 있지만, 화려한 2D, 3D 그래픽 요구를 다 수용하기에는 무어의 법칙으로도 따라잡기가 어려운 상황이다. 그래픽 카드 업체에서는 이러한 사용자의 요구를 수용하기 위해 다각도로 모색하던 중 그래픽 연산에 해당하는 코드를 분석한 결과 상당 부분의 코드가 반복되는 내용은 수행하고 있음을 발견하였다. 따라서 CPU 는 그래픽 연산 이외에도 처리해야 할 일이 많으므로 코프로세서(Co-Processor) 성격으로 별도의 프로세서를 그래픽 카드에 장착하고 그래픽스에서 자주 사용하는 계산 함수들, 즉 예를 들면 텍스쳐 매핑, 폴리곤 랜더링, 이동, 회전등을 빠르게 하드웨어적으로 처리할 수 있도록 시스템을 구성하기 시작하였다. 이때 그래픽 카드에 장착된 프로세서를 GPU(Graphic Processing Unit)라고 부른다. 아울러 OpenGL 이나 DirectX 와 같은 그래픽스 라이브러리도 이러한 추세에 발맞춰 GPU 를 지원하기 시작하였고, 게임 산업의 시장 수요로 인해 GPU 쪽은 비약적인 발전을 맞이하게 된다. 최초 GPU 가 등장한 시점은 업체마다 연구자마다 말이 엇갈리지만, 1990년대 후반부터 GPU 라는 용어가 정착되고 사용자들에게 많이 알려지게 된 사실은 모두들 동의하는 바이다. 범용적인 기능을 가져야 하는 CPU에 비해 특정한 연산에 특화된 GPU 는 하드웨어 제조 공정도 상대적으로 단순하여 그 발전 속도가 빨라, 2000 년대 들어와서는 단순히 실수 연산 성능만을 비교할 경우 GPU 가 CPU보다 앞서게 되었다.
영상처리시스템의 경우도 입력영상데이터의 크기가 크기 때문에 단일 프로세서만으로 실시간 처리가 불가능한 경우가 많다. 이를 해결하기 위해서 최근에 GPU를 이용하는 시도가 다각적으로 진행되고 있다. 대표적으로 의료영상과 같이 고해상도의 영상을 빠르게 분석하는 경우와 실시간연산이 필요한 경우에 GPU가 사용된다.

본 연구에서는 카메라의 모션을 추적하는 시스템을 실시간으로 동작하도록 하기 위해서 GPU로 개발하였다. 카메라 모션 추적시스템은 영상시퀀스에서 특징점을 추출하고 지속적으로 특징점을 추적하면서 이를 통해 카메라의 모션을 추정하고 특징점의 위치를 파악하는 시스템이다. 이 시스템은 로봇의 자기위치를 인식하고 지도를 생성하는 SLAM(Simultaneous Localization And Map-building)이나 카메라로 촬영한 영상에 CG캐릭터를 합성하기 위한 AR(Augmented Reality)에 응용할 수 있다. 본 연구에서는 CPU로 개발된 시스템을 GPU로 병렬프로세싱을 하기 위한 방법론은 소개하고 그 결과를 제시하려고 한다.
2장에서는 카메라 모션 추적 시스템을 소개하고 3장에서는 병렬프로세싱을 위한 방법론을 제시한다. 4장에서는 CPU와 GPU 동작시의 연산속도를 비교한 결과를 제시하고 끝으로 5장에서 결론을 맺는다.

2. GPU

2.1 소프트웨어 프로그램 모델

본 연구에서는 Nvidia의 GPU를 사용하였는데 소프트웨어 관점에서 병렬처리가 이루어지는 구조는 그림1과 같다. 하나의 GPU는 그리드라고 부르는데 하나의 그리드는 다수의 2차원 블록으로 구성되고 각 블록은 다시 다수의 thread로 구성되는 구조를 갖는다. 블록 내에 있는 thread는 3차원까지 구성이 가능하다. thread는 단일 처리의 기본 단위가 되고 block은 병렬 처리의 기본 단위가 된다. 각각의 블록은 고유의 ID를 가지고 있어서 그 ID를 통해 각 블록에 접근할 수 있고 마찬가지로 각 thread도 고유의 번호를 가지고 접근이 가능하다.



