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

Abstract.

Acknowledgements

Acknowledgements.

Contents

List of Figures

List of Tables

List of Algorithms

ï»¿

Chapter 1
Introduction

When ordinary people talk about Robots, first thing that comes to their minds usually 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. If we have similar purpose in designing a robotic system, most probably the system is considered to imitate human behavior in some assigned tasks or 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 our system.

But trying to make a system to imitate human behavior to perform a task, is not always the best solution. As our system usually has different qualifications and also limits compared to a human body. Moreover , it is not always easy to understand how a human really performs a task, and how the brain make decisions. But it can always give a good idea to start with.

Returning to our Find-and-fetch task; When such a task is given to a human, The first step for him would be imagining what does the asked object look like and where and in what situation it is supposed to be located. If the object is known to him, he goes to the fist location that is most probable place for finding it and then looks for visually similar object to the picture he has in mind of the asked object in that location. If he does not have any information about the object, first he needs to get some from a description or a picture of it. Then he gets some information about possible locations as well. It would be the worst case that he starts just by looking everywhere for something similar to the seen picture or heard description. Depending on amount of information he has ,the amount of time he will need to find the object is different.

There are similar points in a robotic system as well. As we can see, 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 a lot but also can increase the chance of success and correctness of the results.

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.

Torralba et al. in [13] point to the fact that there are Many experiments showing that the human visual system uses contextual information as a very helpful knowledge to facilitate search for objects. Although, it is also mentioned that how this information is applied in the process is not completely known to scientists yet.

To clarify what is considered as object context in this work, it should be mentioned that surfaces that surround an object as a single body is a context for the object. It includes surfaces the the object is located on or beside them and also other objects which are usually beside the object of interest. For instance, if the object of interest is a Kitchen Water tap it's context is the Kitchen sink and the wall behind it. (Figure 1.1)

1.1 Contributions

In this work, we proposed a method to build 3D models of object context which not only can be used as a prior to an object detector, but also is a very useful knowledge in building a 3D representation for places. Benefiting from this model, the area needed to be searched in an object detection task is reduced to parts of the scene with high probability of being the context for the object of interest. It also has 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 results 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. 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.
- 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 in which there is a focus on using contextual information for Object detection/recognition or place classification are briefly reviewed. Also some pieces of work which are benefiting 3D features and descriptors for similar purposes are mentioned.

Object Context

In [12] Torralba et al. present a low dimensional global image representation to achieve useful information to recognize places and then they show that how contextual information can provide priors for object recognition. As mentioned this work is done on images using 2D features.

In [14] 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 [7] R.Perko and A.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.

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

In [10] Rusu et al. proposed View point 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.

Place classification

In [15], S.Vasudevan and Roland Siegwart proposed a representation for space based on objects. Several algorithms are discussed by them 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 a little more about motivation, applications and challenges would be presented.
In chapter3: Our method and details about implementation, experiments and evaluation are shown.
In chapter4: The 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.
In chapter5: We have a review and next steps that could be considered for this work and possible improvements and future works are mentioned.

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 [16], also pointed to the fact that object detection is a good prior for place categorization. T. Southey and J. J. Little in [11] 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 Scenario

Guess that your robot wants to look around in a building and make a map including semantic information about the met places in the map, So the robot capture a 3D map of the building using a SLAM application. Using the model proposed in this thesis the robot can be provided with corresponding which are representing possible contexts of different object classes that based on a ground truth (defined in advance) can be considered as key objects for each place categories.

For the point cloud captured of each place, these models can be applied, based on the matches found with high probability on the point-cloud the robot can compute possibility of finding each of the key object classes in the place, then based on the object classes found more probable to be in that place a probability could be computed for the class of that place. combining this information with other applicable descriptors can give a good decision measure for place categorization.

