한국CAD/CAM학회 논문집
Transactions of the Society of CAD/CAM Engineers

Vol. 18, No. 4, pp. 258-266. August 2013
DOI http://dx.doi.org/10.7315/CADCAM.2013.258

〈응용논문〉

ISSN 1226-0606

선박 블록 단위의 대용량 JT 파일을 안드로이드 기기에서 가시화하는 방법

천상욱† · 서흥원

대우조선해양㈜ 정보기술팀

A Method for Visualizing a Large JT File of Ship Blocks in an Android Device

Sanguk Cheon† and Heung-Won Suh

Information Technology Team, DSME

Received 26 March, 2013; received in revised form 17 April, 2013; accepted 6 May, 2013

ABSTRACT

In shipbuilding, 2D manufacturing drawings are crucial for building a ship. Even various types of 3D models are being utilized for supporting ship manufacturing, which does not reduce the importance of 2D drawings. Recently things are changing in the shipbuilding industry. To reduce the number of 2D drawings or to reduce the quantity of information contained in 2D drawings, some attempts that can substitute for 2D drawings are being made. One of the attempts is to visualize lightweight 3D manufacturing models in a mobile device. In this paper, a method for displaying lightweight 3D models of a ship in an Android based device is introduced. To overcome the problem with parsing JT files in Android system, JT files are parsed in a Windows based server and as-simple-as-possible visualization data are transmitted to an Android based viewer. A comparison result with a commercial system is also given.

Key Words: Android, JT viewer, Lightweight model, Mobile device, Shipbuilding

1. 서 론

조선소에서 2D 생산 도면은 주요한 생산 정보를 담고 있어서, 현장에서는 2D 도면에서 생산에 필요한 정보를 얻는다. 간단한 2D 생산 도면에서는 직관적이고 빠르게 정보를 찾을 수 있지만, 복잡한 2D 생산 도면에서는 원하는 정보를 찾기가 어렵다. 예를 들어, 의장품 설치도의 경우, 설치할 자재가 어디에 어떻게 설치될 것인지를 2D 설치 도면에서는 파악하기 어려운 경우가 있다. 2D 생산 도면에서 필요한 설치 정보를 파악하기 곤란한 경우, 사무실 PC의 CAD 시스템에서 필요 정보를 확인한 다음 다시 생산 현장으로 돌아가야 하는 낭비가 발생한다. 생산현장에서 바로 모바일 기기를 통해 3D 모델을 확인할 수 있다면, 작업 모델에 대한 정확한 이해를 바탕으로 생산 오작을 줄일 수 있다.

이는 현재의 국내 조선소에서 중요한 의미를 가진다. 국내 주요 조선소는 종래의 상선 위주의 생산에서 복잡한 해양 위주의 생산 체계로 이미 변화한 상태이다. 상선과 달리 해양 프로젝트에는 그

†Corresponding Author, su.cheon@gmail.com

복잡성과 프로젝트별 특성으로 인해 현장 작업자들에게 생소한 자재가 많이 사용된다. 익숙하지 않은 자재를 2D 생산 도면만을 이용하여 찾고, 설치하는 것은 쉬운 일이 아니다. 이렇듯 생산 현장에서 2D 도면을 보조하여 작업 자재를 3D 모델로 확인하는 것은 단지 3D 모델을 활용하는 것 이상의 의미를 가진다. 곧, 조선 CAD 시스템을 사용하지 않고, 생산 현장에서 모바일 기기를 사용하여 실시간으로 작업할 대상을 3D 모델로 확인하는 것은 현장 작업에 큰 도움을 줄 수 있다.

최근 국내 조선소에서는 설계/생산 환경에서 조선 CAD 모델이 아닌 경량 3D 모델을 적용하려는 노력이 이루어지고 있다. 경량 3D 모델에는 여러 종류가 있는데[1,2], 국내 조선소에서는 주로 JT 파일이 적용되고 있다. JT는 Siemens사에서 개발된 포맷으로 폴리곤 메쉬 또는 B-Rep으로 형상을 표현하고, 임의 메타데이터, 데이터 근사화, 압축을 지원한다.

본 논문에서는 경량 3D 모델인 JT 파일로 된 선박 블록 크기의 대용량 모델을 안드로이드 단말기에서 가시화할 수 있는 요소 기술을 개발하고, 프로토타입 시스템 개발을 통해 기반 기술의 성능을 검증하는 사례를 소개한다.

