GPU를 이용한 스테레오 정합 알고리즘 구현

*최현준 **최지윤 ***서영호 ***김동욱
*안양대학교, **한양대학교, ***광운대학교
***dwkim@kw.ac.kr

Implementation of Stereo Matching Algorithm Using GPU

*Choi, Hyun-Jun **Choi, Ji-Youn ***Seo, Young-Ho ***Kim, Dong-Wook
*Anyang Univ., **Hanyang Univ., ***Kwangwoon Univ.

요약

본 논문에서는 최종 변이영상의 정확도를 높이기 위해 영상의 특징점을 이용한 적응적 가변 정합창 방법과 교차 일치성 검사의 신뢰도를 높이는 방법을 제안한다. 제안한 적응적 가변 정합창 방법은 색상정보를 이용하여 영상을 분할하고 분할된 각 영상의 특징점을 찾아 그 특징점들의 유무에 따라 정합창의 크기를 적응적으로 가변시키는 방법이다. 또한 제안한 알고리즘을 GPU를 기반으로 구현하여 연산속도가 평균 128배 빨라졌다.

1. 서론

영상의 정합(matching)은 스테레오 비전의 중요한 과제로서, 3차원 공간상의 한 점이 좌우 두 영상에 투영되는 점의 대응성(correspondence)을 찾아내는 과정이다[1-3]. 스테레오 비젼에서 이러한 과정을 스테레오 정합(stereo matching)이라 부르며 두 영상에서 선택된 점들이 대응점을 가지는가를 판단하고 그 점에서의 깊이 정보의 지표가 되는 변이(disparity)를 계산하는 것이 핵심이다. 이러한 변이 정보를 이용하여 3차원 공간의 표면을 기술할 수 있다.

영상의 대응점간의 유사성을 통해 변이 값을 찾는 방법으로 크게 전역 기반 방법(global method) [4-6]과 지역 기반 방법(local method)[7]으로 나눌 수 있다. 전역 기반 방법은 영상전체를 대상으로 에너지 함수를 최소화하는 변위 값을 찾음으로써 정합을 수행한다. 이 방법은 높은 정확도를 가지지만, 다른 스테레오 방법에 비하여 상대적으로 많은 연산이 요구되어 실시간 구현에 어려움이 있다. 지역 기반 방법은 영상의 국부에 해당하는 정합창(matching window) 내의 컬러와 밝기 값을 통해 각 화소의 변위 값을 찾게 된다. 전역 기반 방법에 비해 간단하면서 효율성이 높으나, 균일한 색상을 가지는 영역(homogeneous region)과 객체의 깊이 불연속(depth discontinuities) 등에는 좋지 않은 결과를 발생시킨다.

이러한 어려움을 극복하기 위해 균일한 색상을 가지는 영역과 객체의 깊이 불연속 등에 정확도를 높이면서 전역 기반 방법에 비해 연산 량이 적은 방법들이 제안되고 있다. 이와 같은 방법은 적절한 정합창의 크기와 모양을 화소에 특징에 맞게 선택함으로써 정확도를 높이게 된다.

최근 3차원 그래픽스 처리를 위한 가속기 하드웨어가 발표된 이후 일반 사용자용 GPU(Graphic Processing Unit)의 계산 성능은 CPU보다 빠른 속도로 발전해왔다. 특히 shading language 언어를 이용한 프로그래밍이 가능하게 되어 GPU 내부의 기능을 변경하여 사용할 수 있게 되었다. 이를 기반으로 3차원 그래픽 처리뿐만 아니라 다양한 분야에서 GPU의 대용량 병렬처리를 사용할 수 있게 되었다[8].

본 논문에서는 GPU를 기반으로 스테레오 영상으로부터 변이를 계산하는 알고리즘과 구현결과를 소개한다. 이 알고리즘은 초기 변이 지도의 정확도 향상을 위해 색상정보를 이용하여 특징점을 추출한 뒤 이를 통해 영역별로 정합창의 크기를 달리 적용하는 적응적 가변 정합창 방법이다. 제안한 알고리즘은 NVIDIA사의 CUDA(Compute Unified Device Architecture)를 사용하여 구현하였다.

2. 제안한 알고리즘

2.1 제안한 영역 기반 스테레오 정합 방법

① 색상정보를 이용한 특징점 추출

정합창의 크기에 따라 정합창의 크기에 따라 오정합이 발생하는 영역이 다른 문제점을 해결하기 위하여 본 논문에서는 영역에 따라 정합창의 크기를 적응적으로 변화시키는 방법을 제안한다. 영역은 경계영역과 그 외의 영역으로 구분하는데, 이것은 영상의 특징점들을 찾음으로써 가능하다. 본 논문에서는 영상의 에지들을 영상의 특징점들로 사용한다.

