Vision based indoor positioning in a retail environment

Jara Uitto

Helsinki May 31, 2011
Master’s thesis
UNIVERSITY OF HELSINKI
Department of Computer Science
Modern smart phones often come with a significant amount of computational power and an integrated digital camera making them an ideal platform for intelligent assistants. This work is restricted to retail environments, where users could be provided with for example navigational instructions to desired products or information about special offers within their close proximity. This kind of applications usually require information about the user’s current location in the domain environment, which in our case corresponds to a retail store. We propose a vision based positioning approach that recognizes products the user’s mobile phone’s camera is currently pointing at. The products are related to locations within the store, which enables us to locate the user by pointing the mobile phone’s camera to a group of products.

The first step of our method is to extract meaningful features from digital images. We use the Scale-Invariant Feature Transform SIFT algorithm, which extracts features that are highly distinctive in the sense that they can be correctly matched against a large database of features from many images. We collect a comprehensive set of images from all meaningful locations within our domain and extract the SIFT features from each of these images. As the SIFT features are of high dimensionality and thus comparing individual features is infeasible, we apply the Bags of Keypoints method which creates a generic representation, visual category, from all features extracted from images taken from a specific location. A category for an unseen image can be deduced by extracting the corresponding SIFT features and by choosing the category that best fits the extracted features.

We have applied the proposed method within a Finnish supermarket. We consider grocery shelves as categories which is a sufficient level of accuracy to help users navigate or to provide useful information about nearby products. We achieve a 40% accuracy which is quite low for commercial applications while significantly outperforming the random guess baseline. Our results suggest that the accuracy of the classification could be increased with a deeper analysis on the domain and by combining existing positioning methods with ours.

ACM Computing Classification System (CCS):
I.2.10 [Vision and Scene Understanding],
G.3 [Probability and Statistics]
Contents

1 Introduction and related work 1

2 Background information 5
  2.1 Convolution ........................................ 5
  2.2 Scale ........................................... 7

3 Scale-Invariant Feature Transform 8
  3.1 Scale-space representation ....................... 8
    3.1.1 Scale-space pyramid ......................... 9
    3.1.2 Detection of local extrema ................... 11
  3.2 Keypoint localization .......................... 13
    3.2.1 Rejecting points with low contrast .......... 13
    3.2.2 Rejecting edge responses ................... 15
  3.3 Describing local image regions ................ 17
    3.3.1 Finding a consistent local orientation ..... 17
    3.3.2 Local image descriptor ..................... 19
  3.4 Implementation details ........................ 22
    3.4.1 Initialization ............................ 22
    3.4.2 Examples .................................. 23

4 Visual categorization 27
  4.1 Bags of Keypoints ................................ 28
  4.2 Removing noise ................................ 33
  4.3 Extending the local features with color information .... 37

5 Results 40
  5.1 Plain Bags of Keypoints ....................... 42
  5.2 Extensions .................................... 43
6 Discussion and summary 48

References 49
1 Introduction and related work

Many modern mobile phones have integrated digital cameras and a significant amount of computational power. These properties can be used to provide mobile phone users with digital information about an object or a scenery that they are currently looking at. In augmented reality (AR) applications, a real world scenery is integrated with digital information when viewed through a digital display. The early prototypes of AR required custom built equipment [FMHW97] but the increase in multimedia and computing capabilities of mobile phones have enabled providing augmented reality solutions directly on mobile phones. One of the most fundamental problem in the field of AR is recognizing what the camera is pointing at. The most common approach for this problem is to attach easily recognizable artificial markers to meaningful objects. For example ARToolKitPlus [WS07], an ARToolKit [KB99] extension optimized for mobile devices, recognizes the pose and the identity of objects of interest with the help of fiducial black-and-white markers.

Shopping in a grocery store is one of the most common activities that consumers have to perform in their everyday life. Mobile augmented reality can provide a practical and intuitive interface for accessing digital information during the shopping event. For example the packages of grocery products can only contain very limited amount information with often impractical font sizes. In augmented reality, the packages can be digitally augmented with practically unlimited information in a readable context. This would be especially useful to users with allergies, who could be given a warning about a harmful ingredient in a product by simply pointing a mobile phone’s camera at the product. Unfortunately, using artificial markers is impossible in retail environments as marking individual products with unique markers is time-consuming, costly and impractical due to aesthetic reasons. An alternative to marker-based solutions is object recognition, which indicates determining the identity of a perceived object. An example of an object recognition approach is the museum guide system by Bruns and Bimber [BB08], which recognizes museum exhibits using color features. Unfortunately the object recognition approach also proves difficult in a retail environment as there are typically a vast amount of different products and as there are often multiple instances of the same object in a single image.

Mobile augmented reality can also be used to provide users with navigational instructions to desired products and information about special offers near their current location. Hile et. al. [HGL+09] implemented a similar idea in a pedestrian navigation system that augments real world images with navigational arrows that adapt
to the current environment to provide natural paths. Typically, navigation requires knowledge of the user’s current location. A common way to gain location information is to use GPS, which estimates distances to multiple satellites to deduce the user’s current position [VP10]. However, GPS solutions do not work indoors and the accuracy of current indoor solutions, such as WiFi based positioning [HGL+09], provide limited accuracy. In this work, we identify the user’s location in a retail environment by relating groups of products to locations. The location of the user can then be deduced by recognizing a group of products that the user’s mobile phone is currently pointing at.

We avoid using artificial markers and therefore the first step is to find some meaningful characteristics from digital image signals that are distinctive enough to distinguish groups of products from each other [SMB00]. This problem has been widely researched and there are lots of possible solutions for extracting such characteristics, for example blob detection [BTG08] and edge detection [MH80]. The general differences between different approaches for characterizing images are computational efficiency and the distinctiveness of the characteristics. In this work, we have chosen to use the Scale-Invariant Feature Transform SIFT algorithm as it is well-known and it produces characteristics with extremely high distinctiveness [Low04]. Furthermore there are variants of the SIFT algorithm that speed up the computations with the cost of distinctiveness, such as Fast Approximated SIFT [GGB06] that has been designed for mobile phones where computational efficiency is an issue.

