한국섬유공학회지, Vol. 62, No. 5, 319-326
https://doi.org/10.12772/TSE.2025.62.319

ISSN 1225-1089 (Print)
ISSN 2288-6419 (Online)

얇은 쉘 구조 마네킨 메쉬의 3D 프린팅 필라멘트 소모량 예측

설인환†
금오공과대학교 소재디자인공학과

Filament Usage Prediction in 3D Printing of Thin-Shell-Structured Manikin Mesh

In Hwan Sul†
Department of Materials Design Engineering, Kumoh National Institute of Technology, Gumi 39177, Korea

†Corresponding Author: In Hwan Sul
E-mail: snowman0@kumoh.ac.kr

Received August 18, 2025
Revised October 16, 2025
Accepted October 21, 2025

Abstract: Converting a large object such as a human manikin into small pieces can reduce the printing time and filament usage in 3D printing. On the contrary, the smaller thickness of the pieces can lead to more numerical errors in the mesh operation processes. This research proposes a modified algorithm, which can predict filament usage and the optimal object orientation in 3D printing, for a thin-shell-structure human manikin data, even with zero thickness. The new method was verified for a manikin bodice mesh data with 0 mm, 0.1 mm, 1 mm, and 5 mm thickness. The four data's g-code generation was also checked. The algorithm source code was implemented both for CPU and GPU, and uploaded to Github as open-source.

Keywords: *human manikin, shell mesh, 3D printing, support structure, optimal orientation*

1. 서 론

3D 프린팅은 4차 산업혁명의 대표적인 사례로서 섬유 및 의류 분야에서도 많이 활용되고 있다[1]. 그런데 대중적인 3D 프린터의 출력가능한 크기는 20 cm 내외이므로, 인체 마네킨과 같은 대형 물체의 경우 분할 과정이 별도로 필요하고 출력시간과 필라멘트 소모량도 크다. 이를 다소나마 줄일 수 있는 방안으로는, 물체의 배향을 변경하여 지지구조의 양을 최소화[2]하거나, 물체의 껍질부분만을 쉘(shell) 형태로 출력하여 접합하는 방법[3]이 있다.

첫번째 방안인 지지구조 양 최소화에 관련하여, 대형 물체의 분할과 최적 배향 문제는 기계 공학, 컴퓨터공학 등 관련 분야에서 활발하게 다루어진 바 있는 문제이다. 먼저 물체의 분할 알고리즘은 3D 프린팅뿐만 아니라 용접 로봇의 자동화 문제[4], 자율 주행 자동차의 물체 인식[5], 통계학에서의 군집 분석[6] 등 다양한 분야에서 널리 사용되는 알고리즘[7]이다. 또한 3D 프린팅에 있어서 지지구조 발생량을 최소화할 수 있는 최적화 문제에 대해서도, Ezair 등[8], Morgan 등[9]은 그래픽 카드(GPU)를 이용해 빠른 속도로 특정 배향에 대한 지지구조 부피를 계산하는 방법을 제안한 바 있다. 본 연구진도 이미 선행 연구에서 인체 특징점을 기반으로 하여 대형 마네킨 형상을 자동으로 분할[10]하고, 그 조각들을 3D 프린팅할 때 가장 적합한 배향(orientation)을 미리 계산[11]할 수 있는 일련의 연구를 수행한 바 있다. 이들 선행연구를 통해 단순히 물체의 출력 배향만을 조절하는 것만으로는 필라멘트 소모량이 크게 줄어들지 않는 것을 확인할 수 있었다.

이에 반해 두번째 방안인 물체를 쉘 형태로 출력하는 방법은 필라멘트 소모량을 더 크게 줄일 수 있다[3]. 재료공학에서 쉘 구조는 주로 얇은 형태의 3차원 구조[12]를 의미하며, 물체의 내부 공간과 외부 공간이 삼각형 메쉬들에 의해 분리되어 있는 닫힌 부피(closed-volume)를 형성한다. 반면 3D 프린팅에서 메쉬의 일부분을 분해하면 부피가 없는 삼각형의 집합이 얻어지는데, 이것이 컴퓨터 공학 분야에서 주로 사용하는 쉘의 정의이다. 본 논문에서는 쉘이라고 하면 재료공학적 의미를 사용하되, 따로 부피가 없는 삼

각형인 경우에 대해서는 단면 구조(one-sided structure)라고 구분하도록 하겠다.

컴퓨터 공학에서도 이러한 단면구조 쉘은 매우 예외적인 데이터 구조이다. 이를 제외한 대부분의 영역에서는 항상 부피를 가지는 물체, 즉 다양체(manifold) 모델을 기본적인 입력 데이터로 가정한다. 반면 부피가 없는 점, 선, 면 등의 구조를 비다양체(non-manifold) 모델이라고 부르는데, 이는 AutoCAD, SketchUp과 같은 경계 표현(boundary representation; B-rep) 방식의 CAD SW의 초기 작도 과정에서 디자이너가 실수로 남긴 것이라든가, 3D 스캐너 사용 시 얻어지는 점 데이터(point cloud)가 대표적인 사례이다. 이들 비다양체 모델을 그대로 남길 경우 이후 CAD/CAM SW의 알고리즘 처리 과정에서 예측할 수 없는 오류[13]가 존재할 수도 있기 때문에, 대부분의 SW에서는 사용자가 형상을 디자인할 때마다 비다양체 모델이 존재하는지를 오일러 함수(Euler operator)라는 평가함수를 통해 체크하게 된다[14].

