
*International Master's Thesis*

# 3D Context of Objects
## A prior for Object Detection and Place Classification

Nima Behzad
*Technology*

*Studies from the Department of Technology at Örebro University*
Örebro 2021

# 3D Context of Objects

A prior for Object Detection and Place Classification

*Studies from the Department of Technology at Örebro University*


Nima Behzad

# 3D Context of Objects

**A prior for Object Detection and Place Classification**

**Supervisors:** **Dr. Andrzej Pronobis**
**Dr. Todor Stayanov**

*Title:* 3D Context of Objects
A prior for Object Detection and Place Classification

ISSN 1650-8580

# Abstract

Contextual information is helpful for object detection and object-based place representation. 3D data significantly helps to capture geometrical information about scenes. In this work, a feature descriptor for object context in full 3D pointclouds of places is introduced together with a method to extract features and build the context model.

The proposed model is evaluated in experiments on pointclouds from different types of places which include different object categories. Results show the promising ability of the model to predict the possible context of objects in pointclouds or complete 3D maps of an environment.

Among various applications for this, the author suggests object context models to be used in place categorization and semantic mapping and discusses a method for it. To the knowledge of the author, this work is unique regarding its use of full 3D pointcloud of scenes and also introducing this descriptor to be used to represent places.

# Acknowledgements

Great Thanks to my supportive supervisors, Andrzej and Todor, for their help and advices, also Professor Lilienthal, who introduced me to the world of computer vision.

Special thanks to my parents, my lovely fiancee, Shaghayegh, and my sister for their kindness and support.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1
# Introduction

Object detection is one of the most common problems in computer vision, and in most cases, solutions for it are complicated or/and expensive. Therefore finding ways to bring this cost or complexity down can be very valuable. This costly problem is a straightforward task for us as humans, so looking into the methods humans use to find an object or recognize it can assist in proposing a solution for an AI system.

When a human is looking for an object, based on experience, he limits the search area to places he expects to find the object of interest. If the search area is a familiar place, based on their memory, they directly go to the last location they saw the object. If it is a new place and no prior knowledge about that place is available, he looks for sites that he logically expects to find the object.

This can be easier to imagine if we assume that the room is dark and our agent needs to touch objects to find them. For instance, if the object is a Telephone, the agent first look for a table or a desk in the room and then start moving his hand slowly on the desk/table's surface (context for the object of interest) to find the object that is the most similar to what he has in memory of a Telephone (object detection). Only in the worst-case scenario and without the knowledge of the object one would perform a thorough search of the whole scene. In other words, the strategy depends on the amount of information about the target object.

*Torralba et al.* in [22] pointed out that many experiments show that the human visual system uses contextual information to facilitate a search for objects. They propose the use of context for object detection using 2D images. They take two parallel approaches to predict the image regions likely to be focused on by human observers, naturally searching for objects in real-world scenes; One computes local features (saliency), and the other computes global (scene-centered) features.

A similar behavior could be implemented on a mobile robot or any AI system. A piece of valuable prior knowledge for an object detection/recognition problem could be semantic knowledge about places and contexts where the ob-

ject of interest can be found. This can reduce the search time and increase the chance of success and correctness of the results. The role of contextual information is even more significant when available direct object features are not reliable enough due to viewing conditions.

## 1.1 Outline

In the following sections of this chapter, the general formulation of the problem is stated, follows by a brief analysis of related works. Then motivation and our contributions are presented.

Chapter2, describes our challenges, method and details about the implementation. Chapter 3 talks about experiments, results and their evaluation. Finally, our model and descriptor applications in place classification are discussed, and an analytical study is done for the results expected for this application. Then the next potential steps that can be worked on in pursuit of this work is suggested in Chapter 4.

## 1.2 Problem Formulation

In this work, our goal is to propose and develop a method to detect the "3D context of objects", rather than the object itself. The detected context can then be used as a very useful prior for object detection tasks and place categorization. The task of object detection can be formulated, in general, as a probability of observing some features in a scene V, given the presence of an object O:

$$p(O|V) \tag{1.1}$$

Features can be seen as two different sets, one directly related to the object (intrinsic features, such as size, shape, color,etc) and one related to its context. We call the former $V_I$ and the latter as $V_C$.

$$V = V_I, V_C \tag{1.2}$$

$$P(O|V) = P(O|V_I, V_C) \tag{1.3}$$

Using Bayes rules, we will have:

$$P(O|V) = \frac{P(V_I|O, V_C)}{P(V_I|V_C)} P(O|V_C) \tag{1.4}$$

**Our focus is on the second part** ($P(O|V_C)$).
Suppose our system can determine the presence of the context for the object(s) of interest in a scene. In that case, the likelihood of finding the object within the detected context increases significantly. Once this prior is available, it also

increases the confidence about detected intrinsic features. We defined a descriptor for the object's context and a method to model it to reach this goal. The technique and descriptor then are evaluated to show how much improvement it can make in certainty and cost of an object detection task.

## 1.3 Related Work

In this Section, we review other works that focused on using contextual information for object detection/recognition or place classification.

The usefulness of contextual information for object detection and recognition is discussed and considered in [21], [23], [13] and [2].

Torralba et al. in [21], used a low dimensional global image representation for recognition of places and showed how contextual information can provide priors for object recognition. "We show how we can use the context to predict properties of objects without even looking at the local visual evidence". They consider the place category as a piece of contextual information and suggested that knowing the category of a place helps to find objects relevant to that place category. Once the place category (context) is determined, the system selects which object detectors should run. Their system is trained and tested on indoor and outdoor scenes using 2D images as inputs. Their system appears to perform well in known places but not as good for new places(unseen).

*Torralba* in [23], is considering a general representation of a whole scene in which an object is present, as its context (similar to the above). Context with their definition is proportional to the object's size with respect to each scene (image). So if the object is taking a small percentage of the image, the context will be mainly its background. In contrast, in a focused image where most pixels belong to the object, contextual information becomes intrinsic image features. Then they compute a context-centered object likelihood by modeling the relationship between contextual features and object properties.As a result, regions of an image with a higher probability of containing the object of interest get marked.

Perko and Leonardis in [13] also proposed a method to focus attention on regions in an image (2D) where an object of interest is more likely to be found. They showed the performance of their approach for pedestrian detection in urban scenes. However, the method is said to apply to any object category and independent from task-specific models. They concluded that context-awareness provides benefits over pure local appearance-based processing and decreases search complexity while improving the robustness of object detection.

Compared to *Torallba et al.* they used different features and descriptors and evaluated combining their contextual priming method with state-of-the-art object detectors. They used an additional contextual cue, the viewpoint prior, in addition to geometrical and texture features to estimate the position of the horizon.Geometrical features capture probability of each context point to belong to each of three classes of *Ground*, *Sky* or *vertical* (Buildings, trees, ...).

For extracting contextual features, they consider a center for their object of interest then sample the features at 60 points around that object center (5 radii and 12 orientations). The obtained information is concentrated into one feature vector for that object center. They use linear SVM to train their model then convert the results into probabilities of a pedestrian being present at a given pixel position.

Their data acquisition setup, especially for estimating the viewpoint prior, needs to be adjusted based explicitly on the object category and application.

*Aydemir et al.* in [2] argued that "the placement of everyday objects is highly correlated to the 3D structure of an environment- as opposed to being correlated to the appearance of the environment". So they believe in the existence of a correlation between an object and its natural context due to the physical support and reachability that the context can structurally offer.

They also train models for object context to focus the attention of object detectors to reduced search regions that match the learned models. They benefit from the recently available RGB-D sensors to obtain 3D data and show how the 3D context of objects can be a "Strong indicator of object placement in everyday scenes". They collected a large dataset from five different locations (but all university indoor rooms and corridors) using Microsoft Kinect. They are modeling object context using geometrical features of a neighborhood of annotated objects and trained a SVM, using weighted feature responses collected from different neighborhood parts, for capturing the geometry of they calculated histograms of surface normals on the 3D data corresponding to each selected region from an image.

*Ayedemit et al.*'s definition of context is close to the definition we used in our work. However, they used images, plus depth information to model their context and do their evaluations, while we used full 3D reconstructed pointclouds of the scenes. Using the pointcloud allowed us to collect features from the observed environment without the need to focus on objects and their neighborhood (Training and test sets are full Pointlclouds and not pruned and adjusted scenes. Similar to us, they noticed how much height can be an informative factor for most object classes.

In our approach, we defined a point of view for our feature collection, which can address the issue mentioned in [2], for larger objects. Our viewpoint provides a framework for the features to be collected from different points of view with respect to the object and makes it indifferent to the size or distance from the observer (camera). In [23], [21] and [13], objects that takes a small fraction of an image pixels can get neglected, while in [2], the framework performs less effective for larger objects, due to the variance of observable background as a result of difference in the viewpoint.

So our approach with using full pointclouds the unique definition of viewpoint and the way features are collected tries to address limitations mentioned in the works reviewed above.


**Figure 1.1:** A water tap as the object of interest and what we call its context. The context is surfaces within the green bounding box.

## 1.4 Contributions

In this work, we use 3D pointclouds of different place categories and annotated objects in them as input to train 3D models of object context. Then these models are applied on test pointclouds to determine which parts of the pointcloud (i.e., the place) are the most probable context for an object of interest. The object context is defined as surfaces in the vicinity of the object of interest. This includes surfaces the object is located on or next to and other objects that can usually be found beside the object of interest. For instance, if the object of interest is a *kitchen water tap*, its context can be the *kitchen sink* and the wall behind it. (Figure 1.1)

We propose a unique method and features to build 3D models of object context, which not only can be used as a prior to object detectors, but also as instrumental knowledge in building a 3D representation of place categories. By benefiting from our context models, the search area for object detection tasks can be reduced to parts of the scene with high likelihood of being the context for the object of interest. It also significantly increases the number of true positives in prediction results. Combining the information encoded in this model with other features and structural information of different place categories can result in a representation for places.

To build Object Context models, we extracted features from several different regions of pointclouds with annotated objects as the training dataset. Then applied supervised learning methods to train the models. Pointclouds that we used are full 3D representations of a place such as an office, a kitchen, etc. The

learning is done on features which are representing local geometry and spatial location of contexts of objects.

The most important points about this work can be briefly mentioned as follows:

- Emphasizing the role of context in object detection and its related applications.
- The unique feature descriptor and methodology suggested to extract features and model the 3D context of objects.
- A simple method for annotating objects in pointclouds to be used as training data is proposed and implemented.
- The input data used in this work are full pointclouds of places rather than single frames with depth information or 2D images. The benefit of using 3D pointclouds over other data types is emphasized and shown.
- Unique method to evaluate the context models
- Suggesting the application of the Context model in place classification and proposing a method to do it.

## 1.5 Motivation and Applications

A motive for this work is to achieve a complete and robust 3D descriptor for places based on object contexts rather than objects. When a human agent enters a room, intuitively, he can decide about the category of the place based on a simple and shallow perception, with reasonable confidence. Even if the room is not furnished yet and there are not many objects in it, to help the agent in determining its category.

Perhaps more than the presence of some objects in a place, it is the possibility of their presence that helps us determine the place's category. Geometrical and spatial information about different regions in a place can help us imagine what object categories can be placed in it. In other words, what sort of object context can be found in that place.

For sure other information adds up to this knowledge to make a human able to distinguish between places, such as general shape and size of the place also its location regarding other places. A model for the **context** of an object category can help us with estimating the likelihood of that object category's presence in each scene.

Among the applications that are considerable for the model, the most obvious ones are as follows:

### Object Detection

Object detection is an expensive and complex task. Searching for the context of an object is, most of the time, significantly easier than the object itself, especially when the searched object is on a relatively small scale comparing to the search area. Once the context is detected, the search for the actual object is done in a reduced area, within the parts predicted as the context.

