Warsaw University of Technology

Faculty of Mechatronics

Institute of Automation and Robotics

Adam Kosiorek

3D Object Classification based on RGBD images

bachelor thesis written under supervision of

prof. dr. hab. Barbara Siemiątkowska

Warsaw 2014

Degree: Automatic Control Engineering and Robotics

Major: Robotics

Birth date: 28 February 1991

Study starting date: 1 October 2010 r.

Biography

I was born on 28 February 1991 in Olsztyn, Poland. I attended 5$^{\text{th}}$ Secondary School by the name of "The Common Europe" in Olsztyn. I started a Bachelor of Engineering degree on the Faculty of Mechatronics of Warsaw University of Technology on 1 October 2010. I did an internship in Faurecia R&D S.A. in 2012. In 2013 I started working in IBM Poland Sp. z o.o. where I stayed for 6 months. Then I started working full-time in Samsung R&D Centre Poland.

.....

signature

Streszczenie

W pracy przedstawiono podejście Bag of Words do klasyfikacji obiektów na podstawie trójwymiarowych chmur punktów. Wykorzystano metodę uczenia nadzorowanego, dzięki czemu można rozpoznawać dowolne kategorie obiektów. Opisano metodologię BoW, jej znaczenie w wizjii komputerowej, a także algorytmy znajdujące zastosowanie w poszczególnych stadiach procesu. Następnie przedstawiono problem klasyfikacji. Określono wymaganą funkcjonalność, zaprojektowano konfigurowalną i łatwo rozszerzalną aplikację oraz zaimplementowano ją w C++. Zbadano algorytmy pod kątem przydatności do BoW oraz sprawdzono skuteczność aplikacji na dwóch powszechnie uznawanych zbiorach danych.

Abstract

This paper introduces a Bag of Words semantic object classification framework based on three dimensional point clouds. The framework uses machine learning algorithms and a learning by example approach, thus an arbitrary set of categories can be used. Basic concepts of BoW and its role in computer vision are introduced. Then, the most popular algorithms for different steps of BoW are described and a problem of classification is explained. After a requirement analysis was performed a configurable and extensible application was designed and implemented in C++. Finally, algorithms for different steps of BoW classification are evaluated and results on two benchmark datasets are discussed.

Contents

1 Introduction **7**

Introduction **7**

1.1 Bow of Words 8

1.1.1 Processing Pipeline 9

1.1.2 Applications in Computer Vision 12

1.2 Problem of Classification 13

1.3 Libraries 15

1.3.1 PointCloud Library 15

1.3.2 OpenCV 16

1.3.3 Boost 16

1.4 Datasets 16

1.4.1 Berkeley 3D Object Dataset 17

1.4.2 The University of Tokyo Dataset 18

2 Design **19**

Design **19**

2.1 Software Functionality and Architecture 19

2.1.1 Required functionality 19

2.1.2 Architecture 21

3 Experiments and Results **24**

Experiments and results **24**

3.1 Experimental Setup 24

3.2 Experiments 24

3.2.1 Detectors 25
3.2.2 Descriptors 25
3.2.3 Codebook 25
3.2.4 B3DO 26
3.2.5 Tokyo 27
3.3 Conclusion 28

Chapter 1

Introduction

Capturing visual information of any kind have never been easier. Almost everyone of us posses a camera, be it a professional dSLR or a tiny one embedded into one of the ubiquitous smartphones. As human beings we have no difficulty understanding this data, just as we have no difficulty understanding what our eyes see. Unfortunately, the situation of our computers is the very opposite. Storage capacity of modern-day computers is overwhelming. But even if hard-drives are full of images it is impossible to identify their semantic meaning without the aid of specialised algorithms, however. These algorithms often balance on the verge of image processing and machine learning — they are a part of an area called computer vision. As Forsyth and Ponce [15] have it:

> computer vision is (...) an enterprise that uses statistical methods to disentangle data using models constructed with the aid of geometry, physics, and learning theory.

The area of computer vision is not new. The original applications were industrial inspection or mobile robot navigation. More recent approaches consider problems such as human-machine interfaces, medical diagnostics, or image retrieval.

How about semantic classification of scenes or objects? Suppose a mobile robot would know, or understand, what does it have to deal with. Suppose a robot would know that it is about to take, say, a glass of water. How would it affect the robot's behaviour? Could it have any impact on the force of the grip? or the speed and the manner of the planned object movement? Answers for all this questions are yes. Semantic classification is possible, but it requires clever mechanisms of computer vision and powerful computers. It can be seen as a part of a more general classification problem, which machine learning handles

quite well. Suitable algorithms include, but are not limited to, artificial neural networks, deep belief networks, logistic regression, support vector machines and approaches based on probabilistic graphical models. One has to bear in mind that none of them can handle the problem directly. That is, it is inconceivable to feed a whole image as is into a classifier — some degree of preprocessing is required. One such preprocessing technique is translation of image into a feature vector. What should this vector comprise of? A Bag of Words model is one technique that answers this question. This model has been first used for scene classification in 2004 in [6] and maintains its popularity ever since.

In this paper a technique for semantic object classification based on three dimensional point clouds is presented. The rest of this chapter is organised as follows: Section 1 describes the Bag of Words technique in detail. Construction of an image's model is discussed and the most popular algorithms are summarised. Section 2 provides basic information on the problem of classification, introduces basic concepts and methods of classification. Section 3 names C++ computer vision and general purpose libraries used in this project. Finally in Section 4 two datasets of RGBD images are outlined. The following chapters are titled "Design" and "Experiments and Results". The former contains description of required functionality of a programme for BoW object classification, its design and implementation details. The latter addresses topics of experimental setup, performed experiments and a summary of results.

1.1 Bow of Words

The Bag of Words (BoW) model is a an intermediate representation used in natural language processing, information retrieval and data mining [need pub]. The model can be seen as a simplification, for it obliterates any grammatical information and even word order contained within an original text document. What is left is an unstructured group of words, usually in a form of a vector or a histogram of word incidence.