대형 물체를 3D 프린팅하는 경우, 사용자가 고체화(solidify)와 같은 별다른 후처리를 하지 않는다면 분할된 메쉬 조각들은 모두 단면 구조의 쉘 데이터가 된다. 그런데 본 연구진을 포함하여 선행 연구들은 이러한 쉘 데이터에 대한 3D 프린팅 필라멘트 소모량 예측에 대해서는 크게 다룬 바가 없다. 따라서 본 연구에서는 일반 섬유/의류 종사자가 FDM 타입 3D 프린터로 대형 인체 마네킨 형상을 출력하고자 할 경우, 분할된 메쉬 조각이 단면 쉘 구조이더라도 배향별 필라멘트 소모량을 예측할 수 있는 계산기법을 제안하고자 한다. 입력 메쉬 데이터로는 구(sphere) 형태와, 인체 마네킨의 몸통(bodice)을 사용하되 두께를 0, 0.1, 1, 5 mm의 네 가지로 사용하였다. 이에 대한 검증용으로는 3D 프린터의 슬라이싱 SW에서 측정한 이론적 필라멘트 소모량을 사용하였다. 또한 쉘 두께별로 지코드(g-code)가 잘 생성되는지 여부도 확인하였다. 소스코드는 오픈소스화하였으며, 주요 계산은 C++ 언어로, 결과의 렌더링은 파이썬 기반으로 작성하였다. 또한 빠른 연산을 위해 C++ 기반 계산 파트는 CPU 버전와 GPU 버전 두 가지로 개발하였다.

2. 이론적 배경

2.1. 선행 연구의 지지구조 단층 촬영법

본 연구진의 선행 연구[11]에서는 닫힌 부피를 갖는 일반적인 3차원 물체에 대해 그 지지구조량을 가상의 태양으로부터 얻어지는 물체의 그림자 부피로 비유하고, 이를 지지구조 단층촬영법이라 명명하였다. 계산을 간단히 하기 위해 삼각형 메쉬 데이터를 먼저 가로, 세로, 높이가 각각 1 mm 크기인 복셀로 변환하였다. 이후 *Figure* 1(a)와 같이 태양으로부터 가상의 수직선이 물체와 충돌하는 지점을 찾아 픽셀(pixel)이라고 명명하고, 픽셀은 다시 법선 방향(normal vector)이 태양 쪽이면 알파(α) 픽셀, 지면 방향이면 베타(β) 픽셀이라고 명명하였다. 임의의 직선이 닫힌 부피를 갖는 일반적인 도형을 가로지는 경우에는 항상 짝수번 만나게 되므로, 알파 픽셀과 베타 픽셀은 항상 짝을 지어 존재하게 된다. 예를 들어 *Figure* 1(a)의 사례에서는 (α_0, β_0) 와 (α_1, β_1)가 각각 쌍을 이루고 있다. 이 사실을 이용하면 짝을 이루지 않는 나머지 픽셀들은 모두 노이즈로 간주하여 쉽게 제거할 수 있는 장점이 있다. 반면 지지구조를 형성할 수 있는 구간은 베타 픽셀로부터 출발하여 알파 픽셀에서 끝나는 구간으로서, *Figure* 1(a)의 사례에서 (β_0, α_1), (β_1, α_{bottom})의 두 구간이다. 이 중에서 임계각 기준에 따라 (β_0, α_1) 구간이 실제 지지구조 구간(support structure; *SS*)이 되고, 나머지 (β_0, α_1) 구간은 지지구조 형성에는 영향을 미치지 않으므로 비지지구조 구간(Non-oVerhang; *NV*)라 명명하였다.

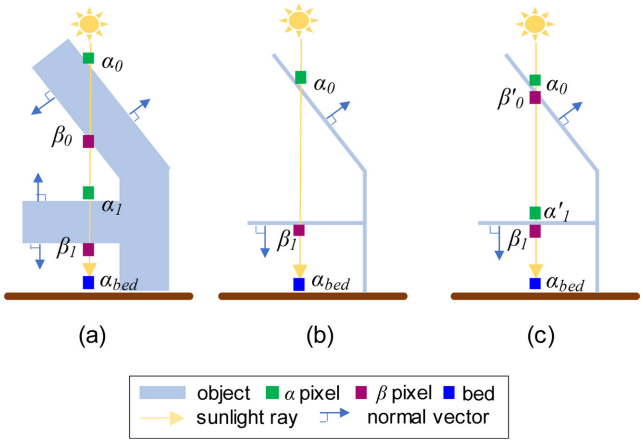


Figure 1. Brief comparison of the proposed algorithm to the previous one; (a, b) previous method for closed/non-closed meshes, respectively, and (c) the modified method for the non-closed mesh.

2.2. 쉘 메쉬용 개선된 알고리즘

Figure 1(b)는 이전 알고리즘을 쉘 형태 메쉬에 대해 적용한 개념도이다. 이 경우 메쉬의 두께가 얇거나 0인 경우에는 물체와 가상의 태양광이 만나는 점이 부피를 형성하지 못하고 α_0와 β_1 같이 점 형태로만 존재하게 된다. 즉, 실제로는 지지구조가 발생해야 하지만 지지구조 양이 0인 것처럼 잘못된 연산이 이루어진다. 이를 개선하기 위해 이번 연구에서는 *Figure* 1(c)와 같이, 물체가 쉘 형태인 경우에는 태양광이 물체를 접할 때마다 추가적으로 법선 방향이 반대인 픽셀을 추가해 준다. 즉 *Figure* 1(c)에서 처음 만나는 접점은 α_0이지만 이를 부피를 가진 형태로 바꿔 주기 위해 일정 거리 아래에 β'_0라는 가상의 픽셀을 추가해 주는 방식이다. 반대로 β_1이라는 접점에 대해서는 아래가 아니라 반

대 방향인 그 위쪽 방향으로 일정 간격을 두고 α'_1라는 가상의 점을 추가해주는 것이다. 이 때 간격의 크기는 복셀화에 사용한 크기, 즉 1 mm를 기본값으로 사용하였다. 이와 같은 방식을 반복하면 단면구조 쉘 메쉬에 대해서도 *Figure* 1(a)의 경우와 마찬가지로 (β_0, α_1)이라는 지지구조 구간을 얻을 수 있다.