But for the building of the model it self. the same steps mentioned above for capturing the point cloud would be performed , on the pre processed captured point cloud of a place using the annotation tool we have developed, key objects for that specific place category would be annotated. Feature extraction would be run on this annotated point cloud giving training samples. Train samples would be combined from several different point-clouds for the same place category but different instances, Resulting data set would be divided into train, validation and test sets. Using SVM with two kernels the model would be trained.

More details would be presented in section 3.2

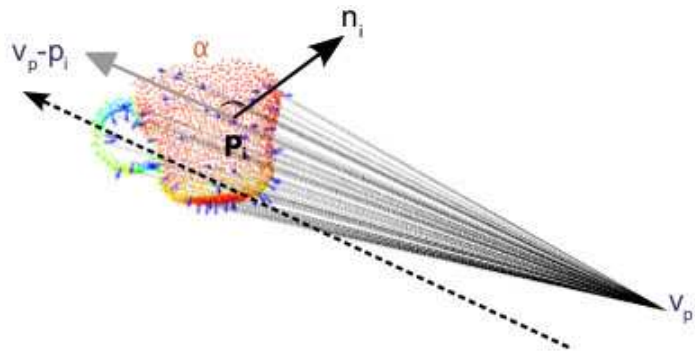

Figure 3.1: Viewpoint part of the feature vector.[8]

3.2 Implementation

The data type we used in this work is PCL point-cloud [9], in this data type 3D coordinates and color information of a point is saved in a record and then an array of these records represent the whole captured scene.

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. second field is the orientation of the blob with respect to vector of gravity. and rest of the fields are in second part of the feature vector with is a histogram. This histogram is called VFH which captures the geometry of the blob with respect to a specific view point. [10]

3.2.1 The Viewpoint Feature Histogram (VFH)

As it is described in PCL official website [8] and [10], is consisted of:

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

The viewpoint component is computed by collecting a histogram of the angles that the viewpoint direction makes with each normal. Note, it is not the view angle to each normal as this would not be scale invariant, but instead it is the angle between the central viewpoint direction translated to each normal.

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

normals on the surface.(extended FPFH) So it consists of FPFH values for points in the query point-cloud.

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 and 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 pair of points in the query point's neighborhood ($O(nk^2)$). The elements consist of three angles and a distance.

In FPFH [6], instead of encoding this information for all pair of points in a neighborhood it will only take pairs between the query point and it's neighbors ($O(nk)$), it will be called simplified PFH or SPFH. Then to keep the amount of information this feature can provide it will also estimates SPFH for all k points in the neighborhood and integrate the resulting values with a normalization to get the FPFH for the query point(equation 3.1).

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

Figure 3.2 shows the encoded elements and their relation in FPFH.

Then in VFH the resulting FPFH of points in the point cloud are integrated (extended FPFH),(4*45 bin) and adding the view point component (128 bin) produces a 308 bin histogram. Unlike FPFH, as a result of adding the viewpoint component VFH is Dependant to viewpoint but it still preserve the property of being scale invariant. Figure 3.3 shows a sample plot of VFH and it's components.

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

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 [9]
- Eigen [5]
- LIBSVM [3]

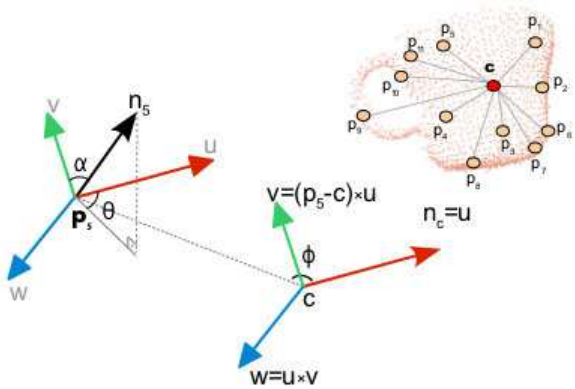

Figure 3.2: Three angles and a distance encoded in FPFH and the points considered.[8]

3.2.2 Modules

There are three modules in the system as :

- PreProcess
- 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. 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 process here a conversion of the transformed point-cloud into that type is also carried out.

