Project Report

3D Point Cloud Variation Measurement

Team no	06
Project no	28
Team members	Harshika Jain, Shivam Nayak, Sonu Guru, Subodh Sondkar
Client	Mr. Anil Kumar Upadhyay, Mrs. Chitralekha Upadhyay, Five Fingers Innovative Solutions

Objective of the project

We live in a world of 3 space dimensions, but most of the pictorial information we have, be it photos or videos, are 2-D. We have gotten accustomed to getting information in 2-D format. All automation in the field of processing photos and videos to get useful data has only been for 2-D. There is a need for automation in processing of 3-D photos and videos. The project mainly focuses on giving a point cloud of the object after extracting it and filtering it.

System Requirement

1. Operating system: Ubuntu 16.04
2. Softwares required to view point cloud: OpenCV, Cloud compare, pcl_viewer
3. Languages required: python3.6
4. Libraries required: tensorflow, pcl, cv2, python-secrets, matplotlib, django
5. Browser: Chrome, FireFox

User Profile

As 3-D scanners are expensive, and 3-D videos are not easily available, our early adopters will mostly be companies involved in 3-D printing, and research teams involved in 3-D model development and analysis.
Rich organisations who want to track progress of on-field jobs automatically can use the system to track changes in topography of objects (for example- land). Doctors and hospitals can use it to track the properties (for example- size) of deformities on the skin, or about tumours inside the body.

If the system reaches to common people inexpensively, they can also use it to track the deformities on objects, though this is far fetched. For example, we can use to analyse damage to objects based on their deformities.

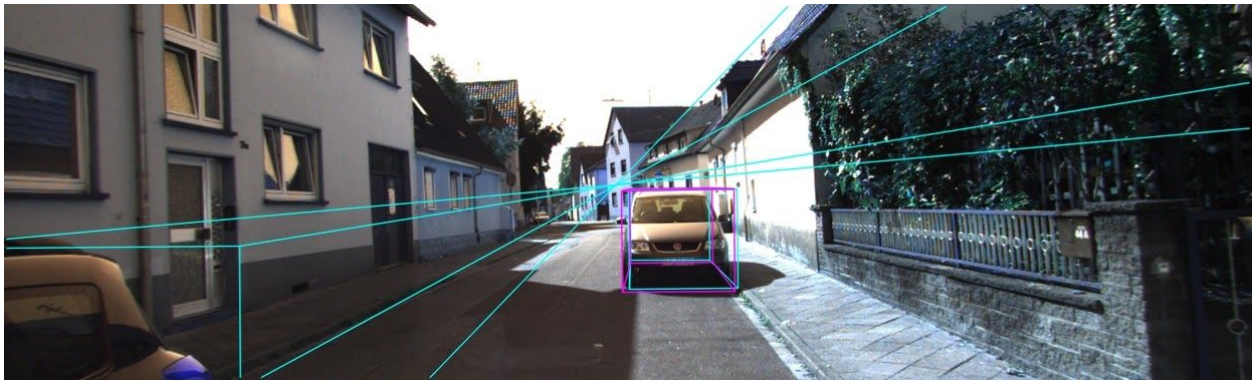
Features description

1. 2-D object detection

This feature is used for 2-D images. This feature required a pre-trained model for detection of objects. In our web app we used a trained model which can detect the “cars” in the image. So it will give a 2-D image as output with cars being highlighted by a cuboidal cage. It will also give the heatmap of the image. So using this feature one can detect whether there is any car present in the input image.