SIFT is a blob detection algorithm, which indicates that it searches for areas in an image that are either brighter or darker than surrounding pixels [DB87]. The intensity changes within each blob are transformed into a high dimensional real-valued vector that is referred to as a feature of the corresponding image. The usual way to detect blobs in natural images is to gradually filter out details while preserving meaningful structures. In the SIFT algorithm this corresponds to smoothing out the highest frequencies of the image signal, which removes image noise and saves only the meaningful structures of images that are possible to be detected from different perspectives and from different viewing distances. This idea can be understood from the following example: consider an image of a tree. From a high resolution image, the viewer can see small details such as leaves. When leaving out the finest structures, which imitates the idea of looking at the same view from a longer distance, the leaves become impossible to recognize while bigger structures such as the branches of the tree can still be seen. Taking the smoothing process further removes the branches and only the tree trunk remains recognizable. The SIFT algorithm utilizes
this idea and stores the characteristics of the image in multiple levels of scale, which correspond to the level of smoothing [Low04].

After capturing the characteristics of images as real-valued SIFT feature vectors, our next task is to recognize different images from similar views with the help of these features. As the amount of characteristic features extracted from images by the SIFT algorithm is relatively large, brute force matching approaches for comparing features from different images are infeasible. Therefore, we fix a group of locations in the grocery store and transform all the images from the corresponding location into a generic representation. As the distinctiveness of the image characteristics is limited, the level of accuracy is dependent on the size of the store. For example, in a small store with only a handful of different product categories, we could be able to distinguish each of these categories. On the other hand in a large store with for example more than 100 product categories, we have to restrict the recognition to a coarser level. Our domain is a Finnish medium size supermarket that contains approximately 45000 products which makes the recognition of individual product groups infeasible. Therefore we perform the recognition on shelf level which implies that the set of different possible locations is bounded by the number of different shelves. After throughout modeling of the locations within the store, any new visual data can be related to any of those locations providing the user knowledge of his or her current location. The vision based indoor positioning problem in a retail environment has also been studied by Yang Li in his Master’s thesis [Li10]. While having similar motivations and domain to ours, he uses different methods to capture image characteristics and to identify locations.

The task of deciding into which category an image belongs to is a typical instance of widely studied machine learning problem of classification [Kot07]. In binary classification we have a predictor, an example dataset with samples whose class is known and two possible class values. With the information given by the example data set, the predictor is supposed to learn a model that can be used to predict the class label of an unlabeled sample. Classification task with more than two possibilities is referred to as multi-class classification. As making decisions based on incomplete knowledge of the domain is a typical setting in machine learning, there exists a wide range of different approaches for classification. The fundamental choice is between classification accuracy and computational efficiency. Two widely used approaches are Support Vector machines [Vap98], known to produce state-of-the-art accuracy, and the Naive Bayes classifier [Lew98], known for its computational efficiency and simplicity. As high accuracy usually demands complex methods in addition to high
computational complexity, we have chosen to restrict this work to the Naive Bayes classifier. The basic idea of the Naive Bayes classifier is to assume that all the features of the samples are mutually independent and to choose the most probable class [Lew98]. Assuming that the features are independent does not generally hold and therefore it is called a naive assumption. The naive Bayes classifier has been shown to produce accurate results in various domains, such as chess end games and lymphography [LWT92], regardless of the naive assumption [DP97].

A real world example of applying classification to computer vision is the Bags of Keypoints visual categorization method [CDF+04]. In the Bags of Keypoints method a visual vocabulary is created with the k-means clustering algorithm [TK04, Chapter 14.5.1] to relate unseen SIFT features to previously seen ones. A bag of keypoints representation of a visual class corresponds to the number of occurrences of visual words in the corresponding class. A classifier is trained with the Bags of Keypoints representations extracted from training images. A class label for an unlabeled image can be predicted by extracting the bag of keypoints representation from the unlabeled image and by choosing the image class which best fits the extracted representation.

The thesis is structured as follows: First in Section 2, we briefly introduce important concepts used throughout the thesis. In Section 3, we introduce the Scale-Invariant Feature Transform SIFT algorithm, which can be used as a tool to gather meaningful information from digital images. Section 4 introduces the Bags of Keypoints method for visual categorization that is applied to vision based indoor positioning in a real world grocery store and the results are evaluated in Section 5. Finally, the results are discussed and the thesis is summarized in Section 6.
2 Background information

The most common way to obtain natural images is to use a digital camera. Natural images are usually stored as a matrix of pixels where each pixel is represented as an integer that contains information about the color and the transparency of the pixel. One way to represent these properties is to use a 32-bit integer where the first 8 bits represent the transparency and the remaining 24 bits correspond to red, green and blue intensities respectively. In this work, we restrict ourselves to grayscale images where we have a single 8 bit channel representing the gray intensity of a pixel. A color image can be transformed into a grayscale image by using a transformation that assigns each pixel a gray value as weighted average of the three color channels. A widely used choice for the weights is $(0.3, 0.59, 0.11)$ for red, green and blue respectively [Hen06]. For simplicity and without loss of generality, it is assumed throughout the thesis that gray pixel values are within the interval $[0, 1]$.

Pixel values themselves do not give much information about the actual contents of an image. In the following section, we introduce some of the most common methods for finding characteristics of natural images. One intuitive approach is edge detection [MH80], which detects outlines of different objects in the image. Another way to characterize images is to form a color histogram, which contains information about the amounts of different colors throughout the image.