2. 관련 연구 및 연구 범위

Yang 등[3]은 모바일 기기에서의 가시화를 위한 실시간 삼각망 생성 알고리즘을 제안하였다. 점진적 경계기반 들로네 삼각화(Incremental constrained Delaunay triangulation) 알고리즘에 기반하여 가시화에 필요한 최소한의 삼각망을 생성하고, 결과를 저장하기 위한 새로운 데이터 구조를 소개하였다. Song 등[4]은 인터넷 환경에서의 가시화를 위한 새로운 경량 모델 구조를 제안하였다. 여러 종류의 상용 캐드 파일을 제안한 경량 모델 파일로 변환하고, 웹에서 가시화하는 뷰어를 개발하였다. 특히, 수치 검증을 위해 삼각망뿐만 아니라 에지 정보를 포함하는 경량 모델 포맷을 제안하였다. 또한, JT와 같은 상용 경량 모델 포맷이 토폴로지가 없는 삼각망 데이터와 토폴로지가 있는 B-Rep 데이터를 다른 구조로 표현하는 것에 반해, Song 등은 하나의 구조에 토폴로지와 삼각망을 같이 표현하였다.

Lee 등[5]은 모바일 기기를 이용하여 항만국통제(Port State Control)에서 요구되는 선박 설비 및 기기에 대한 점검을 하는 시스템을 소개하였다. 설비 인식을 위해 검사 대상에 RFID 태그를 부착하고, 검사원이 이동하면서 탐지된 설비를 검사하는 시스템을 개발하였다. 모바일 기기에 텍스트로 된 검사 항목과 3D 모델이 보여지는데, 3D 모델 가시화에 대한 상세 설명은 나와 있지 않다. Lee 등[6]은 모바일 기기를 이용한 선박내 현장 모니터링 시스템을 제안하였다. 선내 이상 여부 발생시 현장 작업자가 전문가의 도움을 쉽게 받을 수 있도록 현장의 영상을 원격지의 전문가에서 실시간으로 전송하여 효율적인 유지보수가 가능함을 보였다.

모바일 기기에서의 가시화에 적용할 수 있는 제품으로는 JT-Java[7], Cortona3D[8]가 있다. JT-Java는 공개 소프트웨어로 JT 파일을 해석하는 자체 Parser를 가지고 있고, 렌더링에는 jReality[9]를 이용한다. JT-Java를 테스트한 결과, Windows 환경에서는 만족할 만한 성능을 나타내었으나 안드로이드 환경에서는 느린 파싱 속도로 인해 현업에 적용할 수 없는 수준이었다. Cortona3D는 JT 파일을 VRML 파일로 변환한 후, 안드로이드에서 VRML 파일을 가시화하는 방식으로 되어 있다. VRML 파일은 큰 파일 크기로 인해 모바일 환경에서는 부적합하다[4]. 이로 인해 Cortona3D는 선박의 블록 단위의 대용량 모델을 가시화하는데 많은 시간이 걸려서 현업에 적용할 수 없었다. 본 연구에서는 기존 시스템의 단점을 보완하기 위해, 기존 시스템과는 다른 방식을 채택하여 시스템을 개발하였다. 기존 제품과 본 연구에서 개발된 시스템과의 성능 비교는 4장에 기술되어 있다.

모바일 단말기에서 3D 모델을 가시화하기 위해서는 다음과 같은 기술이 필요하다: i) 선박 CAD 모델을 경량 3D 모델로 변환하는 기술; ii) 경량 3D 모델에 포함된 PMI(Product Manufacturing Information) 정보를 추출하는 기술; iii) 모바일 서버에서 경량 모델을 생성/관리/전송하는 기술; iv) 모바일 기기에서 경량 3D 모델을 가시화하는 기술; v) 모바일 기기에서 경량 3D 모델을 검색하는 기술; vi) 모바일 기기에서 PMI로 표현된 생산 정보를 가시화하는 기술.

본 논문에서는 상기 기술 중에서 i, v을 제외한 나머지의 구현에 대해 다룬다. 본 논문에서는 상세하게 다루어지지 않지만, 선박 CAD 모델에서 JT 파일을 생성하는 방법은 다음과 같다. Tribon/

AM에서 Aveva Review 파일 포맷으로 모델 데이터를 추출한다. Aveva Review 파일(REV) 포맷은 형상을 10개의 프리미티브 표현와 1개의 경계 표현으로 정의한다. 경계 표현 방법은 토폴로지 없이 경계 곡선만으로 각 Face를 정의한다. REV 파일을 파싱하여 Siemens사의JT OpenToolkit[10]의 API를 이용하여 JT 파일을 생성한다. JT OpenToolkit API는 폴리곤 집합, 기하 프리미티브, 또는 B-Rep 모델의 3가지 형태로 모델을 정의할 수 있다. 작은 어셈블리 수준의 모델에서는 어느 형태로 정의하더라도 가시화 성능에 차이가 없으나, 대용량 모델 가시화를 위해서는 폴리곤 집합으로 모델을 정의해야 한다. 예를 들어, 32 Bit Windows 환경에서 기하 프리미티브 또는 B-Rep 모델로 정의한 JT 파일은 10개 이상의 선박 블록을 가시화할 수 없으나, 폴리곤 집합으로 정의한 JT 파일은 같은 모델을 가시화할 수 있다. 대용량 모델 가시화는 부분적으로 JT OpenToolkit API의 구조와도 관련이 있다.

