Multi-processor 를 위한 CGH 알고리즘에 대한 성능 분석

이정윤, *이성원
광운대학교 컴퓨터공학과
jeongyoun.yi@kw.ac.kr, *swlee@kw.ac.kr

A performance analysis of CGH algorithm for Multi-processor

Jeong Youn Yi *Seong Won Lee
Kwangwoon University

요 약

홀로그램은 레이저 빔의 간섭을 이용하여 입체영상을 광학적으로 기록하는 방법으로 동영상의 기록 및 재생에는 여러 제약 조건이 따른다. 때문에 광학적 방법으로는 홀로그램 TV 시스템을 구현하기 힘들다. 이런 단점을 극복하기 위해 컴퓨터를 이용해 홀로그램을 구현할 수 있다. 즉, 광학적인 신호들을 근사화한 후 컴퓨터에서 수학적인 연산을 거쳐 간섭 무늬를 계산하는 Computer generated hologram(CGH) 방법을 사용하여 패턴을 생성한다. 하지만 CGH 기법의 경우 상당한 연산을 필요로 하기 때문에 연산을 최적화하여 홀로그램 생성에 걸리는 시간과 비용을 최소화하려는 연구들이 많이 진행되고 있다. 본 논문에서는 이전에 연구된 CGH 고속 연산을 위한 알고리즘을 정리하며 연산 식의 최적화를 통해 연산을 횟수를 줄이는 방법과 look-up table 을 이용한 방식의 연산량과 하드웨어 비용을 계산하여 multi-processor 에 적용 시 어떤 알고리즘이 유리할지 제안한다.

1. 서론

홀로그래피는 object 의 3 차원 정보를 필름에 저장하고, 필름을 사용하여 3 차원 물체를 복원하는 방법이다. 홀로그래피는 3D 디스플레이 기술의 최종 목표라고도 할 수 있는 가장 이상적인 방식이다. Stereoscopic 이나 이전의 3D 디스플레이들처럼 입체영상을 보기 위한 특별한 장비가 필요하지 않기 때문에 3 차원 객체를 보다 정확한 깊이 감을 제공하는 영상을 관찰할 수 있다. 홀로그램을 구현하는 방법에는 여러 기법이 있는데 레이저 빔을 이용하여 홀로그램을 광학적으로 기록할 시에는 매우 안정된 광학계를 요구한다는 제약조건을 가지고 있다. 즉 물체의 작은 움직임에도 세기와 위상과 같은 정보들을 갖고 있는 간섭무늬가 없어질 수도 있어 동영상 기록에 적합하지 않다.[2]

이런 문제를 해결하기 위해 현재 많이 연구되고 있는 것이 CGH 기법이다. CGH 는 광학 신호들을 근사화한 후 PC 에서 수학적인 연산으로 디지털 홀로그램을 생성하는 기법으로, Object beam 과 reference beam 에 의해 만들어지는 간섭무늬를 CGH 연산을 통하여 계산하는 방식이다. CGH 를 계산하기 전에 먼저 depth 카메라 등 장비를 통해 실제 3 차원 object 의 3 차원 좌표(x, y, z)와 pixel 의 luminance, chrominance 를 획득한 한다, 그 후 CGH 수식의 연산을 통하여 간섭 무늬를 계산하고, 이를 디스플레이 하게 된다. 하지만 CGH 의 수식의 경우 한 프레임의 홀로그램을 생성하는 데에 방대한 양의 연산을 필요로 하기 때문에 이 연산을 줄이기 위한 연구들이 많이 진행되고 있다.

본 논문에서는 CGH 를 통해 간섭무늬를 계산하는 방법의 multi-processor 환경에서의 적합성에 대해 다룬다. 2 절 본론에서는 먼저 연산의 최적화를 통한 고속 CGH 연산 방식과 look-up table 을 이용한 논문들을 정리하고, multi-processor 나 multi-GPU 를 사용하여 구현 시 적합한 방법을 제안한다. 마지막으로 3 절에서 본 논문의 결론을 맺는다.

