그래픽 하드웨어를 이용한 고속 스테레오 정합

*이상화 **오준호 **박종일

*서울대학교 **한양대학교

lsh529@snu.ac.kr, jhoh@mr.hanyang.ac.kr, jipark@hanynag.ac.kr

Fast Stereo Matching Using Graphic Hardware

*Lee, Sang Hwa **Oh, Jun Ho **Park, Jong-Il

*Seoul National University **Hanyang University

요약

본 논문에서는 그래픽 하드웨어와 그래픽 프로그램 기술을 이용하여 고속으로 스테레오 영상의 시차를 추정하는 기법을 제안한다. 우선, 컬러 스테레오 영상에 대하여 mean-shift 기법을 이용하여 컬러를 이용한 영역분할을 수행한다. 분할된 컬러 영역 단위로 가중치를 계산함으로써, 화소단위로 가중치를 계산하는 기존의 방식에 비하여 속도를 높일 수 있다. 블록정합함수를 계산하는 과정에서는 슬라이딩 윈도우 방식을 채택하여, 새로 블록안으로 들어오는 화소열과 빠져나가는 화소열의 정합함수값을 가감하여 화소마다 반복적으로 합산되는 정합함수의 계산량을 크게 줄인다. Middlebury 스테레오 영상을 이용하여 실험 및 평가를 수행한 결과, VGA 급 스테레오 영상을 기준으로 10 프레임 이상을 처리하면서도 기존의 적응적인 가중치를 갖는 블록정합 방식의 성능과 유사한 결과를 확인하였다. 이러한 고속화 방법을 통하여, 기존의 적응적인 가중치를 이용한 블록정합 방식에 비하여 훨씬 고속으로 스테레오 정합을 수행할 수 있으며, 실시간 시차추정이 필요한 시스템에 적용하는 것이 가능하다.

1. 서론

스테레오 카메라 시스템은 입체 텔레비전, 3차원 객체 모델 생성, 3차원 영상변환, 그리고 다중 카메라 시스템에서 요구되기 때문에, 많은 관련 연구가 진행되어 왔으며, 특히, 스테레오 정합은 핵심적인 기초기술로서 크게 주목받고 있다 [1], [2]. 스테레오 정합은 서로 다른 위치에서 취득한 두 장 이상의 영상에 대하여 대응관계를 추출하여 시차정보를 추정하는 컴퓨터비전의 주요 연구분야이다. 시차란, 좌우 스테레오 영상에서 서로 대응되는 점이 얼마나 위치차이가 나는지를 화소 단위로 표시하는 것으로서, 카메라의 파라미터를 적용하여 영상의 공간정보로 변환된다. 최근에는 실시간으로 스테레오 카메라 영상으로부터 공간정보를 추출하는 기술과 시스템에 대한 요구가 증가하면서 고속 스테레오 정합에 대한 연구가 활발하게 진행되고 있다.

스테레오 정합은 크게 두 가지 기술적 요소로 구분할 수 있는데, 하나는 블록과 같이 국부적인 화소값의 정합결과를 이용하여 시차를 추정하는 방법이고, 다른 하나는 이웃하고 있는 시차정보간의 연속성과 상관성을 이용하는 전역적인 최적화 기법이다. 전역적인 스테레오 정합 기법은 국부적인 방법과 마르코프 랜덤필드(Markov random field) 모델을 함께 적용하는 경우가 일반적이며, 그래프 컷(graph cut) 또는 신뢰확산(belief propagation) 방법으로 에너지를 최소화하면서 시차를 추정하여 더욱 우수한 결과를 보여주고 있다 [3], [4], [5], [6]. 그러나 전역적인 시차추정 기법들은 확률분포나 에너지 함수의 계산을 위하여 복잡한 연산을 반복적으로 수행하고 병렬연산이 어렵기 때문에, 고속 시차추정에는 적합하지 못하다는 단점이 있다.

