# Introducing Explainability to Pointcloud Place Recognition for AI Assisted Shoreline Navigation

**Luke Michael Thomas, Student Number 905557**

30th September 2020

**Abstract**

Global localization via the use of image based querying/landmark detection has been a successful area within AI for a few years [2] - however, one issue within this field of research is the naturally large variances in locational imagery over time, either due to a change of weather, camera angle or if an image is taken at night rather than in the day. Due to this, many researchers have sought to perform localization based on LIDAR gathered 3D pointcloud data, the motivation being that because these clouds provide locational structure rather than appearance, variance becomes less of an issue as the factors mentioned before rarely affect overall structures.

We set out not only to identify a pointcloud-based place recognition model that can aid in the real world problem of shoreline vessel navigation (in the absence of GPS), but to do so in a way that provides a good deal of explainability to the user, such that they can interpret the machines decision. Explainability is an important aspect of applying AI to real world use cases, as often times the model may produce a less than certain result that, if wrong, can lead to a loss of trust and abandonment of the machine as well as potential accidents, in which case providing an explanation can help the user to better understand the models reasoning as well as display the cause of some fault.

Project Dissertation submitted to Swansea University
in Partial Fulfilment for the Degree of Master of Science


Department of Computer Science
Swansea University

## Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

November 17, 2020

Signed: L.M.Thomas

## Statement 1

This dissertation is the result of my own independent work/investigation, except where otherwise stated. Other sources are clearly acknowledged by giving explicit references. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure of this dissertation and the degree examination as a whole.

November 17, 2020

Signed: L.M.Thomas

## Statement 2

I hereby give my consent for my work, if accepted, to be archived and available for reference use, and for the title and summary to be made available to outside organisations.

November 17, 2020

Signed: L.M.Thomas

# Contents

# 1 Introduction

## 1.1 Context and Motivation

Owned by the United States Government, GPS localization is a key technique for the navigation of almost all forms of travel, allowing users to receive real time geolocation updates when in line of sight of several GPS satellites. For sea vessels specifically, GPS ensures that manned vessels can travel safely and efficiently, making the job of the ships navigator significantly easier and enabling autonomous vessels to travel without any human control.

Because of the need for vessels to be within view of multiple satellites however, if there are enough obstructions such as dense clouds or large mountains blocking a satellites line of sight the system may not be able to provide further geolocation updates for some time. In this case, a manned ship must then turn to a human navigator in order to determine the next course of action, making their job far more challenging. Autonomous vessels are in an even worse position as there is no one on board to perform the role of navigator, forcing the vessel to come to a halt or be placed under remote manual control, hindering it's autonomy.

This paper will cover a proof of concept to create an explainable AI-based GPS backup for sea vessels near visible shorelines, by scanning pointclouds of the shoreline via 3D LIDAR and passing them to a place recognition model in order to retrieve an approximate geolocation. Having an AI analyse nearby structure via LIDAR scans should be able to act as an extension of what a human navigator is capable of, in that it can more easily analyse the topology of the surrounding area in addition to the more typical landmarks that a human would use.

As mentioned in the abstract, the main motivation behind the use of LIDAR rather than a camera ties into one of the main key challenges of AI based place recognition, that being the tendency for locations to appear visually different depending on weather, season and time of day, which may cause an AI to not recognize locations consistently. LIDAR scans counter this issue because unlike a camera taking a 2D image, a LIDAR does not record colour values of nearby shorelines, instead producing 3D structural replicas in the form of pointclouds which is far more invariant to the changing of seasons/time, putting the system more on par with human navigators in identifying certain locations and perhaps even surpass them.

To make the model explainable for the end user, a method that we would be interested in applying is some form of pointcloud saliency mapping to the models output. This has been done to great success within the area of 2D image classification [19] by highlighting key areas of interest that influenced the output, it also falls in line with our human centred focus as we are currently aiming to

develop this AI in accordance with well-known responsible AI principles, which tend to include the need to make results explainable to the user.

Because a user study was not possible for this paper due to COVID, and because deciphering shapes from pointclouds is a good example of the Gestalt principles [31], we propose to use these same principles in order to determine whether the features/shapes highlighted by our saliency maps are also interpretable enough to provide explainability.

## 1.2 Objectives

• **Extraction of features from pointcloud data via deep learning:** In order to perform place recognition on pointclouds we must identify a model that can act as an equivalent to the Convolutional Neural Nets [12] currently used for extracting features from 2D image data, at the moment applying 3D convolution tends to be unviable due to the large increase in complexity and models that have attempted this have had to reduce data size to numbers as low as 30 x 30 x 30 [34] which would not be sufficient for representing the locations we wish to capture.

• **Obtaining a global feature descriptor from extracted features to perform place recognition:** Because the potential for variance between locational based data is so high, current methods such as SIFT [14] make use of more sophisticated global feature descriptors such as VLAD [3] to return only the most important and invariant features within images, so finding something that can perform this on our extracted pointcloud features is key.

• **Accessing a database of geo-tagged 3D pointclouds for place recognition training:** In order to train the model we need access to a set of geo-tagged pointclouds that we can use to determine the approximate location of an input pointcloud by matching it to another captured instance of the same location, preferably these clouds should also overlap so that our saliency mapping can highlight features present in both.

• **Applying saliency mappings to 3D pointcloud data for the purpose of making the AI explainable:** Although effective for 2D data, to our knowledge saliency mappings for pointcloud data are not a well researched area which is likely due to either the added complexity of 3D data or because 2D methods such as RISE [19] tend to work by reducing pixel colour values to zero whereas pointcloud points have no such values.

• **Enhancing identified methods via new model innovations:** Often times the most cited models within a particular field will have had new improved versions in the years since their discovery, therefore where possible it would be a good idea to include these in our final architecture.

## 1.3 Contributions

• **Explaining the results of pointcloud based place recognition models via the use of Saliency Mappings:** If this technology is to be used for real world problems as a form of decision support then it must be explainable in order to qualify as a responsible use of AI, which we hope to achieve through the use of saliency which to our knowledge has thus far only been used for explaining classification models.

• **Evaluating the interpretability of pointcloud saliency mappings using Data Visualisation and Gestalt Principles:** It is possible that even if the saliency mapping can be applied to our results, the initial visualisations may still leave much to be desired due to the addition of point depth, thus we propose to study the best way of visualising saliency for these models using data visualisation techniques while also making sure the results adhere to the gestalt principles.

## 1.4 Thesis Overview

• **Chapter 2 - Background:** We begin by taking a look at recent developments in pointcloud deep learning methods to identify a suitable architecture for our proposed AI, we also look into how place recognition had been performed in the past with classical feature extraction techniques before discussing newer trainable methods. After discussing the technical details of various machine learning architectures, we move on to explainability through the use of saliency mappings for both 2D and 3D data, provide a brief look into the gestalt principles and finally consider whether or not the project satisfies the requirements of responsible innovation.

• **Chapter 3 - Data:** We discuss the data that we will be using to train our architecture - including details of it's collection, format and any challenges associated with it.

• **Chapter 4 - Methodology:** Here we outline the details of our architecture, along with training and evaluation methods. We will also discuss explainability will be introduced to the architecture, including how it will be applied and evaluated.

• **Chapter 5 - Experimental Results:** This section will consist of the results of training our chosen architecture and applying explainability respectively, in addition to an evaluation of both.

• **Chapter 6 - Conclusion:** A reflection on the results shown in chapter 5 with respect to our initial expectations, where we will determine the overall success/potential shortcomings of the project.

• **Chapter 7 - Future Work:** A section that will go over various important aspects of the project that we would like to pursue going forward, will also include anything that was not able to be completed during the course of the Masters.

• **Chapter 8 - Acknowledgements:** Short section that will go over any outside sources that helped with the projects final outcome via the use of code.

# 2 Background

## 2.1 Introduction to Deep Learning

Before diving straight into deep learning on 3D/pointcloud data specifically, we start by introducing the reader to some of the main ideas of deep learning by going over two widely used models, MLP and CNN. These two models tend to make up the backbone of most machine learning architectures and will also appear when covering the more advanced models in the next section.

### 2.1.1 Multi-Layer Perceptron

One of the first ever ML models to see wide spread use, the MLP is an extension of the classic perceptron [24] which was an old method for performing binary classification by carrying out a simple formula of $f(x, w) = y$, where $f$ is a function with weights $w$.

The Multi-Layer Perceptron (AKA Neural Network) then introduced the idea of having intermediate nodes known as hidden units, which can be used to detect features by applying some form of activation function onto the input.
Weights of the model are now allocated to each connection between each part of the input and each of the hidden units to form a matrix.

Instead of using a fixed function $f$, the model now learns to estimate an approximated function $\hat{f}$ that returns a prediction close to the true output $y$ of each training input $x$

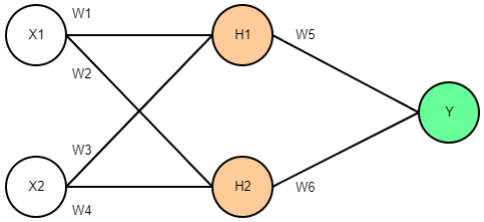


Figure 1: Basic MLP with two inputs $X1$ and $X2$, both of which pass their values to the hidden units $H1$ and $H2$, which then produce output value $y$, note how each connection in the graph has an associated weight value $WX$

As illustrated in Figure 1 learning occurs by performing what's known as forward propagation, where each hidden node applies an activation function to the sum of all it's weighted connections to the input layer, the sum of the hidden nodes weighted connections are then also calculated to form the final output Y.

### 2.1.2 Backpropagation

In order to automate the allocation of optimal weights for models such as the MLP, we must make use of what is known as a loss function.
In it's simplest form, the loss is just the difference between the predicted output of some input $x$ and it's true output $y$ but other more advanced methods can include mean squared error, log loss and maximum likelihood.

To optimise a weight we take the gradient of the loss with respect to each one and adjust it accordingly, known as gradient descent.
However, in order to perform this on MLP's which can potentially pass the input through multiple layers and weights we must use a technique known as Backpropagation [25].

In short, backpropagation applies the chain rule of derivatives [13], a mathematical method that allows one to obtain derivates of multi-layer functions, to the activation function of each hidden layer in order to obtain the partial derivatives for all weighted connections in the network.

### 2.1.3 Convolutional Neural Network [12]

Although a good, generalisable model that can be applied to most tasks, one area in which MLP's struggle in is analysing 2D image data, largely due to the added computational complexity.
This complexity stems from the fact that for an image with, say, 128 x 128 dimensions, there are now 16384 input nodes (one for each pixel), pass these on to 10 hidden nodes (which would be considered a very low number) and you now have 10 x 16384 or 163840 weighted connections!

Convolutional neural networks [12] are an extension of the MLP that introduce the idea of using convolutions in one or more of their layers, which is where a kernel containing multiple weights is applied to each member of the input in order to convert their values into a weighted sum of the input and it's surrounding neighbours, with the kernel size determining the neighbourhood size.

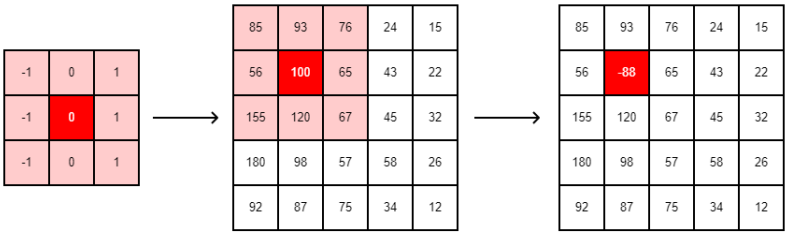


Figure 2: Example of a convolution operation, on the left is a 3x3 kernel with it's weights displayed, in the middle is the kernel being applied to the input pixel highlighted in bright red with it's neighbours also highlighted and on the right the pixel has been converted to a new value based on applying the kernels weights to the neighbourhood before performing a final summation.

