3D OBJECT RETRIEVAL USING POINT CLOUDS (PCS) AND PC DATASETS

PROJECT REPORT OF BBM480 SENIOR PROJECT

BY

MUSTAFA CANDAN
NİLAY DOĞAN
EMRE YAZICI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF BACHELOR OF SCIENCE
IN
COMPUTER ENGINEERING

JULY 2021

ABSTRACT

3D OBJECT RETRIEVAL USING POINT CLOUDS (PCS) AND PC DATASETS

Candan, Mustafa
Doğan, Nilay
Yazıcı, Emre
Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Ufuk Çelikcan

July 2021, 16 pages

In this report, we presented object retrieval using point clouds for 3d object classification. In the first phase, we built an android app for obtaining point cloud data of 3d objects using Unity AR Foundation. We showed the development and the design of the app, and we demonstrated its performance through tests and experiments. Secondly, we proposed a machine learning system that takes the obtained point cloud and returns what it classified for object classification from the point cloud. Through modeling and studies, we showed that the app can predict the object's class from the obtained point cloud. We believe that this study can contribute to the development of more applicable and high-performance Machine learning and Point cloud detection projects for Unity.

Keywords : Point Cloud (PC), Augmented Reality (AR), Dataset, Machine Learning, Python, Tensorflow, Django, Unity, AR Foundation

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation of the Project

In this project we will be exploring how 3D object classification can be achieved via using Unity AR. Apart from the general methods followed in today's object classifications, we aimed to do this with the Point Cloud, where new products are tried to be introduced. With this project, we aimed to provide the user with the point cloud data of the objects in our daily life and present objects that are close to them by classifying them, and then create and compare similar objects in the virtual environment according to their demands.

1.2 Problem Definition and Contribution

This project is based on extracting Point Cloud data through Unity, and making a classification using this data, since there are more areas to spread while doing such a project, we focused on classification on house furniture. We aimed that the product to be created in this way could contribute to the purpose of creating a branch and an example in the mobile world for people who are interested in house furniture such as furniture companies.

We form our development plan as 5 items on our supervisor’s suggestion. These are as seen below:

1. Obtaining Point Cloud with phone camera using Unity AR Foundation
2. Making object classification according to PC dataset chosen by the resulting Point Cloud
3.
    a. Placing 3D models of similar object alternatives in the class of the recognized object in the image. Providing to the user to compare with alternatives.
    b. Changing the color of the alternative object on screen
    c. Changing the shape of the alternative object on screen
4. Converting Point Cloud of the recognised object into 3D object using the model learned from PC dataset
    a. Modifying created the 3D clone on screen
5. Changing the color of the wall by recognising walls of the room

1.3 The Outline of the Report

The organization of this report is as follows. Chapter 2 presents “Obtaining Point Cloud data of 3D objects using AR Foundation“. Then, Chapter 3 presents “Machine Learning System for object classification”. After that, Chapter 4 presents “Integration of the Machine Learning System and Point Cloud Retrieval System”. In each of these chapters, the modeling and approach of the systems are explained in detail. Chapter 5 summarizes results related to this project.

CHAPTER 2

OBTAINING POINT CLOUD DATA OF 3D OBJECT USING AR FOUNDATION

2.1 Introduction

This chapter is focused on the 1st item in our development plan which is the main part of our project “Obtaining Point Cloud with phone camera using Unity AR Foundation” and also requires planning and designing for what to do in Chapter 4. In this chapter, we will explain the beginning of the design of the app, learning AR Foundation and adaptation time, additional development process, simulation and corrections made as a result of the observation of the data created as a result of the tests.

2.2 Modeling of the App

We started to develop our project by adding the AR Foundation in the new Unity 2020.1.15f project we created. AR Foundation is a cross-platform framework that allows you to build augmented reality experiences once, then build for either Android or iOS devices[1]. After this process is done, we followed the Webinar[2] on Unity AR Foundation to take the first step, and created our basic project.

Thereupon, we added the “Point cloud (Feature Points) Visualizer to start the Point Cloud system. With the application created as a result of the build we received, we are able to see the points around us from the phone. As the next step, we worked on saving the data we visualized. To start this phase, we added a script called “Point Cloud Parser” [3] to our project and aimed to keep the Point Cloud data.

Based on our tests and observations, this script was working as it was designed to keep the data of all the points found. For the feasibility of our own project, we had to impose a restriction on this situation. For this restriction, we decided to narrow the scanning area and only save the data of the points within this area. With the changes we made on Point Cloud Parser, the point cloud save system was only provided for points in a cube that we defined in the application.

