분산메모리 머신에서의 병렬 윤곽선 랭킹

정용화 °, 박진원
성능평가연구팀, 한국전자통신연구원

Parallel Contour Ranking on a Distributed-Memory Machine

Yongwha Chung and Jin-Won Park
Performance Evaluation Team, ETRI

요 약

본 논문에서는 분산메모리 머신에서 병렬 이미지 윤곽선 랭킹 문제를 해결하는 새로운 알고리즘을 제안한다. 윤곽선 랭킹 문제는 주어진 이미지의 에지 윤곽선으로부터 에지 윤곽선의 선형적 표현 방식을 생성시키는 것으로, 에지점간의 순차적인 데이터 종속 관계를 갖는 이 문제를 분산메모리 머신에서 수행하려면 입력 이미지에 의한 데이터의 불균형 분포와 불규칙적인 프로세서간 데이터 종속 문제를 해결해야 한다. 본 논문에서는 이 두 가지 문제를 동시에 해결할 수 있는 병렬 알고리즘을 제안하고, 제안된 알고리즘을 IBM SP2에 구현하였으며, 그 결과 윤곽선 랭킹 문제가 효과적으로 해결되었음을 확인하였다.

1. 서론

본 논문에서는 분산메모리 머신에서의 병렬 이미지 윤곽선(contour) 랭킹 알고리즘을 제안한다. 일반적으로 컴퓨터 비전(computer vision) 문제는 하위/중간/상위 레벨의 세 단계 태스크로 구분되는데, 대부분의 하위 레벨 비전 태스크의 에지 오퍼레이터들은 2차원 이미지 면에서의 에지 맵(edge map)으로 표현되는 에지 윤곽선을 생성한다. 그러나 이러한 에지 맵으로 표현되는 윤곽선이 중간이나 상위 레벨 비전에서 보다 효과적으로 처리되기 위해서는 보다 간결한 선형적 표현 방식(linear representation)으로의 변형이 바람직하다 [1,2]. 윤곽선 랭킹 문제는 이미지의 에지 윤곽선으로부터 에지 윤곽선의 선형적 표현 방식을 생성하기 위하여 설정된다.

지금까지 발표된 대부분의 하위 레벨 비전 태스크에서의 병렬화 방법[3-6]은 주어진 입력 이미지에 의한 프로세서간 에지점(edge point)의 불균형 분포만을 고려하였다. 그러나, 에지점간의 순차적인 데이터 종속(sequential data dependency) 관계를 갖는 윤곽선 랭킹 문제를 분산메모리 머신에서 수행하려면 프로세서간 에지점의 불균형 분포뿐만 아니라 불규칙한 프로세서간 데이터 종속을 해결해야 한다. 본 논문에서는 이 두 가지 요소를 동시에 고려한 데이터 재매핑(re-mapping) 알고리즘을 제안하고, 제안된 알고리즘을 IBM SP2에 구현한 결과 윤곽선 랭킹 문제가 효과적으로 수행됨을 확인하였다.

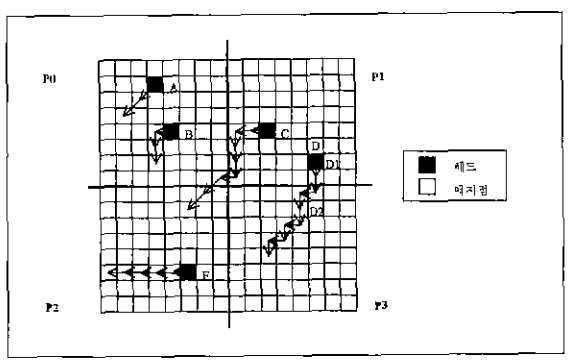
본 논문의 2장에서는 윤곽선 랭킹 문제에 대하여 서술하고, 제안된 병렬 알고리즘과 그 구현을 3장과 4장에서 각각 설명한다. 마지막으로 5장에서는 결론으로 끝맺는다.

