특집논문 (Special Paper)
방송공학회논문지 제28권 제1호, 2023년 1월 (JBE Vol.28, No.1, January 2023)
https://doi.org/10.5909/JBE.2023.28.1.21
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

회절연산 정밀도에 따른 CGH 기반 홀로그램 생성 품질 분석

이 재 홍[a)], 김 덕 수[a)‡]

Quality Analysis on Computer Generated Hologram Depending on the Precision on Diffraction Computation

Jaehong Lee[a)] and Duksu Kim[a)‡]

요 약

컴퓨터 생성 홀로그래피는 일반 이미지에 비해 연산 부하와 메모리 요구량이 크다. 본 논문은 정밀도를 낮추어 연산속도를 높이는 저정밀도(low-precision) 및 혼합정밀도(mixed precision) 연산 방법을 회절연산에 적용하여, 정밀도에 따른 홀로그램의 생성 속도와 품질의 변화를 분석한다. 본 논문은 배정밀도, 단정밀도, bfloat16 정밀도에서의 회전 연산을 비교하였으며, bfloat16의 회절연산의 속도가 배정밀도에 비해 최대 5.94배, 단정밀도에 비해 1.52배 빠른 것을 확인하였다. 또한, MSE, PSNR, SSIM을 기준으로 회절 연산의 오차를 측정하였으며, 정밀도가 낮아질수록 홀로그램 품질이 낮아지는 것을 확인했다. 하지만, 정성적인 이미지 품질에는 유의미한 영향이 없는 것을 확인했다. 이러한 결과는, bfloat16등 낮은 정밀도 연산의 홀로그램 연산으로의 적용 가능성을 보여준다.

Abstract

Computer-generated holography requires much more computation costs and memory space rather than image processing. We implemented the diffraction calculation with low-precision and mixed-precision floating point numbers and compared the processing time and quality of the hologram with various precision. We compared diffraction quality with double, single and bfloat16 precision. bfloat16 shows 5.94x and 1.52x times faster performance than double precision and single precision. Also, bfloat16 shows lower PSNR and SSIM and higher MSE than other precision. However, there is no significant effect on reconstructed images. These results show low precision, like bfloat16, can be utilized for computer-generated holography.

Keyword : low-precision, CGH, holography, half-precision, bfloat16

a) 한국기술교육대학교 컴퓨터공학과(School of Computer Engineering, KOREATECH)
‡ Corresponding Author : 김덕수(Duksu Kim)
E-mail: bluekds@koreatech.ac.kr
Tel: +82-41-560-1498
ORCID: https://orcid.org/0000-0002-9075-3983

※ 이 논문은 2022년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받은 기초연구사업(No. 2021R1I1A3048263 40%), 지차제-대학 협력기반 지역혁신 사업(2021RIS-004, 20%), 그리고 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No.2019-0-00001, 40%)을 받아 수행된 연구임
· Manuscript December 2, 2022; Revised January 13, 2023; Accepted January 13, 2023.

Ⅰ. 서 론

홀로그래피(holography)는 빛의 파동을 홀로그램에 기록하는 기술로 빛의 세기(intensity)를 기록하는 사진(photography)과 다르게 기록한 장면의 3차원 정보를 재현할 수 있다는 장점을 가진다. 컴퓨터 생성 홀로그래피(computer generated holography, CGH)는 컴퓨터 연산을 통해 홀로그램을 생성하는 방법이다. CGH는 공간적, 시간적 제약을 넘어 가상 공간을 활용하여 홀로그램을 생성할 수 있다는 장점을 가진다. 하지만 홀로그램은 일반 이미지나 영상 대비 수~수십 배 높은 해상도를 요구하며 복소수 연산을 요구함에 따라, 연산 부하와 메모리 요구량이 크다. 이러한 연산적 공간적 요구 문제를 해결하기 위해, 다양한 CGH 가속 알고리즘들과 특화 하드웨어 장치들이 연구되고 있다[1-4].

CGH는 빛의 회절(diffraction) 연산에 기반을 두며, 회절 연산은 복소수 행렬 연산과 푸리에 변환(Fourier transform)을 포함한다. 푸리에 변환은 주로 고속 푸리에 변환(fast Fourier transform, FFT)을 통해 수행된다. FFT는 이산 푸리에 변환(discrete Fourier transform, DFT)의 연산 복잡도를 낮춘 고속화 알고리즘으로, CGH 뿐만 아니라, 컴퓨터 비전, 파동 분석, 인공지능, 음향 기술 등 폭넓게 사용된다[5,6]. FFT 연산 속도를 높이기 위해, 다중코어 CPU, SIMD (Single Instruction, Multiple Data), GPU(Graphics Processing Unit) 등과 같은 병렬처리 장치를 활용하는 기술들이 연구되어 왔다[7-16]. 대표적인 병렬처리 기반 FFT 라이브러리에는 oneAPI[17], cuFFT[18], rocFFT[19], 그리고 FFTW[20,21] 등이 있다.

일반적인 연산장치(또는 산술연산코어)에서 연산 정밀도와 속도는 반비례한다. 따라서, 정확도를 일부 포기하며 연산 정밀도를 낮추면, 더 높은 연산 처리 성능을 기대할 수 있다. 정밀도를 낮춘다는 것은 또한 데이터의 크기도 작아진다는 것을 의미한다. 따라서, 데이터 접근시간이 감소하며, 한 번에 처리할 수 있는 데이터의 크기가 증가한다는 장점을 가진다. 최신 Nvidia사의 GPU들은 낮은 정밀도의 행렬 곱셈에 특화된 연산장치인 텐서 코어(tensor core)를 가지고 있으며, bfloat16(brain floating point)[22,23]과 TF32(TensorFloat-32)등의 다양한 산술정밀도를 지원한다. 최근 인공지능, 빅데이터 연구 분야에서는 연산 정밀도(precision)를 낮추고 연산속도를 높이는 반정밀도(half precision, fp16) 연산과 혼합정밀도(mixed precision) 연산을 채용하고 있다[23,24].

