한국컴퓨터정보학회 동계학술대회 논문집 제19권 제1호 (2011. 1)

Cuda를 이용한 가우시언 믹스처 모델 기반 객체 추적 알고리즘

김인수°, 최형일*
°*숭실대학교 미디어학과
e-mail:{yes4627, hic}@ssu.ac.kr

Object Tracking Based on Gaussian Mixture Model Algorithm by Using Cuda

In-su Kim°, Hyung-Il Choi*
°*Dept. of Media, Soongsil University

● 요 약 ●

본 논문에서는 효과적인 객체 추적을 위해 가우시언 믹스처 기반의 그림자 제거 알고리즘을 제안하고, GPGPU(General Purpose GPU) 아키텍처인 NVIDIA 사의 CUDA(Compute Unified Device Architecture)를 이용하여 기존의 객체 추적 알고리즘의 컴퓨팅 시간을 개선하는 모델을 제안한다. 이 시스템은 GPU를 이용한 가우시언 믹스처 모델 기반의 객체 추적 알고리즘으로 전경과 배경 분리 시 CPU와 GPU의 프로세스 시간을 적절히 분배하여 소모되는 연산시간을 줄이고, 고 해상도의 이미지에서의 객체 분리 및 추적의 시스템 처리량을 최대화 한다. 객체 추출 후 효과적인 추적을 위해 예측 모델인 칼만 필터를 사용한다.

키워드: 그림자 제거(Shadow Removal), 쿠다(CUDA), 가우시언 믹스처 모델(Gaussian Mixture Model), 객체 추적(Object Tracking)

I. 서 론

최근 CPU의 다중 코어의 한계로 인해 기존의 그래픽 처리 용 GPU의 활용이 높아짐에 따라 NVIDIA 사에서는 GPGPU (General Purpose GPU) 아키텍처인 CUDA(Compute Unified Device Architecture)를 발표하였다. CUDA API는 프로그래머에게 GPU 구조에 대한 특별한 지식이 없이도 C/C++언어와 유사한 문법을 제공함으로써, 프로그래머에게 친숙한 병렬 처리 개발 환경을 제공한다. 그러나 일반적으로 CPU 기반 프로그램에서 GPU 기반 프로그램으로의 단순한 변환만으로는 큰 성능 향상을 기대할 수 없다[1]. 본 논문에서는 CUDA를 이용하여 구현된 가우시언 믹스처 모델 기반의 전경 배경 분리기법에서 그림자 제거 알고리즘을 추가하여 객체 추적 알고리즘의 전체 연산 시간을 줄이고, 이와 더불어 고 해상도에서 실시간으로 객체를 추적에서의 그림자 영역 문제를 해결하고, 그림자 제거 CUDA 프로그래밍 기법과 이를 사용한 효과적인 객체 추적 알고리즘을 제안한다.

II. CUDA를 이용한 가우시언 믹스처 모델

기본적으로 배경 화소는 하나의 화소 값을 가지며 조명의 변화나 영상 취득 장치의 잡음에 따라 약간의 변화를 보인다. 이러한 배경의 값은 적응적 가우시안 확률 분포로 효과적으로 모델링 할 수 있으며, 객체 추적 모델에서의 전경 배경분리에 효과적인 알고리즘으로 Stauffer가 제안한 적응적 가우시언 혼합 모델을 사용한다[2].

CUDA를 이용한 가우시언 믹스처 모델 구현은 Janaka가 제안한 GPU기반의 배경 전경 분리 알고리즘을 사용한다[3]. GPU 기반 가우시언 믹스처 모델의 메모리 구조는 그림1과 같다.

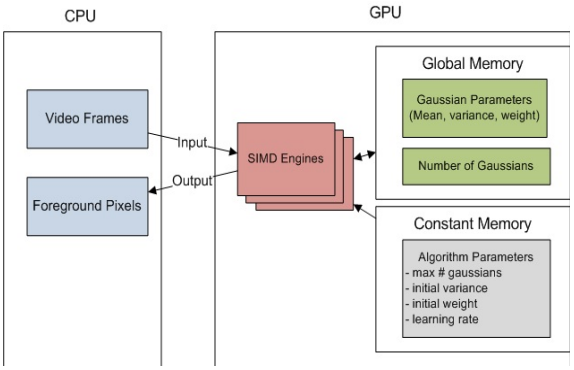

그림 1. 가우시언 믹스처 모델에서 CPU와 GPU의 메모리 구조
Fig. 1. Memory Architecture of CPU and GPU on Gaussian Mixture Model