2. 윤곽선 랭킹 문제

P개의 프로세서로 구성된 분산메모리 머신에서 하위 레벨 비전 태스크를 수행하기 위하여 $n \times n$ 이미지가 P개의 서브 이미지(크기가 $n/\sqrt{P} \times n/\sqrt{P}$)로 분할

되었다고 가정하자. 각각의 프로세서는 하나의 서브 이미지를 처리하고, 각 서브 이미지는 최대 8개의 인접 서브 이미지를 갖는다. 각 프로세서에서 에지 윤곽선을 형성하는 에지점이 이미 추출되어 있다고 가정하면, 윤곽선의 선형적 표현 방식은 프로세서내의 에지점을 연결함으로써 생성될 수 있다. 이때 윤곽선의 길이나 선행 에지점을 포함하는 프로세서에 관한 정보 등이 계산될 수 있으며, 이 정보는 데이터 재매핑시 활용된다.

즉, 각 에지점 e에 대하여, pred(e)는 e의 8개 선행점(predecessor) 중 그 에지 윤곽선에 속하는 하나의 에지점을 가리키는 함수로서, e의 선행점 집합은 e와 pred-관계의 reflective 및 transitive closure 한 모든 에지점을 포함하는 집합이다. 특별히, pred(e) = nil 이 되는 에지점 e를 헤드라 한다. 또한 하나의 에지점은 기껏해야 하나의 다른 에지점의 선행점이 되며, 모든 에지점의 선행점 집합은 단지 하나의 헤드를 갖는다고 가정한다. 따라서 윤곽선 랭킹에서는, 모든 에지점 e에 대하여 e의 선행점 집합에 있는 헤드 및 선행점 집합의 크기를 결정한다. 특히, 이 선행점 집합의 크기를 e의 랭크라 하며, rank(e)=rank(pred(e))+1에 의하여 e와 선행점 사이에는 **순차적인 데이터 종속** 관계가 존재한다. 일단 모든 에지점의 랭크를 알면 데이터 이동에 의하여 선형적 표현 방식을 얻을 수 있으며, 에지 윤곽선이 중간이나 상위 레벨 비전에서 효과적으로 처리될 수 있다.



(그림 1) 4개 프로세서를 이용한 윤곽선 랭킹

그러나 에지점간의 이러한 순차적인 데이터 종속 관계 때문에, 단순히 프로세서당 할당된 에지점의 수를 같게 하는 기존의 부하 분산 방법만으로는 원하는 처리속도의 개선을 기대할 수 없다. 즉, (그림 1)에 나타낸 예에서 보는 바와 같이 각 프로세서에 할당된 에지점의 수는 동일하다. 그러나 하나의 윤곽선을 구성하지만 그 에지점들이 두 개의 프로세서에 걸쳐있는 광역(global) 윤곽선 D의 경우, 세그멘트 D2의 처리는 선행 세그멘트 D1의 처리가 인접 프로세서 P1에서 완료된 후에만 가능하며, 이때까지 프로세서 P3는 idle 상태에 있다.

3. 병렬 알고리즘

본 논문에서 제안하는 병렬 이미지 윤곽선 랭킹 알고리즘은 프로세서간 에지점의 불균형 분포뿐만 아니라 불규칙한 프로세서간 데이터 종속을 동시에 해결하기 위하여 데이터 재매핑을 수행한다. 따라서 데이터 재매핑 후에는 하나의 윤곽선을 구성하는 에지점들의 액세스가 하나의 프로세서 내로 지역화(localize) 되어, 각 프로세서는 독립적으로 윤곽선 랭킹을 수행할 수 있다. 제안되는 알고리즘은 다음과 같이 크게 세 단계로 이루어진다.