부동소수점(floating-point)에서 소수부에 대한 정확도는 사용하는 비트의 수에 따라 달라진다[25]. 따라서 정밀도가 낮을수록 연산결과의 정확도가 낮아진다. 또한, 연산을 거듭해서 수행할수록 연산오류가 누적되어 오차가 커진다. 홀로그램 연산은 부동소수점을 이용한 복소수 연산을 반복한다. 그리고 수의 범위가 10-9-105 수준으로 넓기 때문에, 정밀도가 낮은 경우 홀로그램의 품질이 크게 저하될 수 있다. 아울러, 푸리에 변환은 행렬의 모든 원소에 인자를 곱하고 더하는 과정을 거치기 때문에, 행렬의 크기가 커질수록 연산횟수가 가파르게 증가하며, 정밀도 오차 더욱 커질 수 있다. 따라서, 적정 수준의 품질을 유지하면서 연산 속도를 높일 수 있는 정밀도에 대한 연구가 필요하다. 본 연구는 정밀도에 따라 생성된 홀로그램 결과를 비교 및 분석한다. 연구에 사용한 정밀도는 배정밀도(double-precision), 단정밀도(single-precision), 그리고 bfloat16를 포함하며, 각 스펙트럼 방법(angular spectrum method)을 회절에 연산에 사용하였다.

각각의 정밀도별 회절 연산을 비교분석한 결과 bfloat16이 배정밀도와 단정밀도 대비 최대 5.94배, 1.52배 빠른 연산 성능을 보여주었다. 또한, 해상도가 증가하면서 정밀도에 의한 오차가 홀로그램 정성적 품질에는 큰 영향을 끼치지 않음을 보여주었다. 이러한 결과는, 낮은 정밀도 연산을 CGH에 적용할 수 있는 가능성을 보여준다.

Ⅱ. 관련연구

CGH 생성방식은 3차원 장면의 샘플링 방법에 따라 포인트(point), 메시(mesh), 레이어(layer), 그리고 라이트필드(lightfield) 등으로 분류할 수 있다[2]. 메시, 레이어, 라이트필드 등은 주로 푸리에 변환을 이용하며, 포인트 CGH는 WRP(Wavefront Recording Plane) 방법[26]을 이용할 때 푸리에 변환을 사용한다. 최근 고성능 컴퓨팅 분야에서 방대한 데이터를 빠르게 처리하기 위해 정밀도를 낮추고 연산 속도와 메모리 이익을 얻는 방법이 연구되고 있다[23, 27-32].

이러한 저 정밀도 연산은 홀로그램에도 적용될 수 있다. David Blinder와 Peter Schelkens(2021)는 삼각함수를 8비트 정수로 연산하여 포인트 기반 홀로그램 생성 속도를 높이는 방법을 제안하였다[33]. 8비트 연산을 이용하여 낮아진 정밀도를 보간하여 약 1.3배, 정밀도 보간 없이 최대 3.1배 빠른 홀로그램 생성 속도를 보인다.

텐서 코어는 합성곱 연산에 특화된 연산가속장치로 혼합정밀도를 사용하여 기존 GPU코어 대비 2.5-12배 더 빠른 성능을 보인다[34]. FFT는 연산과정에 행렬 곱셈을 이용하기 때문에 텐서 코어를 활용하는 것이 가능하다[35-38]. A. Sorna 외(2018)는 텐서코어를 이용하여 FFT 연산을 가속화하고 낮은 정밀도에 의한 오차를 줄이는 방법을 제안하였다. 연산 속도가 cuFFT에 비해 낮은 문제가 있었지만, Binrui Li 외(2021), Sultan Durrani 외(2021), Louis Pisha와 Łukasz Ligowski(2021) 등의 연구에서 텐서코어를 활용하여 cuFFT에 비해 높은 연산속도를 갖는 FFT 연산 방법을 제안하였다[35,37,38].

Ⅲ. 부동소수점과 정밀도

컴퓨터는 실수를 표현하기 위해서 부동소수점을 이용한다. 부동소수점은 IEEE754 표준에 따라서 배정밀도, 단정밀도, 반정밀도 등으로 분류된다[25]. bfloat16는 최근 TPU(Tensor Processing Unit), GPU 등의 각종 전용 프로세서에서 지원하고 있는, 반정밀도의 단점을 완화하기 위한 정밀도이다. CPU는 일반적으로 단정밀도와 배정밀도를 동일한 부동소수점 처리장치에서 처리하기 때문에 연산속도 차이가 미미하다. 반면, GPU의 연산코어는 정밀도별 연산속도가 다르다. 최근 GPU는 각 스트리밍 코어(streaming core)에 SIMD연산을 도입하여 반정밀도와 같은 낮은 정밀도 연산을 가속한다. 특히, Nvidia의 텐서 코어는 반정밀도와 단정밀도를 혼합한 혼합정밀도를 지원하며 반정밀도의 오차를 줄이면서도, 단정밀도보다 빠른 연산 성능을 보여준다[34].