The figure 1.1 shows a piece of text and its BoW model, where articles and punctuation marks were left out. In order to compare different documents a global dictionary must be built. The dictionary (codebook) is constructed by taking every word from all available documents and removing duplicates. One can image that if a dictionary is of any considerable size resulting representation of especially small documents will be very

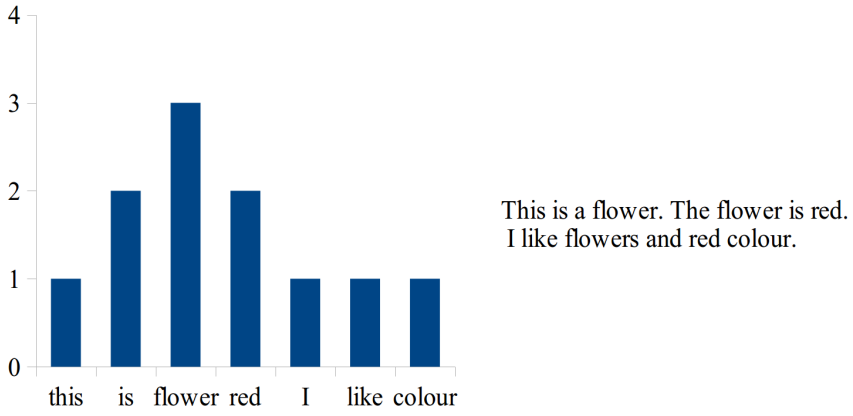

Figure 1.1: Bag of Words histogram

sparse.

In natural language processing BoW representation is used to infer semantic meaning of documents [need pub]. If an image could be translated into a text document it might be possible to employ similar methods. A question arises: How does one make a text document from an image?

1.1.1 Processing Pipeline

Tsai describes a well established pipeline that allows translation of images into text documents or BoW models [23]. It consists of the following steps: (1) region detection, (2) feature extraction, (3) vector quantization and (4) BoW histogram formation as can be seen in the figure 1.2. The author summarises the most commonly used methods for performing these tasks.

Region Detection

Characteristic region detection is the first step in any Bag of Words framework. Numerous detection methods have been developed, but choosing the right one for any particular case might prove tricky. A good overview of various mechanisms is available in [23].

The most common detectors make use of a Harris corner detector or image's first or second derivatives. A Harris-Laplace detector is an example of a Harris-based detector. The Harris function is scale adapted and its outcome is a subject to a Laplacian-of-Gaussian (LoG) operator, which selects relevant points in scale space. Images' regions' 2^{nd} derivatives, namely the regions' Hessians, can be combined with a LoG operator as

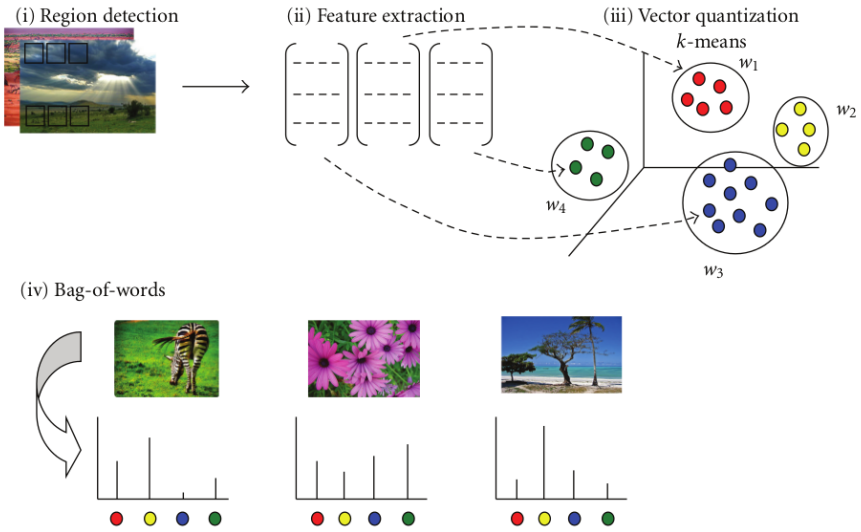

Figure 1.2: Bag of Words pipeline. The figure comes from [23]

well. This combination allows selection of points significant in the two spaces: the scale space and the Hessian's determinant space. The latter entails the speed at which pixel intensities change in the neighbourhood of a point.

A number of more complicated recipes for salient region localisation have been developed and implemented. These include for example SIFT [13], SUSAN [20] and Intrinistic Shape Signatures [25]. The majority of keypoint detection formulas is being developed for the 2D domain. A number of them have been adapted to 3D, however. A comparative evaluation of detection algorithms available in the PointCloud Library (PCL, discussed below) can be find in [8]. Another comprehensive study is [19]. All these formulas, called sparse feature detectors, resort to selection of maxima in specific state spaces. An entirely different scheme is to use a dense feature detector. That is, take a uniformly sampled grid of points. Dense detectors, unlike the sparse ones, take points from slow changing regions such as clear sky and calm ocean water. Li *et al* showed that dense detectors generally outperform the sparse ones [7].

Feature Extraction

Suppose only coordinates of a keypoint were found. In case of even smallest rotation or translation they would change and the keypoint would be lost. Keypoints should be described in a way that makes them invariant to affine transforms, changes in light intensity or colour saturation. All these properties are hard to achieve, but methods that meet some of the criteria exist.

Keypoint description is usually delivered in a form of coordinates in a high-dimensional space. One of the best algorithms is SIFT [14], which is 3D histogram of gradients structured as a 128-dimensional vector of floating point values. It is the most often extracted descriptor in BoW pipelines. Other methods include various colour descriptors, binary descriptors such as 512-dimensional GIST [15]. There are techniques designed for 3D exclusively. Among them are Persistent Point Feature Histogram (PFH) [18], its faster alternative FPFH [16] and PFHRGB, which takes into account colour information, all implemented in PCL.

Vector Quantization

When features are extracted they have to be normalised. Vector quantization step have two main phases. The first one is codebook construction from a training dataset. The second phase is responsible for parsing (or translation) raw image descriptors into a form compatible with the newly constructed codebook. The simplest way of building a dictionary is to find patterns or regions within the training dataset's descriptors. Then, the parsing phase would be nothing but assigning each descriptor to a certain region. Such an assignment can performed by a k-Nearest-Neighbours match. Slightly more advanced techniques of dictionary building involve using many kinds of descriptors, i.e. texture, shape and colour descriptors, or spatial information.