병렬 윤곽선 랭킹 알고리즘
단계1 세그멘트 라벨링
단계2. 데이터 이동
단계3 지역화된 윤곽선 랭킹

먼저, 단계1에서 각 윤곽선의 세그멘트에 라벨 $l(0 \le l<P)$을 할당한다. 세그멘트 라벨링의 목적은 먼저 각 세그멘트의 웨이트(그 세그멘트를 구성하는 에지점의 수)를 계산하고, 계산된 웨이트를 기준으로 각 세그멘트의 수신 프로세서를 결정하는 것이다. 이때 프로세서간 부하 분산을 위해서 윤곽선들을 P개의 그룹으로 분할하는데, 각 그룹에서 처리되어야 될 에지점의 수가 유사하게끔 분할한다. 각 프로세서에서 수행된 라벨링의 결과는 P개의 어레이에 저장되는데. 어레이 i에는 프로세서 i로 전송되어야할 윤곽선 세그멘트가 저장된다. 이때 하나의 윤곽선을 구성하지만 그 에지점들이 여러 프로세서에 걸쳐있는 광역 세그멘트들에게 동일한 수신 프로세서를 지정한다. 따라서, 데이터 재매핑 후에 그 세그멘트의 액세스가 하나의 프로세서 내로 지역화 되어 프로세서간 데이터 종속 문제가 해결될 수 있도록 한다.

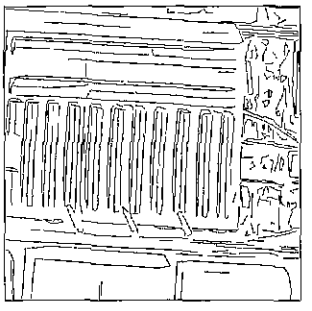
예를 들어, (그림 1)에서 광역 윤곽선 C 및 D를 구성하는 세그멘트들의 웨이트는 그 광역 윤곽선의 마지막 세그멘트를 저장하고 있는 프로세서 P2 및 P3로 각각 전송된다 이때 각 프로세서에서 계산된 웨이트의 합은 7, 0, 15, 10이 되며, 이 값은 다른 모든 프로세서로 전송된다. 그러면 각 프로세서는 웨이트의 총합 32가 4개의 프로세서에 균등히 배분되도록 해당 세그멘트를 라벨링한다. 즉, 윤곽선 A, B의 세그멘트는 프로세서 P0로, 윤곽선 C의 세그멘트는 프로세서 P1으로, 윤곽선 E의 세그멘트는 프로세서 P2로, 윤곽선 D의 세그멘트는 프로세서 P3로 전송되도록 라벨링된다.

단계2에서는 윤곽선의 에지점 액세스를 지역화하기 위해 All-to-All Personalized 통신을 수행한다. 본 논문에서는 데이터 이동에 따른 오버헤드의 상한 값을 측정키 위해 가장 간단한 구현 방법을 채택한다. 즉, 라운드 j $(0 \le j<P)$에서 각 프로세서 $i(0 \le i<P)$는 프로세서 $(i+j)$ mod P로 데이터를 전송한다. 일단

하나의 윤곽선을 구성하는 세그멘트들이 하나의 프로세서로 모아지면, 더 이상의 프로세서간 데이터 통신 없이 윤곽선 랭킹을 수행할 수 있다(단계3).

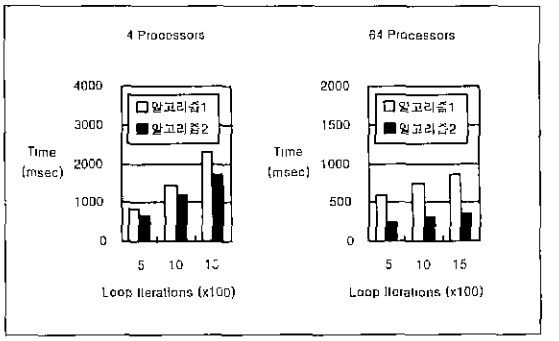
4. 구현

제안된 알고리즘을 MPI를 이용하여 IBM SP2에 구현하였다. 본 논문에서는 다양한 윤곽선 처리 알고리즘 중 특정 알고리즘을 목표로 하는 대신에, 더미(dummy) 계산을 수행하는 루프를 이용하여 다양한 윤곽선 처리 알고리즘에 소요되는 계산시간 t를 시뮬레이트 한다. 여기서 t는 프로세서가 하나의 에지점 처리를 위해 필요한 계산시간을 의미한다. 따라서 더미 계산을 수행하는 루프의 반복(iteration) 횟수를 변경함으로써 다양한 t값을 생성해 낼 수 있다. 또한 비교를 위하여 데이터 재매핑 없이 윤곽선 랭킹을 수행하는 기존의 알고리즘을 구현하여 이를 **알고리즘 1**이라 하고, 본 논문에서 제안된 방법으로 데이터 재배핑 후 윤곽선 랭킹을 수행하는 알고리즘을 **알고리즘 2**라 한다.