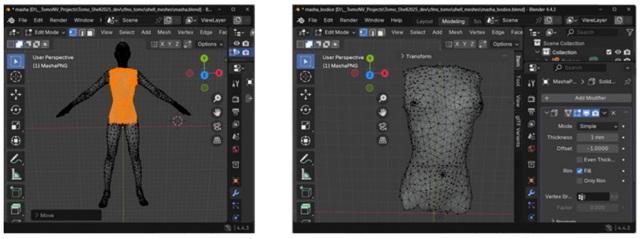
3. 실 험

3.1. 3D 프린터

(주)신도리코의 FDM 방식 3D 프린터 DP-202와 PLA 필라멘트를 출력 대상으로 가정하였다. 이 프린터의 최대 출력 크기는 200×200×189 mm^3이고, 번들 소프트웨어인 3DWOX의 지코드(g-code) 작성을 통해 얻은 필라멘트 소모량 예측치를 실험 대조군으로 사용하였다. 지코드 생성 시 베드 고정 옵션은 래프트(raft) 형상을, 내부채움과 지지구조 형상은 직선형으로 하였으며, 기타 파라미터들은 *Table* 1과 같이 기본 설정된 값을 그대로 이용하였다. PLA 필라멘트의 밀도(1.121 g/cm^3)는 한국고분자시험연구소㈜에 의뢰하여 ASTM E1461에 의해 측정하였다.

Table 1. Specification of the "default" g-code generation options for the slicing software

Category	Option	Value
Filament	Material	PLA
	Diameter	1.75 mm
	Density (ρ_{PLA})	0.0121 g/mm^3
	Layer *height*	0.20 mm
Wall ("clad")	Thickness (T_{clad})	0.8 mm
	Infill ratio (F_{clad})	100%
Infill ("core")	Infill ratio (F_{core})	15%
	Fill pattern	linear
Support structure	Critical angle (θ_c)	60 °
	Infill ratio (F_{ss})	20%
	Fill pattern	linear
	Line width (C_{ss})	100%
	Horizontal expansion (C_{ss})	0.8 mm
Bed plate	Skirt line interval	3 mm
	# of Skirt lines	1
	Bottom layer height	0.2 mm
	# of brim lines	10
	Raft size	2 mm
	Base thickness	0.3
	Interface thickness	0.27
	# of surface layer	2
	Surface layer thickness	0.4 mm



(a) (b)

Figure 2. Preparation of manikin bodice shell mesh using Blender SW; (a) manual selection of bodice part and (b) mesh solidification process.

Table 2. Specification of the mesh data used

Name	Dimension (mm^3)	Thickness (mm)	#of Vtx/ Face	Closed-Volume
Sphere	Radius 90	90	19,602/39,200	O
Hemisphere	Radius 90	0	10,201/20,200	X
Manikin	135×33×173	various	7,265/13,672	O
Bodice0	62×56×119	0	1,025/1,739	X
Bodice0.1		0.1	2,050/3,940	O
Bodice1		1.0	2,060/3,940	O
Bodice5		5.0	2,050/3,940	O

3.2. 입력 메쉬 데이터

사용된 형상 데이터 중 구 데이터는 FinalMesh 소프트웨어를 이용하여 도안한 뒤 스탠포드 .PLY 포맷으로 저장하여 사용하였다. 인체 마네킨 데이터는 모델링된 데이터(삼각형 13,672개)를 온라인 사이트(Turbosquid.com)에서 구매하여 사용하였다. 몸통 부위를 쉘 형태로 준비하기 위해, *Figure* 2(a)와 같이 무료 모델링 SW인 Blender (버전 4.4.3)에서 인체 마네킨 데이터의 몸통(bodice) 영역을 마우스로 수작업으로 드랙하여 선택하였고, 단면 구조인 상태 그대로 일단 저장하여 이를 *Table* 2와 같이 "bodice0"이라는 이름으로 지칭하였다. 별도로 또한 *Figure* 2(b)와 같이 Blender의 입체화(solidify) 기능을 이용해 닫힌 메쉬로 변환하되, 두께를 0.1 mm, 1 mm, 5 mm의 세 가지로 저장하였고, 이름을 *Table* 2와 같이 "bodice0.1", "bodice1", "bodice5"라고 하였다. 한편 원래 마네킨의 키는 173 mm이라고 가정하였지만, 이를 절단한 몸통(bodice)들은 렌더링 시 편의를 위해 크기를 두배로 확대하여 장축 방향 길이가 119 mm가 되도록 조정하였다. *Table* 2는 사용된 메쉬 파일의 세부사항을 나타내고 있다.

3.3. 예상 필라멘트 부피 및 질량 계산

3D 프린팅에 필요한 필라멘트 소모량 계산방법은 선행연구[11,15]와 동일하며 요약하여 정리하면 다음과 같다.