The most relevant method from the viewpoint of this work is blob detection. A blob refers to an area in an image that is either brighter or darker than its surroundings. Blobs can be used for example to find objects or parts of an object for further processing. One of the problems with blob detection is separating valid blobs from the background of an image. One of the main topics of this thesis is to introduce a widely used blob detection algorithm. To help the reader to easily understand the details, we begin by explaining some of the fundamental concepts used in the algorithm.

2.1 Convolution

Convolution [MW98, Chapter 51] is one of the primitive methods used in computer vision. The basic idea of convolution is to apply a window of finite size and shape on each pixel in an image. Applying the window on a pixel results in a weighted average of the local neighborhood of the pixel where the weights are defined by a filter. The window together with the weights is called a convolution kernel. Given
an image $A$ and a filter $f$, convolving $A$ with $f$ produces a new image $B$, where
the corresponding convolution kernel $[f]$ has been applied to the neighborhood of
each pixel in the image $A$. The size of the neighborhood and the distribution of the
weights are fixed by the definition of the corresponding filter. A simple example of
a convolution kernel is

$$[f_{\text{simple}}] = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix},$$

(1)

which gives each pixel a value according to the local neighborhood values while
giving more emphasis to the nearby pixels. To avoid boundary effects, pixels outside
the image borders are assumed to have the same value as the nearest actual pixel,
which prevents new structures from appearing in the images. There are also other
methods to handle boundary effects. Equation 2 gives an example of applying the
simple convolution kernel given in Equation 1

$$\begin{bmatrix} 1 & 2 & 2 & 2 \\ 0 & 5 & 2 & 0 \\ 1 & 2 & 0 & 4 \\ 3 & 2 & 3 & 2 \end{bmatrix} \ast [f_{\text{simple}}] = \begin{bmatrix} 5.00 & 8.25 & 8.25 & 6.50 \\ 5.00 & 9.00 & 8.00 & 6.00 \\ 6.50 & 8.00 & 7.75 & 8.75 \\ 9.50 & 8.75 & 9.00 & 9.75 \end{bmatrix},$$

(2)

where $\ast$ corresponds to the convolution operator.

A popular choice for the kernel is the two dimensional Gaussian filter where the
weights are from the Gaussian distribution given by

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

A common choice for a window radius $r$ for finite approximation of the Gaussian filter
is $r = \lceil 3\sigma \rceil$ or $r = \lceil 5\sigma \rceil$ [MW98, Chapter 51], which provides accurate approximation
of the infinite filter and therefore we have chosen to use $r = \lceil 3\sigma \rceil$ throughout this
work. To avoid loss of information, the kernel is normalized so that the sum of the
kernel weights equals to 1. A well-known interesting property of the two dimensional
Gaussian filter is *separability*, which implies that the convolution operation can be
separated into two one dimensional operations [Lin94], i.e.,

$$G(x, y, \sigma) = G_{1D}(x, \sigma) \ast G_{1D}(y, \sigma),$$

where $G_{1D}(x, \sigma)$ is given by

$$G_{1D}(x, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}}.$$
and $G_1(y, \sigma)$ is defined analogously. In Section 3 the Gaussian kernel is applied to natural images yielding a representation, that allows us to extract characterizing properties.

## 2.2 Scale

The scale of an image is usually considered as the spatial distance of pixels in an image. For example, if the original input image has scale 1, value 2 for scale indicates that the distance between any two pixels has halved. In this work, we define scale using Gaussian convolution, which indicates that changes in scale correspond to performing convolution operations on the image with a Gaussian filter. As the Gaussian convolution smoothens the image, fine scale information becomes suppressed with increasing width of the smoothing kernels. This can be thought of as looking at the same view from increasing distances, which makes smaller details impossible to recognize.

Image *structures* correspond to sets of pixels in the image such as objects. The *characteristic length* of a structure indicates the longest distance between any two pixels within the structure. Convolving an image with a kernel of width $\sigma$ suppresses most structures in the signal whose characteristic length is less than $\sigma$ [Lin94]. This indicates that convolving an image with an initial scale equal to $k$ with a Gaussian filter of width $\sigma$ results in an image with scale equal to $\sqrt{k^2 + \sigma^2}$. Therefore, convolving an image with scale 1 with a Gaussian kernel with $\sigma = \sqrt{3}$ results in an image with scale

$$\sqrt{1^2 + \sqrt{3}^2} = 2.$$ 

Even though scale defined by Gaussian kernel is somewhat different from the definition of scale as pixel distances, we will later see that the convolution operation gives us access to information that is invariant to scale in also in terms of the pixel distances.
3 Scale-Invariant Feature Transform

The Scale-Invariant Feature Transform (SIFT) algorithm [Low04, BL02] is a popular approach for blob detection. This algorithm finds locations or keypoints in the input image that can be repeatably detected when the same view is looked at from different viewpoints or in different settings. Different viewpoints correspond to looking at the same view from different distances or angles. For example, when processing an image of a tree, we might want to identify either the whole tree or just one particular branch of the tree. Different settings, on the other hand, correspond to variations in background, illumination conditions and whether the object to be recognized is fully visible or partially occluded. The keypoints are used to characterize images in a way that enables us to perform automatic detection of similar characteristics in unlabeled input images.

3.1 Scale-space representation

The first step of the SIFT algorithm is to transform the input image into a scale-space representation. This representation is used to locate points in the input image that are invariant to linear transformations, scale, illumination and image noise, and that are partially invariant to changes in three-dimensional viewpoint. The stability of these points is verified by comparing their contrast against neighboring points and an additional verification procedure tests whether these points are located on edges of objects in the image. Points that have low contrast or are edge responses are considered unstable and rejected. The remaining points are assigned a local orientation with respect to local intensity gradients which provides means for achieving rotation invariance. Finally the local intensity gradients are transformed into a representation that tolerates significant levels of shape distortion and changes in illumination. The representation is called the local image descriptor of the corresponding point of interest. After this transformation, these points of interest combined with the local image descriptors are referred to as features of the image.

