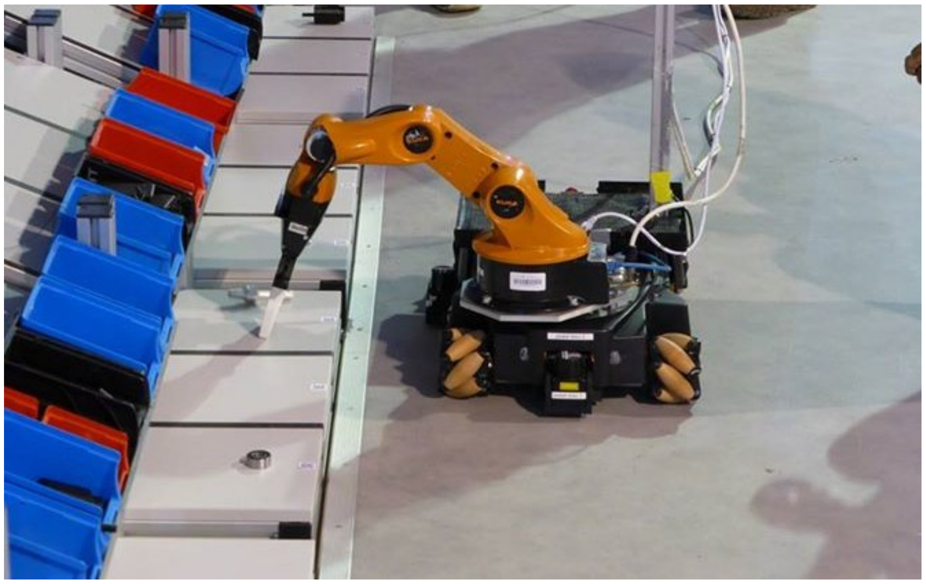
Taking up the RoCKIn@Work Object Recognition Challenge With The Bag of Keypoints Approach

Areg Shahbazian,
10283234

BSc Kunstmatige Intelligentie thesis
18 ECTS

Supervisor
Dr. Arnoud Visser

Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam



Abstract

This research project addresses the problem of object-classification by a robot in an industrial context. Solving this problem brings industrial robots one step closer to being able to operate autonomously and eventually to reproduce themselves. The approach described in this thesis enables a system to distinguish classes of objects using RGB data only, which can be a difficult task when the classes are similar. Other approaches have been to compare generic features of objects to classify them, and to build histograms of more distinct features. By performing experiments I was able to determine some specific parameters for our problem and combine different types of input features. By experimenting with different classification techniques, error-rates of nearly 22% for object type and 11% for object category were reached.

Contents

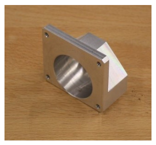
1 Introduction

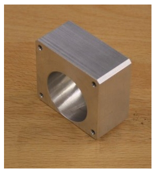
In an increasingly industrialized world, the field of robotics has gained much significance for factories and manufacturers, with the purpose of utilizing machine intelligence to produce faster and more efficiently. Furthermore, having intelligent robots participate in their own reproduction is a big step in developing autonomous robotic intelligence. An important functionality-requirement for such a robot is object recognition and classification using its sensors.
This thesis describes my research project about object-classification using RGB camera-data. The research is done in the context of the RoCKIn@Work Competition[1].
The purpose of the @Work is to challenge competitors to develop innovative industrial solutions using robots. In the RoCKIn competition, the goal is to develop a robot-system that assists with the partial assembly of another robot. This involves locating and transporting necessary parts, checking the quality of these parts and manipulating them with actuators.

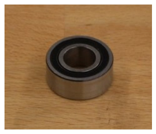
The project focuses on the perception of the different objects, specifically for classification of the object types using camera images. This task is relevant for the location and recognition of objects, as well as for object-manipulation. For the robot to be able to pick up and manipulate the objects at the RoCKIn@Work competition, a pose-estimation functionality is also required. This problem will however not be assessed in this thesis and the focus will be on the classification of the objects.
The problem definition of this research project is inspired by the "Object Perception" functionality benchmark of RoCKIn@Work 2014. In this benchmark, an object is placed on a white-surfaced table, and using a sensor of choice, the competitor-robot must classify the object as one of the known types. The competitor is allowed to make a dataset of images of all objects. The competition also provides the competitors with 3D models (.STL) of the objects, which can be used in combination with depth-images to identify the correct type.
In this project however, no depth images or any other 3D data is used and object-classification is performed using only RGB images. The reason for this is that most of the RoCKIn@Work objects are made of shiny metal (Figure 1.1), which reflects most of the rays used by sensors to make a depth image and leads to incomplete depth maps.

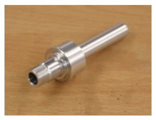
RGB data is used by finding interesting points (key-points) in the image and describing these in vectors. Then, following the Bag of Key-points approach ([5]), image histograms are constructed and are used to classify new images with a classification algorithm. This is the classic way of applying the Bag of Key-points method to the object-classification problem.
In this thesis some questions will be addressed regarding the customization of this method for the RoCKIn@Work object-classification task. First, the effect of the number of clusters used for making the image histograms will be examined. Then, dif-

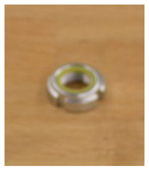
[1] http://rm.isr.ist.utl.pt/projects/rockin-competitions-wiki/wiki/FBWork#FBM1-%E2%80%9CObject-perception%E2%80%9D

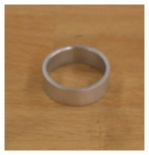

(a) type 0

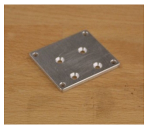

(b) type 1

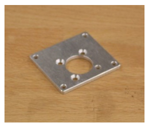

(c) type 2

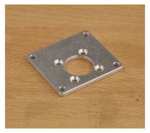

(d) type 3


(e) type 4


(f) type 5


(g) type 6


(h) type 7


(i) type 8

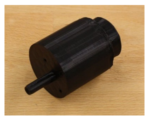

(j) type 9

Figure 1.1: The ten RoCKIn@Work object types. Note the small difference between object type 7 and type 8 (the bottom-right screw hole around the round gap is smaller for type 7).

ferent types of features of objects and their effect on the classification result will be investigated. Finally, experiments will be done using different classification algorithms to try and find the one with the best results.
In the following sections a review will follow of some relevant literature that inspired the approach of this study. Then the method of addressing the object-classification problem will be presented. The type of data used, the way it's represented and the method of classifying it will be explained. Experiments will be performed with a program written in the C++ language and the results of the experiments will be presented. Finally, the thesis will be concluded and suggestions will be made for future improvements of the approach.

