DRACO 3D 메시 압축 기술 소개

□ **변주형, 심동규** / 광운대학교

요약

구글에서 개발한 DRACO는 3D 메시 및 포인트 클라우드의 3차원 그래픽 데이터를 압축하는 오픈소스 라이브러리로 DRACO의 정적 메시 압축 기술은 계산 복잡도 및 압축 성능 관점에서 효과적으로 알려져 있다. DRACO는 개방형 표준화 단체인 크로노스 그룹의 3차원 모델을 표현하는 공개 표준 파일 포맷인 glTF 표준에 채택되었으며 국제 표준화 단체인 MPEG 3DGH(3D Graphics and Haptics)에서 개발 중인 동적 메시 압축 표준에 채택되는 등 활용도가 높아질 것으로 예상된다. 본고는 DRACO의 메시 데이터 압축 기술에 대해 소개하여 관련 분야를 연구하는 사람들에게 도움이 되고자 한다.

Ⅰ. 서 론

3차원 메시 모델은 2차원 영상 및 그래픽스 대비 뛰어난 입체감을 제공함으로써 현실 세계를 보다 더 사실적으로 표현할 수 있다. 이러한 3차원 메시 데이터에 대한 처리 및 압축 기술은 1900년대 후반부터 활발히 연구가 진행되었지만, 일부 항공, CAD, 애니메이션 등의 전문가의 영역에서 한정적으로 사용되었다. 휴대폰, 태블릿 등의 모바일 디바이스가 대중화됨에 따라 2D 영상 처리 및 압축 기술이 급격히 발전했던 것과 같이 최근 글로벌 기업들이 앞다투어 AR/VR 산업에 뛰어들며 3차원 모델 데이터에 대한 취득 및 디스플레이 기기들이 대중화됨에 따라 3차원 데이터에 대한 전송 및 저장에 대한 수요 역시 크게 늘 것으로 예상된다. 이에 따라 산업의 수요를 반영하고자 국제 표준화 기구 ISO/IEC JTC1/SC29/WG 7 산하의 MPEG 3DGH 그룹에서는 시간에 따라 변화하는 3D 메시 데이터에 대한 국제 표준화를 시작하는 등 최근 다시 3D 그래픽 데이터 압축 기술에 대한 국제적인 연구가 활

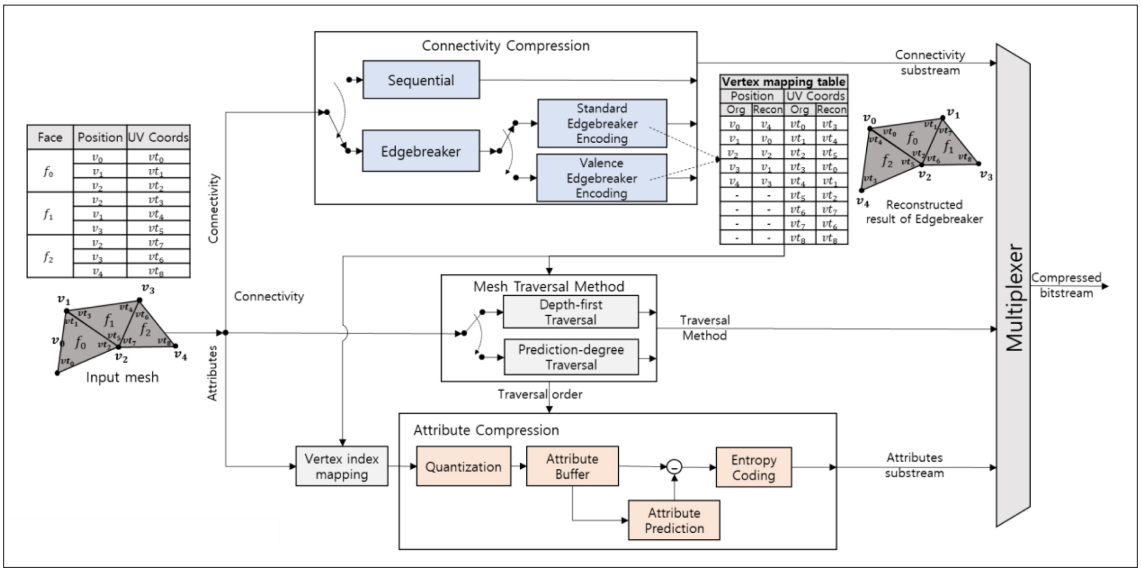
발히 이루어지고 있다. DRACO[1]는 하나의 메시 프레임 내의 공간적/확률적 중복성을 제거함으로써 압축을 수행하는 정적 메시 압축 기술로써 압축률과 더불어 부/복호화 계산 복잡도를 고려하여 설계되어 효과적으로 3D 메시 데이터를 압축하는 기술이다. DRACO의 우수한 메시 압축 성능으로 최근 시작된 MPEG의 동적 메시 압축 표준인 V-DMC(Video-based Dynamic Mesh Coding)[2]의 참조 소프트웨어에서 정적 메시 압축 기술로 DRACO가 채택되어 사용되고 있다.

메시 데이터는 3차원 공간상의 각 정점들의 좌푯값인 기하 정보와 각 정점들이 하나의 표면 정보를 구성하기 위한 연결 관계를 기술하는 연결 정보, 정점 및 표면에 대한 색상, 노말값 등의 속성 정보로 구성된다. DRACO의 메시 압축 기술은 1900년대 후반부터 활발하게 연구된 single-rate 메시 압축 기술[3]들을 토대로 개발되었으며, 특히 1999년 처음 제안된 Edgebreaker[4] 알고리즘을 기본 골조로 메시 데이터의 연결 정보를 압축하고 복원된 연결 정보를 기반으로 기하 정보 및 속성 정보에 대해 압축하는 구조로 이루어져 있다. 기하 정보 및 속성 정보의 경우 예측을 통해 공간적인 중복성을 제거 후 잔차 신호를 엔트로피 부호화를 통해 확률적인 중복성을 제거 후 전송하는 전통적인 압축 방식을 통해 메시 데이터 압축이 수행된다. 본고에서는 DRACO 메시 압축 기술의 전반적인 부/복호화 구조와 데이터 구성 요소별 세부 압축 기술에 대해 소개한다.