2. 본론

2.1 연산 최적화

[1],[3]에서 제안하는 방식은 연산의 횟수를 줄여 빠르게 CGH 를 구하는 것이다. 아래의 CGH 생성 식은 위상 성분만을 이용한 연산 기법이다.

$$I_\alpha = \sum_j^N A_j \cos\left(kR_{aj} + \phi_\alpha + \phi_j\right) \quad (1)$$

$$R_{aj} \cong z_j + \frac{p^2}{2z_j}\left(x_{aj}^2 + y_{aj}^2\right)$$

식(1)는 CGH 생성 식을 빠르게 연산하기 위해 근사식으로 변환한 것이다. α와 j 는 각각 홀로그램과 object 의 인덱스이고, p 는 홀로그램의 화소 크기이다. x_α, y_α는 홀로그램의 좌표, x_j, y_j 는 object 의 좌표이다. z_j 는 홀로그램과 object 사이의 거리이다. ϕ_α, ϕ_j 는 reference beam 과 object beam 의 위상이다. k 는 reference beam 의 파수로써 $2\pi/\lambda$으로 나타낸다. N 은 object 의 point 수이고, A_j 는 object 의 빛의 세기이다.

$$\theta_H\left(x_{aj}, y_{aj}, z_j\right) = kR_{aj} = 2\pi(\theta_Z + \theta_{XY}) \quad (2)$$

θ_H 는 object 의 (x_j, y_j)로 부터 홀로그램 (x_α, y_α)로 전파된 위상으로, 식(2)와 같이 계산할 수 있다. 같은 x 축

위에서의 위상 $\theta_{XY}(x_{aj}+d, y_{aj}, z_j)$ 은 식(3)과 같이 표현할 수 있다.

$$\theta_{XY}(x_{aj}+d, y_{aj}, z_j) = \theta_{XY}(x_{aj}, y_{aj}, z_j) + \Gamma_d \qquad (3)$$
$$(\Gamma_d(x_{aj}, z_j) = \frac{p^2}{2\lambda z_j}\left(2dx_{aj} + d^2\right))$$

Γ_d 를 다음과 같이 일반화 시킬 수 있다. $d=n$ 이라고 하면 식(4)가 만족된다.

$$\Gamma_n = N\Gamma_1 + \frac{n(n-1)}{2}\Delta_x \qquad (4)$$

식(4)에서 Γ_1, Δ_x 값을 미리 계산해 놓으면 x 축 첫 번째 좌표 값들만을 이용하여 CGH 연산을 수행할 수 있다. 이전 연산 값에 상관없이 홀로그램의 각 점에서 연산을 병렬적으로 수행할 수 있다.

2.2 Look-up table

[4]에서 제안한 방식은 look-up table 을 이용한 것이다. CGH 를 구하는 절차는 다음 그림 1 과 같다.

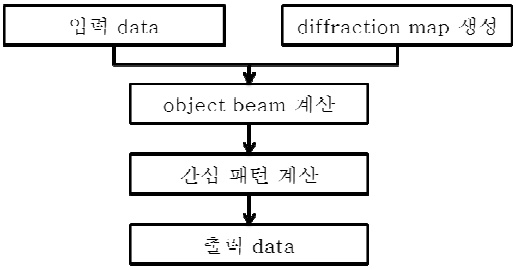

그림 1. CGH 생성 순서도

Fresnel diffraction map 을 계산할 때 look-up table 을 이용하게 된다. Map 은 거리, 깊이 변수인 z 에 따라 hologram 면에서의 회절 무늬를 미리 계산하여 저장된 것으로, object beam 을 계산 할 시에 보다 빠르게 연산 할 수 있다. 또한 object beam 계산시에 8 개의 core 를 사용하여 동작시킴으로 single-processor 일 때보다 8 배 빠른 연산을 기대한다. 그 후 data 동기화를 하고, reference beam 과 구해진 object beam 의 간섭 패턴을 계산하여 홀로그램을 구할 수 있다.

사용된 연산 식은 식(5)와 같다.

$$O_{(x,y)} = \frac{i}{\lambda z}\int_{x_0}\int_{y_0} A_{(x_0,y_0)} \cdot \exp\left[-jk \cdot \frac{(x-x_0)^2 + (y-y_0)^2 + 2z^2}{2z}\right] dx_0 dy_0$$
$$A_{(x_0,y_0)} = l_{(x_0,y_0)} \cdot \exp[random] \qquad (5)$$

2.3 알고리즘 비교

연산 최적화 방법은 기본적인 파라미터와 Γ_1, Δ_x 값 만을 이용하여 전체 CGH 를 구할 수 있다. Look-up table 을 이용할 시에는 미리 계산된 값들을 저장하는 공간이 필요하게 된다. 또한 각각의 processor 가 local memory 를 가지고 있다면 look-up table 역시 각 processor 마다 필요하기 때문에 더욱 큰 메모리 공간이 필요하다.