본 연구는 배정밀도, 단정밀도, 반정밀도, bfloat16를 활용한 홀로그램 생성 품질을 비교한다. 그림 1은 각 부동소수점의 정밀도 별 비트 크기와 구성을 보여준다. 부동소수점 방식은 실수를 부호비트(sign), 가수부(exponent), 소수부(fraction)의 조합으로 표현한다. 부호비트는 양수와 음수를 판단하는 비트이며 가수부는 숫자의 자릿수를 판단한다. 소수부는 표현하는 수의 소수점을 표현한다. 부호비트의 크기는 모든 정밀도가 1비트로 동일하며 가수부와 소수부는 정밀도에 따라 크기가 다르다. 부동소수점의 표현 방법은 수식 1과 같다.

$$\begin{cases} n=(-1)^S\times(\sum_{i=1}^{F}1+b_{F-i}2^{-i})\times 2^{e-E} & \text{if } e>0 \\ n=(-1)^S\times(\sum_{i=1}^{F}b_{F-i}2^{-i})\times 2^{1-E} & \text{if } e=0 \end{cases} \quad (1)$$

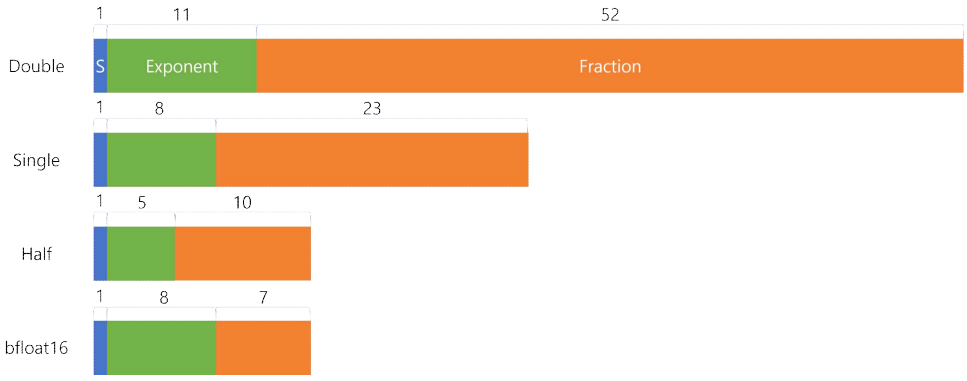

그림 1. 부동소수점의 비트 구성
Fig. 1. Structure of floating point numbers

n은 부동소수점으로 표현된 수, S는 부호비트, F는 소수부의 크기, E는 가수부의 바이어스(bias), e는 가수부의 값(0~2e), b_x는 x번째 비트 값이다. F가 클수록 소수점을 표현할 수 있으며 E가 클수록 더 넓은 범위의 수를 표현하는 것이 가능하다. 예를 들어, 배정밀도의 가수부는 11비트로 $[2^{-1023}, 2^{1024}]$의 자릿수를 표현할 수 있으며 소수부는 52비트로 1~(2^{-252})의 수를 표현할 수 있다. 단정밀도의 가수부는 8비트로 $[2^{-126}, 2^{127}]$의 자릿수를 표현할 수 있으며 소수부는 23비트로 2^{23}개의 수를 표현할 수 있다. 반정밀도의 경우, 가수부의 길이가 5비트로 $[2^{-14}, 2^{15}]$의 수를 표현할 수 있다.

부동소수점에서 e=0인 경우를 준 정규 값(subnormal number)이라 한다. 준 정규 값은 언더플로우(underflow)를 막기 위한 값으로 일반적인 부동소수점 표현방식에 차이가 있으며, 연산속도가 달라질 수 있다. 따라서, 본 논문에서는 준 정규 값을 배제하고 정규 값만을 비교 분석한다.

부동소수점이 표현할 수 있는 가장 작은 수는 수식 2, 가장 큰 수는 수식 3과 같다. 예를 들어, 반정밀도의 경우, F=10, E=15 이므로 표현가능한 가장 작은 수는 $(1+2^{-10})\times 2^{-14} = 6.10*10^{-5}$이며, 가장 큰 수는 $(2-2^{-10})\times 2^{15} = 65{,}504$ 이다. 표 1은 각 정밀도별 표현가능한 수를 보여준다.

$$n_{\min} = (1+2^{-F}) \times 2^{-E+1} \tag{2}$$

$$n_{\max} = (1+1-2^{-F}) \times 2^{E} \tag{3}$$

Ⅳ. 회절연산 구현

회절연산은 홀로그램 면에 도달하는 빛의 파동을 홀로그램 면에 기록하고 분석한다. 회절연산은 주로 각 스펙트럼 방법(angular spectrum method)과 프레넬 회절(Fresnel diffraction)을 이용한다. 본 논문의 실험에서는 각 스펙트럼 방법을 사용한다. 각 스펙트럼 방법은 수식 4와 같다. 수식 4의 연산절차와 본 연구에서 사용한 정밀도는 그림 2와 같다. 수식 4는 Forward FFT, Scaling, Multiply Transfer, Backward FFT 순으로 수행된다. 본 연구에서는 배정밀도

표 1. 부동소수점 정밀도 표현 범위
Table 1. Value range of floating point numbers

	double precision	single precision	half precision	bfloat16
exponent	11	8	5	8
fraction(F)	52	23	10	7
bias(E)	1023	127	15	127
minimum value	$\approx 2.225*10^{-308}$	$\approx 1.175*10^{-38}$	$\approx 6.109*10^{-5}$	$\approx 1.185*10^{-38}$
maximum value	$\approx 1.798*10^{308}$	$\approx 3.403*10^{38}$	$65{,}504$	$\approx 3.390*10^{38}$