This new type is PointXYZRGBL which include 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. And the most important process which is performed here is estimation of surface normals which is essential for feature extraction. The accuracy of the VFH feature is directly Dependant to the accuracy of the normals. At first we were doing this process in feature extract module and for each query blob at the moment which

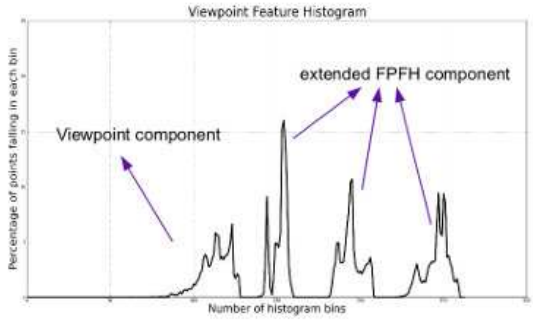

Figure 3.3: A sample plot of VFH and it's components.[8]

added a huge overhead to the process while it could be done independently and once for each point cloud. Here the normals for the whole point cloud is estimated once and saved in a separate pcd file to be used in feature extract. It uses NormalEstimation class in PCL.

There is also one more product for this module which would discussed more in 3.2.2 it is a generated point-cloud based on input cloud's size to be used for feature extract. It is called TFP-cloud.

So the results of this module is:

- Point-cloud's normals
- 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. In figure 3.4 the annotated object could be seen in red.

FeatureExtract

First definition of some terms used in this part is as follows:

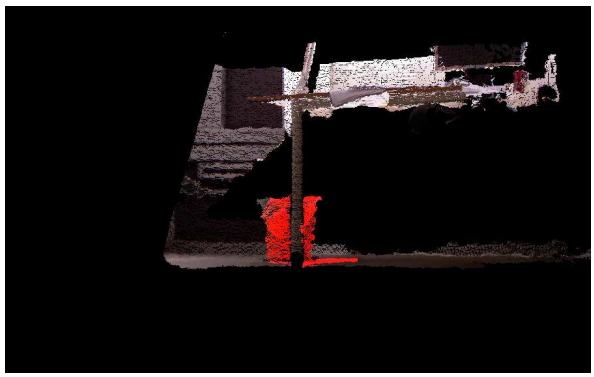

Figure 3.4: An example of annotation result on the point-cloud, red points assumed to belong to the object.

- 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.
- OPoint: The Object point is the point in surrounding of the Qpoint which is an object point in a positive sample.

In this module the converted and transformed point cloud and it's 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 that would be useful in building a model of objects of interest's context. The way we did this is for each key point, which would be considered as a QPoint when it is selected for feature extract, we will take a neighborhood of points and their corresponding point normals to extract features from. So thing that happens here is studding the geometry around the query point as a candidate geometry of a positive context.

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 neighborhood is considered as view points for VFH (OPoints), so for each pair of $\mathtt{Qpoint}$ and $\mathtt{Opint}_i$ one sample is extracted,this way the feature extracted for the same query point and from the same query blob but different viewpoint would be slightly different. This will result in discrimination between same context viewed from different view point which would be to our benefit.

As mentioned above these view points are candidates for being object points, so in each sample if the extracted feature is for a pair of $(\mathtt{QPoint}, \mathtt{OPint})$

in which the OPoint is actually an object point we want this sample to be labeled as positive and if not it would be negative sample. This means that if the feature we extracted from a query blob for a view point which is in a specific distance from the QPoint and is located on the object of interest, the query blob is actually a context for that object. 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 find candidates for context in a test point-cloud where possibly object is not present at the moment but the context is, so 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 for each QPoints we have fixed view points to sample the context from. 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 is in train it will be checked if the OPoint is on an object using labels from annotated cloud, then if the answer is yes 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 will 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 would be selected from the input-cloud with point normals(Query blob), number of OPoints would be considered in a loop as the view point, and feature vector would be extracted for this blob and the OPoint. Figure 3.5 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. For first tries 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 issue which was very important is the distribution of samples in each of positive and negative classes. From the data we extracted in average about 1 percent of the samples were positive and the rest were negative , which is obvious, as usually the annotated object is a small part of the point-