2 Literature Review

The “Object Perception" Functionality Benchmark has been tackled by many competitor teams of the @Work challenge. The approach of the winner of the 2014 RoboCup@Work competition, the smARTLab@Work team[2] was to use various visual features of the objects such as the length of the principal axis and the intensity of the image ([3, p. 9]). Using a labelled dataset, the new images were classified with a decision-tree algorithm.
Although the approach described above led the team to victory, its applications are limited. Among the benchmark-objects used in the RoCKIn@Work competition are objects with very similar shapes and average intensities (Figure 1.1h and 1.1i). Using these features to distinguish the objects does not seem promising. The smARTLab@Work team solved this problem by changing the camera position (moving the robot arm on which the camera is mounted) to see the objects from more distinct angles. Following the rules of the RoCKIn@Work 2014 functionality-benchmark, the camera-sensor is not allowed to move, and image-data from a single point of view has to be used. This requires more distinct features of the objects to be extracted from the sensor-data.
In the RoboCup@Work 2014 challenge object perception functionality is also required. Here, the winning team (Wright Eagle[3]) used depth-images to recognize objects. As described in the introduction, using a depth-sensor is not a good option for the RoCKIn@Work objects, so RGB images are the best choice. The team description papers of the RoboCup@Work 2013[4] and RoboCup@Home 2014[5] competitors are published on their websites. Unfortunately, the team description papers of the RoboCup@Work 2014, which is most relevant for this project, have not been published.

Instead of using generic information about the shape and color of the objects more distinct features could be used to perform the task. The features should be extracted and represented in a way that is invariant to changes in the object-pose and environment lightning. The SIFT method, as described in [9], uses features extracted from an image using multiple scales. This scale-space is searched for interesting key-points and these are described in a way that is invariant to shape distortion and changes in illumination. [9, p. 2]. These descriptors are used to match objects in new images with images in the dataset in order to assign an object-type to the new object.
The SIFT method performs well in cluttered environments. Since there is minimal cluttering in the object-environment of the RoCKIn@Work, good results are expected in matching, if the objects are distinctive enough. However, as Figure 1.1 shows, the objects of the RoCKIn@Work benchmarks are not all highly distinctive.
Similarly to SIFT, the SURF method also uses scale- and rotation-invariant features

[2]http://www.robocup2014.org/?page_id=3297

[3]http://www.robocup2014.org/?page_id=3293

[4]https://staff.fnwi.uva.nl/a.visser/activities/robocup/RoboCup2013/Symposium/TeamDescriptionPapers/RoboCup@Work/index.html

[5]http://fei.edu.br/rcs/2014/TeamDescriptionPapers/RoboCup@Home/index.html

to describe key-points in images. Here, the Hessian-matrix approximation of an image is used to detect key-points. Also, instead of the gradient used in SIFT, Haar wavelet responses are used to describe a key-point. These are calculated within a circular neighbourhood of the detected point in the x and y direction to assign an orientation to it. To calculate the descriptor of a key-point, a rectangular region around it is divided in 16 subregions. For each subregion a 4-dimensional descriptor is calculated using Haar wavelet responses, leading to a 64-dimensional descriptor for each key-point location, instead of the 128-dimensional SIFT descriptors.
SURF is claimed to outperform SIFT [1, p. 7] and is a more recent method. Furthermore, the lower dimensionality of the key-point descriptors is a big advantage when using a large dataset of images.

In addition to feature extraction and description, the Bag of Keypoints method, as described in [5], uses the *Harris affine detector* to extract interesting key-points from grey-scale image and uses the SIFT-descriptors of these key-points to form a vocabulary of key-points. Then, for each image, a histogram is formed that contains the number of key-points per category, where the categories are determined using clustering. The number of clusters to be used is chosen by running the k-means algorithm several times and selecting the value of k that gives the 'lowest empirical risk in categorization' [5, p. 6]. However, this way of empirically selecting the value for k is can be costly when there are many data-points to be clustered. The approach in [5] produces state-of-the-art results on a dataset of 1776 images of seven different classes of objects (faces, buildings, trees etc.). These images contain a significant amount of background cluttering and the objects to be classified are often partially occluded.
In [12] a method is presented to choose the value for k. As k is increased, the error measure of the clustering algorithm, which in the case of [12] is the within cluster dispersion, decreases at a lower rate. From some k-value onwards this decrease flattens significantly and at the location of that k, the plot of the decreasing error rate resembles an 'elbow' shape. [12] presents a procedure to formalize this way of choosing k.

Another aspect of object-recognition using image features is the way the feature-vectors are used to match a new image with the existing dataset. The smARTLab@Work team uses a J4.8 decision tree ([3, p. 9]) to classify observed features. In [5] a Naïve Bayes classifier and Support Vector Machines are used for the classification, with the latter outperforming the former [5, p. 14]. Many other classification algorithms exist. The WEKA Data Mining Software ([7]) provides a workbench for experimenting with different classification methods and a can be convenient tool to compare results and help choose the right method.

3 Method

In the following subsections a method to solve the object-classification problem will be explained. The version of the Bag of Key-points method used in this research project will be explained and enhancements of the system in different areas will be presented. Some of these enhancements are specific to the Bag of Keypoints method, while others are more general customizations of the system for the object-classification problem of the RoCKIn@Work challenge.

