Obstacle Detecting
for Unmaned Vehicle
using CUDA

프로젝트 초안 계획서

200412356
소프트웨어
최 동 민

개요 및 목적

간단한 자동 수평주차가 가능한 시스템을 내장한 자동차를 실제로 양산하고 있는 정도로 Intelligence Vehicle System(IVS)은 우리 생활에 점차 가까워지고 있다. 자동차 보유량의 급격한 증가와 운전자 계층이 다양해짐으로써 자동차의 안전성을 운전자 개인의 능력에만 의존하는 수동적인 방법에서 탈피해, 운전자의 시각 및 지각의 한계를 보완해 줄 수 있는 시스템으로 각광받는 추세이다.

이러한 IVS를 내장한 스스로 움직이는 자동차 혹은 비행체의 개발에 있어서 가장 문제가 되는 것 중 하나는 장애물을 인공 지능이 어떻게 인식할 수 있는가 하는 문제이다. 이것은 어느 한 가지만 개발되어서 해결 될 문제가 아니기에 divide and conquer방식으로 하나씩 해결해 나가도록 한다. 이번 프로젝트에서는 목표를 다음으로 한정한다.

1. 이미지 상에서 장애물 혹은 선행 차량의 위치를 측정하고 이를 marker하기
2. 실시간 처리를 위해 CUDA(Compute Unified Device Architecture)를 기반으로 한 분산 처리

프로젝트 이론

알고리즘

여러 현재까지 관련된 연구에서 다양한 방식을 참조할 수 있었다. 여기서 어떤 것이 가장 효과적인지, 그리고 CUDA기반의 프로세싱에 적합한지를 따져보았다.

1. Edge 잡기

(Hao-Yuan Chang "Real-Time vision-based preceding vehicle tracking and recognition",IEEE,2005)

Obstacle 검색 구간을 도로 이미지로 한정한 후, symmetric edge를 찾아서 obstacle을 인식

- **장점**

- 카메라 한대로 obstacle의 bounding box를 추출해낼 수 있다.

- **단점**

- 장애물과과 카메라 사이의 거리 측정 시 이미지에서 vanishing point가 검출되어야 한다는 가정이 필요하다.

- symmetric obstacle만 추출해낼 수 있다.

- 비 오는 날 또는 밤과 같이 카메라가 정상 동작하기 힘든 환경에서는 장애물을 찾기가 힘들다.

2. 그림자 색 따기

(Ming-Yang Chem "Locating Nearby Vehicles on Highway at Daytime based on Front Vision of A Moving Car",IEEE,2003)

도로의 평균 색상을 바탕으로 도로 위에 만들어진 obstacle 그림자 색상을 계산한 후 obstacle을 검출

- **장점**

- 카메라 한대로 쉽고 간단한 방법으로 장애물을 찾아냄

- **단점**

- 맑은 날에만 사용할 수 있는 알고리즘
- obstacle과 카메라의 거리 계산 불가능
- obstacle의 그림자 색상은 도로보다 70% 정도 어둡다는 현실성 없는 가정 사용

3. 두 대의 카메라를 이용하기

(Massimo Bertozzi, "GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection", IEEE Transactions on Image Processing, 1998)

Stereo Vision System을 이용하여 원근정보를 얻어낸 후 obstacle의 위치와 거리를 측정

- **장점**

여러 가지 비현실적인 가정을 배재한 채 obstacle detecion 이 가능

- **단점**

- 두 대의 camera가 필요
- Camera의 정확한 intrinsic, extrinsic parameter를 알고 있어야 함

위에서 언급한 세가지 방법들 중에서

1. 실제 환경에서 사용이 가능한가?
2. 병렬 처리를 요구할 정도의 자료 양을 가지는가?
3. 구현 과정에서 병렬화가 쉬운 알고리즘인가?

를 따져본 결과 3의 알고리즘(GOLD)을 기본으로 2에서 사용한 아이디어를 빌려오기로 하였다.