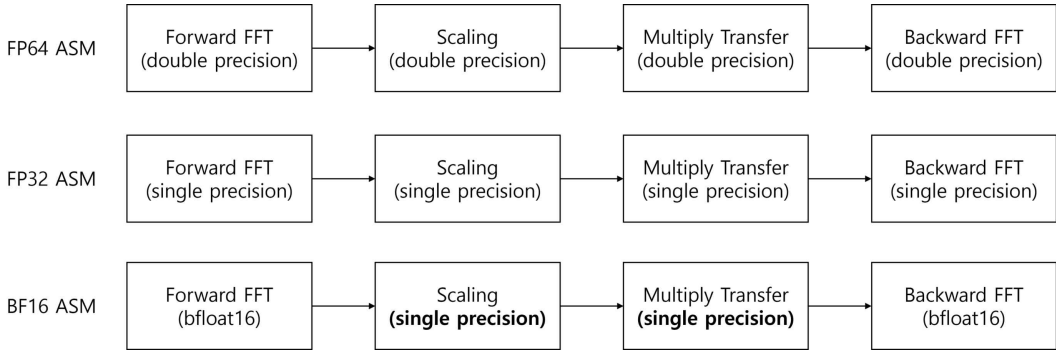

그림 2. 각 스펙트럼 방법의 처리 순서와 연산에 적용한 정밀도(FP64, FP32 and BF16)
Fig. 2. Procedures of ASM(angular spectrum method) and used precisions(FP64, FP32 and BF16)

(FP64), 단정밀도(FP32), bfloat16(BF16) 정밀도를 이용하며 Scaling과 Multiply Transfer 연산에는 BF16 대신 FP32를 이용하였다 (그림 2).

$$u(x,y,z)=F^{-1}\left(F(u(x,y,0))\times H(f_x,f_y,z)\right) \quad (4)$$

$$H(f_x,\ f_y,z)=\exp\left(kzi\sqrt{1-\lambda^2\left(f_x^2-f_y^2\right)}\ \right) \quad (5)$$

각 스펙트럼 방법을 이용하여 평면파 u(x,y,0)를 u(x,y,z)로 전파하기 위해서 푸리에 변환 F(수식 6)와 역푸리에 변환 F^{-1}(수식 7)를 이용한다. x, y, z는 3차원 공간좌표, f_x, f_y는 푸리에 도메인의 공간 좌표이다. k는 파수(wave number), λ는 빛의 파장이다. $H(f_x,\ f_y,z)$는 각 스펙트럼 방법의 전달 함수로 수식 5와 같다. × 는 아다마르 곱(Hadamard product)이다.

고속 푸리에 변환: 본 연구는 cuFFT[18]를 사용하여 고속 푸리에 변환을 수행하였다. cuFFT는 배정밀도, 단정밀도, 반정밀도와 bfloat16을 지원하지만, 반정밀도와 bfloat16에 대해서는 크기가 2n인 행렬만 처리할 수 있다는 제약이 있다. 따라서, 본 연구의 실험에는 크기가 2n인 행렬이 사용되었다. cuFFT는 공간주파수 도메인의 영점을 조절할 수 있는, 중심 이산 푸리에 변환(centered discrete Fourier transform, CDFT)[39,40] 또는 FFT-shift[14,41] 기능을 제공하지 않는다. 따라서, 본 연구에서는 FFT-shift를 CUDA 환경에서 구현하여 FFT 수행 전, 후로 FFT-shift를 수행하였다.

스케일링(scaling): cuFFT는 역푸리에 변환 연산 시 Inverse FFT가 아닌 backward FFT를 수행한다. backward FFT의 수식 7과 같다.

$$FFT_{forward}(u(x))=U(f_x)=\sum_{x=0}^{N-1}u(x)\ \cdot\ exp(-2\pi i\frac{xf_x}{N}) \quad (6)$$

$$\begin{aligned}FFT_{backward}(U(f_x))&=N\cdot u(x)\\&=\sum_{f_x=0}^{N-1}U(f_x)\ \cdot\ exp(2\pi i\frac{xf_x}{N})\end{aligned} \quad (7)$$

backward FFT(수식 7)는 forward FFT(수식 6)의 값에 벡터 크기 N을 곱한 수를 나타낸다. 따라서, 본래 u(x) 값을 얻기 위해서는 backward FFT 값을 N으로 나누어야 한다.

오버플로우와 언더플로우: 스케일링 과정에서 행렬의 크기가 약 1,024*1,024일 때, 약 10^6에 해당하는 값을 나누어야 한다. 이는 반정밀도의 최대 표현범위(65,504)를 초과한다. 전달함수연산은 빛의 파장(5.32e-9)를 제곱하고 곱하는 등의 높은 정밀도를 요구하기 때문에 반정밀도의 최소 표현 값(6.10e-5)의 범위를 벗어나는 문제가 발생한다. 이에 따라, 반정밀도는 본 논문의 실험대상에서 제외하였다. bfloat16 정밀도는 반정밀도에 비해 값을 표현할 수 있는 범위가 크지만 소수부의 크기가 매우 작기 때문에 스케일링과 전달함수 곱셈 수행 시 오차가 크게 증가하는 것을 확인하였다. 따라서, bfloat16의 두 연산은 단정밀도 연산으로 대체하였다 (그림 2).

V. 홀로그램 품질 비교 및 분석

홀로그램 품질 비교를 위해서 본 연구에서는 그림 3과 같이 각 스펙트럼 방법의 정밀도별 오차를 비교하였다. 각 스펙트럼 방법을 이용하여 이미지 u(x,y,0)를 z축 이동하여 u(x,y,z) 로 전파한 후, u(x,y,z)를 -z축 이동하여 u'(x,y,0)으로 복구한다. 이때 원본 이미지 u(x,y,0)와 전파 이미지 u'(x,y,0)를 비교하여 각 스펙트럼 방법의 정밀도별 파동의 진폭(amplitude) 오차와 연산시간을 분석하였다.