3.1 Bag of Key-points

In this research the Bag of Key-points approach is used, inspired by the method of Csurka *et al.* [5], as a basis. A description of the classic implementation of this method will follow below. As described above, this approach leads to good results on a dataset with background cluttering and where the objects are partially occluded. However, the experiments in [5] are performed on a dataset where the different classes of object are very distinct. For example, recognizing the difference between a face and a building is arguably simpler than distinguishing between two very similar metal objects from Figure 1.1. Fortunately, when performing object recognition for the RoCKIn@Work, we don't have to worry about much cluttering in the images or occlusion of the objects. These differences between the experimental setup of this project and that of Csurka *et al.* [5] make it necessary to customize the approach.
The Bag of Key-points method uses a dataset of descriptors of interesting key-points to construct image histograms of length k. In the implementation of this research project, the key-points are detected and described using the SURF detector and descriptor implementations, which are included in the OpenCV library ([2]) and implemented according to the algorithms described by Bay *et al.* [1]. From each of the N images a number of key-points is detected and described. On average around 120 key-points are detected per image in the dataset. Each descriptor is a 64-element vector and represents the neighbourhood of a detected key-point. Denoting the average number of key-points per image as μ, this leaves us with $N_{descr} = \mu N$ descriptors in a $N_{descr} \times 64$ matrix.
Many of the descriptors extracted from images are very similar, and some represent the same key-points transformed between images. They are simply occurrences of local neighbourhood-types in images. These descriptors could be used directly to match images. However, this is computationally impractical, given the great amount of descriptors for the whole dataset (typically around 12000 in this case).
Figure 3.2 shows the descriptors of the key-points which were extracted from 1029 images of the ten objects. Examples of images, with some key-points illustrated as ellipses, are shown in Figure 3.1.
Figure 3.2 shows the projection-image of the data projected on its three main component-vectors (using the Principal Component Analysis (PCA, [11]) implementation from OpenCV). The descriptors of 3.2a are ordered by the object-types from which the key-points are extracted. As can be seen in the figure, the image type is not a distinct feature of the descriptors. Data-points with different colors don't occupy separate subspaces of the data-space and are not ordered or grouped in a distinct way. A de-

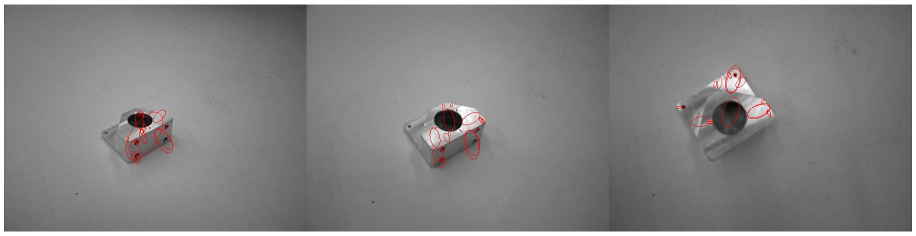
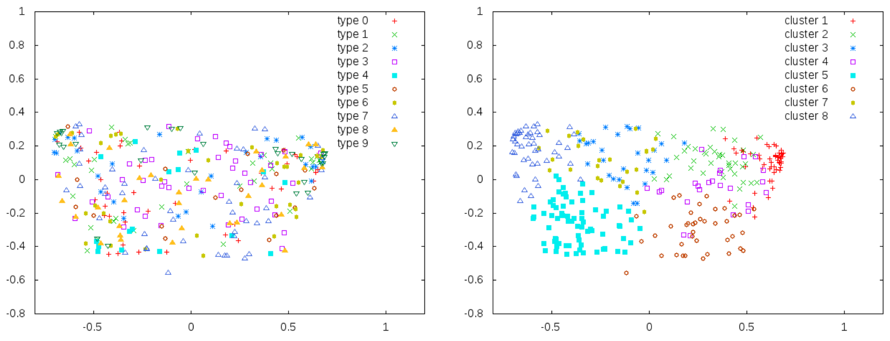


Figure 3.1: Three images of the same object, with SURF keypoints.

scriptor, or a group of descriptors, can not be used directly to determine the object-type of the image.



(a)

(b)

Figure 3.2: Descriptors projected on their main three dimensions. In 3.2a ordered by object-types, in 3.2b all 130000 descriptors ordered by cluster number

Instead, the key-points in each image can be summarized in histograms. This is done by clustering all N_{descr} descriptors into k clusters (categories) and describing each image i in a k-dimensional histogram vector $\vec{h_i}$ (Figure 3.3). Here, the number of descriptors in image i that fall within cluster j (i.e. the number of category-j key-points in image i) is the value at index j of $\vec{h_i}$, with j in $[0,k)$.

Clustering is an important step and in this project it is done using the k-means algorithm included in OpenCV ([2, p. 479]). This algorithm starts with k random points in $\mathbb{R}^{64}$ as cluster centres. It then assigns a cluster number to each data-point (descriptor vector). Then each cluster centre is moved to the centroid of its data-points. The last two steps are repeated until some convergence is reached. The clustered descriptor dataset is shown in Figure 3.2b.
Forming histograms for each image using the clusters is a way of summarizing the higher-dimensional descriptor-dataset into a much smaller, lower-dimensional histogram-dataset (the dimensionality depending on the choice of k for clustering). This re-