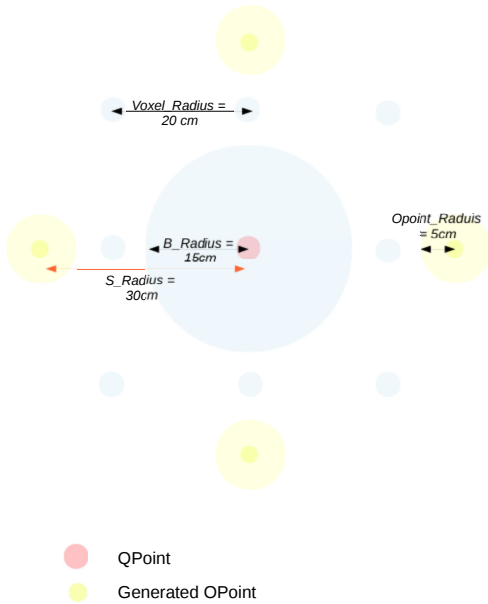

Figure 3.5: QPoint and generated OPoints and their spacial relations.

cloud and positive samples are the samples extracted from it's surrounding. This difference in number of samples in positive and negative class makes the classifier to prefer classifying test samples into the class with more train data.

In addition, the same issue has some other effects as well, 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 extracted from other side of the same table would get a negative label , because there was no annotation nearby to make it positive.

In order to solve the mentioned issue and both of it's 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.

datasets was created in way that will be discussed in section 3.3

3.2.3 Parameters

Feature extract parameters: As mentioned before and depicted in figure 3.5 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 would depend on the object class (Voxel-Radius+ a margin)
- OPoint-Radius : This is the radius of the neighborhood of the generated point to look for actual Object point.

These parameters directly or indirectly are dependent to the average size of the object class we are making the context model for. This dependency is very 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 kernel, 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 get data 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 [4] 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 result can be saved as a pcd file and it's 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 happen. So this could easily made a good test data for us. 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 where divided into two subsets of train and validation. Therefore, although samples from the same point-cloud could be divided into train and validation sets as well, but we preferred to make them separate even from point-cloud level to be sure of having reliable results. Each of these train and validation sets included different category of places.

After acquiring the point-clouds we have written some python scripts 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 used to annotate objects of interest in the point-clouds. All point-clouds was annotated if possible, regardless of them being in the train set or validation and thats why the data set is not called a test set. Although in validation we would not be using labels from annotation till the end of classification, but for evaluation which would be described later in next section 3.4 we will need this information.

In next step another script will pick annotated clouds and their normals (estimated in PreProcess) into Feature extract and based on from which of train and validation set the cloud came from feature vectors are saved in a set with respect to the object class. Then from the train set a subset would be selected randomly in a way that includes all the positive samples and nine

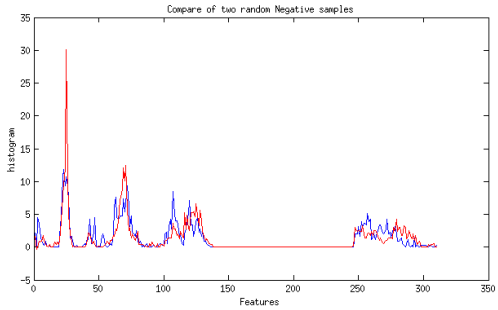

Figure 3.6: Two random Negative samples plotted on top of each other to compare.

times of it's number from negative samples to make the train set. Therefore, the train set will include 10 percent positive and 90 percent negative samples.