The KMeans is the single most popular vector quantization algorithm used in the BoW pipelines [23]. Developed in 1950's, it is well known and simple. kMeans divides all data into a predefined amount of clusters and computes the clusters' centroids. Many modifications and alternative versions have emerged [10]. Some of them are: faster than the original *approximate kmeans*, *hierarchical kmeans*, which automatically chooses the final number of clusters and a *soft kmeans* — a variation of the algorithm that allows a fuzzy alignment (*i.e.* each point can belong to several clusters with different weights). Vector quantization is responsible for the dictionary size, for the clusters' centroids are (in a simple approach) synonymous to the visual words. Thus the higher the number of clusters, the bigger the dictionary, which in turn allows for a more precise image description.

If computational cost is of no concern, or if required precision is of the utmost priority, a Gaussian Mixture Model (GMM) can be used. The GMM partitions data into a set

of clusters and finds their centroids. Additionally, it computes a probability distribution over each cluster, i.e. for every point its probability of membership in each cluster is returned. Both kMeans and soft-kMeans are specific variants of the GMM. The drawback of the GMM is its massive computational cost in comparison with still expensive kMeans.

It should be underlined that the vector quantization step, especially the codebook construction process, is the most time-consuming part of the whole Bag of Words pipeline.

1.1.2 Applications in Computer Vision

The Bag of Words model is nothing but an intermediate representation. As such it is mainly used in the two following areas: Content Based Image Retrieval (CBIR) and Scene/Object Classification.

Content Based Image Retrieval

CBIR is a computer vision approach to image retrieval from large image databases. Tangelder *et al* provides an overview of techniques applicable to 3D objects [21]. The task of image retrieval is to find a database entry fulfilling certain conditions. If it is to be performed efficiently several criteria have to be met [22], namely: (1) All entries should be indexed in a concise way, (2) a (dis)similarity measure should be provided and (3) an efficient search algorithm should be available.

Indexing of objects is required so as not to compare the database entries explicitly. Therefore, for every object in a dataset a compact as well as a discriminative signature should be computed. Suitable algorithms can be divided into three general categories: (1) feature based methods, (2) graph based methods and (3) other methods.

Feature based methods can be further divided into global features and local features. The global features take form of a single vector (or a point in a d dimensional space). They are usually associated with object's mass or volume or distributions of these. Being easy to compute and straightforward to implement, their discriminative power is rather low — they cannot be used for partial matching, but are well suited for an early preprocessing. Local features, on the other hand, describe object's characteristic regions. Their shape is similar to that of the global features, but instead of a single point in space there are multiple ones — one for each considered region. Tangelder argues that local features based approaches are inefficient and lead to a complex indexing problem [21]. At the

same time Toldo *et al* and Li *et al* show that Bag of Words local feature based approach has no such drawbacks. On the contrary — it is easy to implement, efficient and provides state-of-the-art results.

Scene and Object Classification

Inspired by various works on pattern recognition and texture classification and an idea of a "texton", a building block from which textures are made, Csurka *et al* is the first to apply the Bag of Words paradigm to the task of scene classification [6]. In their early work authors suggested a a general framework described above. After translating images into text documents BoW histograms were built and fed into two classifiers: SVM and Naìve Bayes (discussed below).

Fei-Fei *et al* refined the above approach by examining several keypoint detectors and descriptors. The main contribution of their work is, however, the development of a genuine classification algorithm based on a probabilistic graphical model. Accuracy of 76% on a large 13 category dataset was achieved. Object categorisation can be addressed in the same way. The only difference is that a bounding box of an object (or knowledge of object's coordinates) can be helpful. On the other hand, it is possible to consider an object classification problem where images of singled-out objects are provided — e.g. a single object covers a majority of image's area like in [24].

Classification is not an integral part of the BoW pipeline. But the Bag of Words histograms can be used as a compact and discriminative representation that can be fed into any classifier.

1.2 Problem of Classification

To classify means to, given a set of categories, produce a category label for a given set of features [15]. Many problems might not seem like classification problems and image categorisation is one of them. Images are too complex to be processed by any classifier directly. We can think that there are too many details, too many aspects that decide whether a picture fits into a category or not. But it turns out that many problems can be abstracted in such a way that they become pure classification problems. One such abstraction for photographs or point clouds is the Bag of Words model. Classifiers can

be two-class classifiers (e.g. binary) or multi-class classifiers. It is a common practice to build multi-class classifiers from binary ones, however. The most popular classifiers are: k-Nearest Neighbours, Logistic Regression, Naìve Bayes and Support Vector Machine.

Naìve Bayes

Assume that we have a n-dimensional feature vector $\mathbf{X}$ such that $\mathbf{X} = \{x_1, x_2, ..., x_n\}$ and an unknown category label C. The probability of the category C given the feature vector $\mathbf{X}$ is $P(C|\mathbf{X}) = P(C|\{x_1, x_2, ..., x_n\})$. The strong (or naìve) Bayes assumption says that features are independent of each other, that is $P(\mathbf{X}) = P(x_1) * P(x_2) * ... * P(x_n)$. If so, we can compute probability distributions of every class in our training set given every feature. Then, for any new feature vector the joint probability distribution over a set of features can be calculated and an appropriate class label can be assigned.

k-Nearest Neighbours

The nearest neighbours classifier assumes that if there are labelled points in a neighbourhood of a newly added point, then the new point can be classified based on it's neighbours' labels. There is an issue of choosing the correct number of nearest neighbours (or k) to consider

Support Vector Machine

Assume that a set of pairs $\{\{\mathbf{x}_1, y_1\}, \{\mathbf{x}_2, y_2\}, ...\}$ where x_i are features and $y_i \in \{-1, 1\}$ are labels is given. Further assume that points with different labels are two linearly-separable datasets as shown in the figure 1.3, where dots and circles are points with different labels. Then, there exist parameters $\boldsymbol{w}$ and b which satisfies

$$y_i (\mathbf{w} * \mathbf{x}_i + w) > 0 \tag{1.1}$$

for every $\mathbf{x}_i$ or a data point. This equation specifies constraints on a plane (or a hyperplane in general) that separates the dataset. A Support Vector Machine finds parameters $\boldsymbol{w}$ and b so as to maximise the distance of this plane from both datasets.