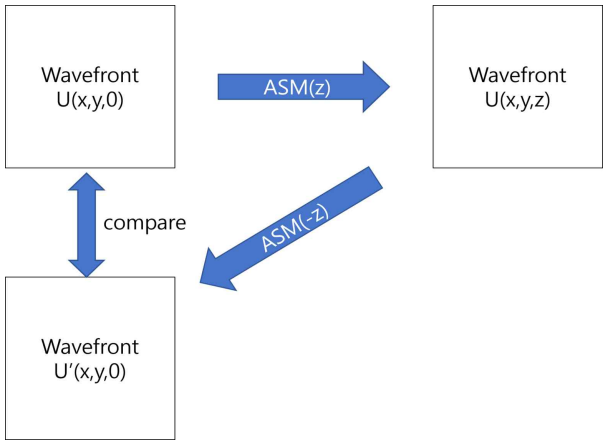

그림 3. 홀로그램 품질 비교 방법
Fig. 3. Comparing method for quality of a hologram

표 2. 정밀도별 각 스펙트럼 방법 연산시간, MSE, PSNR, SSIM
Table 2. Processing time, MSE, PSNR, SSIM of angular spectrum method in three precisions

resolution	precision	size of data (MB)	processing time (ms)	MSE(↓)	PSNR(↓)	SSIM(↑)
8192*8192	double	2048	87.86	3.62E-31	306.48	1
8192*8192	single	1024	31.63	2.29E-13	128.05	1
8192*8192	bfloat16	512	20.80	1.08E-05	51.48	0.99994
16384*16384	double	8192	445.76	4.58E-31	305.66	1
16384*16384	single	4096	102.27	1.66E-13	129.59	1
16384*16384	bfloat16	2048	74.99	2.61E-05	48.33	0.99986

본 실험을 위해 사용한 연산장치는 RTX 3090 24GB이며 CUDA 11.7 환경에서 실험을 수행하였다. 실험에 사용한 각 스펙트럼 방법의 해상도는 8,192*8,192, 16,384*16,384로 DIV2K[42]의 검증 세트를 선형 보간법을 통해 보간 하였다. 각 스펙트럼 연산에 이용한 파라미터는 λ=532nm, pixel pitch=8μm, z=0.09m이다. 정밀도는 배정밀도, 단정밀도, bfloat16을 이용하였다. 반정밀도의 경우, 표현 가능한 수의 범위가 파라미터에 미치지 못하여(4 장) 실험대상에서 제외하였다. 표 1는 각 정밀도 별 연산시간과 MSE(Mean Squared Error), PSNR(Peak Signal-to-Noise Ratio), SSIM(Structural Similarity Index Measure)을 보여준다. MSE, PSNR, SSIM은 각각 DIV2K 검증 세트를 기준으로 계산하였으며, MSE와 PSNR은 낮을수록, 그리고 SSIM은 높을수록 검증 세트와 유사한(정확한) 결과임을 말한다. 8,192*8,192 해상도의 경우, 배정밀도가 단정밀도에 비해 각각 2.78배, bfloat16에 비해 4.22배 연산시간이 더 많이 소요하였다. 이미지의 품질면에서는, 정밀도가 낮아질수록 MSE와 PSNR은 증가하고, SSIM은 낮아지는 결과를 보여준다. 즉, 정량적인 품질은 낮아짐을 의미한다. 하지만, 정밀도가 가장 낮은 bfloat16에서 MSE가 1.08e-05 수준으로, 일반 이미지의 양자화 수준인 8비트(256단계)에 매주 작은 수치이며, 정성적인 비교에서도 큰 차이를 확인하기 어려움을 확인했다(그림 5).

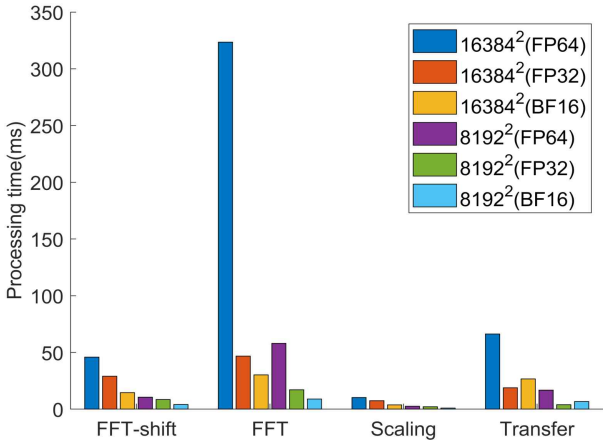

그림 4. 정밀도(FP64, FP32, BF16)에 따른 각 스펙트럼 방법의 수행절차별 연산시간 비교
Fig. 4. Processing time of sub-steps of angular spectrum method with FP64, FP32 and BF16(bfloat16)

해상도가 16,384*16,384인 경우, 배정밀도의 연산시간이 단정밀도의 4.36배, bfloat16의 5.94배 소요되는 것을 확인할 수 있다. 이는 8,192*8,192에 비해 증가한 수치이며, 해상도가 커질수록 연산시간의 차이는 커짐을 보여준다. MSE, PSNR, SSIM 또한, 8,192*8,192 해상도의 경우에 비해 차이가 커졌으며, bfloat16의 MSE 수치가 높아졌고, PSNR과 SSIM 수치가 낮아졌다. 하지만, 여전히 정성적인 이미지 품질에 영향을 주기에는 작은 수치로, 홀로그램의

표 3 정밀도에 따른 각 스펙트럼 방법의 수행절차별 연산시간 비교(ms)
Table 3. Processing time of sub-steps of angular spectrum method with FP64, FP32 and BF16(bfloat16)

	16384^2 $(FP64)$	16384^2 $(FP32)$	16384^2 $(BF16)$	8192^2 $(FP64)$	8192^2 $(FP32)$	8192^2 $(BF16)$
FFT-shift	45.9	29.1	14.5	10.5	8.5	4
FFT	323.5	46.8	30.2	58	17	9
Scaling	10.2	7.5	3.7	2.6	2.2	1
Transfer	66.2	18.8	26.7	16.7	3.9	6.7

유의미한 품질 변화를 확인하기 어렵다 (그림 5).