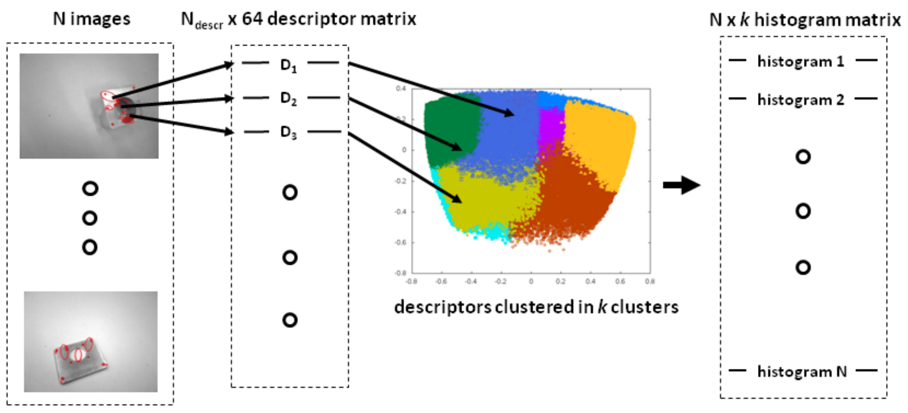
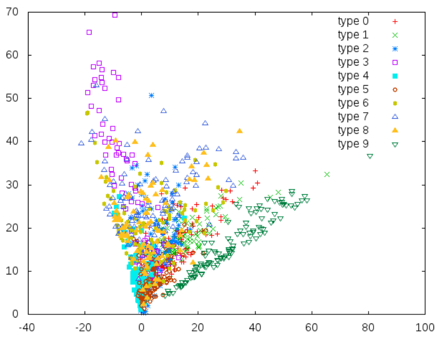
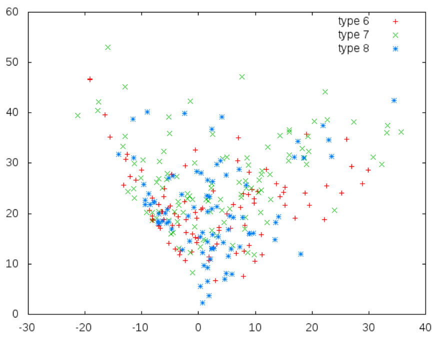


Figure 3.3: Schematic representation of the Bag of Keypoints method: SURF key-points are extracted and represented in descriptors, with on average μ descriptors per image. The descriptors are clustered in k clusters. For each image, the number of descriptor-occurrences per cluster is stored in a histogram.

duces the computational power needed to process the image-data.
Another benefit of clustering the descriptors is its intuitive usefulness when using scale- rotation- and illumination-invariant key-points. Key-points of this kind are meant to have multiple occurrences in different images of the same object. Even though some key-point may have different descriptors in different images, these different descriptors will be close to each other in the 64-dimensional descriptor-space, since they represent the same spot on the same object. These advantages of using histograms are illustrated in Figure 3.4. The histograms of the images containing object-type 9 (the most distinct of the objects, Figure 1.1j) clearly occupy a different subspace of the histogram-space than those with other object types (Figure 3.4a), while the very similar object-types 6, 7 and 8 have histograms largely overlapping the same subspace (Figure 3.4b).
This way, an $N \times k$ histogram matrix is formed, which represents each of the N images as a histogram of the categories it contains. In order to classify an object in a new image, key-points from the image are extracted and the descriptors are calculated. Of the k cluster-centres, the nearest one is assigned to each descriptor, and counting the number of descriptors per cluster-centre a histogram of the new image is formed.
The k dimensional histogram of the sample image can be used to assign an object type to the image. The simplest way of doing this is perhaps assigning to an image the object type of the histogram-vector from the dataset with the smallest Euclidean distance to it in $\mathbb{R}^k$.



(a)



(b)

Figure 3.4: Two 2D plots of the histograms ($k = 8$), projected on their three main dimensions. The histograms are ordered by the object-type of their image. Histograms of the most similar objects are ordered very similarly (right)

3.2 Significance of k

Having seen the intuitive usefulness of clustering, a less intuitive point-of-choice in this process must be acknowledged: the number of clusters to be formed (k). Having a small k reduces the dimensionality of the data but brings with it the risk of insufficient expression of the diversity of the descriptor-types. On the other hand, a large value of k will significantly slow down the process of clustering. In [5, p. 9], this choice is made by empirically experimenting with different values of k and choosing one that presents “a good trade-off between accuracy and speed". Although the way these experiments are performed is not described, it’s implied that the accuracy is the classification-accuracy of the object-recognition algorithm. Similar experiments will be performed in 4.1 to determine the desired value k^* for the rest of the experiments.

3.3 Extra Features

In the classic implementation of the Bag of Key-points algorithm the image histograms contain only vocabulary information: the number of key-points from each category. SURF key-points are easy to extract from images and are scale- and rotation-invariant. They also provide some robustness to occlusion of objects and cluttering in the image. However, it’s hard to distinguish objects that look very similar using only key-points. To the naked eye, other features of objects such as colour, angularity, number of visible lines and the object silhouette can play a role in distinguishing between objects. This thesis will describe the latter two of these features and ways of extracting them from an image. They will be added to the image-histograms and used in the classification.

3.3.1 Distribution of Key-points

From the human perspective, an intuitive way of describing an object would be using its shape. However, while describing simple geometric shapes might be straightforward to implement, most objects, including the ones in Figure 1.1, have more complex forms. Describing the shape or the silhouette of an object can be impractical, especially when the image quality is not that well.
Another way of including object-shape information in the object-classification process is using the locations of the SURF key-points to approximate the area of the image that is occupied by the object. This only takes into account the "interesting" parts of the object, the parts where key-point were found in the first place. This is nevertheless a good way to describe the shape of the image, since as far as the algorithm is concerned, the object ends where the key-points end.
Using the locations of the key-point as data-points in $\mathbb{R}^2$, PCA ([11]) can be used to determine the two principal axes (eigenvectors) of the data-cloud. Projecting data-points on the eigenvectors $\vec{e}_1$ and $\vec{e}_2$ results in the distance of the points from the mean, in the respective principal direction. Using these distances, the standard deviations σ_1 and σ_2 can be calculated:

$$\sigma_i = \sqrt{\frac{1}{N_k}\sum_{k=1}^{N_k}(\vec{p}_k \cdot \vec{e}_i)^2} \tag{3.1}$$

Where $\vec{p}_k$ is the location of the k-th key-point and N_k is the number of key-points found in the image. Figure 3.5 shows the standard deviation ellipses, scaled with a factor 3. The ratio $\frac{\sigma_2}{\sigma_1}$ (in the figure the length of the shorter axis divided by the length of the longer axis) can be added to the histogram of the image and used in the classification.