The scale-space of an image is a continuous function $L(x, y, \sigma)$ that defines a value for point $(x, y)$ at scale $\sigma$. This representation is constructed by incrementally smoothing finer details away from the image using the convolution method described in Section 2.1. Formally, the scale-space of an image is given by

$$L(x, y, \sigma) = G(x, y, \sigma) \ast I(x, y),$$
where \( * \) is the convolution operator and \( G \) is the two dimensional Gaussian filter of width \( \sigma \). In our case the input signals are digital images which are always discrete and therefore we have to settle for an approximation of the continuous function. The scale-space is used to detect scale-invariant locations by searching for extrema within all scales. As the amount of possible scales is infinite, it is impossible to extract all scales, which is why only a fixed amount of different scales are considered. Representations where natural input images are embedded to multiple scales are referred to as scale-space representations.

Koenderink has showed that the Gaussian filter is the optimal choice for the smoothing filter [Koe84]. This is due to several properties that the smoothing kernel should satisfy. First, a smoothing kernel should satisfy *causality*, which indicates that the smoothing must not remove peaks from the input signal. Koenderink has also shown that all spatial positions and scale levels should be treated in similar manner and that all of the previous requirements can be encapsulated to a requirement that the scale-space function \( L \) must satisfy the heat diffusion equation

\[
\frac{\partial}{\partial \sigma^2} L = \frac{1}{2} \nabla^2 L, \tag{3}
\]

where

\[
\nabla^2 L = \frac{\partial}{\partial x^2} L + \frac{\partial}{\partial y^2} L
\]

is the Laplace operator for \( x \) and \( y \) dimensions. Diffusion equations are parabolic second order partial differential equations and it can be shown that the solution to the equation corresponds to the Green’s function of the equation when the domain of the diffusion equation is assumed infinite [GL96, Chapter 5-3]. It has been shown by Koenderink that the Gaussian kernel function is the Green’s function for Equation 3, which implies that Gaussian kernel is indeed an unique choice for a scale-space kernel function.

### 3.1.1 Scale-space pyramid

Creating the scale-space requires heavy computation per each pixel which easily makes straightforward implementations infeasible. A common way to optimize the efficiency of generating the scale-space representation is to use decreasing sample rates [Lin94] when creating the scale-space, where sample rate indicates the resolution of the image with respect to the initial resolution. It has been shown that, when the blurring ratio of an image doubles, the sampling rate can be halved by taking every other pixel from each row and column and combining them into a new image.
Figure 1: The image is taken from [Low04]. The natural input image is first incrementally convolved with Gaussian kernels to produce the first octave of the image pyramid. The widths for the smoothing kernels are chosen so that the blurring ratio has doubled in the second image from the top of the stack, labeled a. The image with the doubled blurring ratio is then downsampled and used as the first image of the second octave, labeled b, and the process is repeated. After creating the pyramid, adjacent images are used to produce the Difference-of-Gaussian images shown on the right.
without loss of information [ESM+06]. The halving of the sample rate has an effect of doubling the widths of the smoothing kernels with respect to the original sample rate. The doubling of the scale parameter is referred to as an *octave* of scale-space.

The downsampling results in a pyramid structure as it rapidly decreases the width of the image. To minimize unnecessary computation, the downsampling should be done as soon as possible. Therefore, the widths of the smoothing kernels should be chosen so that one of the blurred images contains exactly two times the blurring ratio of the initial image of the corresponding pyramid level. That image is then downsampled without loss of information and the next level of the pyramid is generated similarly as the previous one using the downsampled image [Low04]. The construction of the image pyramid with 4 scale samples is visualized in Figure 1. The downsampling with a factor of two is continued until the height or width of the image is reduced below some threshold $p$. As there is no universally correct way for choosing the threshold, we have chosen to use $p = 128$. The reason for choosing this threshold is due to an observation that lower resolution levels yield only few, if any, keypoints.

Also other methods have been proposed to optimize the performance of scale-space construction. An example of these is the Quad-tree representation [Lin94]. The basic idea of Quad-tree is to divide the image into regions based on some measure of variation in gray values. Optimally, the image can be divided into small amount of regions which have relatively uniform gray values and data reduction can be achieved. In the worst case, each pixel corresponds to a leaf in the tree yielding no data reduction. The main advantage of the pyramid representation is that the image size, and hence the computation performed, always decreases exponentially with the scale level.

### 3.1.2 Detection of local extrema

To acquire transformation and scale-invariant locations, the next step is to search for local extrema within the scale-space representation. Mikolajczyk has showed that peaks in the scale-normalized Laplacian of Gaussian provide us with scale-invariant and robust features [Mik02]. As convolving with the Gaussian kernel replaces pixel values with local averages, each convolution decreases the overall amplitude of the whole image signal and therefore the image signal has to be normalized after convolution. The SIFT algorithm approximates the Laplacian with Difference-of-Gaussians (DoG) function $D(x, y, \sigma)$ introduced by Marr et. al. [MH80]. The Difference-of-Gaussians can be efficiently calculated from two smoothed images that are separated
Figure 2: The image is taken from [Low04]. The extremum found in the scale-space representation is marked with $x$. This value is compared to every pixel in its scale-space neighborhood marked with turquoise circles.

In scale by a constant factor $k$:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$

$$= L(x, y, k\sigma) - L(x, y, \sigma),$$

which is a simple image subtraction (subtract values of corresponding pixels from each other). Lowe has showed that

$$\frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k - 1} \approx \sigma^2 \nabla^2 G$$

and therefore, to apply the scale normalization we divide all the values in the Difference-of-Gaussians image with $k - 1 \leq 1$ after subtraction. Computation of the Difference-of-Gaussians images is illustrated on the right hand side of Figure 1. The local maxima and minima can then be detected by comparing each pixel in the Difference-of-Gaussians images to its 1-neighborhood in $x, y$ and scale dimensions. This procedure is illustrated in Figure 2.