Song 등[4]의 연구에서와 같이 ACIS의 MESH-MANAGER로 만들어진 Facet으로 경량 모델의 형상을 정의하고, 면을 선택하는 등의 목적을 위해서는 명시적으로 경계 곡선을 정의해야 한다. 그러나 JT OpenToolkit API로 JT 파일을 만들 때는 Facet을 입력으로 사용하는 대신, Planar Face의 내외부 경계 곡선을 입력으로 사용하기 때문에 명시적으로 경계 곡선을 정의하지 않는다. Facet은 뷰어에서 가시화 할 때 생성되고, JT 파일 자체는 Facet 정보를 포함하지 않는다.

3. 시스템 구현

3.1 구현 개요 및 시스템 구조

이 연구에서는 안드로이드 기반의 모바일 단말기에서 대용량 JT 파일을 가시화할 수 있는 기반 기술을 개발되었다. 특정 목적의 시스템이 개발된 것이 아니라, 향후 특정 목적의 시스템을 개발할 수 있는 요소 기술이 개발되었고, 프로토타입 시스템 구현을 통해 요소 기술을 검증하였다. 아래부터는 요소 기술과 프로토타입 시스템을 명시적으로 구분하지 않고 기술한다. 곧, 문맥에 따라 시스템이라고 표현했더라도 프로토타입 시스템으로 개발된 요소 기술을 의미할 수 있다.

개발된 시스템은 서버와 클라이언트로 나누어진다. 서버의 기능은: i) JT 파일을 읽고; ii) 읽은 내용을 클라이언트서 가시화할 수 있는 가시화 데이터(ILDD: Intermediate Lightweight Display Data)로 변환하고; iii) ILDD를 클라이언트에 전송하는 것이다. 클라이언트의 기능은: i) 서버에 가시화할 JT 파일의 ILDD를 요청하고; ii) 전송 받은 ILDD를 가시화하는 것이다.

서버와 클라이언트로 나누어진 이유는 안드로이드 OS에서 실행되는 JT 파일 파서 라이브러리가 존재하지 않기 때문이다. JT 파일을 읽기 위해 사용하는 JT Open Toolkit은 Windows 환경에서만 사용할 수 있다. 이러한 이유로, 서버는 Windows 기반으로 클라이언트인 모바일 JT Viewer는 안드로이드 기반에서 개발되었다. 본 논문에서 클라이언트는 모바일 JT Viewer를 의미한다. Fig. 1은 개발된 전체 시스템의 구조이고, Table 1은 서버 및 클라이언트의 개발 환경이다.

Windows 기반의 서버에서는 JT Open Toolkit으로 JT 파일을 파싱하여 ILDD를 생성하고, Zlib[11]을 사용하여 ILDD를 압축한 다음, gSoap[12] 기반의 웹서비스(Web Service)를 통해 클라이언트에 압축된 ILDD를 전송한다. 안드로이드 기반의 클라이언트에서는 gSoap 기반의 웹서비스를 통해 서버에 ILDD를 요청하고, 전송받은 ILDD의 압축을 푼 다음, OpenGL ES[13,14]을 이용하여 ILDD을 가

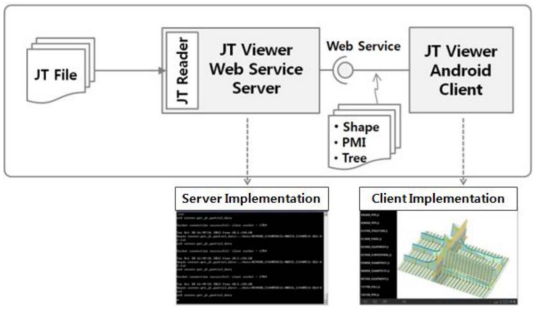


Fig. 1 Overall system architecture

Table 1 Implementation environment

	Server	Client
OS	Windows	Android
Library used	· JT Open Toolkit · Zlib · gSoap	· OpenGL ES · Zlib · gSoap
Communication	Web Service	Web Service