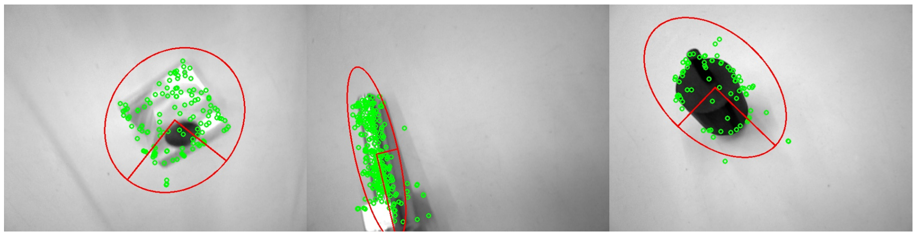


Figure 3.5: Standard deviation ellipses drawn over the key-point locations using 3σ for the ellipse dimensions.

This way of describing the shape of the image is preferable to using the occupied area of the key-points, because it's invariant to different object distances. It is however sensitive to 3D rotations of the object, causing oblong shapes to appear round and vice versa. The hypothesis to be investigated is that these changes, as an effect of 3D rotation of the object, are still distinct characteristics of the object and add enough valuable information to improve classification results.

3.3.2 Occupied Area

Like the σ-ratio described above, the area occupied by the cloud of key-points is sensitive to rotations of the objects and its distance to the camera. It can however still be a useful feature. When using a calibrated camera, the distance to the camera could be dealt with by multiplying the calculated areas by some factor. In the database used here however, we can assume the camera distance to be approximately the same for all images. Since the different objects have observably different areas of occupation, we will also use this feature in classification.

3.3.3 Number of Key-points

The number of SURF key-points can be of significance when classifying object types. It's a measure of the amount of details to be seen on the surface of the object visible to the camera. A small surface or a smooth surface with no holes, edges or other patterns will lead to a low number key-points. Although this number is also dependant on the sharpness and quality of the image, using the same camera, mostly from the same distance, will eliminate this dependence for a great part. Thus, the number of key-points will be added to the object features.

3.3.4 Lines

Another way to characterize an object is to analyse the patterns in the image that resemble lines. These can be patterns in the contour of the object, edges or even patterns printed/engraved on the object.
Finding lines in an image can be done using the Hough-transform ([6]). This method constructs a parameters-space using the parameters θ and r, the angle of a line and its perpendicular distance to the origin. After making this Hough-space discrete, all possible lines of are calculated and votes are counted for each time a pixel is on each line. Using these votes, the Hough-transform of an image can be made, which expresses the presence of lines in the image.
Prior to applying Hough-transform to the image, the edges of the object are calculated using the Canny algorithm in OpenCV ([4]).
For simplicity, only the number of lines are used as a feature, ignoring the line lengths and orientations. Figure 3.6 shows that this feature can be considered a characteristic of the object type. Objects with a round shape have few (if any) lines while oblong objects with straight edges have much more.

3.4 Classification

The eventual goal of the system is to classify a new image as one of the 10 object types. To do this, a histogram of key-points must be made for each new image. After extracting key-points and calculating the corresponding descriptors, the cluster centres calculated for the training-images must be used to assign the k histogram values to the new image. Also, the extra features must be appended to the histogram-vector to form the final data-vector $\vec{d}$ of the image.
From looking at Figure 3.4a a slight linear ordering of the histogram vectors can be

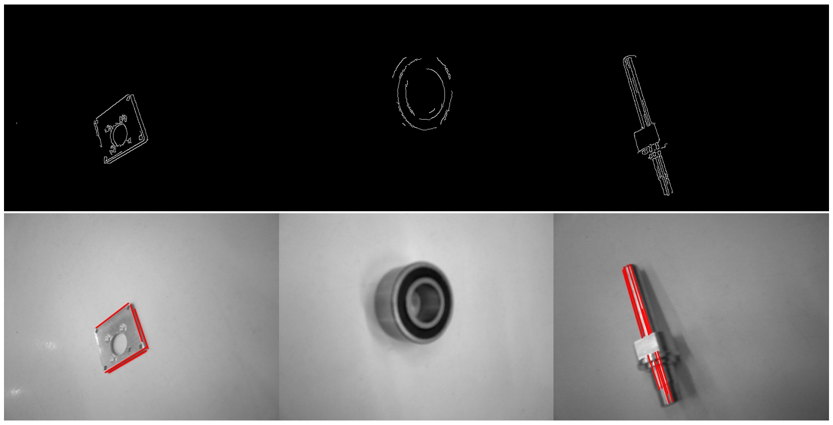


Figure 3.6: Edges of the objects, using a Canny edge detector (top) and the lines found with the Hough-transform drawn on the images (bottom).

seen for object-types, especially for types 0, 1 and 9. The subspaces of the data-space which are occupied by the different groups of histograms with different object-types are also to some extent separable, although partly overlapping.
In Section 2 the Support Vector Machines classification method was mentioned as one of the choices of Csurka *et al.*. [5, p. 7]. The WEKA implementation of this algorithm will be used first and the results of other algorithms will be evaluated using the SVM results.

3.4.1 k-Nearest Neighbours

To classify a new data-vector $\vec{d}$, the simple k-nearest neighbour algorithm will be used first. This algorithm takes the object-type labels of the k nearest data-vectors to $\vec{d}$ and the most frequently occurring label amongst these neighbours is assigned to $\vec{d}$. The Euclidean distance between data-points is used to find the nearest neighbours:

$$d(\vec{a}, \vec{b}) = \left\| \vec{a} - \vec{b} \right\|$$

3.4.2 Naive Bayes

Although k-nearest neighbour is a very efficient and simple algorithm, it only uses k other data-points (images) for classification while the information contained by the rest of the data is ignored. The Naive Bayes classifier ([8]) uses the conditional probability $P(c_j|\vec{d})$ of each class to classify a new image. The class-label (in our case the object type) is chosen which has the highest conditional probability:

$$c^* = \operatorname*{arg\,max}_{c_j \in C} P(c_j|\vec{d}) \tag{3.2}$$

Following Bayes' rule and seeing that $P(\vec{d})$ stays the same for each class, the conditional probability can be written as:

$$\begin{aligned} P(c_j|\vec{d}) =& \frac{P(c_j)P(\vec{d}|c_j)}{P(\vec{d})} \\ =& P(c_j)P(\vec{d}|c_j) \\ \approx& P(c_j)\prod_{i=1}^{d} P(f_i|c_j)^{d_i}\eta \end{aligned}$$