본고의 구성은 다음과 같다. 2장에서는 DRACO 기술의 전반적인 부/복호화 구조에 대해 살펴본다. 3장에서는 DRACO의 압축 대상인 연결 정보와 속성 정보에 대한 압축 기술 및 DRACO에서 사용하고 있는 rANS(range Asymmetric Numeral System)[5] 기반의 엔트로피 부호화 기술에 대해 설명하고 4장에서 결론을 맺는다.

Ⅱ. DRACO 메시 부/복호화 구조

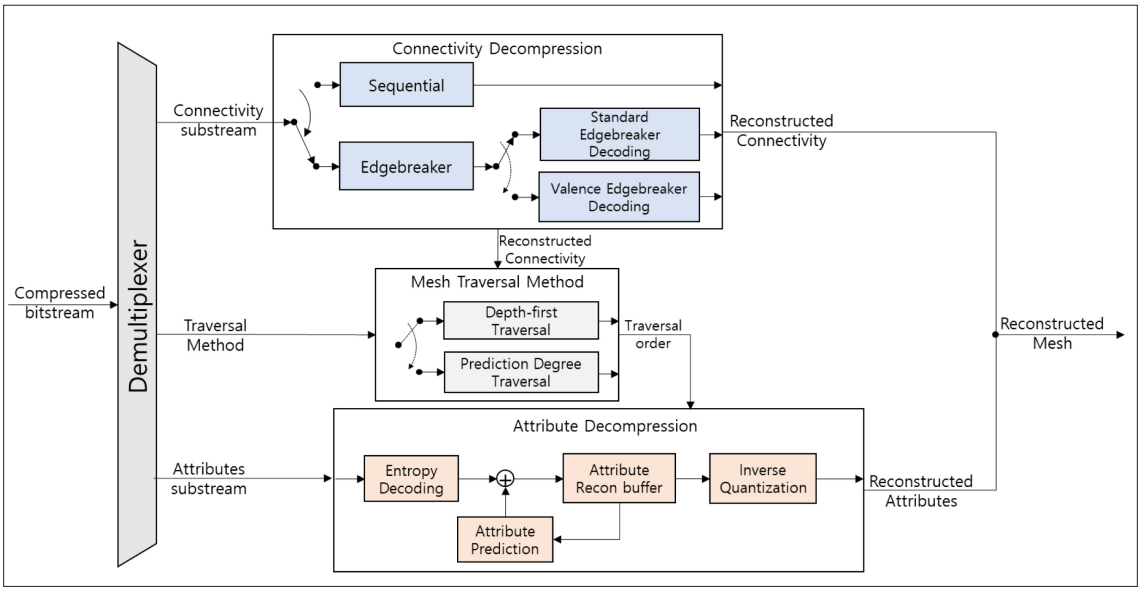
본 절에서는 DRACO의 메시 압축 대상 및 부/복호화 구조에 대해 살펴본다. DRACO 메시 부호화기는 기본적으로 삼각형 메시 데이터에 대한 압축을 지원하며 쿼드



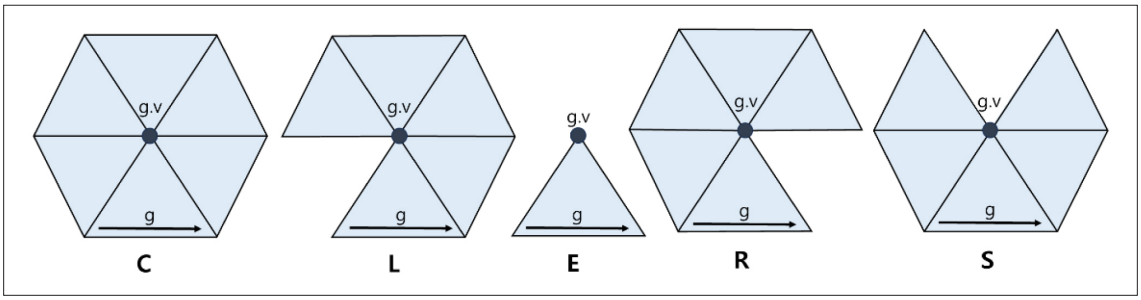
<그림 1> DRACO 메시 부호화 구조도

(quad) 메시의 경우 하나의 사각형을 두 개의 삼각형으로 나누어 삼각형 메시로 변환하는 과정을 통해 간접적으로 부호화를 지원한다. DRACO는 삼각형 메시의 연결 정보와 각 정점들의 위치를 정보 및 각 삼각형의 정점 또는 코너마다 정의될 수 있는 노말, 색상, 텍스처 좌표 등의 정보에 대한 압축을 지원한다. 이때 텍스처 좌표의 경우 메시 표면의 텍스처 정보를 2D 이미지로 저장하고, 렌더링 시 2D 이미지상의 텍스처를 와핑(wrapping)하여 사용하기 위해 2D 이미지와 3D 폴리곤 평면과의 매핑 정보를 저장하는 2차원 좌푯값을 의미한다. 이때 2D 이미지 형태로 저장된 텍스처 맵에 대한 압축은 DRACO에서 지원하지 않는다. 〈그림 1〉은 DRACO 메시 부호화기의 전체 구조도를 나타낸다.

DRACO에서는 연결 정보를 제외한 정점 또는 코너마다 정의되는 데이터를 속성 정보라고 정의하며, 메시 데이터의 연결 정보와 속성 정보를 코너 테이블(corner-table) [6] 구조를 기반으로 관리 및 압축한다. DRACO는 각 데이터별로 다수 개의 부호화 기술 옵션이 존재하고 이는 소프트웨어에서 입력 파라미터인 압축 레벨(compression level, cl)값에 따라 각 부호화 기술이 결정된다. 메시 데이터 중 연결 정보가 순차(sequential) 부호화 또는 Edgebreaker 기반 기술을 통해 부호화가 수행되며 이때 Edgebreaker 기반 기술로 부호화되는 경우 코너 테이블의 인덱스들이 부호화 과정에서 변경되게 된다. 연결 정보에 대한 부호화가 끝난 뒤, 속성 정보에 대한 부호화가 수행되며 속성 정보가 부호화되는 순서는 연결 정보를 기반으로 깊이 우선(depth-first) 또는 추후 소개될 다중 평행 사변형 예측의 예측 차수를 고려하여 부호화 순서를 결정하는 Prediction degree 순서[7]로 부호화가 수행된다. 각각의 속성 정보는 입력 파라미터에 따라 결정되는 비트 깊이에 따라 양자화가 수행되며 양자화된 속성 정보에 대해 예측을 수행하여 최종적으로 원본 신호에서 예측 신호를 제거한 잔차 신호를 부호화한다. 이때 Edgebreaker로 부호화가 수행되어 복호화 후 코너 테이블의 인덱스가 변경되는 경우 정점 매핑 테이블을 통해 복호화 후 변경될 코너 테이블 인덱스로 속성 정보의 인덱스값을 변환 후 부호화를 수행하게 된다. 이를 통해 복호화기에서 복원된 연결 정보와 속성 정보의 인덱스를 매핑할 수 있다.