In addition, false positives are avoided or at least reduced. For example, when a table is detected as the context for a telephone, any similar shape on the floor will not be considered by the object detector, which otherwise would have detected it as a candidate.

### Place Classification

A critical piece of information that can help a robotic system to be able to distinguish between different place categories is knowing about the objects which are expected to be found in a specific category of a place.

If the robot is looking for a kitchen, it helps if it knows about some objects that are most probable to be found in it and perhaps not in other places. For instance, a kitchen sink is such a discriminative object. If the robot can find such an object in a place, it can obtain reasonable confidence about that place to be a kitchen.

*Viswanathan et al.* in [25], also pointed to the fact that object detection is an excellent prior for place categorization. *T. Southey and J. J. Little* in [20] argued that *rooms* are defined by their geometric properties, while the definition of *places* is based on objects they contain, and activities take place in them.

On the other hand, the place classification system would be more robust if it can recognize the place, using object information, even **when the actual object is not present in the place, but the presence is expected**. Here is where the contextual model of the object finds its importance. When the context is present, there is no need to detect the actual object to classify the place.

### Object Placement

An intuitive application for the model of object context is finding a suitable location or surface to put an object. When a robot knows the context model of an object, it can place it correctly by finding matches of that model in its surrounding. For instance, a robot that performs typical household chores, like cleaning or serving guests, can benefit from it.

# Chapter 2
# 3D Model of Object Context

In this chapter, we define our proposed method and the features used to model the Object Contexts. We also discuss the implementation, obtaining train and test datasets, data annotation, and expected results in detail. Nevertheless, first, let us take a look at some of the challenges we had.

## 2.1 Challenges

There are challenges regarding modeling the object context that follows:

- **Train set quality**: There could be situations and scenes which are completely different from what has been considered in the train set to build the context model. The closer to an all-encompassing dataset we can get, the more accurate model we can have. This would be a challenge in most projects that involve learning.
- **Annotation of objects in Training data**: An easy and reliable tool was needed to annotate 3D objects in pointclouds.
- **Pointcloud's quality**: The quality of pointclouds is a critical issue. The amount of smoothness in the pointcloud and missing points significantly affects the extracted features, which are used to describe surfaces, and as a result, on the context model. Upon capturing pointclouds, due to the sensor's viewpoints or obstacles in its line of sight, the resulting pointcloud might miss some essential points. Therefore, the extracted geometry from such surfaces would not be accurate (and in some cases, it would be wrong).
- **Unbalanced sample's distribution**: In feature vectors extracted from training pointclouds, the number of positive samples (feature vectors extracted from a region with annotated object presence) are significantly different against the negative ones.

## 2.2 2D vs 3D context, benefits of 3D

The popularity of devices such as Microsoft Kinect has made 3D data widely accessible. Therefore, many researchers and academics have shown tremendous desire to use 3D information in their researches, which gives a lot more information about our surroundings compared to 2D data. Using 3D data has made it simple to capture geometrical information in addition to visual properties. From the output of this type of sensors, we can quickly and directly have access to depth information and 3D coordinates of any points that are perceived.

Geometrical features, which are obtained from 3D information, enhance the performance of systems significantly in tasks like feature based object detection compared to relying only on visual information. Notably, in the problem we are dealing with, building a model for object context, depending on features that only encode changes in intensity and color is not enough. The geometry of surfaces needed to be analyzed, and 3D information facilitates and improves this analysis.

## 2.3 Methodology

In this section, the procedure of building the context model and how it is applied is described in detail. The input in this procedure is the pointcloud of a scene and the output is a list of probability values assigned to all regions in the scene which corresponds to their likelihood of being the context of a particular object class.

The input cloud is first preprocessed and then annotated with instances of the object class whose context model is being built. Feature extraction runs on this annotated pointcloud and outputs the training samples. Train samples are combined from several different pointclouds for the same object class to make the train set. Then the train set is given to the SVM classifier to make the context model. A block diagram in figure 2.1 shows the system overview.

In the following subsection, we introduce the features used in this work and description of the modules of the system comes next.

We used PCL [18] for pointcloud processing and Microsoft Kinect with OpenNi driver to capture these pointclouds.

### 2.3.1 Defining some used terms

Here are some terms that are used through the rest of the text:

***Point of Interest***: A point from a pointcloud for or around which analysis is being done.

***Blob***: A subset of the pointcloud within a radius r around a point of interest.

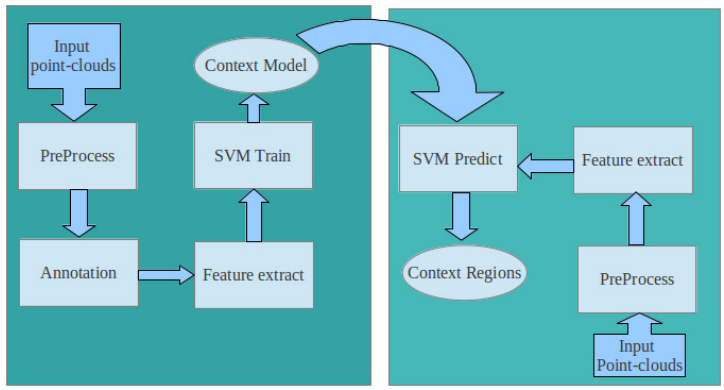



**Figure 2.1:** The overview of the system. Box on the left side shows the Train part of the system while the one on the right shows the prediction part.

***Query blob***: The blub from which we are extracting the features.

***QPoint***: The Query Point that is the center of the query blob.

***Keypoint***: All points in downsampled version of an input pointcloud is called *Keypoint*. QPoints are selected from keypoints in *FeatureExtract* module.

***OPoint***: The Object point; is a point in the vicinity of the Qpoint, which is a candidate to be an object point.

***B-Radius***: Radius of the sphere around *QPoint*. Points that are included in this sphere make the *Query blob*.

***S-Radius***: Search Radius; is the radius around the *QPoint*, within which candidate *OPoints* are considered.

***OPoint-Radius***: To extract samples from an annotated pointcloud to make a train dataset, for each considered *OPoint*, the algorithm look for labeled object point within a small vicinity around it. The radius for this search vicinity is called *OPoint-Radius*. If the considered *OPoint* is located on an annotated object, there should be a labeled object point very close to it (within this vicinity).

***TFP pointcloud***: Template Full Point pointcloud; is an artificially generated set of point coordinates in the pointcloud format (3D Matrix of points) to be used as candidate viewpoints. It makes a dense cubical pointcloud where its

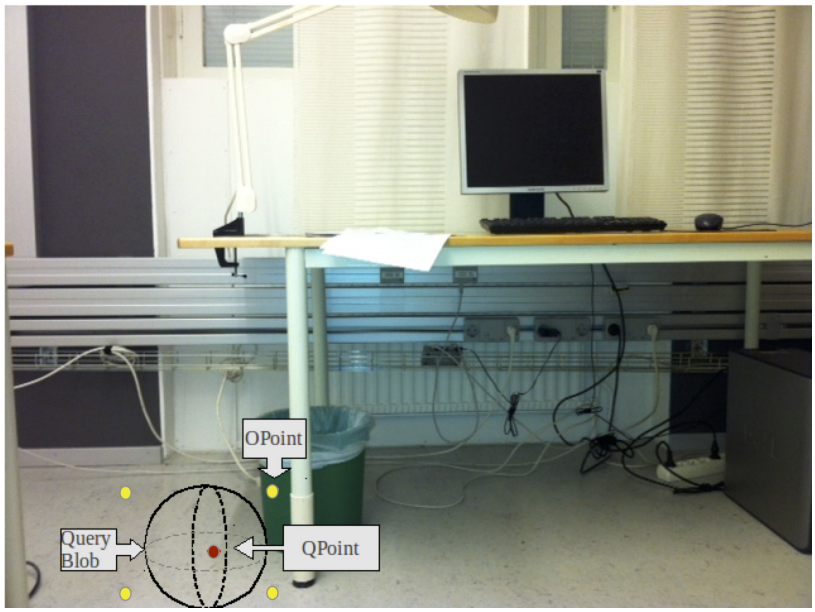



**Figure 2.2:** Structure used in Feature extraction; The red point is the QPoint; The sphere surrounding it, is the Query Blob; The yellow points are some of the OPoints. Opoints are availble in all directions around the QPoint (Best viewed in color).

density is based on the selected voxel size. The dimensions of TFP pointcloud get set in a way that it can cover the entire pointcloud from which we want to extract features (for train or test).

***Voxel-Radius***: This parameter determines the size of the voxel in TFP pointcloud. Its the distance between adjacent points in each dimension.

***Search blob***: A subset of *TFP pointcloud* within *S-Radius* of a *QPoint*.

See figure 2.2 and 2.10.

### 2.3.2 Object context descriptor

The feature descriptor we proposed in this work consists of the following elements:

**local surface orientation:**

This element captures a rough estimation of query surface orientation. It is computed by estimating the angle between the average normal of the surface and the gravity vector. We consider a unit vector along the z-axis in the world coordinate system as the gravity vector. It is noticeable that the position and orientation of the camera with respect to this coordinate system is known and used as the ground truth to transform pointcloud from Camera coordinate system.

**Local surface height:**

This element is the average height of the query surface from which the features are extracted with respect to the floor. The floor is assumed to have the lowest height or vertical position in the pointcloud.

**Viewpoint Feature Histogram:**

This element of the feature vector is a histogram that encodes the geometry of the blob with respect to a specific viewpoint. We found this descriptor very useful for encoding geometry of surfaces in pointclouds for specific viewpoints. The innovative way we took advantage of this powerful descriptor helped us to encode the geometry of surfaces from several different points of view to make our context model independent from the viewpoint. As it was mentioned in 1.3, being dependant to the viewpoint was a challenge in [2]. This descriptor is called VFH ( [19]), which is explained more in the following subsection. [1]

### The Viewpoint Feature Histogram (VFH)

VFH consists of two components ( [17] and [19]):

- A histogram for viewpoints(128 bins)
- A histogram that encodes local geometry(180 bins)

The viewpoint component is computed as a histogram of the angles that the direction of the viewpoint makes with each surface normals (Figure 2.3). So for a query blob $S_q$ if we call the viewpoint as $V_p$, for all points of the blub $p_i$ and the surface normal at that point $n_i$ we have $\alpha$ as the angle between V and $n_i$ where $V = V_p - p_i$.

It is crucial to notice that these angles are between the central direction of the viewpoint and the normal, not each normal's view angle. This is why these features are scale-invariant.

[1] The images and definitions are taken from Point Cloud's official website.

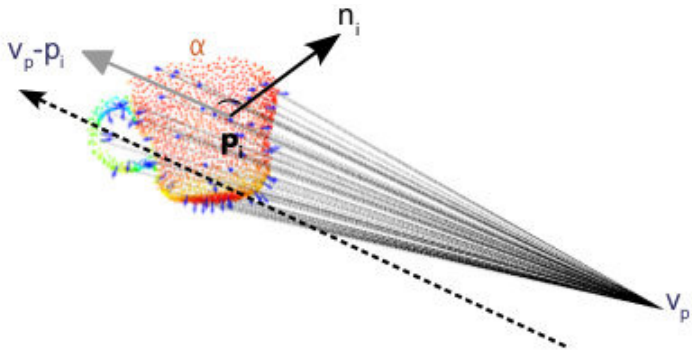



**Figure 2.3:** Viewpoint part of the feature vector.[17]

The second component collects the relative pan, tilt and yaw angles between the direction of the surface normal at the centroid of a query blob and the surface normals at other points in the blob. So for each query blob, the mean curvature around its central point is encoded using a multidimensional histogram of values. If C is the center of the query blub and $p_i$ is one of *k-nearest neighbors* of the centroid and $n_i$ is the surface normal at $p_i$, we have three angles and a distance to be captured in this histogram. $\alpha, \theta, \phi$ and $d$, where the angles are pan, tilt and yaw between each pair of normals and $d$ is the distance between $p_i$ and C.

$$u = n_C \tag{2.1}$$

$$V = (p_i - C) \times u \tag{2.2}$$

$$W = u \times V \tag{2.3}$$

Figure 2.4 Shows the encoded elements and their relations.

