Intro (3D Object Detection – Problem Statement)

State of the art of the project (Why the project is important, what will it allow)

Tasks, concept and results (Chapter: Methodology, i.e. Machine Learning approaches from dataset to network choice to results and evaluation, include the workflow machine learning diagram with yes forward and no back to a previous step, Chapter: literature project specific information, pointnet diags) (following up the methodology with the results Chapter)

Short summary and outlook (chapter results and future)

**Chapter 1: Introduction**

**Introduction**

The problem of object detection and classification is an inherent issue in Computer Vision. IBM defines Computer Vision as “a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs, and based on those inputs, it can take action.” [IBM]

To address this problem, earlier approaches included feature extraction to identify objects, which are time consuming and highly manual. A later approach that emerged is CNNs (Convolutional Neural Networks) which is a more scalable approach that leverages methods from Linear Algebra, such as Matrix Multiplication to identify patterns in an image.

Within the standard 2D object detection, the information and the features used include edges and RGB colors [IBM] which abstracts the features (corners and edges as well as RGB channels). To position these objects, which is a task under the umbrella of object detection, a bounding box is regressed, as follows up, an image would be an input and the output if a box or a box coordinates that are around or correspond to this object (a car in an image for example) [MaxPlank].

**3D Object Detection:**

Recently, higher level applications like scene understanding and object positioning require a richer form of information such as Point Clouds, in which the regression 3D box provides not only viewpoint information, but also information about the position of the object in the 3D space [MaxPlank].

3D information is available as a set of vectors containing the x, y and z pairs representing the vertices that make up the point cloud. As the data in the pointcloud is random, and is hard to learn from, many proposals preprocess the points to voxel grids as views before network consumption which renders the data voluminous as well as introducing transformations that may change the original data [PointNet]. [VoxelNet] is an example of such approaches. Other appraches include working on the 2D images using CNNs to extract features, as the work with 2D images has been far more extensive than working with point clouds. After that, a 2D box is regressed to give the object’s position, which is further relayed to obtain the box’s 3D coordinates or the object’s z position, an example would be the works of [MobileNetSSD with Realsense Repo][MobileNetSSD repo]

Other approaches propose working directly on the pointcloud, feeding it to the network and performing the classification and/or the detection task [PointNet][PointNet++][VoteNet][BoxNet]. Though the nature of the pointcloud is random and lacks order, these approaches, almost all of them, make use of learning the pointcloud features with a modification in the network, specifically, incorporating symmetrical functions to preprocess the data for learning allowing for the learning to take place from directly consuming the pointcloud [Pointnet][Pointnet++] and they oftentimes employ the architecture of pointnet++ as a backbone to the actual network [H3DNet] [PointNet++][VoteNet][BoxNet].

**Overview of the report:**

In chapter 2 the main problem will be discussed as well as the issues related to 3D object detection in general, chapter 3 will go through project specific literature, adapted proposals and current state of the art. In chapter 4 the proposed methodology will be discussed and chapter 5 will conclude the report with the results discussion and future work.

Online REFS:

[IBM] <https://www.ibm.com/cloud/learn/convolutional-neural-networks#toc-types-of-c-yL2bT7qZ>

[MaxPlank] <https://www.mpi-inf.mpg.de/news/spotlights/understanding-images-videos/3d-object-detection/>

**Chapter 2: Problem Statement**

**Problem Description:**

The main goal is to detect and localize 3D structures (KLTs) such that the information is available to a robot manipulator equipped with a gripper and a 3D Stereo-camera. The structures are set upon one another and have different categories depending on the objects inside, in which case the objects vary from C-Parts (fittings, screws, etc.) to dampers and other automotive parts. End goal is to develop a proof of concept, utilizing tools such as deep learning and 3D object detection to confirm the validity of the approach and culminate an in-house knowledge about the approaches available and their viability.

**Setting:**

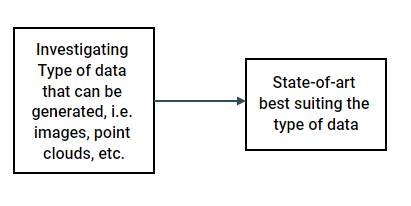
The setting incorporates a manipulator, an end-effector with a greifer and an intel RealSense D435, the information from the camera are extracted, processed by a computer which throws out data fed to the Universal Robot manipulator as commands, which after receiving the position and class information acts accordingly. Fig. () shows the robot manipulator and Fig. () shows the Intel RealSense Depth (stereo-vision) camera.

**Manipulator + Greifer (Pictures)**

**Intel RealSense D4 depth camera**

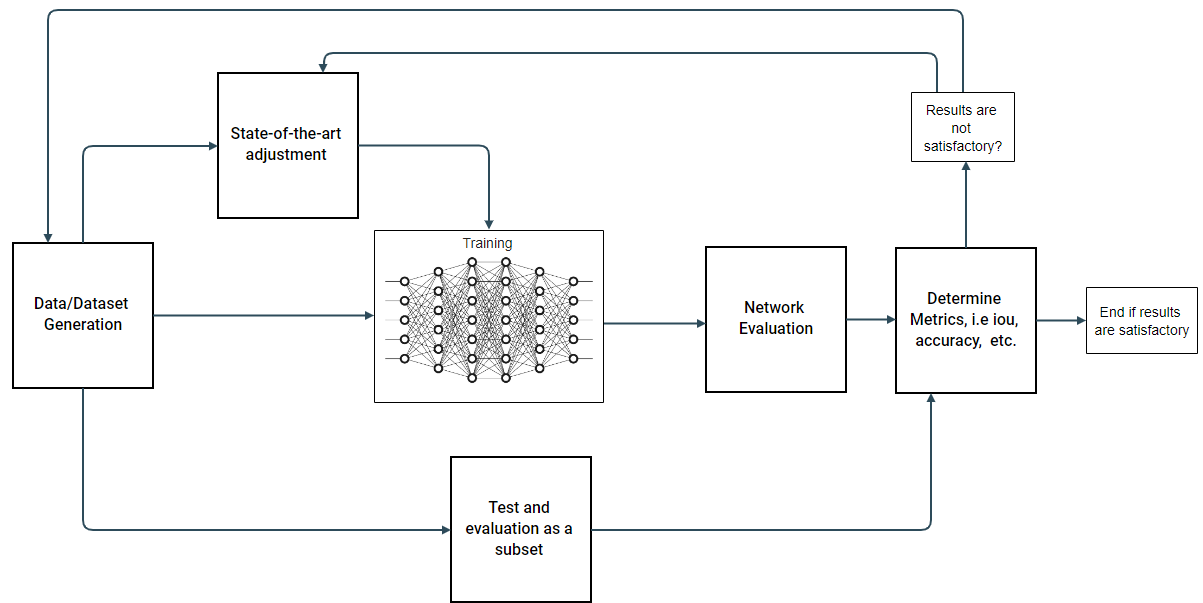
**Targeted Detection Pipeline:**

The detection pipeline includes gathering the data, investigating the state-of-the-art, passing the state of the art to the data gathered, such that the targeted Network is able to learn from this data:



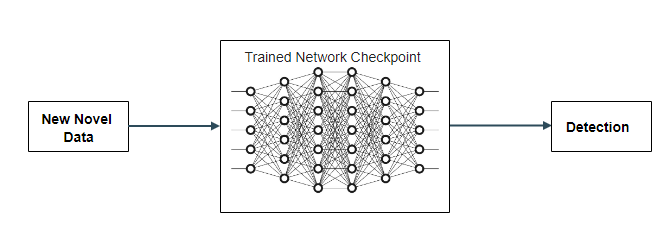
**Data à Network à Training**

After training the network, and for the purpose of testing the employed the state-of-the-art, testing data is prepared in a similar manner as the training data and the necessary code refactoring and edits to the code are carried out, proper metrics are then set, such that in the end, weakness points are determined, and necessary improvements carried out.



