GPU 기반의 해무제거 기술

최운식* · 하준* · 윤우상** · 곽재민* · 최현준*

*목포해양대학교, **미디어디

Technique of Sea-fog Removal base on GPU

Woonsik Choi* · Jun Ha* · Woosang Youn** · Jaemin Kwak* · Hyunjun Choi*

Mokpo National Maritime University, **MediaD

E-mail : hjchoi@mmu.ac.kr

요 약

본 논문에서는 해무제거를 통해 연안을 운항하는 선박의 시계를 확보함으로써 안전 항해에 도움을 줄 수 있는 알고리즘과 GPU기반의 구현결과를 보인다. 최근 세월호 사건을 계기로 해양 사고 및 선박 안전에 대한 관심이 높아지고 있다. 통계에 따르면 연안 선박의 해양사고 원인 중 해무가 있을 시 선박의 시계확보가 이루어지지 않아 선박간의 충돌 및 여러 가지 사고의 발생이 높은 비율을 차지하는 것을 볼 수 있다. 해무가 존재하는 이미지를 위한 알고리즘은 여러 가지가 연구 되고 있다. 하지만 이러한 연구들은 알고리즘을 수행하는 과정에서 많은 연산량을 차지한다. 따라서 본 논문에서는 GPU 기반의 해무제거 기술을 통해 연산 속도를 개선시켜 실시간 영상에 적합하도록 하였다. GPU를 이용하여 구현한 결과 약 250배 정도 연산속도가 향상된 것을 확인할 수 있었다.

ABSTRACT

This paper propose the help of the secure a clear view and safe navigation of the coastal ship through the sea-fog removal algorithm. Interest in marine accidents and vessel safety has increased in recent Sewol ferry event. According to statistics coastal ship cause of the marine accident when sea fog on the sea did not secure clear view the ship's occur several incidents of collisions between ships and can see that accounts for a high percentage. Algorithm for image exist sea fog is number of studies. but, such studies take up a lot of calculation quantity in the course of performing the algorithm. In this paper, we improve the computational speed of sea fog over the GPU-based technique was removed to suit real-time video. Furthermore, by using GPU, we succeeded in accelerating the simulation 250 times.

키워드

Sea fog, Haze removal, GPU, CPU

Ⅰ. 서 론

해상에 해무가 발생할 경우 항해사의 시계확보가 이루어지지 않아 여러 안전상의 문제가 발생하게 된다. 이런 시계저하 문제를 해결하기 위해 디지털 영상기기들을 이용하여 영상신호를 획득한 후 디지털 영상처리 기법을 이용하는 방법들이 제안되었다.

최근에 발표된 안개제거 기술은 두 개의 서로 다른 편광렌즈 및 카메라를 이용[1]하거나, 안개가 없는 날 고정된 장소의 영상을 이용하는 방법[2]등이 있다. 하지만 이러한 방법들은 두 개 이상의 영상이 필요하거나 공간 정보에 대한 데이터가 필요하다는 문제점을 가지고 있다. 따라서 최근 연구되고 있는 안개제거 기술들은 단일 영상만을 이용하여 추세이다. 그러나 단일 영상을 이용하는 DCP(Dark Channel Prior) 알고리즘[3]은 bilateral 필터를 이용하는 정련과정에서의 연산량이 과다하여 실시간 처리에 걸림돌이 되고 있다.

본 논문에서는 DCP 알고리즘을 GPU를 기반으로 구현하여 연산 속도를 개선시키는 기술을 제안하였고, 구현결과를 해무영상에 적용하여 연안을 항해하는 선박의 안전항해 기술로의 가능성을 확인하였다.

Ⅱ. DCP 알고리즘

해무가 낀 영상은 Narsimhan[4] 등이 제안한 알고리즘에서 식(1)을 사용한다.

$$I(x) = J(x)t(x) + A(1-t(x)) \quad (1)$$

여기서 *I(x)*는 입력된 영상, *A*는 해무의 밝기 정도를 나타내며, *t(x)*는 전달량 으로 빛이 산란되지 않고 카메라까지 도달된 정도를 나타낸다. *J(x)*는 해무가 제거된 영상의 밝기 값이다. 따라서 해무 제거 영상처리는 입력 영상 *I(x)*로부터 *A*, *t(X)*, 그리고 *J(x)*를 구하는 것이다.