To make the detection robust, the next step is to derive concrete values for the amount of scales $s$ that are sampled per each level of the pyramid. Each pixel that is checked in the detection phase is required to contain an adjacent pixel in $x, y$ and $\sigma$ dimensions, which is why only those Difference-of-Gaussians images that contain a higher and a lower adjacent scale image can be used for extrema detection. Therefore, for the detection of local extrema to cover a whole octave, $s+2$ Difference-of-Gaussians images are required in each octave. As the DoG images are calculated from adjacent blurred images, $s+2+1 = s+3$ scale samples per octave are required. By experimental results, Lowe has shown that the most stable results are obtained by choosing $s = 3$ scales per octave [Low04], which indicates sampling 6 blurred
images per each octave of scale-space. Therefore, for the detection of local extrema
to cover a whole octave, \( s + 3 \) blurred images are required in each octave to provide \( s \) Difference-of-Gaussians images with both adjacent images.

As we now have the amount of blurred images needed for each level derived, we
can also derive the parameters for the smoothing kernels. As explained before, the
blurred images are separated by a constant multiplicative factor \( k \) in scale and we
want the detection to cover a whole octave. Setting \( k = 2^{1/s} \) yields DoG images with
scales \((\sigma, 2^{1/s} \sigma, \ldots, 2^{(s+2)/s} \sigma)\) which yields the desired result. We choose the blurred
image with scale equal to \( 2^{n/s} \sigma = 2\sigma \) as the basis of the next level of the pyramid.

The Laplacian of Gaussian and its approximations such as Difference-of-Gaussians
are not the only methods for keypoint detection. An example of an alternative
method is to search for maxima of determinants from Hessian matrices, which is
used in the Speeded-Up Robust Features (SURF) algorithm [BTG08].

3.2 Keypoint localization

The scale-space function yields points that are invariant to scale, rotation and trans-
lation. Rotation and translation invariance follow from the fact that all computa-
tions are performed in the local neighborhood of a pixel. The next major issue to
be taken into account is image noise. The previous computations are sensitive to
noise in the sense that points with low contrast to their neighbors can easily lose
their status as extrema even when small amounts of noise is applied.

3.2.1 Rejecting points with low contrast

The easiest way to detect keypoints with low contrast is to check the values of the
keypoints in the Difference-of-Gaussians image. A keypoint will be discarded if its
value is less than a predefined threshold. This way we make sure that every pixel
differs from its neighbors in the scale-space function by a significant amount. Stabili-
ity of this procedure can be significantly improved by approximating the continuous
scale-space function using a Taylor expansion, finding the location of the keypoint
in sub-pixel accuracy and checking the point’s value at the approximated location
instead [Low04, BL02]. Any continuous and differentiable function can be approxi-
mated by Taylor’s series which is a series of derivatives computed at a single point.
For efficiency, we only use the series up to second order derivatives. With the Taylor
expansion we can compute the value of the extrema at subpixel accuracy using the
Quasi-Newton method [Rus06, Chapter 5.6]. First, we evaluate the function $D$ and its derivatives at the sample point $a$ giving

$$D(a + \bar{x}) = D(a) + \frac{\partial D(a)}{\partial x} \bar{x} + \frac{1}{2} \bar{x}^T \frac{\partial^2 D(a)}{\partial x^2} \bar{x},$$

where $\bar{x} = (x, y, \sigma)$ is the offset from the sample point. The subpixel accuracy location $\hat{x} = a + \bar{x}$ of the sample point $a$ can be determined by solving a linear equation given by

$$\frac{\partial D(a)}{\partial x} \bar{x} + \frac{\partial^2 D(a)}{\partial x^2} \hat{x} = 0$$

giving

$$\hat{x} = -\frac{\partial^2 D(a)^{-1}}{\partial x^2} \frac{\partial D(a)}{\partial x}.$$

If the offset is greater than 0.5 in any dimension $(x, y, \sigma)$, it indicates that the extremum is closer to that point and the interpolation is performed at that point instead. The actual value of the extremum on subpixel accuracy can now be calculated by substituting the sub-pixel accuracy extremum to the evaluated series giving

$$D(\hat{x}) = D(a) + \frac{1}{2} \frac{\partial D(a)^T}{\partial x} \hat{x}.$$

Empirical results have shown that pixels with subpixel accuracy $|D(\hat{x})| < 0.03$ should be discarded [Low04].

The values of the derivatives can be approximated using neighboring pixel values in the Difference-of-Gaussians image [BL02]. For example, to approximate the first order partial derivative respective to $y$ at location $a = (x, y, \sigma)$ we use

$$\frac{\partial D(a)}{\partial y} = \frac{D(x, y - 1, \sigma) - D(x, y + 1, \sigma)}{2}.$$

All other first order derivatives are computed in similar fashion [MW98, Chapter 51]. The second order derivatives and the cross derivatives are also computed using neighboring pixel values. As an example second order derive $\frac{\partial^2}{\partial x^2}$ can be computed using adjacent derivative values

$$\frac{\partial^2 D(a)}{\partial x^2} = \frac{1}{2} \left( \frac{1}{2} (D(x - 1, y, \sigma) - D(x, y, \sigma)) - \frac{1}{2} (D(x, y, \sigma) - D(x + 1, y, \sigma)) \right)$$

$$= \frac{1}{4} (D(x - 1, y, \sigma) - 2D(x, y, \sigma) + D(x + 1, y, \sigma)).$$

Note that the computation of the second order derivative differs slightly from the computation of the first order derivative. The reason for this is to keep the influential neighborhood small. Cross derivatives are approximated in the same fashion as the
first order derivatives. Cross derivative with respect to $\sigma$ and $x$ is given by the following equation

$$\frac{\partial^2 D(a)}{\partial x \partial \sigma} = \frac{1}{2} \left( \frac{\partial D(a)}{\partial x} - \frac{\partial D(a)}{\partial \sigma} \right).$$

The values for the derivatives calculated from a neighborhood given by

$$\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 1 \\
1 & 1 & 1
\end{bmatrix}$$

are as follows: $D_x = (4 - 1)/2 = 3/2$, $D_{xx} = (4 - 10 + 1)/4 = -5/4$ and

$$\frac{\partial^2 D(a)}{\partial x \partial y} = \frac{1}{2} \left( \frac{4 - 1}{2} - \frac{2 - 1}{2} \right) = \frac{1}{2},$$

where $D_x$ represents the derivative of the function $D$ with respect to $x$.