단정밀도와 bfloat16의 연산속도를 비교하였을 때, 단정밀도의 연산시간이 8,192*8,192에서 1.52배, 16,384*16,384에서 1.36배 더 많이 소요되는 것으로 나타났다. 해상도가 커질수록 연산속도의 차이가 줄어드는 것으로 파악되었다. 이는 해상도가 커질수록 FFT 연산시간의 차이가 줄어들기 때문이다. 그림 4와 표 2은 정밀도가 각 스펙트럼 연산 과정에 미치는 영향을 분석하기 위해 각 스펙트럼 방법 연산 과정의 수행시간을 비교하였다. 본 연구에서 각 스펙트럼 방법의 연산과정은 그림 2와 같지만, Forward FFT와 Backward FFT 두 연산은 동일한 연산절차 (FFT-shift, cuFFT, FFT-shift)를 수행한다. 따라서, 그림 4와 표 3에서 FFT-shift, FFT, Scaling, Transfer(전달 함수)의 연산시간을 비교하였다. bfloat16과 단정밀도의 FFT 연산시간의 차이

그림 5 회절연산 이미지 비교
Fig. 5. Comparison of diffraction images

가 8,192*8,192는 1.89배, 16,384*16,384는 1.55배로 줄어드는 것을 확인할 수 있다. 또한, 전달함수 곱셈의 경우, bfloat16이 단정밀도 보다 많은 연산시간 소요하는 것을 확인할 수 있다. 본 연구의 실험에서, bfloat16은 정밀도가 낮아 전달함수를 표현하기 어렵기 때문에, 전달함수(수식 5)는 단정밀도로 연산하도록 하였다(4 장). 따라서, bfloat16와 단정밀도 사이의 데이터 변환 과정이 필요하며, 이 부하(overhead)로 전달함수 계산에 있어 bfloat16이 단정밀도에 비해 느린 현상이 발생한다. 하지만, 전달함수 외 다른 연산 부분에서는 데이터의 크기와 연산시간이 비례하는 것을 확인할 수 있다.

Ⅵ. 결 론

본 논문은 CGH에서 핵심 연산인 회절 연산, 그중 각 스펙트럼 방법에 대한 연산 정밀도의 영향을 분석하였다. 배정밀도, 단정밀도, 그리고 bfloat16을 이용한 회절 연산을 수행하고, 생성된 홀로그램의 품질을 분석하였다. 그 결과, 반정밀도는 수 표현 범위의 한계로 홀로그램 생성에 사용하기 어렵지만, 같은 데이터 크기를 가지는 bfloat16은 데이터의 표현범위가 반정밀도에 비해 넓으며 홀로그램 연산에 대한 적용이 가능하다는 것을 확인하였다. 하지만, 일부 연산(예, 전달함수와 스케일링)에서는 bfloat16를 사용할 수 없음을 발견하였다. 따라서, 전달함수와 스케일링에 대해서는 단정밀도와 혼합하여 각 스펙트럼 방법을 구현하였다. 그 결과, bfloat16에 기반을 둔 회절 연산으로 준수한 품질의 홀로그램을 생성할 수 있었다. 또한, bfloat16를 사용함으로써, 배정밀도 및 단정밀도 사용 대비 각각 최대 5.94배, 1.52배 높은 연산 성능을 달성할 수 있음을 확인하였다. 또한, 생성된 홀로그램 품질에 유의미한 변화는 없는 것을 확인하였다.

본 논문에서는 낮은 정밀도를 이용한 홀로그램 연산의 가능성과 높은 연산성능을 확인하였다. 하지만, bfloat16의 정밀도 오차가 홀로그램의 해상도가 증가하면서 증가하는 것을 확인했다. 향후 연구에서는 해상도 증가에 따른 오차 증가를 상쇄할 수 있는 방법에 대해 연구하고자 한다. 특히, 홀로그램 연산에 사용하는 파라미터(예: 픽셀간격, 빛의 파장)등은 값이 너무 작기 때문에 연산 오차를 증가시킨다는 점에 착안하여, 파라미터의 값을 스케일링하여 연산할 수 있는 방법을 연구하고자 한다. 또한, 최근 텐서코어를 이용하여 합성곱 외 연산문제를 해결하는 방법이 연구되고 있으며, 홀로그램 생성에도 텐서코어를 이용하는 방법도 연구하고자 한다.

참 고 문 헌 (References)

[1] R. Corda, D. Giusto, A. Liotta, W. Song, and C. Perra, "Recent Advances in the Processing and Rendering Algorithms for Computer-Generated Holography," Electronics, vol. 8, no. 5, 2019, doi: https://doi.org/10.3390/electronics8050556.

[2] T. Shimobaba and T. Ito, Computer Holography: Acceleration Algorithms and Hardware Implementations. CRC press, 2019.