At first, this cube was fixed but not efficient in terms of usability, so we added unique UI buttons to axes to change the cube size and position. These buttons were designed to increase or decrease the size and position values of the cube by certain amounts with the functions we defined earlier. Adjusting the cube size and position before scanning is a necessary part. When this adjustment process was done with the button system we designed,

it was observed that it took more time than necessary, so it was not very efficient, so a change had to be made.

We decided to make the adjustment process possible with “pinch” and “drag” movements, and for this we added the “Lean Touch”[4] asset to our project. When we added “Lean Pinch Scale” and “Lean Drag Translate” components to our cube, we were able to perform scaling and translation operations to a certain extent.

Since there was an adjustment to make changes to all axes provided by these components at the same time, it did not provide efficient point cloud data creation. So we have modified these components to use them on the axes we want and added them to the application. In its new form, three unique scale buttons to three axes and translate buttons unique to three axes have been implemented.

As a next step, we dealt with the case of re-saving and visualizing the saved point, as it may cause problems. Our visualization system at this stage was provided by instantiating cube objects to the location of the detected points. When the same point was detected, more cubes were created than it should have, so this could cause performance problems when receiving and visualizing point clouds. As a solution, we worked on two designs that we planned. The first design was to check whether the captured points were found and saved by the system before, and to decide whether the point was a duplicate.

This design was not preferable in terms of performance, as it created a large number of loops, as the system was provided by circulating and checking all points at each point discovery. With the scripts we added to the cubes we visualized, which is the second design, we have found a solution to the duplicate point problem by destroying the cube created later as time from nested cubes.

As a result of these steps, the only thing left to do was to print out the point cloud data we obtained to use it on the machine learning side.

Before connecting the two systems, we made a design in the form of sending the data we receive as a string in order to be able to perform manual operations. This process was provided by taking the positions of the children from our parent object containing our points in order, converting them to an array string and assigning this string to the input field we created. After this process, which is operated with a button after the scanning process, we are able to receive our point cloud data. With this, Chapter 2 is concluded.

2.3 Simulations and Test Results

Tests we made for this chapter includes trying different types of objects and different ranges for scanning. With these tests we see that objects with pattern or textures are more likely to have more points found rather than solid colored objects. This works for the fifth item in our plan, which is changing wall color. Generally walls do not have patterns if they are just paints and this makes it hard to find walls to change color. Because of this problem we decided to take out this part from our project.

CHAPTER 3

MACHINE LEARNING SYSTEM FOR OBJECT CLASSIFICATION

3.1 Introduction

Our project is about point cloud classification so we needed to implement a machine learning algorithm that takes point cloud data and classifies the given data to predetermined classes.First thing we did was doing research. We found some similar projects to be our reference project. We decided to use Point cloud classification with PointNet[5] as our reference project since it is the best project among similar projects.

3.2 Modeling of the Machine Learning System

We decided to use Python as our programming language for machine learning. Reason for this is, Python has a really simple syntax and is widely used in the machine learning area. As it is a widely used language it has a number of libraries for machine learning applications such as Sklearn, Tensorflow etc. We chose to use the Tensorflow library. Usually this library is used for image classification but we saw that a lot of similar projects use this library while we were doing research.

As we are doing research we have found a number of datasets for point cloud classification such as ModelNet10, ModelNet40, Semantic3D etc. In the end we decided to use the ModelNet10[6] dataset. This dataset consists of house furniture which is perfect for our project which focuses on house furniture. This dataset has a total of 10 classes of objects. We used 5 of those classes for our project. Reason for this is that those classes are easy to access for testing our model in real life usage. We also tried to split our dataset into more specific classes for more accurate suggestions but our dataset did not have enough data for this kind of speciation so we forfeit that idea.

As the machine learning model we created a Convolutional Neural Network(CNN) which has a total of 65 layers and 748,334 parameters. 742,254 of those parameters are trainable. Our dataset is already splitted into train and test sets but we needed to normalize our data before we start to train our model because our dataset had some models that had absurd scales. We used the Min-Max Normalization algorithm to normalize our dataset. With this we srinked all of our models into a $8m^3$ spaces. The purpose of this process is to speed up our training and prediction speed and increase the prediction accuracy. As we normalized our training data we need to normalize the data that is given for prediction also.

