ÖREBRO UNIVERSITY

International Master's Thesis

Modeling 3D Object Context

Nima Behzad
Technology

Studies from the Department of Technology at Örebro University
örebro 2013

Modeling 3D Object Context

Studies from the Department of Technology
at Örebro University

Nima Behzad

Modeling 3D Object Context

Supervisors: Dr. Andrzej Pronobis
Dr. Todor Stayanov

Title: Modeling 3D Object Context

ISSN 1650-8580

Abstract

In this work we introduce a method and descriptor for object context in full 3D pointclouds of places. Among various applications for this we are suggesting it to be used for place classification and semantic mapping. 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 supervisors, Andrzej and Todor and also Professor Lilienthal who introduced me to the world of computer vision. Special thanks to my parents and my lovely fiancÃ©e for their kindness and support.

Contents

1 Introduction 1
  1.1 Contributions . . . 2
  1.2 Related Work . . . 3
  1.3 Outline . . . 4

2 Modeling Object Context 5
  2.1 Problem Statement . . . 5
  2.2 Motivation and Applications . . . 5
  2.3 Challenges . . . 6
    2.3.1 Challenges related to Applications . . . 6
    2.3.2 Challenges related to modeling the Object Context . . . 7
    2.3.3 2D vs 3D context, benefits of 3D . . . 7

3 3D Model of Object Context 9
  3.1 An example Scenario . . . 9
  3.2 Implementation . . . 9
    3.2.1 The Viewpoint Feature Histogram (VFH) . . . 10
    3.2.2 System Modules . . . 12
    3.2.3 Parameters . . . 19
  3.3 Experimental Setup . . . 21
  3.4 Results . . . 22
    3.4.1 Evaluation . . . 23

4 Using Object Context for Place Classification 29
  4.1 Brief overview of approaches to place classification . . . 30
  4.2 Ground Truth . . . 31
  4.3 Benefits of Object Context for Place Classification . . . 32
  4.4 Place classification . . . 32

5 Conclusion and future work 35

References 37

List of Figures

1.1 Illustration of a sample Context. 2

3.1 System Overview 10
3.2 ViewPoint Component of VFH 11
3.3 FPFH elements. 12
3.4 VFH histogram 13
3.5 Example scene and object for Annotation tool 16
3.6 Annotation tool result 17
3.7 Illustration of the items used in Feature Extract. 18
3.8 Feature extract structure 19
3.9 Compare two random negative samples 22
3.10 Compare two random Positive samples 23
3.11 Distance between random negative samples 24
3.12 Distance between random Positive samples 25
3.13 Distance of Positive and Negative samples 26
3.14 Evaluation result of 80 experiments 26
3.15 Evaluation diagram(1) 27

4.1 Overview of a semantic mapping system 30
4.2 Probabilistic graphical model of place-object. 32

List of Tables

List of Algorithms

1 PreProcess. 14
2 Annotation. 15
3 Feature Extract. 20

Chapter 1
Introduction

A stereotypical image of a robot is a humanoid in a servant costume which can do all the chores at home and perhaps even more. This is not too far from scientific projects which are being run in research labs. Assuming such goal, human behavior becomes a natural inspiration for the algorithms governing the behavior of a robot in the assigned tasks and situations. Imagining a simple but common task of find-and-fetch, first solution that could be thought of, is to learn how exactly a human would perform the task. Then, how the procedure can be translated to an algorithm executable on a robotic system. Often such translation can not be direct due to the differences in embodiments and lack of complete understanding of underlying processes.

In case of the find-and-fetch task, a human is likely to imagine what does the target object look like, where and it what situation it is often found. If the object is known, a natural behavior is to visit the most probable location for the object first. 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.

A similar behavior could be implemented on a mobile robot. A valuable prior knowledge for an object detection/recognition problem could be semantic knowledge about places and possible context for the object, which not only can reduce the search time but also can increase the chance of success and correctness of the results.

Torralba et al. in [22] point to the fact that there are many experiments showing that the human visual system uses contextual information to facilitate search for objects. Moreover, he proposes the use of context for object detection in 2D images.

In this work, we focus on 3D information and therefore, consider surfaces surrounding the object as its context. This includes surfaces the object is located on or next to and also other objects which are usually beside the

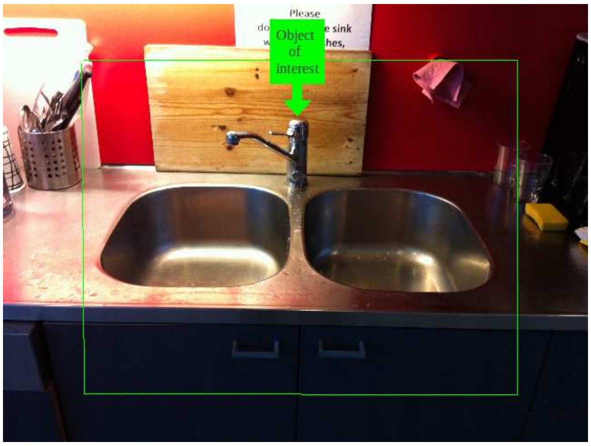

Figure 1.1: A water tap as the object of interest and what we call its context. The context is surfaces within the green bounding box.

object of interest. For instance, if the object of interest is a *kitchen water tap*, it's context ca be the *kitchen sink* and the wall behind it. (Figure 1.1)

1.1 Contributions

In this work, we propose a method to build 3D models of object context, which not only can be used as a prior to an object detector, but also as a useful knowledge in building 3D representation for places. Benefiting from this model, the area needed to be searched in an object detection task can be reduced to parts of the scene with high probability of being the context for the object of interest. It can also have a positive effect on number of true positive 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 applied supervised learning methods on data extracted from manually annotated objects in point-clouds. Point-closed that we used as input are ful 3D representaions of a place such as an office, a kitchen, etc. The learning is done on features which are representing local geometry and spacial location of object context.

Most important points about this work ca be briefly mentioned as follows:

- Emphasizing on the role of context in object detection and it's related applications.
- The features and methodology suggested to model the Object Context.

- The input data used in this work are full point-clouds of places rather than single frames with depth information or 2D images.
- Suggesting the application of Context model in place classification.

Applications in place classification would be discussed later in chapter 4.