**Testing à Network Training Checkpoint à Inference**

After the training has been carried out, the checkpoint where the network has stopped training, i.e. the last training epoch, is saved and an inference is carried out using new novel data. Additionally the network, is the state-the-art allows it, can be tested in real-time



Chapter: Implemented States of the art: votenet – etc

**Chapter 3: State-of-the-art , deep learning and transfer learning**

**MobileNets:**

MobileNets employ a set of depth-wise separable convolutions, with the main point of reducing the computation in the first few layers [MobileNets]. The structure is mainly depthwise separable convolutions, except for the first layer which is a full convolutional network. This structure is carried out to ensure the networks ability to be employed on mobile and embedded devices carrying out inference/realtime inference on the spot, considering the limited set of computational and memory available in such devices. [MobileNets]. Fig. () is an example of vanilla convolutions.

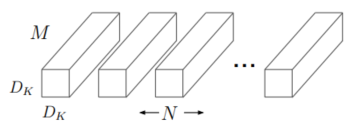


Fig. () [MobileNets]

In MobileNets the depthwise seperable convolutions are applied in 2 steps [MobileNets]:

1. A single Dk xDk x 1 input filter to each channel [MobileNets], as illustrated by Fig. (2)

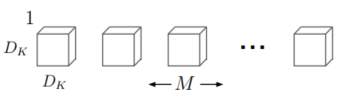


Fig. (2) [MobileNets]

1. A pointwise M x 1x 1 convolution to combine the outputs of the previous filter [MobileNets]

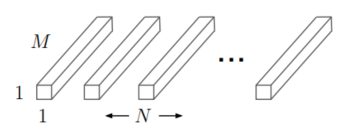


Fig. (3) [MobileNets]

MobileNets which apply 3x3 depthwise seperable convolutions use 8-9 times less computational power, comprated to regular convolutions with a slight reduction in accuracy [MobileNets].

The work of MobileNets is based on [Xception], which introduces the depthwise seperable convolutions, Fig. (4) illustrates the general structure applying both depthwise and pointwise convs.

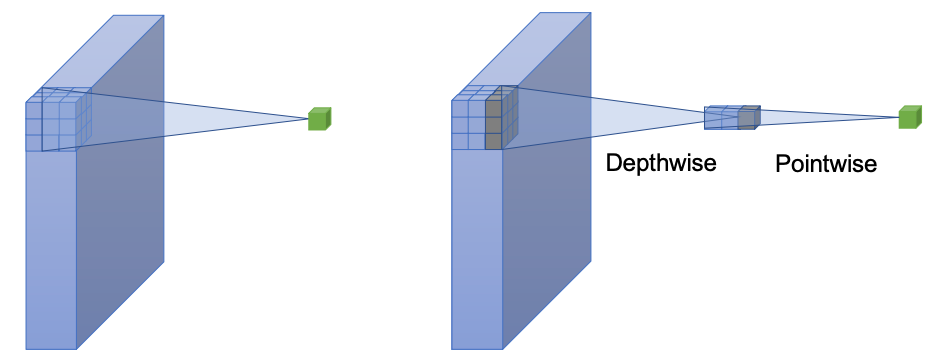


Fig. (4) [Xception]

Fig. (5) demonstrates the architecture of MobileNetV1 according to [MobileNetV1-Figure-Paper; see lit in praktikum stuffs for reference pdf]

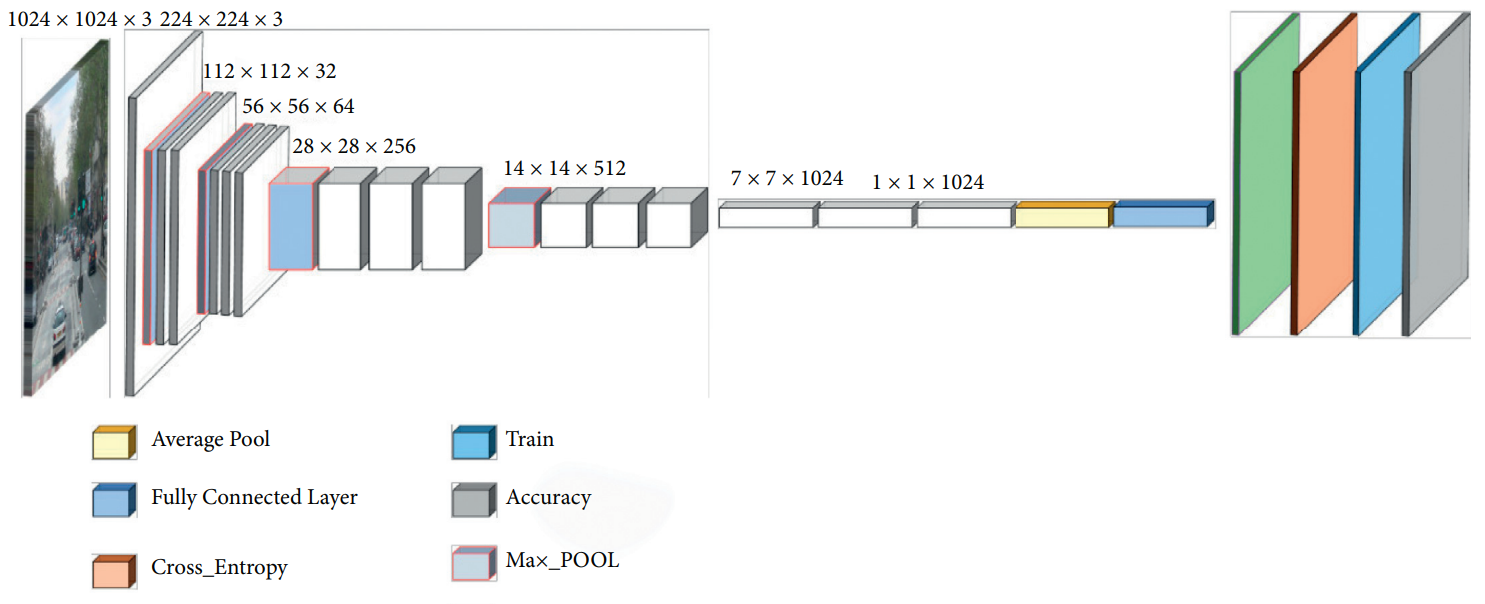


Fig. (5) [MobileNetV1-Figure-Paper; see lit in praktikum stuffs for reference pdf]