Convolution can also be represented using the following equation, where I is a 2D image at pixel $i, j$ and K is a kernel of size $x$ by $y$:

$$I(i,j)K(x,y) = \sum_{x} \sum_{y} I(i+x, j+y)K(x,y) \tag{1}$$

Because weights are now tied to the kernel rather than connections, the number of weights in the network is reduced to the dimensions of the kernel multiplied by the number of different kernel filters applied per layer (i.e. 64 3x3 kernels equals 9 x 64 = 576 weights).

Kernels are also a natural fit for image data due to the fact that in previous years kernels with static weights have been applied to images in order to perform tasks such as edge detection, thus it is a good method for detecting local image based features that take the surrounding neighbourhood into consideration.

## 2.2 Pointcloud-based Deep Learning

In past years, CNN models such as VGG16 [28] and AlexNet [11] have been dominant in image based deep learning for tasks such as multi-class classification, however as more and more researchers are now shifting their focus towards newer, 3D datasets gathered using LIDAR there is a growing need for deep learning models that can analyse volume/pointcloud data.

Knowing how effective convolution networks have become, it would seem obvious to attempt to apply some 3D convolution kernel to this data, however the issue here is that 3D data introduces a cubic increase in computational time.

This is due to the dramatic increase in the number of weights per kernel which so far has rendered many proposed methods too time-consuming to be considered practical, leading to a situation where the highest resolution that can be

processed in any reasonable amount of time was found by the creators of 3D ShapeNets [34] to be 30 x 30 x 30 which introduces a new problem due to these samples often now being too small to provide an acceptable representation of complex shapes.

In this section, we will go over a series of methods that attempt to alleviate these issues in order to efficiently process 3D data for the purpose of deep learning.

### 2.2.1 Multi-view CNN [30]

Proposed by Hang Su and co. in 2015 [30], the Multi-view CNN performs 3D deep learning by opting to analyse multiple 2D projections of a 3D object, each of which providing a different angle of view, which are then passed to a traditional 2D-based CNN and the results of all of these are then aggregated to get an overall result.

Rather than trying to extract a shape descriptor of each input using 3D convolution, this approach instead extracts multiple 2D view descriptors, allowing us to make use of more efficient existing CNN models which we know provide superior performance to 3D equivalents and then pooling these view descriptors to get a single shape descriptor.

This way, the cubic computational increase of 3D convolution is bypassed, with computational time now increasing squarely depending on the projected image resolution, allowing the 3D object itself to keep it's original resolution thus retaining a far higher degree of graphical fidelity.

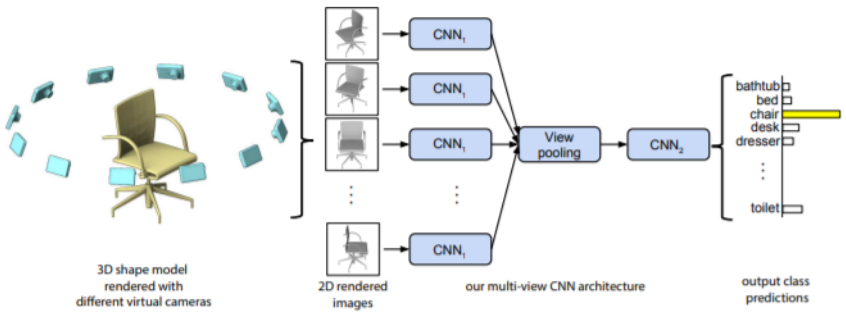


Figure 3: Virtual cameras are used to create multiple 2D projections of a 3D chair object, the results from passing these to a CNN are then pooled and passed through a second CNN, providing the final result. Figure taken from [30]

By calculating the saliency of each view to the overall result, the researchers were also able to find the optimum angle for performing classification on each object, with views showing the objects head-on tending to have the highest

saliency and those from the side having lower saliency.

This brings up an important question however, if all this architecture does is find the best projections of 3D data to 2D and applying traditional methods then have we actually leveraged the additional information 3D provides or have we just found a workaround?

Also, for our project we wish to find a model which can analyse 3D pointclouds of shorelines where most of the points are on the outer regions of the cloud and the centre is mostly empty, would this mean that many of the angles in Figure 3 would retrieve minimal information? In addition, although this method works well for classification of single objects, there is nothing to say that it would work well for more complex scenes with multiple features at varying depths.

### 2.2.2 Volumetric CNNs

Another improvement upon the 3D ShapeNets [34] model is a Volumetric CNN referred to as VoxNet [16] which instead of taking pointclouds as input directly, instead opts to generate what is called an occupancy grid of some fixed size, in this case 32x32x32.

To briefly explain, an occupancy grid is a technique usually employed by sonar [18] in order to detect areas of 3D space that are occupied, free or unknown, choosing to use the grid not only means that the original pointcloud input can be analysed at a higher dimension but the unknown areas returned by the grid give us a richer representation compared to raw pointcloud input.

Essentially, unknown areas are locations where there could be an object but the LIDAR (or other device) was not able to capture i.e. the side of a car facing away from a deployed LIDAR, this information is therefore useful as it can give a clue to an observed objects full shape.

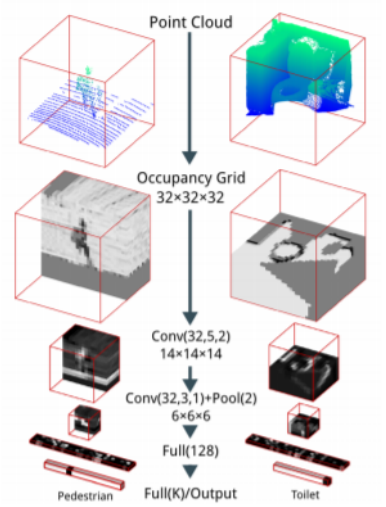


Figure 4: Pointcloud is first converted into an occupancy grid input, this is then passed to two separate 3D conv layers, the latter of which also pools the feature before being passed to an FC layer and finally producing a classification output. Figure taken from [16]

This method was found to be on par with ShapeNets while still being fully 3D based, with the most notable improvement being the drastically reduced number of parameters needed for the overall model, however the researchers were still not convinced as to whether or not the model could fully exploit the information available.

Also, richer as it might be the occupancy grid is still just 32x32x32 dimensions, meaning that attempting to reduce a more complex scene into such a representation would likely result in a lot of finer details being lost still.

At this point Multi-View CNN still appeared to be a superior choice to true volumetric based models for classification performance, as a response to this in 2016 new variations of the traditional Volumetric CNN were proposed by Charles R Qi and co. in order to make them perform on-par with the Multi-View method.

The first suggestion was to provide the 3D ShapeNet architecture with auxiliary training tasks on subvolumes of each training sample, motivated by the original models tendency to overfit the given training samples.

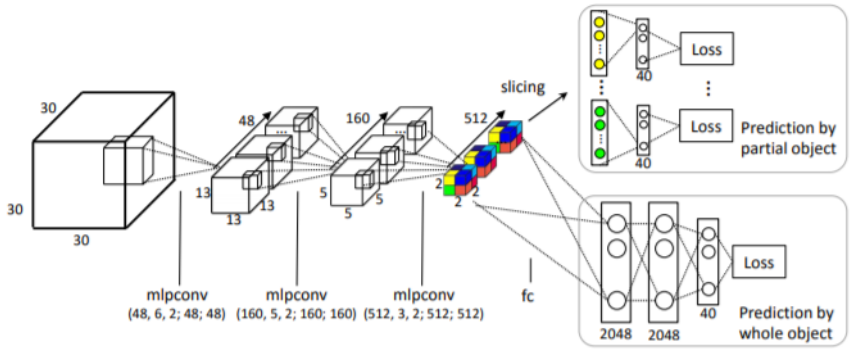


Figure 5: Visual representation of ShapeNet with the additional auxiliary training tasks visible, which in this example is done by slicing the extracted features in the final conv layer and passing them to a partial object classifier, forcing the model to take advantage of local features. Figure taken from [21]

The second proposal was to use anisotropic probing in order to achieve a 3D-to-2D projection of the volume data using an elongated kernel, which is able to better capture the internal structure of volume data and allows the architecture to scale to higher resolutions.

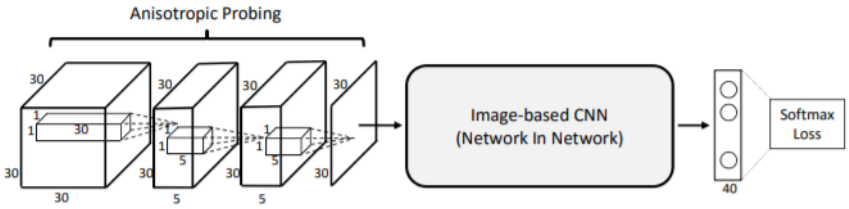


Figure 6: Elongated kernel slowly convolves the 3D input 'cube' into a 2D 'plane' projection. Figure taken from [21]

Both of these methods were found to match the performance of Multi-View CNN at the same resolution, however despite these innovations the researchers concluded that the efficiency of Volumetric CNNs was still bottlenecked by the input resolution, thus taking advantage of these improved local feature extracting techniques to gain a better understanding of complex scenes was likely not possible.

### 2.2.3 PointNet [20]

Taking into account the resolution issues of Volumetric CNNs, PointNet is a more recent alternative to both Volumetric and Multi-View, which acts as an improvement over the latter when given fully 3D tasks such as point classification and automated shape completion.

The architecture of PointNet is surprisingly simple and can take in raw point-cloud data directly as it's input, the model is designed to work in tandem with the unordered nature of pointclouds such that it is invariant with respect to different permutations, while also being able to extract local features based on euclidean distance and to be invariant to rotational/jitter transformations.

To achieve this, PointNet passes the input through various MLP layers (with additional input and feature transformations to further improve performance), with the features in the last MLP being aggregated through the use of max-pool as a symmetric function, resulting in the extraction of the clouds most informative points which can then either be used to form a global descriptor for classification or can be used to predict per point labels for segmentation.

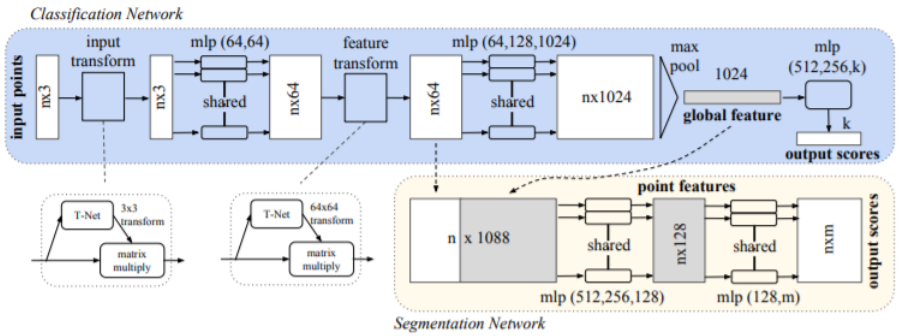


Figure 7: Visual representation of the architecture described above, note that the model can actually output a segmentation classification in addition to an overall classification, the architecture for the former being highlighted in yellow. Figure taken from [20]

Due to PointNet's ability to extract 3D point based information instead of the image descriptors retrieved by Multi-View CNN while also being able to maintain good pointcloud resolution, it is arguably the first model to be able to perform 3D deep learning on complex scenes such as LIDAR scans of rooms, which is showcased in the paper during the semantic segmentation examples.