### 3.2.2 Rejecting edge responses

The Difference-of-Gaussians function has strong peaks at the edges of objects in an image [Low04]. The problem with the edge responses is that they might have significant contrast to their neighbors while still being sensitive to image noise. Even small amounts of noise can cause the locations of the edge responses to shift significantly which makes them unstable. To identify edge responses, we measure the curvatures of the DoG function in the proximity of the keypoints. A characteristic property of a point along an edge is that the DoG function has a high curvature across the edge and a low curvature in the perpendicular direction. This property is visualized in Figure 3.

The principal curvatures of a function at a point provide us with information about the highest and lowest curvatures of the function around that point. The extrema along edges will have large principal curvatures across the edge but small ones in the perpendicular direction. To reject points located along edges, we search for keypoints with a high ratio of principal curvatures. The principal curvatures can be calculated from a 2x2 Hessian matrix $H$ computed at the location and scale of the keypoint

$$H = \begin{bmatrix}
D_{xx} & D_{xy} \\
D_{yx} & D_{yy}
\end{bmatrix}.$$ 

The derivatives in the Hessian matrix are calculated in the same fashion as in the Taylor expansion. Due to the separability of the scale-space kernel, see Section 2,
Figure 3: A typical characteristic of an edge is the low variation of pixel values in one
direction and high variation in the other. The curvature of the illustrated function
is high when passing over the edge along $x$ axis [MH80] and low when traversing
along the $y$ axis.

we have
\[
\frac{\partial D}{\partial x \partial y} = \frac{\partial}{\partial x} \left( \frac{\partial}{\partial y} D \right) = \frac{\partial}{\partial y} \left( \frac{\partial}{\partial x} D \right) = \frac{\partial D}{\partial y \partial x}
\]
and therefore $D_{xy} = D_{yx}$ [Lin94].

The eigenvalues of the Hessian matrix are proportional to the principal curvatures
and the ratio of the principal curvatures can be evaluated without calculating the
actual values of the principal curvatures. Let $\alpha$ be the eigenvalue with the largest
magnitude and $\beta$ the smaller one. We can calculate the sum $\alpha + \beta$ from the trace
of the Hessian matrix and their product from the determinant
\[
\text{Tr}(H) = D_{xx} + D_{yy} = \alpha + \beta
\]
\[
\text{Det}(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha \beta.
\]

Let $r$ be the ratio between the largest eigenvalue and the smaller one, so that $\alpha = r \beta$. Then
\[
\frac{\text{Tr}(H)^2}{\text{Det}(H)} = \frac{(\alpha + \beta)^2}{\alpha \beta} = \frac{(r \beta + \beta)^2}{r \beta^2} = \frac{(r + 1)^2}{r}.
\]

Now $(r + 1)^2/r$ only depends on the ratio of the eigenvalues instead of their actual
values. Its value is at minimum when the eigenvalues are equal and the value in-
creases when $r$ increases. To check if the extremum crosses a threshold defined by $r$
we only need to check that
\[
\frac{(D_{xx} + D_{yy})^2}{D_{xx}D_{yy} - (D_{xy})^2} = \frac{\text{Tr}(H)^2}{\text{Det}(H)} < \frac{(r + 1)^2}{r}.
\]

Lowe showed empirically that setting $r = 10$ yields the most stable results [Low04].
3.3 Describing local image regions

The last step of the SIFT algorithm is to transform local image regions of keypoints into distinctive representations which allow us to repeatably detect the corresponding regions from a similar image that has been subjected to transformations and deformations. Keypoints combined with these local image descriptors are called local image features and are the output of the SIFT algorithm. To make the features comparable, the representation into which the local regions are stored must be invariant to transformation and deformations. The locality of the information used to form these features makes them invariant to translation but other issues such as rotation and illumination have to be taken into account while generating the representation.

3.3.1 Finding a consistent local orientation

To ensure rotation invariance for the local image descriptors, we need to find a consistent orientation within the local neighborhood of each keypoint to be able to normalize the computations relative to this orientation. Lowe argued that the following approach to find the orientation gives the most stable results among many other approaches [Low04]. First, we select the Gaussian smoothed image $L(x, y, \sigma)$ with the closest scale for each keypoint to make the following calculations invariant to scale. For each scale-space point $\bar{x} = (x, y, \sigma)$, the gradient orientation $\theta(x, y)$ and gradient magnitude $m(x, y)$ are calculated in the following way:

$$
\theta(\bar{x}) = \tan^{-1}\left(\frac{L(x, y + 1) - L(x, y - 1)}{L(x - 1, y) - L(x + 1, y)}\right)
$$

$$
m(\bar{x}) = \sqrt{(L(x - 1, y) - L(x + 1, y))^2 + (L(x, y - 1) - L(x, y + 1))^2}.
$$

For efficiency, the gradient values are precomputed for all scales. For each keypoint, an orientation histogram is formed from the gradient values of the corresponding local region. The gradient values are divided into histograms consisting of 36 bins, each covering 10 degrees of orientation. For example, all the local gradient magnitudes with orientation value of 52 are put into the bin assigned for orientations within the range $[50, 60]$. To make local gradients invariant to small changes in the position of the local region, which might occur due to various possible errors in capturing the image, we give less emphasis to the gradient magnitudes that are far away from the keypoint and therefore most affected by the changes in locations. We weight each magnitude value around the corresponding keypoint with a a Gaussian
Figure 4: The orientation histogram has degrees as the x-axis and magnitude as the y-axis. The dominant orientation is derived by fitting a parabola to the adjacent bins of the dominant bin and by choosing the peak of the parabola. The points that define the parabola are marked with red circles.

circular window, which is a finite approximation of a two dimensional Gaussian function, with the keypoint in the middle. The \( \sigma \) used as a parameter for the window is the scale of the keypoint multiplied by a factor 1.5. The value of a bin is the sum of the gradient magnitudes in the corresponding bin.