VFH is robust to different sampling densities or noise levels in the neighborhood of the query point. The descriptor is computed by estimating a histogram with concatenated 45 bins for the four different elements mentioned above.

The histogram in this component has 180 bins (4*45 bin). By adding the viewpoint component (128 bin), it produces a 308 bin histogram.[2] The viewpoint component is the element that makes a distinction between features captured from a single query blob viewed from different directions. It is discussed

[2]We used the available implementation in the Pointcloud Library (PCL)

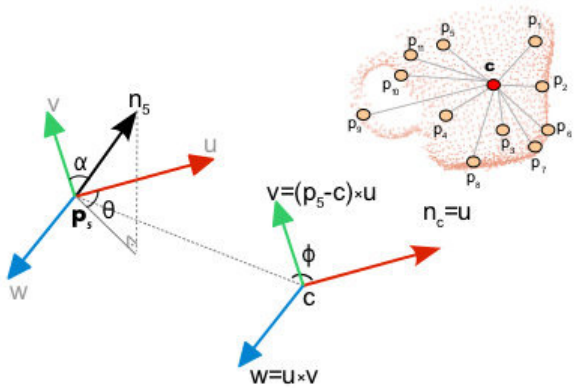



**Figure 2.4:** Three angles and a distance encoded in the descriptor.[17]

**Table 2.1:** Structure of object context descriptor.

| Field number | Content |
|---|---|
| 1 | Orientation |
| 2 | Height |
| 3-130 | Histogram of viewpoints |
| 131-310 | Histogram of surface angles |

in more detail in Section 2.4.3. Despite its dependency on a viewpoint, it still preserves the property of being scale-invariant.

The resulting object context descriptor is a vector with 310 elements that encodes a general orientation, height and geometrical properties of the query blob. The structure of this descriptor and its elements are shown in Table 2.1.

## 2.4 Implementation

The main implementation is done in C++. Some available libraries are also used in the code, which are listed below.

- *PCL* [18]
- *Eigen* [7]
- *LIBSVM* [5] & [10]

Besides the main code, some scripts are written for results evaluation and creating the datasets for experiments in MATLAB. Also, scripts creating the experiments and the experimental environment are in python.

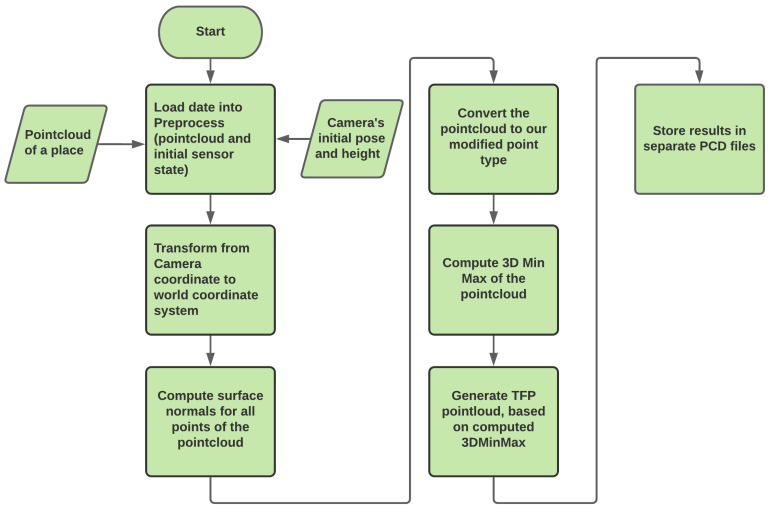



**Figure 2.5:** An overview of the PreProcess

As illustrated in Figure 2.1, there are five modules in the system:

- PreProcess
- Annotation
- FeatureExtract
- Learning
- Result visualization

Here we describe each module in detail.

### 2.4.1 PreProcess

In this module, the input pointclouds get prepared for annotation and feature extraction (see Figure 2.5). The processes for capturing these pointclouds are discussed in Section 3.1.

To make all of our calculations consistent and more straightforward, as the first step, we transform the captured pointclouds from the camera coordinate

system to the world's coordinate system. Based on the position and orientation of the sensor in time $t_0$, the transformation is done, using the available functions in *PCL*. [3]

The essential step in PreProcess, is the estimation of surface normals which is required for feature extraction. In the earliest vestions, we started with estimating surface normal from each query blob in the feature extract module. However, later we moved this step to the PreProcess module and estimated the normals for the entire pointcloud once, and saved them in a PCD file to be used whenever needed. During feature extract, estimated normals can be loaded from the file and used to compute features for each query blob. Normals are estimated for each point, using *NormalEstimation* class in *PCL* which uses PCA method on the covariance matrix of the points.

In order to make it easier to classify points, we defined a new point type by adding a field to the original point type (*XYZRGB*). This filed is used for labeling each point as *object point*, *Context point* or *others*. The new point type is in *XYZRGBL* format, where L is the label field. In case we are modeling contexts for multiple classes of objects, the label can encode all different classes into the points in the pointcloud at the same time. In this step, converting the transformed pointcloud (in the world coordinate system) into XYZRGBL point type is also carried out.

There is also one more product for this module, which is discussed more in 2.4.3. It is a generated cube-shaped pointcloud with the input cloud's length, width, and height, filled with points. To get the required dimension for this grid pointcloud, we use the 3DMINMAX function from *PCL*. A fixed radial distance between them determines the density of these points. It is used as a source for picking viewpoints that are needed in feature extraction.

Therefore, the results of this module are as follows:

- Point-cloud's normals
- The transformed version of the input cloud
- Converted version
- TFP-cloud

### 2.4.2 Annotation

In this module, the transformed and converted version of the pointcloud is loaded and visualized.(see Figure 2.6) The visualization is in 3D, and the user can easily manipulate the pointcloud (rotate, zoom in and out, move, etc.) to

[3]Sensor is in a fixed orientation and vertical position in time the first frame is captured for all pointclouds used in experiments of this work.

**Algorithm 1** A brief algorithmic description of PreProcess.

**Require:** Input Point-cloud(XYZRGB).
**Require:** Sensor Location and orientation.

1: Transform the input cloud, based on the Sensor position and orientation from the camera coordinate system to the world coordinate system, and store the result in pointcloud_Transformed.
2: Estimate Normals and store.
3: Convert the Pointcloud_Transformed to Point type XYZRGBL and store in Pointcloud_Converted.
4: Estimate 3DMINMAX of the Pointcloud_Converted.
5: Using the result from the previous step, generate TFP pointcloud.

**Ensure:** Pointcloud_Transformed.
**Ensure:** Pointcloud_Normals.
**Ensure:** Pointcloud_Converted(XYZRGBL).
**Ensure:** Pointcloud_Generated(TFP).

find the object and select them. The objects in the scene can be selected by clicking roughly on their centroid.

Once the centroid is selected, the points that are most likely belonging to the same object (within a defined vicinity, closer to each other, compared to points from other adjacent surfaces) get segmented and labeled as the points belonging to the object of interest. Labeling the points means assigning the label field (of each of the points) to a value representing a particular object class. Objects from different classes get labeled with different values (2).

Figure 2.7, shows a scene in an office at *Royal Institute of Technology(KTH)*. The object of interest here is the trash bin, which is marked by a green bounding box. This figure is a 2D image of the scene whose pointcloud is available in our dataset, and the annotation is done on the pointcloud. In Figure 2.8 the annotated object could be seen in red within the scene's pointcloud.

It can be seen that some parts of the floor are also colored in red. It means that part of the floor is also annotated as the object. The reason is that the object's radius given to the Annotation tool was not accurate enough, or the center of the object was not picked accurately. However, it does not harm the result, and this much accuracy is more than enough. In similar works, annotation is done using a bounding box (2D or 3D) that considers a box with a specific dimension around the object. The content of that box approximates object points while other points, which do not belong to the object itself, are also included. Here, we strictly separate the points of the object by benefiting from the 3D coordinates of the points in the pointcloud, so the annotation is more accurate in comparison.

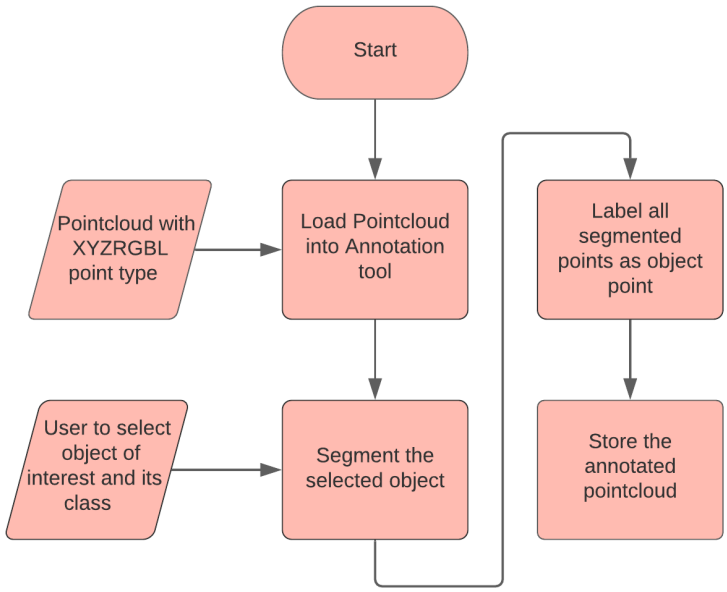



**Figure 2.6:** An overview of the Annotation

As described in the algorithm 2, object segmentation is done automatically in a straightforward way. Based on the picked object center, selecting points from the 3D neighborhood of it separates object points from the rest of the pointclouds.

We have published this package with its source in GitHub to be used by other researchers and improved by other developers.[3]

### 2.4.3 FeatureExtract

A feature vector, as described in 2.3.2, gets extracted for several pairs of (QPoint,OPoint) from the entire input pointcloud. Figure 2.2 shows an example for *QPoint*, *Query blob* and *OPoint* for the scene, we saw in Figure 2.7. The red point in the figure shows the *Query Point*, which is a point from the downsampled version of the same pointcloud. The sphere shown around the *QPoint* is the *Query blob*. The yellow point is the *OPoint* or candidate object point, which in this example is sitting on the annotated object (Trash bin). Therefore it is an object point. In this figure, only some of the *OPoints*, with respect to the shown *QPoint* are depicted to clarify the structure.

**Algorithm 2** A brief algorithmic description of Annotation.

**Require:** Pointcloud_Converted(XYZRGBL).
**Require:** Object radius(rough estimate).

1: Visualize input cloud.
2: **for all** objects to be annotated **do**
3: Run Point picking procedure to get the centroid of an object.
4: Extract closes neighbor points indexes with respect to the input object radius from the vicinity of the picked centroid.
5: Label points whose indexes are extracted in the previous step.
6: **end for**
7: Store the pointcloud with the labels in the output cloud.

**Ensure:** Pointcloud_Annotated(XYZRGBL).

These *OPoints*, coming from the overlapped TFP pointcloud, are uniformly distributed in 3D. Feature extract picks the ones that are within the *S-Radius*, in a loop to obtain feature response. In this section, the feature extract process is explained in detail.(See Figure 2.9)

Figure 2.10 shows points and their related parameters. The values mentioned in the figure are for an example object class.

*FeatureExtract* module, browses through all surfaces and points in an input pointcloud and extracts feature responses from all regions. As mentioned before, our descriptor encodes height, pose and the 3D geometry of the surfaces in a *Query blob* with respect to several viewpoints (*OPoint*). Therefore, we could consider every single point of the pointcloud as *QPoint* and extract the feature response from the *Query blob* around it.