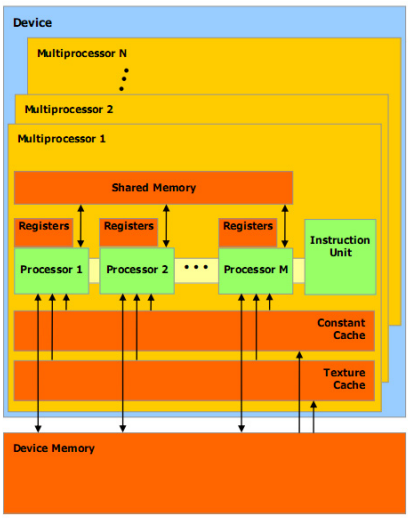
<그림 1> Nvidia GPU의 소프트웨어 모델

2.2 하드웨어 모델

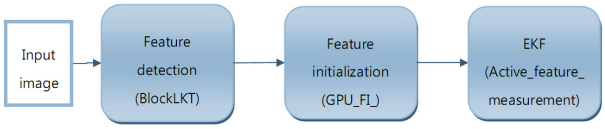
실제 GPU의 하드웨어를 보면 그림2와 같다. GPU 디바이스는 다수의 멀티프로세서로 구성되어 있고 각 멀티프로세서는 다시 다수의 스칼라 프로세서라고 부르는 프로세서로 이루어져있다. 그리고 레지스터, shared memory, constant cache, texture memory, device memory의 5종류의 메모리 영역이 있는데 각 메모리의 특성이 달라서 성능을 높이기 위해서는 적절한 사용이 필요하다. Nvidia 그래픽카드의 경우 하나의 멀티프로세서는 8개의 스칼라 프로세서로 구성되고 그래픽카드의 모델에 따라서 멀티프로세서의 개수는 다양하게 구성된다. 예를 들어, 본 연구에서 사용한 Geforce GTX295의 경우는 30개의 멀티프로세서가 있어서 총 240개의 스칼라 프로세서가 있고 병렬로 최대 240개의 thread가 동작할 수 있다.

3. 카메라 모션 추적 시스템

전체 프로그램의 구성은 그림<>과 같이 크게 세 개의 부분으로 나누어진다. BlockLKT는 영상에서 추적을 위한 특징점을 추출하는 것으로 Lukas-Kanade tracker 에서 사용하는 corner 추출 방법을 이용하여 특징점을 추출한다. 이렇게 추출된 특징점은 GPU_FI에서 feature initialization을 수행하는데 각 특징점의 projection ray를 따라 100개의 샘플을 정하고 이 샘플 주변의 탐색 영역에서 NCC를 계산하고 가장 유사도가 높은 위치를 찾는다. 이러한 과정이 GPU_FI에서 수행된다. GPU_FI에서 정해진 특징점들은 EFK를 이용하여 추적하는데 여기서 Active_feature_measurement함수에 의해서 원래의 특징점 패치와 현재 프레임에서의 패치를 비교하는 NCC 계산이 이루어진다. 이와 같은 세 개의 함수는 많은 연산을 필요로 하는 부분으로 이 함수들을 GPU에서 병렬처리하여 실시간 연산이 가능하도록 한다.



〈그림 2〉 Nvidia GPU의 하드웨어 모델



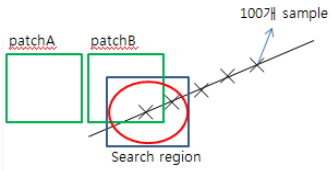
〈그림 3〉 카메라 모션 추적 시스템

3.1. Feature detection

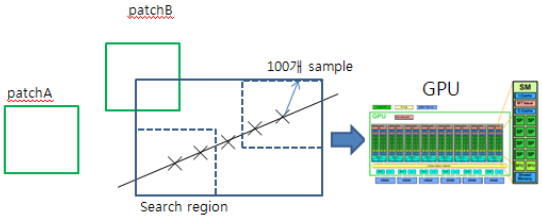
입력 영상이 들어오면 가장 먼저 특징점을 추출하는데 영상의 크기가 크기 때문에 전체 영역을 각 프레임에서 처리하는 것은 시간이 많이 소요되는 작업니다. 따라서 실시간을 가능하게 하기 위해서 전체 영상을 6x4의 블록으로 나누고 랜덤으로 블록을 선택하여 선택한 블록에 대해서 Lukas-Kanade feature를 추출한다. 85x85픽셀의 블록에 대해 특징점을 추출하는 작업을 CPU에서 순차적으로 처리할 경우 동일한 연산이 7225번 수행되어 39ms가 소요된다. 하지만 이를 GPU에서 85x85의 2차원 thread를 만들어서 하나의 thread가 하나의 픽셀에 해당하도록 구현하면 7225의 연산이 거의 동시에 수행된다.