<그림 2> DRACO 메시 복호화기 구조도



<그림 3> Edgebreaker 심볼 종류

〈그림 2〉는 DRACO의 메시 복호화기 구조도로 부호화기와 동일하게 연결 정보부터 복원을 수행 후 복원된 연결 정보를 기반으로 속성 정보 복호화를 수행한다. 이때 부호화 과정에서 속성 정보에 대한 코너 테이블 인덱스를 복원 연결 정보를 기준으로 변환하였으므로 복호화 과정에서는 별도의 인덱스 매핑 없이 복원이 가능하다.

Ⅲ. DRACO 메시 압축 기술

본 절에서는 DRACO의 메시 컴포넌트별 세부 압축 기술에 대해 소개한다.

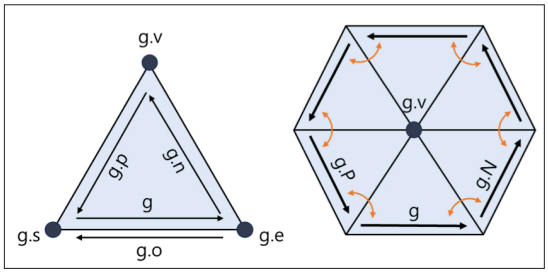
1. 연결 정보 압축 기술

DRACO의 연결 정보 압축 기술로는 크게 순차 부호화 방법과 Edgebreaker 기반의 압축 기술로 나눌 수 있다. 삼각형 메시 데이터의 연결 정보는 하나의 삼각형을 구성하고 있는 정점 또는 코너 속성 정보의 인덱스로 구성될 수 있으며 순차 부호화는 해당 인덱스들을 입력 메시 데이터에 저장되어 있는 삼각형 표면(face) 순서대로 연결정보를 부호화한다. 각 삼각형을 구성하는 인덱스를 각각 n-bit(n은 8, 16 또는 32-bit 등)으로 부호화하거나 이전 인덱스와의 차분 신호를 엔트로피 부호화를 수행한다. 순차 부호화 방법은 Edgebreaker 방법에 비해 압축 성능이 떨어지지만 매우 간단한 방법으로 부/복호화 속도가 빠르고 압축 전후의 코너 테이블 인덱스가 바뀌지 않는다.

Edgebreaker 기반의 연결 정보 압축 기술로는 코너 테이블 기반의 일반 Edgebreaker 부호화 방법[6]과 일반 Edgebreaker 방법에 정점의 valence에 따른 엔트로피 확률 정보를 사용하여 부/복호화를 수행하는 valence Edgebreaker[8]가 있다.

Edgebreaker는 임의의 엣지로부터 시작하여 현재 삼각형과 주변 삼각형들의 연결 관계에 따라 정해진 규칙 기반으로 순회하며 하나의 삼각형을 하나의 심볼로 매핑함으로써 삼각형 메시의 연결 정보 부호화를 수행한다. 이때 부호화되는 심볼은 {C, L, E, R, S}로 5개 중 하나의 심볼로 매핑하게 되며 이는 〈그림 3〉과 같다.

일반적인 manifold 메시 데이터의 경우 5개의 심볼 중 대부분의 삼각형이 C 또는 R 심볼로 부호화되기 때문에 심볼을 엔트로피 코딩 시 삼각형당 약 1.5~2비트로 표현