[3] K. Matsushima, Introduction to Computer Holography: Creating Computer-Generated Holograms as the Ultimate 3D Image. Springer Nature, 2020.

[4] Y. Wang et al., "Hardware implementations of computer-generated holography: a review," Optical Engineering, vol. 59, no. 10, 2020. doi: https://doi.org/10.1117/1.Oe.59.10.102413.

[5] Y. Yuan, G. Xun, K. Jia, and A. Zhang, "A Multi-view Deep Learning Method for Epileptic Seizure Detection using Short-time Fourier Transform," presented at the Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics, 2017. doi: https://doi.org/10.1145/3107411.3107419.

[6] R. N. Bracewell, The Fourier transform and its applications. McGraw-hill New York, 1986.

[7] O. Yasuhito, E. Toshio, M. Naoya, and M. Satoshi, "An efficient, model-based CPU-GPU heterogeneous FFT library," presented at the 2008 IEEE International Symposium on Parallel and Distributed Processing, 2008. doi: https://doi.org/10.1109/IPDPS.2008.4536163.

[8] L. Gu, J. Siegel, and X. Li, "Using GPUs to compute large out-of-card FFTs," presented at the Proceedings of the international conference on Supercomputing, Tucson, Arizona, USA, 2011. [Online]. doi: https://doi.org/10.1145/1995896.1995937.

[9] S. Chen and X. Li, "A hybrid GPU/CPU FFT library for large FFT problems", 2013 IEEE 32nd International Performance Computing and Communications Conference (IPCCC), 6-8 Dec. 2013. doi: https://doi.org/10.1109/PCCC.2013.6742796.

[10] A. Gholami, J. Hill, D. Malhotra, and G. Biros, "AccFFT: A library for distributed-memory FFT on CPU and GPU architectures," arXiv preprint arXiv:1506.07933, 2015.

[11] Z. Zhao and Y. Zhao, "The Optimization of FFT Algorithm Based with Parallel Computing on GPU," 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference

(IAEAC), 12-14 Oct. 2018.
doi: https://doi.org/10.1109/IAEAC.2018.8577843.

[12] D. Takahashi, Fast Fourier transform algorithms for parallel computers. Springer, 2019.

[13] H. Kang, J. Lee, and D. Kim, "HI-FFT: Heterogeneous Parallel In-Place Algorithm for Large-Scale 2D-FFT," IEEE Access, vol. 9, pp. 120261-120273, 2021.
doi: https://doi.org/10.1109/ACCESS.2021.3108404.

[14] J. Lee, H. Kang, H.-j. Yeom, S. Cheon, J. Park, and D. Kim, "Out-of-core GPU 2D-shift-FFT algorithm for ultra-high-resolution hologram generation," Opt. Express, vol. 29, no. 12, pp. 19094-19112, 2021/06/07 2021.
doi: https://doi.org/10.1364/OE.422266.

[15] D. Sharp, M. Stoyanov, S. Tomov, and J. Dongarra, "A More Portable HeFFTe: Implementing a Fallback Algorithm for Scalable Fourier Transforms," 2021 IEEE High Performance Extreme Computing Conference (HPEC), 20-24 Sept. 2021 2021, pp. 1-5.
doi: https://doi.org/10.1109/HPEC49654.2021.9622811.

[16] A. Ayala, S. Tomov, M. Stoyanov, A. Haidar, and J. Dongarra, "Performance Analysis of Parallel FFT on Large Multi-GPU Systems," 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2022.
doi: https://doi.org/10.1109/IPDPSW55747.2022.00072.

[17] Intel® oneAPI Math Kernel Library, https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html#gs.i3mm29 (accessed Nov. 24, 2022).

[18] cuFFT :: CUDA Toolkit Documentation, https://docs.nvidia.com/cuda/cufft/index.html (accessed Nov. 24, 2022).

[19] rocFFT API, https://docs.amd.com/bundle/rocFFT-release-rocm-rel-5.2/page/library.html (accessed Nov. 24, 2022).

[20] M. Frigo and S. G. Johnson, "The Design and Implementation of FFTW3," Proceedings of the IEEE, vol. 93, no. 2, pp. 216-231, 2005.
doi: https://doi.org/10.1109/JPROC.2004.840301.

[21] M. Frigo and S. G. Johnson, "FFTW: an adaptive software architecture for the FFT," Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181), 15-15 May 1998 1998, vol. 3, M. Frigo S. G. Johnson, pp. 1381-1384 vol.3.
doi: https://doi.org/10.1109/ICASSP.1998.681704.

[22] S. Wang and P. Kanwar, "BFloat16: The secret to high performance on Cloud TPUs," Google Cloud Blog, vol. 4, 2019.

[23] N. Burgess, J. Milanovic, N. Stephens, K. Monachopoulos, and D. Mansell, "Bfloat16 Processing for Neural Networks," 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH), 2019.
doi: https://doi.org/10.1109/ARITH.2019.00022.

[24] Y. Kikuchi, K. Fujita, T. Ichimura, M. Hori, and L. Maddegedara, "Calculation of Cross-correlation Function Accelerated by Tensor Cores with TensorFloat-32 Precision on Ampere GPU," Computational Science – ICCS 2022, (Lecture Notes in Computer Science, 2022, ch. Chapter 37, pp. 277-290.
doi: https://doi.org/10.1007/978-3-031-08754-7_37.

[25] "IEEE Standard for Floating-Point Arithmetic," IEEE Std 754-2019 (Revision of IEEE 754-2008), pp. 1-84, 2019.
doi: https://doi.org/10.1109/IEEESTD.2019.8766229.