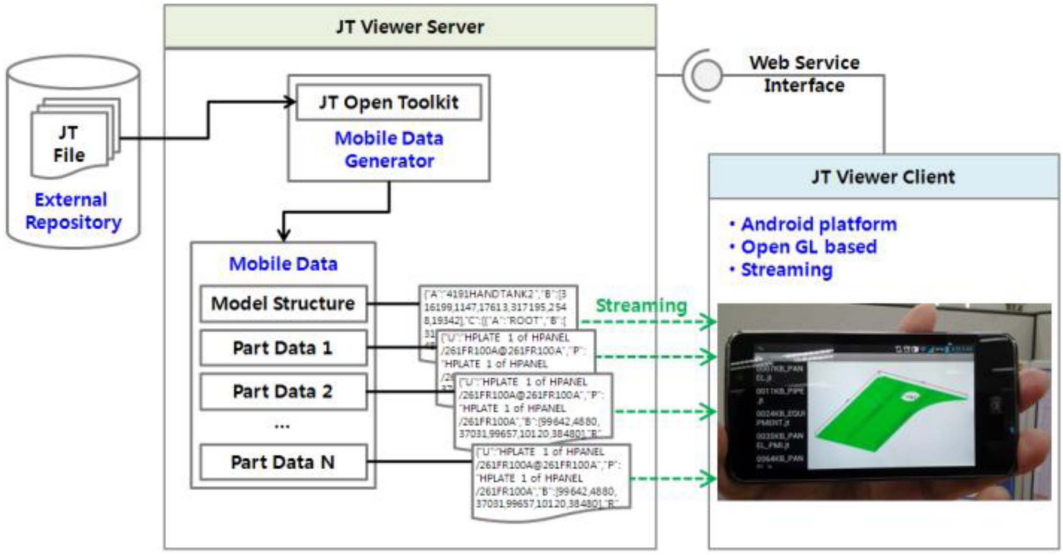


Fig. 2 System architecture in the viewpoint of ILDD

시화한다. 서버는 Windows Console 프로그램으로, 클라이언트는 안드로이드 네이티브 앱(Native app)으로 개발되었다. 클라이언트가 안드로이드 네이티브 앱으로 개발되었기 때문에 기종에 관계없이 안드로이드 OS를 사용하는 모든 모바일 기기에서 실행 가능하다.

Fig. 2는 전체 시스템에서 기술적인 요소를 강조한 것이다. 서버는 서버에 독립적인 저장소에서 JT 파일을 가져온다. 현재 구현된 시스템은 네트워크 상의 특정 디렉토리에서 JT 파일을 가져온다. 향후 PLM 시스템의 저장소(Repository)에서 JT 파일을 가져오기 위해서는 별도의 인터페이스(Connector)가 개발되어야 한다.

서버에서 클라이언트로 전송되는 ILDD는 형상 정보, PMI 정보, 모델 트리 정보를 포함한다. PMI 정보는 치수, 이름, 가공 속성으로 이루어진다. 서버에서는 1개의 JT 파일에 대해 생성된 ILDD는 여러 개의 파일로 이루어진다. 예를 들어, 1개 JT 파일에 있는 모델이 N개의 파트로 구성되어 있을 경우, 모델 트리 구조에 대한 1개의 ILDD 파일, N개의 파트 각각에 대한 N개의 ILDD 파일, 그리고 ILDD 파일 목록에 대한 1개의 ILDD 파일이 생성된다. 상기에서 파트는, 선체의 경우 플레이트(Plate)나 보강재(Stiffener) 단위의 것을 의미하고, 배관의 경우 직관(Straight Pipe)나 곡관(Elbow) 단위의 것을 의미한다.

ILDD는 서버에서 클라이언트로 웹서비스를 통해 스트리밍된다. 스트리밍된 ILDD는 클라이언트에서 OpenGL ES를 이용하여 가시화된다. 스트리밍된 데이터는 클라이언트에 파일 형태로 저장되지 않기 때문에 클라이언트가 종료되면 클라이언트에 가시화 모델의 흔적이 남지 않는다. 따라서, 모바일 기기에서 중요한 문제 중 하나인 보안 문제가 최소화된다.

3.2 ILDD(Intermediate Lightweight Display Data) 정의

ILDD는 경량모델 가시화 데이터의 네트워크 전송을 위해서 본 연구에서 정의된 것으로, 폴리곤으로 표현된 JT 모델을 또 다른 형식으로 표현한 것이다. JT 파일에서 형상 정보는 폴리곤이나 B-Rep으로 표현될 수 있다.

N개의 파트로 구성된 1개의 JT 파일에 대해, 서버에서는 3종류, N + 2개의 ILDD 파일이 생성된다. 서버에서 ILDD를 여러 개의 파일로 나눈 이유는 2가지이다.

첫째, 본 논문에서는 ILDD를 JSON[15] 포맷으로 표현하였는데, 안드로이드 환경의 JSON 라이브러리는 1 Mbytes 이상의 JSON 포맷의 파일을 파싱할 수 없다. 곧, 서버에서 클라이언트로 전송한 ILDD가 1 Mbytes 이상인 경우 클라이언트에서 이것을 파싱할 수 없기 때문에, 이 문제를 회피하기

위해 평균 10 Kbytes 정도의 크기로 쪼개진 ILDD 파일을 생성한다.