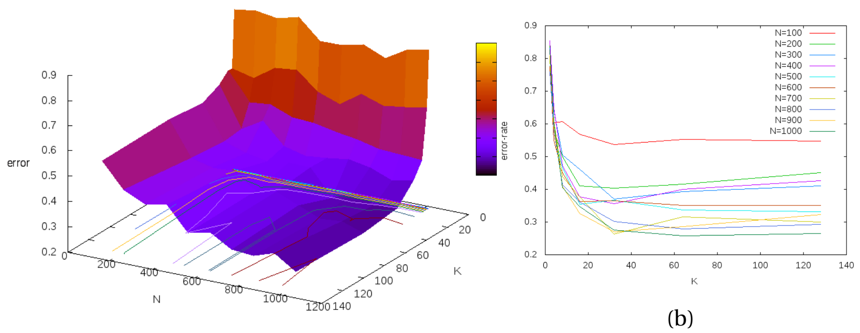
With the number of dimensions of $\vec{d}$ (the number of descriptor clusters plus the 4 extra features) denoted as d and f_i being the i-th dimension (feature) of the dataspace. Using Laplace smoothing, the conditional probability of a feature, given a class, is defined in by Csurka *et al.* in [5, p. 7].
With the Naive Bayes classifier, the assumption is made that each value d_i is statistically independent of the other values of the $\vec{d}$ vector ([8, p. 2]). However, in the case of the key-point histograms, the sum $\sum_{i=1}^{d} \vec{d}_i$ is the total number of key-points found in an image. Although the distribution of the sum amongst d_i depends on the cluster-centres, the total stays constant, which means that if one element changes, the other elements have to compensate. Thus, each value in $\vec{d}$ is clearly dependent of the others. Despite this weakness the Naive Bayes classifier will be used for it's efficiency.

4 Experiments and Results

In the previous Section 3 some ideas were presented for the improvement and customization of the Bag of Key-points method, applied to the problem of this study. In this section experiments are described to investigate the effect of these improvements. The software used for these experiments is mostly written in C++ and makes extensive use of the OpenCV library. For classification, the algorithms implemented in the WEKA Data Mining Software are used in a Java program.

4.1 Significance of k, Experiments

When performing the clustering-step of the Bag of Key-points method it is interesting to look at the effect of k on the eventual classification. Using different dataset-sizes and different values of k, the classification error-rate, using k-nearest neighbour classification ([10]), is illustrated in Figure 4.1.



(b)

(a)

Figure 4.1: Classification results using k-nearest neighbour and different dataset sizes/k-values

The value of k is incremented exponentially, using powers of 2. As the figure shows, the right values of k doesn't depend very much from the dataset-size. It's interesting to see that there is a minimum in each line of Figure 4.1b. This minimum is somewhere around $k = 50$ for our dataset of 1029 images.
It might seem intuitive that the error-rate of the classification is proportional to the result of the clustering process. However, if we take the compactness of the formed clusters as a measure to evaluate the clustering result, it can be shown that this is not the case.

$$C_D(k) = \sum_{i=1}^{N_{descr}} \left\| D_i - C_{labels_i} \right\|^2 \tag{4.1}$$

In Eq. 4.1 the compactness of the clusters is defined, with D being the 130000×64 matrix containing the descriptors. If the value of k is taken to be 130000, the distances $\left\| D_i - C_{labels_i} \right\|^2$ of the data-points to their cluster-centres become 0. However, most of the lines in Figure 4.1b stop declining at some point and don't seem to be approaching 0 for higher value of k.

4.2 Extra Features, Experiments

By adding the extra features of 3.3 to the dataset, some extra variance is added. In Figure 4.2 the data-points of all images, created with the different features are shown. The data-clouds of Figures 4.2b and 4.2c look almost identical. This means that most of the variance in the data with the combined features is contributed by the four extra features. Also, most of the clouds of data, ordered in color by object type, look separable.

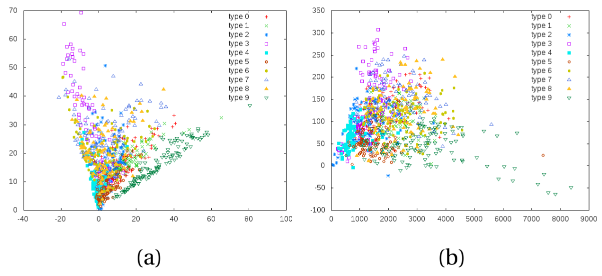


Figure 4.2: From left to right: the histogram data, the extra features data, and the combination.

Using these extra features, experiments were performed and the resulting error-rates can be seen in the table below.

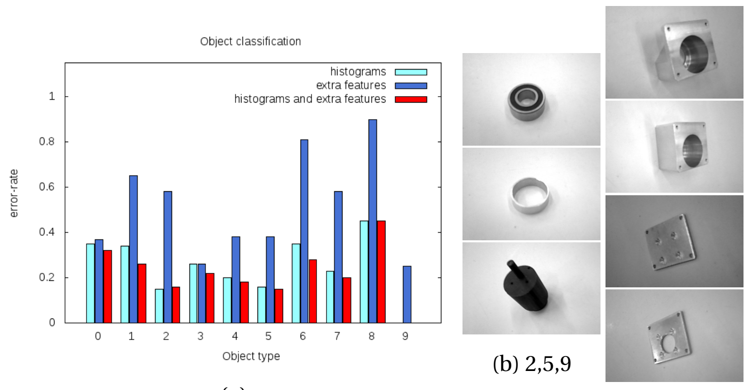
	histograms	only extra features	histograms and extra features
error-rate	0.247446	0.502294	0.216262

