V-PCC 부호화기를 위한 고속 결합 법선 추정 방법

김용환, 김유라
한국전자기술연구원 지능형영상처리연구센터
yonghwan@keti.re.kr, keyura@keti.re.kr

Fast Joint Normal Estimation Method for V-PCC Encoder

Yong-Hwan Kim, Yura Kim
Intelligent Image Processing Research Center, Korea Electronics Technology Institute

요 약

최근 들어 세계적으로 크게 관심을 받는 메타버스 및 몰입형(가상현실, 확장현실, 및 라이트필드) 콘텐츠 서비스의 응용 범위를 확대하기 위해서는 3D 객체의 실시간 전송을 위한 압축 기술이 필요하다. ISO/IEC 23090 MPEG-I Part 5 로 2021 년 표준화 완료된 V-PCC (Video-based Point Cloud Compression)는 이러한 산업계의 관심 및 필요에 의해서 국제 표준화된 동적 3D 포인트 클라우드 객체 부호화 기술이다. V-PCC 기술의 압축 성능은 기존 산업계 기술에 비해 매우 우수하나, 부호화기의 연산 복잡도가 매우 높다는 단점을 가지고 있다. 본 논문에서는 V-PCC 부호화기에서 가장 높은 연산 복잡도를 갖는 법선 추정 알고리즘의 결합 고속화 기법을 제안한다. 법선 추정은 2 개의 알고리즘으로 구성되어 있다. 첫번째는 “방향을 무시하는 법선 추정 알고리즘(normal estimation)”이고, 두번째는 첫번째 알고리즘에서 추정된 법선들을 대상으로 하는 “법선 방향 추정 알고리즘(normal orientation)”이다. 본 논문에서 제안하는 고속화 기법은 2 개 알고리즘을 결합하여 첫번째 법선 추정 알고리즘에서 획득한 부가 정보를 두번째 법선 방향 추정 알고리즘에서 활용함으로써 연산량을 대폭 줄이고, 또한 법선 방향 추정 알고리즘 내의 우선순위 큐 자료구조를 변경하여 추가적인 고속화를 달성한다. 7 개 테스트 영상에 대한 실험 결과, 압축 효율 저하 없이 법선 방향 추정 알고리즘의 속도를 평균 89.2% 향상시킬 수 있다.

1. 서론

최근 들어 메타버스 서비스 및 몰입형(가상현실, 확장현실, 및 라이트필드) 콘텐츠에 대한 관심이 커지면서 관련 연구개발이 활발히 진행되고 있다. 이러한 메타버스 및 확장현실(AR; Augmented Reality) 서비스의 응용 범위를 확대하기 위해서는 동적 3D 객체의 실시간 전송을 위한 객체 압축 기술이 필수적이나, 기존에 존재하는 동적 3D 객체 압축 기술은 압축율이 낮아서 공용망에서의 실시간 전송에 거의 사용할 수가 없었다. 그런 이유로 기존의 동적 3D 객체 기반 몰입형 콘텐츠는 실시간 전송이 아닌 다운로드 방식으로만 서비스되었다. 2021 년 표준화 완료된 ISO/IEC 23090 MPEG-I Part 5 V-PCC (Video-based Point Cloud Compression)는 이러한 산업계의 관심 및 필요에 의해서 국제 표준화된 동적 3D 포인트 클라우드 객체 부호화 기술이다[1]. V-PCC 부호화기의 압축 성능은 기존 산업계 기술(Draco[2], CWI-PCL[3, 4])에 비해 매우 우수하여 실시간 전송이 가능한 압축율을 가지고 있으나, 연산 복잡도가 매우 높다는 단점을 가지고 있다[5, 6]. V-PCC 기술은 동적 3D 포인트 클라우드 객체의 한 프레임을 각 포인트의 법선과 객체 바운딩 박스(bounding box)를 기준으로 여러 개의 3D 패치로 나눈 후 6~18 면체 투영면에 직각 투영(orthographic projection) 하여 2D 패치를 생성한다. 여기서 생성된 2D 패치들을 최소 3 개