INPUT

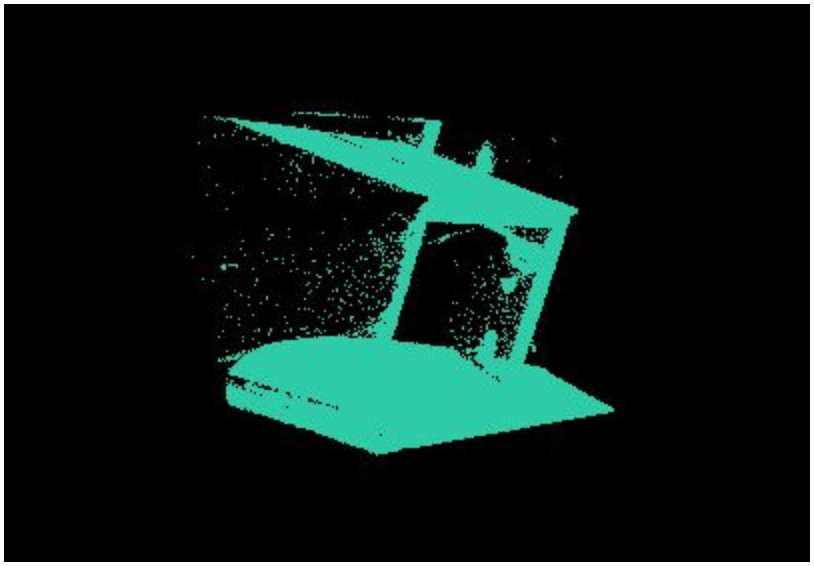
OUTPUT



2. Noise removal

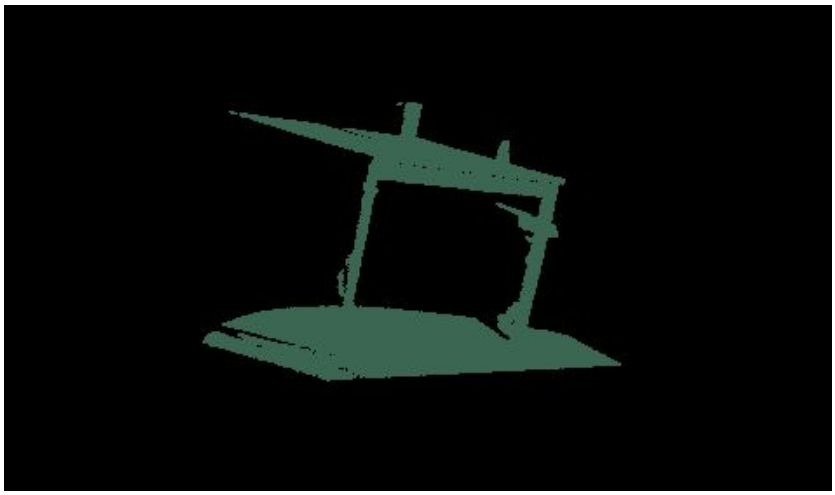
This feature is used for 3-D point clouds. So the user can give a noisy point cloud having scattered points as input. It will remove the point from the point cloud based on the no. of neighbouring point present. In our web app we define filtering on the basis of having neighbouring points less than 50. It will give 2 point clouds as output, one is inlier noise removal and the other one is outlier noise removal.