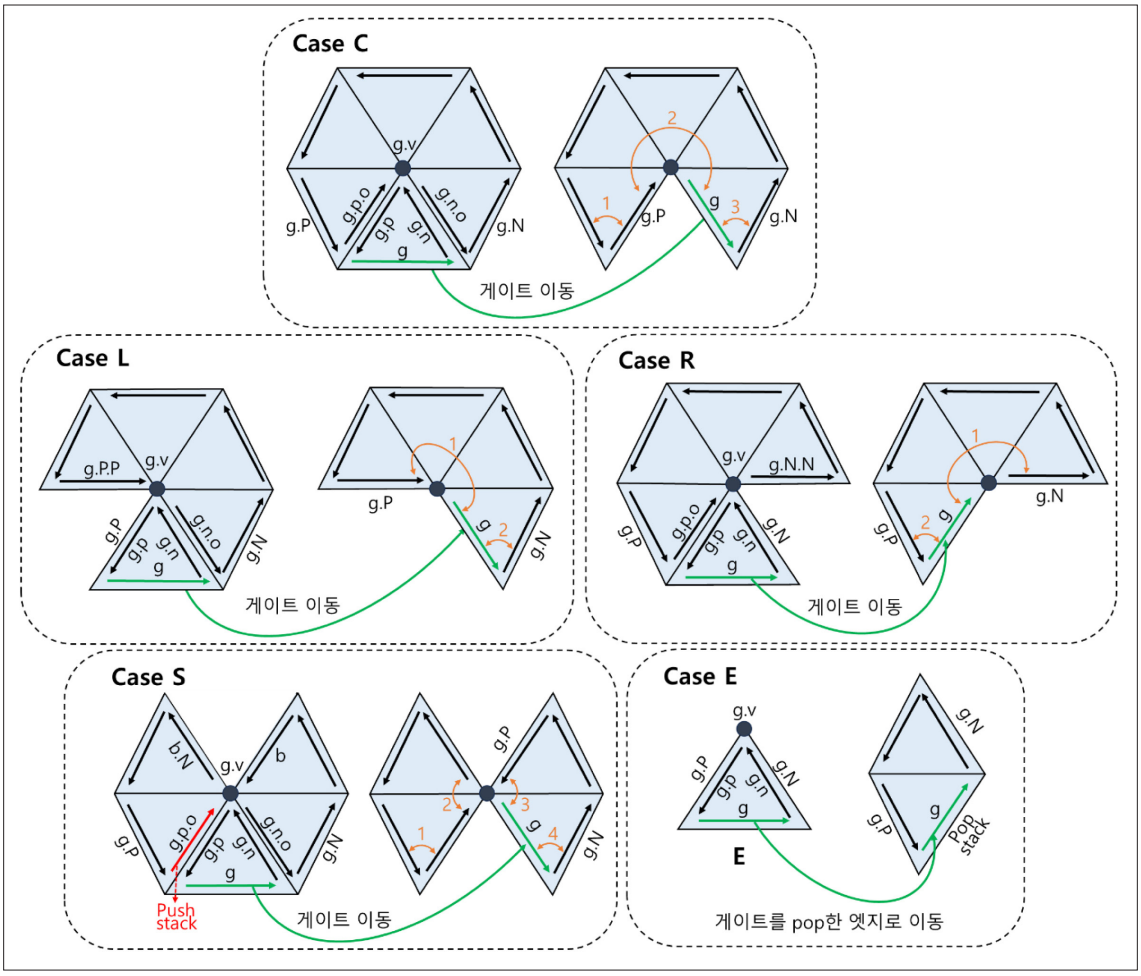
<그림 4> Half-edge 데이터 구조

<알고리즘 1> Edgebreaker 심볼 결정 규칙

```
if (g.v ∉ B) case C
else {
        if (g.p == g.P){
          if (g.n == g.N) case E
          else case L
        }
        else {
          if (g.n == g.N) case R
          else case S
        }
}
```

가능한 것으로 알려져 있다[4]. Edgebreaker의 5개의 심볼은 〈그림 4〉의 half-edge 데이터 구조[9]를 기반으로 〈알고리즘 1〉의 과정을 통해 결정된다.

g는 게이트 엣지로 현재 삼각형 중 엣지 g에 포함되지 않는 정점을 $g.v$라고 했을 때 g와 $g.v$를 통해 구성되는 삼각형과 주변 삼각형과의 연결 관계에 따라 심볼을 매핑한다. 〈알고리즘 1〉의 규칙에 따라 결정되는 심볼을 정리하면 정점 $g.v$가 바운더리(B) 위에 존재하는 경우 심볼 C로 코딩되며 나머지 4개의 심볼들은 $g.v$가 바운더리 위에 존재하지 않으며 추가적으로 다음 조건을 만족할 때 각 심볼로 코딩이 수행된다. 인접한 삼각형이 없는 경우 심볼 E,



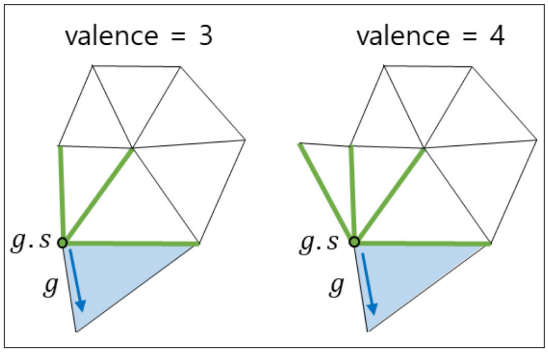
<그림 5> Edgebreaker 심볼별 동작

오른쪽에 인접한 삼각형만 존재하는 경우 심볼 L, 왼쪽에 인접한 삼각형만 존재하는 경우 심볼 R, 양쪽에 인접한 삼각형이 존재하는 경우 심볼 S. 각 심볼로 코딩됨에 따른 심볼별 동작은 〈그림 5〉와 같다.

현재 게이트 엣지(g)에서 주변 삼각형과의 관계를 통해 현재 삼각형의 심볼을 코딩한 후 현재 게이트 엣지를 제거하고 심볼에 따라 다음 게이트의 위치가 결정된다. 이때 심볼마다 정의된 규칙에 따라 순차적으로 엣지들 간의 연결 구조가 변경된다. 심볼 S의 경우 게이트를 이동하기 전에 스택(stack)에 $g.p.o$ 엣지를 삽입 후 게이트를 이동하며 심볼 E의 경우 주변의 삼각형이 존재하지 않을 때 선택되는 심볼로 스택을 확인하여 스택에 저장된 엣지가 존재하는 경우 팝(pop)하여 해당 엣지를 게이트로 해서 코딩이 계속되며, 스택이 비어 있다면 모든 삼각형을 코딩한 것으로 코딩이 종료된다. Edgebreaker의 디코더는 심볼의 히스토리(H)를 전송받아 연결 정보를 복원한다. 이때 기존 Edgebreaker의 경우 모든 심볼을 파싱하고, 각각의 심볼을 파싱받을 때마다 전처리 과정[4]을 통해 스플릿 심볼 S가 복호화되었을 때 현재 엣지와 연결할 정점에 대한 오프셋값을 계산해 주어야 한다. 따라서 복호화 계산 복잡도가 증가하며 심볼을 받는 즉시 삼각형을 복원할 수 없고, 모든 심볼을 파싱 후 복원을 시작할 수 있다는 단점이 존재한다. 이러한 단점을 해결하기 위해 심볼을 역순으로 복원하는 방법[10]이 제안되었으며 역순 복원의 경우 인코딩된 순서의 역순으로 심볼에 대해 복원 시 추가적인 전처리 과정 없이 규칙 기반으로 삼각형을 복원할 수 있으며 인코더에서 코딩된 심볼 히스토리를 역순으로 바꿔 전송 시 디코더에서 심볼을 복원한 즉시 삼각형을 복원할 수 있다. DRACO는 일반 Edgebreaker와 valence Edgebreaker 모두 역순 복원을 채택하여 사용하고 있다.

Valence Edgebreaker는 특정 정점에 대해 해당 정점을 공유하고 있는 삼각형의 개수에 따라 심볼의 발생 확률이 다른 특성을 이용하여 심볼의 엔트로피 부호화 효율을 높이는 기술이다.

〈그림 6〉은 게이트 엣지의 시작 정점에 대해 각각 valence가 3, 4인 경우를 나타내는 예시로 valence란 게이트 엣지를 제외한 해당 정점에 입사하는 엣지의 개수로 정의된다. DRACO의 valence Edgebreaker는 valence가 2 이상, 7 이하인 경우에 대해 각 심볼이 발생할 확률을 엔트로피 부호화를 통해 전송하고 해당 확률 테이블을 통해 심볼 히스토리를 부/복호화한다. 이때 인코더에서 계산되는 valence별 확률 테이블의 경우 역순 복원을 고려하여 복호화 시점의 $g.s$에 대한 valence 값을 계산하여 부/복호화에 사용하며 valence 값이 2보다 작은 경우 valence=2의 확률 테이블을 통해, valence 값이 7보다 큰 경우 valence=7의 확률 테이블을 통해 엔트로피 부호화의 확률값을 결정한다.