MobilenetV1:

MobilenetV2:

*is based on an inverted residual structure where the shortcut connections are between the thin bottleneck layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. Additionally, we find that it is important to remove non-linearities in the narrow layers in order to maintain representational power*

The structure of MobileNetV2 differs from that of MobileNetV1. It is based on the concept of inverted residuals, where a shortcut connection is established between bottleneck layers. Bottleneck layers, known as projection layers, and as the name suggests, reduce multi-layer tensors. An intermediate layer is also introduced, it uses the lightweight depthwise convolutions to filter features as a source of non-linearity. Additionally, the authors remove non-linearities in the narrow layers and the reasoning behind it is to maintain the features that are representative [MobileNetV2].

Fig. (6) demonstrated the general architecture of MobileNetV2 according to [https://machinethink.net/blog/mobilenet-v2/]

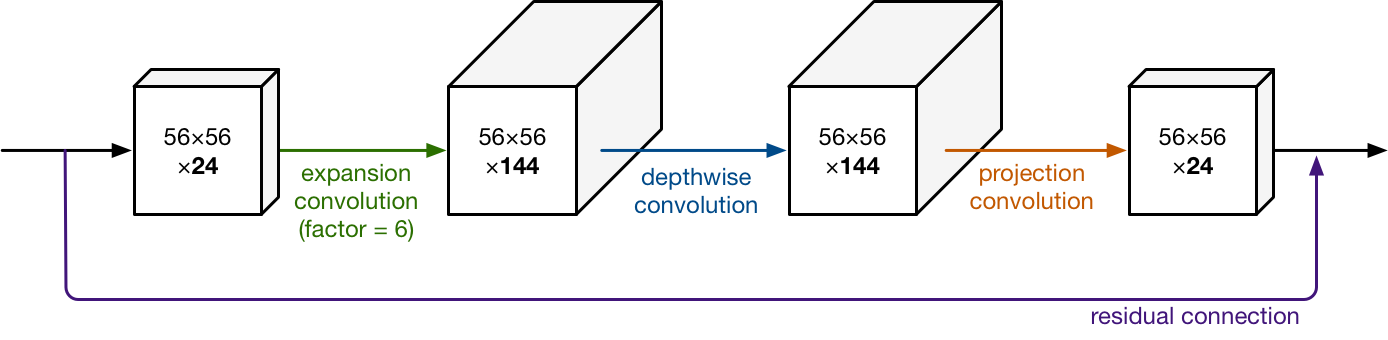


Fig. (6) MobileNetV2 general architecture according to [<https://machinethink.net/blog/mobilenet-v2/>]

Online REFs:

[MobileNetV2 Fig.] <https://machinethink.net/blog/mobilenet-v2/>

**Transfer Learning:**

**As the [**[Chapter 11: Transfer Learning](ftp://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf), [Handbook of Research on Machine Learning Applications](https://amzn.to/2fgeVro), 2009.**] puts it: “***Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned.”*

*Transfer learning is effective and desired when the features learned from one set are general.* **[**[Chapter 11: Transfer Learning](ftp://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf), [Handbook of Research on Machine Learning Applications](https://amzn.to/2fgeVro), 2009.**]**

VoteNet:

VoteNet leverages PointNet++, as PointNet++ processes the point cloud as it is and learns features in the point cloud structures without having to convert it to regular structures. This consequently indicates that the information is taken as is, rather than partly losing information in the conversion process.

A standard approach to apply PointNet++ is to propose 3D bounding boxes is to follow suit with the standard box proposal methods by proposing the boxes that has dense objects within. This approach is not favourable in 3D proposals for a few reasons, firstly, the inherent sparsity in point clouds and secondly, depth sensors tend to capture only the surface of the objects in the scene, so the centers of the objects are usually spatially in a void place in that point cloud. Herein comes VoteNet to leverage Hough Voting and customize to establish a relationship between the centers and the points, which after aggregation are used to generate bounding box proposals.

The input is first sampled through the backbone network a set of seed points is sampled, from which feature are derived and votes are generated. The votes, as aforementioned are targeted to indicate an object’s bounding box’s center location and to generate the bounding box proposals they are clustered and aggregated through a learning module.

1. PointNet:

PointNet is a unified approach that consumes point clouds directly as inputs and outputs labels and/semantic labels, labels are generated per point in the point set. The authors of pointnet argue that it has a simple architecture. Each point is processed individually as a coordinate (*x; y; z*), the paper leverages the use of symmetric functions, such as max pool, and including an MLP (Multi Layer Perceptron). The network then learns special relationships between interesting points in the given input set, the selection grounds are embedded in the network parameters. The final layers are fully connected layers that aggregate the learned features into a ‘global’ descriptor for the shapes in the case of shape classification and the point labels in the case of segmentation. [PointNet]

Fig. (7) illustrates the architecture of PointNet according to [PointNet]

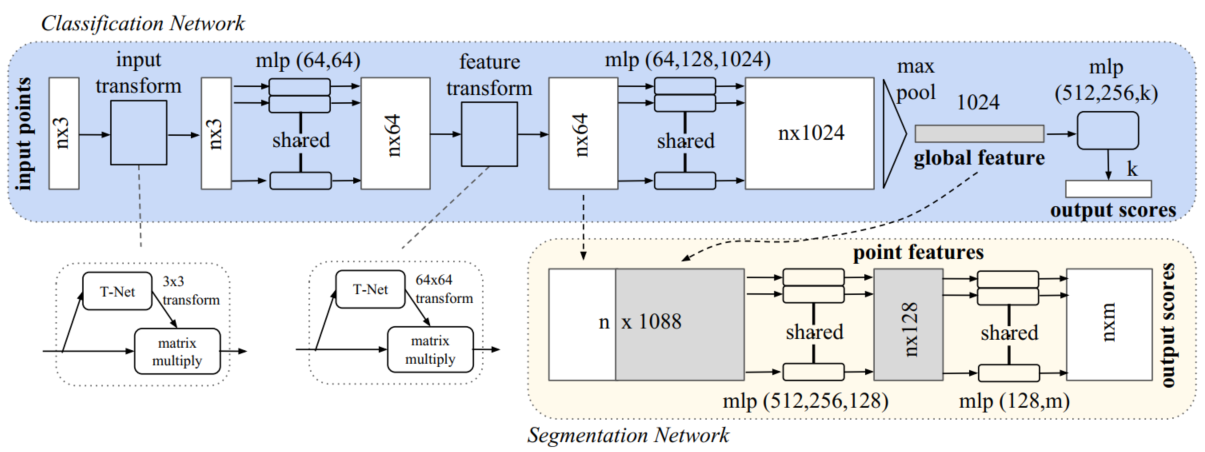


Fig. (8): Architecture of PointNet

1. PointNet++:

One of the problems that lead to the inception of PointNet++ is that PointNet does not capture local structures into features, it rather constructs the relationship function based on the whole consumed point cloud, regardless of the individual and distinct geometries embedded within the point cloud. CNNs takes datapoints in grid-like shapes and is able to capture features in different scales and among different resolutions. The field receptivity of the neurons are proportional to weather the levels are low or high, the authors of PointNet++ argue that abstracting local patterns in a more explicit way allows for better generalization regarding the network parameters when the network is shown net data. [PointNet++]