However, this way, since the feature response is computed for the blob in the neighborhood of the *QPoint*, the process would be very long and studies each region several times, redundantly. [4]

To make the process more efficient, first, the pointcloud gets downsampled, and the *QPoints* are selected from the downsampled pointcloud, while the *Query Blob* comes from the original one. This way, we make sure all regions of the pointcloud get encoded with the least redundancy.

Downsampling is done based on a voxel size which this module receives as input. This voxel size, together with the *B-Radius*, determines the efficiency and accuracy of the pointcloud sampling. As described in 2.4.6, these parameters are selected based on experiment and cross-validations done for the selected object classes.

So the input pointcloud, which is preprocessed (Transformed and converted to XYZRGBL) and its downsampled version, is used to obtain *Query blobs* and

[4]For every point in each region, features are extracted from blobs which largely overlap with each other; therefore, their feature response are very similar

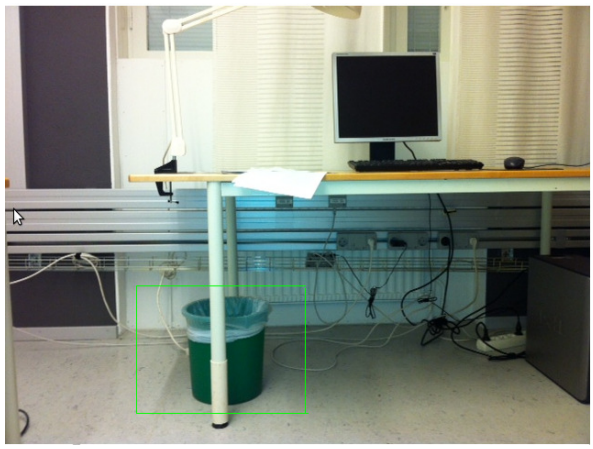


**Figure 2.7:** An example scene and a trash bin marked by a bounding box as the object which is being annotated (Best viewed in color).

*QPoints*. The surface normals for every point in the pointcloud are available from the *PreProcess*, which gets loaded from its PCD file. In order to build the data set for train or test, Features are extracted for several keypoints from the entire pointcloud. Keypoint selection is described later in this section.(See figure 2.2)

As mentioned, a feature vector gets extracted for each pair of ($\mathtt{Qpoint}_i$, $\mathtt{OPoint}_j$). A two-layer nested loop selects $\mathtt{QPoint}_i$ from all *keypoints*, where

$i = 0$ to $n$
$n = \mathtt{size}(\mathtt{PointcloudDownsampled})$

and for each $\mathtt{QPoint}_i$, the inner loop selects $\mathtt{OPoint}_j$ from points in *TFP Pointcloud* that are within *S-Radius* of $\mathtt{QPoint}_i$.

$j = 0$ to $m$
$m = \mathtt{size}(\mathtt{SearchBlob})$

The *Query blob* for each QPoint is grabbed from the original pointcloud, and their corresponding surface normals are used to obtain feature response with respect to the selected viewpoint. We consider $\mathtt{OPoint}_j$ as the viewpoint. As the inner loop picks different OPoints, the feature response for *Query blob i* gets obtained from several different viewpoints surrounding it. This way, the features extracted for a single query point and its query blob would be different due to the difference in viewpoint.

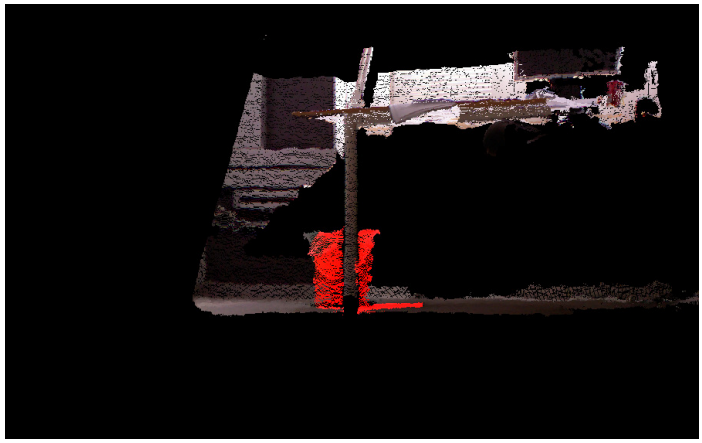


**Figure 2.8:** An example of annotation result on the pointcloud, red points assumed to belong to the object(Best viewed in color).

If the pointcloud is annotated and samples are being extracted for the Train dataset, they should be labeled. The algorithm checks if the selected OPoint is located on an annotated object or not. If it is, the sample gets labeled as positive and, if not, as negative. In 2.4.4, we see how we deal with negative and positive samples.

The viewpoint also helps to encode the most probable locations of the object with respect to the predicted regions as positive context in test pointclouds. Another critical point here is that we want the model to be able to locate candidates for context in a test pointcloud where the object may not be present at the moment, but its context is. Therefore, we need the OPoints to be independent of actual points in the pointcloud. *TFP-cloud* helps us to provide independent points to be considered as viewpoints.

### 2.4.4 Learning

After Feature extraction, the train set is made in a way that includes samples from several different pointclouds for the same object class. A bigger portion of the train samples gets selected for training and the rest for validation. In the first experiments, we used a single Gaussian kernel for *SVM* on the whole feature vector whose fields are of two different types. As mentioned before, the first two fields are float numbers, and the rest is a histogram. Later we separated these two parts and applied independent kernels on each, which, as expected, improved the results significantly.(See 3.3)

Another important issue here is the distribution of samples in each of the positive and negative classes. From the data we extracted, on average, about

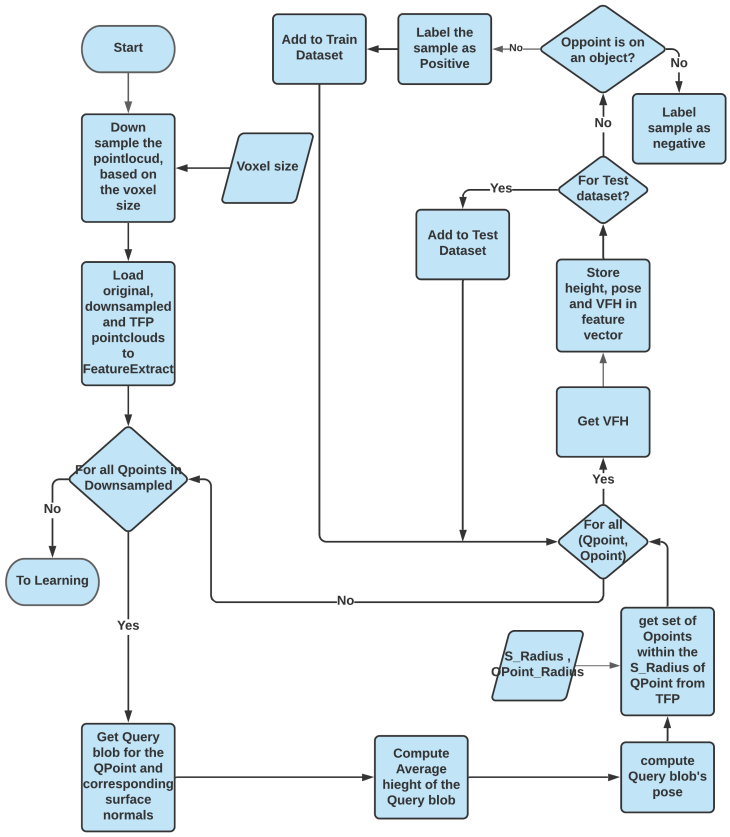



**Figure 2.9:** An overview of the FeatureExtract process

one percent of the samples were positive, and the rest were negative. This is because the annotated object is a small part of the pointcloud, and positive samples are only extracted from its surroundings. This significant difference in the number of samples in positive and negative classes makes the classifier biased toward the negative class.

In addition, this issue has some other effects that make the classier confused. The features extracted from blobs in the object's neighborhood may be so similar to some features extracted from a blob far from the object. At the same time, the first one would be a positive sample in train data, and the second one would be a negative sample. For instance, the scene depicted in figure 2.2, for the marked *Query blob*, when the viewpoint is the one which is located on the trash bin, extracted feature vector is labeled as a positive sample. While

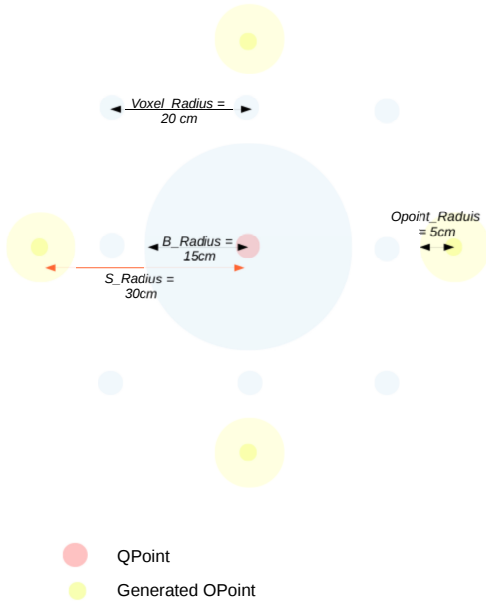



**Figure 2.10:** QPoint and generated OPoints and their spacial relations.

for the same blob, when the viewpoint is any other depicted *OPoints*(shown in yellow), the extracted sample would be a negative one, since they are not located on any annotated object.

In order to solve the mentioned issue and both of its effects, or at least improve the result toward our goal, we needed to do an unbalanced weighting for our samples. We decided to assign different weights for misclassification costs in the SVM binary classifier. It should be in a way that compensates for the lower amount of positive samples and emphasizes the importance of positive samples compared to negative ones. It can be inferred that there should be a big weight for the positive class and a relatively small value for the negative class. In Section 3.1 a parameter selection procedure used to find suitable values for them is discussed. The way datasets and experimental environment are created, discussed in that Section, as well.

**Algorithm 3** A brief algorithmic description of Feature Extract.

**Require:** Pointcloud_annotated or Pointcloud_converted(Depending train or test).
**Require:** Pointcloud_Normals.
**Require:** Pointcloud_Generated (TFP).
**Require:** S_Radius
**Require:** OPoint_Radius

```
Key-point selection.
for all Key-points do
    Extract Query blob.
    Extract indexes of generated points within S_Radius from the key-point.

    for all extracted generated points do
        Assign generated point to OPoint.
        Extract features for the OPoint and the Query blob
        if Features are for train then
            if There is an object point within OPoint_Radius of the OPoint then
                Label the sample as positive
            else
                Label the sample as negative
            end if
        end if
    end for
    Store the sample
end for
```

**Ensure:** Pointcloud_Extracted samples.

### 2.4.5 Visualization

Once the context model for each object class is trained, it can be applied to any new pointclouds to obtain regions with a high likelihood of being the context for that object class. The pointcloud goes through the feature extract process, as mentioned in the previous section, and the obtained samples for each *keypoint* (*QPoint*) get a value between 0 and 1, as their likelihood of being a context point.

These likelihood values then get stored in each corresponding *keypoint*. Then the pointcloud gets visualized with these keypoints and their corresponding query blobs marked with a recognizable color. Figure 2.12 shows the process overview. In the next chapter, the results are evaluated, and some example visualizations are presented.

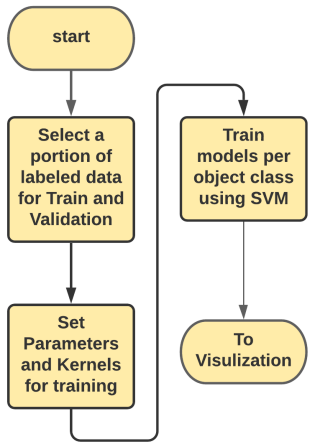



**Figure 2.11:** An overview of the Learning process

### 2.4.6 Parameters

**Feature extract parameters:** As mentioned before and depicted in figure 2.10 some parameters play a significant role in achieving good results. These parameters are:

- B-Radius: Radius of the query blob (from which features are extracted) of points from the pointcloud with initial density (not downsampled).
- Voxel-Radius: Radius used for voxel down-sampling.
- S-Radius: Radius of the sphere to search object point in. It depends on the object class.
- OPoint-Radius: This is the radius of the neighborhood of the generated point to look for the actual Object point.

