Boston University

Electrical & Computer Engineering

EC463 Senior Design Project

First Semester Report

**Scan it! Pack it!**

Submitted to

Benni Trachtenberg

btt@bu.edu

by

Team 14

Scan it! Pack it!

Team Members

Tristen Liu [tristenl@bu.edu](mailto:tristenl@bu.edu)

Daniellia Sumigar [dsumigar@bu.edu](mailto:dsumigar@bu.edu)

Juan Vecino [jvecino@bu.edu](mailto:jvecino@bu.edu)

Ramy Attie [attiera@bu.edu](mailto:attiera@bu.edu)

Submitted: 12/10/2023

#### **Table of Contents**

Executive Summary 2

1.0 Introduction 3

2.0 Concept Development 3

3.0 System Description 4

4.0 First Semester Progress 7

5.0 Technical Plan 8

6.0 Budget Estimate 8

7.0 Attachments 8

7.1 Appendix 1 – Engineering Requirements 8

7.2 Appendix 2 – Gantt Chart 9

7.3 Appendix 3 – Other Appendices 9

# Executive Summary

Scan it! Pack it!

Team 14

Container and packaging material waste poses issues for resource management and sustainability. Shipping companies, storage industries, and ordinary people alike face the difficult task of finding optimal packaging. To address these issues, Scan It! Pack it! will be a user-friendly mobile application that can automatically detect object and container dimensions and generate an optimal packaging schematic. The application will utilize camera imaging to perform dimension detection, cloud technology for data processing, and machine learning models to identify optimal packaging accurately. It will encourage proper packaging methods to reduce waste and ensure the safety of goods when transported.

# 1 Introduction - Daniellia Sumigar

In 2018, the Environmental Protection Agency (EPA) reported 82.2 million tons of solid waste from containers and packaging in the United States [1]. Packaging containers are necessary to protect various products, including food and medications. They are used especially in shipping and delivery services to safely transport goods from one destination to another. However, the excessive waste from containers only poses further harm to the environment.

Nonetheless, finding the optimal way to pack and ship is a tedious task encountered by all kinds of audiences, ranging from large e-commerce businesses to everyday people. Delivery companies need to maximize cargo in their shipment vehicles to increase efficiency. College students leaving campus for the year need to fit all their belongings in their car trunk or in a fixed-size storage room. Our project aims to provide a solution to address both high-level and low-level situations.

To address this challenge, Scan It! Pack It! is a mobile application that automatically detects container and object dimensions and generates optimal packaging. The application will have two primary functions: (1) automatic dimension detection and (2) packaging optimization. The only hardware requirement for our application will be a phone camera, which will be accessible through mobile devices. For (1), we will leverage Neural Radiance Fields (NeRF), which is a machine learning model for generating a 3D structure or landscape using a set of 2D images. For (2), we will integrate the dimension data into a machine learning optimization algorithm. The algorithm will first arrange the objects in the container to minimize wasted space and then generate the packaging schematic that will be sent back to the user.

Scan It! Pack It! is designed to be accessible and convenient to all audiences. When the packaging schematic guides them, users will be less inclined to purchase unnecessary containers to package their goods. Our application will be a step towards reducing packaging waste.

# 2 Concept Development - Tristen Liu

**2.1 Project Description**

Our client wants us to create an application that can automatically detect dimensions of containers and objects, and output a user-friendly packing schematic. The provided user stories include: placing books on a shelf, fitting items into a U-Haul truck and packing items into a closet. In order to accomplish this, we must split the problem into four main parts: User Application, Dimension Detection, Packing Determination, and Schematic Generation.

**2.2 User Application**

The user application must be intuitive and easy to use. Since we want the application to be accessible for ordinary people, it does not need to deeply involve any engineering methods. From the home screen, the user should be able to choose between creating a new scan or accessing an old scan. Upon selecting the new scan creation, the application will branch into the remaining two main parts. Currently, we are not planning on any extra features in the home screen other than these two.