본 논문에서는 최근에 컴퓨터비전의 알고리즘을 고속으로 구현하는데 널리 사용되고 있는 그래픽 하드웨어 및 GPU 프로그램 기법을 적용하여 고속으로 시차를 추정하는 기법과 그 구현에 초점을 맞추고 있다 [9], [10]. 실시간으로 3차원 정보를 추출함으로써, 공간정보를 이용한 다양한 응용시스템을 개발하는 것이 가능한데, 현재까지의 스테레오 카메라에서는 이러한 고속 시차추정 기술이 부족하여 상용화도지 못하고 있다.

본 논문에서는 고속 스테레오 정합을 위하여 컬러영상을 세밀하게 분할하여 컬러영역 단위로 정합함수를 고속으로 계산하는 방법을 제안한다. 정합함수는 적응적인 가중치 기법을 단순화하고 [7], [8], 컬러 영역 단위로 정합함수를 계산하도록 변형하여 병렬 처리 및 고속 연산이 가능하도록 설계한다. 그리고 제안된 기법은 그래픽 하드웨어를 이용하는 GPU 프로그램으로 구현되어 고속으로 동작하도록 한다.

본 논문의 구성은 다음과 같다. 제 2절에서는 GPU 환경에서 구현되도록 고속 스테레오 정합 알고리즘을 제안하고, 제 3절에서는 Middlebury 스테레오 영상을 이용한 실험결과를 제시한다. 그리고 제 4절에서는 본 논문에 대한 결론과 향후 개선할 점을 언급한다.

2. 고속 스테레오 정합 알고리즘

본 논문은 적응적인 가중치를 기반으로 정합함수를 계산하여 정합을 수행하는 기법을 기반으로 하고 있다 [7], [8]. 이러한 기법에서는 블록안의 화소에 대한 정합함수를 더함에 있어서 중심화소와 블록내

의 화소간의 컬러값과 거리에 따라서 적응적으로 가중치를 둔다. 이러한 과정으로 인하여 다양한 형태의 영역 및 컬러분포에 대해서 더욱 정확하게 정합을 수행할 수 있다. 그러나 이러한 방법은 가중치를 계산하기 위하여 많은 연산이 필요할 뿐만 아니라, 이 연산과정을 병렬화하는 것이 어렵기 때문에, 고속 스테레오 시스템에서는 사용할 수 없다. 본 논문은 이러한 적응적 가중치를 계산하는 과정을 근사화함으로써, 성능저하를 최소화하면서 수행속도를 고속화하는 기법을 제안하고자 한다.

2.1. 컬러영역 분할

우선 스테레오 컬러영상에 대하여 유사한 컬러값을 갖는 화소들을 하나의 영역으로 묶는 컬러영역분할을 수행한다. 컬러 영역은 컬러값의 차이를 이용한 적응적인 가중치를 계산함에 있어서, 화소단위가 아니라 컬러영역단위로 가중치를 계산하기 위한 것이다. 유사한 컬러값을 갖는 영역의 화소들은 중심화소의 컬러값과 유사한 정합함수값을 갖기 때문에, 하나의 값으로 가중치를 계산하더라도 그 차이가 거의 없음을 이용하는 것이다.

영상의 잡음에 의한 영역분할 오류를 줄이기 위하여 본 논문에서는 우선 컬러영상에 대하여 3x3 블록으로 가우시안 필터링을 수행하고, 이에 대하여 컬러벡터공간에서 mean-shift 기법을 이용하여 컬러영역분할을 수행한다 [11]. 그림 1은 mean-shift 기법으로 수행한 컬러영역을 세밀하게 분할한 예를 보여주고 있다. 그림 1에서 좌측영상들은 원영상이고 우측영상들은 해당 원영상을 컬러분할한 결과를 보여주고 있다. 그림 1에서 분할된 각각의 컬러영역은 정합함수를 계산하는 과정에서 적응적인 가중치를 결정하는 단위가 되는데, 화소단위의 정합함수를 더하는 과정을 단순화시켜서 전체 수행속도를 향상시키는데 사용된다.

