Framework for Reconstructing 2D Data Imported from Mobile Devices into 3D Models

WooSung Shin*, JaeEun Min**, WooRi Han*** and YoungSeop Kim†

* Dankook Univ. Deparment of Electronics and Electrical Engineering, ** Dankook Univ. Deparment of Applied Computer Engineering, *** Satreci, † Dankook Univ. Deparment of Electronics and Electrical Engineering

ABSTRACT

The 3D industry is drawing attention for its applications in various markets, including architecture, media, VR/AR, metaverse, immersive broadcast, and etc.. The current feature of the architecture we are introducing is to make 3D models more easily created and modified than conventional ones. Existing methods for generating 3D models mainly obtain values using specialized equipment such as RGB-D cameras and Lidar cameras, through which 3D models are constructed and used. This requires the purchase of equipment and allows the generated 3D model to be verified by the computer. However, our framework allows users to collect data in an easier and cheaper manner using cell phone cameras instead of specialized equipment, and uses 2D data to proceed with 3D modeling on the server and output it to cell phone application screens. This gives users a more accessible environment. In addition, in the 3D modeling process, object classification is attempted through deep learning without user intervention, and mesh and texture suitable for the object can be applied to obtain a lively 3D model. It also allows users to modify mesh and texture through requests, allowing them to obtain sophisticated 3D models.

Key Words: 3D Reconstruction, Backend, Frontend, AI, Mobile

1. Introduction

Recently, 2D to 3D reconstruction technique is required in a lot of fields, such as architecture, medical, VR/AR, metaverse, immersive broadcast, and etc.. However, specialized equipment, like RGBD camera[1] or Lidar camera[2], are needed to get 3D data, so public has difficulties in getting 3D model. Therefore, we suggest a new framework for reconstructing 3D model from 2D images, which are obtained by mobile devices. This framework consists largely of Backend, Frontend, and Client layer, through which 2D images are reconstructed into 3D.

2. Framework for reconstruction from 2d to 3d

When a user collects and enters new 2D data (scenes) using a cell phone camera or other mobile devices, it is stored in the server DB. The main algorithm then converts scenes to RGB-D set and uses it to create a point cloud data. Then main algorithm classifies data through deep learning in part segmentation. When object classification is completed, mesh and texturing are applied to complete creating 3D model. Completed 3D model data is passed to the DB and App Viewer, which client can check 3D model. If a user wants to change mesh or texture of a completed 3D model, the request is passed to the Backend, and the 3D model data previously stored in the DB is passed to give mesh/texturing inside the main algorithm to generate the modified 3D model according to the request. Similarly, data for the

†E-mail: wangcho777@gmail.com
corresponding 3D model is passed to both DB and client. In this way, various versions of a scene can be managed by the user from the Client Layer, and versions removed by management are also deleted from both the DB and the client’s local repository.

2.1 Backend

The Backend is a part of storing and managing data and performing practical tasks on the functions requested by clients. The Backend of the architecture presented in this paper is largely classified into three parts: DB, main algorithm, and getter/sender.

DB is responsible for storing user-supplied scenes and 3D data that has been completed with the main algorithm.

These data are passed to the main algorithm to perform key functions and can be added or deleted by user’s requests.

The main algorithm is classified into two parts, one that converts the scene to RGB-D and the other that converts the RGB-D to a 3D model. The main algorithm receives scenes from DB to generate 3D models or receives 3D data to modify mesh or texture to return the applied 3D models.

The scene-to-RGB-D set receives user-supplied scenes from the DB, converts them into RGB-D datasets, and forwards them to the RGB-D to 3D model. Previously, the data needed to be provided to convert 3D was primarily images and datasets taken with special equipment rather than simple 2D images, but due to this part, the dataset required by the user becomes simple 2D images taken with regular cameras. Therefore, it makes 3D modeling more accessible to users.

The RGB-D to 3D model is divided into 3D model generation parts and mesh/texturing parts. After performing the algorithm, the 3D object data is backed up to the DB.

The 3D model generation part uses a library provided by Open3D [3]. The library uses the RGB-D set to create the original set as a 3D model through the make-register-refine-integrate fragment process. In make fragment, the sets are used to generate fractions, and in register fragment, the positions are registered between each fragment. Refine purifies the corresponding positional array. In this process, unnecessary overlap is removed and alignments between more refined fractions are obtained. Finally, the integration phase is integrated using the reinforced alignments and fractions obtained earlier.

The mesh/texturing phase is divided into three categories: Part segmentation, Give mesh, and texturing. This step is responsible for object classification and based on deep learning of point cloud data transformed into scenes, and for receiving and modifying 3D models from DB when requested to modify mesh/texturing of 3D models. At the end of the phase, data about the completed 3D model is sent to both DB and sender. The reason why 3D model data is sent to DB is to respond faster when a user requests...
modifications to the 3D model.