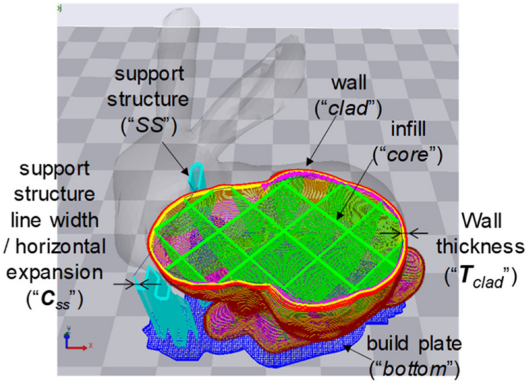


Figure 3. Cross-sectional view of internal and support structure in slicing software [15].

입력 메쉬 데이터는 3D 프린터의 최대 크기를 고려하여 256×256×256 mm^3의 복셀 공간에 위치시킨 후 삼각형이 위치하는 복셀만 선택적으로 저장하는 복셀화 과정을 거쳤다. 3D 프린터의 슬라이싱 SW는 보통 출력 속도를 위해 물체의 내부 등 일부는 *Figure* 3과 같이 최대한 비운 상태로 지코드를 생성한다. 이를 반영하기 위해 각 부위를 외벽(clad), 내부(core)로 구분하고, 복셀 개수를 이용해 부피를 계산한 뒤 밀도 값(ρ_{PLA})을 곱해 질량을 구하였다.

먼저 물체 자체의 외벽과 내부를 출력하는데 필요한 필라멘트 질량은 식 (1), (2)와 같다. 이 때 외벽 충진율(F_{clad})은 부위에 상관없이 항상 1.0이며, 내부 충진율(F_{core})은 설정에 따라 다르지만 통상 15–20% 정도의 값을 갖는다.

$$M_{o\_clad} = V_{o\_clad} \cdot F_{clad} \cdot \rho_{PLA} \quad (1)$$

$$M_{o\_core} = V_{o\_core} \cdot F_{core} \cdot \rho_{PLA} \quad (2)$$

전체 지지구조의 부피(V_{ss})는 *Figure* 1(c)에서 구한 복셀들의 총 부피이며, 이를 내외부 충진율(F_{clad}, F_{core})을 반영하기 위해 식 (3)~(5)와 같이 외벽(V_{clad})과 내부(V_{core})로 분리한 뒤 질량(M_{ss_clad}, M_{ss_core})을 구하였다.

$$V_{ss\_clad} = V_{ss} - V_{ss\_clad} \quad (3)$$

$$M_{ss\_clad} = V_{ss\_clad} \cdot F_{clad} \cdot \rho_{PLA} \quad (4)$$

$$M_{ss\_core} = V_{ss\_core} \cdot F_{core} \cdot \rho_{PLA} \quad (5)$$

바닥구조의 부피(V_{bottom})와 질량(M_{bottom})은 바닥을 형성하는 복셀의 총 개수(N_{bottom})와 복셀의 한 변 크기(d_{voxel})로부터 식 (6), (7)을 통해 알 수 있다.

$$V_{bottom} = N_{bottom} \cdot d_{voxel}^3 \quad (6)$$

$$M_{bottom} = V_{bottom} \cdot \rho_{PLA} \quad (7)$$

최종적으로 전체 필라멘트 소모량은 식 (8)와 같이 전체 부위들의 합으로 나타내어 진다.

$$M_{total} = M_{o\_clad} + M_{o\_core} + M_{ss\_clad} + M_{ss\_core} + M_{bottom} \quad (8)$$

Table 3. Specification of the CPU

ID	Product name	Base/max clock (GHz)	# of cores	Single-precision TFLOPs†
CPU1	AMD Ryzen5 7500f	3.7/5.0	6	1.3
CPU2	AMD Ryzen9 7950X	4.3/5.7	16	1.8
GPU1	NVIDIA RTX 4060	1.8/2.7	3,072	15.1
GPU2	NVIDIA RTX 4090	2.2/2.6	16,384	82.5

†: Tera FLoating point Operation Per Second.

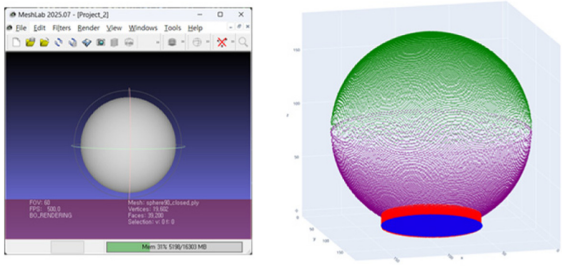
3.4. 소프트웨어 및 하드웨어

복셀화 등 주요 알고리즘은 C++ 언어를 이용해 .DLL 파일로 작성하였다. 연산 속도 최적화를 위해 CPU 버전에서는 Visual Studio 2019컴파일러에 내장된 OpenMP 병렬 연산 기능을 이용하였고, GPU 버전에서는 NVIDIA CUDA 툴킷(버전 12.5)을 사용[11]하였다. 연산된 결과의 렌더링은 matplotlib, plotly, polyscope 등 패키지를 이용해 python 기반으로 구현하였다. 이들 개발된 소스코드는 깃허브(https://github.com/cfms-lab/Tomo_Shell2025)를 통해 공유하였다. 중저가형과 고급형 하드웨어에서의 실행 속도를 비교하기 위해 *Table* 3과 같이 각각 두 가지의 CPU 및 GPU를 사용하였다.