The dominant orientation of the histogram is selected as orientation of the corresponding keypoint. The dominant orientations is invariant to other transformations in addition to image rotation and therefore, it can be used for normalizing the descriptor to rotation. In case there are other peaks in the histogram that are within 80% of the highest peak, a new keypoint is created with that orientation. In other words, we have multiple keypoints with same spatial and scale coordinates but with differing orientations. To get the best estimate of the dominant orientations, peaks in the orientation histogram are interpolated with adjacent bins. This is performed by fitting a parabola to the dominant bin and the two bins adjacent to it. The parabola has orientation as the x-axis and magnitude as the y-axis. The peak of the parabola is assigned as the final orientation of the keypoint. Fitting a parabola into an orientation histogram is illustrated in Figure 4.
3.3.2 Local image descriptor

Each keypoint has now been assigned with a location, scale and orientation, which provide us with a repeatable image region that is invariant to these parameters. These parameters however, do not yield any information that could be used to relate two different image regions together. The next step is to generate a descriptor that captures the characteristics of the region in a distinctive way while providing invariance to the other variations such as changes in lightning conditions and three-dimensional viewpoint. These descriptors are used to measure the similarity between two distinct image regions.

The SIFT algorithm uses local gradient values to represent the local region of a keypoint [Low04]. The first step of forming the local descriptor is to normalize local information with respect to rotation. This is performed by rotating local gradient orientations and local coordinates with respect to the dominant orientation assigned to the keypoint; see the previous section. An example of rotating local gradients is given in Figure 5. To make the representation invariant to small shifts in the gradient positions of the local region, the local gradient magnitudes are weighted with a Gaussian circular window with parameter $\sigma$ equal to half of the width of the region. The local gradient values and the weighting function are illustrated in the left hand side of Figure 6.

To achieve invariance to shifts in gradient positions, the local neighborhood is divided into subregions from which orientation histograms are calculated to represent the gradient values of the corresponding subregion. Lowe argued that the most stable results are obtained when using $4 \times 4$ subregions and histograms with 8 orientation bins, where the histograms are formed in the similar manner as before. This however, results in a high dimensional feature vector (128 dimensions) which leads into a computationally expensive solution. In mobile solutions, the dimensionality is usually reduced to $3 \times 4$ (36 dimensions) [WRM+08], which increases computational efficiency with the cost of accuracy. A bin for each local gradient value is derived in the following way: the matrix cells correspond to eighths in the neighborhood and the bin is determined by the gradient orientation. To avoid boundary effects, each sample is added to the 2 nearest orientation bins with weight $1 - d$, where $d$ is the distance from the bins measured in units of the histogram bin spacing. For example in the case of 8 bins and a sample with orientation of 60 degrees, the distances to the nearest bins are $d_1 = |60 - 90|/45$ and $d_2 = |60 - 45|/90$. Therefore the gradient magnitude of the sample is added to the 90 degree bin with weight equal to $1 - d_1$. 
Figure 5: The initial locations of local gradients are shown in Figure a. The dominant orientation within the local neighborhood is 90 degrees and it is visualized by the big arrow starting from the center of the neighborhood. Rotating the coordinates of the local gradients counterclockwise results in the state illustrated in the Figure b. Figure c shows the gradient values explicitly and Figure d illustrates the final result after normalizing the dominant orientation to 0 degrees by rotating the gradient values counterclockwise according to the dominant orientation.
A keypoint descriptor is created by first calculating the gradient values at each image sample point around the keypoint location and the gradient magnitudes are weighted with a Gaussian circular window. Then the $4 \times 4$ subregions are summarized into orientation histograms on the right. The arrows on the left represent the gradient orientations and the length of the arrows correspond to the magnitude of the corresponding gradient. This figure shows a $2 \times 2$ descriptor computed from $8 \times 8$ neighborhood but the $4 \times 4$ descriptor from $16 \times 16$ neighborhood is computed in a similar manner.

and into the 45 degree bin with the weight $1 - d_2$. Now we have formed a $4 \times 4 \times 8$ dimensional descriptor that contains all the needed information from local gradients. Transforming a local region into a descriptor is illustrated in Figure 6.

The last step is to normalize the descriptor magnitudes to reduce the effects of changes in illumination [Low04]. As the feature vectors are computed from pixel differences, a global brightness change in which a constant is added to all pixel values does not affect the computations. Local changes in illumination on the other hand increase the contrast of pixel values by some constant multiplicative factor, which is canceled by vector normalization. Therefore, the normalization makes the vectors invariant to linear illumination. Camera saturation and variations in illumination on three-dimensional surfaces generate non-linear illumination, which cannot be countered by a simple normalization. Lowe argued that giving less emphasis to high gradient magnitudes makes the comparison of large intensity vectors unimportant and therefore, the distribution of the gradient orientations gains more emphasis [Low04]. He showed empirically that cutting all normalized values above 0.2 and normalizing the vector again yields the most robust results against non-linear illumination. Cutting in this context denotes assigning the value 0.2 to all
gradient magnitudes above 0.2.

The gradient vectors are the final result of the SIFT algorithm. Each vector \( v \) contains the following information: location \((v_x, v_y)\) within the input image, the scale \( v_s \) of the Gaussian blurred image from which the intensity values were derived, orientation \( v_\theta \) of the dominant direction of intensity changes and the local image descriptor vector \( v_{lid} \) containing the orientation histograms. The resulting SIFT vectors are 4 + 128 dimensional vectors, where the first 4 elements correspond to x-coordinate, y-coordinate, orientation and scale respectively and the rest correspond to the 128 dimensional intensity vector constructed from the 4 \( \times \) 4 subregions using orientation histograms with 8 bins.