각 방법의 프로세서 통신량, 메모리량, 계산량은 표 1 과 같다. 통신량은 하나의 프로세서에 어떤 계산이 할당되었을 때 그 계산을 완료하기 위해 외부에서 전달해주어야 하는 데이터량이고, 메모리량은 계산을 위해 프로세서에 미리 저장되어 있어야 하는 데이터량이다. 고정된 시간 내에 하나의 프로세서에서 수행되는 연산을 계산량이라 한다. Object 의 크기가 N*N 이고, 홀로그램의 크기가 A*A, look-up table 의 크기는 LT 라고 하자.

프로세서 수	1 방법(연산 최적화)			2 방법(look-up table)		
	통신량	메모리량	계산량	통신량	메모리량	계산량
1	1	1	$(2N \times A^2)$ $+N^2$	1	LT	$N^2 \times A^2$
4	4	4	$\left(\frac{N}{2} \times A^2\right)$ $+N^2$	4	2× LT	$\frac{N^2}{16} \times A^2$
16	16	16	$\left(\frac{N}{8} \times A^2\right)$ $+N^2$	16	4× LT	$\frac{N^2}{256} \times A^2$
64	64	64	$\left(\frac{N}{32} \times A^2\right)$ $+N^2$	64	32× LT	$\frac{N^2}{4096} \times A^2$

표 1. CGH 생성 연산 비교

표 1 에서 통신량은 프로세서가 늘어나는데 비례해서 각각의 프로세서에 보내주어야 하는 데이터가 일정하게 늘어나는 것을 볼 수 있다. 메모리량은 1 방법에서 Γ_1, Δ_x 값만 메모리에 미리 가지고 있으면 되므로 아주 작은 양인 반면 2 방법에서는 각각의 processor 가 lookup table 을 가지고 있어야 하므로 메모리량은 lookup table 크기에 비례하여 커지게 된다. 2 방법의 계산량은 프로세서 수가 적을 경우 1 방법에 비해 상당히 큰 값을 가진다. 프로세서 수가 늘어날수록 2 방법은 제곱으로 연산량이 줄어들지만, 1 방식은 파라미터 계산량이 프로세서 수에 상관없이 일정하고, 줄어드는 연산량도 크지 않다.

3. 결론

본 논문에서는 multi-processor 에 적합한 CGH 연산에 대해 연구하기 위하여 연산 최적화 방법과 look-up table 을 이용한 방식을 비교해 보았다. 연산 최적화 방법의 경우, 기본 계산량과 메모리량이 적은 장점이 있다. Look-up table 방식은 메모리는 대용량이 필요하지만 프로세서 수가 늘어날수록 계산량이 크게 줄어든다. 대용량의 메모리와 프로세서의 수가 많은 시스템에서는 look-up table 방식이 적합하며, 적은 메모리와 프로세서를 가진 시스템에서는 연산 최적화 방법이 보다 효율적이다.

ACKNOWLEDGEMENT

본 논문은 2010 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2010-0028178, 2010-0015441)

참 고 문 헌

[1] T. Shimobaba, T. Ito, “An efficient computational method suitable for hardware of computer-generated hologram with phase computation by addition”, Computer Physics Communications, vol.138, no.1, pp.44-52, July 2001

[2] 류원현, 정만호, “3 차원 좌표변환에 의한 입체 컴퓨터 형성 홀로그램에 관한 연구”, 한국광학회지, vol.17, no.6, Dec. 2006

[3] 최현준, 서영호, 김동욱, “반복 가산 기법을 이용한 Fresnel 홀로그램의 고속 계산 알고리듬”, 한국통신학회논문지, vol.33, no.5, pp.386-394, May 2008

[4] R. Oi, T. Mishina, K. Yamamoto, T. Senoh, T. Kurita, “Fresnel hologram generation using an HD resolution depth range video camera”, SPIE Practical holography, vol.7619, Jan. 2010