(a) Teddy 영상과 분할된 영역.

(b) Cone 영상과 분할된 영역.

그림 1. Mean-shift 기법으로 세밀하게 분할된 컬러영역. 좌측은 컬러영역, 우측은 컬러분할된 영상의 경계부분.

2.2. 고속 정합 함수의 계산

최근에 알려진 국부적인 정합함수는 [7]에서 제안한 방법이 매우 우수한 것으로 알려져 있다. [7]에서는 블록내의 정합 함수를 계산하는 과정에서 중심 화소와 블록내의 화소간의 기하학적 거리 및 컬러값의 차이를 가중치로 모델링하여 적응적인 윈도우를 갖는 블록정합 형태가 되도록 한 것이다. 이러한 방법은 우수한 정합 성능을 보여주지만, 정합함수를 계산하는 과정에서 해당 가중치를 매번 계산하여야 하기 때문에, 연산량이 많아지고 병렬처리를 통한 고속화가 불가능하다는 문제가 있다고 앞에서도 언급한 바 있다. 본 논문에서는 [7]에서 제안된 적응적인 정합함수를 단순화하고 GPU 프로그램으로 고속화하는데 초점을 맞추고 있다.

본 논문에서는 2.1절에서 얻어진 세밀한 컬러 영역을 정합함수를 합산하는 적응적인 윈도우로 설정한다. 즉, 다음과 같이 표현할 수 있다.

$$w(p_i, p_j) = \exp\left(-\frac{\alpha(p_i, p_j)}{\gamma_r} - \frac{\beta(m_i, m_j)}{\gamma_c} \right) \qquad (1)$$

식(1)에서 중심 화소 p_i 의 컬러값과 블록내의 화소 p_j 의 화소값은 컬러 분할된 영역의 평균값 (m_i , m_j)으로 대체되며, 컬러값간의 차이에 의한 거리를 $\beta(m_i, m_j)$ 로 계산한다. 이 경우에, 화소마다 정합함수의 가중치를 계산할 필요가 없다. 다른 함수 $\alpha(p_i, p_j)$ 는 두 화소간의 거리를 나타내는데, 분할된 영역의 중심점간의 거리로 구해진다. 이 경우에는 두 위치가 주어지면, 자동적으로 그 거리가 사전에 정해져 있기 때문에 룩업 테이블(lookup table)의 형태로 고속화하는 것이 가능하다. 이와 같이 컬러영역을 단위로 적응적인 가중치를 계산함으로써, 화소단위로 가중치를 계산하는 중복적인 과정을 생략할 수 있으며, 다음 절에서 설명하는 슬라이딩 윈도우 개념으로 정합함수를 합산하는 과정을 더욱 고속화할 수 있다.

2.3. 고속 정합 함수의 계산

슬라이딩 윈도우 기법은 블록정합과 같은 정합함수를 합산하는 과정을 고속화 하는 방법이다. 블록정합시 한 화소씩 밀려가는 블록의 화소부분에 대하여 빠져나가는 화소들과 새로 블록안으로 들어오는 화소부분들에 대해서만 정합함수를 선택적으로 계산하여 전체 블록정합함수를 구함으로써, 정합함수의 합산 과정을 고속화하는 것이다. 기존의 [7], [8]과 같이 화소마다 적응적인 가중치를 계산하는 방법은 블록이 이동할 때마다 각 화소에 대한 가중치를 다시 계산하여야 하기 때문에, 이러한 슬라이딩 윈도우 방법을 사용할 수 없었다.

본 논문에서는 식 (1)을 이용하여 컬러영역을 기반으로 가중치를 계산하기 때문에, 다음과 같이 슬라이딩 윈도우 기법을 적용할 수 있다.