INPUT



OUTPUT

1. **Inlier output**

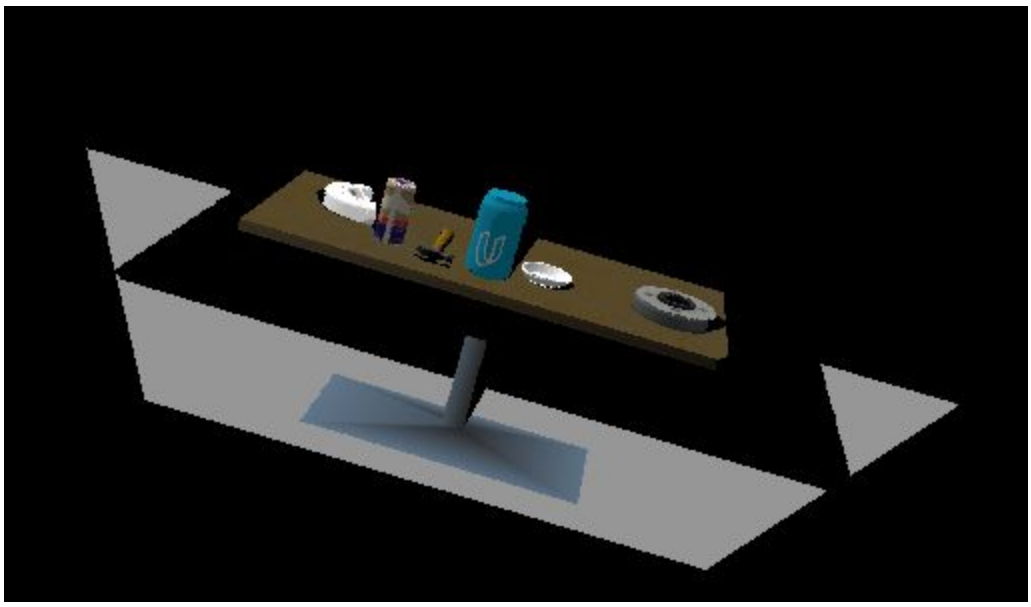


2. **Outlier output**

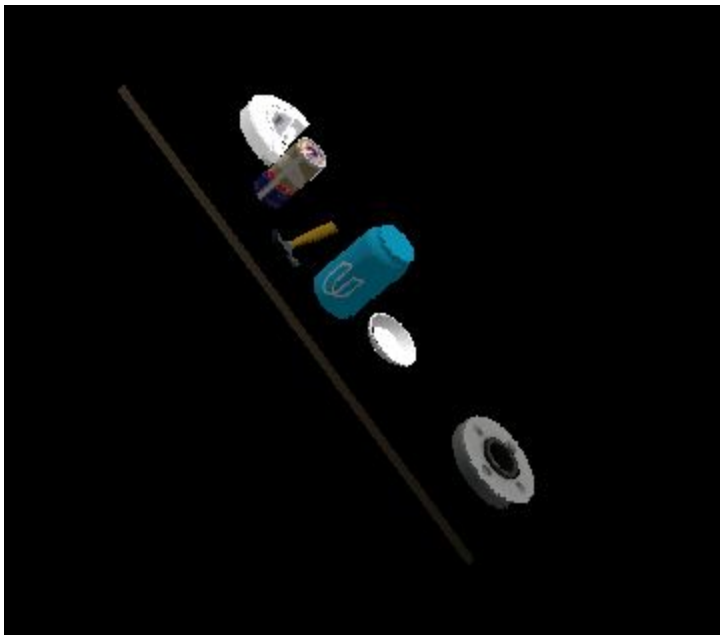
3. Object extraction

This feature is also for 3-D point clouds. Object extraction is basically separation of object from background. A point cloud scene consists of the objects kept on some support. So using this feature user can extract the objects from the given point cloud scene. It will give only one point cloud consisting of the objects only.

INPUT



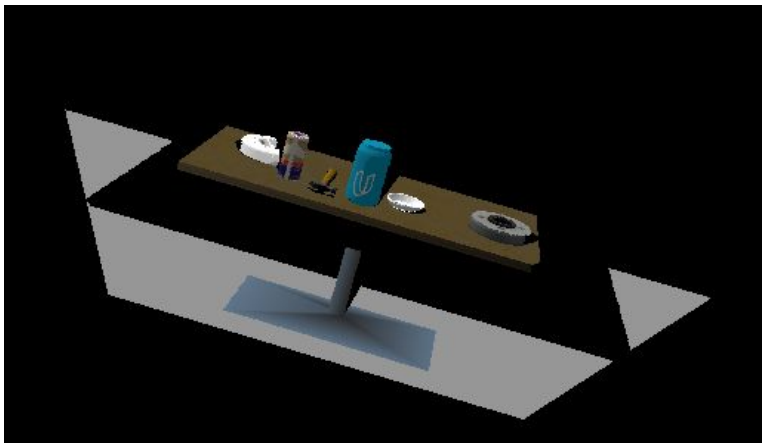
OUTPUT



4. Object separation and downsampling

This feature also works for 3-D point clouds. Using this feature, users can separate objects and backgrounds and store them as separate point clouds. Users can also get a downsampled point cloud with less resolution. It can also be used for top view extraction in which users get a point cloud of the top view of the scene. So as described before this feature gives 4 output point clouds, one is point cloud of object, second point cloud of background, third point cloud of downsampled object, forth is top view of the point cloud.