1.2 Related Work

In this section, some related works focusing on using contextual information for object detection/recognition or place classification are briefly reviewed. Moreover, other works which contribute 3D features and descriptors for similar purposes are mentioned.

Object Context

In [21] Torralba et al. present a low dimensional global image representation capturing useful information for recognition of places and show how contextual information can provide priors for object recognition. As mentioned before, this work considers 2D images only.

In [23] Torralba uses the relationship between object and it's context properties from 2D images to build model that helps the focus of an object detector's attention and also for scale selection.

In [13] Perko and Leonardis extract and learn contextual information for objects from examples. Then they use this learned context to calculate a focus of attention, that represents a prior for object detection. Their work is also used 2D features extracted from images.

Aydemir et al. in [2] learn 3D context of objects to predict object locations in real world scenes. They used separate RGB and depth frames and used histograms of surface normals as a 3D geometrical feature to model object context.

In [19] Rusu et al. proposed Viewpoint Feature Histogram that encodes 3D geometry and pose. They suggested that this descriptor can be used for object recognition and pose estimation with high reliability. In this work we employ a descriptor, which contains their descriptor, for modeling context of objects.

Place classification

In [25], Vasudevan and Siegwart proposed a representation for space based on objects. The authors discuss several algorithms, some of which only uses object category presence and some are more sophisticated using both objects and their relationships and also spacial structure of places.

1.3 Outline

The rest of this thesis is structured as follows: In chapter 2, the problem definition and more about motivation, applications and challenges would be presented. chapter3, dwscribes our method and details about implementation, experiments and evaluation. In chapter4, applications of our model and descriptor in place classification is discussed and an analytical study is done for the results that could be expected for this application. Finally, we have a review while next steps that could be considered for this work and possible improvements and also future works are mentioned in 5.

Chapter 2
Modeling Object Context

2.1 Problem Statement

As discussed in the previous chapter, in order to facilitate object detection and provide a means to classify places based on key objects for a place category, we have proposed a structure and a method to build models for object context.

Based on the definition stated in chapter 1, by context of the object we mean the surfaces surrounding it. The problem that is addressed in this research, is how to define suitable features and a method to train a model, using those features, for the context of an object class. The trained model would be applied on point-clouds of new scenes to determine location of possible contexts that are available in the scene. The expected result, is a prediction probability for all areas in a point-cloud that shows the possibility of that area to be a context for the object class of interest.

2.2 Motivation and Applications

The main motive for this work is achieving a complete and robust 3D descriptor for places. When a human agent enters a room, he is able to decide about the category of the place based on a simple and shallow perception ,with a good confidence. Even if the room is not furnished yet and there is not many objects in it to help the agent in determining it's category.

Although, the Context Model, which we are talking about in this work, is not enough for making this decision , but is a useful prior. Among the applications that is considerable for the model, most obvious ones are as follows:

Object Detection

Object detection is an expensive task and not always an easy job. Scale of the object is an issue here. Searching for context of an object is significantly easier than the object itself, particularly when the searched object is in a rather small scale. When the context is detected, then the search for the actual object is done in a reduced area, where is predicted as the context. Scale selection in object detection is another problem that detecting the context can diminish it.

In addition, false positives are avoided or at least reduced. When a table is detected as the context for a cup, a cup like object which is located on the floor is not predicted as a positive detection.

Place Classification

A very important 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 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 a discriminative object. If the robot can find such an object in a place, it can have a good confidence about that place to be a kitchen.

Viswanathan et al. in [26], also pointed to the fact that object detection is a good prior for place categorization. T. Southey and J. J. Little in [20] argued that, rooms are defined by their geometric properties, while definition of places are based on objects they contain and activities takes place in them.

On the other hand, the place classification system would be more robust if it is able to 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 object finds it's application. When the context is present, there is no need to detect the actual object for the purpose of classifying the place.

Object Placement

When a robot needs to put an object in a suitable place, benefiting from Object context information, it needs only to search for matches for the context of the object it is holding.

2.3 Challenges

2.3.1 Challenges related to Applications

Place Classification

Finding object classes which are discriminative for places is not easy for all place categories. In addition, Context model for different object classes can be similar which decreases certainty of place classification.

Object Detection

Object detection can be challenging when the size of the object compared to the scene size is very small. Illumination condition , occlusion and pose of the object can affect the time needed for the search and performance of the system so much.

Context model can do a great help here. Detecting a table as a context for cups is easy due to its scale, compared to the cup itself and is less affected by illumination and occlusion. When the table is detected, recognizing objects on the table is facilitated regardless of the challenges mentioned above.

Where to put the Object

If the context model provided for an object class, which a robot wants to find a place to put it, is not general enough it can cause a problem. For instance, in a scenario that a robot is looking for any suitable surface, on which it is possible to put a cup, if the context model is just suggesting a table as a suitable place, it confuses the robot. There may be no table around, but flat surfaces which can hold a cup.

2.3.2 Challenges related to modeling the Object Context

- The ability of the features to separate context from non-context points. The features that are used to train the context model should be discriminative to some extent. The context , as is described in Chapter 1, usually is not a single body. It consists of separate surfaces or even objects, which reduces the distinctiveness of the features.

- Employing a train set without enough generalization. There could be situations and scenes which are completely different from what have been considered in train set to build the context model.

- The quality of point-clouds is a critical issue. The amount of smoothness in the point-cloud and missing points have a significant effect on the extracted features, which are used to describe surfaces, and as a result, on the context model. When point-clouds are being captured, if due to the viewpoints of the sensor some parts of a surface get ignored, the result misses many important points. The geometry that is extracted from such surfaces is not accurate and in some cases are wrong.

2.3.3 2D vs 3D context, benefits of 3D

Benefiting from devices such as Microsoft kinect , has made 3D data widely accessible. Therefore, many researchers and academics have shown great desire to use 3D information in their researches, which gives a lot more information about our surrounding 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 easily and directly have access to depth information and 3D coordinates of any points that are perceived.

Geometrical features, which are obtained from 3D information, enhances performance of systems significantly in tasks like feature based object detection compared to relying only on visual information. Particularly, in the problem we are dealing with, to build a model for object context , depending on features which only encode changes in intensity and color is not enough. The surfaces needed to be studied, 3D information facilitate and improves this study.