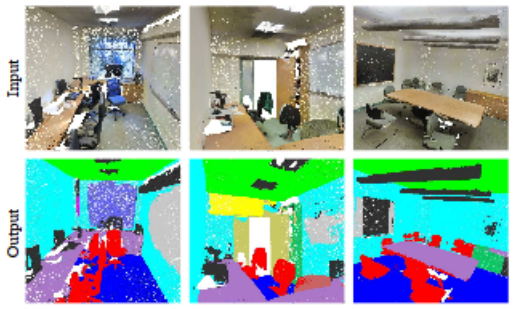


Figure 8: Examples of PointNet's segmentation network applied to pointclouds of indoor office rooms, from one scan the model is able to identify multiple features within the scene (i.e. Chairs, Walls, etc.) with a good level of accuracy, all while keeping the data purely 3D. Figure taken from [20]

Because PointNet takes in raw pointcloud data directly (i.e. 4096 points per cloud) which is then passed to a simple MLP layer, computational time is drastically reduced compared to previous methods as the time taken now increases linearly depending on the size of the pointcloud rather than squarely or cubicly, meaning the pointcloud can potentially be analysed at it's full resolution.

#### 2.2.4 PointNet++ [22]

PointNet [20] marked a big breakthrough in the world of deep learning on pointclouds, due to it's ability to efficiently aggregate spatial features of each point within the cloud into an a global descriptor of the cloud itself.

However, not long after being published the same team who had built PointNet came out with an extension called PointNet++ [22] which sought to not only analyse all points within a cloud, but also exploit local structures similarly to CNNs which are able to progressively capture larger scale features through the use of multi-resolution hierarchies.

As such, PointNet++ is a hierarchical network that works by partitioning pointclouds into overlapping local regions by the underlying distance metric of the cloud itself, similarly to a CNN architecture these are then grouped into larger units which can be processed in order to retrieve higher level features.

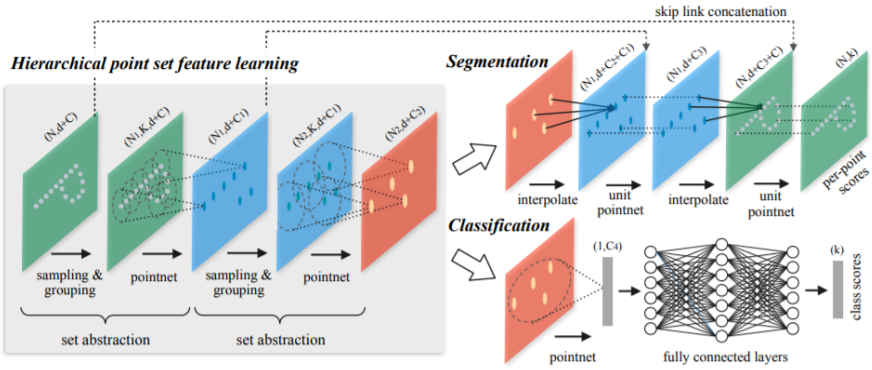


Figure 9: Illustration of PointNet++ architecture, feature learning consists of a sampling and grouping layer in order to extract local subsets which are then followed by a traditional PointNet layer which learns the higher level features contained within these subsets. Figure taken from [22]

Note that in the illustration above, the actual operations performed within the "sampling & grouping" layers consist of an iterative farthest point sampling of the input set followed by a grouping of points into subsets using a ball query.

With these additions, PointNet++ was able to achieve a new level of state-of-the-art performance on benchmark datasets, including ModelNet40 [34] which is a dataset that all of the previously talked about methods have also been tested on.

## 2.3 Obtaining Feature Descriptors for Place Recognition

Place recognition is a special form of image based retrieval, where a new 'query' image is matched to an existing image that depicts the same location, for place recognition specifically these queries are matched with instances from a geo-tagged database, allowing the user to receive an approximate geolocation.

The main challenge in this area identified by previous researchers is the fact that most locational images tend to contain a lot of non-distinctive features [4], including generic office buildings, pavements, trees etc. and, as we mentioned in our introduction, a key issue is the change of areas over time and potentially the change of angle or position from which an image or scan of a location is taken from at two separate instances.

Because of this, one of the main challenges of place recognition is providing representations of the input data that show local invariant features.
In this section we will go over two methods of obtaining feature descriptors, the classic SIFT based approach and the newer trainable NetVLAD layer.

### 2.3.1 SIFT [14]

SIFT, standing for Scale Invariant Feature Transform, is a feature extraction method for images that works in four main stages.

The first stage works by searching over all scale and image locations using a difference-of-Gaussian function to identify scale and orientation invariant areas of an image in order to retrieve several locations of interest, the second stage then fits a model to determine the location and scale of these locations, filtering them down to a set of keypoints based on stability.

Then, at the third stage, orientations are assigned to each keypoint based on their gradient direction and the transformations applied to these keypoints up to this stage are kept and used in further operations to ensure invariance.

Finally, in the fourth stage, keypoint descriptors are obtained by measuring local image gradients at the selected scale around the keypoints which can then also be transformed in order to allow for some level of distortion/illumination variance.

Figure 10: An example of SIFT applied to a query image (top right), based on a set of previous training images (top left) the method is able to identify these areas as keypoints within the test image, allowing it to be used for place recognition

What is important to note, is that although it can be used as part of a pipeline, SIFT itself is not a machine learning model and as such it cannot be 'fit' to a specific set of training images, typically the features learnt by SIFT are aggregated using methods such as bag-of-words [29] or VLAD [3] and then

matched to some nearest neighbour image.

### 2.3.2 NetVLAD [2]

A much more recent alternative to SIFT-based place recognition that seeks to leverage the power of existing CNN's such as VGG16 and AlexNet, NetVLAD is a generalised version of the existing VLAD layer that can be plugged into a CNN architecture and trained via backpropagation.

When given a set of image features, NetVLAD learns $K$ cluster centres and outputs a $(D*K)$-dimensional vector which forms an aggregated representation of the local feature descriptors, also known as a VLAD descriptor.

Because the VLAD descriptor has high dimensionality it is expensive to compute, thus the NetVLAD layer also makes use of a dimensionality reduction layer followed by L2 normalisation to get the final image descriptor.

What allows NetVLAD to learn over time compared to regular VLAD is tha instead of using K nearest neighbours in order to perform a hard assignment of local features to cluster centres, NetVLAD instead uses a soft assignment where weights are applied to each input feature based on their distance to the closest cluster centre relative to all of the others, a convolution operation is then performed using these weights and softmax activation is applied to assign each feature to a cluster.

Using backpropagation, NetVLAD is able to tune these weights over time to ensure the generation of optimal global feature descriptors.

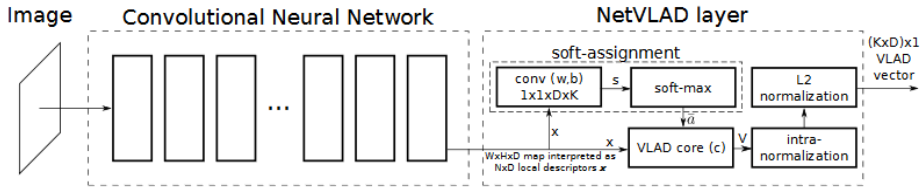


Figure 11: Visual example of the trainable NetVLAD architecture described above. Figure taken from [2]

## 2.4 PointNetVLAD [1]

At the beginning on this project, we believed that in order to identify an architecture capable of pointcloud place recognition we would have to build one ourselves from the previously mentioned resources. However, during our background reading we found out that another group of researchers had already developed an architecture that performs the desired task by taking the NetVLAD

layer and, instead of plugging it into a CNN, used PointNet instead to produce the model known as PointNetVLAD [1].

By combining PointNet and NetVLAD together, a new place recognition pipeline is created whereby we take a 3D pointcloud $P$ as input, pass on it's feature descriptor $P'$ to NetVLAD which learns a set of $K$ cluster centres, from which the model returns a $(D * K)$-dimensional pointcloud feature descriptor.

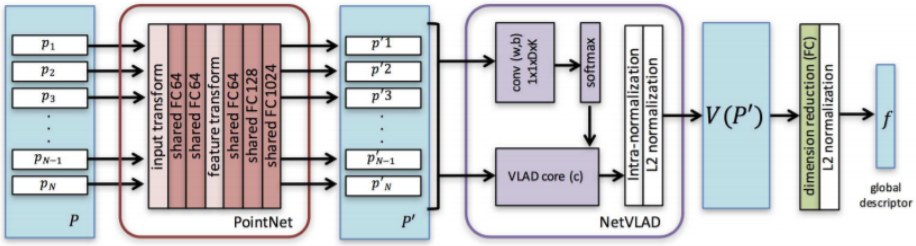


Figure 12: Visual representation of the PointNetVLAD pipeline. Figure taken from [1]

Instead of using a database of images, here the task of place recognition is performed by taking a pointcloud reference map, dividing this reference into several submaps, and training the network to recognize each of these such that it can identify which submap each test instance belongs to.

The motivation behind this model is the same as our initial motivations for finding a pointcloud based model, that being the fact that pointclouds are naturally more invariant than images for retrieval/recognition tasks, because although locations tend to change in visual appearance at different times (i.e. day/night, seasons, etc.) they do not tend to change structurally, meaning we may be able to rely less on data augmentation methods to achieve invariance.

PointNetVLAD trains itself by using a modified version of what is known as triplet loss referred to as "lazy triplet", whereby a set of training tuples $T(P_a, P_{pos}, P_{neg})$ with $P_a$ representing an anchor pointcloud, $P_{pos}$ representing a structurally similar pointcloud and $\{P_{neg}\}$ representing a set of structurally dissimilar pointclouds are provided to the model as input.

The loss then works by minimizing the distance between $P_a$ and $P_{pos}$ while maximizing the distance between $P_a$ and some pointcloud from $\{P_{neg}\}$ called $P_{neg_j}$, the two of which are denoted as $\delta_{pos}$ and $\delta_{neg_j}$ respectively, with squared euclidean being used to calculate distance.

This process can also be defined by the following equation, where $\delta_{neg_j}$ is based on the $P_{neg_j}$ closest to $P_a$, $[...]_+$ represents the hinge loss and $\alpha$ is a constant parameter used as the margin:

$$L_{lazyTrip}(T) = max_j([\alpha + \delta_{pos} - \delta_{neg_j}]_+) \tag{2}$$

A second loss known as lazy quadruplets is also used, whereby we also attempt to maximize the distance between $P_a$ and some randomly sampled $P_{neg_k}$, in order to prevent situations where lazy triplets is reduced to finding the distance between $P_{neg_j}$ and some pointcloud $P_{false}$ that is structurally dissimilar to it.

This results in the following new equation, where $\beta$ is another constant parameter used as a margin:

$$L_{lazyQuad}(T, P_neg*) = max_j([\alpha + \delta_{pos} - \delta_{neg_j}]_+) + max_k([\beta + \delta_{pos} - \delta_{neg*_k}]_+) \tag{3}$$

The only difference between the authors "lazy" losses versus their original counterparts is the fact that these losses use max operations rather than sum, meaning the contribution of each negative cloud diminishes compared to a single hardest negative, which actually leads to a faster training and more discriminative function.

Essentially this architecture provides us with everything we were looking from at the start, thus we have chosen to make this our architecture of choice heading in to the Methodology/Experiment sections. The github page for the project is also extremely helpful as it provides all of the code and data used in the paper, thus this model can be used off-the-shelf so to speak.

## 2.5 Explainable AI

From this point of the background section onwards, we switch our focus from the technical aspects and history of various pointcloud/place recognition ML models and shift the conversation towards the more human-centred aspect of this project, that being the application of explainability to the model.
Before we discuss our proposed method for achieving this in the form of saliency mappings, we give a brief overview of Explainable AI and it's importance.