둘째, 클라이언트에서의 점진적 가시화를 위해서이다. 안드로이드 기기의 낮은 가시화 성능으로 인해 큰 모델의 경우 30~60초 정도의 가시화 시간이 소요된다. 곧, 점진적 가시화가 가능하지 않으면 선체 블록 1개와 같은 크기의 모델은 30초 이상 응답이 없다가 한번에 모델이 보이게 된다. 이러한 늦은 응답성은 사용자들에게 시스템 기능에 대한 의심을 초래하고 사용하기에 불편하다고 생각하게끔 만든다. 점진적 가시화는 큰 모델일 경우에도 클라이언트에서 멈춤 현상 없이 가시화를 가능하게 함으로써 사용자 불편을 줄인다.

3종류의 ILDD 파일에 대한 설명은 다음과 같다. 첫째, 파일 목록 ILDD 파일. 하나의 JT 파일이 여러 개의 ILDD 파일로 나누어져서 클라이언트로 전송되기 때문에 전송될 파일 목록이 필요하다. 전송 파일 목록에 대한 ILDD 파일의 구조는 Fig. 3과 같다. 클라이언트는 서버로부터 최초로 전송 파일 목록에 대한 ILDD 파일을 전달받고, 이를 해석한 후 목록에 있는 파일을 하나씩 서버에게 전송 요청한다.

```
{
    "F":
    [
        filename, string
        filename, string
        ...
    ]
}
```

Fig. 3 ILDD for a file list to transmit

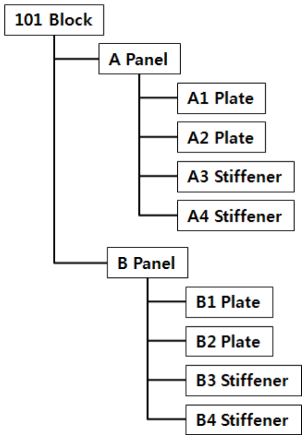


Fig. 4 Model hierarchy of a hull block in a JT file

둘째, 계층 구조 ILDD 파일. 하나의 JT 파일에 있는 모델은 계층 구조를 가지는데, 이 계층 구조에 대한 정보를 표현한다. 선체 모델을 예로 들면, Fig. 4와 같이 Block-Panel-Plate 순의 계층 구조가 부재의 이름과 함께 표현된다. 1개의 JT 파일이 여러 개의 ILDD 파일로 나누어지기 때문에 계층 구조를 담고 있는 파일이 따로 필요하다. Fig. 5는 계층 구조에 대한 ILDD 파일의 구조를 보여준다. 특

```
{
    "U":fullname, string
    "A":assembly name, string
    "B":bounding box, array of six floating numbers
    "M":
    [
        {
            "P":
            [
                PMI data
            ]
        }
    ],
    "C":
    [
        {
            "U": fullname, string
            "A": assembly name, string
            "B": bounding box, array of six floating numbers
            "G":
            [
            ],
            "M":
            [
            ]
        }
    ]
}
```

Fig. 5 ILDD for model hierarchy

```
{
    "U":fullname, string
    "A":assembly name, string
    "B":bounding box, array of six floating numbers
    "M":
    [
        {
            "P":
            [
                PMI data
            ]
        }
    ],
    "C":
    [
        {
            "U": fullname, string
            "A": assembly name, string
            "B": bounding box, array of six floating numbers
            "G":
            [
            ],
            "M":
            [
            ]
        }
    ]
}
```

Fig. 6 ILDD for geometry representation

히, PMI 정보가 파트가 아닌 어셈블리에 속해 있을 경우, 계층 구조 파일에 PMI 정보를 포함시킨다.

셋째, 형상 정보 ILDD 파일. 가시화될 형상에 대한 기하 정보와 PMI 정보를 담고 있는 파일이다. Fig. 6은 형상 정보에 대한 ILDD 파일의 구조이다. 기하 정보는 정점(Vertex)와 법선 벡터(Normal Vector)로 이루어진 Triangle Strip 구조로 표현된다. ILDD에 기하 정보를 Triangle Strip 구조로 표현하는 것의 장점은 클라이언트에서 OpenGL ES로 가시화할 때 추가적인 데이터 처리없이 바로 OpenGL 가시화 함수의 입력으로 전달할 수 있다는 점이다. 형상 정보에 대한 ILDD 파일은 이외에도 경계상자, 색깔에 대한 정보를 포함한다. ILDD 파일의 단위는 파트인데, 선체 모델의 경우 플레이트(Plate) 또는 보강재(Stiffener)의 단위이기 때문에 Fig. 8과 같은 간단한 부재의 경우, 형상 정보에 대한 ILDD 파일은 3개(Plate 1개, Stiffener 2개)로 이루어진다.