PoitNet++ hence processes points hierarchically, the points are partitioned according to a defined distance metric, and the features are extracted from these points in a manner similar to that of CNNs, these features are then aggregated further into a larger point feature hierarchy, and so on the process is undertaken from the bottom up until the whole point cloud is enveloped in-whole. [PointNet++]

The generation of the overlapping in the point set is first by selecting centroids of the areas of interest with the aid of the FPS (Farthest Point Sampling) algorithm. [PointNet++]

Fig. (9) demonstrated the architecture of PointNet++ according to [PointNet++]

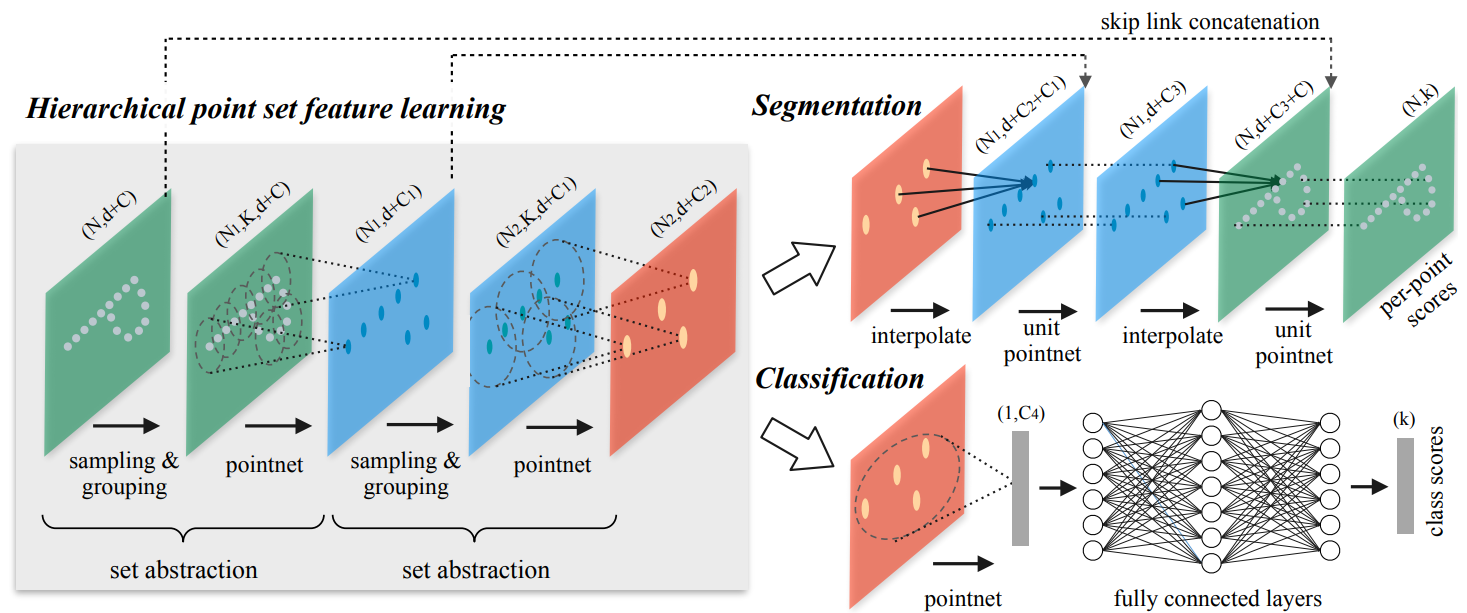


Fig. (9) Architecture of PointNet++

----Talk about the Fig. (9) in detail and possible include additional figs...

2.1 Hierarchical Point Set Feature Learning

As aforementioned and in contrast to PointNet that lumps features from point sets using a max pool operation, PointNet++ does this feature lumping in a bottom-up approach in what is called hierarchical feature learning.

As seen in Fig. (9) in the feature abstraction layers 3 types of processing take place, firstly sampling the points (sampling layer), grouping points (grouping layer) and the PointNet layer.

The points are processed in these layers as follows: centroids are selected from the point set via a query algorithm, in PointNet++ this is FPS (Farthest Point Sampling). After that regions around these centroids are selected via the grouping layer/process. Finally, the PointNet layer then aggregates local features from these set of local points.

*Sampling Layer:* Given the point set: {*x*1, *x*2,…, *xn*} as input to the network, the FPS algorithm is deployed to select a subset: {*xi*1, *xi*2, …, *xim*} such that *xij* that is the most distant point form the set: {*xi*1, *xi*2, …, *xim*}. The advantage of FPS over random sampling is that the point set is included as a whole and all of the regions of the point set are represented. Another key feature that sets PointNet++ ahead of CNNs with reggard to data representation is that CNNs as put by the PointNet++ authors: “In contrast to CNNs that scan the vector space agnostic of data distribution, our sampling strategy generates receptive fields in a data dependent manner.”

*Grouping Layer:* The input to the grouping layer is the point set of size *N ×* (*d* + *C*) as well the set of centroids that the sampling layer outputs. The output is a group of points of size *N’ × K ×* (*d* + *C*), where N’ is the number of centroids and K is the number of points within the centroid’s neighborhood. The Ball Query algorithm is deployed to limit the points in the centroid’s neighborhood number of points.

*PointNet Layer:* the input is *N’* local regions of points with data size *N’×K×*(*d*+*C*), the features of the local regions are encoded and the layers throws out an output of size: *N’ ×* (*d* + *C’’*).

2.2 Multi Scale Resolution:

It is not uncommon for point clouds to have varying densities, not only within the same point cloud, but also compared to other point clouds in the same dataset. This is a problem, as it can be more difficult to match sparse and dense point clouds, i.e., features learned from dense point clouds/areas cannot be generalized to sparse ones/areas.