Chapter 3
3D Model of Object Context

3.1 An example Scenario

Consider the situation that your robot wants to look around in a building and make a map including semantic information about the places that it meets in the map. The robot builds a 3D map of the building using, for instance, a SLAM application. Employing the context models, which has been trained for different key object classes, the robot can estimate which object classes are probable to be found in each place that is present in the map.

By key objects we are referring to distinctive objects for a place category. These specific object classes should be chosen based on a ground truth which is defined in advance for a place category. It is discussed in 4 more detailed.

Then using the results, showing which object classes are expected in a place and based on the ground truth for key objects of each place category the place is classified.

3.2 Implementation

In this section the procedure for building the context model and applying it is described in detail. The input in this procedure is the point-cloud of a scene and the output is a list of probability values assigned to all regions in the scene. The probability value shows if the region is likely to be a context. The input cloud is first preprocessed and then get annotated with instances of the object class whose context model is being built. Feature extraction would be run on this annotated point-cloud and result in the train samples. Train samples would be combined from several different point-clouds for the same object class to make the Train Set. Then the Train Set is given to SVM classifier to make the Context Model. A block diagram in figure 3.1 shows the system overview.

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

The data type we used in this work is PCL point-cloud [18]. In this data type 3D coordinates and color information of a point is saved in a record and

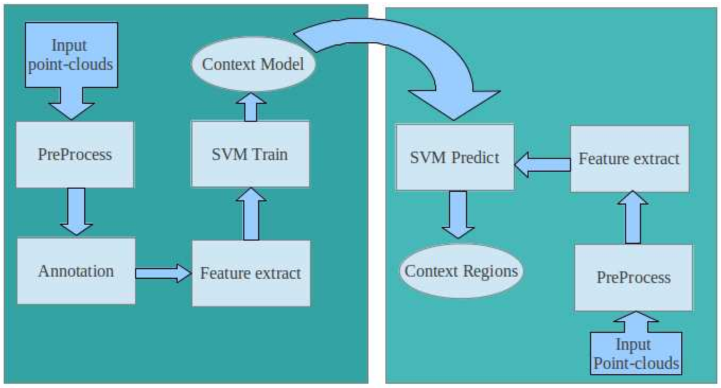

Figure 3.1: The overview of the system. Box on the left side shows the Train part of the system while the one on the right shows the prediction part.

then an array of these records forms the point-cloud. We used Microsoft Kinect with OpenNi driver to capture these point-clouds.

The feature vector we used consists of 2 parts. The first part includes two fields : First is the average height of the blob from which the features are extracted with respect to the floor. The Second field is the orientation of the blob with respect to the vector of gravity. And the rest of the fields are in second part of the feature vector which is a histogram. This histogram is called VFH which captures the geometry of the blob with respect to a specific view point. [19]

3.2.1 The Viewpoint Feature Histogram (VFH)

As it is described in PCL official website [17] and [19], VFH consists of two component as follows:

- First component is a histogram for viewpoints(128 bins)
- Second component is extended Fast Point feature histogram(FPFH:4*45 bins)

The viewpoint component is computed as a histogram of the angles that direction of the viewpoint makes with each surface normals. It is important to notice that, this angle is between the central direction of the viewpoint and the normal, not the view angle of the normal which would have made it not scale invariant.

The second component measures the relative pan, tilt and yaw angles that are measured between direction of the viewpoint at the central point and each

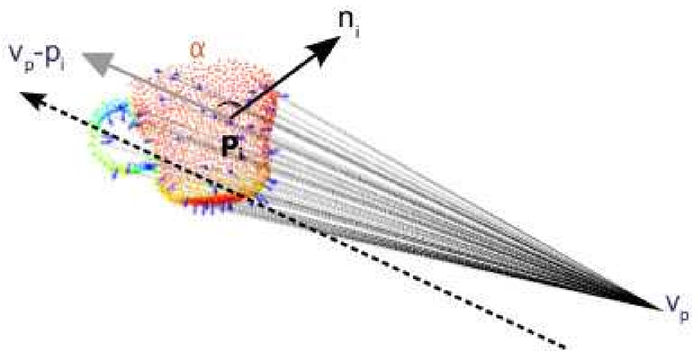

Figure 3.2: Viewpoint part of the feature vector.[17]

of the normals on the surface. So it consists of accumulated FPFH values for points in the query point-cloud. It is the reason it is referred here as extended FPFH.

FPFH or Fast Point Feature Histogram itself can be considered as a revised version of PFH. FPFH reduces the complexity of PFH algorithm from $O(nk^2)$ to $O(nk)$ where n is the number of points in the query point-cloud and k is the number of points in the considered neighborhood of each query point. This is the reason it is called Fast PFH. Although it reduces the informativeness of the feature but it would be still enough powerful and impressively faster to compute.

PFH captures 3D geometry of the surfaces around a query point and it is invariant to the 6D pose of the query blob's surface. It 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 four different element measured for every possible pairs of points in the query point's neighborhood ($O(nk^2)$). The elements consist of three angles and a distance.

In FPFH [10], instead of encoding this information for all pairs of points in a neighborhood, It will only take pairs between the query point and its neighbors ($O(nk)$). At this stage it is referred to as simplified PFH or SPFH. Then to keep the amount of information that this feature can provide, it also estimates SPFH for all k points in the neighborhood. The resulting values are integrated with a normalization to result in the FPFH of the query point(equation 3.1).

$$\mathsf{FPFH}(P_q) = \mathsf{SPFH}(P_q) + \frac{1}{k}\sum_{i=1}^{k} \frac{1}{\omega_k} \cdot \mathsf{SPFH}(k) \tag{3.1}$$

Figure 3.3 shows the encoded elements and their relation in FPFH.