These parameters directly or indirectly are dependent on the average size of the object class that we are making the context model for. This dependency is not so tight; the object size does not need to be very accurate. As long as these values do not cause the object to be missed, they are acceptable. The object can be missed if the downsampling radius, is too large and the *S-Radius* is too small, at the same time. On the other hand, small values cause the complexity to increase and computation time to get too long. As a result, we reduced the number of dependencies to a single parameter which a rough estimate of the object size.

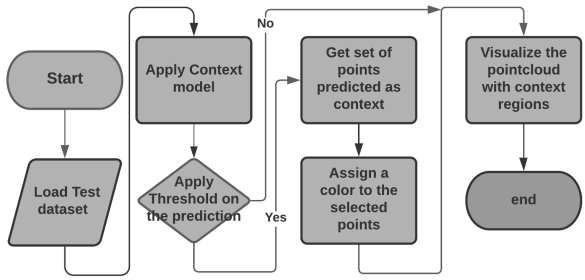



**Figure 2.12:** An overview of the Visualization

**Learning parameters:**

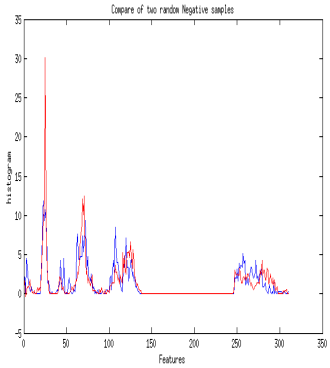
- $w_i$: Weight for class labeled (i)
- c: Sets the cost value for misclassification. $w_i$ acts as a coefficient for c, so the combination of their values will set the cost value for each class.
- g: Sets the value of gamma in Gaussian or chi-square kernels, which is clearly a very important factor for the result we can expect from the classifier.
- kernel type
- weight for kernels in multi kernel setup

## 2.5 Qualitative study of samples

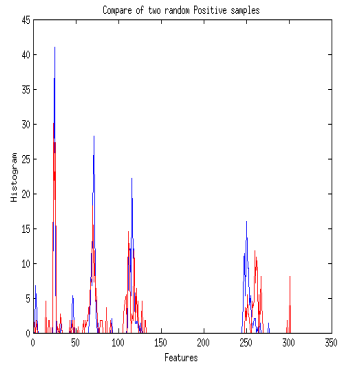
Before describing the experiments and evaluating the results, the extracted samples are briefly reviewed in this section. First, let us check some plots showing how does some random samples look like and how discriminative the feature vector can be. Figures 2.13(a) and 2.13(b) show the plot of two random negative and two random positive samples.

In order to have a more precise idea about differences between samples, we can also look at the distances between random samples in the same class and compare them to the distance of samples from different classes. Figure 2.13(c) shows the distance in two random negative samples. It should be noticed that the first part of the plot reflects the viewpoint component of the feature vector.

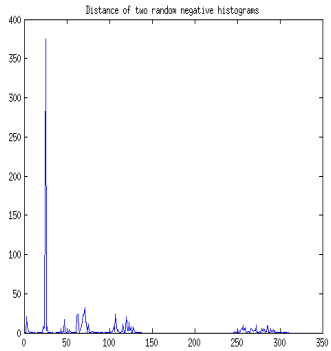
And figure 2.13(d) is a similar plot for positive samples. Features extracted for each class can have significant differences as well, which was predictable. Considering different surfaces captured as the context by these features that



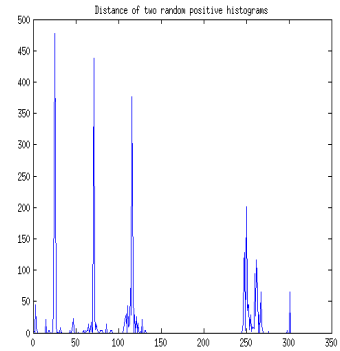

(a) Compare two random negative samples.




(b) Compare two random Positive samples.




(c) Distance between random negative samples.




(d) Distance between random Positive samples.

**Figure 2.13:** Plots of feature vectors of random positive and negative samples and the euclidean distances between them. This comparison shows the challenge in training a classifier for these samples.

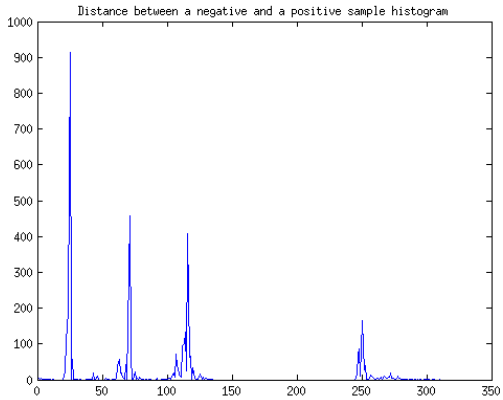



**Figure 2.14:** Distance between random positive and negative samples.

can belong to the same class causes these differences. A positive sample can represent, for instance, the surface of a table or a wall or the meeting region of these two surfaces. This comparison gives us an idea about how challenging it is to train a classifier for such data.

As mentioned before, it is also vital to put more attention on the positive samples rather than negative ones when training the model. Figure 2.14 shows the distance between random positive and negative samples.

The flowchart in figure 2.15 shows the whole workflow of the systems with more details comparing to the block diagram in 2.1. In the next chapter, the setup for the experiment is described, and results are evaluated.

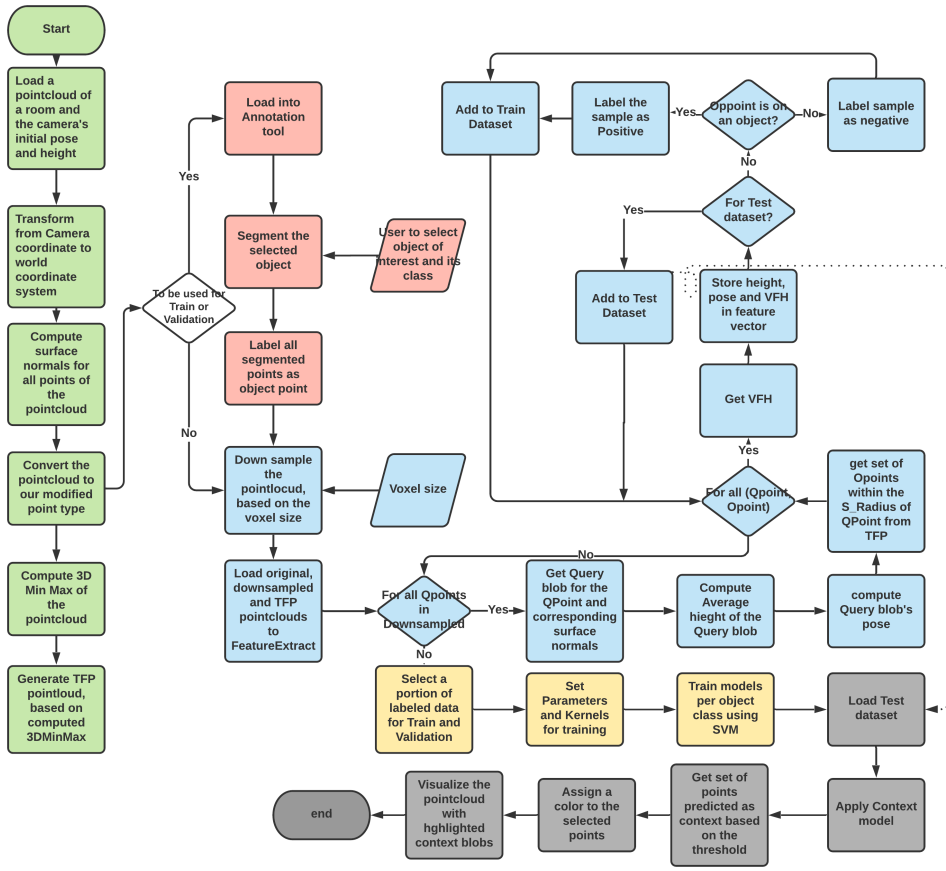



**Figure 2.15:** Detailed Overview of the system.

# Chapter 3
# Evaluation

## 3.1 Experimental Setup

In order to evaluate our method and the model, some experiments are carried out. To prepare a dataset for our experiments, we needed to capture several pointclouds from different scenes and places which include different object classes.

To obtain some usable pointclouds, considering the requirements of our project, we used *Kinect* sensor with *OpenNi* driver. Benefitting from *RGBD-SLAM* [6] which is an open-source package available in ROS, we recorded complete rooms in 3D. The results are saved as pointclouds in pcd files. Pointclouds were captured from different categories of places from KTH campus like offices, Kitchens, and bathrooms, including different categories of objects found in those places.

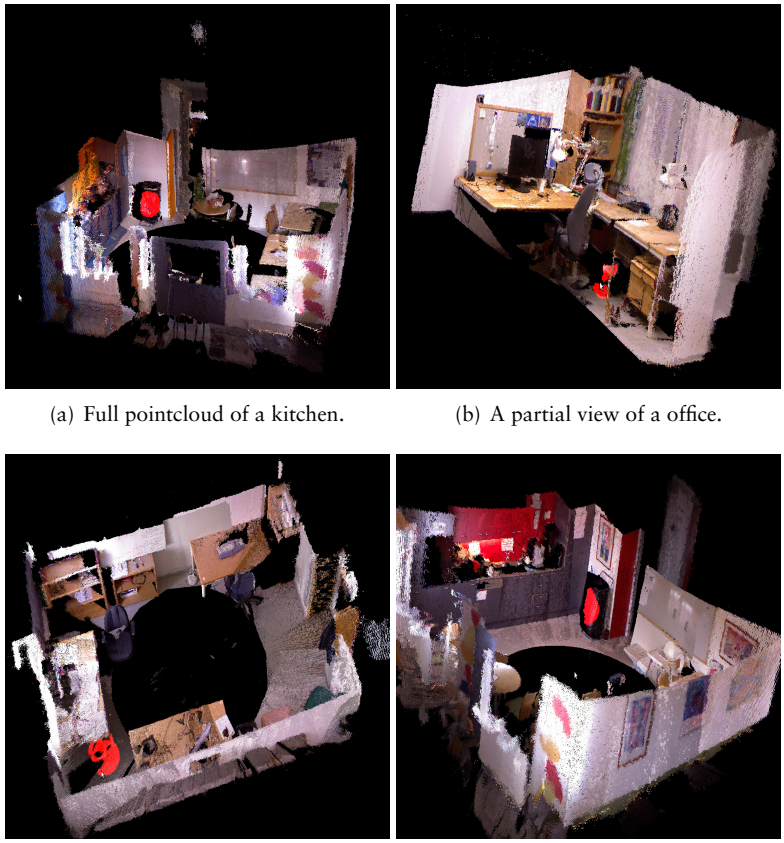
There is also a project in CVAP[1] at KTH called KINECT@HOME [1] which is a web-based application that uses Kinect to build 3D mesh models of objects and places. An excellent property of this system is that people from any part of the world capture their video with Kinect from different places and scenes and post the videos to a server where the 3D reconstruction happens.

Using this tool a good dataset can be prepared to train and test models. Of course, not all of the resulting pointclouds from this database are useful for our purpose due to the content and quality, but still, there are applicable ones.

All resulting pointclouds are saved, and an appropriate name gets assigned to them, based on their content. Then, they are divided into three subsets of train, validation, and test.