1) 블록정합함수를 계산하는 현재 중심 화소와 다음 중심 화소가 동일

한 컬러영역 안에 있는 경우에는 슬라이딩 윈도우를 이용하여 정합함수의 합산이 가능하다. 그림 2에서 보는 바와 같이 블록이 이동하더라도 중심화소의 컬러영역이 변하지 않는 경우에는 블록안의 화소들에 대한 가중치가 변하지 않게 되어 이전에 계산된 값을 그대로 활용하는 것이 가능하다. 새로 추가되는 화소열에 대해서만 정합함수를 계산하여 합하고, 빠져 나가는 화소열의 정합함수 부분을 제외하면 된다. 이때, 정합함수의 값은 세로 방향의 화소열 단위로 계산하여 저장하여 가감 연산을 빠르게 수행한다.

2) 현재의 중심화소와 다음 위치의 중심화소의 컬러영역이 달라지는 경우에는 새로운 평균컬러와 위치에 대한 거리 가중치를 계산하여야 한다. 이때는 슬라이딩 윈도우 기법을 적용할 수 없지만, 컬러영역 기반의 가중치를 계산하기 때문에, 비교적 빠른 연산을 수행할 수 있다.

따라서 슬라이딩 윈도우 기법은 동일한 컬러영역으로 중심화소가 이동하는 경우에 적용되며, 유산한 컬러분포가 많은 영상에서는 속도 향상이 더욱 두드러진다.

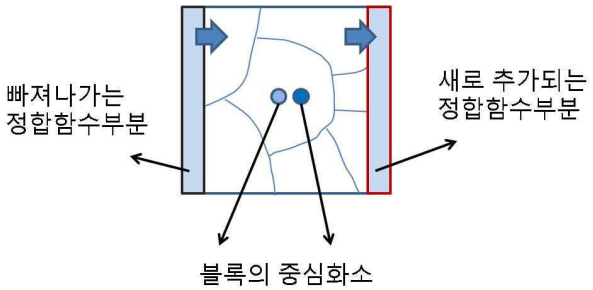


그림 2. 슬라이딩 윈도우 기법을 이용한 정합함수의 합산과정. 블록이 이동하면서 새로 추가되는 우측의 화소열의 정합함수는 합산되고, 좌측의 빠져나가는 화소열의 정합함수값은 제외한다.

3. 실험결과 및 응용

본 논문에서는 Middlebury 스테레오 사이트에서 제공하는 영상을 이용하여 시차추정의 정확도와 속도를 평가하였다. 실험은 펜티엄 쿼드코어 CPU와 지포스 285GTX 그래픽 하드웨어를 이용하였다. 처리 속도는 영상의 크기와 시차범위에 따라서 초당 처리 가능한 프레임 비율이 달라지기 때문에, 다음과 같이 de/s(disparity evaluation/second) 측정방식을 이용한다.

de/s = 영상의 화소수*시차범위 / 처리시간.

그림 3은 실험에 사용된 그림 1의 영상(크기: 450x385, 시차범위: 60화소)에 대하여 [7]에서 제안한 방법과 제안한 방법에 의한 시차지도를 비교한 것이다. [7]에 의한 시차지도는 제안 기법에 비하여 정확도 측면에서 다소 우수하지만, 처리속도 측면에서는 4.22 Mde/s 정도 소요되는 것으로 측정되었기 때문에, 고속 스테레오 시스템에서는 사용할 수 없다. 반면에 본 논문에서 구현한 고속 시차추정 알고리즘은 107.7 Mde/s 정도의 속도를 보여주었다. 이러한 속도는 VGA (640x480) 급, 30 화소 정도의 시차범위를 갖는 영상에 대하여 약 12 프레임 처리 속도로서 해당한다. 완전한 실시간 처리에는 다소 못 미친다고 볼 수 있지만, 현재 하드웨어로 구현된 범블비 카메라의 시차추정 속도와 비슷하면서도 정확도는 향상되었다는 점에서, 시차추정을 위한 속도 개선은 크게 이루어졌다고 할 수 있다.