**2.3 Object Dimension Detection**

Object Dimension Detection is the first step of creating an efficient packing schematic. The objects that need to be scanned include the objects to be stored, as well as the container in which the objects will be stored in. Ideally, these dimensions will all be obtained from a single scan. This scan will be able to be conducted via two methods: LiDAR scanning or pure camera image processing. The LiDAR scan option will be limited to iPhone Pro 12+ devices, due to these devices being the only smartphones with an in-built LiDAR scanner. The camera imaging option will be made available on all devices that have cameras. Two options will be provided due to the Pros and Cons of each option. Although the LiDAR scanner will be severely limited to a few devices, it will be faster and more accurate than the camera imaging option. On the other hand, the camera image processing will be available on any device with a camera but will not always return accurate measurements. In both scanning methods, the user must organize the objects on a flat surface with ample space between each object. After recording a comprehensive view of all the objects and angles, the application should automatically segment each object and retrieve proportions or dimensions. These dimensions will then be pipelined to another algorithm to determine the packing order.

**2.4 Packing and Schematic Generation**

The packing order determination should be done automatically without any further user input. After the user distinguishes between the container and the objects in the scan, the application should input the dimensions into a provided packing algorithm and generate the most efficient packing order possible.

After the packing order is decided, a schematic must be generated for the user. The schematic should be able to be viewed from all angles, and each step of the packing should be able to be chosen to display. Thus, the schematic should be rendered in 3D with scrollable functionality, allowing the user to swipe the screen in order to move the model around. If possible, the colors of the objects should also be retained such that it will be easier for the user to determine which object is next to be placed. Then, there should be two buttons on screen allowing the user to view the next step, or the previous step. The object that was most recently placed may also be highlighted in some way to make it easier to spot for the user.

**2.5 Reasoning and Alternatives**

In our client’s project description, the problem of efficient packing and resource management for ordinary people was explicitly discussed. As a result, we wanted to make this application usable on most smartphone devices rather than limit the usability to a small set of devices. This is why we are implementing the Camera Image Processing option of measuring object dimensions and proportions. Initially, we thought that the LiDAR scanner would be the easiest and most accurate way of pulling object measurements in real time. While this still holds true, using LiDAR without adding support for any other measurement method would severely limit our consumer base to only those with iPhone Pros generation 12+. By implementing image processing for object measurement determination, we are able to expand our consumer base to most devices with a simple Camera requirement.

# 3 System Description - Tristen Liu

**3.1 System Overview**

Scan it! Pack it! will be fully accessible through a single mobile application for iOS and Android devices. The application front-end will be coded using React Javascript, while the back-end will be coded with Python.

**3.2 User-Side Application**

In the User-side application, the user will be able to view past scans as well as create new scans. When creating a new scan, the user will be prompted to record a slow and steady 360 degree view of the container or to choose from a preset of common container sizes. If the user chooses to record a video, the video will be sent to the back-end Machine Learning processes and the determined dimensions will be returned to the user. If the user rejects the dimensions, the user can record the container again in order to try to obtain better results or input the size of the container manually. After the container’s dimensions have been accepted, the user will then be prompted to follow the same procedure for each of the objects that they wish to pack into the container. When the user has successfully scanned all of the desired objects, they will end the scanning process and the container/object dimensions will be sent to the packing algorithm. After waiting up to 30 seconds, the packing schematic will be generated and sent to the user’s screen. The user will have the option to save this schematic and name it for future reference. The 3D packing schematic will be rendered and interacted using Unity.

Saved scans will be stored in a cloud database hosted by AWS and queried using MongoDB. When accessing previous scans, users will be able to modify them by changing the container size or removing/adding objects to the list. After modifying, users will have the option to save as a new scan or update the previous scan.