(그림 2) 빌딩 이미지에서 추출된 에지점

(그림 2)에 512×512 크기를 갖는 빌딩 이미지에서 추출된 에지점을 나타내었고, 이를 이용한 알고리즘 1 및 2의 성능을 (그림 3)에 비교하였다. 알고리즘 2에서 불균형 부하를 분산시킬 때 불규칙한 프로세서간 데이터 종속을 동시에 고려함으로써 얻은 이득이 그에 따른 오버헤드 보다 큰 것을 알 수 있다. 예를 들어 4개의 프로세서를 사용한 경우, 데이터 재매핑을 수행하기 전과 후의 각 프로세서에 할당된 에지점 수는 9288/10245/9002/9929와 9638/9657/9559/9610으로 그 분포의 차이가 크지 않다. 그러나, 불규칙한 프로세서간 데이터 종속을 해결한 알고리즘 2의 성능이 더 좋은데, 이는 전체 윤곽선의 8%에 해당하는 광역 윤곽선의 에지점이 데이터 재매핑 후에 지역화 되었기 때문이다. 또한, 프로세서 수가 증가할수록 제안된 알고리즘의 효과가 현저히 나타남을 알 수 있는데, 이는 프로세서 수가 증가하면 주어진 입력 이미지내의 전체 윤곽선 중 광역 윤곽선의 비율이 증가하여 프로세서간 데이터 종속이 상대적으로 성능에 큰 영향을 주기 때문이다. 본 논문에서는 지면 제약상 하나의 빌딩 이미지에 대한 결과만을 나타내었지만, 여러 가지 다른 이미지에 대해서도 유사한 결과를 얻을 수 있었다.



(그림 3) 알고리즘 1 및 2의 수행시간

5. 결론

본 논문에서는 분산메모리 머신에서의 병렬 이미지 윤곽선 랭킹 알고리즘을 제안하였다. 입력 데이터의 불균형 분포와 불규칙한 프로세서간 데이터 종속 문제를 동시에 고려한 데이터 재매핑 알고리즘을 제안하였고, 제안된 알고리즘을 IBM SP2에 구현한 결과 기존의 알고리즘에 비하여 처리속도가 2.4배까지 개선되었음을 확인하였다.

참고문헌

[1] *Dunham J., "Optimum Uniform Piecewise Linear* Approximation of Planar Curves", IEEE Tr. Pattern Analysis and Machine Intelligence, Vol. 8, No 1, pp. 67-75, 1986

[2] Nevatia R. and Babu K., "Linear Feature Extraction and Description", Computer Graphics and Image Processing, Vol. 13, No 1, pp. 257-269, 1980.

[3] *Chen L., Davis L., and Kruskal C., "Efficient* Parallel Processing of Image Contours", IEEE Tr. Pattern Analysis and Machine Intelligence, Vol. 15, No. 1, pp. 69-81, 1993.

[4] Gerogiannis D. and Orphanoudakis S., "Load Balancing Requirements in Parallel Implementations of Image Feature Extraction Tasks", IEEE Tr. Parallel and Distributed Systems. Vol 4, No 9, pp. 994-1013, 1993

[5] Bader D and JaJa J., "Practical Parallel Algorithms for Dynamic Data Redistribution, Median Finding, and Selection", Proc. of IPPS'96, pp. 292-301, 1996.

[6] Choudhary A., Narahari B, and Krishnamurti R., "An Efficient Heuristic Scheme for Dynamic Remapping of Parallel Computations", Parallel Computing, Vol 19, No. 9, pp. 621-632, 1993.