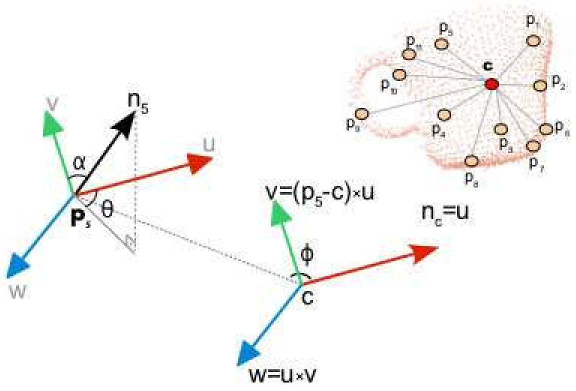

Figure 3.3: Three angles and a distance encoded in FPFH and the points considered.[17]

Finally, in VFH the resulting FPFH of points in the point cloud are integrated to produce one of the two components(extended FPFH). The histogram in this component has 180 bins (4*45 bin). By adding the view point component (128 bin), it produces a 308 bin histogram. Unlike FPFH, as a result of adding the viewpoint component VFH is dependent to viewpoint but it still preserve the property of being scale invariant. Figure 3.4 shows a sample plot of VFH and its components.

The images and definitions are taken from Point Cloud official website.

Used Libraries in Implementation

Most of the implementation is done in C++. There are some scripts for evaluation part of the results and creating the dataset for experiments written in MATLAB and also scripts creating the experiments and the experimental environment are in python. Libraries used in this work are as follow:

- PCL [18]
- Eigen [7]
- LIBSVM [5]

3.2.2 System Modules

As illustrated in figure 3.1, apart from SVM there are three modules in the system:

- PreProcess

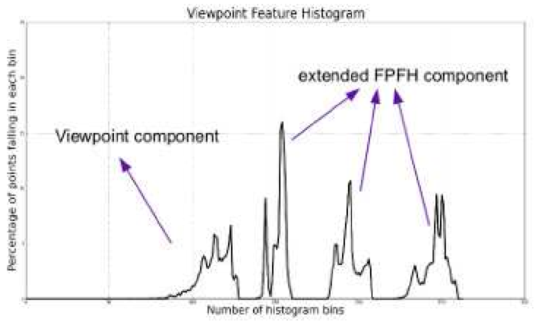

Figure 3.4: A sample plot of VFH and its components.[17]

- Annotation
- FeatureExtract

PreProcess

In this module the input point-clouds get prepared for annotation and feature extract. The point-clouds we used in this work are saved in "PCD" format and their point type is PointXYZRGB which is PCL point type. The processes for capturing these point clouds and tools we used would be discussed in section 3.3. First step in preparation of the input is transforming it. Based on the location and orientation of the sensor in time t_0 the resulting point cloud is transformed from sensor coordinate system to the world's coordinate system. As we are using another PCL point type in the rest of the process here a conversion of the transformed point-cloud into that type is also carried out.

This new type is PointXYZRGBL which includes a field to save a label for each point so we could use later in annotation tool and also in feature extract to decide about the class of the sample in train data set. The most important step, which is taken here is estimation of surface normals which is essential for feature extraction. The accuracy of the VFH feature is directly dependent to the accuracy of the normals. Here the normals for the whole point cloud is estimated once and saved in a separate pcd file to be used in feature extract.

There is also one more product for this module, which is discussed more in 3.2.2. It is a generated point-cloud which preserves size of the input cloud while complementing points, with respect to the input cloud, are added in it. 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

Algorithm 1 A brief algorithmic description of PreProcess.

```
Require: Input Point-cloud(XYZRGB).
Require: Sensor Location and orientation.
1: Transfer the input cloud based on Sensor location and orientation from
   camera coordinate to world coordinate, and store the result.
2: Estimate Normals and store.
3: Convert the Pointcloud_Transfered to Point type XYZRGBL and store in
   Pointcloud_Converted.
4: Estimate 3DMINMAX of the Pointcloud_Converted.
5: Using result from previous step, generate TFP pointcloud.
Ensure: Pointcloud_Transfered.
Ensure: Pointcloud_Normals.
Ensure: Pointcloud_Converted(XYZRGBL).
Ensure: Pointcloud_Generated(TFP).
```

- The transformed version of the input cloud
- Converted version into PointXYZRGBL
- TFP-cloud

Annotation

In this module the PointXYZRGBL version of the point-cloud is loaded to get annotated. The objects needed to be annotated in the scene would be selected by clicking roughly on their center. And the labels of the points belonging to them would be marked as object points. Objects from different classes could be labeled with different values (2).

Figure 3.5, shows a scene in an office at KTH. The object of interest here is the trash bin , which is marked by a green bounding box. This figure is an image of the scene whose point-cloud is available in our dataset and the annotation is done on the point-cloud. In figure 3.6 the annotated object could be seen in red within the scene's point-cloud. It can be seen that some part of the floor is also colored in red, which means that part of the floor is also annotated as the object. The reason is the object radius, that is given to annotation tool was not accurate enough or the center of the object was not picked accurately. But it does not harm the result that we need and this much of accuracy is more than enough. Usually annotation is done using a bounding box which consider the box surrounding the object as the object. Here, we exactly separate the points of the object by benefiting from the 3D coordinates of the points in the point-cloud.

Algorithm 2 A brief algorithmic description of Annotation.

```
Require: Pointcloud_Transfered(XYZRGB).
Require: Object radius(rough estimate).
1: Visualize input cloud.
2: for all objects to be annotated do
3:     Run Point picking procedure.
4:     Extract neighbor points indexes with respect to the input object radius.

5:     Label points whose indexes are extracted in previous step.
6: end for
7: Store the pointcloud with the labels in output cloud.
Ensure: Pointcloud_Annotated(XYZRGBL).
```

This package is available with source to be used by other researchers and get improved by developers.[3]

FeatureExtract

First, we Introduce some terms used in this part. They are as follows:

- Blob: A sub set of the Point-cloud which includes a number of points in a neighborhood within a specific radius.
- Query Blob: The blob from which the features are being extracted.
- QPoint: The Query Point that is the center of the query blob. It is a point on the context of the object.
- OPoint: The Object point is the point in surrounding of the Qpoint which is an object point in a positive sample.

Instances of these elements are illustrated in Figure 3.7 for the example scene that we saw in Figure 3.5. The red point in the figure shows the `Query Point`, which is a point picked from the context of the object and the features are extracted from the sphere surrounding it. The sphere is also visible in the figure which is the `Query blob`. The yellow point is the `OPoint` or object point which is a point on the annotated object.