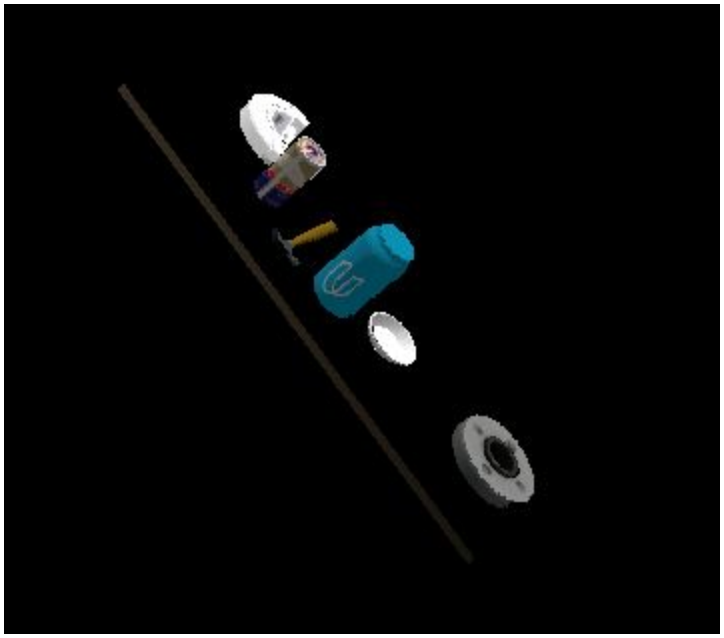
INPUT



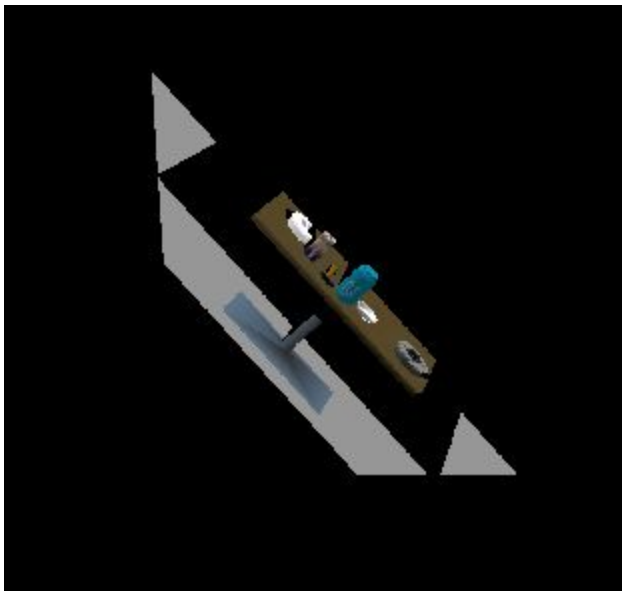
OUTPUT

1. **Background table**

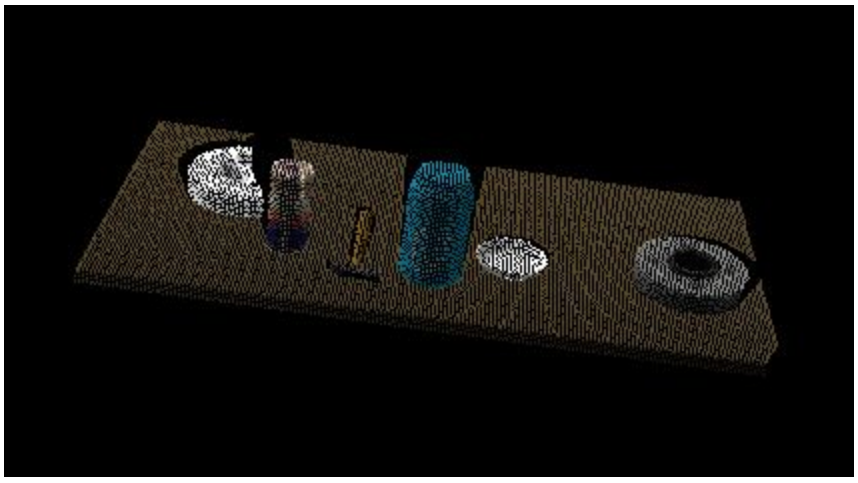
2. **Objects**



3. **Downsampled image**



4. **Top view**



Design Overview

Architectural Design

Modules

1. 2-D object identification
2. Noise removal from point cloud

3. Object Extraction
4. Separation of objects and background

Every module takes the path of the image and path of out output directory as user input and stores the output in the corresponding directory after processing.

Functional Flow

So for every function we need to take the path of the input image first followed by the path of the output directory. The corresponding function is called according to the option selected by the user. The following are the functions for different features

1. Filter_noise - for noise removal
2. Filter_object - for object extraction
3. Filter_downsampled_objects - for seperation of object and background and downsampled the cloud

System Interfaces

User Interface

We are creating a web app so that the user can interact with the system. The home page will have the main options that the user can take. These are explained serially.

Options available for user

1. Upload image
2. 2-D Image identification
3. Upload cloud
4. Noise removal
5. Object extraction
6. Object separation

1. Upload image

This page consists of a form in which users need to give some information regarding different URLs. Like for object detection in 2D images we required information like calib, image, label and velodyne. Users need to provide the URLs of these stuff. The information is stored into a database for further computation.