3.3 서버 구현

서버는 Windows 기반의 Console 응용 프로그램으로 구현되었다. 서버는 클라이언트에서 요청이 오면 JT 파일 목록을 클라이언트로 전송한다. 클라이언트에서 특정 JT 파일을 요청하면, JT Open Toolkit 라이브러리 함수를 이용하여 요청 받은 JT 파일을 파싱하여 3 가지의 정보를 추출한다: i) 계층 구조; ii) 파트 단위 형상 정보; iii) PMI 정보. 추출된 정보는 JSON 포맷의 ILDD 파일로 생성되어 지정된 디렉토리에 저장된다. 클라이언트에서 ILDD 파일을 요청 받을 때마다 하나씩 전송한다. 여러 명의 사용자가 같은 JT 파일에 대한 정보를 요청할 때, 매번 새로이 ILDD 파일을 만들 필요는 없다. A 사용자의 요청으로 생성된 ILDD 파일이 서버에 존재할 경우, B 사용자의 요청에 대해서는 다시 ILDD 파일을 만드는 대신, 이미 생성된 파일을 전달한다. 이 때, 같은 이름을 가진 JT 파일의 모델 변경에 따른 JT 파일과 ILDD 파일의 불일치를 막기 위해 JT 파일과 ILDD 파일에 파일 생성 이력을 기록할 수 있다.

3.4 클라이언트 구현

클라이언트는 안드로이드 앱으로 구현되었다. 현재 구현된 클라이언트는 단말기의 기종이나 안드로이드 버전에 관계없이 동작한다. 즉, 안드로이드 스마트폰이나 안드로이드 태블릿에서 모두 동작한다. 이는 클라이언트가 기본적인 안드로이드 개발 API[16]만을 사용하여 개발되었기 때문이다.

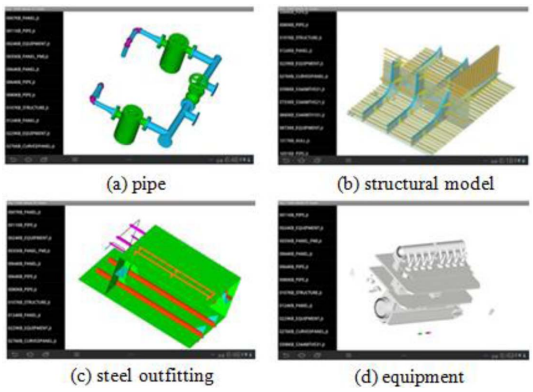


Fig. 7 An example of the ship model displayed in an Android tablet

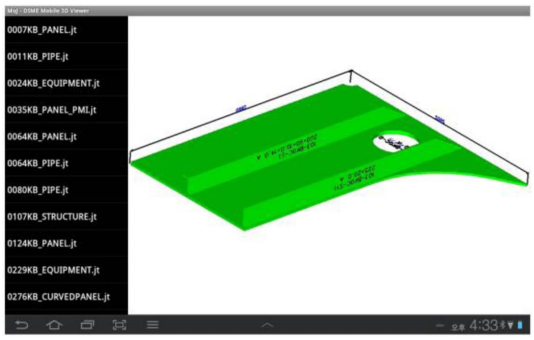


Fig. 8 PMI display

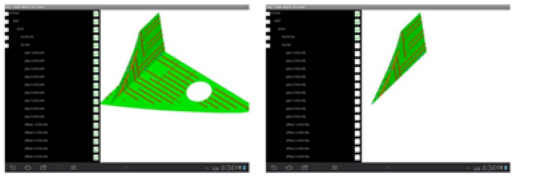


Fig. 9 Selective display

클라이언트 화면의 왼쪽에 서버에 있는 JT 파일이 표시되고, 특정 JT 파일을 선택하면 서버로부터 파일 목록, 계층 구조, 형상 정보의 순으로 JSON 포맷의 ILDD 파일이 전송된다. 먼저, 계층 구조 ILDD 파일을 파싱하여, 모델 트리 구조를 생성한다. 생성된 모델 트리의 각 노드에 형상 정보 ILDD 데이터를 매핑시킨다. 매핑된 형상 정보 ILDD 데이터는 OpenGL ES를 이용하여 가시화된다. Fig.

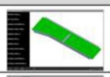
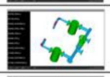
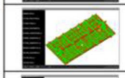
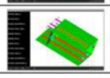
Model	Size (KB)	Time (Sec)	Model	Size (KB)	Time (Sec)
	7	0.35		124	15.44
	11	0.63		229	20.10
	64	3.24		1317	61.42
	107	5.59		1691	143.60

Fig. 10 Display time of various ship assemblies

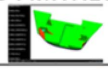
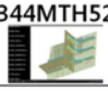
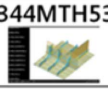
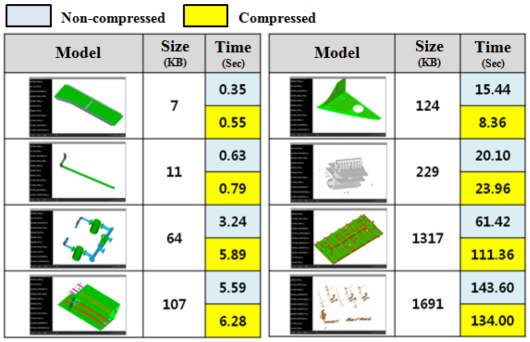
JT File	S Viewer	DSME Viewer
	Galaxy S3	Galaxy Tab
5344MTH101 	370.72 sec	47.04 sec
5344MTH521 	199.80 sec	38.13 sec
5344MTH531 	238.04 sec	20.44 sec