4. 결과 및 고찰

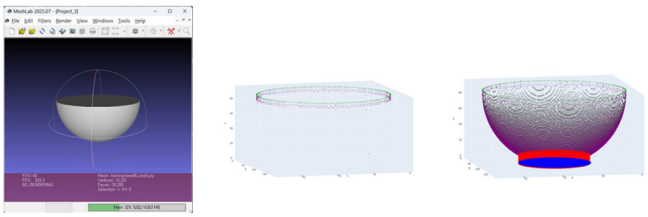
4.1. 단면 쉘 메쉬의 지지구조량 분석

Figure 4는 반지름 9 cm의 구(sphere) 형상을 갖는 닫힌 메쉬(closed mesh)를 이전 알고리즘으로 지지구조를 예측한 결과이다. *Figure* 4(a)는 MeshLab SW에서 메쉬 원본의 형상을 캡쳐한 것이며, *Figure* 4(b)는 이 데이터를 *Figure*



(a) (b)

Figure 4. Example of the previous method applied to a closed mesh; (a) sphere data captured by MeshLab software and (b) pixelization result. Green/purple/red/blue pixel colors correspond to α/β/SS/bed, respectively.



(a) (b) (c)

Figure 5. Example of non-closed method and its pixelization results; (a) hemisphere's screenshot, (b) the previous method, and (c) the modified method.

1(a)의 알고리즘에 입력하여 부위별 픽셀을 분석한 결과이다. 픽셀의 초록, 보라, 빨강, 파랑 색상은 각각 알파 픽셀, 베타 픽셀, 지지구조 픽셀, 바닥 구조를 의미한다. 지지구조 생성의 임계각은 *Table* 1에서와 같이 60 °이므로, 이에 해당하는 구간에만 지지구조가 발생한 것을 확인할 수 있다.

Figure 5(a)는 *Figure* 4(a)에서 사용한 구 형상의 아랫 부분 절반을 잘라내서 단면 쉘 메쉬로 저장한 것이다. 이 반구(hemisphere)는 법선 벡터가 구의 바깥쪽에만 존재하기 때문에, 바깥쪽은 밝게 보이고 안쪽은 검게 렌더링되어 있는 것을 확인할 수 있다. 이를 *Figure* 1(a)의 알고리즘에 적용한 결과, *Figure* 5(b)와 같이 대부분의 픽셀이 사라지는 비정상적인 결과가 도출되었다. 이는 반구 표면에 베타 픽셀만 생성되었다가, 이에 대응하는 알파 픽셀의 짝을 찾을 수 없으므로 알고리즘이 모든 베타 픽셀을 노이즈로 처리하였기 때문이다. 반면 *Figure* 1(c)의 새로운 알고리즘을 적용한 결과가 *Figure* 5(c)이다. *Figure* 4(b)에서와는 달리 각 초록색 베타 픽셀마다 1 mm 아래에 새로 생성된 보라색 알파 프라임 픽셀이 짝을 지어 존재하고 있고, 이로 인해 빨간색 지지구조 영역이 정상적으로 검출된 것을 확인할 수 있다. 즉 개선된 알고리즘은 기존의 닫힌 부피를 갖는 물체에 대해서도 적용가능할 뿐만 아니라, 두께가 0인 단일 방향 쉘 구조 메쉬에 대해서도 정확하게 필라멘트 양을 예측할 수 있음을 보여준다. 참고로 입력 메쉬가 닫힌 부

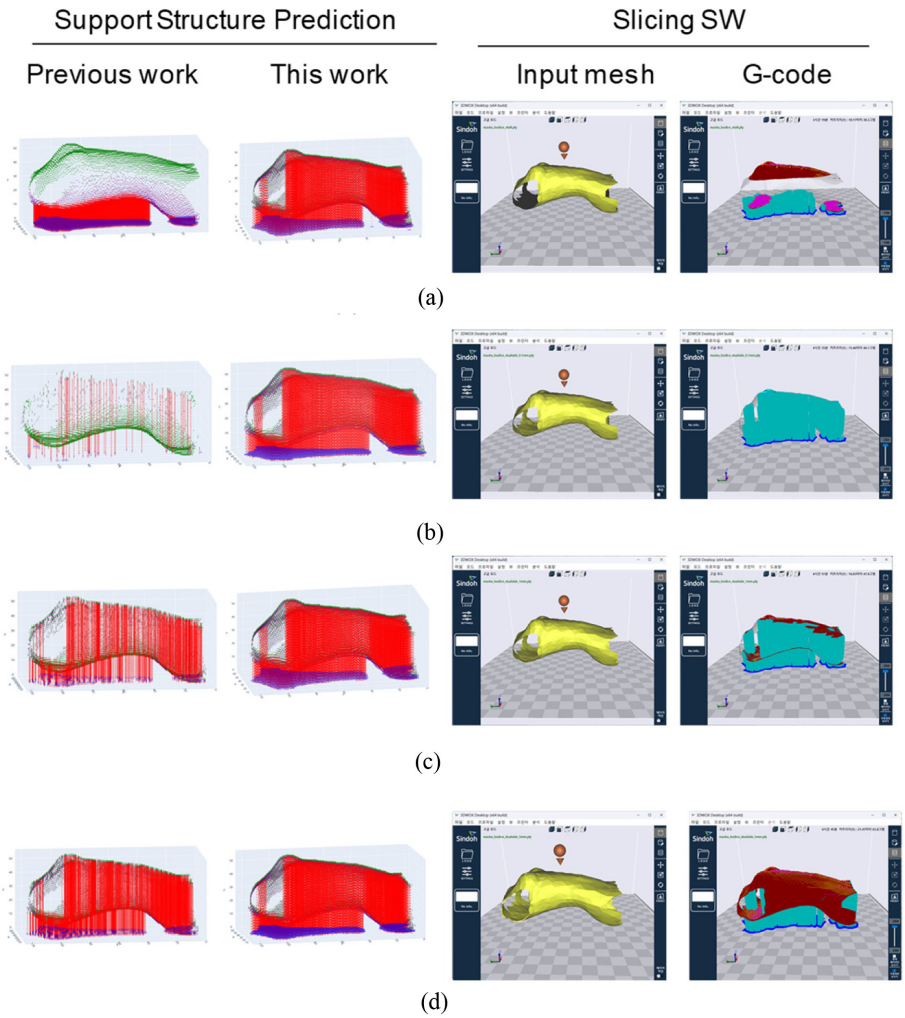


Figure 6. Result of support structure prediction and comparison with slicing software; (a) bodice0, (b)bodice0.1, (c) bodice1, and (d) bodice5.