CUDA 언어를 이용한 프로그래밍 코드는 그림 2와 같다.

```
int cuda_update(CGMMImage2* pGMM, pUINT8 imagein, pUINT8 imageout)
{

    //wait for the previous memory operations to finish
    cudaStreamSynchronize(pGMM->copyStream);
    //copy into and from pinned memory
    memcpy(pGMM->pinned_in, imagein, ....);
    memcpy(imageout, pGMM->pinned_out, ....);

    //make sure previous exec finished before next memory transfer
    cudaStreamSynchronize(pGMM->execStream);
    //swap pointers
    swap(&(pGMM->d_in1), &(pGMM->d_in2));
    swap(&(pGMM->d_out1), &(pGMM->d_out2));
    //copy the input image to device
    cudaMemcpyAsync(pGMM->d_in1, pGMM->pinned_in, ...., pGMM->copyStream);
    cudaMemcpyAsync(pGMM->pinned_out, pGMM->d_out2, ...., pGMM->copyStream);

    //call kernel
    backSubKernel<<<gridB, threadB, 0, pGMM->execStream>>>(pGMM->d_in2, pGMM->d_out1, ...);
    return 0;
}
```

그림 2. 가우시언 믹스처 모델의 CUDA 구현 예제
Fig. 2. Example of GMM Based on Cuda

III. 그림자 제거 알고리즘

그림자는 배경과 같은 방향의 색상분포를 갖고, 밝기는 어둡다는 특성을 갖는다. 다음 그림은 그림자와 배경의 관계를 나타낸다.

$$\alpha = B_r/r + B_g/g + B_b/b \tag{1}$$

$$\beta = ((B_r - r)/std)^2 + ((B_g - g)/std)^2 + ((B_b - b)/std)^2 \tag{2}$$

즉, α는 배경 모델과의 방향의 차이를 나타내고, β는 밝기의 차이를 나타낸다. $B_{r,g,b}$는 배경모델의 평균 RGB값이고, r,g,b는 입력 영상의 색상 값 std는 배경모델의 표준 편차를 나타낸다. 구해진 α와 β는 다음 식에 의해 그림자 여부를 판단한다.

$$Shadow(x,y) = \begin{cases} 0 & \alpha < T_\alpha,\ \beta < T_\beta \\ 1 & otherwise \end{cases} \tag{3}$$

본 논문에서 가우시언 믹스처 모델에서 그림자 제거 알고리즘을 사용하기 위해 $B_{r,g,b}$의 값과 std의 값은 σ/ω로 정렬된 가우시언 모델의 0번째 평균 R, 평균 G, 평균 B값과 표준편차의 값을 사용한다. 적용된 수식은 다음과 같다.

$$\alpha = G^0_{meanR/r} + G^0_{meanG/g} + G^0_{meanB/b} \tag{4}$$

$$\beta = ((G^0_{meanR} - r)/G^0_{std})^2 + ((G^0_{meanG} - g)/G^0_{std})^2 + ((G^0_{meanB} - b)/G^0_{std})^2 \tag{5}$$

추출된 객체 영역에서 판별된 그림자 영역은 0으로 지우고 최종적으로 그림자가 제거된 전경영역만 결과 값으로 Host 메모리로 값을 반환 한다. 다음 그림은 그림자 제거 알고리즘을 추가한 시스템 흐름도를 나타낸다. T_α, T_β 값은 그림자를 판별하는 임계값으로 적절한 값을 사용한다.

IV. 실험 결과

본 실험을 위해 사용한 컴퓨터는 Intel Pentium Dual Core CPU 3.40GHz, 1.70GHz과 2.00Gb RAM 그리고 CUDA에 사용된 그래픽 카드는 nVidia GTS 250을 사용하였다. OS는 Windows7 32bit를 사용하였다. 사용 언어는 Microsoft Visual Studio 2005, CUDA 3.1을 사용하였다. 실험에 사용된 가우시언 믹스처 모델의 수는 3개를 사용하였고, 배경과 전경 분리 시 사용한 임계값은 0.5, 학습률은 0.01, 본 논문에서 제안하는 그림자 제거의 임계값은 T_α = 4.3, T_β = 30을 사용하였다. 객체 추출 후 효과적인 추적을 하기 위해 CPU에서 예측 모델인 칼만 필터를 사용한다[4].