Fig. 11 Performance comparison



Model	Size (KB)	Time (Sec)	Model	Size (KB)	Time (Sec)
	7	0.35 0.55		124	15.44 8.36
	11	0.63 0.79		229	20.10 23.96
	64	3.24 5.89		1317	61.42 111.36
	107	5.59 6.28		1691	143.60 134.00

Fig. 12 Performance comparison between the compressed file and the normal file during transmission

7은 Galaxy Tab에서 가시화된 선박 모델의 예이다. 각 모델을 가시화하는데 걸리는 시간은 Fig. 10과 Fig. 11에 있다.

현재까지 개발된 모바일 기기에서 JT 파일을 가시화할 수 시스템은 많지 않다. 이는 기술적인 어려움이라기 보다는 현장에서의 필요성 때문이다. 최근에서야 안드로이드 단말기의 대중화와 더불어 생산 현장에서 도면을 보조하여 3D 모델을 확인하려는 시도가 이루어지고 있다. 국내에서는 아직 모바일 기기에서의 JT 파일 가시화 시스템이 개발된 사례가 없고, 해외에 몇몇 개발된 사례가 있으나 아직까지 PMI 정보를 가시화한 시스템은 없다. 이 역시 기술적인 어려움이라기 보다는 현장에서의 필요성 인식 차이에 따른 결과로 보인다. 생산 현장, 특히 조선소에서는 3D 모델의 형상 뿐 만 아니라 부재의 이름과 치수가 중요하다. 본 연구에서 개발된 클라이언트는 JT 파일에 포함된 PMI 정보를 가시화할 수 있다. Fig. 8은 선체 부재에서 PMI 정보로 표현된 부재의 이름과 치수를 안드로이드 기기에서 가시화한 예이다.

계층 구조 ILDD로부터 생성된 모델 트리 구조가 클라이언트의 왼쪽 화면에 표시된다. 모델 트리에서 일부 노드를 선택함으로써 선택된 모델만 가시화할 수 있다. Fig. 9는 모델 트리에서 부분 선택을 통한 Hide/Show 기능의 예를 보여준다.

4. 성능평가

스마트폰이나 태블릿은 PC에 비해 낮은 성능의 그래픽 하드웨어를 가진다. PC에서는 수 초 만에 가시화되는 모델이 모바일 기기에서는 수 십 초 만에 가시화될 수도 있다. 모바일 기기에서의 3D 모델 가시화의 경우, 소요 시간과 가시화할 수 있는 모델의 최대 크기가 가장 중요한 성능 평가 항목이다. Fig. 10은 주요한 종류의 선박 모델에 대해 본 연구에서 개발된 클라이언트의 가시화 시간을 보여준다. Fig. 10에서 크기는 JT 파일의 크기를 의미한다.

본 연구에서 개발된 클라이언트의 성능을 해외 S사에서 개발 중인 Viewer와 비교하였다. 비교 대상은 VLCC의 3개 블록의 선체 모델이다. Fig. 11은 2012년 10월 31일 기준으로 S사에서 개발 중인 Viewer와 본 논문에서 개발된 Viewer의 가시화 시간 결과를 비교한 것이다. 본 연구에서 개발된 Viewer가 5~12배 정도 빠른 것으로 나타났다. S사의 Viewer가 더 좋은 하드웨어 성능을 가진 단말기에서 테스트되었다는 것을 고려하면, 실제 성능 차이는 더 커질 것이다.

본 연구에서 개발된 시스템이 서버/클라이언트로 구성되어 있고, 서버에서 가시화 데이터를 클라이언트로 전송하기 때문에 전송 속도가 중요하다. 본 연구에서는 전송 속도를 높이기 위해, 서

버에서 클라이언트로 압축된 ILDD 파일을 전송하는 것을 테스트하였다. ILDD 파일 압축은 공개용 압축 라이브러리인 Zlib을 사용하였다. ILDD 파일을 압축했을 때와 압축하지 않았을 때의, 전송 시간을 포함한 전체 가시화 시간을 비교한 결과가 Fig. 12에 있다. 이 결과는 ILDD 압축이 항상 좋지는 않다는 것을 보여준다. 이는 ILDD 파일의 개수가 많을 경우 서버에서의 파일 압축 및 클라이언트에서의 파일 압축 풀기에 걸리는 시간이 압축하지 않은 파일을 그냥 전송하는 시간보다 많이 걸릴 수 있기 때문으로 추정된다. 결론적으로, 파일이 클 때 전송이 조금 빨라지나, 모바일 기기에서는 큰 모델을 가시화할 수 없기 때문에 전송시의 압축이 실제로는 도움이 되지 않는다.