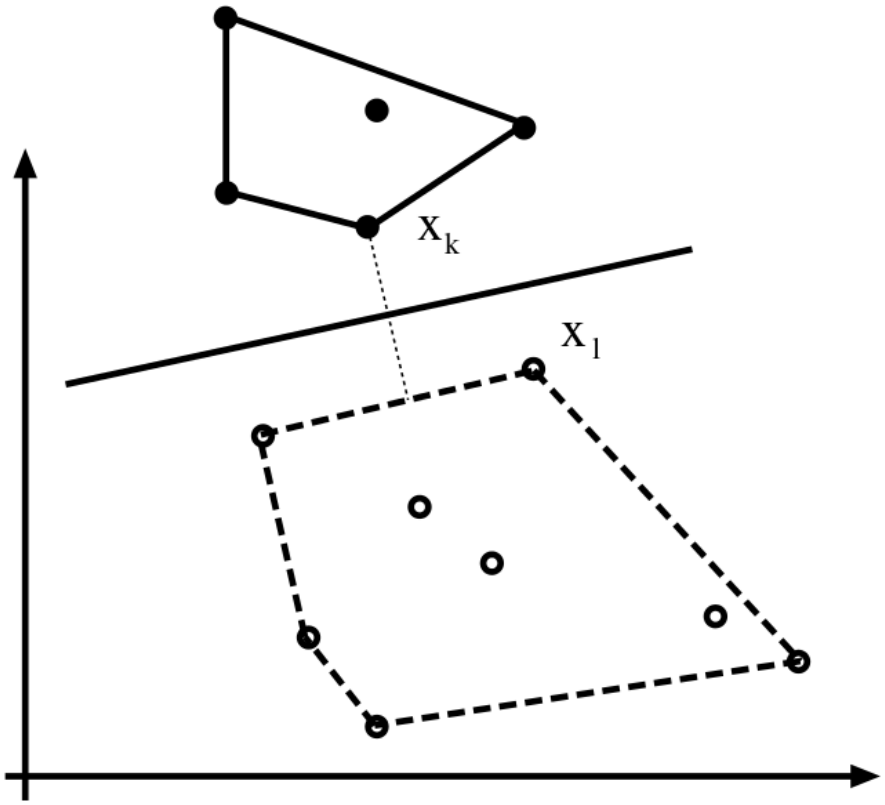

Figure 1.3: Classification by a Support Vector Machine. The distance of the separating plane from both datasets is maximal. The figure was originally published in [15]

1.3 Libraries

One of the advantages of object oriented programming is that it is easier to reuse once created code. There exists a variety of third party libraries for the C++ programming language. One of the biggest and the most widely used are the general purpose Boost libraries [2]. In the fields of computer vision there are, among others, OpenCV [5] and PointCloud Library [17]. Easy to use and intuitive logging is provided by the Apache log4cxx [1].

1.3.1 PointCloud Library

Being nothing but a large scale open source project, it is a great processing tool for 2D images and 3D point clouds. Often abbreviated as PCL, it contains numerous state-of-the-art algorithms for feature detection, surface reconstruction, segmentation and many other subareas of computer vision. It is well documented and easy to use. Recently a machine learning library containing algorithms such as Support Vector Machine has been

added. Some algorithms are multi-threaded thanks to the OpenMP library. A GPU module boosts some algorithms. Unfortunately the PCL is relatively new and its stability could be better. Currently available 1.7.0 version is going to be replaced by soon-to-be-announced 2.0 version with incompatible API.

Every feature detector and descriptor used in this project comes from the PCL.

1.3.2 OpenCV

Considerably more mature, the OpenCV library is yet another great image processing library. A multitude of methods implemented in both C (OpenCV 1.x) and C++ (OpenCV 2.x) allows for production of fully functional image processing or computer vision application with this single library. There are many machine learning mechanisms as well. Among them one can find kMeans, Gaussian Mixture Model, Naïve Bayes or a Support Vector Machine. Some of the more demanding in terms of computational power are parallelized with Threading Building Blocks (TBB). A module exploiting powers of the nVidia's CUDA technology exists in order to enhance computation even more.

In this paper OpenCV is used for input/output operations, image preprocessing. KMeans and SVM implementations are used as well.

1.3.3 Boost

Boost is a general purpose library. It contains more than 50 separate libraries, 12 of which were incorporated into the new C++11 standard. One of the most popular are libraries for smart pointers, multi-threading or meta-programming. Tagger3D uses only the functionality provided by the Boost.program_options library. It enables convenient parsing of configuration files and command line arguments.

1.4 Datasets

Multiple RGBD datasets were created. The purpose of authors of the vast majority of them were to provide a benchmark for tasks of tracking or instance-level object recognition. Others have small number of categories or very few examples in every category. Unfortunately, the most are not fit for the object classification problem. The RGB-D object dataset from the University of Washington [12] would be perfect, if not for the fact

that it consists of video sequences. A technology like Kinect Fusion [3] makes it relatively easy to build a detailed 3D model (a point cloud or a mesh) from such data. The only problem is in the computational cost of this method and actions required to gather such data — in a real setting a mobile robot would have to circle an object. Two datasets are used are for the purpose of this paper. The first one is the Berkeley 3D Object dataset [11] and the second one is a dataset compiled by Zhang *et al* at the University of Tokyo [24]. The datasets are going to be abbreviated as B3DO and Tokyo datasets respectively

1.4.1 Berkeley 3D Object Dataset

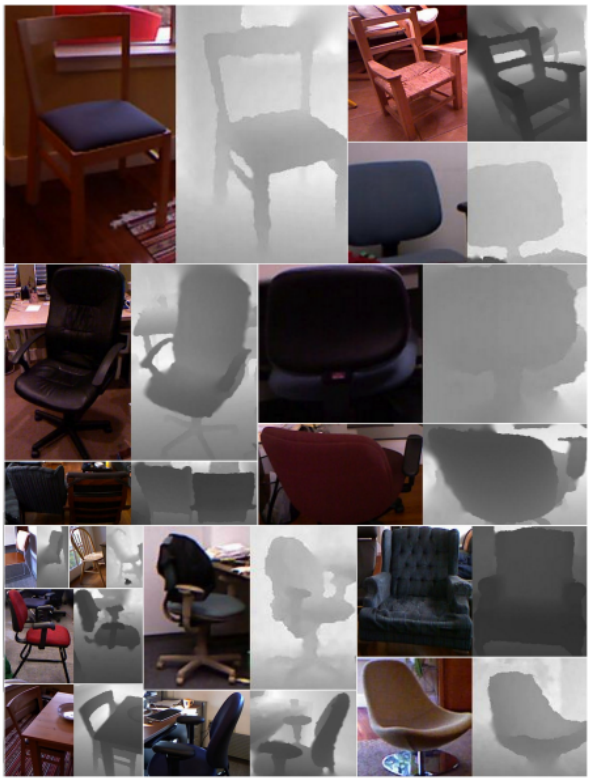

Figure 1.4: Berkeley 3D Object Dataset. The figure comes from [11]

The Berkeley 3D Object dataset was specifically designed for the purpose of object classification. It consists of cluttered images in indoor environment. There are around 50 classes with more than 20 examples in each of them. RGBD data is provided as pairs of images. There are 8 bit RGB jpeg pictures and 16 bit png files containing depth data in millimetres. Images are densely labelled — for every pair of images there is an xml with annotations. Every annotation is comprised of a category label, bounding box coordinates and object's orientation.