One of the first heavily cited papers covering the topic of Explainable AI was "What do we need to build explainable AI systems for the medical domain?" [9], where it was identified that a key issue of using AI and ML for real world tasks was that they do not have an explicit declarative knowledge representation meaning they do not possess an innate sense of understandability.

This can cause serious legal issues in cases where a uncertain prediction on the AI's part leads to some accident occurring, at which point it is not certain whether the blame lies with the AI itself or with the user.

Therefore, providing a sense of explainability during times of uncertainty (such

as a model having only middling confidence in it's prediction) can help, as now the user themselves is given the opportunity to analyse the machines reasoning to reach a more informed human made decision, helping to avoid accidents as well as making it clearer as to whom the blame falls onto in situations where they do still occur.

However, reducing the importance of explainable AI to legal issues would be a huge simplification, as there is also the matter of overall user experience and trust to consider.

To explain these, we cite Derek Doran et al.'s paper [7] where they divide AI models into four groups: Opaque, Interpretable, Comprehensible and Explainable.

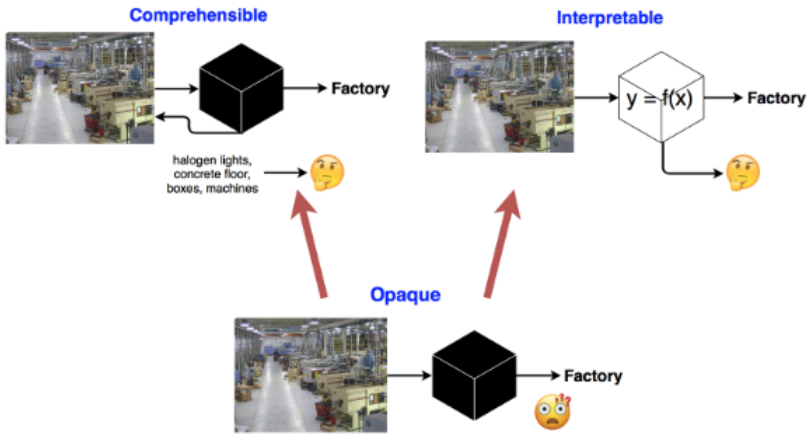


Figure 13: Visual representation of a comprehensible, interpretable and opaque model. Figure taken from [7]

For a model to be opaque is for it to be purely black box, meaning not only is it not explainable but the user is given no information on it's technical workings (i.e. The inner mechanisms are withheld by the company that made it).

The issue with opaque models is that they effectively demand a sort of blind trust from the user, offering no explanation of their results and giving the user no opportunity to study it's inner workings, meaning that if the model periodically produces an incorrect prediction this trust can be eroded, as researchers have noted in the past trust in AI is usually dynamic [27] so even if one has tended to give correct predictions in the past a sudden error can still quickly lead to it's abandonment.

Interpretable models are defined as those which provide the user access to documentation of it's inner workings, including open source ML models whose

associated research papers are freely available.

However, we would argue that requiring all end users to be able to understand these models is unreasonable, machine learning is an inherently complex field and as such expecting the average person to study enough to gain insight is not realistic. Because of this for certain users an interpretable model might as well be opaque, introducing the same issues as before.

The last two types, comprehensible and explainable are both defined as models that provide some form of reasoning to the user, with the former providing a set of simple symbols (i.e. a set of keywords that describe what features of an image lead to a classification) for the user to reason with and the latter making use of a more complicated reasoner in order to provide higher level explanations of what is happening.

Arguably both methods are sufficient, the only difference that separates them in the figures present within this specific paper are that the explainable model states what the comprehensible model is also saying much more explicitly, enhancing the user experience.

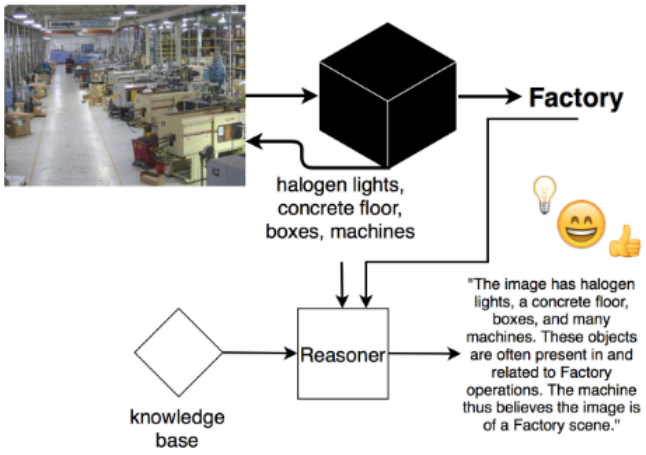


Figure 14: Visual representation of a more advanced explainable model that makes use of some knowledge base in combination with comprehensible keywords to provide a very high level explanation. Figure taken from [7]

With these additions, in situations where the machine fails the user can now at least understand where the model may have gone wrong as well as being able to disagree with the model, enhancing the human-AI interaction and potentially keeping the users trust in the model intact.

## 2.6 Saliency Mappings

Now that we have a good understanding of Explainable AI and it's importance with regards to ensuring trust and aiding in legal disputes, we move on to analyse our proposed method of explainability to apply to PointNetVLAD, Saliency Mappings.

A saliency mapping is an explainability technique that, rather than providing the user with keywords, instead visually highlights areas of an image that had a large affect on the models final decision based on what are known as saliency values.

These values reflect the importance of each part of a machines input during the calculation of it's loss, researchers have found that applying a colour mapping to these values and overlaying this over the original input tends to produce a visualisation that points the user to various "hot zones" in an image that lead to the final decision.
In other words, we make the model more comprehensible (as defined in [7]) by providing a data visualisation based on importance of input.

There are many methods for applying this to 2D so for this section we will simply go over one of the more impressive methods, RISE [19], before going into the inherent differences between 2D image and 3D pointcloud data that necessitate newer methods specific to the latter.

### 2.6.1 RISE for 2D classification saliency mapping [19]

Designed specifically for the purpose of explaining black box models, RISE makes use of randomized upscaled binary masks applied to an input image in order to estimate the overall importance of each pixel in the image towards the classification outcome.

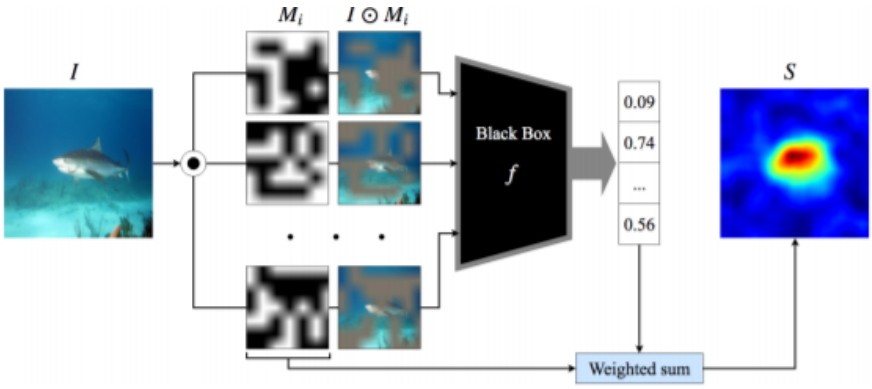


Figure 15: Showcase of RISE's architecture, a series of binary masks $M_i$ are applied to the input image $I$ before being fed to some black box model, a saliency map is then generated from a linear combination of the masks, where the score for the target class being the weight. Figure taken from [19]

Not only is the process of RISE simple to understand, but the saliency map is highly understandable for end users by clearly highlighting areas of high importance and low importance in red/blue respectively.

When compared to previous attempts such as GradCAM [26] and LIME [23], RISE was found to outperform both when using a "deletion" evaluation, whereby the important pixels were removed in order to see how much the classification score drops (indicating that those pixels were in fact highly important).
RISE's classification score dropped more sharply than either of the previous methods, suggesting that it's mapping was able to better localize the important areas within an image.

What stops methods such as RISE from being applicable to 3D pointclouds is due to a fundamental difference between image and pointcloud data, that being that the fact that each pixel in an image has both a position and an RGB value, whereas points only have the former.
This means masking groups of points is simply not possible without removing them, meaning whatever architecture they are passed to would have to be dynamic in terms of the input shape, adding additional complexity to the model.

### 2.6.2 PointCloud Saliency Maps for 3D classification saliency mapping [35]

The first method for efficiently extracting per point saliency from pointclouds builds upon the critical subset theory proposed by the makers of PointNet [20], which suggested that in each classification there exists a subset of points such that their inclusion always results in the same outcome.

Here, calculating saliency is done via the use of a point-shifting method rather than point-dropping, as the latter would require a brute force process whereby every possible combination of points is passed to determine importance, whereas the former simply shifts each point towards the centre of the cloud which tends to be ignored during classification and therefore has a similar effect to simply removing the point.

By doing this, the change in loss for a point can be approximated by the gradient of the loss under a spherical coordinate system.

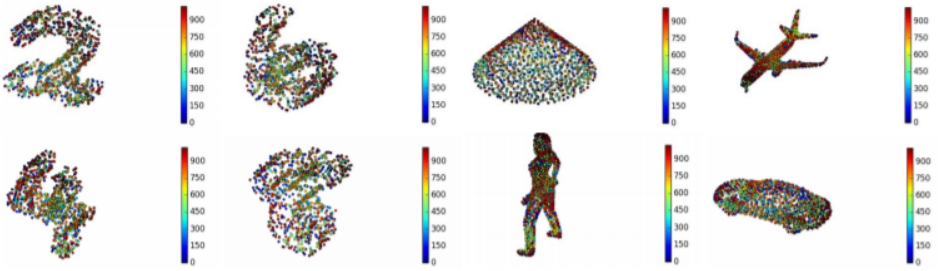

Figure 16: Examples of the saliency map being applied to pointclouds as a colour map. Figure taken from [35]

This method was found to outperform the critical subset theory and it's corresponding point drop, as dropping points that were deemed to have high saliency in this model reduced classification accuracy further and dropping low scoring points kept the accuracy the same if not better, meaning that for now these mappings are the best methods for displaying per point saliency/importance.

## 2.7 The Gestalt principles

The Gestalt principles [31] are a set of properties that allow the human eye to create shapes even when they are not fully visible, these are instantly applicable to pointclouds due to the fact that they are in themselves an example of the principles at work, being a set of disconnected points that make up a more clear shape.

To understand these principles, we provide a few summaries of the most accepted ones:

• **Similarity**: The ability to group objects that are similar in shape, colour etc. despite them not being next to eachother, this can be achieved through saliency by highlighting only the most important pointcloud structures in certain colours.

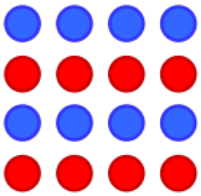

Figure 17: Examples of the similarity principle, due to their different colours, the mind will mentally group together all blue/red points even if they are not next to each other.

• **Continuation** When viewing any shape, the human eye tends to follow the smoothest path, therefore areas with high saliency will be more perceptible to the end user in cases where all highlighted points line up smoothly rather than being erratic.

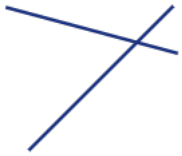

Figure 18: Examples of the continuation principle, both lines can easily be followed by the viewer due to their straight paths despite overlapping.

• **Closure** The idea that the human brain will often 'fill in' missing parts of some shape, this is very important for pointcloud saliency in particular as often times important structures may not be highlighted in there entirety, so we can leverage this principle in order to naturally fill in certain gaps.