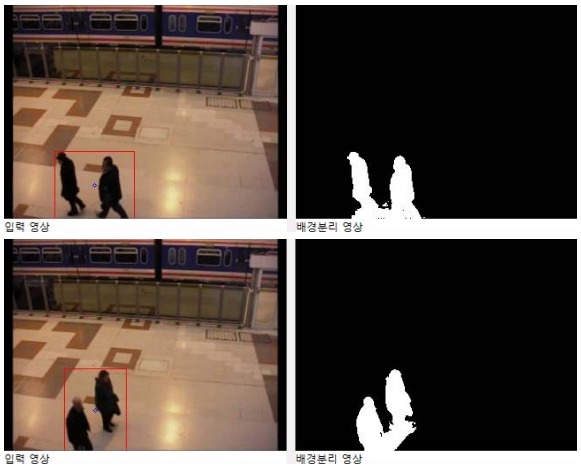

그림 3. 그림자 제거 전 배경 분리 영상
Fig. 3. Background Subtraction Image Before Shadow Elimination

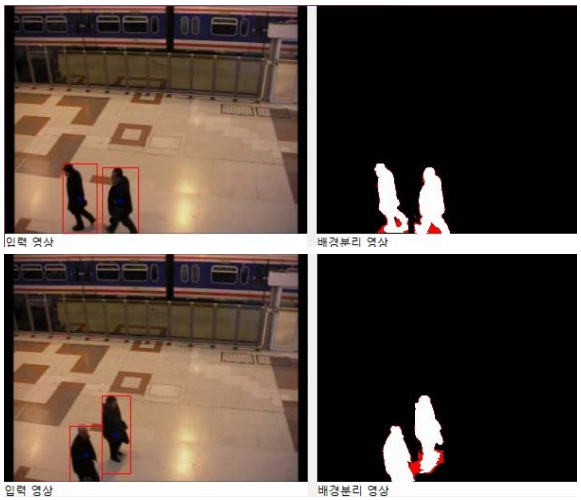

그림 4. 그림자 제거 후 배경 분리 영상
Fig. 4. Background Subtraction Image After Shadow Elimination

그림 3에서 그림자 영역에 의해서 두 객체가 하나의 객체로 추출 되었지만, 그림자를 추출하여 제거한 그림 4에서는 객체가 제

대로 분리된 결과를 확인할 수 있다.

표 1. 그림자 제거 알고리즘 정확도
Table 1. Accuracy ratio of Shadow Removal Algorithm

항목	정확도
그림자 제거	75%

표 1은 그림자 제거 알고리즘의 정확률을 측정한 결과이다. 정확률은 샘플 프레임 150장에서 그림자 영역과 추출된 영역의 차이로 비교 하였다.

$$Accurate\ Ratio = ER/SR_{true} \quad (6)$$

ER은 그림자 영역으로 추출된 영역의 픽셀 수이고, SR_{true}은 실제 그림자 영역의 픽셀 수이다.

표 2. 그림자 제거 알고리즘 연산 시간 비교
Table 2. Computing Time of Shadow Removal Algorithm
(단위 : ms)

그림자 제거 (320x240)	연산시간	
	CPU	GPU
전	22	25
후	45	30

표 2는 이미지 해상도 320x240에서의 CPU와 GPU의 그림자 제거 알고리즘 사용전과 후의 연산시간을 측정한 결과이다. CPU에서의 그림자 제거 알고리즘을 사용했을 때의 연산시간이 2배 이상 느려지는 결과를 얻었고, GPU에서는 그림자 제거 전과 후의 차이가 0.2배 차이가 나는 결과를 얻었다.

표 3. 그림자 제거 알고리즘 연산 시간 비교
Table 3. Computing Time of Shadow Removal Algorithm
(단위 : ms)

그림자 제거 (640x480)	연산시간	
	CPU	GPU
전	123	30
후	150	32

표 3은 이미지 해상도 320x240에서의 CPU와 GPU의 그림자 제거 알고리즘 사용전과 후의 연산시간을 측정한 결과이다. GPU에서 이미지 해상도가 증가 할 수록 연산 시간의 차이가 줄어드는 결과를 볼 수 있다.

V. 결 론

본 논문에서는 CUDA를 이용한 가우시언 믹스처 모델에서 연산량이 많은 그림자 제거 알고리즘을 추가하여, 기존의 CPU 기반의 가우시언 믹스처 모델에서의 속도를 개선하고, 효과적으로 그림자를 제거하여 객체 추출을 하였다.

참고문헌

[1] Shane Ryoo, Christopher I. Rodrigues, Sara S. Baghsorkhi, Sam S. Stone, David B. Kirk, and Wen-mei W. Hwu, Optimization Principles and Application Performance Evaluation of a Multi-Threaded GPU Using CUDA, Proc. 13th ACM SIG-PLAN Symp. Principles and Practice of Parallel Programing, ACM Press, 2008.

[2] C.Stauffer and W.E.L Grimson “Adaptive Background Mixture Models for Real-Time Tracking”, Proc. Conf. Computer vision and Pattern Recognition, vol. 2, pp. 246~252, June 1999.

[3] Janaka Liyanage, “GMM based Background Subtraction on GPU” CDA 6938: Project Technical Report.

[4] Weissman, T., “EE378 Handout: Kalman Filter”, Lecture notes on EE378 Statistical Signal Processing, http://eeclass.stanford.edu/ee378/.