This is where Multi Scale Resolution comes to play, as shown in Fig. (10)

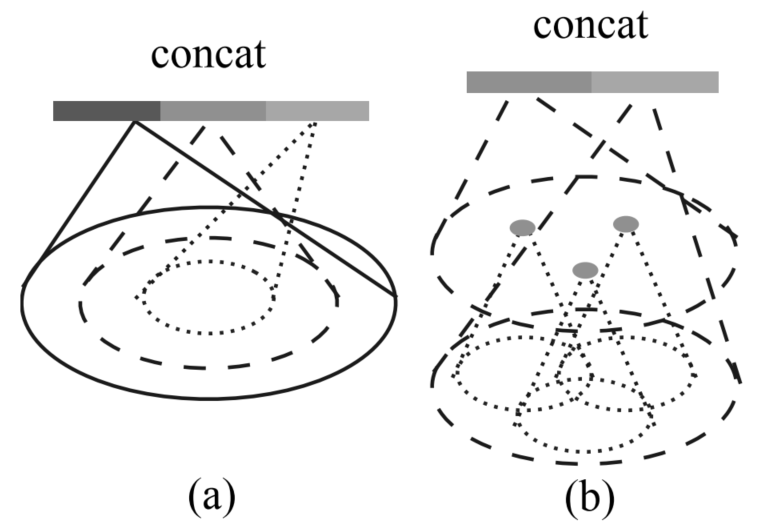


Fig. (10): Multi Scale Resolution, as seen in [PointNet++]

It is used to capture the features in multiple scales, mainly done by applying grouping layers that would then be the input to PointNet but this is done in different scales. The features are then concatenated, solving the dense vs. sparse point cloud problem.

1. Hough Voting and the Hough Transform:

Introduced in the late 1950s, it is originally used to detect patterns in a set of points in some arbitrary coordinate system, but extends this pattern detection to peak detection in the parametric space.

It achieves this by finding the parameters that correspond to these patterns [JIllingeorth-ASurveyoftheHoughTransform]. The patters are transformed into another space in which the derived features will be mapped in a compact way [JIllingeorth-ASurveyoftheHoughTransform]. Hence the HT transforms the detection problem into a space where they can more easily solved problem of peak detection [JIllingeorth-ASurveyoftheHoughTransform].

Furthermore, in the context of object detection the Hough Voting or the Hough Transform is used to detect abnormal or complex objects in image patches. [24] introduced implicit shape methods, other work used it for plane extraction from 3D point clouds and pose estimation [VoteNet]. Hough Voting has also been combined with learning techniques. (All paragraph [VoteNet], cite it everywhere outside of the part with [24] and [JIllingeorth-ASurveyoftheHoughTransform]).

[24] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. International journal of computer vision, 77(1- 3):259–289, 2008. 2, 3

3.1. Deep Hough Voting

VoteNet leverages Hough Voting in 2 distinct ways, the first one is by establishing that voting based detection is more compatible with point sets that are sparse, as the Region Proposal Networks (RPN) will have to extra computations in void or sparse areas. Secondly, the information is aggregated and is built up from small sets up to engulfing the whole point set (point cloud).

To adapt the concept of Hough Voting into a 3D state-of-the-art VoteNet follows the following approach:

Firstly, interest points are selected in line with the aforementioned information. The Votes are then generated via a learning module the is part of the larger neural network. These votes are then aggregated and ‘bad’ votes are then excluded. Finally using these aggregated features and votes objects are proposed, the proposal is in the form of location of the object in the particular coordinate frame, the dimensions and orientation of the bounding box that are generated via the bounding box and finally, the semantic class. Fig. (11) illustrated the architecture of VoteNet.

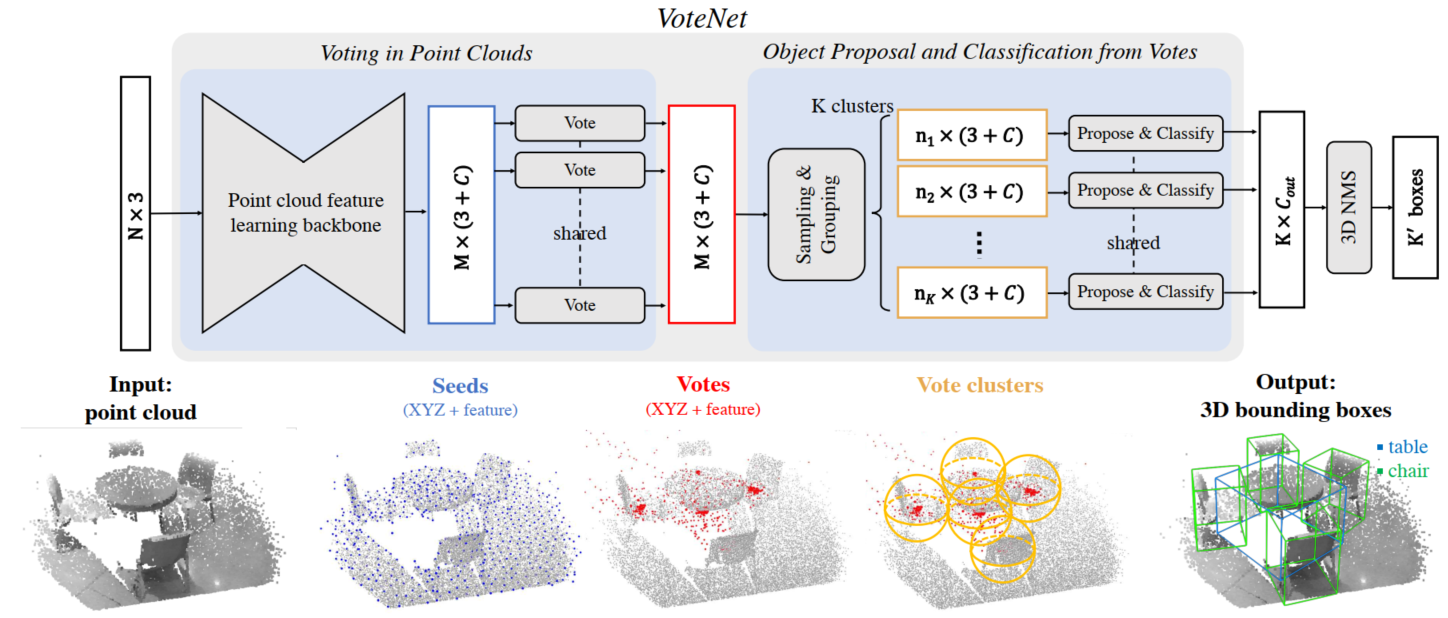


Fig. (11) architecture of VoteNet

Chapter 4: Implementation, tools and data type, preparation and results

In this chapter the implementation details will be illustrated as well as dataset preparation, training and results.

To achieve good results on a state-of-the art, a careful implementation of the code to fit a certain application is of the essence, this includes diving into the code base to understand and verify the results on benchmark datasets to consequently training on a custom dataset.