**3.3 Backend Overview**

The backend of the application will be coded using Python and hosted on an AWS cloud service. It will combine and process the outputs of multiple Machine Learning algorithms. These algorithms will consist of a 3D Object Scanner, an Object Segmenter and a 3D Bin Packing Algorithm. We will use third party open-source Machine Learning applications in order to fulfill these requirements.

**3.4 Scanning Methods**

We will use both NVIDIA’s Instant NGP software as well as LiDAR to generate 3D models. Two methods are necessary in order to diversify our consumer audience. NVIDIA’s Instant NGP model is able to generate 3D models from a video input, while the LiDAR option will utilize the in-built LiDAR scanner on iPhone Pros.

**3.4.1 NVIDIA Instant NGP**

NVIDIA’s Instant NGP Software requires a set of images as well as a transforms.json file in order to generate a Neural Radiance Field (NeRF), a technique that can generate 3D environments using 2D images with Machine Learning, which can then be exported as a 3D mesh model. This transform.json file is a file that assigns specific environment coordinates and camera positioning to each image, allowing the model to know exactly where and how each image fits into the 3D environment. This file can be obtained via two methods. The first method involves underlying data collection as the user records the video around the objects. Images will be directly clipped from the video while the recording is in progress, and each time the image is clipped the environmental data such as camera position, rotation and translation will be recorded. This significantly reduces the processing time of the images and the NeRF model will be generated much quicker. On the other hand, the second method involves utilizing the python scripts available in NVIDIA’s open-source repository. One of these scripts is colmap2nerf.py, which takes a .MOV file as input and outputs a series of images clipped from the video as well as the necessary transform.json file. This script will take a long time to execute however, as it will attempt to manually determine the camera rotation and translation values based solely on the image. Despite the lengthy execution time, this script will be useful for devices that may not have the processing power or resources to generate the transforms file during video execution. Finally when both the image set and the corresponding transform.json file are made available, NVIDIA’s Instant NGP executable will use the files in order to generate a 3D NeRF, which can then be segmented into individual objects.

**3.4.2 LiDAR Scanning**

The LiDAR option for scanning objects will be limited to only iPhone Pros, as those are currently the only device on the market that provide in-built LiDAR functionality. Using Apple’s ARKit API, we can access LiDAR data in order to generate a Point Cloud and display the generating Point Cloud to the user in real-time. This point cloud will then be exported as an .xyz file to a python script, which will automatically segment the floor, reduce noise and cluster the remaining points to find the required objects. Using these clusters, the script will then determine the dimensions of the bounding box of each object and obtain the dimensions of the objects.

**3.5 Object Segmentation**

The Object Segmenter will operate on the NeRF model created by NVIDIA’s instant NGP, and is not required when using LiDAR scanning. Segment Anything 3D (SA3D) is another Machine Learning framework that is able to segment an object from a provided NeRF from a single image view. This algorithm requires user input to select which objects to segment from the NeRF, and will output each object as a new isolated NeRF. By using this framework, we will be able to retrieve multiple objects from a NeRF model, allowing us to mesh that segmented object and determine its proportions. Unfortunately, exact dimensions are not retained when generating a 3D model via NeRF. As a result, if the user decides to utilize camera image processing in order to generate the 3D objects, they will need to be able to scan all of the required items in one scan. This will allow the segmented objects to all have the same proportions, meaning they will still be able to be sorted regardless of if the actual dimensions are wrong. Alternatively, the user can provide one measurement for an object which will then be generalized for all of the segmented objects. These resulting proportions will then be formatted for the 3D Bin Packing algorithm.

**3.6 Bin Packing and Schematic Generation**

The 3D Bin Packing algorithm is the final framework necessary for this application. We will use an open-source 3D Bin Packing algorithm. Using the dimensions of the container and all of the objects found by the previous frameworks, this algorithm will find the optimal packing schematic. The output of this algorithm will be a packer object that holds the size of the container, the sizes of the objects, and the positions of each object within the container in xyz coordinates. The algorithm also provides extra information such as space utilization efficiency, which we can display to the user screen.