[26] T. Shimobaba, N. Masuda, and T. Ito, "Simple and fast calculation algorithm for computer-generated hologram with wavefront recording plane," Opt. Lett., vol. 34, no. 20, pp. 3133-3135, 2009/10/15 2009.
doi: https://doi.org/10.1364/OL.34.003133.

[27] M. Seznec, N. Gac, A. Ferrari, and F. Orieux, "A Study on Convolution using Half-Precision Floating-Point Numbers on GPU for Radio Astronomy Deconvolution," 2018 IEEE International Workshop on Signal Processing Systems (SiPS), 21-24 Oct. 2018 2018, pp. 170-175,
doi: https://doi.org/10.1109/SiPS.2018.8598342.

[28] C. Maass, M. Baer, and M. Kachelriess, "CT image reconstruction with half precision floating-point values," Med Phys, vol. 38 Suppl 1, p. S95, Jul 2011.
doi: https://doi.org/10.1118/1.3528218.

[29] N. M. Ho and W. F. Wong, "Exploiting half precision arithmetic in Nvidia GPUs," 2017 IEEE High Performance Extreme Computing Conference (HPEC), 12-14 Sept. 2017 2017, pp. 1-7.
doi: https://doi.org/10.1109/HPEC.2017.8091072.

[30] P. Luszczek, J. Kurzak, I. Yamazaki, and J. Dongarra, "Towards numerical benchmark for half-precision floating point arithmetic," 2017 IEEE High Performance Extreme Computing Conference (HPEC), 12-14 Sept. 2017 2017, pp. 1-5.
doi: https://doi.org/10.1109/HPEC.2017.8091031.

[31] A. Abdelfattah, S. Tomov, and J. Dongarra, "Towards Half-Precision Computation for Complex Matrices: A Case Study for Mixed Precision Solvers on GPUs," 2019 IEEE/ACM 10th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), 2019.
doi: https://doi.org/10.1109/ScalA49573.2019.00008.

[32] A. Abdelfattah, S. Tomov, and J. Dongarra, "Fast Batched Matrix Multiplication for Small Sizes Using Half-Precision Arithmetic on GPUs," 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2019.
doi: https://doi.org/10.1109/IPDPS.2019.00022.

[33] D. Blinder and P. Schelkens, "Fast Low-Precision Computer-Generated Holography on GPU," Applied Sciences, vol. 11, no. 13, 2021.
doi: https://doi.org/10.3390/app11136235.

[34] S. Markidis, S. W. D. Chien, E. Laure, I. B. Peng, and J. S. Vetter, "NVIDIA Tensor Core Programmability, Performance & Precision," 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2018.
doi: https://doi.org/10.1109/IPDPSW.2018.00091.

[35] L. Pisha and L. Ligowski, "Accelerating non-power-of-2 size Fourier transforms with GPU Tensor Cores," 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2021.
doi: https://doi.org/10.1109/IPDPS49936.2021.00059.

[36] A. Sorna, X. Cheng, E. D. Azevedo, K. Won, and S. Tomov, "Optimizing the Fast Fourier Transform Using Mixed Precision on Tensor Core Hardware," 2018 IEEE 25th International Conference on High Performance Computing Workshops (HiPCW), 17-20 Dec. 2018

2018, pp. 3-7.
doi: https://doi.org/10.1109/HiPCW.2018.8634417.

[37] B. Li, S. Cheng, and J. Lin, "tcFFT: A Fast Half-Precision FFT Library for NVIDIA Tensor Cores," 2021 IEEE International Conference on Cluster Computing (CLUSTER), 2021.
doi: https://doi.org/10.1109/Cluster48925.2021.00035.

[38] S. Durrani et al., "Accelerating Fourier and Number Theoretic Transforms using Tensor Cores and Warp Shuffles," 2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT), 2021.
doi: https://doi.org/10.1109/PACT52795.2021.00032.

[39] D. H. Mugler, "The Centered Discrete Fourier Transform and a parallel implementation of the FFT," 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 22-27 May 2011 2011, pp. 1725-1728.
doi: https://doi.org/10.1109/ICASSP.2011.5946834.

[40] J. G. Vargas-Rubio and B. Santhanam, "On the multiangle centered discrete fractional Fourier transform," IEEE Signal Processing Letters, vol. 12, no. 4, pp. 273-276, 2005.
doi: https://doi.org/10.1109/lsp.2005.843762.

[41] M. Abdellah, "cufftShift: high performance CUDA-accelerated FFT-shift library," Proceedings of the High Performance Computing Symposium, Tampa, Florida, 2014.

[42] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2017, pp. 126-135.
doi: https://doi.org/10.1109/CVPRW.2017.150.

저 자 소 개

이 재 홍

- 2020년 2월 : 한국기술교육대학교 컴퓨터공학부 학사
- 2022년 2월 : 한국기술교육대학교 일반대학원 컴퓨터공학 석사
- 2022년 3월 ~ 현재 : 한국기술교육대학교 일반대학원 컴퓨터공학 박사과정
- ORCID : https://orcid.org/0000-0002-8311-5975
- 주관심분야 : 고성능 컴퓨팅, 컴퓨터 생성 홀로그래피

김 덕 수

- 2008년 성균관대학교 정보통신고학부 (공학사)
- 2014년 KAIST 전산학과 (공학박사)
- 2014년-2018년 한국과학기술정보연구원 선임연구원
- 2018년-현재 한국기술교육대학교 조교수
- ORCID : https://orcid.org/0000-0002-9075-3983
- 주관심분야 : 고성능 컴퓨팅, 컴퓨터 생성 홀로그래피, 로보틱스, 그래픽스/가시화, 딥러닝