피인지 아닌지는 파이썬에서 trimesh 패키지의 is_watertight() 함수를 이용해 쉽게 알 수 있다.

4.2. 마네킨 몸통 형상에 대한 적용

개선 전후의 알고리즘을 인체 마네킨 형상에도 적용한 결과가 *Figure* 6이다. 각 그림의 가장 왼쪽 세로 줄은 *Figure* 1(a)의 이전 알고리즘의 적용결과를 나타내고, 그 다음 줄은 *Figure* 1(c)의 개선된 알고리즘의 적용 결과이다. 네 가지 몸통 데이터의 벽 두께인 0 mm, 0.1 mm, 1 mm, 5 mm는 사용된 3D 프린터의 레이어 높이(0.2 mm), 외벽 두께(0.8 mm) 및 지지구조 분석 알고리즘에서 사용한 복셀의 크기(1 mm)를 고려하여 설정한 것이다. 이전 알고리즘의 경우 *Figure* 6(d)와 같이 벽 두께가 복셀 크기보다 커야 정확한 픽셀 정보가 얻어진데 반해, 개선된 알고리즘은 *Figure* 6(b)와 같이 필라멘트 레이어 높이보다 작거나 심지어 *Figure* 6(a)와 같이 두께가 0인 경우에도 지지구조 분석이 가능하였다. 인체 부위 중에서 몸통을 선택한 이유는 팔, 다리, 머리 등 다른 부위가 원통에 가까운 형상인데 반해 다소 복잡한 형태를 가지고 있기 때문이다. 물론 필요에 따라 몸통을 추가로 분할할 필요가 있을 수도 있으나, 이에 대해서는 차후 연구에서 자세히 다룰 예정이다.

Figure 6의 세로 세번째 줄과 네번째 줄은 메쉬 데이터를 슬라이싱 SW에 넣어 지코드를 생성한 결과이다. *Figure* 6의 네 가지 데이터 중에서 a의 경우는 필라멘트 높이(0.2 mm)보다 항상 작은 두께를 가지고, b, c의 경우는 경사 각도에 따라 사라지는 구간이 발생하게 되었다. 이와 관련하여 본 연구진은 3D 프린터의 지코드 생성 과정에 있어서 필라멘트 두께보다 얇은 부분은 사라지게 되는 현상을 이전 연구[16]에서 확인한 바 있다. 즉 3D 프린팅 시 필라멘트 레이어 두께가 0.2 mm이더라도, 물체의 배향에 따라 실제로는 물체의 두께를 0.5 mm 이상으로 디자인해야 안전하게 지코드가 생성가능하다. 이와 마찬가지 이유로 인해 쉘 구조의 지지구조 분석에서도 *Figure* 6(d)처럼 레이어 높이(0.2 mm)와 벽 두께(0.8 mm)보다 충분히 큰 두께를 갖는 형상에 대해서만 완전한 지코드가 생성되었음을 알 수 있다. 물론 쉘의 벽 두께가 커질수록 출력 시간이 증가하는 단점은 존재한다.

따라서 쉘 구조에 대해 지지구조량 분석을 할 때에는 개선된 알고리즘을 사용하되 부피가 없는 단면 쉘 메쉬 그대로 입력하고, 이후 최적 배향이 결정되면 지코드 생성에 있어서는 고체화한 메쉬를 생성하여 부피를 갖는 메쉬 데이터로 변환하되 두께는 작은 값부터 순차적으로 적용하여 실행해 보는 것이 바람직해 보인다.

마지막으로 예측된 지지구조량을 정량적으로 검증하기 위해, *Figure* 6(a)와 같은 지지구조 분석 과정을 X, Y축을 중심으로 각각 30도 간격으로 회전시켜가면서 측정하고, 이에 대응한 실제값으로는 슬라이싱 SW에서 얻은 필라멘트

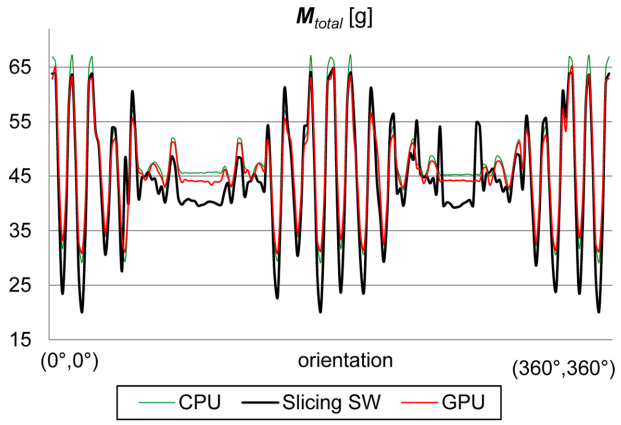


Figure 7. Quantitative comparison of total filament mass predicted by our modified method (black line) and the slicing software (green line).