At this points , all data sets will be scaled with the same parameters and then scaled train set is given to SVM train to make the models for each of the object class's context. In time of train different values for the parameters would be considered , and each resulting model would be applied on the validation set. In evaluation procedure results with different values of these parameters are compared and best models are selected. The results and evaluation procedure would be 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 does our method perform. It can help if we first check some plots showing how does the feature vector look like and how discriminative it can be. Figures 3.6 and 3.7 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 it to the distance of samples from different class. Figure 3.8 shows 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 3.9 is the same plot for positive samples. Features extracted for each class can have big differences as well, which was predictable. Considering different situations captured by features that can belong to a class shows this fact. A positive sample can be representing the surface of a table or a wall, or the region that these two surfaces meet for example. So this comparison can give us a feeling of how accurate should the classifier be, and as mentioned

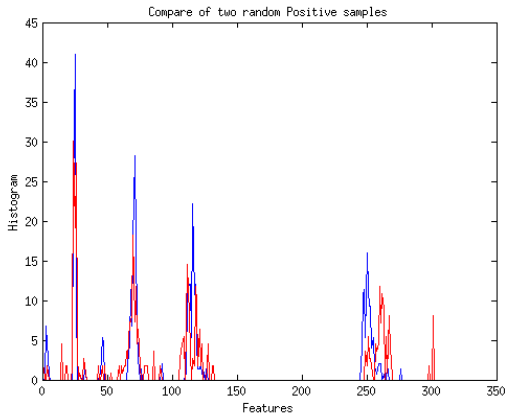

Figure 3.7: Two random Positive samples plotted on top of each other to compare.

before how much it should put the attention on learning the positive samples rather than negative ones.

Figure 3.10 is showing the distance between random positive and negative samples.

3.4.1 Evaluation

As mentioned before there could be negative samples in train set which are very similar to positive samples which are labeled as negative just because they were not captured from a neighborhood of the annotated object, but we would like these samples to be predicted as positive.

So we should define what result we expect as a good result, as usual evaluation metrics like accuracy is not something that can show the performance of our system. Based on the problem definition we had, we are looking for possible context of the object, which is locations that an object could be expected to be found not just location that an object is present.

Therefore, if the predictions give us point of the point-cloud where there is a high probability of finding the object , it would be what we want. The point here is , that the location of the object would be a subset of this predicted set. We defined the good result as predicted set of locations that reduces our search space for the object as much as possible while 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 τ. This threshold sets the boundary for the lowest value that the resulting probability from prediction

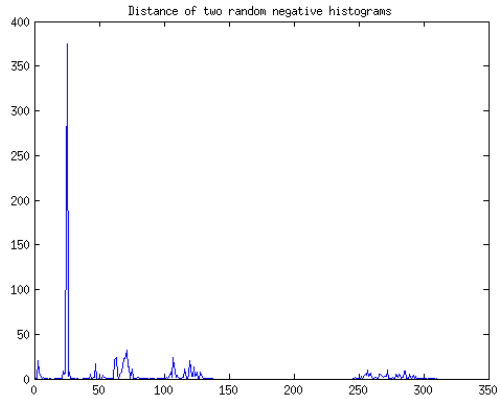

Figure 3.8: Distance between random negative samples.

should have so that the sample could be considered as positive. This threshold would be assigned with a value in a loop so that in each iteration a top fraction of the samples could be picked and analyzed. For instance we will pick top 5 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 $\mathfrak{n}(\mathfrak{x})$ is the number of $\mathfrak{x}$. Lp is the Labeled positive in the dataset or in other word the annotated object.

$$\mathsf{E}(\tau) = \frac{\mathfrak{n}(\mathsf{Pp}_\tau \cap \mathsf{Lp})}{\mathfrak{n}(\mathsf{Lp})} \tag{3.2}$$

This value would be 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 considering from which point in the point cloud the sample is taken the result would be assigned to that point. So the prediction will directly show if the query point is a possible context or not.