<그림 6> 게이트의 시작 정점(g.s)에 대한 valence 예시

2. 속성 정보 압축 기술

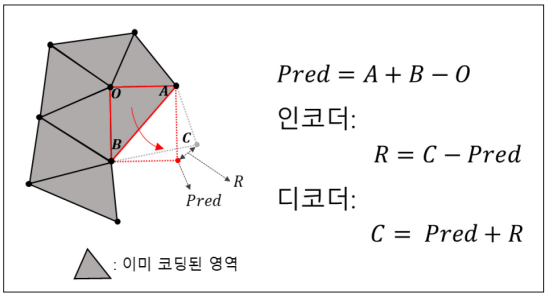
DRACO의 속성 정보 압축 기술은 주변의 이미 부/복호화된 속성 정보를 기반으로 공간적 중복성을 제거하는 예측 기술을 기반으로 수행된다. DRACO에서 사용되는 속성 정보의 예측 기술로는 차분 예측(difference prediction), 평행사변형 예측(parallelogram prediction), 다중 평행사변형 예측(multi-parallelogram prediction), 텍스처 좌표 예측, 기하학적 법선 예측 등이 있으며, 이 중 입력 속성 정보에 따라 선택적으로 하나의 예측 기술을 사용하여 부/복호화가 수행된다.

1) 차분 예측 기술

차분 예측 기술은 주변 정점의 속성 정보를 사용하여 현재 정점의 속성 정보를 예측하는 기술이다. 차분 예측 기술은 모든 속성 정보에 대해 사용될 수 있는 예측 기술로 부호화기에서는 속성 정보 부호화 순서에 따라 바로 이전에 부호화된 속성 정보를 이용하여 현재 정점의 속성 정보 간의 차분값을 계산한 후 이를 압축하고, 복호화기에서는 전송받은 차분값에 부호화기와 동일한 예측 신호를 더하여 현재 정점의 속성 정보를 복원할 수 있다.

2) 평행사변형 예측 기술

평행사변형 예측 기술은 정점 위치 정보 부호화에 효과적인 예측 기술로 〈그림 7〉과 같이 예측을 수행한다. 속성 정보를 코딩하는 시점에 연결 정보가 이미 코딩되어 현재 코딩하고자 하는 정점과 연결된 인접 정점 및 삼각형에 대한 정보를 알 수 있다. 따라서 정점 C를 예측할 때 C가 포함된 삼각형과 엣지를 공유하는 두 개의 정점(A, B)과 공유 엣지를 기준으로 반대편에 위치하는 정점 O를 사용하여 평행사변형 예측을 수행한다.

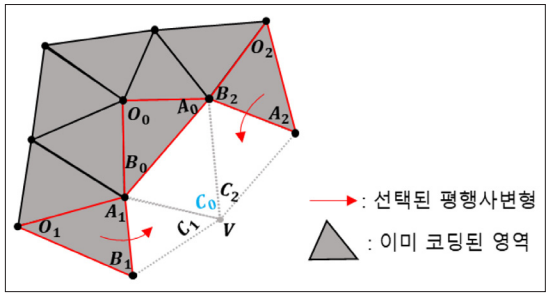


<그림 7> 평행사변형 예측

평행사변형 예측 기술은 현재 코딩하고자 하는 정점이 포함된 삼각형과 예측에 사용된 삼각형이 non-convex 또는 non-planar 정도가 클수록 예측 성능이 떨어지게 된다.

3) 다중 평행사변형 예측 기술

다중 평행사변형 여러 개의 평행사변형 중 선택적으로 하나 이상의 평행사변형 예측을 통해 생성한 예측 신호를 평균내어 최종 예측 신호를 생성한다. 다중 평행사변형 예측은 단일 평행사변형 예측을 통해 여러 평행사변형 예측 신호 중 선택적으로 평균내어 최종 예측 신호를 생성하여 단일 평행사변형의 단점을 보완할 수 있어 일반적으로 예측 성능이 우수하다. 〈그림 8〉은 다중 평행사변형 예측의 예시를 나타낸다.

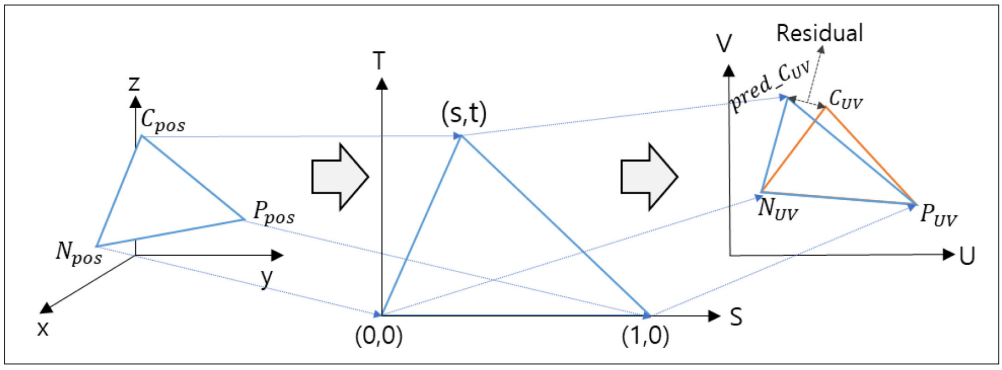


<그림 8> 다중 평행사변형 예측

예측을 수행할 때, 정점을 탐색하는 순서는 현재 코너점을 기준으로 왼쪽 방향으로 탐색한 후, 더 이상 예측에 사용 가능한 코너점이 없다면 오른쪽 방향으로 탐색하여 예측 시 사용하고자 하는 코너점을 탐색한다. 만약, 사용 가능한 코너점의 개수가 DRACO에서 정의된 최댓값인 4개가 탐색된 경우 탐색이 조기에 종료된다. 부호화기에서 최대 4개의 예측 후보 중 최적의 예측 코너를 선택하여 각 코너에 대해 예측에 사용될지 여부를 플래그로 전송한다.