The results were acquired with 10-fold cross validation using 1029 images. As a classifier the 1-nearest neighbour algorithm provided in WEKA was used. The extra features improve the result slightly but consistently through the cross-validation. Using only the four extra features doesn't provide a usable classification result on its own, but the error-rate for 10 object types is still smaller than a 0.9 error-rate expected by chance, see Figure 4.3a.
The confusion matrices of the classification results are included in Appendix A. Note

that a different number of instances is used for every object type. The number of images per object type can be seen in Figure A.1.
As could be expected, when using only the key-point histograms (A.2), object types 0, 1 and 6, 7, 8 are often confused with each other. When using the extra features (width/height ratio, area, number key-points and number of lines) even without using key-point histograms the number of false negatives for object type 3, the oblong object, are the same as when using the key-point histograms (A.3). Types 0, 1, 2 and 6, 7, 8 still have lots of false negatives, because the values of the extra features, especially the width/height ratio, are very similar (Figure 1.1).
In the table below the error rates are shown of classification using only one of the extra features, without the key-point histograms. The best features turned out to be the number of lines that can be fitted on the object and the occupied area of the key-points.

feature	key-point distribution	area	nr. of key-points	nr. of lines
error-rate	0.83	0.79	0.83	0.73

Table A.4 and Figure 4.3 show the results of the extra features being combined with the key-point histograms. Performance increases slightly, but the same objects remain hardest to distinguish. Also, the great amount of extra variance brought in the data by the extra features is not justified with increase in performance. Despite this variance, most of the useful information is still represented in the remaining dimensions of the data (the 2D plots are projections of the higher dimensional data on its two main components).



(a) (b) 2,5,9 (c) 0,1,6,8

Figure 4.3: The error-rate using histograms only, extra features only, and the combination (Fig. 4.3a). The object types with lowest and highest error-rates, using only histograms or the combination, are shown in Fig. 4.3b and 4.3c respectively.

The RoCKIn@Work objects are also divided into categories and it's also relevant to look at the classification results using only the object-categories as labels. The error rates of assigning of the right category, using all features, is shown in the table below.

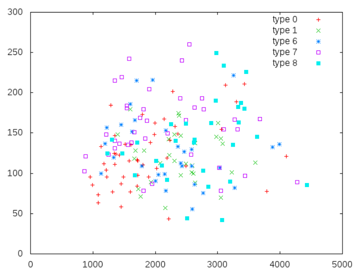
category	I (6,7,8)	II (0,1)	III (2,3,9)	IV (4,5)	**total**
error-rate	0.11	0.14	0.10	0.08	**0.11**

Although a different dataset is used here than in the RoCKIn@Work competition, the objects are the same and it's interesting to compare the results with those of the 2014 competition[6]. The winning team (*b-it-bots*) had a 80% object category accuracy (89% here), and a 60% object type accuracy (78% here).

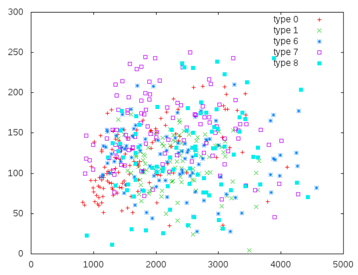
4.3 Classification, Experiments

With the classification algorithms described in 3.4 experiments were performed on the dataset of 1029 images. Using 10-fold cross validation, the error-rates are shown in Figure 4.4c.
The Naive Bayes classifier was outperformed by the other two. This might be caused by the incorrect assumption of statistical independence. The best results were acquired using the k-nearest neighbour algorithm and 1029 images, although the difference with the SVM error-rate is small. The data-points of the three most similar objects using 350 and 1029 images are shown in Figures 4.4a and 4.4b respectively. The good performance of the SVM classifier on the smaller dataset can be understood by looking at the figures. In the smaller dataset, linearly separating a part of the data-cloud of one object from all other points looks easier than in the larger dataset, where the multiple object-types overlap even more.

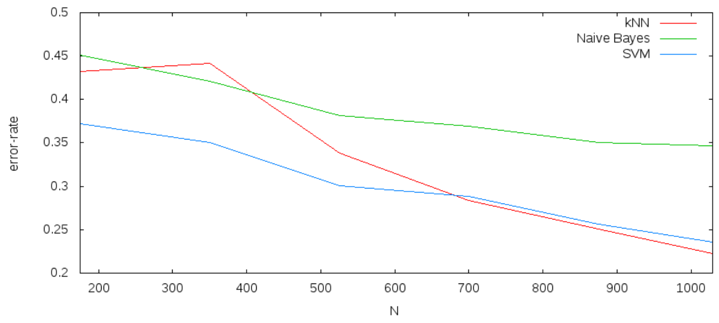
[6]http://rockinrobotchallenge.eu/work2014_scores.pdf



(a)



(b)



(c)

Figure 4.4: Fig. 4.4a and 4.4b: object types 0, 1, and 6, 7, 8 (highest error-rates), with with different dataset-sizes, Fig. 4.4c: error-rates per classifier.

5 Evaluation and Conclusion

Three aspects of the Bag of Key-points methods have been investigated in this thesis. In Section 4.1 the value of k was determined by choosing a minimum for the error-rate. Although this is exactly what we need, performing multiple runs of clustering and classification with the whole dataset of 130000 descriptors is very time-consuming[7]. It would be more efficient to be able to predict the error-rate caused by a k value using a less time-consuming process. It's interesting to see that unlike the clustering result, the classification result has an optimum for some k. Using a clustering algorithm that determines k might be a solution.

In Section 4.2 the histograms of key-points combined with generic features of the objects showed to have the best classification results, although the contribution of the histograms was much bigger. As an extension of the methods of this study, a combination of other sorts of key-point detectors and generic object features might lead to even better results.