Before the B3DO database could be used it had to be preprocessed. Neither image segmentation nor object detection are addressed in this paper. Because of that the objects had to be extracted from the cluttered images.

1.4.2 The University of Tokyo Dataset

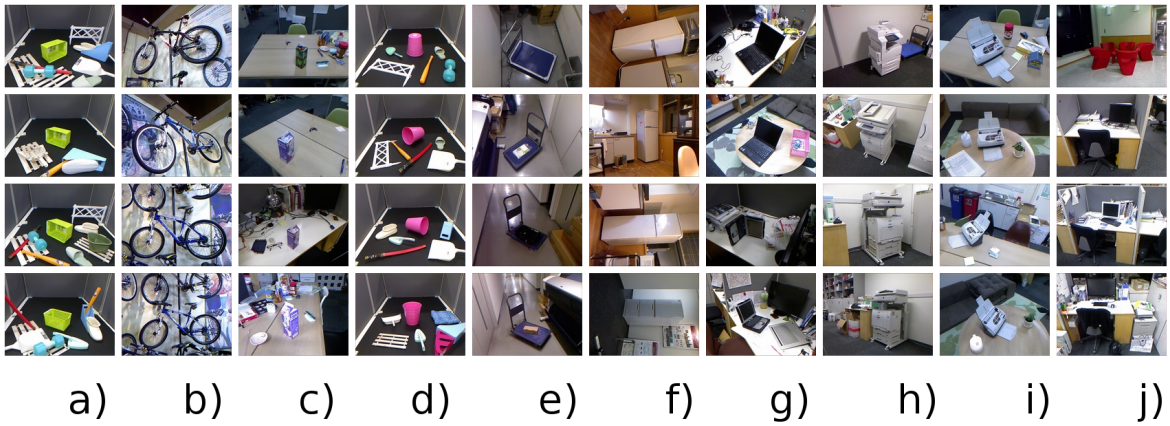

Figure 1.5: The Tokyo dataset: a) basket b) bicycle c) box d) bucket e) cart f) freezer g) notebook h) printer i) scanner j) scene

The dataset compiled at the University of Tokyo is comprised of high quality images of objects in different settings. Factors such as viewpoint, object orientation, object's texture and environments are changing on different images. The aim of the authors of this dataset was slightly different - it was object detection and classification in casual images. There are 10 classes with varying amounts of per class examples. The database is available as two sets of files: 8 bit RGB jpeg files and csv files containing euclidean coordinates of every pixel in the corresponding colour image.

Chapter 2

Design

2.1 Software Functionality and Architecture

The programme resulting from this work is called "Tagger3D", for it's purpose is to tag an RGBD image of an object with a category label. The Bag of Words classification pipeline consists of four main parts discussed previously. In order to determine the best combination of algorithms for any given task an extensive research and evaluation has to be performed. In order to simplify the process the software system should be configurable, extensible and easy to maintain. Moreover, if it is to be used in any realistic setting it has to be efficient as well.

2.1.1 Required functionality

There are three major functions that have to be supplied by the programme: (1) estimation, (2) batch inference and (3) inference. The first one is responsible for the model training. In order to perform this operation the following steps have to be completed: (i) detection and description of keypoints on the training set, (ii) codebook generation, (ii) classifier model training. The functions (2) and (3) are somewhat similar. Both require: (i) detection and description of keypoints on the test set, (ii) parsing keypoints' description into a visual vocabulary description and (iii) classification. The only is in the (i) step, e.g. the batch inference requires every image from the test set to be processed, whereas the inference can be done on any single image. The (3) have to be distinguished because it is the function that is to be used in a fully functional system mounted on i.e. a mobile robot.

Input/Output operations

Additional functionality have to be provided in the field of input/output operations. Before any processing can be done each image have to be read into memory. Both batch inference and estimation are dataset dependant. What I mean by this is that a previously compiled database have to be accessible. The inference, on the other hand, is meant to be used in a production environment i.e. with a Kinect connected only. While the OpenNI Grabber framework from the PointCloud Library is essentially what is required for the inference, a little more elaboration has to be done on the demands of the estimation and batch inference environments. The two third party datasets used in the project have different formats and need to be tackled separately. The B3DO object dataset contains multiple objects in a single image. The locations of the images are available in xml files, while rgb and depth data are stored in separate files, 8-bit 3 channel jpeg and 16 bit one channel png respectively. Single objects have to be extracted from both pictures using the provided annotations and point cloud have to be build. The Zhang's dataset contain single objects. The rgb data is stored in 8 bit 3 channel jpegs and csv files contains euclidean coordinates of every rgb pixel. Both databases contain objects' labels that have to be loaded separately.

A different problem is serialisation. If the models from the training stage are to be used during inference, they have to be serialised. This means that a standardised way of saving and loading of kMeans and SVM models have to be provided.

Configuration

Every algorithm employed in the project have several configuration parameters. If the parameters were to be compiled with the code, every change would require a rebuild of the application or its parts. In order to address this issue a means of configuration from outside the code should be provided.

Interchangeability of algorithms

Multitude of available algorithms makes it time consuming and computationally expensive to evaluate every available option. It should be possible to compile the programme with many algorithms suitable for performing the same task and decide which one should be used at run-time.