In the part segmentation module, object classification in the point cloud is achieved by leveraging an integrated Point Net library. This leads to the separation of point sets by object through deep learning.

When object classification for points is completed in part segmentation, deep learning is used to properly determine which mesh and texturing to that object, resulting in a more delicate and realistic 3D model. Furthermore, when additional requests from users occur, they receive 3D data from the DB and give the mesh/texture appropriate for the request to generate modified 3D models and data. When all processes are performed, 3D data and models are delivered to the sender and DB.

A getter/sender part is a part that sends and receives user-related information through Endpoint on the Frontend. The getter receives user-supplied scenes and user request-related data, and the sender delivers the 3D model that has completed the desired function.

### 2.2 Frontend

In the current architecture, the Frontend is the part that allows the server to respond to user requests and finally deliver data to the client. Therefore, it is the only module that interacts with both client and Backend. To the client, the first time the application is used, it creates a 3D viewer that allows the application to run only once to view the 3D model and forwards it to an html file. It also interacts with the client Backend, which receives the client's scenes and requests and sends the request to the Backend. It then receives the resulting 3D model generated by the algorithm from Backend and forwards it to the client. The Frontend of the current proposed architecture consists of four types: endpoint, 3d_model_sender, 3d viewer sender, and responser.

The endpoint is the part responsible for data delivery between the Backend and the Frontend. It interacts with the client end via 3d_model_sender, responser, and interacts with the getter/setter at the Backend, making it macroscopically responsible for data delivery between the Backend and the client.

3D_model_sender is the part where the 3D model created or modified through the process of the main algorithm is received from the endpoint and delivered to the client.

The first time an application is run on a client, 3d viewer sender creates a viewer that can view the 3D model and forwards it to the client in an html file. The html file is stored in the client's local repository, allowing the 3D viewer to run within the application, even if the client device is not connected to the Internet.

Responser is the part that responds to a request from a client. These requests can be communicated to the Backend through the endpoint to enable the desired process to be performed.

### 2.3 Client

The client is targeted at Android devices to enhance the accessibility of the user, and the user can use the application. Heavy algorithms that require complex processes with respect to client requests can be performed on a server, allowing users to create, modify and verify 3D models more smoothly.

The client is largely divided into two parts: local repository and activity.

The local repository receives and stores the 3d model viewer from the Frontend as an html file. Because it is stored in a local repository, it can be operated without having to access the Internet. In other words, except for the function of interacting with the server, the part that opens the 3D model and uses it will be available without the Internet. The saved html file will be called from Activity to URL.

At the Frontend, 3D model sender also stores 3D models into a local repository and delivers them to the app's viewer. So, user can check the 3D models in local repository even if Internet is not connected.

Activity is the part where the user directly interacts with
the 3D model function. It is stored in the form of an application on the user's Android device. This allows users to directly verify the 3D model stored on that device, allowing users to autonomously manage the version of the model. Furthermore, when we attempt to modify and generate 3D models, we receive requests for them and forward them to the Frontend. Activity has three main modules: Android Web Viewer, request, and input 2D Data.

The application activity internal module, Android Webviewer[4], calls the 3D viewer stored within the Android device to URL.

Requester allows users to easily perform their desired requests via Internet-connected devices. Requests are forwarded to the server's responder. The main requests are the first two types of 3D models that allow users to pass scenes, create new 3D models, and re-output 3D models by setting mesh and texture in the desired direction. This allows us to reset 3D models with mesh and texture set automatically as a result of object classification through deep learning, as well as point clouds (3Data) generated from previous requests. In other words, the user can also obtain the desired form of 3D models, which can increase the user's degree of freedom. An additional request allows the user to pass information about versions stored in the DB that are no longer needed, and remove them from the DB.

Input 2D data allows users to enter 2D data for 3D model generation using a mobile phone camera. These 2D data are sent to the Frontend via request. When creating a new 3D model, it must undergo a single input 2D data process.

3. Conclusion

Our framework focuses on more streamlined equipment, lower costs, and improved accessibility to the public than conventional 2D to 3D reconstruction methods. We believe that future research value is significant in that we can reconstruct 3D models based on 2D data obtained from mobile devices instead of 3D models obtained from existing specialized equipment, and that the obtained 3D models can be viewed directly from the client's mobile devices. We will continue our research focusing on optimization through demonstration tasks so that this framework can be applied to commercialized 3D reconstruction techniques.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP: Ministry of Science, ICT and Future Planning) (NRF-2020R1A2C2009717).

References


Fig. 4. Client Framework.