Choosing a classifier is not arbitrary, as shown in Section 4.3. Three classifiers were examined in this study, but there are of course many more. Using distinct object-

[7] 30+ minutes on a 2.7 Quad-Core machine with 8GB RAM

types as labels seems to exclude linear classifiers, but seeing the linear ordering of the data of some of the object-types in Figure 3.4a, linear classifiers might be worth investigating after all.
To conclude this study, it can be said that the Bag of Key-points methods leads to good results in the RoCKIn@Work Object Recognition challenge. An error-rate of 22% was reached with cross-validation of the 1-nearest neighbour classifier. Using only the object-categories, an error-rate of 11% was reached. The experiments were done using a dataset with only the objects in the images. The official datasets[8] of the RoCKIn@Work have not been tried out and it's a logical next step to experiment with this data. It can be concluded however, from looking at the results of this study and the application of its methods on the @Work competition, that using a big dataset enhances the result. Data is not scarce in most industrial environments, and making grateful use of this fact can improve the performance.

[8] http://thewiki.rockinrobotchallenge.eu/index.php?title=Datasets

References

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[2] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[3] Bastian Broecker, Daniel Claes, Joscha Fossel, and Karl Tuyls. Winning the robocup@ work 2014 competition: The smartlab approach. In *RoboCup 2014: Robot World Cup XVIII*, pages 142–154. Springer, 2015.

[4] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[5] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[6] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[7] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[8] David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Machine learning: ECML-98*, pages 4–15. Springer, 1998.

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

[10] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[11] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[12] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

Appendices

A Confusion Matrices

Table A.1: Number of images per object type.

Object type	0	1	2	3	4	5	6	7	8	9	**total**
Nr. of Images	122	101	101	105	92	102	95	107	86	118	**1029**

Table A.2: Key-point histograms only, using k-nearest neighbours

Classified as →	0	1	2	3	4	5	6	7	8	9	**error**
0	**79**	21	1	6	1	7	4	0	3	0	**0.35**
1	20	**67**	1	1	3	3	0	0	1	5	**0.34**
2	0	0	**86**	2	7	3	0	0	3	0	**0.15**
3	6	4	4	**78**	4	8	0	0	1	0	**0.26**
4	1	2	3	1	**72**	10	0	0	1	0	**0.20**
5	1	1	4	2	10	**82**	0	0	0	2	**0.16**
6	3	2	1	2	2	0	**62**	10	13	0	**0.35**
7	0	1	1	4	3	2	6	**82**	8	0	**0.23**
8	3	2	3	4	3	1	13	9	**47**	1	**0.45**
9	0	0	0	0	0	0	0	0	0	**118**	**0**

Table A.3: Four extra features only, using k-nearest neighbours

Classified as →	0	1	2	3	4	5	6	7	8	9	**error**
0	**77**	16	9	2	1	3	4	6	1	3	**0.37**
1	33	**35**	3	0	1	6	9	8	3	3	**0.65**
2	14	7	**42**	11	17	6	0	0	3	1	**0.58**
3	4	1	5	**78**	9	5	2	0	1	0	**0.26**
4	0	2	16	6	**56**	9	0	0	1	0	**0.38**
5	7	4	14	4	9	**63**	0	0	0	1	**0.38**
6	19	25	0	2	0	0	**18**	21	6	4	**0.81**
7	22	13	0	0	0	1	22	**45**	4	0	**0.58**
8	23	17	5	2	1	2	14	7	**9**	6	**0.90**
9	7	9	1	1	0	8	2	0	2	**88**	**0.25**

Table A.4: Histograms and extra features, using k-nearest neighbours

Classified as →	0	1	2	3	4	5	6	7	8	9	**error**
0	**83**	19	1	3	1	7	3	1	4	0	**0.32**
1	15	**75**	1	1	1	2	0	0	2	4	**0.26**
2	0	0	**85**	1	9	4	0	1	0	1	**0.16**
3	5	1	4	**82**	6	7	0	0	0	0	**0.22**
4	1	2	4	0	**74**	9	0	0	0	0	**0.18**
5	1	1	2	2	8	**87**	0	0	0	1	**0.15**
6	3	1	2	1	1	1	**68**	7	11	0	**0.28**
7	0	0	1	1	4	2	6	**86**	7	0	**0.20**
8	5	3	3	2	1	1	11	12	**47**	1	**0.45**
9	0	0	0	0	0	0	0	0	0	**118**	**0**

Table A.5: Histograms and extra features, using Naive Bayes

Classified as →	0	1	2	3	4	5	6	7	8	9	**error**
0	**80**	31	0	0	1	3	1	2	4	0	**0.34**
1	28	**61**	3	0	1	5	1	1	0	1	**0.40**
2	3	0	**75**	1	5	11	0	0	1	5	**0.26**
3	12	0	1	**54**	8	18	1	1	9	1	**0.49**
4	2	2	3	0	**54**	24	3	0	1	1	**0.40**
5	3	0	6	0	7	**81**	0	0	2	3	**0.21**
6	2	5	0	1	0	2	**50**	16	19	0	**0.47**
7	0	4	0	0	0	0	19	**65**	19	0	**0.39**
8	4	3	2	0	3	1	18	16	**35**	4	**0.59**
9	1	0	0	0	0	0	0	0	1	**116**	**0.02**

Table A.6: Histograms and extra features, using Support Vector Machines

Classified as →	0	1	2	3	4	5	6	7	8	9	**error**
0	**95**	16	0	4	2	3	1	0	1	0	**0.22**
1	36	**54**	0	2	1	5	1	0	0	2	**0.47**
2	0	0	**88**	1	4	7	0	0	0	1	**0.13**
3	4	1	1	**89**	3	6	0	1	0	0	**0.21**
4	2	0	0	0	**71**	15	0	2	0	0	**0.21**
5	1	1	2	0	6	**92**	0	0	0	0	**0.10**
6	0	3	0	0	1	0	**63**	15	13	0	**0.34**
7	1	1	0	2	2	0	14	**74**	13	0	**0.31**
8	2	0	2	2	6	1	14	17	**41**	1	**0.52**
9	0	0	0	0	0	0	0	0	0	**118**	**0**