In this module the converted and transformed point cloud and its normals are loaded. To build the data set that can be used later for train or test, Features are extracted for a number of keypoints from all over the point-cloud. The goal here is to get samples which are useful in building the context model for objects of interest. The way we did this is that for each key point, which is considered as a QPoint when it is selected for feature extract, we will take

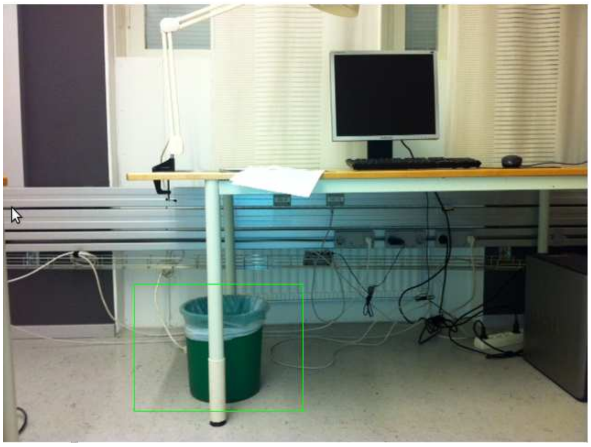

Figure 3.5: An example scene and a trash bin marked by a bounding box as the object which is being annotated (Best viewed in color).

a neighborhood of points and their corresponding point normals to extract features from (Query Blob).

The important idea applied here is the view point used in VFH. For each QPoint and its corresponding Query blob, a number of points in a radius from the QPoint is considered as the view points (OPoints). A sample is extracted for each pair of Qpoint and Opint_i. This way due to the difference in viewpoint, the features extracted for a single query point and its query blob would be slightly different. This will result in discrimination between same context viewed from different view point which is to our benefit.

As mentioned above these view points are candidates for being object points. In each sample if the OPoint in a pair of $(\mathsf{QPoint}, \mathsf{OPint})$, is actually an object point we want this sample to be labeled as a positive sample and if not, to be a negative one. The view point also can help to encode the most probable location of the object with respect to the positive context.

Another important point here is that we want the model to be able to locate candidates for context in a test point-cloud where object may be not present at the moment but its context is. Therefor, we need the OPoints to be independent from actual points in the point-cloud.

This is where TFP-cloud finds it role. This is a point-cloud generated in accordance to the input point-cloud which includes points with fixed structure to be used as candidate OPoints in feature extract. Using these points as OPoints, we have fixed view points to sample the context for. To have the same structure in train and test data, we use the same OPoints in feature extract for train set as well. The only difference in feature extract for train and test

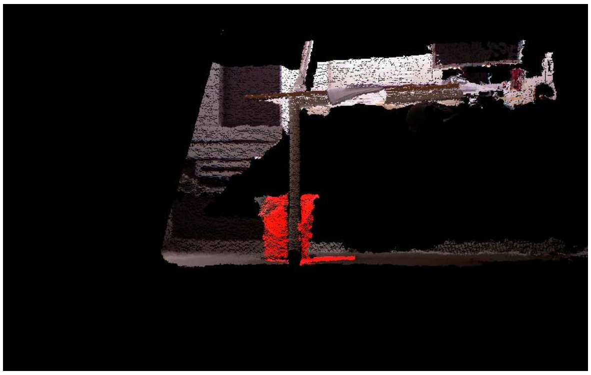

Figure 3.6: An example of annotation result on the point-cloud, red points assumed to belong to the object(Best viewed in color).

is that in train it is checked if the OPoint is on the object, using labels from annotated cloud. Then if the answer is positive the extracted sample would be a positive one, if not it would be a negative sample. But in feature extract for test we just don not do this check.

QPoints would be selected in a loop on key points in the point-cloud. The key point selection is done using VoxelGrid with a radius that would be dependent to the size of the object of interest. This way we have a normally distributed key points in the input point-cloud which will make the sampling more informative.

For each of these QPoints a blob around them is considered from the input cloud with point normals(Query blob). In another loop, the OPoints are assigned as the view point in feature extraction of each sample. Feature vector would be extracted for this blob and the OPoint. Figure 3.8 shows these points and their related parameters. The values mentioned in the figure is for an example object class.

Learning

After Feature extraction, the resulting data set for train is made in a way that it includes samples from several point-clouds for the same object class. In first experiments we used a single Gaussian kernel for the whole feature vector whose fields are of two different types. As mentioned before part one is two float numbers and part two is a histogram. Later we separated these two parts to apply independent kernels on each.

Another important issue here is the distribution of samples in each of positive and negative classes. From the data we extracted, in average, about one percent of the samples were positive and the rest were negative. This is because

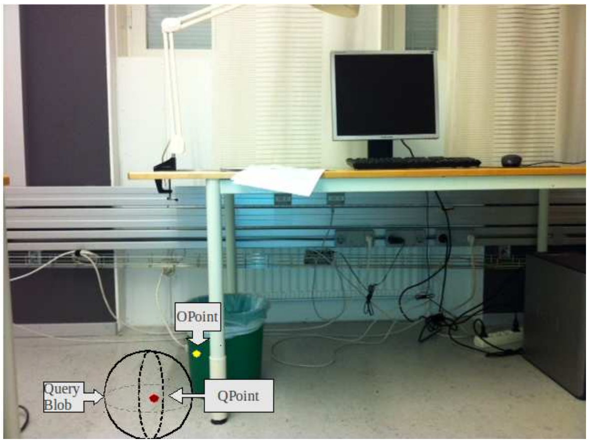

Figure 3.7: Structure used in Feature extraction; The red point is the QPoint; The sphere surrounding it, is the Query Blob; The yellow point is the OPoint (Best viewed in color).

the annotated object is a small part of the point-cloud and positive samples are the samples extracted from its surrounding. This significant difference in number of samples in positive and negative classes makes the classifier to prefer classifying test samples into the class with more train data.

In addition, this issue has some other effects that make the classier confused. The features extracted from blobs in object's neighborhood may be so similar to some feature extracted from a blob far from the object, while the first one would be a positive sample in train data and the second one would be a negative sample. For instance, features extracted for a cup context, which can be a table's surface, will get positive label when it is extracted from a blob close to where the annotated cup is located. But features from the same surface which is extracted from a region far from the annotated object gets a negative label. Because there was no annotation nearby to make it positive.