Although samples from the same pointcloud could be divided into train and validation sets, we preferred to separate them from the pointcloud level to ensure reliable results. Figure 3.1 shows full pointclouds with different types of trash bins annotated in them. These are the pointclouds used in the experiments.

[1] Computer Vision and Active Perception lab.



(a) Full pointcloud of a kitchen.

(b) A partial view of a office.

(c) A full pointcloud of an office.

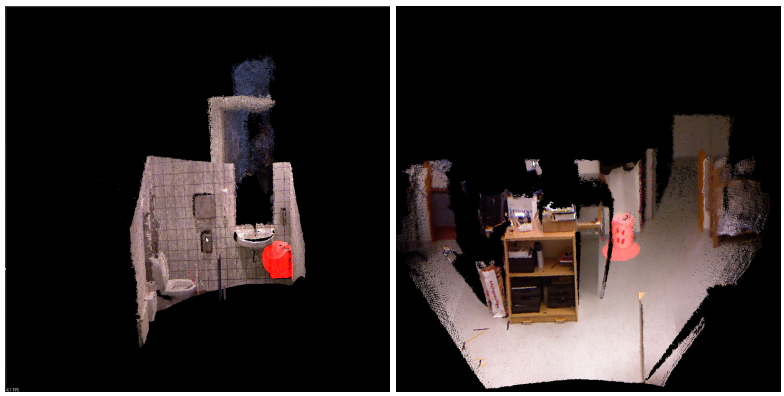
(d) A full pointcloud of another kitchen in the same building.

**Figure 3.1:** Point clouds used for extracting train samples with trash bin annotations. Bins are in different types and locations.(best viewed in color)

**Table 3.1:** Object classes used in experiments with number of samples extracted for each of them for training and ratio of positive samples in them.

| Object | | #Train samples | #Positive Train samples | Ratio (%) |
|---|---|---|---|---|
| 1 | Trash Bin | 142150 | 2493 | 1.7 |
| 2 | Telephone | 119725 | 554 | 0.4 |
| 3 | Mouse | 286825 | 1541 | 0.5 |
| 4 | Wiper | 170025 | 3086 | 1.8 |

In Figure 3.2, we see examples of pointclouds with missing points due to shadowing or the presence of shiny objects like a mirror.



(a) A partial view of a bathroom.

(b) A partial view of an office with lots of missing points.

**Figure 3.2:** Point clouds with shadows and shiny surfaces (Mirror).(best viewed in color)

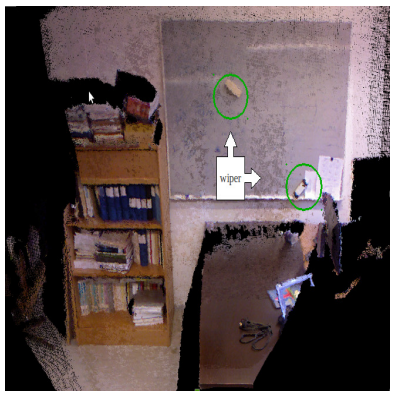
Table 3.1 includes names of four different object classes used in experiments and the number of train samples extracted for each of them. It also shows what fraction of those samples were positive ones that are samples captured from actual context point-blobs.

In experiments with four different object categories mentioned in table 3.1, a few different settings are considered to make the train set more general. So, instances of objects are captured in different expected locations, heights, etc. Also, other objects that can be in the scene together with the object of interest (which can be a part of their context) are considered in these different settings. Therefore, in the resulting train set, each object has several instances in different scenes. (figures 3.3(a) and 3.3(b))

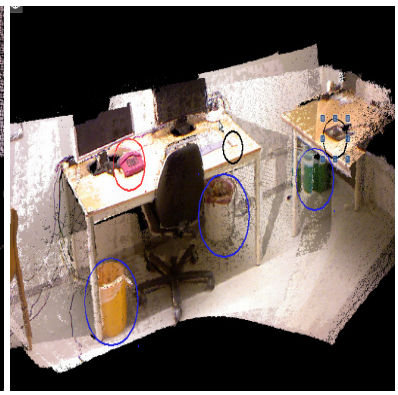
The acquired pointclouds are then taken through an automated workflow (python scripts) that picks a pointcloud from the input dataset, loads it into


(a) Partial cloud including telephone, mouse and trash bin.

(b) The same scene with different location of the objects.




(c) Partial cloud including white board wipers.



(d) A partial cloud including several instances of 3 object categories.

**Figure 3.3:** Partial pointclouds including four object categories: Trash bin(marked with blue circles), mouse(back circles), telephone(red circles) and wiper(green circles) in different location settings used in training.(Best viewed in color)

PreProcess module. Then the preprocessed pointclouds get annotated (labeling points belonging to objects of interest) in the Annotation tool.

Some of the annotated pointclouds are used for training and some for validation. For the ones used in validation, labels from annotation do not get used until after classification. This information is needed for evaluation which would be described in the next Section ( 3.3).

In the next step, the annotated clouds and their normals (estimated in PreProcess) get loaded into the Feature extract module and the resulting feature vectors are saved in a folder for the corresponding object class, under train or validation set, based on the set their source pointcloud belonged to.

Then, a subset is randomly selected from the train samples to make the train set. As mentioned before, the distribution of positive and negative samples is significantly heavier for the negative ones. We used train sets with 10 percent positive and 90 percent negative samples in the first experiments for trash bin.

In later experiments and other objects, train sets are generated with all extracted positive samples and two times more negative samples. In these experiments, positive samples have a 33% share in the train set. The new distribution showed significant improvement in the resulting model. At this point, all data sets are scaled with the same parameters, and then the scaled train set is given to the SVM train to make the models for each of the object class's context.

Different values for the parameters are considered during training, and each resulting model is applied to samples from the validation set. Classification results with different examined values of these parameters are compared, and the best models are selected. The results and evaluation procedure are discussed in Section 3.3.

## 3.2 Evaluation metric

Here we define a specific metric and evaluation method that we proposed and considered in this work. Metrics such as accuracy of prediction, with their standard definition, are not the best ways to evaluate the performance of our system. Accuracy is measuring the number of correctly classified samples with respect to the total number of samples. Since the expected result is a correct likelihood map of the object context and not the object itself, we do not have direct access to the number of correct classifications (The annotation labeled the object and not its context).

Based on the problem definition, we are looking for the possible context of an object, which is the regions in a scene where the object is expected to be found, not just regions where an instance of the object is actually present. A table is a potential context for a cup, regardless of a cup being on it or not. Intuitively, a set of points that belong to an actual object (if there is an actual object in the scene) is a subset of all potentially candidate points for the object class in that scene. Therefore, actual object points or labeled points (Lp) are a

subset of the predicted positive points (Pp) if the model and classifier perform well (equation 3.1).

$$Lp \subset Pp \tag{3.1}$$

In other words, from a good result, if we randomly pick a point that belongs to an actual instance of the object of interest, that point should be within the predicted positive points.

A similar metric is used in the recently published and praised work [2]. With this definition, there would be a problem: If all points get predicted as positive, then actual object points would definitely be a subset of it, while our classification has failed. To address this issue, we define a good result as **predicted set of locations that reduces our search space for the object as much as possible while it preserves the high probability of finding the object**.

Equation 3.2 shows the evaluation metric, where $E(\tau)$ is the the value of the metric with respect to a probability threshold $\tau$. This threshold sets the minimum probability value that the prediction for a point should have, to be considered as positive.

The threshold is computed through a tuning process where its value gets changed in a loop. For each threshold value, the probability of having the labeled object within the positive predictions is calculated. The optimal outcome would be a value for the threshold, which results in the smallest subset of the pointcloud, predicted as positive, while the chance of finding the object in the predicted region stays %100. For instance, the threshold takes a value that separates the top five percent of the points with respect to their prediction value. Then, the probability of having the object in this fraction of points is computed.

$Pp_\tau$ is Predicted Positive sample with respect to the value of the threshold, and $n(x)$ is the number of members in the set $x$, e.g., $n(Lp)$ is the number of points belonging to Lp. Lp is the labeled positive points in the dataset or, in other words, the annotated object.

$$E(\tau) = \frac{n(Pp_\tau \cap Lp)}{n(Lp)} \tag{3.2}$$

This value is between zero and one, and a value closer to one, while the threshold is high enough, shows a better result. This metric is also used in experiments to find the best models and parameters. Here we also translate the sample-base results into point-base, which means the result is assigned to the point for which the features are taken.

## 3.3 Results

In this section, we look at some results and analyze and evaluate them to see how good our method performs.

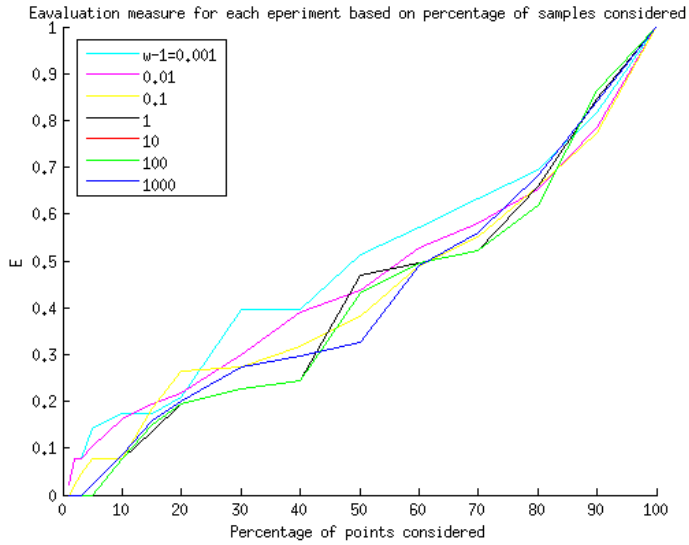



**Figure 3.4:** Evaluation diagram for experiments with a single Gaussian kernel, applied on the whole feature vector. Each line corresponds to a value for w-1.(Best viewed in color.)

We have two sets of experiments; in the first one, the learning is done using one Gaussian kernel on the whole feature vector. In the second set, we used two kernels of type chi-square with equal weights. One is applied on the first two fields of the feature vector(orientation and height) and the other on the VFH part.

Figure 3.4 shows this evaluation for seven experiments with different values for w(-1) which is the parameter, setting the weight of the negative class in SVM classifier. In these experiments, a single Gaussian kernel is employed to build the model of trash bin context. The value is depicted with respect to the fraction of points considered.

If we take the best models, it can be inferred from the curves that, for instance, considering the top five percent of the points contains twenty percent of the object, which is not a bad result. In other words, by reducing the search space to only five percent, we still have four times more a chance of finding the object.

Figure 3.5 shows the prediction of trash bin context on a full pointcloud of an office. The marked region shows the points with the highest probability values(top 1%). This predicted context reaches the trash bin, which is present in the scene when the probability threshold reaches the probability of the top

**Table 3.2:** Performance of the system for four object classes.

| Object | E in top 5% of points | The ratio that E reaches to 1 |
|---|---|---|
| Mouse | 0.6667 | 15 |
| Telephone | 1 | 5 |
| Trash Bin | 0.3333 | 40 |
| Wiper | 0.42 | 60 |

5% of points. The interesting point is that this trained model is suggesting that the corner is the best location in this scene to find a trash bin or place one (based on the train data).

In later experiments that we used multi-kernel of type chi-square, there was a significant improvement in the results. As the first two fields of our feature vector are two float values and the rest is a histogram, they are to be treated in different ways. So we separated the kernels applied to each of these parts. The visualized results of predictions using these kernels with equal weights are shown in Figure 3.6 for the four objects we used in these experiments.

In Figure 3.6, the predicted contexts of four object used in experiments are projected on the pointcloud of an office. It should be noticed that the regions shown in cyan are blobs with a keypoint in the center, which is predicted as context.