4) 텍스처 좌표 예측

텍스처 좌표 예측 기술은 텍스처 좌표 예측을 위한 전용 기술로 3차원 공간상의 삼각형과 텍스처 좌표 공간에서의 삼각형이 닮음을 가정하여 텍스처 좌표를 예측하는 기술이다. 텍스처 좌표 예측은 정점의 위치 정보를 기반으로 예측이 수행되어 텍스처 좌표 복호화 시점에 위치 정보가 복원되어 있어야 한다. 따라서 DRACO에서 텍스처 좌표 예측이 수행되는 경우 정점의 위치 정보를 먼저 부/복호화한 후 텍스처 좌표의 부/복호화가 수행된다. 〈그림 9〉는



<그림 9> 텍스처 좌표 예측 개념도

텍스처 좌표 예측의 개념도를 나타낸다.

텍스처 좌표 예측은 C, N, P 위치의 위치 정보와 N, P 위치의 텍스처 좌표를 알고 있을 때, C 위치의 텍스처 좌표 C_{UV}를 예측하는 것으로 삼차원 공간에 3개의 정점 위치에 의해 정의되는 삼각형을 N을 (0,0), P를 (1,0)으로 하는 좌표축으로 변환했을 때 C의 ST 좌표축에서의 좌푯값 (s, t)는 다음 수식을 통해 유도할 수 있다.

$$s = \frac{pn \cdot cn}{|pn|}, t = \frac{|cn - pn * s|}{|pn|} \quad (1)$$

$$pn = P_{pos} - N_{pos},\ cn = C_{pos} - N_{pos} \quad (2)$$

계산된 (s, t) 좌표를 UV 좌표축으로의 변환은 이미 알고 있는 텍스처 좌표인 N_{UV}와 P_{UV}를 기반으로 강체(rigid) 어파인 행렬 M을 다음 수식을 통해 계산할 수 있다.

$$M = \begin{bmatrix} PN_{UV}[0] & -PN_{UV}[1] & N_{UV}[0] \\ PN_{UV}[1] & PN_{UV}[0] & N_{UV}[1] \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$PN_{UV} = P_{UV} - N_{UV} \quad (4)$$

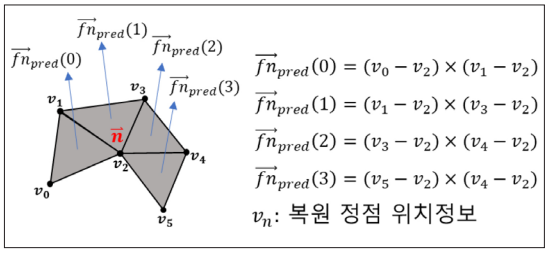
따라서 최종 예측 텍스처 좌표는 복원 기하 정보로부터 계산된 (s, t) 좌표에 어파인 행렬 M을 적용하여 계산되며 텍스처 좌표의 매개 변수화 과정에서 3차원 공간상의 삼각형이 UV공간으로 매핑될 때 PN_{UV}축을 기준으로 뒤집힐 수 있으므로 다음 식 (5)와 식 (6)을 통해 유도된 텍스처 좌표 예측값인 $Pred_C_{UV}^{0}$와 $Pred\_C_{UV}^{1}$ 중 C_{UV}와 유사한 값을 사용하여 예측을 수행한다. 이때 부호화기에서 둘 중 선택된 예측값에 대한 플래그와 선택된 예측값을 통해 생성한 차분값을 전송한다.

$$Pred\_C_{UV}^{0} = M * \begin{bmatrix} s \\ t \\ 1 \end{bmatrix} \quad (5)$$

$$Pred\_C_{UV}^{1} = M * \begin{bmatrix} s \\ -t \\ 1 \end{bmatrix} \quad (6)$$

5) 기하학적 법선 예측

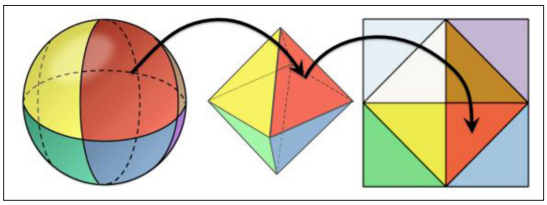
기하학적 법선 예측 기술은 복원된 위치 정보를 기반으로 법선 벡터를 예측하는 기술로 해당 기술도 위치 정보에 대한 의존성이 존재한다. 코너 또는 정점 노말의 예



<그림 10> 노말 벡터 예측 예시

측은 주변 인접한 삼각형들의 표면 노말(face normal) 벡터를 복원 위치 정보를 통해 〈그림 10〉의 예시와 같이 계산한 후 계산된 표면 노말 벡터들을 평균을 취해 예측을 수행한다.

기하학적 법선 예측 기술은 계산된 예측 노말 벡터와 입력 노말 벡터를 각각 Octahedron 노말 벡터 공간[11]으로 변환 후 잔차 신호를 계산하여 전송한다. Octahedron 노말 벡터 공간으로의 변환은 〈그림 11〉과 같이 표현할 수 있으며 3차원 노말 벡터를 L1 정규화 후 2차원 공간에 매핑시키는 과정을 통해 최종적으로 2차원의 잔차 노말 신호만 전송을 해주면 된다.



<그림 11> Octahedron 매핑 도식화[12]

디코더 측에서는 전송받은 Octahedron 잔차 노말 벡터를 역변환하여 3차원 노말 벡터로 변환 후 복원 위치 정보를 통해 예측한 3차원 노말 벡터 예측 신호와 더하여 복원을 수행할 수 있다.

3. 엔트로피 코딩 기술

1) rANS(range Asymmetric Numeral System)

DRACO에서 Edgebreaker를 통해 부호화된 심볼 및 속성 정보 예측을 통해 생성된 잔차 신호는 rANS를 통해 엔트로피 부호화가 수행된다. 비대칭 숫자 시스템(ANS, Asymmetric Numeral System) 계열의 엔트로피 부호화기는 산술 부호화(arithmetic coding)와 유사한 압축률을 제공하면서 허프만 부호화(Huffman coding)보다 빠른 복호화 속도로 구현될 수 있는 기술[5]로 최근 다양한 압축기에 채택되어 적용되고 있다. rANS의 부호화 과정은 산술 부호화와 개념적으로 유사하며 두 기술의 비교는 〈표 1〉과 같다.

rANS의 기반 기술인 비대칭 숫자 시스템은 우리가 일반적으로 사용하는 대칭 숫자 시스템(SNS, Symmetric Numeral System)의 일반화된 케이스로 10진수 대칭 숫자 시스템의 심볼 인코딩 과정은 다음 〈알고리즘 2〉로 표현할 수 있다.