량을 수작업으로 계측하여 비교하여 보았다. 초록색과 빨간색은 *Figure* 1(c)의 개선된 알고리즘의 CPU 버전과 GPU 버전으로 각각 계산한 총 필라멘트 무게(M_{total})이고, 검정색은 슬라이싱 SW에서 지코드 생성시 출력되는 필라멘트 총 무게값이다. 슬라이싱 SW를 기준으로 할 때 본 알고리즘은 CPU와 GPU 버전 모두 오차가 최대 65% 발생하였다. 이는 본 알고리즘이 3D 프린터의 슬라이싱 옵션 중 일부 파라미터만을 모델링에 반영했기 때문이다. 그러나 극대와 극소점의 경향성에 있어서는 매우 유사한 면을 보이고 있으므로, 필라멘트 소모량의 절대값을 찾기 보다는 소모량이 최소가 될 수 있는 물체 최적 배향(orientation)을 찾는데 사용하는 것은 가능해 보인다. GPU 버전의 오차가 CPU보다 조금 더 큰 편인데, 이는 GPU의 하드웨어 구조가 CPU와 달라 CPU의 알고리즘을 그대로 이식할 수 없었기 때문이다.

연산시간은 *Table* 4, *Table* 5과 같다. 단일 배향에 대한 *Table* 4의 결과에서는 연산량이 크게 많지 않으므로 CPU1와 CPU2, 또는 CPU와 GPU 간의 속도 차가 크지 않았다. GPU가 오히려 느린 것은 GPU 초기화와 GPU 메모리로의 데이터 복사에 시간이 걸리는 것이 원인으로 보인다. 반면

Table 4. Calculation time of the modified algorithm for the support structure prediction at the default (0,0) orientation (unit: second)

Name	CPU1	CPU2	GPU1	GPU2
Sphere	0.05	0.02	0.33	0.17
Hemisphere	0.02	0.01	0.15	0.15
Manikin	0.01	0.01	0.12	0.12
Bodice0	0.01	0.01	0.09	0.12
Bodice0.1	0.02	0.01	0.12	0.13
Bodice1	0.01	0.01	0.13	0.13
Bodice5	0.02	0.01	0.11	0.14

Table 5. Calculation time of the modified method for optimal orientation search (angle interval=1 °, unit: [minute:second])

Name	CPU1	CPU2	GPU1	GPU2
Sphere	3:57.4	1:34.1	1:02.9	0:20.2
Hemisphere	2:59.1	1:00.5	0:53.2	0:14.8
Manikin	2:07.2	0:56.2	0:37.8	0:13.1
Bodice0	1:40.8	0:50.2	0:33.5	0:11.2
Bodice0.1	1:51.6	0:54.5	0:38.7	0:11.2
Bodice1	1:54.1	0:54.0	0:39.8	0:10.8
Bodice5	1:54.2	0:51.7	0:37.9	0:10.8

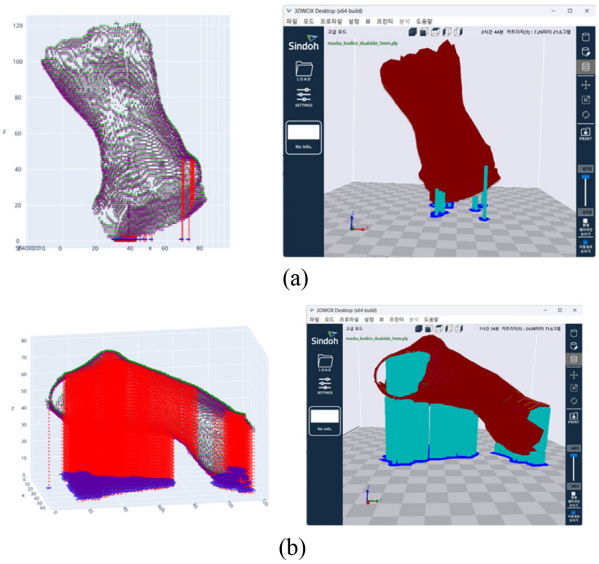


Figure 8. Best and worst orientations for the 5mm thickness bodice mesh data; (a) best orientations (85 °, 157 °), M_{total} = (29.0 g/21.6 g) and (a) worst orientations (197 °, 174 °), M_{total} = (70.3 g/71.6 g).

Table 5은 최적 배향을 찾기 위해 yaw, pitch의 탐색 간격을 각각 1도로, 즉 *Table* 4의 연산을 360×360=129,600번 반복하는 셈인데, 이 경우에는 개별 연산 속도는 CPU에 비해 떨어지지만 대량 병렬 연산에 강한 GPU가 최종적으로는 더 빠른 속도를 보였다. 그러나 딥러닝 분야에서는 데이터가 일렬로 벡터화되어 있어서 GPU의 속도가 CPU보다 100여배 이상 빠른 것이 일반적인데, 이번 연구와 같이 픽셀 데이터들이 이산되어 있는 형태에는 GPU 연산의 속도 개선 효과는 크지 않아 보인다. *Figure* 8은 필라멘트 소모량이 최소, 최대일 때의 배향과 필라멘트량을 각각 나타낸 것이다.