The corresponding evaluation values for results shown in Figure 3.6 are depicted in Figure 3.7. A simple comparison between curves in this figure and Figure 3.4 shows the improvement of the results.

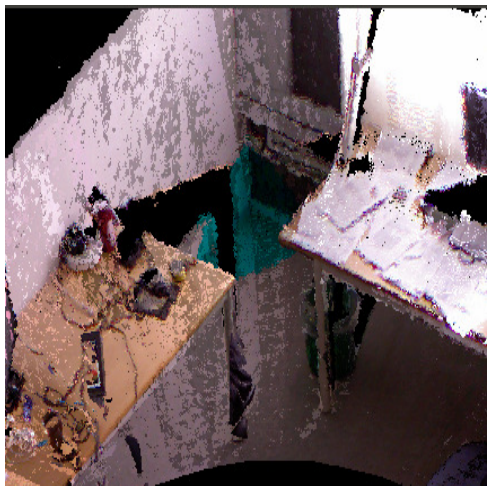
In order to compare the system's performance regarding each of these four object classes, the values of E corresponding to each selected threshold (resulting in, a certain ratio of selected points with respect to all points in the pointcloud) are considered for each of these object classes. Table 3.2 contains these values for the ratio of top 5%.

As it can be seen in table 3.2, the best performance of the system is for Telephone and after that for Mouse. These two are often found in a similar context which can be a good reason for these results. At least based on our train sets, the context for trash bin can have a wider variety compared to mouse and telephone. That is why its performance value is the lowest.

In the wiper case, the reason for its low performance should be the wide range of heights this object can be located in. It can also be placed on a table. We consider the height as an important feature in building our models, which makes it very effective. The effectiveness of height seems to be not so helpful in the case of wiper. A solution can be using smaller weights for the height feature in modeling these object classes.

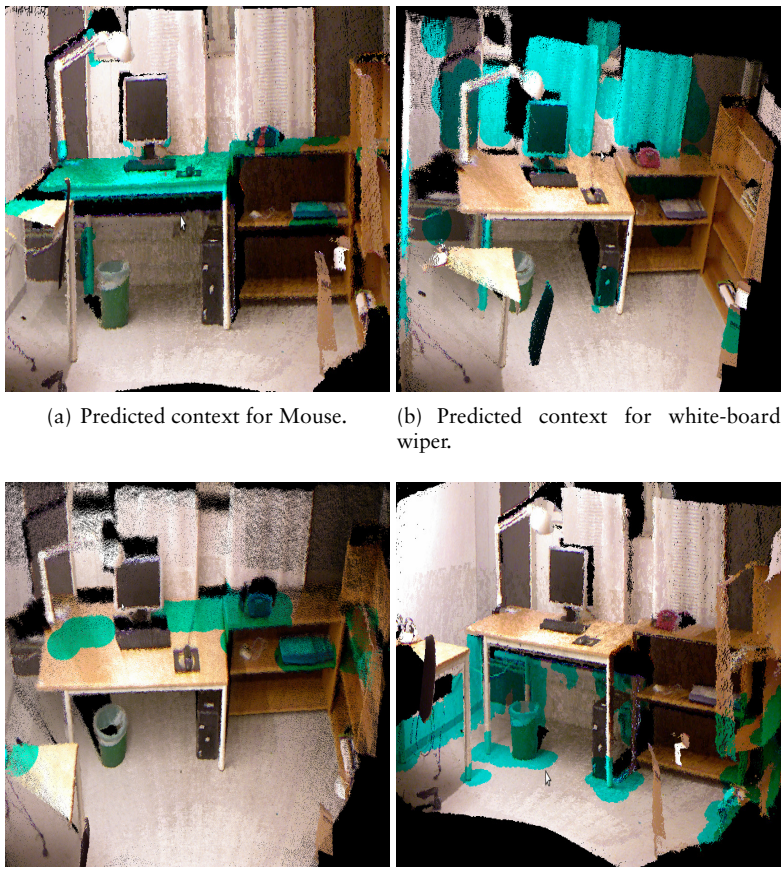
Figure 3.8 shows the predicted contexts of these four object class on the pointcloud of an office from the test set. In this test cloud, no instance of wiper,

(a) A test pointcloud for trash bin model, marked region with red circle is the predicted context.



(b) Focused to predicted region.

**Figure 3.5:** Prediction of Trash bin context on a full pointcloud of an office. The marked region shows the points with highest probability values(top 1%).(best viewed in color)



(a) Predicted context for Mouse.

(b) Predicted context for white-board wiper.

(c) Predicted context for Telephone.

(d) Predicted context for Trash Bin.

**Figure 3.6:** Visualization of context model prediction on sample cloud 1 as a validation cloud, for four objects. The key point predicted as positive and the blob around it is projected on the pointcloud in cyan color.(best viewed in color)

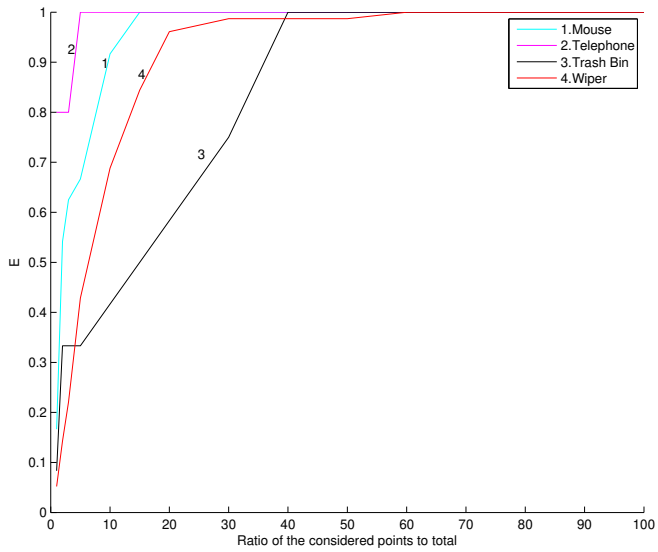



**Figure 3.7:** Evaluation results of context prediction for four objects Mouse, Telephone, Trash Bin and wiper. These results are from experiments using chi-square multi kernel.

(a) Predicted context for Mouse.

(b) Predicted context for white-board wiper.

(c) Predicted context for Telephone.

(d) Predicted context for Trash Bin.

**Figure 3.8:** Visualization of context model prediction on sample cloud 2 as a test cloud for four objects. This pointcloud does not include these objects except trash bin.(best viewed in color)

mouse, or telephone is present. The reasonability of the predicted object contexts is visible in this figure despite the absence of those instances.

## Summery

As mentioned in Section 3.1, in the experiments, we used pointclouds of different place categories like kitchen, bathroom, and offices, which included some instances of the four object category. Each of these pointclouds, on average, contained around eight million points, out of which we extracted samples for several keypoints. The average number of samples extracted from each pointcloud is approximately 50000. Samples from several pointclouds are assembled in a set from which a train set is derived. The set of clouds are divided into train (45%), validation (15%), and test(40%). Parameter and best model selection are done using validation sets.

The results show that the predicted context can reduce the search space significantly while preserves the high chance of finding the object of interest. It also implies that the context of objects are modeled reliably, making it useful in place categorization where we need to know just the possibility of the presence of an object class in a query place (discussed in Chapter 4). By tweaking the prediction threshold, the search space can be extended in a smart way rather than a greedy manner.

The time needed to train a model depends on the size of the train set and the used parameters, which in these experiments is in average 89 second. Test takes approximately 45 seconds, and the time needed for feature extraction per sample is 68 ms.

# Chapter 4
# Conclusion and future work

In this thesis, we proposed a method to build 3D models for objects' context, using geometrical features extracted from pointclouds of scenes or rooms. Pointclouds are captured using *Microsoft Kinect* with *OpenNi* driver and *RGBD-SLAM*. A set of pointclouds is gathered from different place categories, containing different objects.

Using our Annotation tool, we annotated some object classes in some of these pointclouds to be used as the source of our train samples. We defined a descriptor for the context which can efficiently encode some valuable features into the model.

The descriptor consists of average height, pose, and the local geometry of the context. The definition of this descriptor includes a viewpoint for each of the feature vectors, which allows us to study the surfaces from several different directions. Therefore, the resulted model from this method is not limited to particular viewpoints. This was a limitation in most of the other works we reviewed in 1.3. The features extracted from entire pointclouds, with annotated objects, get divided into train and validation sets to be used by the SVM and build context models. In order to address the different formats used in the feature vector (floats and histograms), two kernels are used.

Finally, models are applied to test data, and predictions are projected on their corresponding regions in the visualized pointclouds.

Furthermore, we proposed a metric and an evaluation method to assess the results in a meaningful way. The results show that the proposed method is very effective for modeling and detecting the object's context. It can significantly **reduces the search space** and **the chance of having false positives** in object detection tasks. So it is providing a very useful prior for object detection. It should work for any ambient condition, and scaling issues applicable to 2D images do not apply to it.

Using our model and method, besides predicting the context of objects, it can suggest where, on, or around that detected context we should expect to find the object. This is another unique benefit of our method. It can be done using

the *OPoints* from the pair of (QPoint, OPoint) (see 2.4.3). As mentioned, samples are extracted for several instances of this pair of points in test sets too. After applying the model on the test samples and getting the predictions, we assigned the predicted class to the QPoints and if the prediction for it is context, marked the points belonging to the *Query blob* as the predicted context, which was the expected result from the system. Now, if we pick the Opoint from the same sample, it represents a point that can potentially be an object point. This can help to show, for example, the object should be expected on which side of a surface that is detected as context. A desk's top surface can be detected as a context for a mouse. Using the OPoint, we can show that the mouse is expected to be above the surface and not on the other side. And it is less likely for a mouse to be very close to the edges of the desk or on the corners.

In addition, we present a method for employing the object context model in place categorization. In the next section, this method is analytically discussed. To the best of our knowledge, this work is a pioneer in suggesting to use 3D context models of objects in building place descriptors and using full pointclouds of places. These full pointclouds can be extended to a 3D map of an environment.

## 4.1 3D Object Context for place Classification

In this section, we discuss the proposed idea of using object context in classification of places. Place classification in here, mainly mean to determine the category of a visited place so that some resulting semantics can be added to and understood by the system.

Adding semantic information to maps, to be used by mobile robots, enhances human-robot interaction. It helps in mobile robots localization and object detection. The ability of a robot to understand semantics of space and associate spatial locations with semantic terms such as *kitchen* or *corridor*, provides a more clear idea for the robot and its users of its location than a pure topological or metric position [14].

Regarding object detection, having semantic information of a place in association with a ground truth about the objects expected to be present in that place, makes the search for an object faster and more successful. It helps robotic systems to communicate with humans and interpret environments built by and for them.

Concepts such as *office* or *kitchen* are different categories for a room based on its functionality. Other categorizations can be based on its spatial properties such as its shape, size or general appearance.

Figure 4.1 shows an overview of a semantic mapping system proposed by Pronobis and Jensfelt in [14]. As it is illustrated in this picture, inputs to this system are models of objects, shape, size and appearance in the company of a common-sense knowledge database. It is considered as a future work for

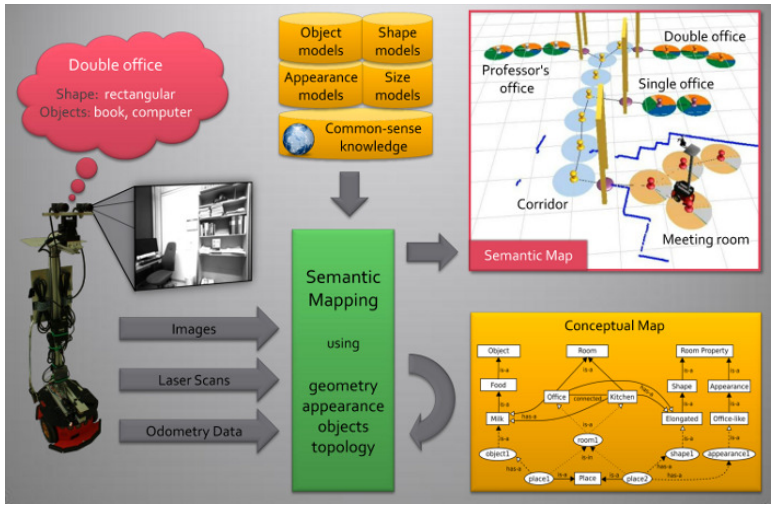



**Figure 4.1:** Overview of a semantic mapping system by Pronobis in [14](best viewed in color).

this thesis that 3D *context* models replace *object* models in this system and with some improvements a more robust and complete representation for places would be achieved.