3.2. Feature initialization

Feature initialization 함수는 그림4와 같이 영상에서 특정한 선을 따라 분포하는 샘플 주변으로 탐색 영역을 설정하고 그 탐색영역의 각 픽셀마다 패치B를 패치A와 비교하여 유사도를 계산하는 함수이다. 유사도는 NCC(Normalized Cross Correlation)에 의해 결정된다. 이러한 과정은 각 특징점 마다 수행되는 것으로 특징점의 개수만큼 반복 수행된다.



〈그림 4〉 Feature initialization



〈그림 5〉 Feature initialization의 병렬화

CPU에서 동작하는 코드는 6중의 for 문 루프가 존재하여 연산시간이 많이 소요되는 부분인데 이를 GPU에서 병렬화하기 위해 sample에 대한 for문 루프 이하를 GPU에서 한 번에 처리하도록 병렬화하였다. 샘플 100개에 대한 유사도를 GPU에서 한 번에 처리하도록 하기 위한 구조는 그림5와 같다. 각 샘플에 대한 탐색영역을 모두 포함하는 하나의 커다란 bounding box를 결정하고 그 영역 전체를 GPU에서 처리하도록 하는데, 이 때 GPU의 하나의 thread가 탐색영역의 각 픽셀을 담당하여 그 위치에 해당하는 patchB와 patchA의 NCC 유사도를 계산한다.

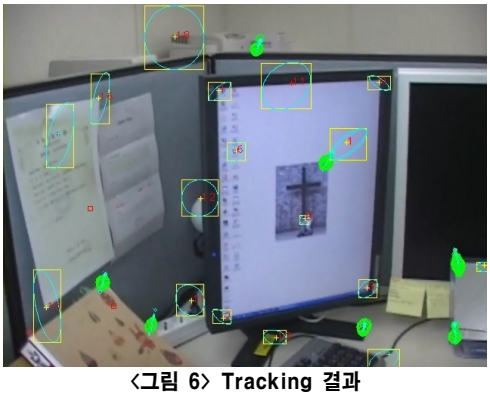
3.3. Active feature measurement (AFM)

AFM함수는 추적 중인 특징점에 대해 이 전 프레임에서의 위치 주변으로 특정 탐색영역을 설정하고 그 탐색영역 내에서 patchB와 기억하고 있는 patchA를 비교하여 유사도를 구한다. 이 함수는 앞에서의 GPU_FI와 유사하지만 patchB가 입력영상 그대로가 아닌 warping 된 패치라는 것이 다르다. Feature initialization과 마찬가지로 각 특징점마다 일정한 탐색영역의 크기에 대해서 NCC로 유사도를 계산한다. 여기서는 병렬화를 최대화하여 GPU를 최대로 활용하기 위해서 모든 특징점에 대한 탐색영역을 한번에 GPU로 전달하고 GPU의 각 thread는 모든 특징점의 탐색영역의 각 픽셀을 담당하여 NCC를 계산한다. 이렇게 하므로써 현재 추적 중인 모든 특징점들의 주변 탐색영역에 대한 매칭이 동시에 처리되어 매우 빠르게 연산이 수행된다.

4. 결 과

CPU와 GPU의 성능을 비교하기 위해 1024x768의 입력영상에 대해서 각각 테스트를 수행하였다. 사용된 컴퓨터는 인텔 코어2쿼드 2.4GHz 프로세서를 사용하고 2GB 램이 장착되어 있다. 그래픽카드는 Nvidia의 Geforce GTX295를 사용하였다. 이와 같은 환경에서 각 모듈을 CPU와 GPU에서 동작시켰을 때의 처리 속도는 표1과 같다.

그림 6은 일반환경에서 특징점을 추적하고 카메라의 모션을 추적하는 과정을 보여주는 결과이다.



〈그림 6〉 Tracking 결과

〈표 1〉 CPU vs. GPU 처리 속도 비교

	CPU	GPU	비율
Feature detection	39ms	5ms	7.8
Feature initialization	431ms	11ms	39
Active feature measurement	1092ms	9ms	121

5. 결 론

본 연구에서는 연산량이 많은 영상처리알고리즘을 GPU를 이용하여 연산하도록 병렬화 구조로 개발하였다. 이를 통해 실시간 연산이 어려운 카메라 모션 추적 시스템을 실시간으로 동작할 수 있도록 구현하였다. GPU를 이용하므로써 처리속도를 비약적으로 향상시킬 수 있게 되었다.

[Acknowledgement]

This work was supported by the IT R&D program of MKE/IITA and MCST/KOCCA. [2007-S-051-02, Software Development for Digital Creature].

[참 고 문 헌]

[1] Andrew J. Davison, Ian D. Reid, “MonoSLAM: Real-Time Single Camera SLAM”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, No. 6, pp. 1052-1067, 2007.
[2] Nvidia, “Nvidia CUDA Programming Guide”, 2008.