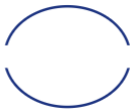

Figure 19: Examples of the closure principle, despite having gaps in the middle many people would still perceive this as a circle.

• **Proximity** When a group of objects are packed together with high density, it provides a greater sense of proximity and the human brain will group them together, therefore saliency mappings should not be too sparse as it may be difficult to see what features they are trying to represent.

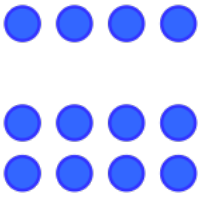

Figure 20: Examples of the proximity principle, most viewers would separate the points on the top and bottom into two groups due to the larger gap between them.

• **Symmetry and Order** The tendency to perceive large shapes as being symmetrical, whether or not we can ensure this principle is uncertain.

Figure 21: Examples of the symmetry principle, due to both brackets being symmetrical to one another, viewers will observe these as a pair.

## 2.8 Responsible Innovation

In accordance with the human centred values of Swansea University's EPSRC CDT, as we are developing AI for real world use it is important that throughout development we take the time to focus on responsibilities associated with designing such systems.

If we use Google's Responsible AI Principles as a guide based on our proposition so far, we can make some initial judgements on how the final product

may satisfy these needs along with potential issues:

• **AI should be socially beneficial:** Assuming our model matches the strength of the original PointNetVLAD, this could allow for more efficient and safe travel of manned and autonomous vessels during GPS downtimes can potentially aid any currently existing maritime activities, and, more specifically, could help in the investment of autonomous vessels by making them more consistent to navigate, all while being interpretable by a human user.

• **AI should not create or reinforce unfair bias:** For the most part issues of bias should not be a concern for this project as we are not dealing with any personal data, the only issue that could arise is the model tending to favour submaps with more distinct features, which can be remedied through the use of the lazy quads loss.

• **AI should be built and tested for safety:** For the masters portion of the project we will not be doing any real world testing, with the result being more of a 'proof of concept', however taking into account the potential safety hazard of faulty navigation we will hopefully be able to produce a model that can identify the vast majority of submaps with high certainty, with uncertain cases being handled by an explainable saliency map provided to the end user.

• **AI should be accountable to people:** As mentioned, explain ability and interpretability of the model will be handled via the use of saliency mappings provided to the end user which they can then decide to agree/disagree with, important to note is that this model will be a decision support and thus the final decision of how to navigate should still be handed to a human navigator.

• **AI should incorporate privacy design principles:** The only situation in which this principle would apply to our project is if the models predicted geolocation could be intercepted somehow, otherwise as we are only dealing with location based data privacy should not be much of a concern here.

• **AI development should uphold high standards of scientific excellence:** In terms of the models raw performance we must make sure to at least match that of the original PointNetVLAD model.

• **AI should be made available for uses that accord with these principles:** The intended final product serves as a backup to a technology that already exists, that being GPS, therefore it should not be possible for it to be used for any purposes that GPS can't already be used for.

# 3 Data

## 3.1 Details

Unfortunately, outside of automated car research large databases of 3D LIDAR scans are quite rare and due to limitations imposed on us due to COVID creating a reference map based on a significant portion of the UK coast is out of the question.

As a result we have decided to fall back on the same benchmark datasets used for PointNetVLAD [1] which is made up of four datasets, the first being based on the Oxford RobotCar database [15] and the other three consisting of in-house LIDAR generated datasets of a University Sector, Residential Area and a Business District respectively over multiple different runs.

Because we are training the model more for experimental purposes rather than performance, we have decided to stick to just the Oxford RobotCar database in order to simplify things.

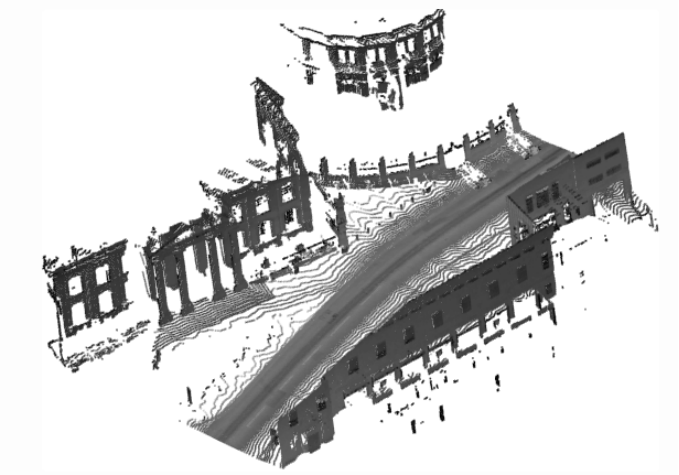

Figure 22: Example of a street converted into pointcloud form from the Oxford RobotCar website [15]

On the github page for PointNetVLAD the overall reference map for each run has already been divided smaller submaps which are then stored in a folder in binary file format where the filename is equal to the timestamp at which they were scanned, folders are generated on a per run basis of which there were 45 in total.

Each submap covers 20m of car trajectory, and, in the case of submaps intended for training, were scanned 10m apart form one another thus they are

not mutually disjoint whereas testing submaps are mutually disjoint.

For each run, GPS locations of each submap within the run are stored in a single csv file which contains the northing and easting coordinate for each timestamp.

## 3.2 Challenges

The main challenge with using this dataset as a backup to a potential future LIDAR dataset of shoreline scans, is that we don't know whether or not the model will act/perform differently once such a dataset is introduced during the follow-up PhD project.

For example, there may be a big difference between how the PointNetVLAD model will handle pointclouds of shoreline topology versus how it handles pointclouds from Oxford RobotCar database which typically contains features such as terrace houses.

A second challenge is the fact that because the testing submaps are mutually disjoint, there will likely be no overlap between each query map and it's positives, thus their highlighted portions after the application of saliency will not be shared between each other and therefore it may be worth sticking to just the training data for saliency evaluation.

# 4 Methodology

## 4.1 Data

### 4.1.1 Pre-processing

Interestingly, the datasets provided by the creators of PointNetVLAD on github are already pre-processed, to briefly explain what has been done essentially all the submaps have had any uninformative ground planes removed and were then downsampled to 4096 points, with these points being shifted and rescaled within the range [-1,1].

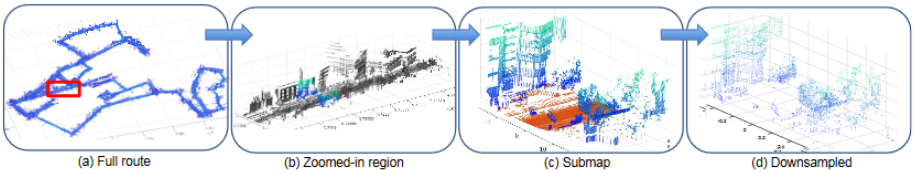


Figure 23: From left to right: Training reference map viewed in full, zoomed view of reference map with submap highlighted in blue, image of submap before removing ground plane, submap after ground removal plus downsampling. Figure taken from [1]

For generating training tuples, the authors of PointNetVLAD defined similar pointclouds to be those that were less than or equal to 10m apart according to there UTM coordinate and dissimilar pointclouds to be those that are 50m apart or more using the same method.

In the code, this is done by applying a KDTree operation (with average leaf size of 40) on the submaps based on their northing/easting values in the corresponding csv files before performing two queries on the tree, one with radius 10 to define all positive submaps and another with radius 50 to define negative submaps.

Each query submap is then stored as part of a query dictionary, which contains the query itself, a list of all it's positives and a shuffled list of all it's negatives. These dictionaries are stored in pickle files to be loaded later.

To achieve the train/test split, several disjoint regions were defined for each dataset in order to define multiple smaller test maps contained within the overall reference map.

## 4.2 PointNetVLAD

### 4.2.1 Architecture

As the outcome of this project hinges on whether or not saliency could be applied to the existing PointNetVLAD architecture, we opted to stick as close as possible to the training parameters used for the original.

However, most of the code for PointNetVLAD provided at the authors github page was written to work with some version of tensorflow 1.0, hence we have taken it upon ourselves to piece together a version that is compatible with the more recent 2.0 version of tensorflow as well as with the keras API.

This version makes use of newly made keras implementations of PointNet and NetVLAD respectively, the former being constructed by keras team themselves and the latter being taken from an ongoing project to convert the LOUPE library [17], a tensorflow toolbox implementing several learnable pooling methods, from tensorflow 1.0 to 2.0.

With access to these two parts, putting the two together in order to simulate the PointNetVLAD model was relatively straightforward and the original code used for the actual training and evaluation process were easily converted into jupyter notebook format, mostly by replacing the old feed dict method with more recent .fit/.predict functions.

As for the layers of the architecture, the model follows the example of PointNetVLAD's architecture seen in Figure 12, this means the input is initially passed through a transformation network to generate a 3x3 transformation which is then matrix multiplied with the input, the purpose of which is to canonicalize the data before feature extraction in order to ensure invariance to geometric transformations [20].

The input is then passed through two fully connected layers with 64 hidden nodes, which here has been simulated using the Conv1D operation with kernel size of 1 and relu activation, the extracted features are then provided to a second transformation net to get a 64 x 64 transformation which is then matrix multiplied by the features for the same reasons as before.

Before passing the features to the NetVLAD layer, we apply three Conv1D layers with 64, 128 and 1024 filters respectively. Recalling that in PointNet these would then be passed to a maxpooling function to extract the most notable features, for PointNetVLAD this has been cut and replaced with the NetVLAD layer to retrieve a global feature descriptor.

Our NetVLAD layer is then used to produce a global feature descriptor from the local descriptor learnt from previous layers, which is then L2 normalized

and reshaped into the correct output format.

Despite wanting to use more recently developed models such as PointNet++ as part of our architecture, the only implementation we could find that worked with keras required access to a CUDA root directory, which, because the code for this project was being run on a shared Swansea University machine, we did not have access to.

### 4.2.2 Training

Training/Testing tuples are provided to the model by taking a query submap along with a number of positives and randomly sampled negatives from the query dictionary, in this case 2 positives and 18 negatives as outlined in PointNetVLAD's supplementary material. Note that the reason for using 2 positives is so that when calculating the loss, we can simply take the closest one to the query, ensuring more model stability.

In addition to this, because we opt to use the lazy quadruplet loss in order to avoid the model becoming biased towards simply finding the distance between the query and some negative submap, we also provide an additional negative for each tuple.

The loss is calculated in the same way depicted as in Equation 3, however in order for it to work with keras which requires a true output $y$ along with a predicted $y'$, we had to pass the query, positive and negative vectors to the loss as $y'$ before using tf.split (with the split being hardcoded to fit with the input of 1 query + 2 positives + 18 negatives + 1 other neg) to retrieve each component of the output with the true $y$ being a dummy value passed to the loss during training.

Each epoch consists of 21711 steps, due to the original code setting the step number to the amount of training queries divided by the number of batches, the latter of which we had to set to 1 due to hardware based constraints. At each step the model is fit to a single training tuple which is provided using the steps mentioned above, with the models optimizer being Adam and the learning rate being set to 0.00005.

Every 200 steps, the model is evaluated against 5 test set query tuples and after every 3000 steps the weights of the model for that epoch are saved.

This process remains unchanged until after epoch 5, where two things occur. First off, the learning rate now begins to reduce after every 5 epochs, on github there was a comment in the original code saying that it should be halved but upon inspection the actual operation consists of multiplying the base learning rate (0.00005) by $0.9^{epoch//5}$, with the minimum rate being 0.00001, this is essentially a way of preventing the model from overfitting over the course of

extended training.