3.3 Simulations and Test Results

We have achieved an accuracy of 98% in our training sessions. We tested our model both in real life scans and point cloud data that are not in the training dataset. In the beginning our model makes predictions with around 50% of accuracy even though our model is supposed to have a 98% of accuracy. When we tried to find the problem, we realized that the position of the point cloud data that was created through the scan is highly effective on the prediction. So we made an adjustment that repositions the scanned data before sending it to the machine learning model. After this adjustment the accuracy of the model returned to what it was supposed to be.

CHAPTER 4

INTEGRATION OF THE MACHINE LEARNING SYSTEM AND POINT CLOUD RETRIEVAL SYSTEM

4.1 Introduction

We decided to create a backend server for the classification process so that our Unity application would need a lot less disk space. For the backend server we chose to use Django Framework[8] which is a Python framework for creating backend servers. Reason for this was the compatibility with our model. Since our model is created by using Python Programming Language[7].

4.2 Modeling of the Server

We did not need to have a complicated backend server. We just needed to process the data given in the body of the HTTP request, make a prediction with it, then return this prediction to the mobile application via HTTP response.So we created a very simple django app that contains a single method which is a POST method named ‘predict’. As the name suggests this method takes an array of coordinates and processes them for the machine learning model to use. After the processing is done it sends the data to the machine learning model and returns the prediction to the mobile application along with the percentages of all of the classes.

4.3 Display of the Prediction

We created two parts for UI and world objects for visibility of the result. In Scan mode the user only sees the cube, scanned points and scan UI panels . In the result mode the user only sees the created objects and result UI panels. When the prediction result returns we check this prediction result and show 5 similar types of objects which can be instantiated in the scene. In the object selection panel we have the option just in case if the prediction is wrong which opens up the class menu. With the user's choice of object, after selecting the object the user can change instantiated objects colors. When we first developed the changing color of result, it was only one color selection, but with the newly created objects we can change three colors which are fabric, wood and metal parts of the objects.

4.4 Simulations and Test Results

When the implementation of the backend is done we hosted our backend on the DigitalOcean[9]. DigitalOcean is a web site that provides hosting service. We hosted our backend on a Ubuntu 20.04 operating system with 2 GB Memory and 50 GB Disk capacity that was located at Frankfurt.

At first we only did the tests based on the prediction that returned from the server. But after a while we realized that this was not enough. So we added a saving system that saves the processed data that is about to send to the machine learning model as a txt file. We took these txt files and sent them to the server through Unity Editor manually after doing some changes to find out the problems with the system and fix those problems and update the server.

CHAPTER 5

CONCLUSIONS

In this project, we created a mobile application that scans an object, turns it into point cloud data, sends it to the server and makes a prediction on what this object is, using CNN machine learning model. We used Unity for implementing the mobile app which scans the object and creates the point cloud model of it, Python and Tensorflow Framework for the machine learning model which classifies the point cloud data given from the mobile app, Django for implementing the backend server which connects the mobile app and the machine learning model and DigitalOcean for hosting the server which allow us to send request to the backend server by keeping the backend server running for 24/7.

While working on this project we learned Unity AR Foundation and improved ourselves on Unity, Python, Tensorflow and Django Frameworks.

REFERENCES

[1] https://developers.google.com/ar/develop/unity-arf#:~:text=AR%20Foundation%20is%20a%20cross,either%20Android%20or%20iOS%20devices.

[2]https://www.youtube.com/watch?v=kvsCvHbDdRs

[3]https://tutorialsforar.com/accessing-and-saving-point-cloud-data-using-unity-and-ar-foundation/

[4] https://assetstore.unity.com/packages/tools/input-management/lean-touch-30111