### 4.1.1 Brief overview of approaches to place classification

Place categorization based on vision in its first stages was focused on classifying single 2D images of an indoor or outdoor scene. Data achieved from laser range sensors are also popular due to their robustness to environmental variations and faster process time [14]. Several researchers in computer vision community addressed the problem of place classification, some examples are reviewed here.

Li and Perona in [9] represented images of scenes as a collection of local regions called *codewords* achieved from unsupervised learning. Similarly, Lazebnik et al. in [8] extended a *bag-of-words* approach in a computationally efficient way by introducing a spacial pyramid which contains approximate global geometric correspondences between local features.

Olivia and Torralba proposed a scene representation called *gist of the scene* in [12] which is used by Torralba et al. in [23] and [21] for place categorization. They used 2D object context information as mentioned in previous chapters in their work. Later, Quattoni and Torralba in [15] combined the gist of the scene with local features and reported significant improvement in their results.

Also in robotics, researchers worked through capturing some semantics mostly using laser range data. Buschka and Saffiotti classified different parts of grid maps into two categories: rooms and corridors in [4].

There are approaches that mostly rely on object information for place categorization. Usually a more fine grained description of a place based on its functionality needs association of some key objects for each place category. As mentioned in Chapter 1, rooms are usually classified into categories like Office or kitchen based on their functionality which is tightly dependent to the objects found within them.

In [24], Vasudevan et al. suggest a hierarchical probabilistic representation of space using object information. They use SIFT features to detect and recognize objects then create a local probabilistic object graph for the place. They also detect doorways to separate rooms from each other in their map. Using object graphs for each visited place they make a global topological representation of an environment.

[16] and [25] also used object based approaches using models trained in advance in a supervised manner. In [16], Ranganathan and Dellaert train object models using visual features capturing their shape and appearance from roughly segmented and labeled images. They also added 3D locations of the objects using stereo range data.

Pronobis in [14] integrated multiple visual cues with geometric information from laser range data. Object detection in association of a common-sense knowledge database completes the information needed to determine the category of places in his semantic mapping system.

### 4.1.2 Ground Truth

As mentioned before, a ground truth is needed to make the relation between the class of a place and object contexts present in it. This knowledge can be achieved by analyzing available common-sense knowledge databases like **Open Mind** [11], popular search engines such as Google Image Search, image repositories like Flicker or directly from a human user inputs.

Viswanathan et al. in [25], have performed an automated learning of object-place relations on an on-line annotated database such as Label me. Then object detectors are trained on some of the most frequently occurring objects. In our case instead of objects we train their context detectors.

Based on the dependencies between place category and its objects also between different object categories we can make probabilistic models that represent these relations and dependencies. Some objects are more discriminative for a place category which should have a more significant role in deciding the category of the place. For instance, a water tap in the kitchen or its sink as its context are quite discriminative for this place category.

$$w_i = p(R_c \mid O_i) \tag{4.1}$$

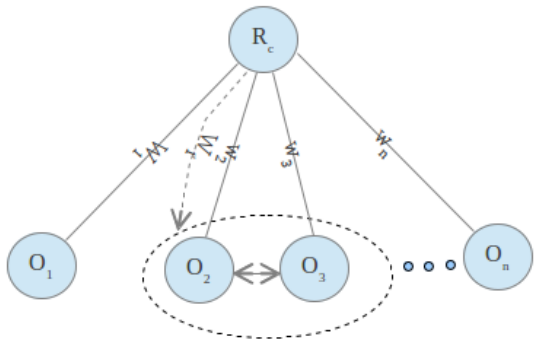



**Figure 4.2:** This figure shows dependencies between a room category and key object classes, dashed lines shows dependency to a combination of object categories.

Equation 4.1 shows the dependency between a room category ($R_c$) and each object category ($O_i$). Sometime, the dependency lies on a combination of objects. A combination of shelves and books can be discriminative for an office, while shelves and dishes are good for kitchen.

$$w_{i'} = p(R_c \mid O_j, O_k) \tag{4.2}$$

Dependency of the room category to a combination of object categories is shown in equation 4.2. A model can be achieved like the one depicted in figure 4.2 in which singular dependencies are shown by continuous lines and combinational dependency is illustrated by dashed line. A normalized weight can be computed based on this analysis for each object category.

### 4.1.3 Benefits of Object Context for Place Classification

Some of the benefits of employing context models in place classification can be briefly mentioned as follows:

- It provides a simple way to include human annotations from commonsense knowledge bases into place representation.
- It brings in object information while no fine grained object detection is needed.
- It is a way to make place models more universal representing them based on object context information instead of object information which gives a generalization to the model.

### 4.1.4 Place classification

A method is proposed here to compute a score for each place category based on the amount of objects possible to be found in a place. In each place category based on its ground truth learned from a lot of examples we compute a weighted sum of the scores of each key object class. The score for place categories should represent the followings:

- Probability of presence of an object class in a place.
- Weight or ratio for this presence.
- Based on a ground truth a probability of being a specific place category.

Applying context models on the point-cloud of a room, we get a subset of points with highest probability of being the context. In other words, we can estimate the amount of points belonging to a context category in that point-cloud. An average probability for context points belonging to each context category can be computed and assigned to that category. Using the ratio of points in each category and the value of the average probability a score is estimated for each context category (Equation 4.3).

$$Sc_i = Avg(p(c_i)) * \frac{points \in c_i}{points} \tag{4.3}$$

$Sc_i$ is the score for object category $i$, $Avg(x)$ computes average and the last element of the equation shows the ratio of points in context category $i$ with respect to all points in the point-cloud. Employing weights resulted from ground truth analysis (4.1.2) a score for each place category can be estimated.

$$Spc_j = w_1 * Sc_1 + w_2 * Sc_2 \ldots + w_k * Sc_k \tag{4.4}$$

In equation 4.4, $Spc_j$ is the score for place category $j$, $k$ is the number of key object categories. The corresponding weight for each object category is shown by $w_k$. Both scores should be normalized to a range between zero and one to be comparable for different place categories. Here we used object category and context category as equivalents. It is noticeable that the proposed equations meant to introduce the general relation between elements. The exact relations need to be achieved through experiments.

## 4.2 Future work

There are some suggestions for improvements and some potential future works which are mentioned here.

- The quality of the point-clouds are an important factor, so using better means and methods to capture and build point-clouds with higher quality

makes an improvement in results. Doing some pre-process like smoothing and removing noise is even a more practical improvement.

- Larger train set with more variety of places and object instances is essential. Gathering data in a smart way for the train set which considers several possible situations can be more efficient than just increasing the number of samples.
- Some more features can be added to make the model more descriptive and discriminative. Visual features should be considered for this improvement.
- A comparative evaluation can be done between object detection by a known high performance detector with using context model and without.
- Develop the system to be able to run in real time and on-line situations.
- Employ and experiment place categorization using this model and the proposed method.

# References


[1] A. Aydemir, D. Henell, P. Jensfelt, and R. Shilkrot. Kinect@ home: Crowdsourcing a large 3d dataset of real environments. In *2012 AAAI Spring Symposium Series*, 2012. (Cited on page 31.)

[2] A. Aydemir and P. Jensfelt. Exploiting and modeling local 3d structure for predicting object locations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012. (Cited on pages 3, 4, 13, and 36.)

[3] Nima Behzad. Point cloud annotation tool. `git@github.com:NimaB/PointCloud--Annotation.git`, 2012. (Cited on page 19.)

[4] Pär Buschka and Alessandro Saffiotti. A virtual sensor for room detection. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 637–642. IEEE, 2002. (Cited on page 48.)

[5] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. (Cited on page 15.)

[6] "Nikolas Engelhard, Felix Endres, Jürgen Hess, Jürgen Sturm, and Wolfram Burgard". "real-time 3d visual slam with a hand-held rgb-d camera". In "*Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*", "Vasteras, Sweden", "April" "2011". (Cited on page 31.)

[7] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. `http://eigen.tuxfamily.org`, 2010. (Cited on page 15.)

[8] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169 – 2178, 2006. (Cited on page 47.)

[9] Fei-Fei Li and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 524–531, Washington, DC, USA, 2005. IEEE Computer Society. (Cited on page 47.)

[10] Fuxin Li, João Carreira, and Cristian Sminchisescu. Object recognition as ranking holistic figure-ground hypotheses. In *CVPR*, 2010. (Cited on page 15.)

[11] MIT. Open mind common sense. `http://openmind.media.mit.edu/`. (Cited on page 48.)

[12] Aude Oliva, Antonio Torralba, et al. Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, 155:23, 2006. (Cited on page 47.)

[13] Roland Perko and AleÅ¡ Leonardis. Context driven focus of attention for object detection. In Lucas Paletta and Erich Rome, editors, *Attention in Cognitive Systems. Theories and Systems from an Interdisciplinary Viewpoint*, volume 4840 of *Lecture Notes in Computer Science*, pages 216–233. Springer Berlin Heidelberg, 2007. (Cited on pages 3 and 4.)

[14] Andrzej Pronobis. *Semantic Mapping with Mobile Robots*. PhD thesis, KTH Royal Institute of Technology, Stockholm, Sweden, jun 2011. (Cited on pages 46, 47, and 48.)

[15] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes, 2009. (Cited on page 47.)

[16] Ananth Ranganathan and Frank Dellaert. Semantic modeling of places using objects. In *Proceedings of the 2007 Robotics: Science and Systems Conference*, volume 3, pages 27–30, 2007. (Cited on page 48.)

[17] "Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu". "estimating vfh signatures for a set of points". (Cited on pages 13, 14, and 15.)

[18] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. (Cited on pages 10 and 15.)

[19] R.B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155 –2162, oct. 2010. (Cited on page 13.)

[20] T. Southey and J.J. Little. Object discovery through motion, appearance and shape. In *AAAI Workshop on Cognitive Robotics, Technical Report WS-06-03. AAAI Press*, 2006. (Cited on page 7.)

[21] A. Torralba, K.P. Murphy, W.T. Freeman, and M.A. Rubin. Context-based vision system for place and object recognition. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 273 –280 vol.1, oct. 2003. (Cited on pages 3, 4, and 47.)

[22] A. Torralba, A. Oliva, M. S. Castelhano, and J. M. Henderson. Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search. *Psychological Review*, 113(4):766–786, October 2006. (Cited on page 1.)

[23] Antonio Torralba. Contextual priming for object detection. *International Journal of Computer Vision*, 53:169–191, 2003. (Cited on pages 3, 4, and 47.)

[24] Shrihari Vasudevan, Stefan GÃ¤chter, Viet Nguyen, and Roland Siegwart. Cognitive maps for mobile robots an object based approach. *Robotics and Autonomous Systems*, *55*(5):359–371, 2007. <ce:title>From Sensors to Human Spatial Concepts</ce:title>. (Cited on page 48.)

[25] P. Viswanathan, T. Southey, J.J. Little, and A. Mackworth. Automated place classification using object detection. In *Computer and Robot Vision (CRV), 2010 Canadian Conference on*, pages 324 –330, 31 2010-june 2 2010. (Cited on pages 7 and 48.)