2.1.2 Architecture

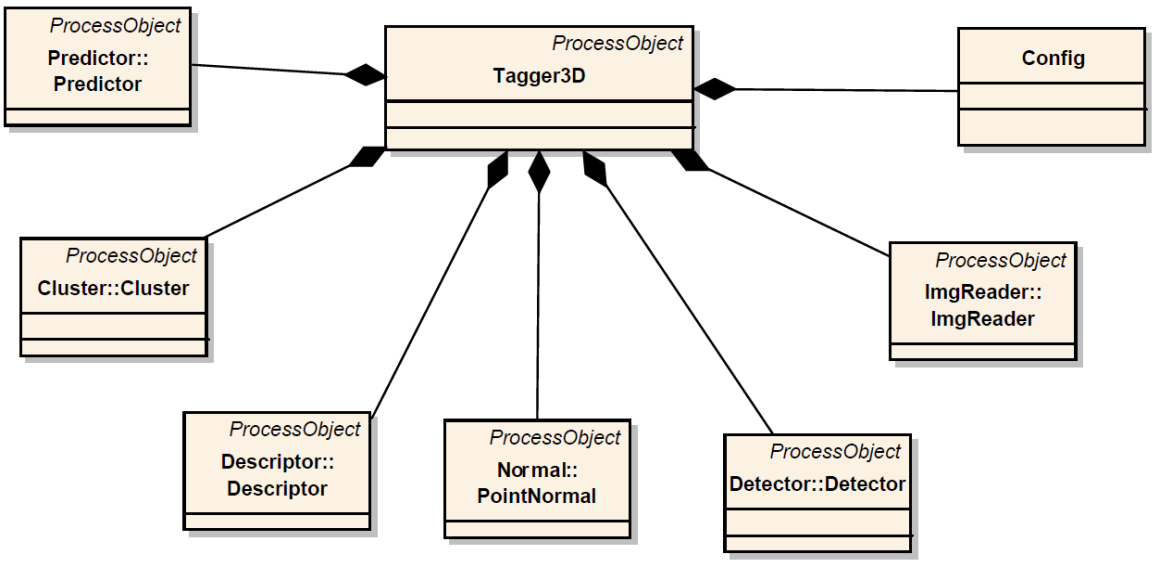

Figure 2.1: Tagger3D class diagram

The described functionality can be achieved in the following way. The utilisation of Object Oriented Programming (OOP) can lead to production of modular, flexible and extensible software. Using C++ as the language of choice can result in high efficiency. The excellent meta-programming tools provided by the C++ language help minimise the length of resulting hand-written code.

Configurability

In order to achieve configurability all the configuration parameters are stored in a configuration file. A separate object has been constructed for configuration file parsing. Simply called 'Config', it makes use of the Boost.program_options library to parse both the configuration file and command line arguments. It returns an associative array of (parameter, value) pairs. The convention is that if an object has to be configured the configuration map has to be passed in the object's constructor (other constructors are private).

Strategy Design Pattern

Design patterns has been introduced for the very first time by Gamma *et al* in [9]. They are best practices that should be used to solved common, recurring problems in software design. One of the patterns is the strategy pattern. It enables implementation of multiple

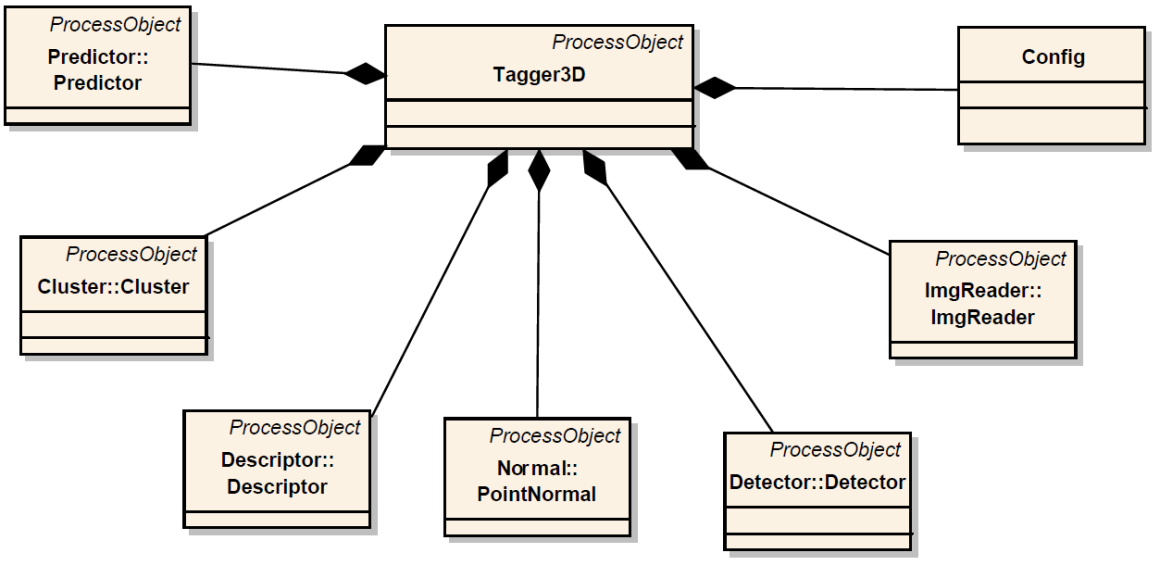

Figure 2.2: Strategy pattern

algorithms handling some task and selection the desired behaviour at run-time. It is the exact problem encountered here. It is stated that an interface should be defined for each task and multiple classes encapsulating different algorithms for handling the tasks should realise those interfaces. One modification has been done: Algorithms have often some common operations, thus an idea of an interface (a class without any implemented methods) was dropped in favour of a virtual class.

The strategy pattern was used for every processing step of the pipeline so as to make it extensible. One might want to enhance this solution by using an abstract factory design pattern, which has not been done in this case. It might make the code easier to maintain and read. However, the project is relatively small and the bulk of an abstract factory is unnecessary.

There six five classes that can be treated as algorithms' interfaces in a strategy pattern. They are: ImgReader, PointNormal, Detector, Descriptor, Cluster, Predictor. Each of them contains pure virtual computation methods — 'read' in case of the ImgReader or 'predict' in case of the Predictor. Additionally, they have some common methods defined — I/O methods in Cluster or batch processing methods (such that take a vector of elements instead of a single element as input) for the Detector and the Descriptor. Every pure virtual class has its specialisations. For example there is the PFHDescriptor class with a Point Feature Histogram descriptor or a KMeansCluster employing kMeans clustering algorithm. The Tagger3D class chooses specialisations of the algorithms at

run-time, depending on the configuration parameters.

Chapter 3

Experiments and Results

3.1 Experimental Setup

Thanks to the configurability of the programme the majority of experiments were run in a batch processing mode. That is a script in bash was written that allowed running multiple tests in series without any maintenance from the user. In a single series of tests any parameters could be modified or any data processing algorithm could be exchanged. All experiments were run on a notebook with Intel Core i7 quad core 2.2 GHz CPU, 8 GB of RAM and a nVidia M540 GPU.