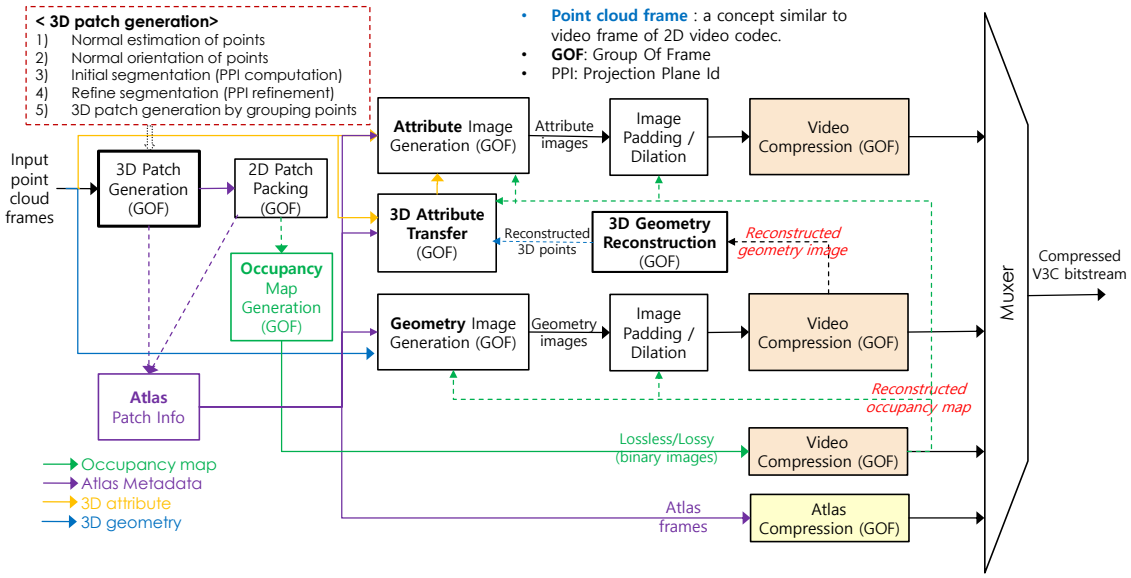

그림 1. V-PCC 부호화기 블록도

이상의 2D 이미지들(점유 맵 (occupancy map), 기하 맵 (geometry map), 속성 맵 (attribute map))에 각각 최적 배치한 후에 3 개의 2D 비디오 부호화기(HEVC[7], VVC[8], AV1[9] 등)을 이용하여 각각 압축하는 방식이다. 여기에서 사용할 수 있는 2D 비디오 코덱의 종류는 사실상 제한이 없다[6]. 그리고 3D 패치와 2D 패치의 크기, 좌표, 방향 등을 기술하는 아틀라스(atlas) 프레임(메타데이터)도 2D 이미지들과 별개로 압축된 후 먹싱(muxing)된다. 그림 1 에 V-PCC 부호화기 블록도를 보인다.

V-PCC 표준 참조 SW 인 TMC2 (Test Model Category 2)[10]에 채택된 포인트 클라우드 법선 추정 알고리즘은 객체의 날카로운 경계면 주변에서의 법선 추정 정교함은 다소 떨어지지만 연산 복잡도가 상대적으로 낮다고 알려져 있는 "주성분분석 (PCA: Principle Component Analysis) 기반 법선 추정"과 "최소생성트리 (MST: Minimum Spanning Tree) 기반 법선 방향 추정" 방법이다[11]. 3D 포인트 클라우드 법선 추정 알고리즘의 종류는 다양하게 존재하지만, 본 논문에서는 연산량이 상대적으로 낮은 TMC2 채택 알고리즘만을 대상으로 하는 고속 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2 절에서는 3D 포인트 클라우드 법선 추정 알고리즘의 개요와 복잡도를 분석하고, 3 절에서는 고속화 기법을 제안하다. 4 절에서는 제안한 기법의 성능을 실험을 통해서 확인하고, 마지막 5 절에서는 본 논문에 대한 결론을 맺는다.