Finally, the schematic will need to be parsed and rendered in Unity. Using the provided dimensions and labeled positions of each object, we will create an interactable Unity environment for the user. This environment will be rotatable so that it is visible from all angles, and the user will be able to view each step of the packing schematic one at a time to see the individual placements of each object in the container.

**3.7 Pseudocode**

A high-level Pseudocode starting from the home screen is provided below:

if (view\_old\_scan):

query database for saved scan names

list all saved scan names

query database for selected scan

display rendered 3D schematic

if (modify\_scan):

if (change\_container):

start new scan

replace old dimensions

run packing script

if (add\_object):

start new scan

add new dimensions

run scripts

if (remove\_object):

remove dimensions

run scripts

if (done):

ask user to save scan

if (create\_new\_scan):

if (lidar):

check device compatibility

if (incompatible):

print error message

open camera and record

when finished scanning:

process point cloud

generate .xyz

remove floor

reduce noise

cluster points

retrieve dimensions

output dimensions

else if (camera\_processing):

start recording

while recording:

clip images

record camera data

generate NeRF model

display NeRF model to user

select objects to segment

select container object

process object proportions

if (specify\_dimensions):

input one object size

attempt to generalize

output dimensions

run bin packing algorithm

generate step by step schematic

render the schematic in Unity

display schematic to user

while (not\_done):

if (next\_step):

display next step

if (previous\_step):

display previous step

prompt user to save scan

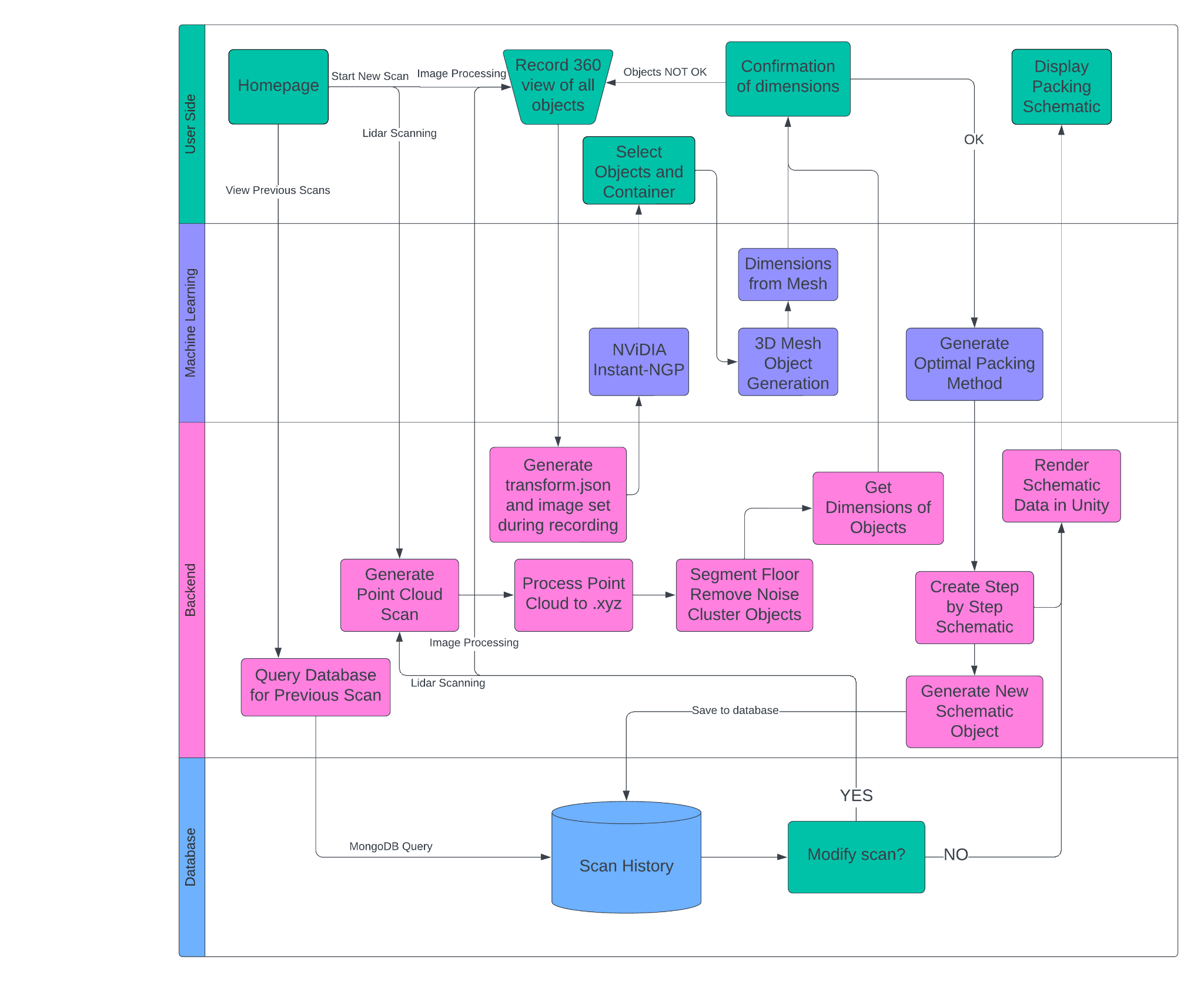
if (save\_scan):