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 cost in SVM binary classifier. It should have been in a way that not only compensates the lower amount of positive samples, but also emphasize the importance of positive samples compared to negative ones. It can be inferred that there should be a big weight for positive class and rather small value for negative class. In section 3.3 a parameter selection procedure used to find suitable values for them is discussed.

The way datasets and experimental environment is created is discussed in section 3.3.

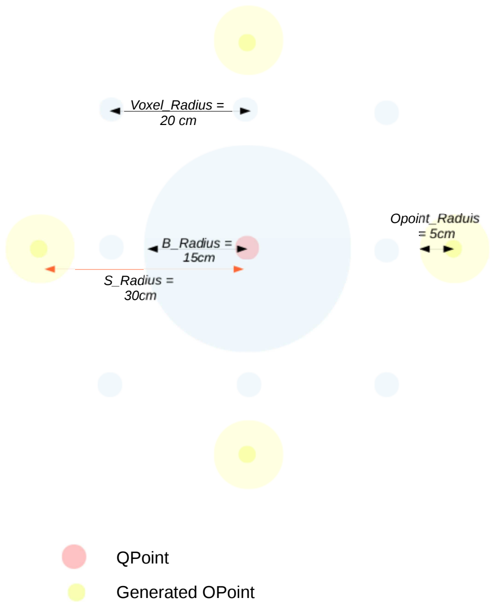

Figure 3.8: QPoint and generated OPoints and their spacial relations.

3.2.3 Parameters

Feature extract parameters: As mentioned before and depicted in figure 3.8 we have some parameters that play great role in achieving good results. These parameters are:

- B-Radius : Radius of the query blob (from which features are extracted) of points from the point-cloud with original density.
- 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 actual Object point.

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.
Require: S_Radius
Require: OPoint_Radius
 1: Key-point selection.
 2: for all Key-points do
 3:    Extract Query blob.
 4:    Extract indexes of generated points within S_Radius from the key-point.