2D to 3D methods:

MobileNet-V1 and OpenCV:

The idea is to utilize a trained MobileNet-V1, use the checkpoint to generate bounding box proposals and then leverage the x and y coordinates of the bounding box to re-reflect those on the point cloud, setting an approximate location to where the points constituting the object in the point cloud are.

This is then followed by a depth calculation, which in this case calculated as the average z-coordinate of all of the isolated points deemed belonging to the object by the 2D network. This is carried out with the aid of OpenCV’s “DNN” module, the depth is then calculated using the cameras pipeline and the information is combined into an OpenCV label, Fig. (12) demonstrates an example of the results.



Fig. (13) person is the label, the percentage is the confidence of the prediction and the last 0.34 is the distance from the camera in meters.

MobileNet-V2

The dataset is prepared from 2D images that are labeled in .xml format, the network is then trained to these images and labels and the checkpoint saved for inference. The 2D bounding box information is just as MobileNet-V1 used to retrieve the depth information using the corresponding z-coordinate. In the case of MobileNet-V2 this is done via the Tensorflow API (used for training as well as retrieving information such as labels, confidence for inference) and custom functions that leverage the output information from MobileNet-V2, combines it with the camera pipeline (pyrealsense) and throws out, in addition to labels and confidence scores, the distance to the object. Table (1) demonstrates the results of the training.

* Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.915
* Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.901
* Loss zwischen 1.4-2.0

Table (1)

The network is not trained from scratch, as this will be extremely lengthy until meaningful results are reached, therefore Transfer Learning is deployed in this case. Particularly, a MobileNet-V2 that is trained on the COCO dataset, which is constituted from more than 93 classes that generate ‘general’ features. After setting the checkpoint to a trained COCO dataset, the last layer of the network is fine-tuned, i.e. the weights and biases are reset. The network is then trained in this setting on the net custom dataset, and as far as 2D methods go the results obtained are good.

VoteNet, BoxNet, PointNet and PointNet++ :

To investigate how PointNet processes the data for classification an example is taken into consideration. As PointNet consumes a subsampled version of the original point cloud. Fig. (14) illustrated the process of classifying an object by establishing geometric relations through PointNet’s symmetric functions.

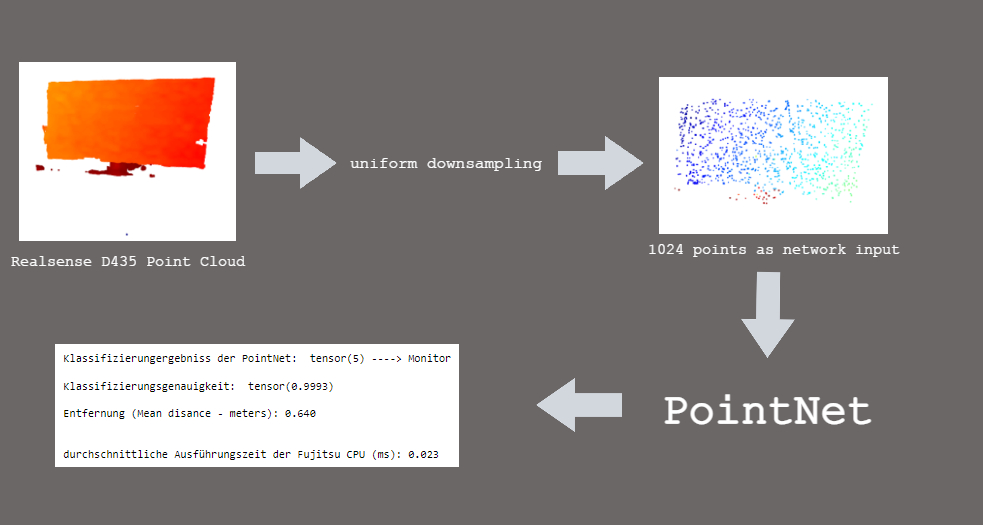


Fig. (14)

To deploy this on a larger scale, particularly, in point sets with several objects as well as to leverage this method for object detection PointNet++ is utilized as a backbone for most of 3D detection techniques that consumes the point set in an ‘as is’ fashion such as VoteNet.

An example of a pretrained VoteNet is shown in Fig. (15), this network is trained with the benchmark SUNrgbd dataset (a dataset of indoor items), the point cloud in Fig. (15) is one of the validation set of the SUNrgbd dataset.