본 논문에서는 해무 제거를 위해 DCP 알고리즘을 사용한다. Dark Channel 이란 영상에서 일정한 패치사이즈 안에서 한 화소의 RGB값 중에 가장 작은 값의 집합으로 정의한다. 또한 해무가 없는 깨끗한 영상에서는 각 화소의 RGB값 중에 한 값이 0에 가까운 값이 되는 것을 볼 수 있다. 이를 식으로 표현하면 다음의 식 (2)와 같다.

$$J^{dark}(x) = \min_{c\in r,g,b}(\min_{y\in\Omega(x)}(J^c(y))) \quad (2)$$

식 (2)에서 J^c는 영상의 각 채널을 의미하고, $\Omega(x)$는 y점을 중심으로 한 일정 구간을 나타낸다. 안개가 없는 일반적인 영상의 Dark Channel 0에 가까우므로 아래 식(3) 같이 표현된다.

$$J^{dark} \approx 0 \quad (3)$$

식 (1)의 양변에 min 연산자를 취해주고, A값으로 나누어 주면 아래와 같은 식(4)를 얻는다.

$$\min_c(\min_{y\in\Omega(x)}(\frac{I^c(y)}{A^c})) = \tilde{t}(x)\min_c(\min_{y\in\Omega(x)}(\frac{J^c(y)}{A^c})) + (1-\tilde{t}(x)) \quad (4)$$

식 (2) 와 (3)에 의해 식 (4)의 우측 항 중 J가 들어간 항을 0으로 가정할 수 있으므로 결과적으로 다음의 최종 식(5)를 얻을 수 있다.

$$\tilde{t}(x) = 1-\min_c(\min_{y\in\Omega(x)}(\frac{I^c(y)}{A^c})) \quad (5)$$

해무 값 A는 He[3]가 실험적으로 연구한 결과를 바탕으로 해무영상의 Dark Channel 중에서 가장 밝은 곳으로부터 10% 밝기를 가지는 픽셀의 값을 선택한다.

식 (2) 와 (3)을 정리하면 최종 해무가 제거된 다음의 식 (6)을 얻는다.

$$J(x) = \frac{I(x)-A}{\max(t(x),t_0)} + A \quad (6)$$

하지만 예상한 전달량 t(x)로 해무를 제거할 경우 후광 효과가 나타나게 된다. 따라서 본 논문에서는 정련 과정에서 bilateral필터를 사용하여 후광효과를 해결하였다. bilateral 필터를 한회 수행한 화면은 후광효과가 사라지긴 하지만 일정량이 남아있다. 이러한 후광 효과를 제거 하기위해 반복적으로 bilateral 필터를 수행하면 경계선이 점점 선명해 지면서 블록효과로 발생하는 후광효과가 사라지게 된다.

Ⅲ. GPU 구현

3.1 GPGPU(General Purpose computing on Graphic Processing Units)

CUDA는 쓰레드(thread)와 블록(block), 및 그리드(grid) 단위로 GPU내의 연산 단위를 구분한다. 하나의 블록은 3차원의 쓰레드로 구성할 수 있고, 하나의 그리드는 3차원의 블록으로 구성할 수 있다. 각 쓰레드는 GPU내의 SP에 맵핑되며 연산하는 최소 단위로 하나의 블록 내의 최대 쓰레드의 개수는 GPU의 성능에 따라 결정된다. 또한 블록 내에 구성된 쓰레드는 인덱스를 가지고, 워프(warp) 단위로 실행된다. 따라서 모든 쓰레드를 효율적으로 동작시키기 위해서는 블록 당 쓰레드의 개수를 워프의 배수로 하는 것이 유리하다. 또한 예외처리가 필요한 부분에서는 워프 단위 내에서는 같은 조건으로 예외 처리하는 것이 동시 실행의 병렬성을 유지할 수 있다.

3.2 Optimization in Host Program

가. Memory & Variable Sharing

시스템을 구성할 때 입력 영상(카메라로부터 입력)의 크기가 고정되어 있다면 매 프레임마다 메모리 할당(초기화 과정)을 수행하는 것은 필요 없는 과정이다. 따라서 이 과정은 시스템이 처음 시작될 때 한 번의 초기화 과정을 통하여 모든 메모리를 할당하고 시스템이 종료되는 시점에서 한 번의 해제를 하는 것이 시스템의 성능 저하를 줄일 수 있다.

나. Pined Memory Allocation

일반적인 메모리 할당을 했을 때 영상과 같은 큰 메모리를 할당할 경우 가상 메모리 기술에 의하여 보조 기억장치(HDD)에 도움을 받아 해결을 한다. 이는 컴퓨터 혹은 운영체제에 의해서 제어되는데 현재 사용하는 메모리의 데이터를 DRAM에 올리고 사용하지 않는 메모리의 데이터를 보

조 기억장치로 가상으로 큰 메모리 공간을 페이지 단위로 분할하여 제공한다. 현재 사용하지 않는 데이터의 물리적 주소는 보조 기억장치로 이동하는데 이는 GPGPU를 이용한 병렬 프로그래밍을 수행할 경우 두 번의 메모리 복사를 수행한다. 이러한 제약을 해결하기 위해 CUDA API의 고정된 메모리(Pined Memory)를 이용하여 성능을 향상 시킬 수 있다.