 5:    for all extracted generated points do
 6:       Assign generated point to OPoint.
 7:       Extract features for the OPoint and the Query blob
 8:       if Features are for train then
 9:          if There is an object point within OPoint_Radius of the OPoint
             then
10:             Label the sample as positive
11:          else
12:             Label the sample as negative
13:          end if
14:       end if
15:    end for
16:    Store the sample
17: end for
Ensure: Pointcloud_Extracted samples.
```

These parameters directly or indirectly are dependent to the average size of the object class that we are making the context model for. This dependency is not so tight, it means that the object size does not need to be very accurate. As long as these values are in a way that does not cause the object to be missed because for example the down sampling radius was too big, or such that the complexity and computation time get too long , it would be acceptable. As a result we reduced the number of dependencies to a single parameter which a rough estimate of the object size.

Learning parameters:

- w_i :weight for class labeled (i)
- c
- g

- kernel type
- weight for kernels in multi kernel setup

Among these parameters 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 will set 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.

3.3 Experimental Setup

In order to do an evaluation on our method and the model we proposed, we carried out some experiments. To prepare a dataset for our experiments we needed to capture several point-clouds from different scenes and places which include different object classes.

To capture point-clouds we used different tools: RGBDSLAM [6] is an open source package available in ROS. Using this package and kinect sensor with OpenNi driver a 3D model of a scene can be captured. The results can be saved as a pcd file and its point type would be PCL PointXYZRGB. Point-clouds were captured from different types of places from KTH campus like offices, Kitchens and bathrooms including different types of object that can be found in those places.

There is also a project in CVAP at KTH called KINECT@HOME [1] which is a web based application uses kinect out puts to build 3D mesh model of objects and places. A very good property of this system is that people from any part of the world capture their own video with kinect from different places and scenes and post the videos to a server where the 3D reconstruction happens. Through this means a good dataset can be prepared to train and test models. Of course not all of the models from them where useful for our purpose, but there were a lot of models that we chose among and converted them to the point-cloud type that was compatible with our system.

All resulting point-clouds were saved and based on the content a name was assigned to them. They were divided into two subsets of train and validation. Although, samples from the same point-cloud could be divided into train and validation sets, but we preferred to make them separate even from point-cloud level to be sure of having reliable results.

After acquiring the point-clouds, some python scripts were used that picks the point-cloud from the input dataset and load them into PreProcess module and saves the results of each point-cloud separately in a useful way into the automatically generated environment. Then the annotation tool were used to annotate objects of interest in the point-clouds. All point-clouds were annotated with present objects of interest, regardless of them being in the train set or validation to make evaluation possible on ones used for test as well. Although in validation we would not be using labels from annotation until the

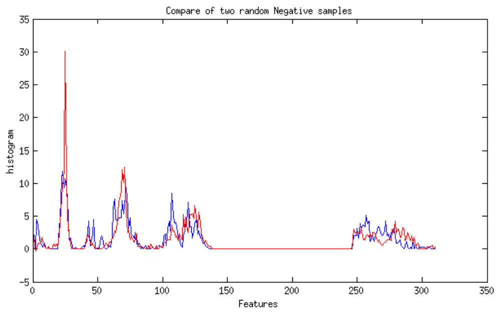

Figure 3.9: Two random Negative samples plotted on top of each other to compare.

end of classification, this information is needed for evaluation which would be described later in next section 3.4.

In next step another script picks annotated clouds and their normals (estimated in PreProcess) into Feature extract and resulting feature vectors are saved in a folder for the corresponding object class under train or validation set, regarding which set their source point-cloud belong to. Then, a subset is randomly selected from the train samples in a way that includes all the positive samples and nine times of its number from negative samples to make the train set. Therefore, the train set includes 10 percent positive and 90 percent negative samples.

At this points , all data sets will be scaled with the same parameters, then the scaled train set is given to SVM train to make the models for each of the object class's context. During training, different values for the parameters are considered, and each resulting model is applied on samples from validation set. Classification results with different examined values of these parameters are compared and best models are selected. The results and evaluation procedure are discussed in section 3.4.

3.4 Results

In this section we will take a look at some results and Analise and evaluate them to see how good our method performs. It can help if we first check some plots showing how does the feature vector look like and how discriminative it can be. Figures 3.9 and 3.10 show the plot of two random negative and two random positive samples.

In order to have a more clear idea about differences between samples we can also look t the distances between random samples in the same class and compare then to the distance of samples from different class. Figure 3.11 shows

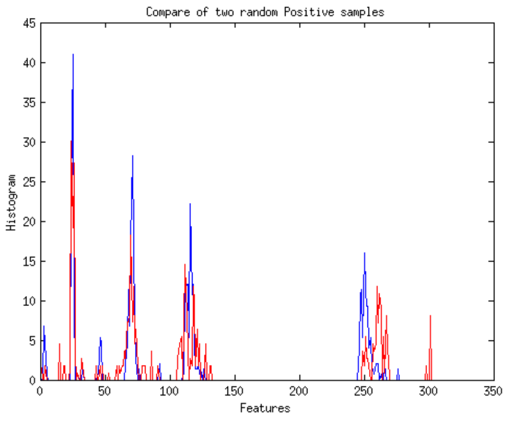

Figure 3.10: Two random Positive samples plotted on top of each other to compare.

the distance in two random negative samples. It should be noticed that the first part of the plot reflects the view point component of the feature vector.

And figure 3.12 is the same plot for positive samples. Features extracted for each class can have big differences as well, which was predictable. Considering different surfaces captured as context by these features that can belong to a class causes the differences. A positive sample can be representing, 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 important that more attention to be put on learning the positive samples rather than negative ones.

Figure 3.13 is showing the distance between random positive and negative samples.

3.4.1 Evaluation

Here we define a metric and an evaluation method that we considered in this work. Metrics such as accuracy of prediction is not something that can show the performance of our system. Accuracy is measuring number of samples that are correctly classified with respect to the total number of samples. Here we do not have direct access to the number of correct classification.

Based on the problem definition, we are looking for possible context of the object, which is locations that an object is expected to be found not just location that an object is present. Intuitively, set of points that belong to an actual object (if there is an actual object in the scene) is a subset of all possible object points in that scene. Therefore, actual object points (`Lp`) are

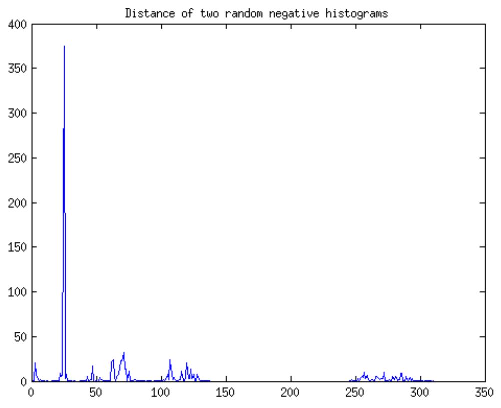

Figure 3.11: Distance between random negative samples.

a subset of predicted positive points (Pp) if the model and classifier perform well (equation 3.2).

$$\mathsf{Lp} \subset \mathsf{Pp} \tag{3.2}$$