2. 3D 포인트 클라우드 법선 추정 복잡도 분석

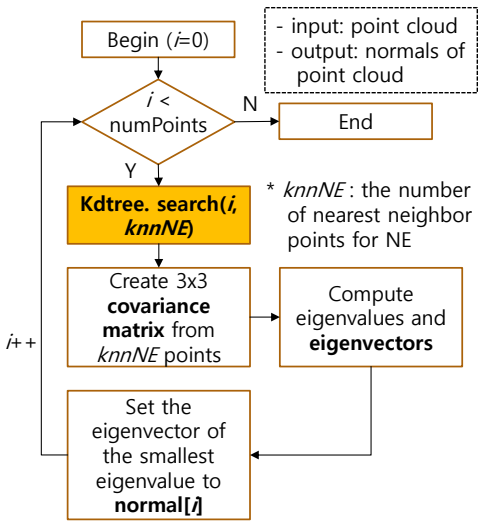

그림 2. PCA 기반 법선 추정 알고리즘

그림 2 에 PCA 기반 법선 추정 알고리즘을 보이는데, 야코비 변환(Jacobi transformation)으로 구현된 고유벡터 계산과 kd-tree 검색이 가장 큰 복잡도를 갖는다. 표 1 에 765,821 포인트를 가진 longdress 객체 첫번째 프레임에서 *knnNE*=16 일때의 법선 추정 알고리즘 복잡도를 보인다. 분석 도구는 Intel VTune Performance Analyzer 를 사용하였다. 표 1 을 보면, 방향을 무시하는 법선 추정 시간만으로도 한 프레임을 처리하는데 2 초 이상 소요됨을 알 수 있다.

<표 1. PCA 기반 법선 추정 알고리즘의 복잡도 분석>

모듈	Self Time [sec]	Ratio [%]
Kd-tree 검색	1.075	50.37
고유벡터 계산(야코비 변환)	0.791	37.07
공분산 행렬 생성	0.164	7.69
Kd-tree 자료구조 생성	0.104	4.87

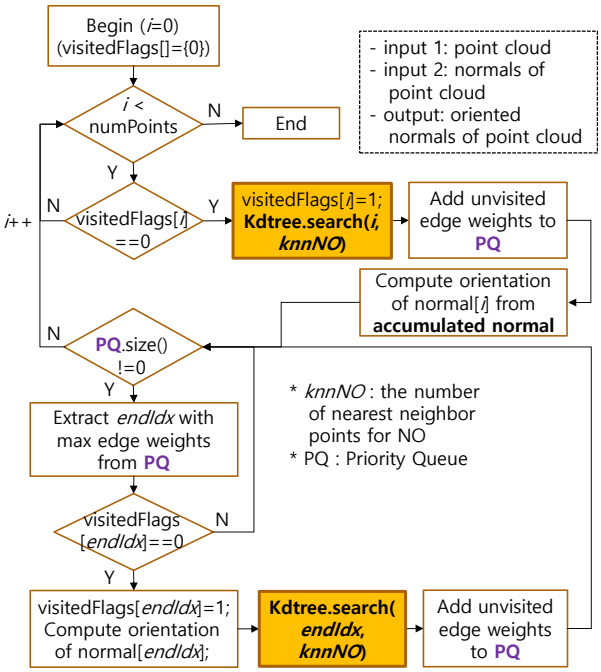

그림 3. MST 기반 법선 방향 추정 알고리즘

그림 3 에 보이는 MST 기반 법선 방향 추정 알고리즘에서 에지 가중치(edge weight)는 연결되는 두 점의 법선 벡터 내적의 절대값으로 정의되고, 연결되어 있는 두 점의 법선 벡터 내적 값이 음수이면 법선 벡터의 방향을 역전시킨다. longdress 객체 첫번째 프레임에서 *knnNO*=16 일때의 복잡도를 표 2 에 보인다. 법선 방향 추정 알고리즘은 '방향을 알 수 없는 법선들'을 가지고 법선들의 방향만을 추정하는데, 법선 추정 알고리즘보다 약 1.5 배 정도의 복잡도를 갖는다. 이러한 높은 복잡도의 첫번째 이유는 MST 생성을 위해 필수적으로 이용되는 우선순위 큐(PQ: Priority Queue)에 저장되는 데이터의 최대 개수가 약 3 백만개로 너무 많아서 데이터 가져오기(pop) 과정에서 CPU 캐쉬 효율이 크게 떨어지기 때문이다. 두번째 이유인 kd-tree 검색은 법선 추정 알고리즘과 유사하게 복잡도가 높다. 따라서 법선 방향 추정 알고리즘의 고속화를 위해서는 kd-tree 검색 뿐만 아니라 MST 생성 최적화도 필수적으로 필요하다.

<표 2. MST 기반 법선 방향 추정 알고리즘의 복잡도 분석>

모듈	Self Time [sec]	Ratio [%]
Kd-tree 검색	1.178	34.19
MST 생성 (PQ 동작: push(), pop())	1.969	57.16
방향 계산(벡터 내적) 등	0.298	8.65

이렇게 법선 추정을 2 개의 알고리즘으로 나누어서 수행하는 이유는 방향을 무시하는 법선 추정 알고리듬은 개별 포인트에 대한 법선만을 구하지만, 법선의 방향은 객체 표면 모양에 영향을 받는 전역 속성을 갖고 있기 때문에, 개별 포인트에서 방향을 추정할 수 없기 때문이다[11]. 따라서, 모든 개별 포인트에서 방향을 무시한 법선 추정을 완료한 후에, 전역 속성을 보면서 법선 방향만을 추정하는 별도의 알고리즘이 필요한 것이다.