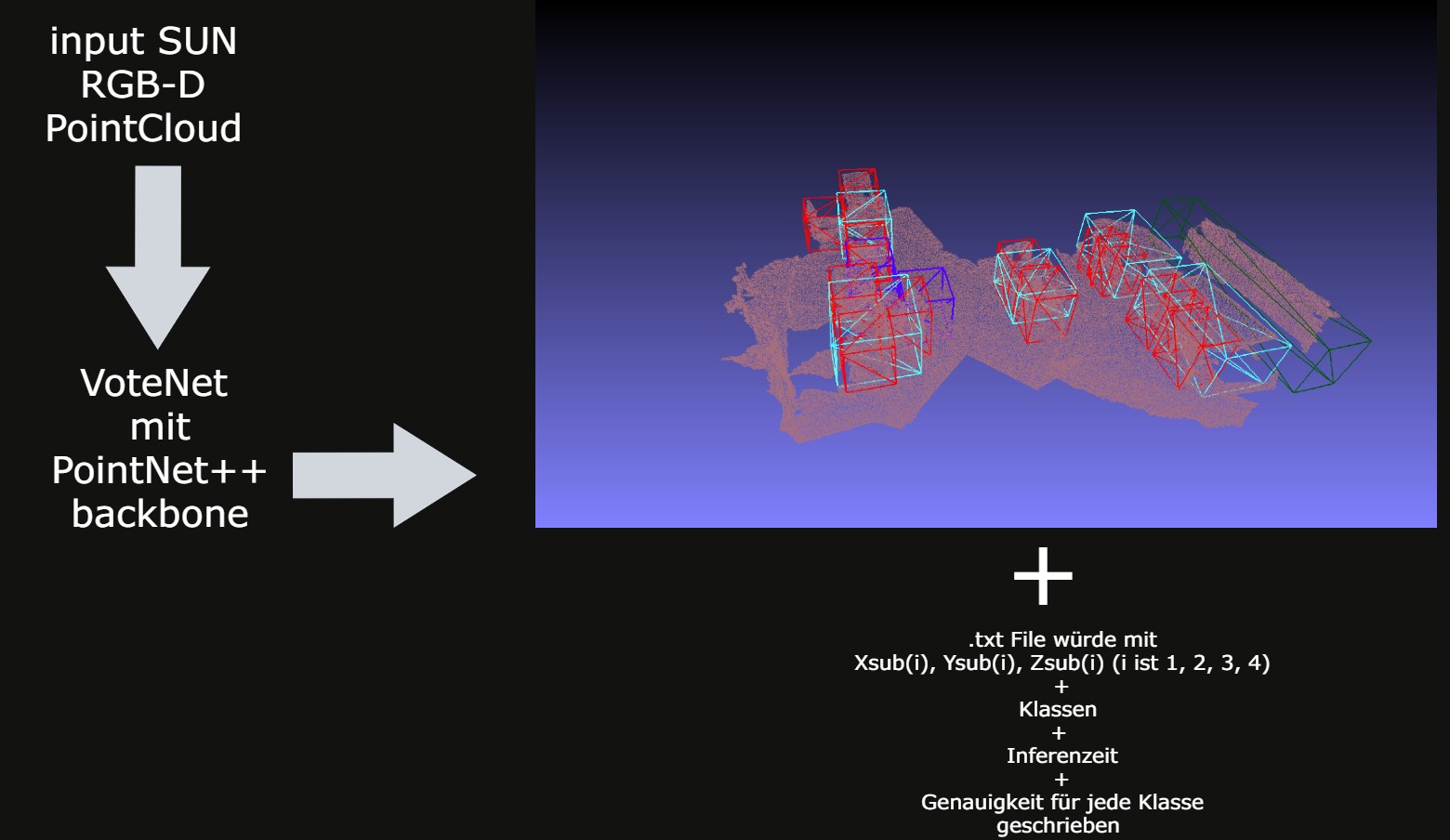


Fig. (15)

the point cloud in Fig. (16) is taken through the RealSense, i.e., the network has not seen the point cloud before and Fig. (16) constitutes an inference example on a custom point cloud.

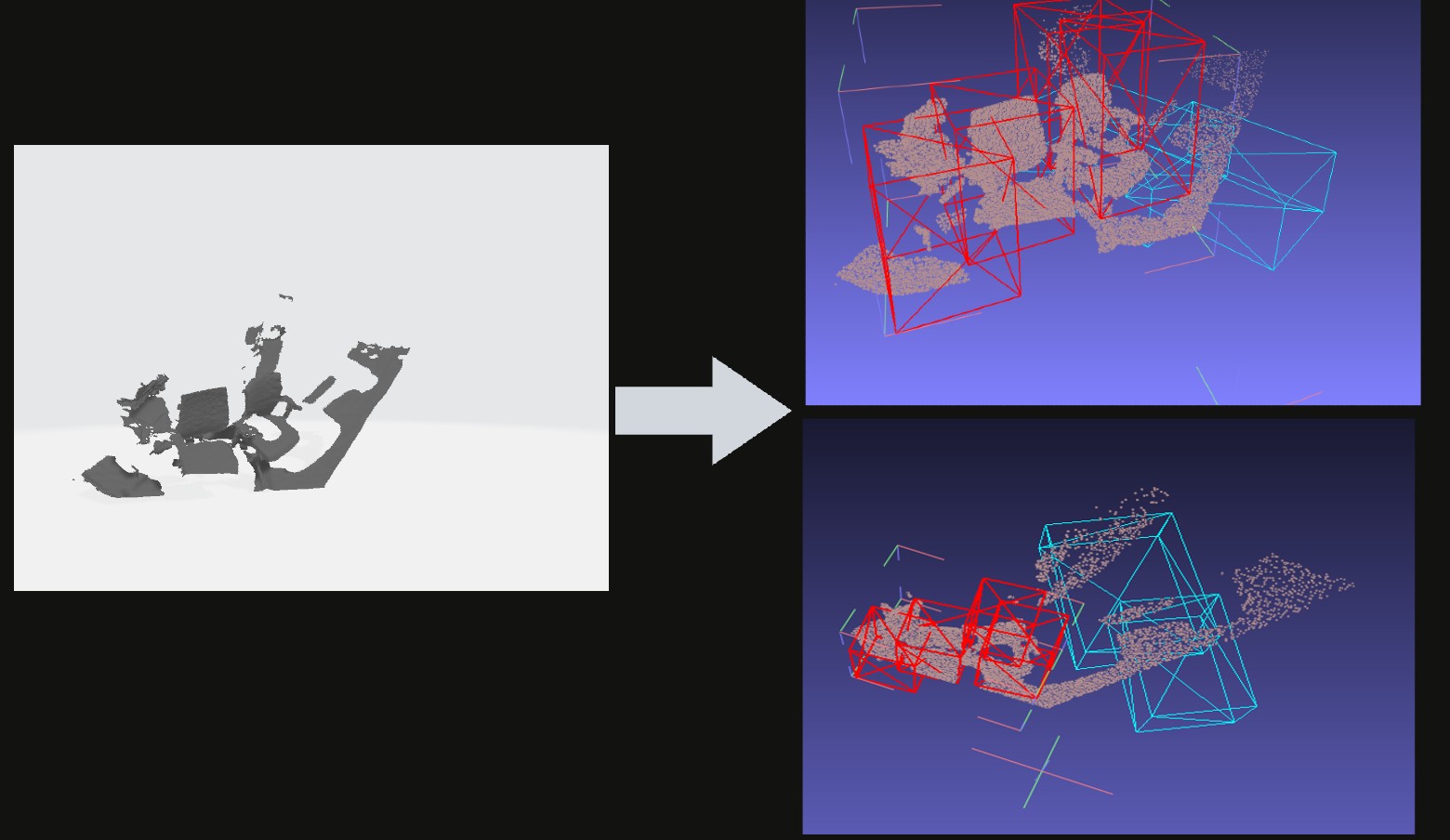


Fig. (16)

For the custom dataset and to annotate the data, LabelCloud is used, which is an Open Source software that generates 3D bounding boxes and labels for point clouds. The coordinates and the center of the bounding box are calculated in the camera coordinate frame and are then fed to the network as Ground Truth bounding boxes. Fig. (17) illustrates an example snapshot of the Software and the point cloud.