[5] https://keras.io/examples/vision/pointnet/

[6] http://3dvision.princeton.edu/projects/2014/3DShapeNets/ModelNet10.zip

[7] https://docs.python.org/3/

[8] https://docs.djangoproject.com/en/3.2/

[9] https://www.digitalocean.com

APPENCICES

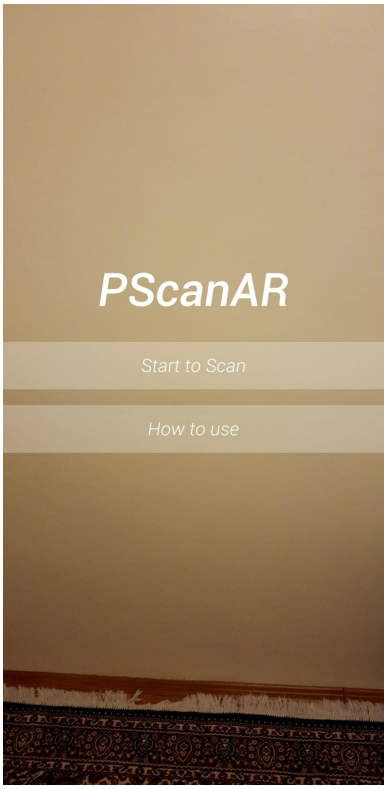

Image 1: Main Menu of the App

Image 2: Before the Start page for instructions

Image 3: Start of the Scan Mode

Image 4: UI Panels for Scan Mode

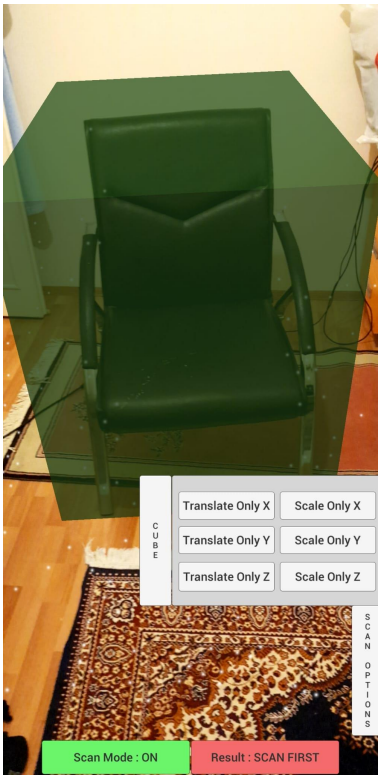