Figure 3.11 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 by picking top five percent of the points twenty percent of the object which

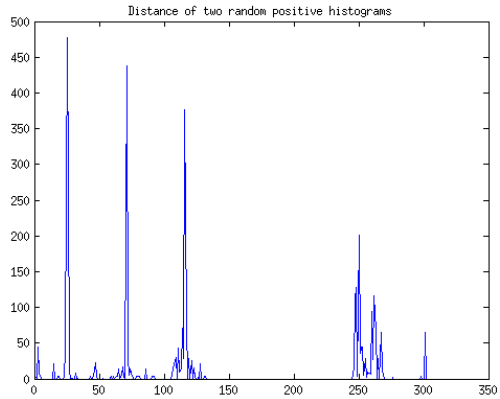

Figure 3.9: Distance between random Positive samples.

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

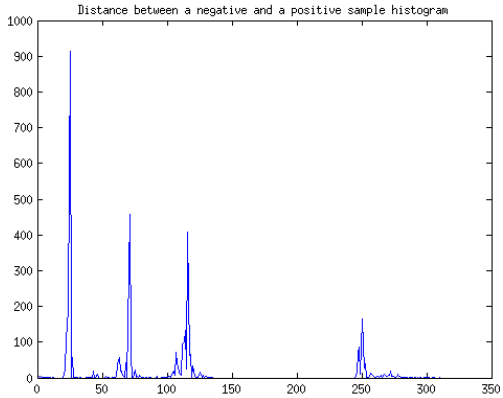

Figure 3.10: Distance between random positive and negative samples.

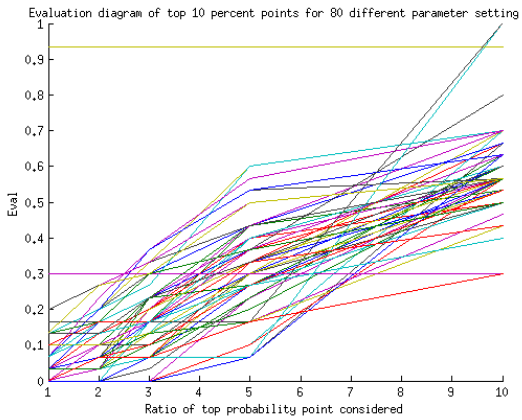

Figure 3.11: Top 10 percent of positive predictions in experiments with 80 different configurations for parameters.

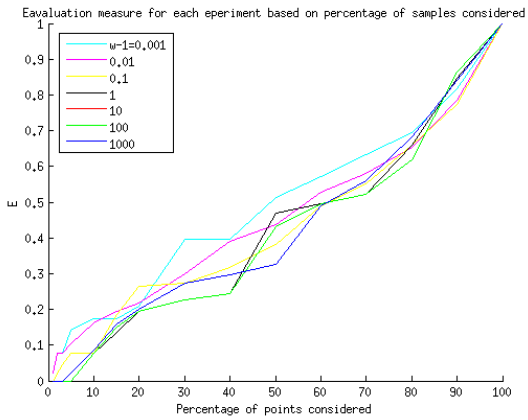

Figure 3.12: Evaluation diagram(1)

Chapter 4
Using Object Context for Place Classification

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 20.)

[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 5.)

[3] 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 13.)

[4] "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 20.)

[5] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. (Cited on page 13.)

[6] R.B Rusu Blodow Nico Beetz Michael. Pointclouds.org ,estimating fpfh. `http://www.pointclouds.org/documentation/tutorials/fpfh_estimation.php#fpf` (Cited on page 13.)

[7] 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 5.)

[8] "Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu". "estimating vfh signatures for a set of points". (Cited on pages 12, 14, and 15.)

[9] 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 12 and 13.)

[10] 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 5 and 12.)

[11] 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 8.)

[12] 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 page 5.)

[13] 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 4.)

[14] Antonio Torralba. Contextual priming for object detection. International Journal of Computer Vision, 53:169–191, 2003. (Cited on page 5.)

[15] 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 5.)

[16] 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 page 8.)