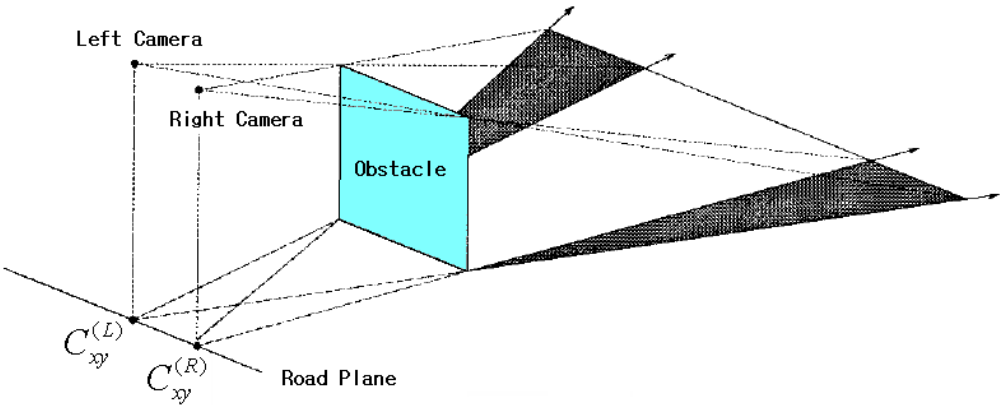

그림 1. 두 대의 카메라가 장애물을 인식하는 상황

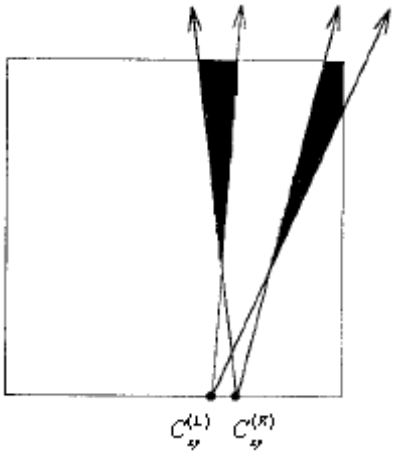

그림 2. 그림 1의 상황을 수직으로 내려다 본 그림(IPM 결과)

병렬처리

어째서 이 문제에서 CPU보다 CUDA를 이용한 연산이 유리한지에 대해서 보자면, 3번 알고리즘(GOLD)에서는 obstacle detection을 위해 위치를 찾아낸다. 이를 위해 두 대의 카메라에서 이미지를 받아온 뒤 이를 IPM(Inverse Perspective mapping)하여 시점을 위에서 아래로 본 이미지로 각기 변환한 뒤, 이 이미지에서의 Difference를 찾아 하나의 이미지로 만들게 된다. 이 과정에서 CPU에서 연산을 할 경우 적은 ALU로 많은 행렬 연산을 요구하게 된다. 이 과정에서 잦은 kernel 호출이 발생하기에 필연적으로 속도가 떨어지게 된다. 하지만 GPU 상에서 처리를 할 경우 많은 ALU와 실제적으로 동시에 동작하는 많은 Thread가 있다. CPU상에서 IPM 처리를 위한 간단한 행렬 계산을 통해 일반적인 디지털 카메라의 깨끗한 출력물을 돌려본 결과 약 3분이 소요되었다. 이런 상황에서 실시간 처리를 위해 추후 동영상으로 바꾼다면 3분마다 한번 눈을 뜰 수 있는 운전자에 해당하는 시스템인 것이다.