5. 결 론

경량 3D 모델 포맷 중의 하나인 JT 파일을 안드로이드 기반의 모바일 기기에서 가시화할 수 있는 라이브러리를 개발하였다. 안드로이드에서 사용 가능한 빠른 JT 라이브러리가 없고, 기존 상용 제품이 선박 블록 단위의 대용량 JT 파일을 느리게 가시화하는 문제를 해결하기 위한 새로운 방법을 소개하였다.

Windows 기반의 서버에서 파싱한 JT 파일 정보를 본 연구에서 자체적으로 정의한 가시화 정보 포맷으로 변환하여 클라이언트에 전송하고, 클라이언트에서 가시화하였다. 클라이언트에서의 빠른 파싱을 위해 가시화 정보를 JSON 포맷의 새로운 형식으로 표현하였고, 점진적인 가시화를 위해 파트 단위의 작은 파일로 나누어서 전송하였다. 특히, JT 파일에 있는 PMI 정보의 가시화도 가능하도록 개발되었다. 현재 개발 중인 타사의 상용 제품과의 성능 비교 결과, 본 연구에서 개발된 라이브러리가 더 큰 모델을 더 빠르게 가시화할 수 있었다.

본 연구에서 개발된 라이브러리를 이용하여 3D 스풀(Spool) 뷰어, 3D DAP(Detailed Assembly Procedure) 뷰어, 자재 형상 뷰어와 같은 현장용 응용 프로그램을 개발할 수 있다. 자재 형상 뷰어는 자재코드를 입력하여 현장에서 바로 3D 모델 형상을 확인할 수 있는 시스템으로 특히 해양 공사에 유용하게 사용될 수 있다. 향후, PLM 시스템과 연동하여 JT 파일을 가져오는 부분에 대한 추가적인 개발이 필요하다.

References

1. Mun, D.H., Song, I.H. and Yang, S.W., 2011, Lightweight 3D CAD Data Formats. *CAD/CAM Review*, 17(3), pp.18-21.
2. Cheon, S.U., Park, K.P. and Suh, H.W., 2011, Lightweight 3D Model Technology for 3D Based Collaboration and Simulation Applications. *Bulletin of the Society of Naval Architects of Korea*, 48(3), pp.55-60.
3. Yang, S.W., Choi, Y. and Lee, H.C., 2009, CAD Data Visualization on Mobile Devices Using Sequential Constrained Delaunay Triangulation. *Computer-Aided Design*, 41(5), pp.375-384.
4. Song, I.H. and Chung, S.C., 2009, Data Format and Browser of Lightweight CAD Files for Dimensional Verification Over the Internet. *Journal of Mechanical Science and Technology*, 23, pp.1278-1288.
5. Lee, J.M. and Lee, K.H., 2011, Handheld Device Utilization for Smart PSC. *Proceedings of the Annual Spring Meeting, SNAK*, Busan, 2-3 June, pp.1279-1283.
6. Lee, Y. and Ahn, Y.H., 2012, Design and Implementation of a Field Monitoring System in a Ship. *Proceedings of the Annual Autumn Meeting, SNAK*, Changwon, 15-16 November, pp.168-171.
7. jt-java, 2013, http://code.google.com/p/jt-java.
8. Cortona3D, 2013, ttp://www.cortona3d.com/Products/Viewer/Cortona-3D-Viewer.aspx.
9. jReality, 2013, http://www3.math.tu-erlin.de/jreality.
10. JT Open Toolkit, 2012, JT Open Toolkit Functional Description, Siemens.
11. Zlib, 2013, http://www.zlib.net.
12. gSoap, 2013, http://www.cs.fsu.edu/~engelen/soap.html.
13. OpenGL ES, 2013, http://www.khronos.org/opengles.
14. Blythe, D., Munshi, A. and Leech, J., 2008, OpenGL ES Common/Common-Lite Profile Specification Version 1.1.12, The Khronos Group.
15. JSON, 2013, http://www.json.org.
16. Kim, S.H., 2011, Android Programming, Hanvit Media, Seoul.

천 상 욱

1994년 한국과학기술원 산업공학과 학사
2002년 포항공과대학교 산업공학과 석사
2008년 한국과학기술원 기계공학과 박사
2008년~현재 대우조선해양 중앙연구소 정보기술팀
관심분야: Ship CAD, CAD/CAM

서 흥 원

1985년 인하대학교 조선공학과 학사
1991년 부산대학교 조선공학과 석사
1991년~현재 대우조선해양 중앙연구소 정보기술팀장
관심분야: Ship CAD, PLM, 선박 제품 모델링, 시뮬레이션, 동시공학 설계