<알고리즘 2> 10진수 대칭 숫자 시스템 인코딩 과정

```
encode_step_sns(x_prev, s) {
    x = x_prev*1/0.1 + s
    return x
}
```

{2, 3, 1, 9}의 심볼을 위의 인코딩 과정을 통해 부호화 시 2319를 얻을 수 있으며 이는 10개의 심볼이 0.1의 동일한 확률을 가질 때 최적으로 심볼들을 부호화할 수 있게 된다.

비대칭 숫자 시스템은 심볼들 간의 확률값이 동일하지 않은 경우에도 최적으로 동작할 수 있도록 대칭 숫자 시스템을 일반화한 기술로 비대칭 숫자 시스템의 인코딩 과정은 〈알고리즘 3〉과 같다.

<표 1> 산술 부호화와 rANS 부호화 비교

	산술 부호화	rANS 부호화
각 심볼 코딩 시	간격(interval)이 심볼의 확률값에 반비례하여 줄어듦	상태(state) 값이 심볼의 확률값에 반비례하여 늘어남
비트 발생 시점	재정규화 과정: 확률의 정밀도를 유지하기 위해 간격을 늘리기 위한 비트 발생	재정규화 과정: 상태값이 특정 값보다 커지는 것을 방지하기 위해 상태값을 줄이기 위한 비트 발생

<알고리즘 3> 비대칭 숫자 시스템 인코딩 과정

```
encode_step_ans(x_prev, s) {
   # prob[s] = freq[s]/M, M=∑_∀s freq[s]
   # cumul[s] = ∑_{i=0}^{i=s-1} freq[i]
   x = (x_prev//freq[s])*M + cumul[s] + x_prev%freq[s]
   return x
}
```

대칭 숫자 시스템의 상태값이 매 심볼 코딩 과정마다 동일한 값이 곱해졌다면 비대칭 숫자 시스템 부호화는 각 심볼의 확률의 역수만큼 곱해진다. 각 심볼의 확률값이 주어진다면 ANS 부호화는 최적의 부호화가 가능하다. ANS 부호화 과정에서 상태값은 심볼들을 코딩함에 따라 지수적으로 증가하여 일반적으로 30~40개의 심볼 부호화 시 64비트 범위를 초과하게 된다. 따라서 상태값을 컴퓨터가 처리할 수 있는 특정 범위 내로 유지하는 방법 중 하나로 rANS 부호화가 수행된다. rANS는 심볼 부호화 시 상태값을 특정 범위, [L, H]로 조정하기 위해 재정규화를 수행한다. rANS의 재정규화 과정은 〈알고리즘 4〉와 같다.

<알고리즘 4> rANS 부호화기 재정규화 과정

```
renormalization_enc(x_prev, s) {
   while encode_step(x_prev, s) not in [L, H] {
         output_bits.prepend(x_prev%2)
         x_shrunk = x_prev>>1
   }
   return {x_shrunk, output_bits}
}
```

재정규화 과정은 〈알고리즘 3〉의 ANS 부호화 과정 이전에 수행되어 현재 상태값과 현재 부호화하고자 하는 심볼값을 기반으로 현재 상태값을 오른쪽으로 쉬프트 연산을 수행하고 쉬프트된 LSB(Least Significant Bit)를 비트로 출력함으로써 상태값을 범위 내로 유지한다.

rANS의 복호화 과정은 〈알고리즘 5〉와 같으며 각 복호화 과정에서 부호화와 반대로 현재 상태값이 특정 범위 내에 존재하도록 재정규화 과정을 통해 상태값을 왼쪽으로 쉬프트해 준다. 확장된 상태값을 기반으로 심볼을 복호화하며 심볼은 확률 정밀도 값 M의 나머지 연산을 취한 값을 통해 해당 값이 누적 분포 함수에서 어느 심볼 공간에 존재하는지를 확인하여 심볼을 복원할 수 있다.

<알고리즘 5> rANS 복호화 과정

```
ans_decode_step(x, input_bits) {
   x_expand = renormalization_dec(input_bits)
   quo = x_expand//M
   rem = x_expand%M
   s = find_symbol(cumul, rem)
   x_prev = quo*freq[s] + rem − cumul[s]
   return {s, x_prev}
}
```

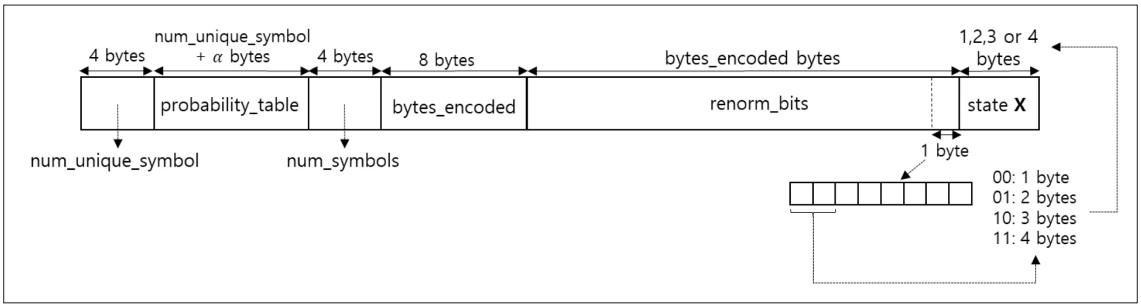
ANS 계열의 알고리즘은 부호화했던 심볼의 순서의 역순으로 심볼들이 복호화된다. 따라서 부호화 순서와 동일한 순서로 복호화하기 위해서는 부호화하고자 하는 심볼을 역순으로 재정렬 후 부호화를 수행해야 한다는 단점이 존재한다. 하지만 Edgebreaker의 역순 복원 방법과 조합되었을 때 별도의 재정렬 없이 전송이 가능하여 효과적으로 Edgebreaker의 심볼 히스토리를 부/복호화할 수 있다.