영상의 에지들을 찾기 위해서 본 논문에서는 먼저 영상을 영역분할하고 각 분할된 영역의 에지들을 찾아 그 에지들을 결합하는 방법을 사용하였다. 영역을 분할하는 방법은 생상정보를 사용하는 방법으로서, 기본적으로는 R, G, B, W(white)의 네 영역으로 분할하는 방법을 사용하였다.

그러나 일반작인 영상 자체에서 추출한 에지들은 그 경계가 모호한 경우가 많다. 따라서 본 논문에서는 먼저 각 영역을 더욱 뚜렷이 구분하기 위해서 평균 이동(mean-shift) 알고리듬을 사용하였다. 이 방

법은 각각의 화소의 위치 정보와 색상 정보를 동시에 고려하여, 비슷한 색상 특징 값을 갖는 영역을 분할할 뿐 아니라, 그 분할된 영역의 대표값 또한 출력한다. 이 방법은 객체와 객체간의 경계 영역을 뚜렷하게 해주는 효과를 준다.

평균 이동 알고리듬을 적용한 영상을 R, G, B, W로 분할한 후, 각 분할된 영상의 에지는 소벨 연산자를 적용하여 구하였다. 전체 영상에 대한 에지영상은 각 분할영상의 에지들을 모두 합하여 구하였다. 그림 1은 Teddy 영상에 대해 소벨 연산자를 적용한 결과, R, G, B, W로 영상분할한 결과, 각 분할영상에서 추출한 에지, 그리고 분하영상의 에지들을 합하여 전체영상의 에지영상을 만든 결과를 예로 보이고 있다.

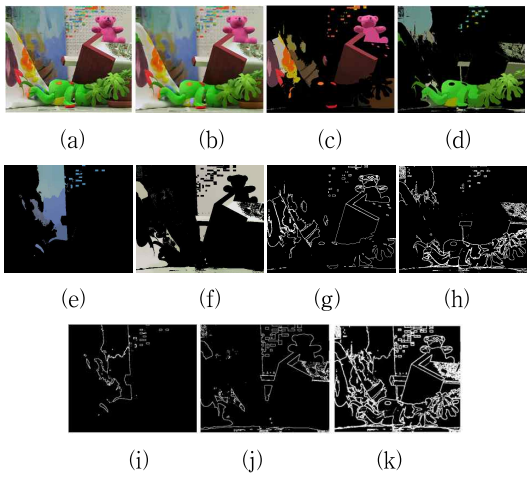

그림 1. 에지영상 추출 ; (a) 원 영상, (b) 평균 이동 알고리즘 적용 결과, 분할된 (c) R 영상, (d) G 영상, (e) B 영상, (f) W 영상, 분할된 영상에서 추출한 에지 (g) R 영상, (h) G 영상, (i) B 영상, (j) W 영상, (k) 결합된 에지 영상

② 특징점들을 이용한 적응적 가변 정합창 방법

정합창을 이용한 스테레오 매칭은 정합창의 크기에 따라 영역별로 정확도가 달라진다. 이러한 특성을 고려하여 초기 변이지도를 생성할 때 영상의 특징점들, 즉 에지성분에 따라 적응적으로 정합창의 크기를 조절하는 방법을 제안한다. 이 방법은 먼저, 3×3 정합창 내에 에지화소(에지에 해당하는 화소) 수를 찾는다. 정합창 내에 에지화소가 있을 경우에는 현재 윈도우의 크기를 정합창의 크기로 선택하고 경계 정보가 없을 경우에는 윈도우의 크기를 증가시킨다. 정합창은 홀수×홀수의 크기로 증가시켜 최대 25×25의 크기를 가지도록 하였다. 이 방법을 사용함으로써 경계영역, 즉 폐색영역을 포함하는 에지성분이 많은 영역에서는 작은 정합창을 갖고, 균일한 영역에서는 큰 정합창을 갖게 된다.

③ 다수로 정합되는 경우의 처리

영역 기반 정합을 수행할 때 기준영상의 특정화소와 정합되는 참조영상의 화소는 이론적으로는 한 개이어야 한다. 그러나 최적의 비용함수 값을 갖는 참조영상의 화소가 두 개 이상이 되어 정합되는 화소수가 두 개 이상인 경우가 많이 발생한다. 이것은 적응적 가변 정합창을 사용하는 제안한 방법에서도 마찬가지이다. 그러나 실제의 경우 정합되는 화소는 한 개이어야 하므로 정합되는 여러 화소들 중 하나를 선택하여야 한다. 일반적으로는 가장 먼저 정합되는 화소를 선택하게 되는데, 이것이 부정확한 변이값을 갖는 원인이 되기도 한다.