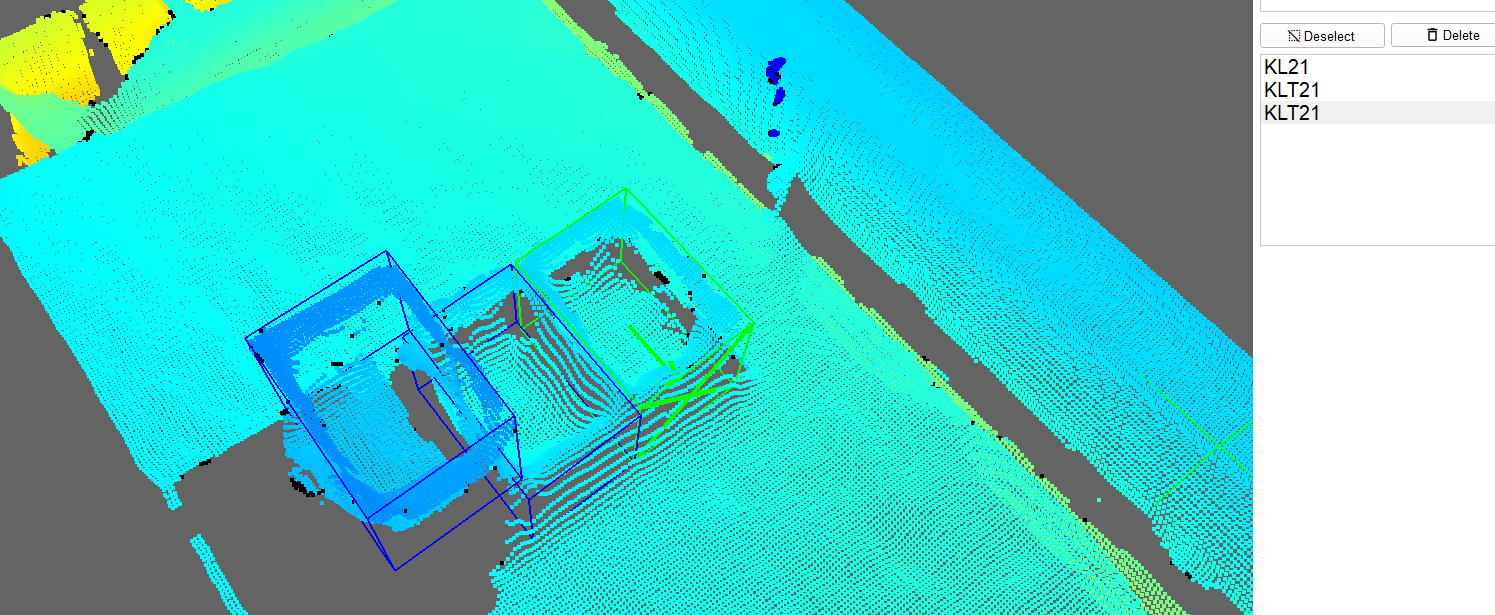


Fig. (17)

Table. (2) and Fig. (18)demonstrates the training summary of VoteNet:

| # Epochs | subsampled points | batch | accuracy | box loss | mean loss | vote loss |

| ------------- | ------------- | ------------- | ------------- | ------------- | ------------- |------------- |

| 180 | 20000 | 10 | 0.997882 | 0.165145 | 3.764862 | 0.149045 |

| 1000 | 35000 | 8 | 0.997606 | 0.163451 | 3.843967 | 0.150954 |

As only preliminary results were required to establish the validity of the concept, comparing VoteNet to a benchmark is out of the scope of the project, it is however useful to see how votes affect the results. A proposal is to train BoxNet on the same dataset, since BoxNet does not leverage the votes and only utilizes the features obtained from PointNet++, the following shows how the 2 networks compare to one another after several training rounds:

BoxNet gives a 2.4-2.7 mean loss when trained from 180-700 epochs

VoteNet gives 3.5-3.8 mean loss when trained from 180-1000 epochs

This is a preliminary indication that BoxNet performs better on the given dataset. An example of a pretrained BoxNet on the custom dataset is illustrated in Fig. (18):

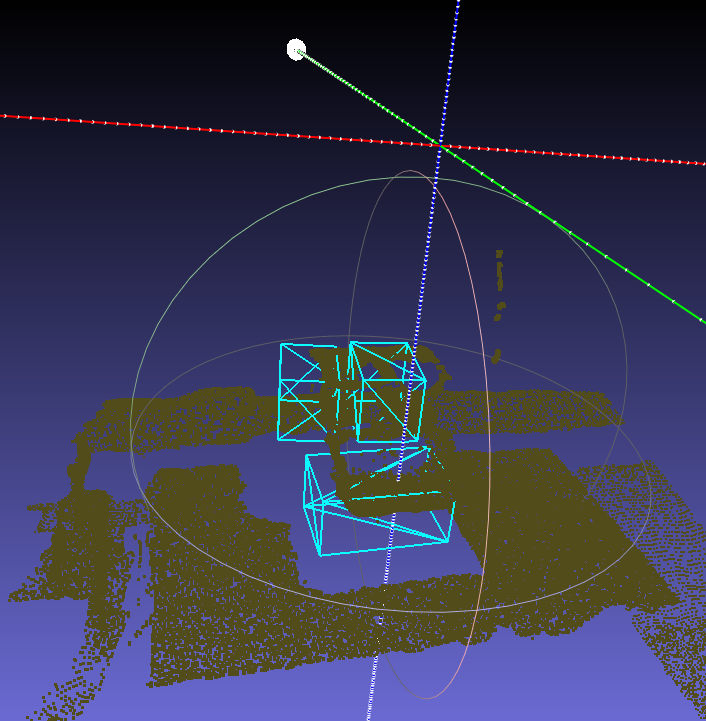
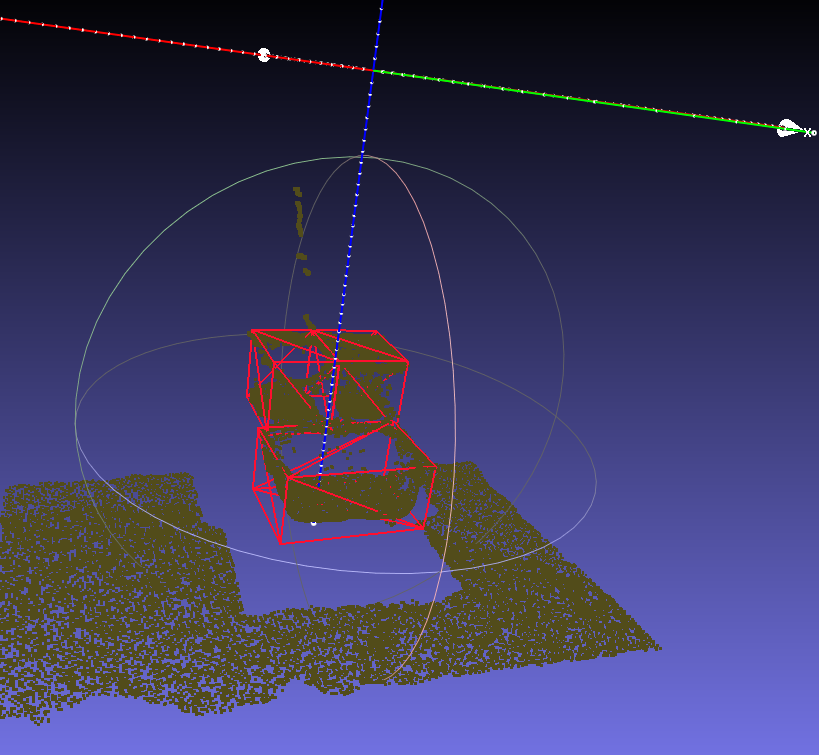


Fig. (18): The red labels represent the Ground Truth and the blue labels represent BoxNe’ts results.

Chapter 5: Conclusion, would be short I think

* Synthetic dataset or a combination
* Larger dataset
* Transfer Learning
* Using both image and point cloud features to train the network