In other words, from a good result we expect any random point sampling which is an actual object point to be within predicted positive points. 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. As a result, 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 3.3 shows the evaluation metric, where $\mathsf{E}(\tau)$ is the the value of the metric with respect to a probability threshold τ. This threshold sets the boundary for the lowest probability value that the prediction should have, so that the sample can be considered as positive. This threshold is assigned with a value in a loop that as a result in each iteration a top fraction of the samples are being picked and analyzed. For instance we will pick top five percent of the samples and measure how much would be the probability of having the object in this fraction.

The threshold is computed based on an increasing value of the number of samples to be considered. Pp_τ is Predicted positive sample with respect to the value if the threshold and $n(x)$ is the number of x. Lp is the Labeled positive in the dataset or in other word the annotated object.

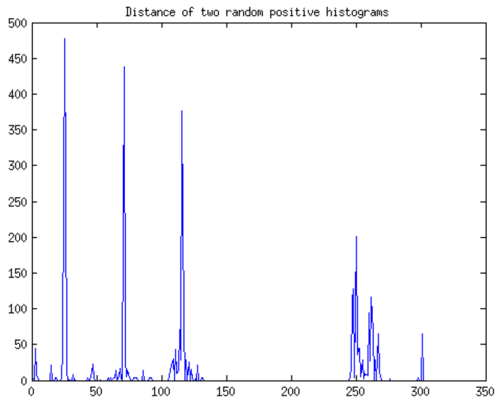

Figure 3.12: Distance between random Positive samples.

$$E(\tau) = \frac{n(Pp_\tau \cap Lp)}{n(Lp)} \tag{3.3}$$

This value is between zero and one, and closer value to one while the threshold is high enough shows good 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 sample is taken. Therefore, the prediction directly shows if a candidate OPoint is a possible object point or not.

Figure 3.14 shows the value of E for top ten percent of the points with respect to their prediction probability in experiments with 80 different configuration of weights and gamma. As it can be seen the results look pretty good.

Considering best models, it can be inferred from the curves that for instance, top five percent of the points contains twenty percent of the object which is a very good result. In other words, reducing the the search space to only 5 percent we still have four time chance of finding the object.

Figure 3.15 shows this evaluation for seven experiments with different weights for class -1. The value is depicted with respect to the fraction of points considered.

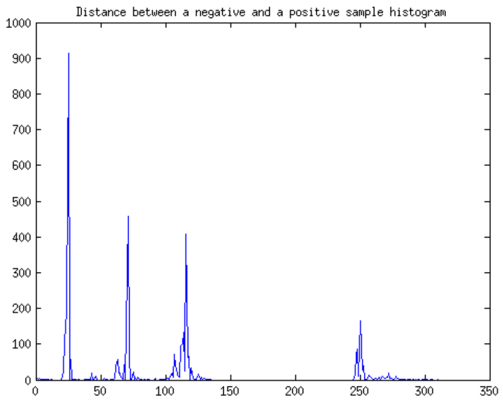

Figure 3.13: Distance between random positive and negative samples.

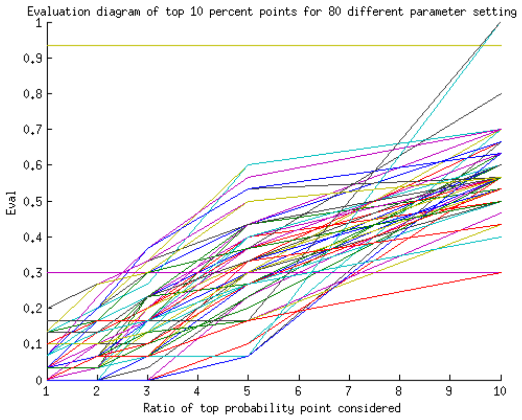

Figure 3.14: Top 10 percent of positive predictions in experiments with 80 different configurations for parameters.

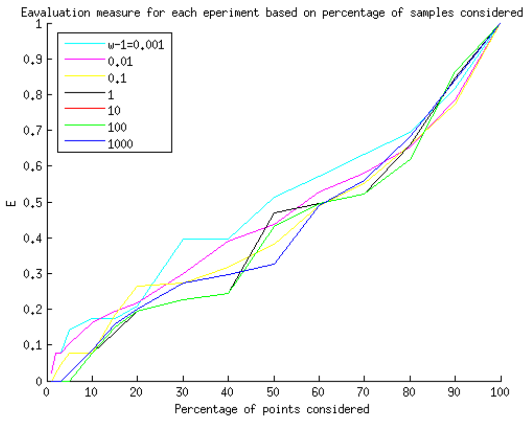

Figure 3.15: Evaluation diagram(1)

Chapter 4
Using Object Context for Place Classification

In this Chapter, we discuss the proposed idea of using object context in classification of places. In this discussion by place classification we 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 robot localization and object detection. Obviously, 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 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. In other words, 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 features categorized 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 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 commonsense knowledge database. As it is considered a future work for this work, 3D context models replace object models in this system and with some improvements a more robust and complete representation for places would be achieved.

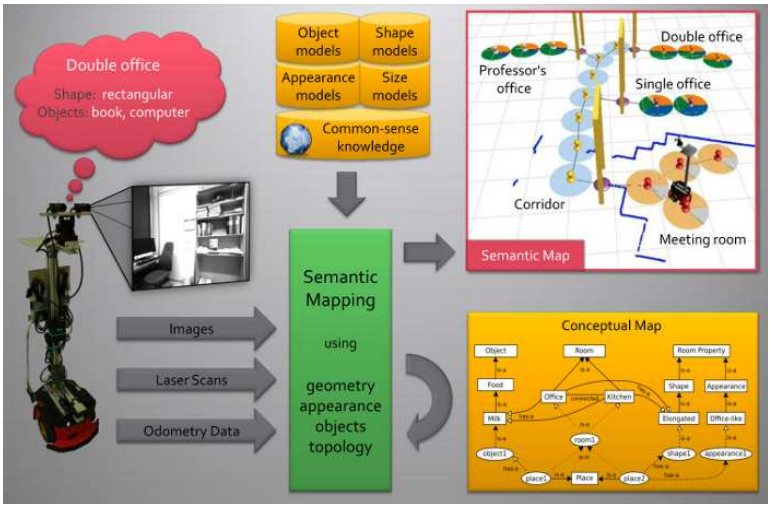

Figure 4.1: Overview of a semantic mapping system by Dr. Pronobis in [14](best viewed in color).

4.1 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 were 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 1 rooms are usually classified into categories like Office 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 [26] 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.

4.2 Ground Truth

As mentioned before, a ground truth or common-sense knowledge database 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. 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.

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

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.

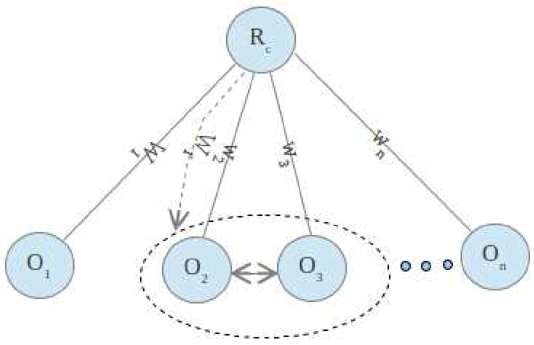

Figure 4.2: This figure shows dependencies between a room category and key object classes, dashed lines shows dependency to a combination of object categories.

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

Dependency of the room category to a combination of object categories is shown in equation 4.2. A probabilistic graphical 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.

4.3 Benefits of Object Context for Place Classification

Some of the benefits of employing context models in place classification can be briefly mentioned as followings.

- It provides a simple way to include human annotations into place representation.
- It is a way of including spatial relations into place representation.
- It is a way to make place models more universal.
- It brings in object information while no fine grained object detection is needed.

4.4 Place classification

A 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 an 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 (4.2) 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.

Chapter 5
Conclusion and future work

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 21.)

[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 page 3.)

[3] Nima Behzad. Point cloud annotation tool. `git@github.com:NimaB/PointCloud--Annotation.git`, 2012. (Cited on page 15.)

[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 30.)

[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 12.)

[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 21.)

[7] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. `http://eigen.tuxfamily.org`, 2010. (Cited on page 12.)

[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 30.)

[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 30.)

[10] R.B Rusu Blodow Nico Beetz Michael. Pointclouds.org ,estimating fpfh. `http://www.pointclouds.org/documentation/tutorials/fpfh_estimation.phpfpfh` (Cited on page 11.)

[11] MIT. Open mind common sense. `http://openmind.media.mit.edu/`. (Cited on page 31.)

[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 30.)

[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 page 3.)

[14] Andrzej Pronobis. Semantic Mapping with Mobile Robots. PhD thesis, KTH Royal Institute of Technology, Stockholm, Sweden, jun 2011. (Cited on pages 29, 30, and 31.)

[15] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes, 2009. (Cited on page 30.)

[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 31.)

[17] "Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu". "estimating vfh signatures for a set of points". (Cited on pages 10, 11, 12, and 13.)

[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 9 and 12.)

[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 pages 3 and 10.)

[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 6.)

[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 and 30.)

[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 and 30.)

[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 31.)

[25] Shrihari Vasudevan and Roland Siegwart. Bayesian space conceptualization and place classification for semantic maps in mobile robotics. Robotics and Autonomous Systems, 56(6):522 – 537, 2008. <ce:title>From Sensors to Human Spatial Concepts</ce:title>. (Cited on page 3.)

[26] 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 6 and 31.)