The second thing that occurs after epoch 5 is that now after every 700 steps a set of training latent vectors are generated for all query tuples in the dictionary, these are then used during the retrieval of training tuples to ensure that a number of submaps within the negative groups are now hard negatives, which can be found by building a KDTree on a set of random negatives latent vectors and then querying them against the query submaps latent vector.
The reason we do this is because it helps the model reach convergence at a faster rate.

### 4.2.3 Evaluation

For the purposes of our project, which is to evaluate the effectiveness of applying saliency to PointNetVLAD, evaluating the model itself mostly acts as a check to make sure that it is working as intended, as the model itself has already been thoroughly evaluated in it's original paper [1].

Essentially, the evaluation of the model works by using the trained model to retrieve the latent vectors of all the queries within the database, then taking query vectors from a certain run of the test set along with all submaps from another run of the overall reference map, if after applying a KDTree to the submap vectors there is a true neighbour pointcloud to the query within 25m then it is considered to be successfully localized.

After performing this operation for test queries in all of the runs the final metric used is a top 1% recall, of which the original model achieved 80.31 using Lazy Quadruplet Loss on Oxford alone.

The reason that top 1% is a good metric for this model is due to the fact that it only takes into consideration the recall for the top 1% of examples with respect to the quadruplet loss, meaning that obvious mismatches are not counted as part of the overall metric allowing us to focus more on how accurate the model is when performing at top accuracy.

## 4.3 Saliency Mapping

### 4.3.1 Application

Once the model is trained, we can test saliency map application using the method described in Pointcloud Saliency Maps [35], since this method was originally applied to single pointclouds inputs intended for a classifier, we decided that it would make sense to apply the method to each pointcloud in our query tuples individually.

The reason for this is because in order to calculate the gradient of the loss

per point under a spherical coordinate system, we need to find both the sphere core and radius which for the former involves finding the median x,y and z position of all points in the cloud, therefore it makes sense to calculate these separately due to the clouds varying shapes.

Once the saliency values have been calculated for all pointclouds, we then propose to visualise them by applying a colour map similar to the continuous jet colourmap used in RISE [19].


Figure 24: Image of several the commonly used continuous colour maps, where colours on the right and left hand side of each represent the minimum and maximum data range of whatever value they are being mapped to respectively.

It may also make sense to experiment with a more sequential colour mapping scheme, to see if dividing points into more clearly defined lower and higher importance groups makes the results any more clear.

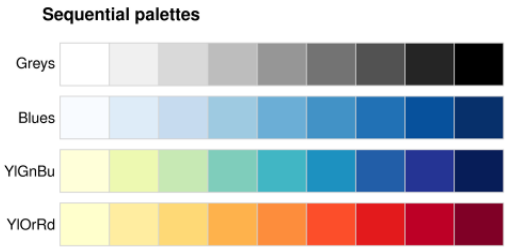


Figure 25: Several examples of sequential colour maps, essentially gradual changes are removed in favour of more clearly divided groups.

In addition to simple mappings, we believe it would be a good idea to experiment with some of the critical subset ideas discussed in the Pointcloud Saliency Maps paper [35] by simply removing points of low importance so that only a "critical subset" of points is visualised, potentially removing visual clutter.

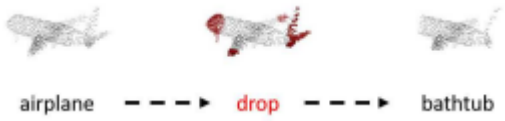


Figure 26: Table taken from [35] showing the highlighting and removal of a critical subset of points from a cloud for the purpose of adversarial training evaluation, if we were to do the opposite and remove the non-critical subset it may make for an interesting visualisation method.

Another aspect of the visualisation to take into account is rotation, as point-clouds are typically visualised using 3D scatter plots if viewed from one angle it is difficult to discern where each point falls within the overall space due to the lack of any perceived depth on a computer screen [33].

Because of this we must at least provide the end user multiple views of each pointcloud at different angles if not a fully interactive display that allows for full free rotation, in order to more easily view the structure contained within the plot.

Finally, because we are applying these mappings to the overlapping training query tuples to find areas where both query and positive pointcloud features potentially overlap, it may be interesting to see if plotting both clouds in the same visualisation (with both being positioned at their original GPS coordinates) results in them having clear overlaps with one another.
Because overlapping two clouds could result in a lot of clutter in cases where they are particularly close to one another, it may be a good idea to augment this visualisation with the critical subset method.

### 4.3.2 Evaluation

Once the visualisations appear on screen, we can quantitatively evaluate them by comparing them to the gestalt principles in order to see if the mapping is able to highlight areas of importance in a way that a human can easily perceive.

Similarity and proximity will be two key principles that the mappings should abide by, in the case of the former we would hope to see a good degree of separation between points with low to high saliency values and, in the case of the latter, it would be beneficial if points of high importance are grouped together in such a way that a perceivable feature/structure visible in both the query and positive cloud (or one that appears in a negative cloud that isn't in the query) is highlighted.

It may be that the principles of closure and continuity can be leveraged in order to aid in perceiving partially highlighted features, although they should not have to be relied on too much in order to fully comprehend what the visu-

alisation is attempting to display.

For the critical set visualisation, the evaluation would largely be the same although in this case we would hope for there to be at least one or two perceivable structures within the critical set that maintain good proximity as a set of sparse points would likely be difficult for an end user to piece together.

Evaluating the overlapping technique should be far simpler, as the only thing we truly need to evaluate is whether or not the query and positive do in fact overlap in areas of similarity, removing clutter from both using the critical set method may also help with this evaluation in particular.

# 5 Experimental Results

## 5.1 PointNetVLAD

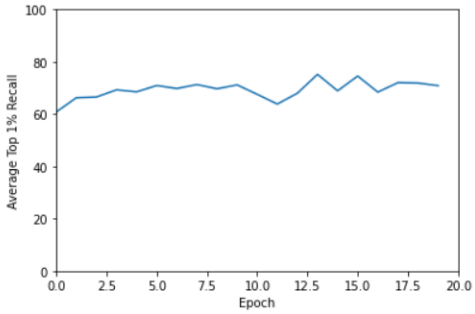


Figure 27: Plot of the Average Top 1% Recall of our PointNetVLAD over the course of 20 epochs.

After training our version of PointNetVLAD it was able to perform at what we would consider to be an acceptable level by the 20th epoch, where it was able to obtain an Average Top 1% Recall of 70.1%. However if we compare this to the 80.31% achieved by the original model with lazy quadruplet loss it is a significant decrease in performance and certainly not optimal.

The training process is also suboptimal, tending to vary up and down across each epoch making slight overall improvements until a large dip to 63% at epoch 11 to the highest result of 75% at epoch 13, from which point the result varies a bit more wildly until epochs 18 through 20 where we stabilise around 70%.

It is very likely that this is either due to certain hyper parameters within the original model being lost during the conversion to tensorflow 2.0 or it could be that because the architectures we pieced together have only been recently developed they may be suboptimal compared to their original counterparts in

some way.

What's important about this result however is that we are relatively confident that our PointNetVLAD architecture is at least working correctly, thus we can test our saliency mappings with relative confidence.

## 5.2 Saliency Mapping

### 5.2.1 Initial Mapping Results

Once PointNetVLAD had finished training, we preceded to visualise a number of randomly selected training query tuples by calculating the saliency values of all the points in each individual cloud in the tuple and applying the standard "plasma" colour map (out of personal preference) to the points based on their exact saliency values.

Important to note is that in all cases we were able to drastically cut down the number of clouds visualised per tuple, due to the fact that the majority of negative cloud samples, along with the worst positive, tended to receive saliency values of zero for all of their points, thus offering no important information.

For the worst positive this is self explanatory, the lazy quadruplet loss only uses the best positive for loss calculation so it makes sense that the other would receive no saliency value, if anything this serves to reaffirm the fact that the mapping is working as intended.

As for the negatives, it appears that the model tends to place importance on just a handful of negative pointclouds when calculating loss, which also falls in line with our discussion of the lazy loss variants earlier where the authors of PointNetVLAD stated that by using max function in the loss rather than sum, the contribution of each negative cloud diminished compared to a single hardest negative.

However, an issue with this is that in cases where the model achieves a loss of zero by finding an exact positive match for the query tuple, the similar features cannot be highlighted due to all saliency values also being zero, but this isn't a massive issue as most users will likely recognize that both query and positive clouds are almost exactly the same.

On the next page is an example of our initial colour mapping results based on the raw saliency calculations:

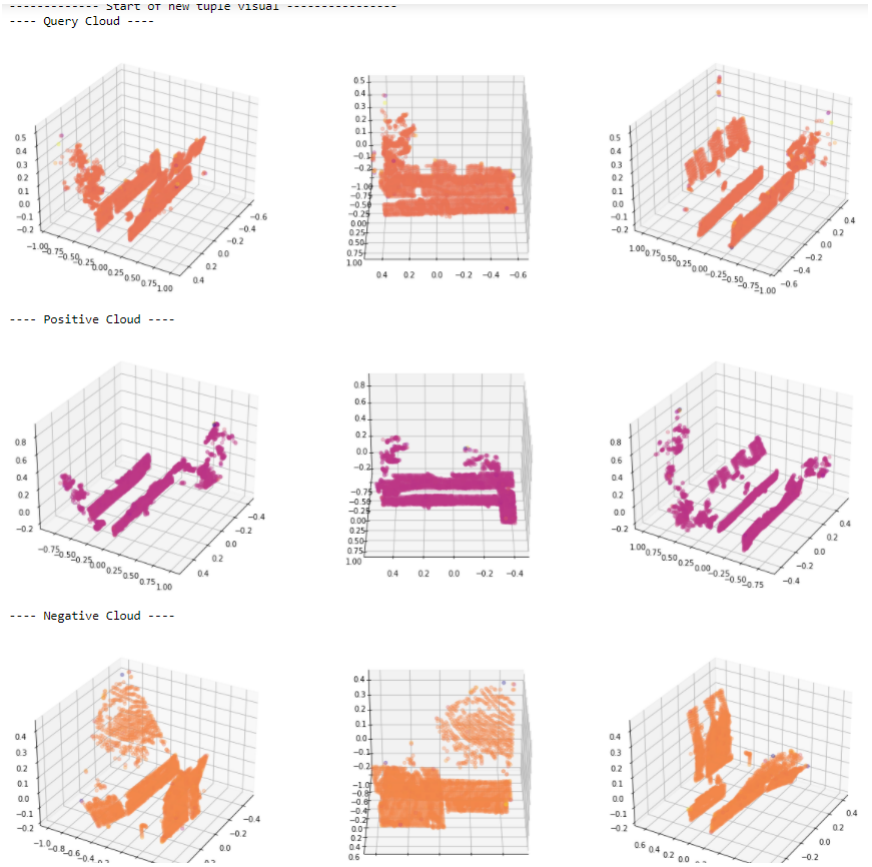


Figure 28: Visualisation of all salient clouds within a sampled training query with a continuous saliency colour map applied. Each row contains a separate cloud visualised at angles of 30, 90 and 210 degrees from left to right respectively.

As seen in Figure 28, the continuous colour scale unfortunately fails to highlight similar features between the query and positive clouds as well as dissimilarities between the query and negative clouds. Instead, it appears that applying saliency to PointNetVLAD's output highlights the fact that the model tends to award a consistent level of importance to all points within clouds that are deemed to be integral to the overall loss.

If we look at why this visualisation fails by comparing it to the gestalt principles, then the main issue is that the principle of similarity is not being leveraged in order to highlight key areas of higher saliency within each individual pointcloud, instead all the points are effectively put into a single colour group representing the overall saliency of the cloud itself, which is not very informative as we already know that these clouds were integral to the loss.

Because of this single failure, assessing the mapping based on proximity, continuity and closure is largely pointless due to the fact that the clouds have essentially taken on their original forms with the only difference being a single colour applied to all points.