3. 제안하는 고속 결합 법선 추정 기법

TMC2 부호화기의 MST 기반 법선 방향 추정 알고리즘에서 사용하는 우선순위 큐인 C++ 표준 라이브러리의 std::priority_queue<T>는 이진 힙(binary heap) 자료구조로 구현되어 있다. 이진 힙 자료구조는 적은 데이터에서 사용하기엔 별 무리가 없지만, 대용량 데이터에서는 CPU 캐쉬 효율이 떨어져서 처리 속도가 저하된다[12]. 따라서, 첫번째 고속화 기법으로서, 이진 힙 자료구조를 대용량 데이터 처리에 더 적합한 4 진 힙 (quaternary heap)으로 변경하면 자료구조 내 tree 높이가 1/2 로 줄어들면서 CPU 캐쉬 효율 향상에 의해 처리 속도가 증가된다. 결과적으로, 단순히 우선순위 큐 자료구조를 4 진 힙으로 변경하면 법선 방향 추정 알고리즘 전체 속도를 평균 13% 향상시킬 수 있다.

두번째 고속화 기법은 그림 2 에 보인 법선 추정 알고리즘의 kd-tree 검색 결과(즉, 현재 포인트와 가장 가까운 *knn* 개의 포인트들의 인덱스)를 별도의 인덱스 버퍼(IdxBuffer[*i*][*knn*])에 모두 저장 후에, 그림 3 의 법선 방향 추정 알고리즘에서 kd-tree 검색 대신 인덱스 버퍼(IdxBuffer)에 미리 저장된 인덱스 데이터를 그대로 이용하는 결합 기법이다. 구체적으로, 그림 2 의 kd-tree 검색식은 kdtree.search(*i*, *knn*)으로 변경된다. 여기에서 *knn* = max(*knnNE*, *knnNO*)이다. 이러한 결합 방식에 의해서 법선 방향 추정 알고리즘 전체 속도는 대폭 향상되지만, 추가적인 인덱스 버퍼에 의해 메모리 사용량이 증가하는 약간의 단점이 생긴다. 예를 들어, *knn*=16 이고 1,000,000 포인트 프레임의 경우, 인덱스 버퍼의 크기는 16 * 1,000,000 * sizeof(uint32_t) = 64 MB 이다.

4. 실험 결과

제안하는 고속 알고리즘의 성능 분석 및 실험은 다른 복잡한 요인에 의한 영향력을 배제하기 위해서 TMC2 소스에서 법선 추정 알고리즘만을 별도로 추출하여 별개의 SW 로 구성 후 수행하였다. 실험 장비는 AMD Ryzen 9 5900X@3.7GHz (70MB Cache) CPU 및 Windows 10 x64 탑재 PC 이고, 실험 영상은 V-PCC 표준화에서 사용했던 7 개의 테스트 영상 각각의 첫번째 프레임이다. 표 3 에 제안하는 알고리즘에서 우선순위 큐 자료구조 변경에 의한 법선 방향 추정(NO) 알고리즘 속도 비교

결과를 보인다. Basketball player 와 Dancer 영상의 경우처럼 포인트 수가 많아질수록 힙 자료구조의 캐쉬 효율에 영향을 많이 받으므로 속도 향상율이 커진다.

<표 3. 제안한 우선순위 큐 자료구조 변경에 의한 MST 기반 법선 방향 추정 알고리즘의 속도 비교 (*knnNO* = 16)>

		NO 시간 [msec]		속도 향상 비율 [%]
영상	포인트 개수	이진 힙	4 진 힙	
Basketball player	2,880,057	11,342	9,432	20.25
Dancer	2,592,758	10,311	8,315	24.00
Longdress	765,821	2,197	1,978	11.07
Loot	784,142	2,241	2,040	9.85
Queen	1,006,509	2,849	2,585	10.21
Redandblack	728,133	2,022	1,879	7.61
Soldier	1,059,810	3,210	2,923	9.82
평균				13.26

<표 4. 제안한 우선순위 큐 자료구조 변경 및 결합 알고리즘에 의한 MST 기반 법선 방향 추정 알고리즘의 통합 속도 비교 (*knn* = *knnNE* = *knnNO* = 16)>