3.2 Experiments

The Bag of Words classification pipeline consists of four previously discussed steps. For there are many different algorithms suitable for each part of the pipeline, an extensive research is required in order to discover the best possible combination in a particular setting. The number of options grows combinatorial with number of candidates for each processing step. Moreover, many algorithms have different parameters that should be adjusted for any given conditions. Evaluation of every possibility is unfeasible due to the lack of time and computational power required. Even if only a single library, PCL, is considered, there are far too many options with 12 keypoint detectors and more than 20 feature description algorithms.

3.2.1 Detectors

In a paper [8] some keypoint detection algorithms from PCL are evaluated. It is stated that , albeit under slightly different conditions, the SIFT and ISS keypoint detectors achieve the highest performance. Therefore this paper compares only these two modules with respect to their usability in the BoW pipeline.

3.2.2 Descriptors

Recently a PCL's descriptors evaluation was performed in [4]. The PFHRGB and SHOT-COLOR scored best. Only a little bit worse were PFH, FPFH, SHOT and USC. In this paper FPFH, PFH and PFHRGB are compared, motivation being that PFH is the original descriptor, FPFH is its approximate and thus faster version, while PFHRGB is an extension which makes use of colour information.

Descriptor	Accuracy [%]
FPFH	65.22
PFH	59.32
PFHRGB	63.35

Table 3.1: Highest accuracy obtained with FPFH, PFH and PFHRGB descriptors on the B3DO dataset

The figure 3.1 shows highest accuracy obtained with the tested descriptors on the B3DO dataset. FPFH scored the best result, even tough it is an approximate and theoretically the least precise of all evaluated descriptors. On the other hand, the PFHRGB scored better than PFH. It does not surprise, since the colour information is believed to have discriminative value.

3.2.3 Codebook

KMeans is used in almost every implementation of the Bag of Words image processing pipeline [22, 23]. Csurka *et al* showed that the number of visual words or centroids have a significant impact on the final results [6]. Therefore a number of tests were run in order to determine the optimal dictionary size.

The results from the figure 3.1 partially confirm Csurka's findings. The performance raises with the increasing number of centroids up to the point of 1500 centroids and 65.22%. Then it starts to diminish. The discrepancy might be caused by the following factors: (1) ISS detector finds too many or irrelevant points, thus introducing noise or

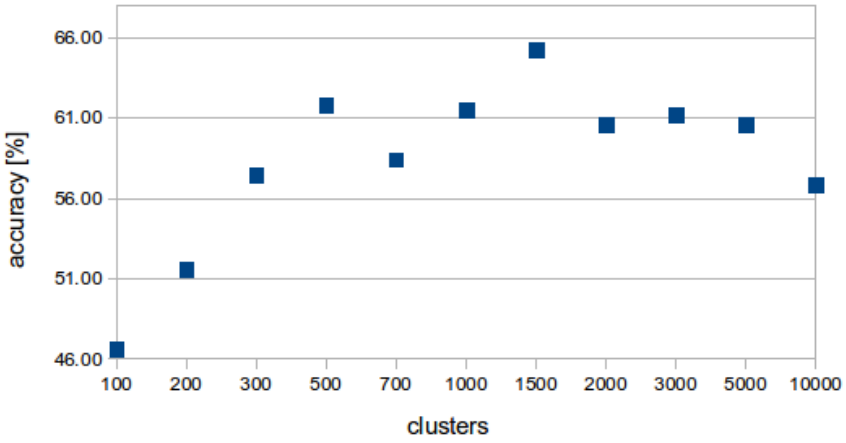

Figure 3.1: Influence of dictionary size on the overall accuracy. B3DO dataset with ISS keypoint detector and FPFH feature descriptor

(2) The FPFH descriptor has too few dimensions (33) to be divided into more than 1500 regions in a meaningful way.

3.2.4 B3DO

Extraction of single objects yielded 78 separate categories with number of entries ranging from 1 (tape) to 299 (table). 8 categories were picked at random with a restriction that there should be at least 50 instances in each of them. Further, the objects were split into two sets (training and testing) with a proportion of 1:1. As names suggest, the estimation is performed on the test set and the evaluation on the test set.

The highest accuracy achieved for this dataset is 64%. Table 3.2 contains a confusion matrix, number of examples per category and accuracy. The confusion matrix depicts misclassification errors. Let $m_{i,j}$ be an element of the confusion matrix at the i^{th} row and j^{th} column. It shows how many elements from the i^{th} category was assigned to the j^{th} category. A high value of $m_{i,j}$ such that $i \neq j$ indicates that the classifier cannot distinguish between those two classes.

More than half of the bottles were assigned to the cup category and keyboards were often mistook for books. These two error types are easily explained by a high degree of similarity of objects (bottles and cups are usually round and tall, books and keyboards are flat and rectangular) Surprisingly, however, objects from half of the categories were frequently marked as cups. Some of these misclassification errors might be caused by very

Table 3.2: Results on the B3DO dataset with ISS keypoint detector, FPFH features and a dictionary of 1500 words. **Overall accuracy is 65.22%**

Assigned to category \ Category	cup	book	sofa	chair	table	bottle	monitor	keyboard	Entries per class	Accuracy [%]
cup	**59**	0	0	0	0	5	0	0	64	92.19
book	17	**14**	0	0	0	7	0	2	40	35.00
sofa	0	0	**17**	3	1	0	0	0	21	80.95
chair	0	0	2	**14**	4	0	2	0	22	63.64
table	0	0	0	0	**31**	0	0	0	31	100.00
bottle	34	0	0	0	0	**30**	2	0	66	45.45
monitor	7	1	0	2	0	10	**30**	1	51	58.82
keyboard	8	4	0	0	0	0	0	**15**	27	55.56