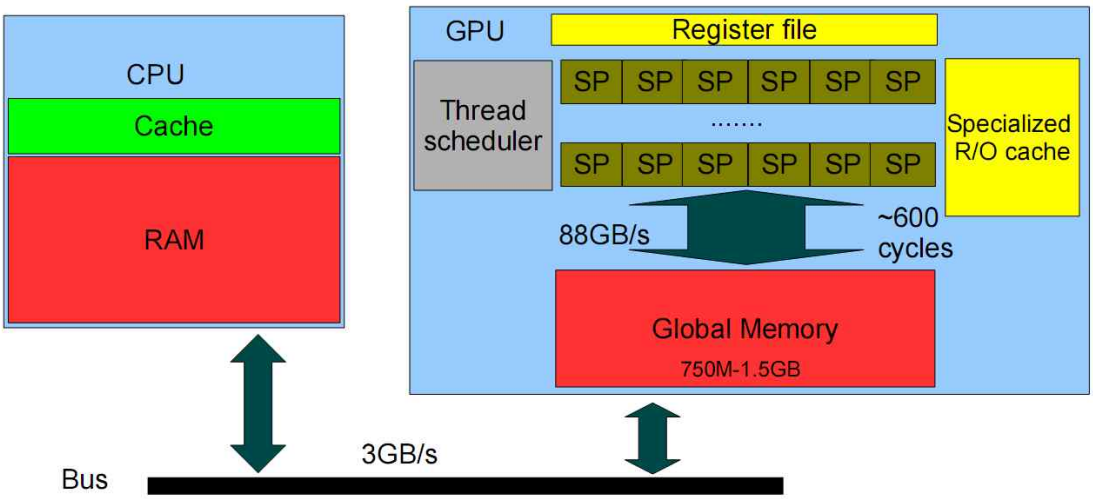

그림 3. 일반적인 환경 하에서의 CPU(RAM포함)와 GPU

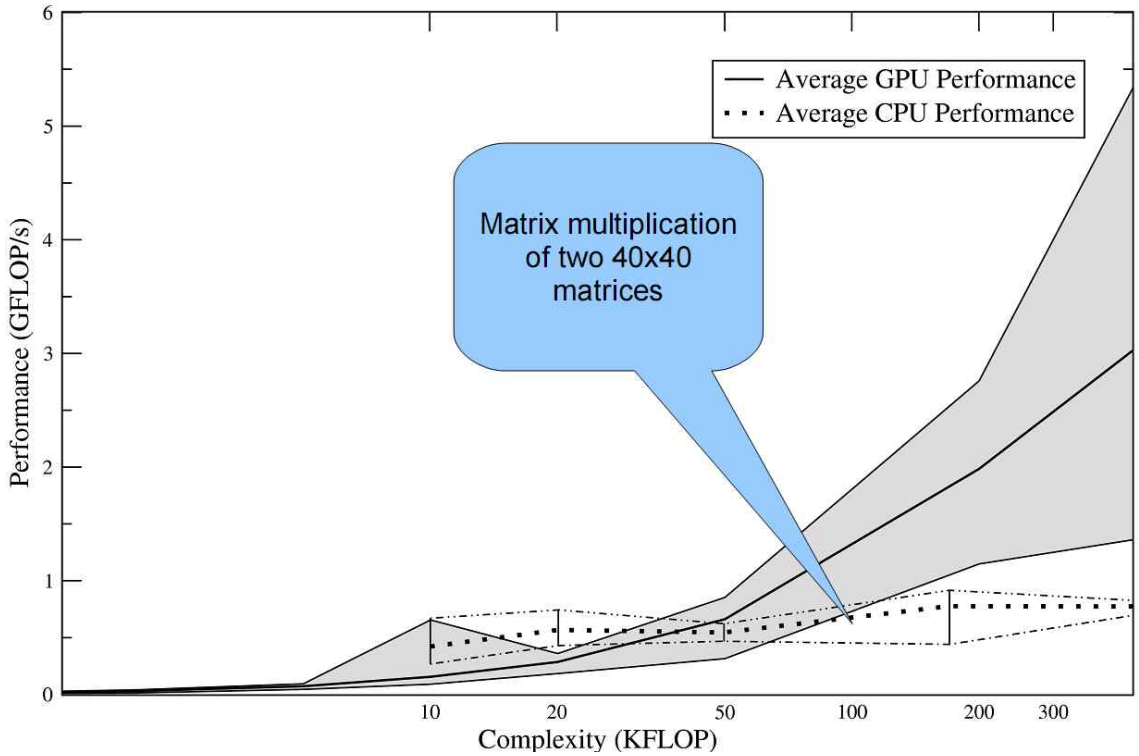

그림 4. 처리할 데이터가 커질 수록 GPU를 통한 분산처리가 효과적

처리순서도

1. Stereo Camera를 통해 2개의 이미지를 입력받는다.
2. 이미지를 그래픽 메모리에 적재한다.
3. IPM(Inverse Perspective mapping)을 CUDA상에서 각기 적용한다.
4. 이를 하나의 difference image로 만들어서 밝기 차이가 특정 값 이상인 부분만 남긴다.
5. Difference image를 이용 camera position을 기준으로 obstacle이 존재하는 angle, distance에 대한 정보를 얻어낸다.
6. Perspective mapping algorithm을 이용하여 obstacle의 크기와 좌표를 찾는다.
7. 찾아낸 obstacle의 시작점을 화면상에 출력한다.

참조문헌

Haifux Linux club, Elbit, GIT lab “CUDA tutorial: High Performance Computing on Graphics Processing Units (GPUs)”

NVIDIA, UC Berkeley, UC San-Francisco, IBM Haifa Research Labs, Intel Israel, Clubnet EE Technion, Cisco Israel “Efficient sum-product computations on GPUs“

Andrzej Rusiecki "Robust LTS Backpropagation Learning Algorithm", Springer Berlin

Hao-Yuan Chang “Real-Time vision-based preceding vehicle tracking and recognition”, IEEE, 2005

Ming-Yang Chem “Locating Nearby Vehicles on Highway at Daytime based on Front Vision of A Moving Car”, IEEE, 2003

Massimo Bertozzi, “GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection”, IEEE Transactions on Image Processing, 1998

Massimo Bertozzi, “Stereo Inverse Perspective Mapping: theory and applications”, IEEE Image Vision Conputing, 1988

이진우, “CCD 카메라를 이용한 차선인식과 차량 위치 검출에 관한 연구”,동아대학교 정보기술연구소 논문지, 2001