prompt for scan name

gather metadata

save to database

**3.8 System Block Diagram**

This system block diagram lays out the entire application in four main sections: User Side (Green), Machine Learning (Purple), Backend (Pink) and Database (Blue). It demonstrates the connections between each component at a high level. The block diagram was designed in Lucidchart.

# 4 First Semester Progress - Juan Vecino

Throughout this semester, we have advanced in two principal areas: creating 3D objects from videos and developing an algorithm for efficient packaging.

**4.1 Creating 3D Objects**

We employed a neural radiance field (NeRF) AI model to generate a 3D space from images, which are extracted at 2 frames per second from a video. This model queries 5D coordinates along camera angles and uses volume rendering to produce images. As this process is differentiable, it only requires images with known camera poses for optimization. We use nerfStudio to convert these images into 3D models of the scanned objects.

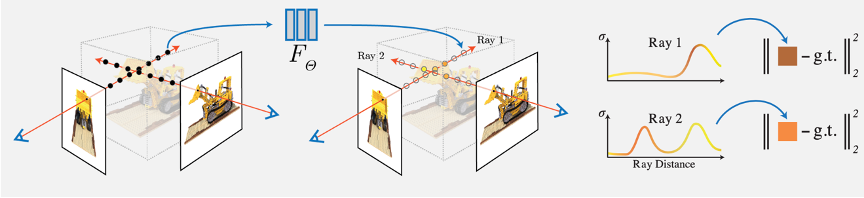


Figure 1. Example creation of a 3D object from a video of a box:

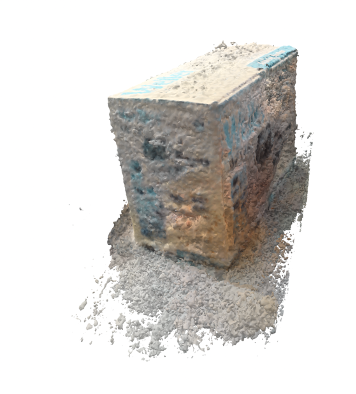


Figure 2: 3D obj. file from a video of a box

Using NVIDIA’s Instant NGP software, we were able to successfully generate 3D mesh files from: A single box, A single bucket, and multiple boxes on a flat surface. The mesh files were able to retain the original proportions and shape of the imaged object. The resulting mesh files are shown below.