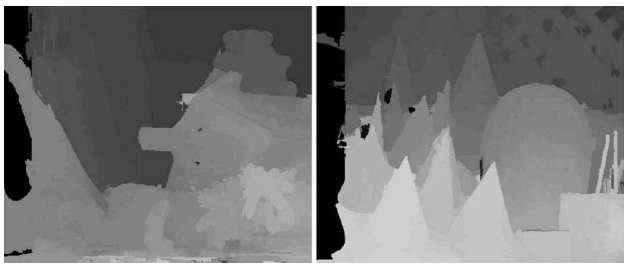
일반적인 블록정합과의 비교를 위하여 본 논문에서는 슬라이딩 윈도우 개념을 적용하여 GPU 프로그램을 통하여 스테레오 정합을 구현하였다. 블록정합의 경우는 1252.4 Mde/s 의 결과를 얻었으며, VGA급 영상을 기준으로는 135 프레임 정도로 처리할 수 있는 수준이지만, 정확도는 다소 떨어지는 것을 확인하였다. 표 1은 3가지 방법에 대하여 걸린 시간, 시차지도의 정확도 등을 정리한 것이다.

본 논문에서의 개선사항으로는 우선 프로그램 최적화를 통하여 속도를 좀 더 개선시킬 여지가 있다. 특히 컬러영역 분할 과정에서의 속도 개선이 필요하며, 정합함수를 계산하기 위한 블록의 크기를 줄여서 연산량을 줄이는 것이 요구된다. 단, 블록의 크기가 줄어들 경우 시차추정 성능이 크게 저하되기 때문에, 시차정보간의 상관성을 활용하는 전역적인 최적화 기법을 추가로 도입할 필요가 있다. 향후 연구에서는 이러한 속도 개선과 후처리 과정으로서의 전역 최적화 기법을 개발하는데 초점을 맞추고자 한다.

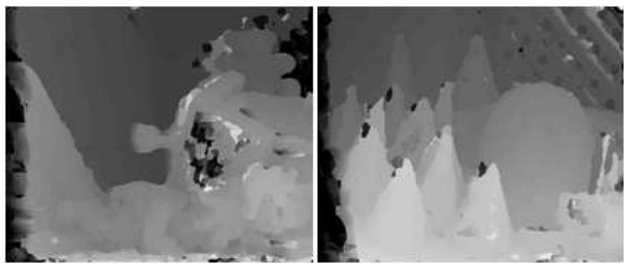
본 논문에서 구현한 고속 스테레오 정합은 실시간 시차추정이 필요한 시스템에서 사용이 가능하다. 예를 들면, 실시간으로 시차를 추정하여 물체의 거리 및 크기를 측정할 수 있으며, 고속 이동 물체의 위치 탐지 및 추적 시스템에 적용하는 것이 가능하다. 고속으로 날아가는 물체에 대한 탐지를 위하여 향후, 300Mde/s (30 프레임/초) 이상의 속도로 처리할 수 있는 알고리즘과 구현을 할 필요가 있다.

표 1. 시차지도 성능 비교.

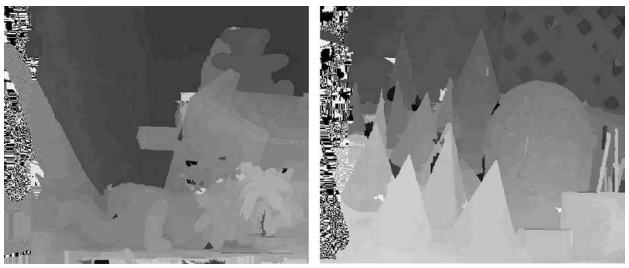
방법	시간 (ms)	teddy (%)	cone (%)
[7]기법	2400	12.7	6.5
블록정합	8.3	19.6	12.0
제안기법	94	13.3	5.6