		NO 시간 [msec]		속도 향상 비율 [%]
영상	포인트 개수	이진 힙	4 진 힙 + 결합 알고리즘	
Basketball player	2,880,057	11,836	6,176	91.65
Dancer	2,592,758	10,112	5,412	86.84
Longdress	765,821	2,180	1,170	86.32
Loot	784,142	2,221	1,175	89.02
Queen	1,006,509	2,856	1,487	92.06
Redandblack	728,133	2,019	1,051	92.10
Soldier	1,059,810	3,280	1,759	86.47
평균				89.21

표 4 에 제안한 우선순위 큐 자료구조 변경 및 결합 법선 추정 알고리즘에 의한 법선 방향 추정 알고리즘의 통합 속도 비교 결과를 보인다. 제안한 우선순위 큐 자료구조 변경에 의해 우선순위 큐의 데이터 처리 속도가 향상되고, 결합 알고리즘에 의해 법선 방향 추정 알고리즘에서 kd-tree 검색을 모두 생략함으로써 통합 평균 89.2%의 속도 개선 결과를 보인다. 참고로, 본 논문에서 제안한 2 개의 기법 모두 원본 알고리즘과 결과값이 100% 일치하므로 화질 열화는 전혀 발생하지 않는다.

5. 결론

본 논문에서는 최신 V-PCC 코덱 참조 SW 인 TMC2 부호화기에서 가장 복잡도가 높은 모듈 중의 하나인 3D 포인트 클라우드 법선 추정 알고리즘의 고속화 기법을 제안하였다. 제안한 기법은 화질 열화 없이 법선 방향 추정 알고리즘의 속도를 평균 89.2% 향상시켰다. 또한 제안한 방법은 간단히 구현할 수 있다는 장점도 가지고 있다. 향후에는 CPU 벡터 연산 (SIMD: Single Instruction Multiple Data)을 이용하여 공분산 행렬 생성, 야코비 변환, 및 법선 방향을 판단하기 위한 벡터 내적 연산 등의 속도 최적화를 통해 법선 추정 및 법선 방향 추정 알고리즘의 추가 고속화를 진행할 예정이다.

ACKNOWLEDGEMENT

이 논문은 2022 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No.2022-0-00981. 전배경 정합 3D 객체 스트리밍 기술개발)

[참고 문헌]

[1] MPEG 3DG, Draft Text of ISO/IEC FDIS 23090-5 V3C and V-PCC 2nd edition, ISO/IEC JTC1/SC29/WG7, Doc. M59216, Jan. 2022.

[2] Google Draco. https://google.github.io/draco/

[3] Mekuria, R.N, Blom, C.L, & César Garcia, P.S., "Design, implementation and evaluation of a point cloud codec for Tele-Immersive Video," *IEEE Trans. On CSVT*, vol. 27 no. 4, pp828-842, April 2017.

[4] CWI-PCL. https://github.com/cwi-dis/cwi-pcl-codec/

[5] 박종근, 김용환, "3 차원 포인트 클라우드 코덱 성능 평가", *한국인터넷정보학회 춘계학술발표대회 논문집*, 제 22 권 1 호, 2021 년 4 월.

[6] T. Dong, K. Kim, and E. S. Jang, "Performance evaluation of the codec agnostic approach in MPEG-I video-based point cloud compression," *IEEE Access*, Vol. 9, Dec. 2021.

[7] ISO/IEC 23008-2 & ITU-T Rec. H.265, "High Efficiency Video Coding," Feb. 2018.

[8] ISO/IEC 23090-3 & ITU-T Rec. H.266, "Versatile Video Coding," Aug. 2020.

[9] The Alliance for Open Media, "AV1 Bitstream & Decoding Process Specification", Version 1.0.0 with Errata 1, Jan. 2019.

[10] MPEG Point Cloud Compression Category 2 Reference Software. [Online]. Available: http://mpegx.int-evry.fr/software/MPEG/PCC/TM/mpeg-pcc-tmc2.git

[11] H. Hoppe, et al., "Surface reconstruction from unorganized points," in *Proc. SIGGRAPH*, pp. 71-78, July 1992.

[12] Malte Skarupke, "On Modern Hardware the Min-Max Heap beats a Binary Heap," 2020.08, [online]. Available: https://probablydance.com/2020/08/31/on-modern-hardware-the-min-max-heap-beats-a-binary-heap/