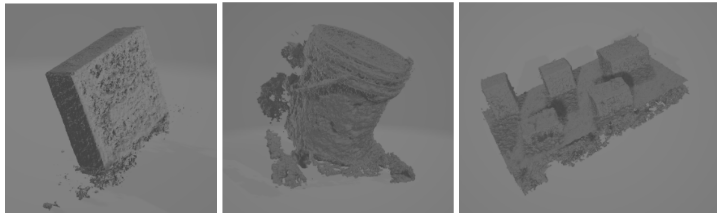


Figure 3. Three outputted .obj files of various objects

We were also able to successfully create boundary boxes around five separate boxes within the same LiDAR scan. Using a third party LiDAR scanning application, we exported a scan in .xyz format and then processed this file in Python using the open3d library. The Python script filters out the floor, excessive noise and then clusters points that are deemed significant based on the number of neighbors and specified neighbor distance. For this example, we scanned five boxes ordered with approximately 10 cm of distance between each box.

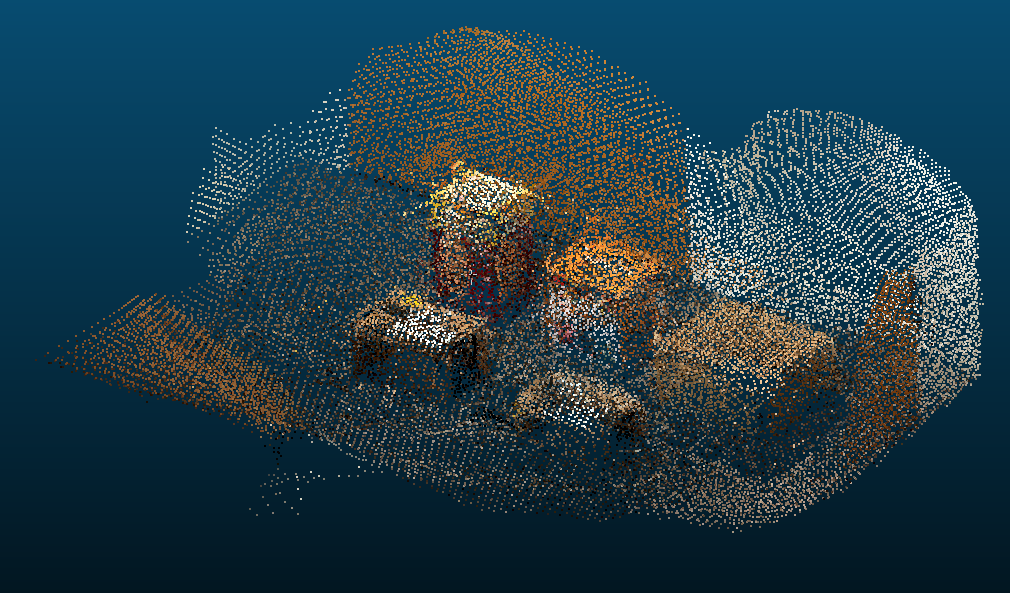


Figure 4. The generated .xyz file viewed through CloudCompare

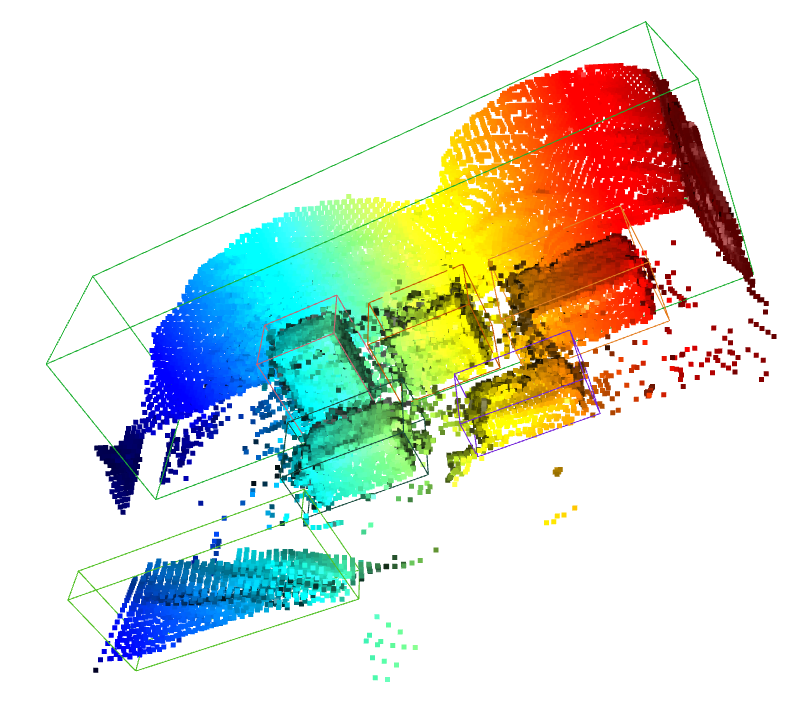


Figure 5. The generated bounding boxes of each object

This method still requires some work in terms of filtering noise and generating tight fitting bounding boxes.

**4.2 Algorithms for Efficient Bin Packing**

For packaging, we utilize a Python-based algorithm informed by the paper "*Optimizing Three-Dimensional Bin Packing Through Simulation*," [2] which determines the optimal arrangement of objects. We found that the algorithm provided by 3D-Bin-Packing provides a good