이에 본 논문에서는 적응적 가변 정합창의 크기를 결정할 때 사용한 에지정보를 이용하여 다수의 정합화소 중 한 개를 선택한다. 일반적으로 변이값은 경계부분을 제외하고는 연속적이다. 따라서 정합되는 화소가 다수 개 발생한 경우 그 회소가 에지이면 가장 먼저 정합된 화소를 선택하고, 에지가 아니면 그 전 화소의 변이값과 가장 근접한 변이값을 갖는 화소를 선택한다.

그림 2는 앞서 제안한 방법들을 적용하여 구한 초기 변이영상의 예이다. 일반적인 가변 정합창을 이용한 변이지도보다 경계영역과 균일영역 모두에서 좋은 결과를 보이는 것을 알 수 있다.

그림 2. 제안한 적응적 가변 정합창 방법을 이용한 초기 변이영상

3. 구현결과

스테레오 정합에서 가장 연산량이 많은 부분은 입력된 스테레오 영상에서 화소 혹은 정합창 사이의 연산이다. 각 정합창 사이의 연산은 독립적으로 수행되기 때문에 GPU의 활용 가능성이 더욱 커질 수 있다.

스테레오 정합 알고리즘의 총 연산 복잡도는 $O(N^2WH)$이고 이때 N은 입력 영상의 수, W와 H는 영상의 가로와 세로방향 화소의 수이다. GPU상에서의 구현에서는 WxH 만큼의 스레드를 발생시켜 각각의 스레드에서 하나의 정합창에 대한 시차를 계산한다. 하나의 기준 영상에 대한 시차 영상을 얻기 위해서는 한 번의 커널 함수 호출이 일어나고 이 때 커널의 계산 복잡도는 O가 된다. 따라서 병렬 처리를 이용해 스테레오 영상의 시차를 계산하는 과정의 전체 복잡도는 커널을 N번 호출하는 것으로 생각할 수 있다. 결국 복잡도는 N^2으로 줄일 수 있게 되는 것이다.

제안한 알고리즘은 NVIDIA사의 CUDA를 기반으로 구현하였다. CUDA는 그래픽스 하드웨어를 하나의 독립적인 플랫폼으로 간주하고 프로그래밍 할 수 있는 환경을 제공한다. 사용자는 더 이상 그래픽스 pipeline에 대한 이해가 필요 없게 되었으며, PC환경과 유사한 메모리 모델로 인해 입출력에 대한 제약에서도 벗어나게 된다.

CUDA는 cudaMalloc(), cudaMemcpy(), cudaFree() 등의 함수를 통해 C언어와 유사한 방식으로 메모리 할당 및 복사, 해지를 할 수 있다. 이때의 메모리는 그래픽카드 상에 존재하는 DRAM을 말하며 모니터의 화면출력 지원을 위해 사용하고 남은 공간을 사용하게 된다. CUDA는 메모리 페이징(memory paging) 기능이 지원되지 않기 때문에 항상 메모리를 할당할 때 여유 공간을 고려해야 한다. 효율성을 고려해 전체 프레임 순서에 해당하는 만큼의 메모리를 사용하지 않고 프

로그래밍 초기에 고정된 메모리를 할당하였다. GPU에서 제안한 스테레오 정합 알고리즘은 그림 3과 같은 실행모델로 수행된다.

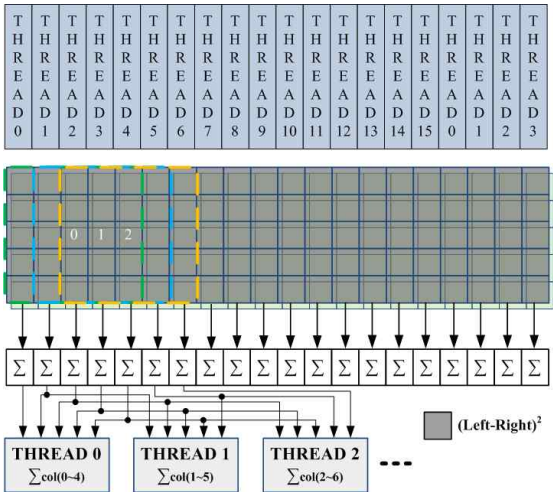

그림 3. 스레드 실행 모델

제안한 알고리즘의 구현환경은 Intel Quad core 2.66, GeForce GTX 296, RAM 3GB이다. 관련 프로그램은 OpenCV, OpenGL, Pointgrey SDK, Visual C++ 등의 프로그래밍 환경에서 구현되었다.