However, mapping colour to raw saliency is thankfully not our only option, if we take a look back at the Pointcloud Saliency Mapping [35] paper there were some figures that visualised the map by highlighting only points from the critical set in red with all other points shown in white, which served to clearly highlight important shapes within each object that lead to a classification.

Taking inspiration from this, we created an alternative data range by grouping points within each cloud into ordinal categories based on the top % of the saliency values they fall into and then applying a sequential version of the "plasma" colour map based on this, which we believed would help to ensure that the key areas/areas of low importance within each cloud themselves can clearly stand out...

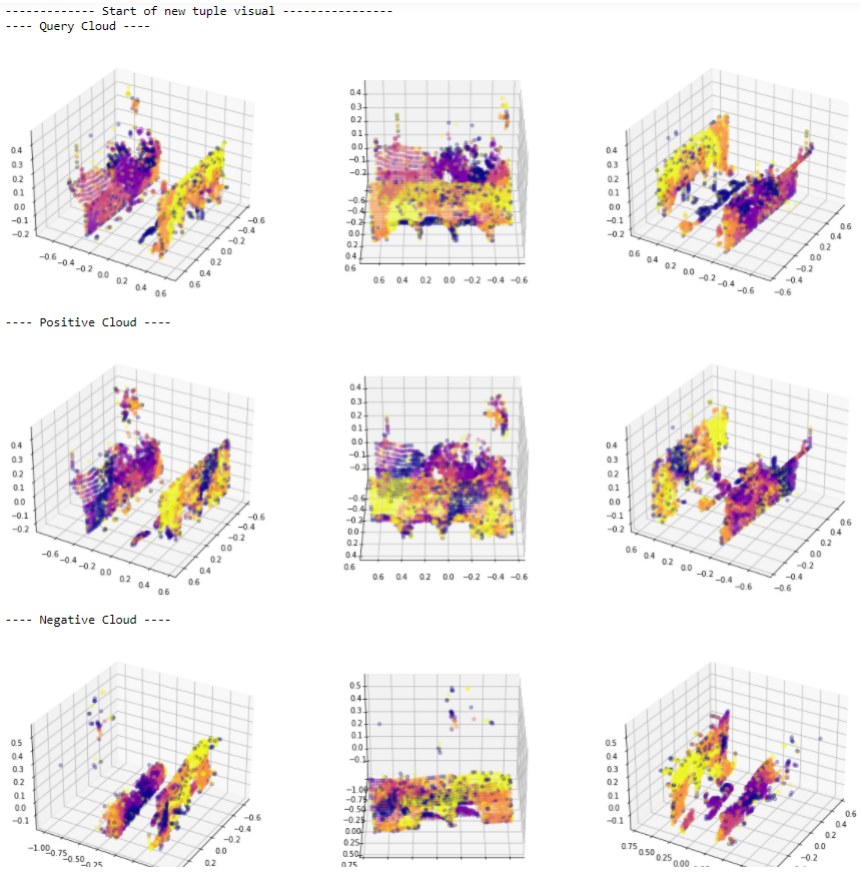


Figure 29: Visualisation in the same arrangement as the previous figure, a sequential colour mapping has been applied to each cloud by setting the colour values of the points within the top 0-20%, 20-40%, 40-60%, 60-80% and 80-100% to 1, 0.75, 0.5, 0.25 and 0 respectively and then applying the "plasma" colourmap to achieve a sequential effect.

This mapping was much more successful than the result for applying a continuous mapping to the raw values, as now we can see that within both the query cloud and positive clouds the most highlighted features tend to be visible within the other, in this case two groups of points have both been highlighted in orange/red (indicating that they belong to the second and first highest top saliency % groups) between the query and positive cloud.
The negative cloud is also clearly highlighting features that are not present within the query cloud.

Interesting to note however is that the query and positive are not highlighting the exact same features within one another, instead they appear to highlight different features visible within the other respectively, which on one hand allows

the user to see two examples of why the clouds have been paired but on the other hand does introduce some inconsistency.

If we evaluate the example given based on the gestalt principles, we can see that the principle of similarity is now being used to much greater effect to show which areas are of low, moderate and high importance. The principle of proximity also seems to be satisfied, as each group in the colour map tends to be closely grouped together allowing key areas to be revealed in full.

Because changing the range of our data to top % was so successful, we wanted to try and go back to the idea of a more continuous mapping by making another sequential mapping that maps each point to the top % they fall under, this time from points that fall within the top 1% to those that only fall within the top 99% (meaning their saliency value and thus importance is lower relative to all the other points) in increments of 1, which, because there are now a vast number of smaller groups, should maintain the illusion of being a continuous mapping.

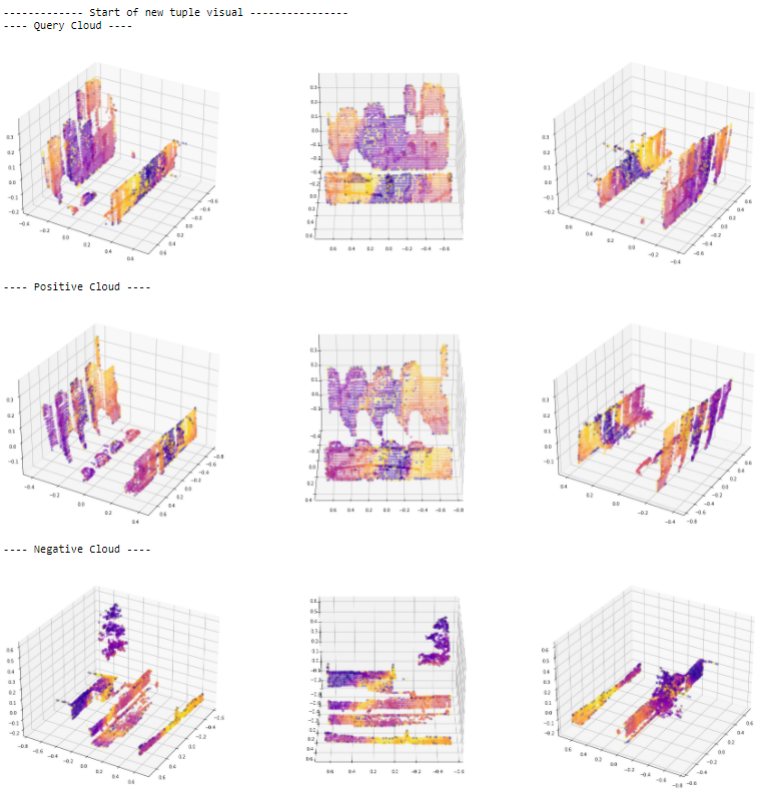


Figure 30: Visualisation displaying the alternative top % based sequential mapping from 1 to 99 described on the previous page, with "plasma" still being the colourmap of choice.

In our opinion using the top % metric to produce this semi-continuous mapping is far more successful in reflecting which points have higher/lower saliency values than the initial attempt and succeeds in the same areas as Figure 29, highlighting noticeable similarities/differences between the query and positive/negative clouds.

Because highlighted areas continue to be closely grouped together around certain features (in Figure 30 structures such as building faces and diagonal rooftops are quite visibly highlighted) this version mostly retains the sense of proximity and similarity of the previous mapping.

In addition however, we would argue that the now more gradually shifting colour gradients create a greater sense of continuity, as the user can now perceive the shift from low to high importance much more smoothly.

### 5.2.2 Visualising Critical Sets Based on Saliency Mappings

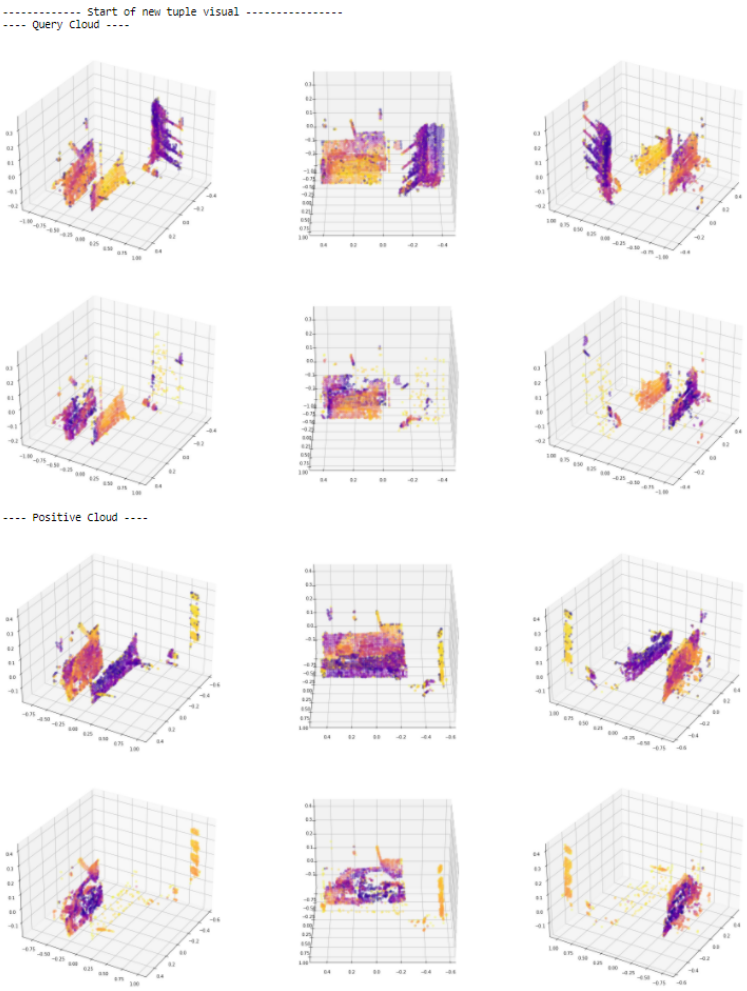


Figure 31: Visualisation displaying the same saliency mapping seen in Figure 30 but now with additional critical set version of the query and positive clouds displayed beneath the original

Now that we know our method of calculating per point saliency can be successfully visually reflected by dividing the points into the highest top % they fall into and applying a standard colourmap, we can begin to experiment further by first re-introducing the critical set ideas presented in [35].

To do this, we calculated the saliency values and ordered them into top % groups as described before but now we also create another version of the cloud

that cuts out all the points that only fell into the top 51% and below (essentially leaving all the points that represent the upper half of the calculated saliency values), the full and critical set version of the query and positive clouds were then visualised for comparisons sake, this can be seen in Figure 31 above.

By removing the points with lower importance we can now see the most highlighted areas with a bit more clarity, in addition the critical set version of the query cloud specifically appears to reveal a number of points of high importance that were previously difficult to see in the original, together these points appear to act as a more sparse representation of the clouds overall structure.

It is likely then that during feature extraction the model likely reduced large sections of low importance down into sets of key points, whereas structures of high importance seem to contain a higher concentration of points with large saliency values.

Between the two we would argue that the positive cloud more successfully highlights the most discriminant similarity between itself and the query, that being the elevated wall-like structure seen in both which is almost entirely highlighted in red in the positive cloud's full mapping.

Do note as well that the reason the colourmap does not stay consistent between the two versions of each cloud is because of the change in data range (i.e. Areas whose saliency values resulted in them being mapped to the colour orange may appear as green/yellow in the critical set due to them now falling into the mid/lower half of the overall value range).

Because once again the query and positive cloud appear to highlight differing areas of similarity between one another, we thought it might be interesting to merge the critical sets of both into a single pointcloud to see if they 'slot' together to create something similar to the original query...