3.3 Optimization in Device Parallel Program

가. Shared Memory

GPU로 수행되는 Bilateral 필터의 밝기에 의한 커널 $f_r(\|I(x_i)-I(x)\|)$는 입력 영상에 따라 변하는 함수 이므로 글로벌 메모리로 접근하여 계산을 하여야 하지만 좌표에 의한 커널인 $g_s(\|x_i-x\|)$의 x_i와 x는 정수형태의 데이터 이므로 최대 커널의 크기에 따라 LUT(Look-up table) 형태로 공유메모리를 이용할 수 있다.

나. Asynchronous Operation

CUDA API에서는 비동기 수행을 지원하는데 스트림(Stream)을 통하여 비동기로 수행하고, 호스트는 디바이스의 동작과는 독립적인 동작을 수행할 수 있다. 독립적인 동작이 종료된 후에 스트림 동기화(Synchronous Stream)을 통하여 디바이스에서 수행된 결과를 획득 할 수 있다.

Ⅳ. 결 론

실험에 사용된 PC는 Intel i7 CPU와 16GB의 호스트 메모리의 PC를 사용하였고 GPGPU는 NVidia의 GTX680을 이용하여 구현하였다. 그림 1은 실험에 사용한 GPGPU의 특성을 나타내었다. GTX 680은 계산 능력은 3.0으로 Fermi 구조이고 코어 클럭은 1.6GHz에 메모리의 밴드폭은 256bit이고 글로벌 메모리는 2GB이다. 또한 공유 메모리의 크기는 48KB이고 워프 크기는 32이다. 표 1은 실험 영상을 이용하여 각 단계별 평균 계산시간을 나타내었다. Dark Channel을 찾기 위한 윈도우 크기는 9x9이고, bilateral 필터를 위한 커널의 크기는 15x15로 하였다. 반복정제를 위한 반복수는 3번으로 제한하였다. 호스트의 메모리 공유 및 고정 메모리 사용시 약 5배의 성능 향상을 보였고, GPU를 이용한 병렬 처리를 하였을 때에는 약 250배의 성능 향상을 보였다.

```
**************************************************
  Device 0 - Name                : GeForce GTX 680

  ------ Core   Properties ------
  Computation Capablility        : 3.0
  Core Clock Freq.               : 1.06 GHz
  Number of Registers(SM/Block)  : 65536 / 65536 EA
  Support L1 Cache(Global/Local) : n / y
  L2 Cache Size                  : 512 KB
  Number of Multi-Processors     : 8 EA
  Number of Async. Engines       : 1 EA
  Support Concurrent Kernels     : y
  Support ECC                    : n
  Support Stream Priority        : n

  ------ Board  Properties ------
  PCIe Bus ID                    : 1
  PCIe Device ID                 : 0
  PCIe Domain ID                 : 0
  Multi-GPU on Board             : n
  Multi-GPU Group ID             : 0

  ------ Memory Properties ------
  Memory Clock Freq.             : 3.00 GHz
  Memory Bus Width               : 256 bits
  Global Memory Size             : 2048 MB
  Shared Memory Size(SM/Block)   : 48 / 48 KB
  Constant Memory Size           : 64 KB

  ------ Thread Properties ------
  Number of Max Threads(SM/Block): 2048 / 1024 EA
  Max Dimension of Blocks(x,y,z) : 1024, 1024, 64
  Max Dimension of Grids(x,y,z)  : 2147483647, 65535, 65535
  Warp Size in Threads           : 32 EA
**************************************************
```

그림 1. 실험에 사용된 GPGPU의 성능

표 1. 성능 향상표

내용	평균 수행 시간 [ms]
원본 코드	59,533.78
호스트 메모리 공유 및 고정 메모리 사용	11,710.67
GPGPU를 이용한 수행	245.97

감사의 글

본 논문은 중기청 산학연협력 첫걸음 기술개발 사업의 일환으로 수행하였음.

참고문헌

[1] S.K. Nayar and S.G. Narasimhan, “Vision in bad weather,“ in Proc ICCV, pp. 820-827, Sep. 1999

[2] S.G. Narasimhan and S.K Nayer. “Contrast restoration of weather degraded images,” IEEE Trans. PAMI, vol. 25, no. 6, pp. 713-724, Jun. 2003.

[3] K. He, J.Sun, and X.Tang, “Single image haze removalusing dark channelprior“ inProc. CVPR, pp. 1956-1963, Miami, USA, June2009

[4] S.G. Narasimhan, S.K. Nayar, “Chromatic framework for vision in bad weather“ CVPR, pages 598-605, 2000