(a) [7]에 의한 시차지도 (오차: 12.7%, 6.5%)



(b) 블록정합에 의한 시차지도 (오차: 19.6%, 12.0%)



(c) 제안기법에 의한 고속 시차지도 (오차: 13.3%, 5.6%)

그림 3. [7], 블록정합에 의한 시차지도와 제안한 기법에 의한 시차지도의 비교. Occlusion 영역에 대한 오차를 제외하면, 시차지도의 성능이 크게 저하되지 않는다.

4. 결론

본 논문에서는 그래픽 하드웨어를 이용한 고속 스테레오 정합 기법을 제안하고 구현하였다. 컬러영역 분할과 적응적인 윈도우, 슬라이딩 윈도우의 개념을 이용한 정합함수의 합산 기법을 이용하여 GPU 환경에서 고속으로 시차를 추정하는 알고리즘을 구현하였다. 본 논문에서 구현한 알고리즘은 기존의 알고리즘에 비하여 성능 측면에서 크게 떨어지지 않으면서 더욱 빠르게 동작됨을 확인하였다. 초고속 실시간 스테레오 환경에서 사용하기에는 다소 속도 측면에서 미흡하기 때문에, 컬러영역 분할 기법을 고속화하고 블록의 크기를 줄여서 속도를 개선할 필요가 있으며, 다소 저하되는 성능을 보정하기 위하여 전역 최적화 기법을 도입할 필요가 있다. 본 논문에서 구현한 실시간 스테레오 정합은 현재의 PC 기반의 스테레오 카메라에서 적용이 가능하다. 추가적인 연구개발과 최적화를 수행한다면, 고속 이동체의 위치 탐지 및 추적 시스템과 같은 30프레임 이상의 초고속 시스템에서도 적용할 수 있을 것으로 기대된다.

감사의 글

본 연구는 방위사업청과 국방과학연구소가 지원하는 생존성기술 특화연구센터사업의 일환으로 수행되었습니다.

참고문헌

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

[2] D. Scharstein and R. Szeleiski, “A taxonomy and evaluation of dense two frame stereo correspondence algorithm,” *International Journalof Computer Vision*, vol. 47(1/2/3), pp.7-42, June 2002.

[3] Y. Boykov, O. Veksler, and R.Zabih, ``Fast approximate energy minimization with graph cuts,'' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222-1239, Nov. 2001.

[4] S. H. Lee, Y. Kanatsugu, and J.-I. Park, ``Stochastic diffusion for stereo matching and line fields estimation,'' *International Journal of Computer Vision*, vol. 47 (1/2/3), pp. 195-218, June 2002.

[5] D. Scharstein and R. Szeliski, ``Stereo matching with non-linear diffusion,'' *International Journal of Computer Vision*, vol. 28, no. 2, pp. 155-174, Feb. 1998.

[6] J. Sun, N. N. Zheng, and H. Y. Shum, ``Stereo matching using belief propagation,'' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 7, pp.399-406, July 2003.

[7] K. J. Yoon and I. S. Kweon, “Adaptive support weight approach for correspondence search,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 650-656, April 2004.

[8] F. Tombari, S. Mattoccia, L. D. Stefano, “Segmentation-based adaptive support for accurate stereo correspondence,” *IEEE Pacific-Rim Symposium on Image and Video Technology*, LNCS 4872, pp.427-438,(2007).

[9] M. Gong and T. Yang, “Near real-time stereo matching using programmable graphics hardware,” *Proc. of CVPR*, 2005.

[10] M. Gong and Y. Yang, “Real-time stereo matching using orthogonal relability-based dynamic programming,” *IEEE Trans. on Image Processing*, vol. 16, no. 3, pp. 879-884, March 2007.

[11] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *IEEE Trans. PAMI*, vol. 2, no. 4, May 2002.