2) DRACO의 rANS

rANS는 다중 심볼 부호화에 특화된 엔트로피 코딩 방법으로 Edgebreaker 부호화 결과인 심볼 히스토리와 속성 정보 부호화의 결과인 잔차 신호를 별도의 이진화 과정 없이 양의 정수값으로 매핑하여 다중 심볼 부호화를 수행한다. 이때 잔차 신호(*residual*)는 다음 수식을 통해 잔차 심볼(*resSymbol*)로 변환된다.

$$resSymbol = residual\ ?\ 2 * residual : -2 * residual - 1 \quad (7)$$

rANS 기반의 엔트로피 부호화를 수행 시 생성되는 비



<그림 12> DRACO의 비트스트림 구조

트스트림 구조는 〈그림 12〉와 같다.

DRACO는 Edgebreaker의 심볼 히스토리 및 속성 정보의 잔차 심볼을 코딩함에 있어 각 심볼들의 확률값을 부호화기에서 계산하고 이를 전송하는 방식을 채택하고 있다. DRACO 비트스트림은 코딩하고자 하는 심볼의 종류(num_unique_symbol)를 4바이트로 전송하고, 각 심볼별 확률 테이블(probability_table)과 전송할 심볼의 개수(num_symbols)를 비트스트림에 적어준다. 그 후 재정규화를 위한 비트의 크기(bytes_encoded)를 8바이트로 전송하며 이를 통해 재정규화를 위한 비트스트림(renorm_bits)의 마지막 바이트에 접근하여 MSB(Most Significant Bit) 2비트를 통해 초기 상태값의 크기를 전송한다. Valence Edgebreaker의 경우 2~7의 valence별 확률 테이블을 전송하여 Edgbreaker 복호화 과정에서 현재 게이트 엣지의 시작 정점(g,s)에 따라 각 확률 테이블의 확률값을 통해 엔트로피 복호화를 수행하게 된다.

Ⅳ. 결론

본고에서는 구글이 개발한 오픈소스 프로젝트인 DRACO의 메시 압축에 대해 소개하고 DRACO에서 사용하고 있는 메시 데이터의 구성 요소별 세부 압축 기술에 대해 살펴보았다. DRACO는 Edgebreaker를 기반으로 메시의 연결 정보를 압축하고 복원된 연결 정보를 기반으로 속성 정보 예측을 통해 공간적 중복성을 줄이는 압축 구조로 메시 데이터를 압축한다. DRACO는 개발 단계에서부터 압축률과 부/복호화 복잡도를 함께 고려하여 설계되어 매우 빠른 부/복호화 속도로 준수한 압축률을 달성하였다.

하지만 DRACO는 하나의 프레임 내에서만 데이터의 중복성을 제거하는 정적 메시 압축 기술로 추후 시간에 따라 변화하는 동적 메시를 효율적으로 압축하기 위한 기술에 대해 고려가 필요하며 메시 데이터의 양자화를 데이터 요소별로 비트 깊이로만 조절할 수 있어 실제 응용에 사용하기 위해서는 더욱 정밀한 양자화 설계가 필요할 것으로 보인다.

※ 본 연구는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(NRF-2021R1A2 C2092848)의 연구결과로 수행되었음.

참 고 문 헌

[1] Google. Draco: 3D Data Compression, https://github.com/google/draco
[2] ISO/IEC JTC 1/SC 29/WG 07 MPEG 3DGH, "WD 3.0 of V-DMC," N00615, May 2023.
[3] Pierre Alliez, Craig Gotsman. "Recent advances in compression of 3D meshes," *In Advances in Multiresolution for Geometric Modelling*. 3-26. Springer, 2005.
[4] J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes," *IEEE Trans. Vis. Comput. Graphics*, vol. 5, no. 1, pp. 47-61, Jan. 1999.
[5] J. Duda, "Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding," arXiv preprint arXiv:1311.2540, Dec. 2013.
[6] J. Rossignac, "3D compression made simple: Edgebreaker with Zip&Wrap on a corner-table," *In Proc. Int. Conf. Shape Modeling Appl. (SMI)*, pp. 278-283, 2001.
[7] D. Cohen-Or, R. Cohen, and R. Irony. Multiway geometry encoding. Technical report, School of Computer Science, Tel Aviv University, 2002.
[8] A. Szymczak, "Optimized edgebreaker encoding for large and regular triangle meshes," *In DCC '02 Proceedings*. Washington, DC, USA: IEEE Computer Society, 2002, p. 472.
[9] H. Bronnimann, "Designing and implementing a general purpose halfedge data structure," *In Proceedings of the International Workshop on Algorithm Engineering*, vol. 2141, 51-66, Berlin: Springer, 2001.
[10] M. Isenburg and J. Snoeyink, "Spirale Reversi: Reverse Decoding of the Edgebreaker Encoding," *In Proceedings of 12th Canadian Conference on Computational Geometry*, pages 247-256, 2000.
[11] Q. Meyer, J. Süßmuth, G. Sußner, M. Stamminger, and G. Greiner, "On floating-point normal vectors," *In Computer Graphics Forum*, vol. 29, no. 4. Hoboken, NJ, USA: Wiley, 2010, pp. 1405-1409.
[12] Z. H. Cigolle, S. Donow, D. Evangelakos, M. Mara, M. McGuire, and Q. Meyer, "A survey of efficient representations for independent unit vectors," *Journal of Computer Graphics Techniques*, vol. 3, no. 2, 2014.

저 자 소 개

변 주 형

- 2019년 2월 : 광운대학교 컴퓨터공학과 학사
- 2021년 2월 : 광운대학교 컴퓨터공학과 석사
- 2021년 3월 ~ 현재 : 광운대학교 컴퓨터공학과 박사과정
- 주관심분야: 영상신호처리, 영상압축, 컴퓨터비전

심 동 규

- 1993년 2월 : 서강대학교 전자공학과 공학사
- 1995년 2월 : 서강대학교 전자공학과 공학석사
- 1999년 2월 : 서강대학교 전자공학과 공학박사
- 1999년 3월 ~ 2000년 8월 : 현대전자 선임연구원
- 2000년 9월 ~ 2002년 3월 : 바로비젼 선임연구원
- 2002년 4월 ~ 2005년 2월 : University of Washington Senior research engineer
- 2005년 3월 ~ 현재 : 광운대학교 컴퓨터공학과 교수
- 주관심분야 : 영상신호처리, 영상압축, 컴퓨터비전