5. 결 론

대형 인체 마네킹을 FDM 방식 3D 프린터로 제작함에 있어서, 마네킹 메쉬 데이터를 분할하고 내부가 빈 단면 쉘 형태로 변환함으로써 필라멘트 소모량과 출력시간을 줄이는 것을 목표로 하였다. 이때 부피가 0인 단면 쉘 메쉬를 처리하기 위해, 생성된 알파, 베타 픽셀에 대해 각각 베타 프라임, 알파 프라임 픽셀을 추가하는 방법을 도입하였다. 개선된 알고리즘을 구 형상 메쉬와 인체 마네킹의 몸통 부위에 적용한 결과, 쉘의 두께에 관계없이 모두 필라멘트 소모량과 최적 배향을 계산할 수 있었다. 슬라이싱 SW의 파라미터 중 일부만을 모델링에 적용하였으므로, 일부 배향 각도에서는 슬라이싱 SW와 본 알고리즘으로 계산한 필라멘트 총 무게값이 최대 65% 정도의 차이를 나타났다. 그러나 본 연구에서 필요로 하는 극소점에서는 슬라이싱 SW와 CPU, GPU 모두 동일한 경향성을 나타내었다. 특히 고성능 GPU를 이용할 경우에는 1만~4만 개의 많은 삼각형을 갖는 메쉬에 대해서도 10~20여 초의 짧은 시간안에 필라멘트 소모량이 최소가 되는 최적 배향을 계산할 수 있었다. 그런데 0.05 mm 이하의 얇은 쉘의 경우는 지코드가 생성되지 못하는 문제점이 있었다. 따라서 슬라이싱 과정에 있어서는 쉘 메쉬가 적절히 두꺼운 두께를 가질 수 있도록 고체화 변환하는 과정이 필요하였다. 또한 필라멘트 총 무게량 오차를 개선하기 위해서는 슬라이싱 SW의 파라미터들을 모델링에 추가할 필요가 있다. 알고리즘은 연산 속도를 위해 C++ 언어와 파이썬을 이용하였고, 소스 코드는 관련 연구자가 쉽게 활용할 수 있도록 깃허브를 통해 공유하였다. 이번 연구는 인체 형상을 수작업으로 분할하여 입력 데이터로 사용하였으나, 차후 연구에서는 통계적 기법을 이용해 마네킹 메쉬를 자동으로 분할하는 내용을 추가적으로 다룰 예정이다.

감사의 글: 이 연구는 한국연구재단 이공분야 기초연구사업에 의하여 지원된 논문입니다(NRF-2022R1A2C1010072).

References

1. EMPA. “12th International Manikin and Modelling Meeting (12i3m)”, Empa - Federal Laboratories for Materials Science and Technology, St. Gallen, Switzerland, 2018.
2. J. Y. Jung, S. Chee, and I. H. Sul, “Support Structure Tomography Using Per-pixel Signed Shadow Casting in Human Manikin 3D Printing”, *Fash. Text.*, 2022, **9**, 1–18.
3. J. Vanek, J. G. Galicia, B. Benes, R. Měch, N. Carr, O. Stava, and G. Miller, “PackMerger: A 3D Print Volume Optimizer”, *Computer Graphics Forum*, 2014, **33**, 322–332.
4. B. Massoni and M. I. Campbell, “Automated Decomposition of Complex Parts for Manufacturing with Advanced Joining Processes”, *J. Manuf. Sci. Eng.*, 2020, **142**, 1–35.
5. M. Herb, T. Weiherer, N. Navab, and F. Tombari, “Lightweight Semantic Mesh Mapping for Autonomous Vehicles”, 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, pp.6732–3738.

6. A. Jaeger and D. Banks, "Cluster Analysis: A Modern Statistical Review", Wiley Interdisciplinary Reviews: Computational Statistics, 2023, **15**, e1597.
7. R. S. Rodrigues, J. F. Morgado, and A. J. Gomes, "Part-based Mesh Segmentation: A Survey", *Computer Graphics Forum*, 2018, **37**, 235–274.
8. B. Ezair, F. Massarwi, and G. Elber, "Orientation Analysis of 3D Objects Toward Minimal Support Volume in 3D-printing", *Comput. Graph.*, 2015, **51**, 117–124.
9. H. Morgan, J. Cherry, S. Jonnalagadda, D. Ewing, and J. Sienz, "Part Orientation Optimisation for the Additive Layer Manufacture of Metal Components", *Int. J. Adv. Manuf. Technol.*, 2016, **86**, 1679–1687.
10. J. Y. Jung, S. Chee, and I. H. Sul, "Automatic Segmentation and 3D Printing of A-shaped Manikins using a Bounding Box and Body-feature Points", *Fash. Text.*, 2021, **8**, 1–21.
11. J. R. Kim and I. H. Sul, "Fast Prediction of 3D Printing Optimal Orientation Using General-purpose Graphic Processor Unit Calculation (Accepted)", 3D Printing and Additive Manufacturing, 2025.
12. O. C. Zienkiewicz and R. L. Taylor, "The Finite Element Method for Solid and Structural Mechanics", Elsevier, 2005.
13. D. Naresh, "Effect of Naked and Non-manifold Errors on Polymer Printing Process", *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 2023, **17**, 1417–1428.
14. J. G. Lambourne, K. D. Willis, P. K. Jayaraman, A. Sanghi, P. Meltzer, and H. Shayani. "Brepnet: A Topological Message Passing System for Solid Models", Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp.12773–12782.
15. J. Y. Jung and I. H. Sul, "Calculation Speed Improvement of Optimal 3D Printing Orientation Prediction Using Closed-Volume-Voxel Structure", *Text. Sci. Eng.*, 2023, **60**, 306–312.
16. J. Y. Jung, Y. J. Park, J. W. Lim, D. B. Park, S. M. Lee, J. E. Kwon, and I. H. Sul, "Observations on Dimension Change of Segmented 3D Printing Human Manikin Parts for Optimal Combination", *Text. Sci. Eng.*, 2022, **59**, 88–93.