packing schematic for our Minimum Viable Product. The approach uses a combination of Best Fit and First Fit Decreasing heuristics. The algorithm works by attempting to pack items at the origin of the 3D coordinate plane. If the current object doesn't fit in its initial orientation, it is rotated until a suitable configuration is found. If not, then the algorithm moves on to the next item, while keeping track of those that could not be packed for subsequent attempts. This algorithm accepts rectangular prism and cylindrical shaped object inputs, with dimensions for both containers and objects specified in the form Height X Width X Length. The algorithm can also pack multiple bins and account for the weight and fragility of each input object. Some test outputs are provided below.

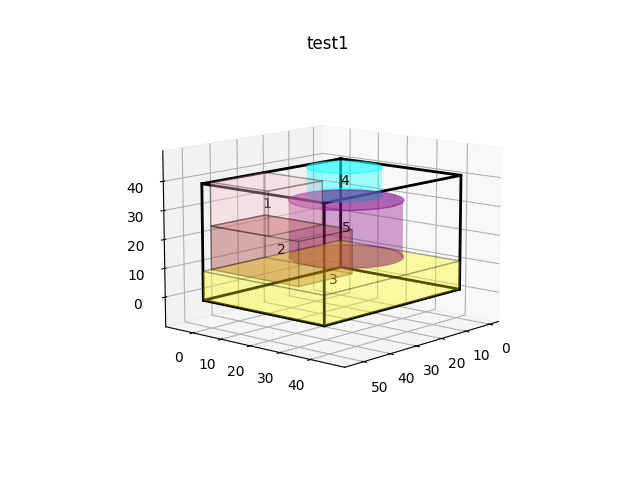


Figure 6. All objects fit in the container with trim

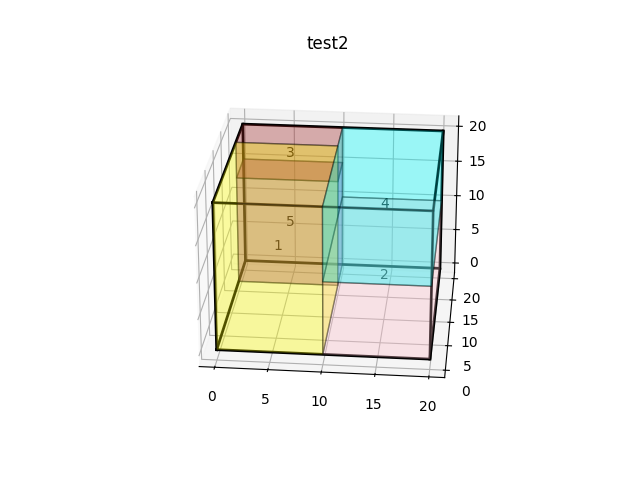


Figure 7. All objects fit in the container without trim

**4.3 Future Goals**

Our future goals include automating these processes on an AWS cloud server, which could be operated directly from a phone. Ultimately, we aspire to develop an iOS app that can seamlessly interface with AWS to perform these tasks efficiently and on a large scale.