Image 5: After Modifying the Cube

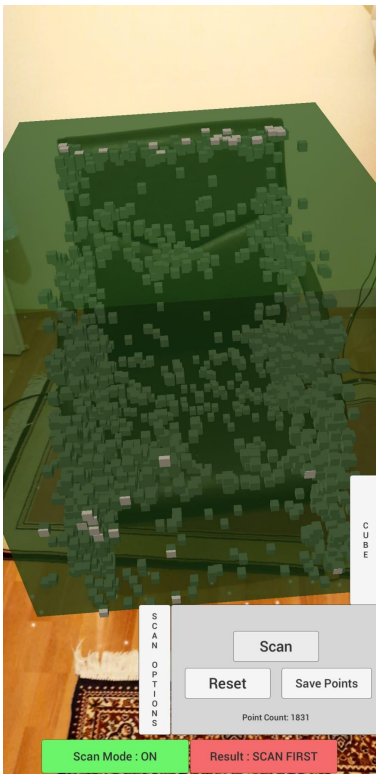

Image 6: After Scan is finished

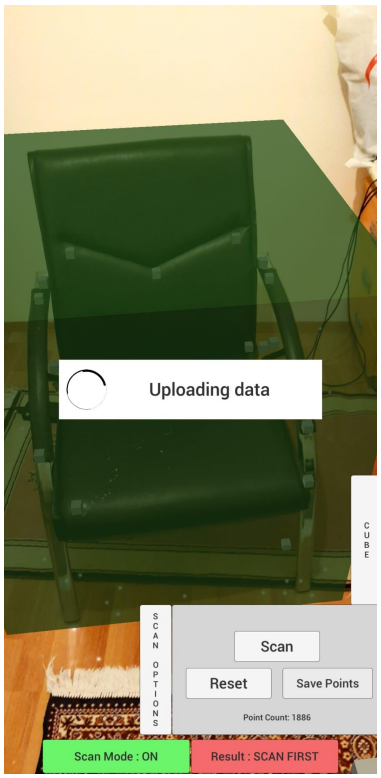

Image 7: Uploading data with Save Points

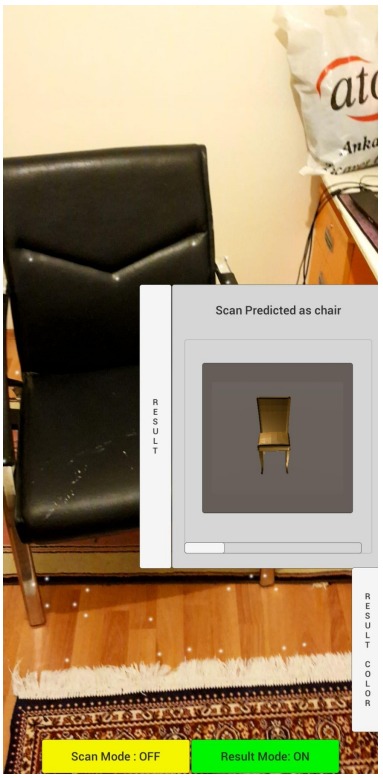

Image 8 Result Mode Predicted Results tab

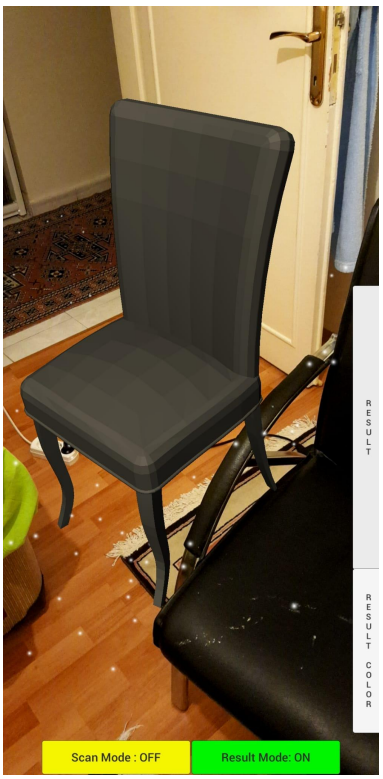

Image 9: Instantiated Object

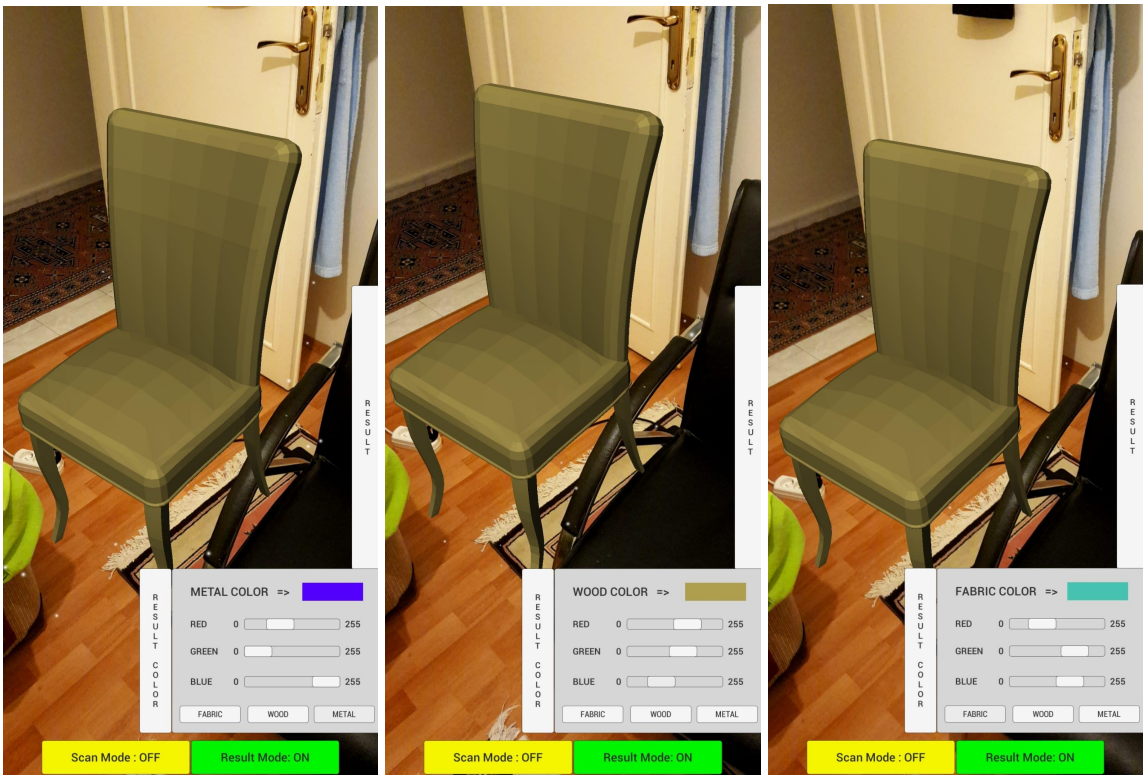

Image 10,11,12: Color change panels for Fabric, Wood and Metal Color