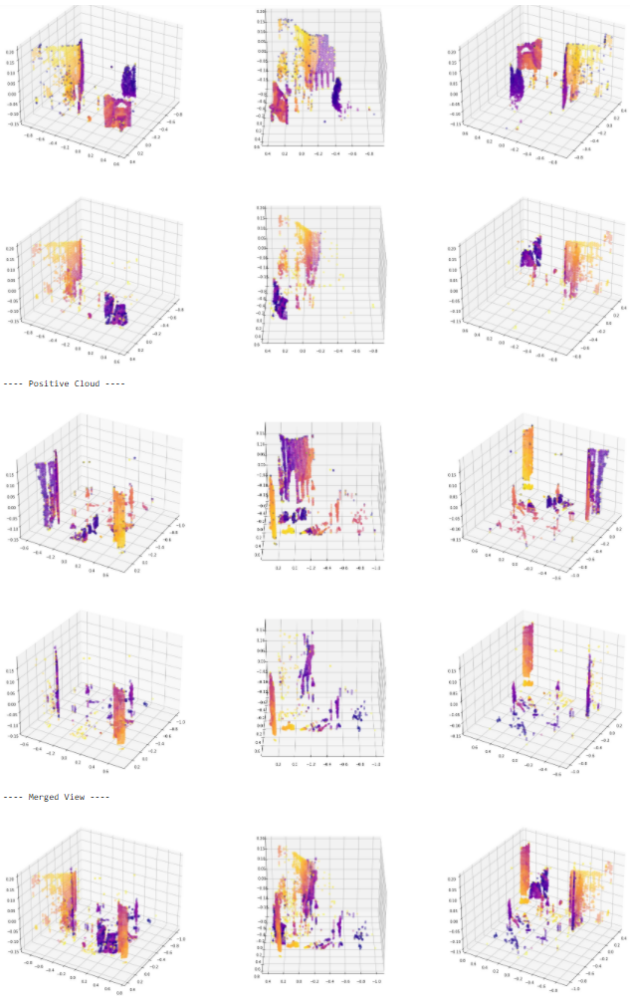


Figure 32: Another example of a query and positive cloud being visualised with a full and critical set version, this time with a final visualisation at the bottom where both critical sets have been merged into a single pointcloud

As you can see, because both sets serve to reduce their respective clouds down into the two separate sets of similarities reflected by each ones saliency map, the merged version of the two at the bottom ends up being extremely visually similar to the original query cloud at the top.

# 6 Conclusion

In conclusion, after setting out to find an architecture capable of performing pointcloud based place recognition which we could then successfully apply an informative saliency mapping to, the project has largely been a success. This is thanks in huge part to the efforts made by Mikaela Angelina Uy and Gim Hee Lee in developing PointNetVLAD [1], a model which essentially provided us with everything we were looking for, including the use of the PointNet architecture for efficient pointcloud deep learning and NetVLAD for learning optimal global feature descriptors.

Once we found and upscaled this model to tensorflow 2.0 using code from various researchers (Each of whom will be mentioned in the Acknowledgements section), we were able to get the model working on the more recent Keras API, albeit with some performance loss which we may set to fix in our future work.

The second most important discovery was the Pointcloud Saliency Map technique proposed by Tianhang Zheng and co. [35], which we know works correctly due to specific interactions with our input such as giving a saliency of zero to the worst positive during loss calculation.

Once the gradients of the models loss are obtained w.r.t the input, applying the method was incredibly straightforward (Requiring only 4 lines of code in python) and although the results were not instantly noticeable when visualising the raw saliency values, when a new top % based mapping was used the visualisations ability to reflect areas of importance/similarity became more visually apparent, although it was interesting that the query and positive clouds would highlight different similarities respectively.

In addition to this, applying a filter on the points based on which do or do not belong to a defined "critical set" such that only the former are visible allowed for the removal of visual clutter in the form of uninformative features, and the experiment we carried out with merging the query and positive clouds critical sets in order to see if their different similarities could "slot together" to create something similar to the original query also worked in several cases.

# 7 Future Work

In terms of future work much of what we are going to talk about are things that we wanted to cover during this project but could not due to constraints. Firstly, due to hardware constraints we were unable to update the PointNetVLAD architecture via the inclusion of the newer PointNet++ [22] deep learning method, therefore this is something we would definitely like to explore during the follow-up PhD as many papers have cited PointNet++ as a complete upgrade to PointNet.

The second was a user study, which we neglected to mention earlier in the paper due to our complete inability to carry out because of the current COVID outbreak separating us from the majority of researchers and staff at the UKHO, although we were able to evaluate some of our saliency mappings objectively good qualities according to data visualization and gestalt principles it is of the utmost importance that we also share these results with our potential end users, which could especially aid in determining which colour scheme is the most informative or whether we could make use of some hybrid between those visualised.

A key part of Andy Kirk's book on data visualisation [10] is designing the visualisation to suit the audience, however at the moment we are unsure what our audience wants and, because this may be their first time observing a saliency mapping, it may be the case that they will not initially know this themselves, thus some back and forth user studies will be crucial going forward.

Thirdly is data, in general although there are a handful of excellent pointcloud dataset examples such as Oxford RobotCar [15] the actual quantity of these is low and one specifically aimed at collecting shoreline pointclouds from the point of view of a sea vessel was simply not available.

If UKHO is willing to use it's resources to generate several runs of some local shoreline which we could then compile into a single dataset, we will be able to get better insight into what topological features of shorelines stand out most to the model as well as a look into how the less steady nature of sea vessels due to moderate to high tide affects pointcloud retrieval.

Fourthly, we would like to go over a type of model that we were only made aware of after completing our model - that being the Dynamic Graph CNN [32] which is an exmaple of what is known as a geometric deep learning model [6]. Geometric deep learning is a field focused on extracting features from data that is inherently non-Euclidean, with the two major ones being graph based information and Manifolds (aka Meshes).

In short, by using geometric deep learning methods we can extract features from higher order 3D data regarding things such as curvature, DGCNN specifically does this through the use of a new layer developed by the authors called Edge-

Conv, which can extract local neighbourhood features from pointcloud data by computing edge features between points in the cloud, effectively treating them like nodes in a graph.

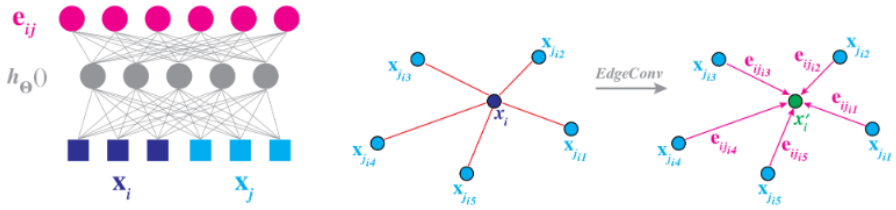


Figure 33: Visualisation of how EdgeConv works, starts by computing edge features from input pairs before aggregating the edge features associated with each vertexes identified edges. Figure taken from [32]

Not only could this provide a further raw boost to model performance (DGCNN has been found to be on par if not better than even PointNet++), but it would be interesting to see how the different approach affects our saliency mapping results, perhaps because it would be taking in more geometric based features complex shapes would now be more clearly highlighted with even less outlying high saliency points.

Finally, if all of the above are achieved the next step would be to not only perform submap based localization but to also regress the vessels true location within the submap, which if combined with SLAM [8, 5] technology could be used to create real time local pointcloud maps of the current shoreline, where the ships coordinates can be continuously updated based on SLAM once a correct location has been identified with the methods presented in this paper.

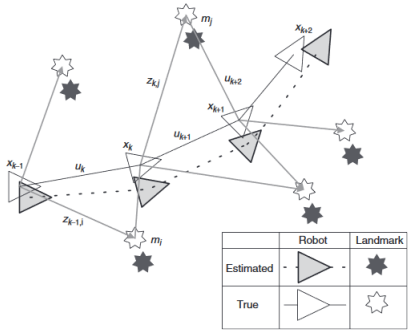


Figure 34: Example of the typical SLAM problem, as an object (in this figures case, a robot) moves through an environment made up of several landmarks we can approximate it's general location in real time based on how the landmarks shift due to motion of the object. Figure taken from [8]

# 8 Acknowledgements

First off, as far as acknowledgements go with developing the tensorflow 2.0 PointNetVLAD architecture, thanks go to David Griffiths on keras for their implementation of PointNet at https://keras.io/examples/vision/pointnet/ and shamangary on github for their keras implementation of the LOUPE library which included NetVLAD, found at https://github.com/shamangary/LOUPE_Keras.

In terms of training, generating training tuples, evaluating and accessing data for PointNetVLAD, major credit goes to the code provided by Mikaela Angelina Uy on github available at https://github.com/mikacuy/pointnetvlad, which effectively supplied us with everything we needed to use PointNetVLAD as an off-the-shelf machine learning model.

Thirdly, credit to Tianhang Zheng for providing the code for the Pointcloud Saliency Mapping technique on github at https://github.com/tianzheng4/PointCloud-Saliency-Maps.

Major thanks to project supervisor Mike Edwards, who provided crucial guidance and feedback throughout the project.

# References


[1] M. Angelina Uy and G. Hee Lee, "Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4470–4479.

[2] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5297–5307.

[3] R. Arandjelovic and A. Zisserman, "All about vlad," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2013, pp. 1578–1585.

[4] R. Arandjelović and A. Zisserman, "Dislocation: Scalable descriptor distinctiveness for location recognition," in *Asian Conference on Computer Vision*. Springer, 2014, pp. 188–204.

[5] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE robotics & automation magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[6] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[7] D. Doran, S. Schulz, and T. R. Besold, "What does explainable ai really mean? a new conceptualization of perspectives," *arXiv preprint arXiv:1710.00794*, 2017.

[8] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (slam):part i the essential algorithms," 2006.

[9] A. Holzinger, C. Biemann, C. S. Pattichis, and D. B. Kell, "What do we need to build explainable ai systems for the medical domain?" *arXiv preprint arXiv:1712.09923*, 2017.

[10] A. Kirk, *Data visualisation: A handbook for data driven design*. Sage, 2016.

[11] A. Krizhevsky and I. Sutskever, "H. geoffrey e.,"alex net,"," *Adv. Neural Inf. Process. Syst*, vol. 25, pp. 1–9, 2012.

[12] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, pp. 143–155, 1989.

[13] G. Leibniz, "Memoir using the chain rule," *Cited in TMME*, vol. 7, pp. 321–332.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[15] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[16] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 922–928.

[17] A. Miech, I. Laptev, and J. Sivic, "Learnable pooling with context gating for video classification," *arXiv preprint arXiv:1706.06905*, 2017.

[18] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proceedings. 1985 IEEE international Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 116–121.

[19] V. Petsiuk, A. Das, and K. Saenko, "Rise: Randomized input sampling for explanation of black-box models," *arXiv preprint arXiv:1806.07421*, 2018.

[20] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[21] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5648–5656.

[22] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108.

[23] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.

[24] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[26] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE Conference on Computer Vision*, 2017, pp. 618–626.

[27] K. Siau and W. Wang, "Building trust in artificial intelligence, machine learning, and robotics," *Cutter Business Technology Journal*, vol. 31, no. 2, pp. 47–53, 2018.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[29] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *null*. IEEE, 2003, p. 1470.

[30] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE Conference on Computer Vision*, 2015, pp. 945–953.

[31] D. Todorovic, "Gestalt principles," *Scholarpedia*, vol. 3, no. 12, p. 5345, 2008.

[32] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[33] C. O. Wilke, *Fundamentals of data visualization: a primer on making informative and compelling figures*. O'Reilly Media, 2019.

[34] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[35] T. Zheng, C. Chen, J. Yuan, B. Li, and K. Ren, "Pointcloud saliency maps," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1598–1606.