구현결과를 검증하기 위해 사용된 영상은 Middleburry 사이트의 테스트 스테레오 영상들인데[9], 스테레오 비전 분야에서 많이 사용되는 영상들이다. 이 영상들은 평행식으로 배열된 스테레오 카메라로 획득한 영상들이며, 영상의 캘리브레이션(calibration)과 보정(rectification)이 된 영상들이다. 이러한 영상은 대부분 물체를 기준으로 좌측 영상이 얼마만큼 오른쪽으로 이동하느냐를 변이로 선택한다.

제안한 적응적 가변 정합창 방법에 대한 실험결과 제안한 방법이 기존 방법에 비해 초기 변이지도에서 네 영상의 평균 오차율이 2.5%에서 18.2%까지 감소하는 결과를 얻어 제안한 방법의 우수성을 입증하였다. 또한 제안한 교차 일치성 검사를 실험한 결과 제안한 방법이 전체영상 대비 1.7%에서 7.4%, 교차 일치성 검사를 통과한 변이값들 대비 1.0%에서 7.1%의 신뢰도를 향상시켰음을 확인하였다.

표 1에서는 영역의 정합 창 크기를 6가지 종류(3X3, 5X5, 7X7, 9X9, 가변, 적응적 가변)로 하여 CPU와 GPU를 기반으로 구현한 결과를 보이고 있다. 연산시간이 가장 짧게 측정된 Tsukuba의 경우 GPU를 기반으로 구현한 결과 약 125배, 연산시간이 가장 길게 측정된 Cones의 경우 약 128배 정도 속도가 빨라지는 것을 확인할 수 있었다.

4. 결론

본 논문에서는 GPU를 기반으로 영역 기반 스테레오 정합에 있어서 폐색영역에 대한 처리와 낮은 초기 변이정보의 정확도 문제를 개선하여 성능을 향상시키는 알고리즘을 구현하였다. 이 방법은 크게 적응적 가변 정합창을 사용하는 방법과 교차 일치성 검사의 신뢰도를 높이는 방법으로 이루어졌다

또한 GPU를 기반으로 제안한 알고리즘을 구현한 결과 평균 연산속도가 158.86초에서 1.24초로 약 128배 빨라지는 것을 확인할 수 있었다.

표 1. 연산시간 비교 (단위: [sec])

Window size		3×3	5×5	7×7	9×9	Var.	Ad. Var.
Tsukuba	CPU	1.047	2.291	4.232	6.844	31.327	42.201
	GPU	0.022	0.034	0.040	0.048	0.214	0.336
Venus	CPU	1.811	3.924	7.643	11.647	58.015	80.709
	GPU	0.038	0.058	0.071	0.081	0.388	0.629
Teddy	CPU	4.671	11.112	22.225	35.085	165.044	245.216
	GPU	0.098	0.164	0.208	0.245	1.105	1.912
Cones	CPU	8.632	15.082	26.755	38.602	192.760	267.330
	GPU	0.18	0.2232	0.25145	0.270	1.291	2.085

감사의 글

본 논문은 서울시 신기술 연구개발 지원사업(NT080528)의 연구결과입니다.

참고문헌

[1] T. Kanade, M. Okutomi, “A stereo matching algorithm with an adaptive window: theory and experiment,” IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 16, pp. 920-932, 1994.

[2] A. Fusiello, V. Roberto, E. Trucco, “Experiments with a new area-based stereo algorithm,” International Conference on Image Analysis and Processing, 1997.

[3] I. J. Cox, S. L. Hingorani, S. B. Rao, “A maximum likelihood stereo algorithm,” Computer Vision and Image Understanding, 63:3. pp. 542-567, May. 1996.

[4] Jian Sun, Nan-Ning Zheng, Heung-Yeung Shum, “Stereo matching using belief propagation,” IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 27, Issue 25, pp. 787-800, July. 2003.

[5] Peter N. Belhumeur, “A Bayesian approach to binocular stereopsis,” International Journal of Computer Vision, Vol. 19, Issue 3, pp.237-260, Aug. 1996.

[6] I. Gallo, E. Binaghi, M. Raspanti, “Neural disparity computation for dense two-frame stereo correspondence,” Pattern Recognition Letters, Vol. 29, Issue 5, pp. 673-687, April 2008.

[7] P. Seitz, “Using local orientation information as image primitive for robust object recognition,” SPIE Visual Com. and Image Processing IV, Vol. 1199, pp. 1630-1639, 1989.

[8] 이만희, 박인규, “GPGPU를 위한 Shading 언어”, 전자공학회지 제 36권 제 5호, pp. 572-579, 2009년 5월.

[9] http://vision.middlebury.edu/stereo