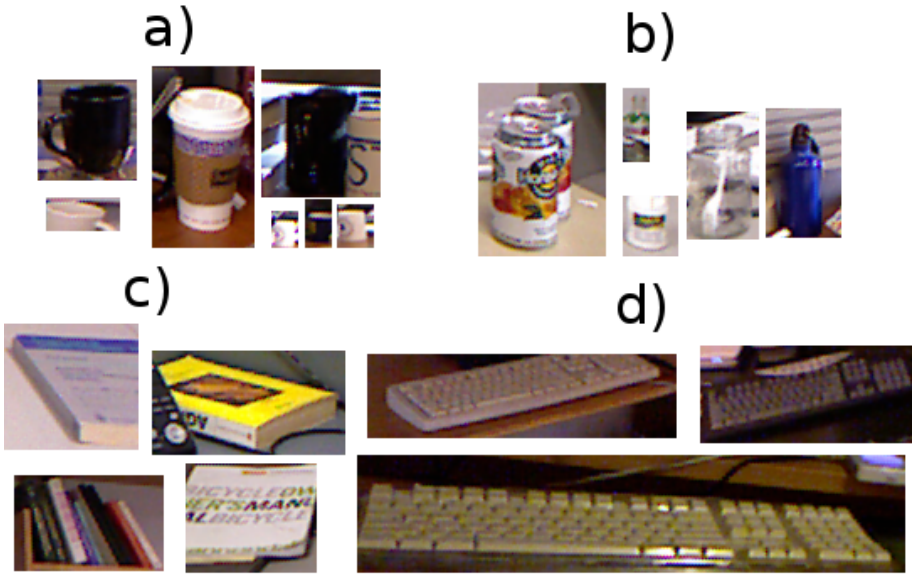

Figure 3.2: B3DO objects. Images are in their original sizes. a) cups b) bottles c) books d) keyboards

poor quality of images. Many of them are occluded poorly lit low resolution images.

3.2.5 Tokyo

Every of the 343 images from Tokyo dataset was used. Data was split into test and train set in proportions 1:2 in order to provide more training examples due to low number of objects in some classes.

Even tough the overall accuracy achieved on Tokyo dataset is similar in value to that of the B3DO, the structure of the result is very different. It is clearly visible that there is a strong correlation between a per class accuracy and the number of entries in this class.

Table 3.3: Tokyo confusion matrix with ISS keypoint detector, PFH features and a dictionary of 3000 words

Assigned to category \ Category	box	cart	notebook	scene	scanner	basket	bucket	freezer	bicycle	printer	Entries per class	Averages [%]
box	**11**	0	2	0	0	0	0	0	0	0	13	84.62
cart	0	**0**	1	0	0	0	3	0	0	0	4	0.00
notebook	6	0	**3**	1	0	0	1	0	1	0	12	25.00
scene	0	2	2	**4**	0	0	1	0	1	0	10	40.00
scanner	2	0	0	0	**0**	0	0	0	0	0	2	0.00
basket	1	0	0	1	0	**2**	7	1	1	0	13	15.38
bucket	3	0	0	0	0	0	**22**	1	0	0	26	84.62
freezer	0	0	2	0	0	0	1	**1**	0	1	5	20.00
bicycle	1	0	0	0	0	0	1	0	**33**	0	35	94.29
printer	0	0	0	0	0	0	1	0	1	**0**	2	0.00
											Average	**62.30**

The highest performance in the bicycle category is coupled with the largest number of entries. On the other end of the scale there are cart and printer categories with only 2 and 4 entries respectively. On top of that there are differences between some classes are marginal. The majority of carts and baskets were put into the bucket category. It does not surprise, for they are simply akin as can be seen in figure 1.5 on the page 18.

3.3 Conclusion

List of Figures

1.1 Bag of Words histogram 9
1.2 Bag of Words pipeline. The figure comes from [23] 10
1.3 Classification by a Support Vector Machine. The distance of the separating plane from both datasets is maximal. The figure was originally published in [15] 15
1.4 Berkeley 3D Object Dataset. The figure comes from [11] 17
1.5 The Tokyo dataset: a) basket b) bicycle c) box d) bucket e) cart f) freezer g) notebook h) printer i) scanner j) scene 18

2.1 Tagger3D class diagram 21
2.2 Strategy pattern 22

3.1 Influence of dictionary size on the overall accuracy. B3DO dataset with ISS keypoint detector and FPFH feature descriptor 26
3.2 B3DO objects. Images are in their original sizes. a) cups b) bottles c) books d) keyboards 27

List of Tables

3.1 Highest accuracy obtained with FPFH, PFH and PFHRGB descriptors on the B3DO dataset 25
3.2 Results on the B3DO dataset with ISS keypoint detector, FPFH features and a dictionary of 1500 words. **Overall accuracy is 65.22%** 27
3.3 Tokyo confusion matrix with ISS keypoint detector, PFH features and a dictionary of 3000 words 28

Bibliography

[1] Apache log4cxx.

[2] Boost c++ libraries.

[3] Microsoft kinect fusion.

[4] L. A. Alexandre. 3d descriptors for object and category recognition: a comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal*, 2012.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, page 22, 2004.

[7] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 524–531. IEEE, 2005.

[8] S. Filipe and L. A. Alexandre. A comparative evaluation of 3d keypoint detectors.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: Abstraction and reuse of object-oriented design*. Springer, 1993.

[10] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[11] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3d object dataset: Putting the kinect to work. In *Consumer Depth Cameras for Computer Vision*, pages 141–165. Springer, 2013.

[12] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[13] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[15] J. Ponce, D. Forsyth, E.-p. Willow, S. Antipolis-Méditerranée, R. d'activité RAweb, L. Inria, and I. Alumni. Computer vision: a modern approach. *Computer*, 16:11, 2011.

[16] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009.

[17] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[18] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz. Persistent point feature histograms for 3d point clouds. *Intelligent Autonomous Systems 10: Ias-10*, page 119, 2008.

[19] S. Salti, F. Tombari, and L. D. Stefano. A performance evaluation of 3d keypoint detectors. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on*, pages 236–243. IEEE, 2011.

[20] S. M. Smith and J. M. Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997.

[21] J. W. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia tools and applications*, 39(3):441–471, 2008.

[22] R. Toldo, U. Castellani, and A. Fusiello. A bag of words approach for 3d object categorization. In *Computer Vision/Computer Graphics CollaborationTechniques*, pages 116–127. Springer, 2009.

[23] C.-F. Tsai. Bag-of-words representation in image annotation: A review. *ISRN Artificial Intelligence*, 2012, 2012.

[24] Q. Zhang, X. Song, X. Shao, R. Shibasaki, and H. Zhao. Category modeling from just a single labeling: Use depth information to guide the learning of 2d models. In *Computer Vision and Pattern Recognition, 2013. CVPR 2013. IEEE Computer Society Conference on*. IEEE, 2013.

[25] Y. Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 689–696. IEEE, 2009.