# 5 Technical Plan - Daniellia

For the remainder of the performance period, we will be implementing the following tasks:

Task 1. 3D Object Segmentation

3D objects generated from NVIDIA’s instant NGP shall be segmented and formatted to send into the packing algorithm. The dimensions of each segmented object shall be determined.

Lead: Tristen; Assisting: Daniellia.

Task 2. LiDAR implementation

LiDAR functionality will be researched and implemented into our application in order to be able to generate Point Clouds. These point clouds will then be inputted into a Python script that will process the data points and output the dimensions of found objects for the packing algorithm.

Lead: Tristen

Task 2. Scanning to Packing Pipeline

A pipeline shall be created to pass the dimensions of the objects specified from the scanning stage into the packing algorithm. This will connect the scanning and packing modules such that packing will be automatically performed using object dimensions specified from the 3D models.

Lead: Juan; Assisting: Daniellia.

Task 3. 3D Irregular Packing

The existing 3D Regular Packing code shall be modified to accommodate irregular shapes defined from the scanning module. The algorithm should generate an optimal packing schematic in <5 s.

Lead: Daniellia; Assisting: Ramy.

Task 4. Create Application

An application shall be created using React Javascript for the front-end and Python for the back-end. The application should have an intuitive user interface.

Lead: Tristen; Assisting: Juan.

Task 5. Unity Rendering

An optimal packing schematic shall be rendered in Unity. The schematic shall be able to be interacted with by the user and illustrate step by step procedure to optimally package objects within the container.

Lead: Ramy; Assisting: Daniellia.

Task 6. Application Testing

The application shall be tested to ensure seamless integration from the back-end to front-end. The application should be compatible for both iOS and Android mobile devices.

Lead: Juan; Assisting: Ramy.

A detailed timeline of the project tasks are outlined in the Gantt Chart (**Appendix 7.2**).

# 6 Budget Estimate

The budget estimate for the Scan It! Pack It! mobile application is variable, due to the on-demand services of AWS. Currently, we estimate that our user base will not exceed the free-tier of the AWS services that we will utilize, so the Budget Estimate comes out to $0.

# 7 Attachments

## 7.1 Appendix 1 - Engineering Requirements

**Scanning**

* Object dimensions/proportions should be accurate at least 95% of the time
* Object dimensions/proportions should be precise with an error margin of 5%
* The object dimensions should be generated and found in less than 1 minute
* The video scanning process should be done in less than 2 minutes

**Packing**

* The packing algorithm should be at least 90% space efficient if all the items can theoretically fit within the container

**Packing Schematic**

* The packing schematic should be generated within 30s of the submission of object and container dimensions

## 7.2 Appendix 2 - Gantt Chart

## 

**References**

1. United States Environmental Protection Agency, “Facts and figures about materials, waste and recycling,” [Online]. Available: <https://www.epa.gov/facts-and-figures-about-materials-waste-and-recycling>.
2. E. Dube, L. R. Kanavathy, “Optimizing Three-Dimensional Bin Packing Through Simulation,” The IASTED International Conference on Modelling, Simulation and Optimization, 2006. [Online]. Available: <https://www.researchgate.net/publication/228974015_Optimizing_Three-Dimensional_Bin_Packing_Through_Simulation>
3. Müller, Thomas, et al. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding.” GitHub, NVIDIA Corporation, <https://github.com/NVlabs/instant-ngp>.
4. jerry800416. “3D Bin Packing.” GitHub, <https://github.com/jerry800416/3D-bin-packing>.