2. 2-D Image identification

This page consists of the links to the path of the image stored so far. Users can select any one image which will direct the user to a page consisting of all the information of the image like URL

of calib, image, label and velodyne. It also consists of the date of storing the data i.e. filling the upload image form. The main objective of this page is to take the path of the output directory where users want to store the output. So this page consists of a form which takes the URL of the output directory as input. When the user clicked on the submit button, the corresponding functions were called to compute the output. The output consists of an image with the object being highlighted in the image. It also outputs the heatmap of the same.

3. Upload cloud

This page consists of a form in which users need to give some information regarding the URL of the point cloud . Users need to provide the URL. The information is stored into a database for further computation.

4. Noise removal

This page consists of the links to the path of the point clouds stored so far. Users can select any one point cloud which will direct the user to a page consisting of all the information of the cloud like URL of input cloud along with the date of storing the data i.e. filling the upload cloud form. The main objective of this page is to take the path of the output directory where users want to store the output. So this page consists of a form which takes the URL of the output directory as input. When the user clicked on the submit button, the corresponding functions were called to compute the output. The output consists of a point cloud with less noise and it will filter the cloud by removing extra points scattered in the cloud of the image.

4. Object Extraction

This page consists of the same format as the one for noise removal. The function of this page is to give the output of the objects in the whole point cloud after removing the background. Using this feature user can extract objects from the point cloud after removing irrelevant things like table on which object is kept

5. Object Separation

This page consists of the same format as the one for noise removal. The function of this page is to give the output after seperation of object with background. So for example if a point cloud consists of fruits kept on a table, it will store fruits and table separately.

APIs used

Our coding is done in python3.6 and cpp. All the function calls from the web would be to python3.6. To link python3.6 and the web, we would use **django** as the API. Any cpp modules we want to run would be through the python code only.

Sprint Wise Plan

S No.	Feature	Sprint
1	Create a basic web app using django	3
2	Form a pre-trained model for object detection	3
3	Write code for object identification	4
4	Develop a terminal based project for object detection	4
5	Create a navigation bar to navigate through different pages	3
6	Create a form to take URL information about image	3
7	Create a page consisting of the path of the image stored so far	3
8	Make these path as a link to download page	4
9	Created a form to download image after object detection of the selected image	4
10	A page for taking input the URL of the point cloud	5
11	Write a terminal based code for noise removal	5
12	Write a terminal based code for object extraction	6
13	Write a terminal based code for separation of object	7
14	Form a page consisting of the clouds stored so far for noise removal	5
15	Form a page consisting of the clouds stored so far for object extraction	5
16	Form a page consisting of the clouds stored so far for separation of object	5
17	Make these url as links	5

18	Create a form to download cloud after noise removal	6
19	Create a form to download cloud after object extraction	7
20	Create a form to download cloud after object separation	7

Testing approach and type of testing

1. Functionality testing
    a. Check for all the links working properly
    b. Check for any broken link
    c. Check whether the image uploaded stored properly in the system
    d. Check the input format of the image
    e. If the input was wrong it should show an error

2. Usability testing
    a. The website should be user friendly
    b. The instructions provided should be clear to easy to understand
    c. The main menu should be provided on each page
    d. The output should have an option to download
    e. Functionalities should be easy to handle and can be used by general public

3. Interface testing
    a. Check the interaction between server are executed properly
    b. Check for proper error handling in case of any broken interaction

Test Environment

Setup required for each environment:

- Operating system: Linux
- Front-end running environment: Python
- Back-end framework: django
- System and applications: Gitlab, VS code/sublime
- Browser: Google Chrome
- Network: local-host defined by the terminal

Restoring strategy:

We all are using Gitlab and we push on the gitlab after every update in the project. So we can keep a track of every commit and any commit can be restored from Gitlab directly in case we require restoring.

Testing tools:

- Terminal: We test our code(backend) on the terminal by executing the python file and checking whether we got an image with a highlighted object in our media folder.
- Browser: We directly test the UI for any broken link and error handling.
- Pcl-viewer: to view the point cloud we used the pcl-viewer of pcl-tool.

Problems Faced

1. For object detection we can train a model as it required a lot of data and memory which is out of scope of out laptops
2. Finding a pre-trained model is a problem itself
3. This is very new for us that's why it takes a considerably large amount of time to learn all this.
4. Pcl library is not compatible with Ubuntu 18.04 so we have to work inside the virtual box.
5. Due to current conditions we face a lot of difficulties to work as a team. Communication is always a problem.
6. For testing we don’t have enough data like noisy point clouds etc.