### 3.4 Implementation details

The previous sections introduced the Scale-Invariant Feature Transform algorithm in detail. To help the reader with implementing the algorithm, this section discusses implementation issues arisen from camera sensors, gives some concrete values to the Gaussian smoothing parameters and shows examples of smoothed images. The examples are created by the author’s implementation, which is available at [http://universe.hiit.fi/sift](http://universe.hiit.fi/sift).

#### 3.4.1 Initialization

The first issue is the smoothing parameter \( \sigma \) on the natural input image \( I(x, y) \). Theoretically, the input image \( I(x, y) \) corresponds to the scale-space image \( L(x, y, 0) \), which indicates a situation where the sampling is arbitrarily accurate. However, natural images are always discrete samples from a "continuous" world, which means that initially some of the finest details have been lost. This indicates that the natural input image actually corresponds to a blurred image \( L(x, y, \sigma_0) \), where \( \sigma_0 \) depends on the accuracy of the camera sensor. Lowe argued that for purposes of the SIFT algorithm, it is sufficient to assume that \( \sigma_0 = 0.5 \) [Low04]. The initial scale parameter \( \sigma_0 \) affects the size of the image scale domain where the Gaussian smoothed images are sampled and Lowe has suggested that the input image should be further smoothed before creating the scale-space pyramid. The downside of the initial smoothing is that it increases the widths of the kernels that are used in the SIFT algorithm which is computationally expensive. Lowe showed empirically that the best results are obtained when choosing 1.6 as the initial amount of blur [Low04].
While the initial blurring expands the domain of sampling, it is effectively discarding the highest frequencies of the input signal. Therefore, the SIFT algorithm first doubles the size of the input image by linear interpolation before performing the pre-smoothing. This procedure doubles the initial blurring parameter $\sigma_0$ relative to its new pixel spacing and a smaller Gaussian kernel is needed to achieve the desired 1.6 of blur for the first image in the scale-space representation. More precisely, the 1.6 initial smoothing can now be obtained by convolving the input image with a Gaussian kernel of width

$$\sigma = \sqrt{1.6^2 - (2 \cdot \sigma_0)^2} = \sqrt{1.6^2 - (2 \cdot 0.5)^2} = \sqrt{1.56}.$$  

Lowe has empirically shown that this procedure also increases the number of stable keypoints by approximately a factor of 4 [Low04].

As some of the calculations performed in the SIFT algorithm are computationally expensive and the resulting feature vectors are high dimensional, numerous optimization schemes have been proposed. For example, the SURF algorithm uses a rough approximation of the Gaussian filter called the box filter [BTG08]. While achieving faster performance, the box filter approximation provides slightly less stable features due to the loss of information caused by the approximation. The SURF algorithm uses only 64-dimensional feature vectors to improve the efficiency of feature vector comparison. Another option is to use binomial kernels to approximate the Gaussian kernels [ESM+06]. This approach utilizes the fact that Gaussian kernels can be approximated by a sequence of small binomial kernels. For example,

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \ast \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \ast \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \ast \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \approx G(x, y, 1)$$

3.4.2 Examples

To narrow down the gap between theory and practice, it is reasonable to explicitly present some concrete values for the smoothing kernels used in an implementation. We have chosen to use $s = 2$ as the number of scales sampled per octave and $\sigma_0 = 1.6$ as the amount of smoothing before generating the first octave of scale-space. To make the creation of the scale-space fast and to enable parallel computation, all the blurred images are typically derived from the first image of an octave. Let $n \in \{2, 3, 4, 5\}$ be a running number for a blurred image in the first octave. The width $\sigma$ for a smoothing kernel, which generates the $2^{n/s} = 2^{n/2} = k$ gap between the basis of the
octave and the $n$:th blurred image, can be calculated from the formula

$$L(x, y, k\sigma_0) = G(x, y, \sqrt{(k\sigma_0)^2 - \sigma_0^2}) * L(x, y, \sigma_0)$$

giving

$$\sigma = \sqrt{(k\sigma_0)^2 - \sigma_0^2} = \sqrt{\sigma_0^2(k^2 - 1)}.$$

The values for smoothing kernels on other octaves are computed similarly using the scale of the first blurred image of the corresponding octave instead of $\sigma_0$. A scale-space pyramid with two levels is illustrated in Figure 7.

The Difference-of-Gaussian images derived from the images in Figure 7 are illustrated in Figure 8. The DoG images have strong responses on the edges and on the blob-like regions of the images. The scale-space extrema extracted from the DoG images are illustrated on the left hand side of Figure 9 and finally, the SIFT feature vectors remaining after localization step are illustrated on the right hand side of Figure 9.
Figure 7: Two scale-space octaves with different sample rates are generated from a basis image refsub: init with initial Gaussian blur adjusted to 1.6. The resulting scale values are shown below the corresponding images and the values show that, as intended, the basis image 7f of the second octave has double the smoothing value in comparison with the initial image 7a. For clarity, we have chosen to use $s = 2$ sampled scales per octave in the example which requires $s + 3 = 5$ blurred images per octave. This figure also illustrates the constant gap $k = 2^{1/s} = \sqrt{2}$ between each adjacent pair of blurred images.
Figure 8: Two Difference-of-Gaussians octaves generated from the blurred images illustrated in Figure 7. The scale parameter for each DoG image is shown below the image. The white areas denote the peaks in the DoG function whereas the black areas correspond to non-interesting parts of the image. The detection of the local extrema is performed in the images 8b, 8c, 8f and 8g.

Figure 9: The image 9a on the left hand side visualizes the keypoint locations extracted in the SIFT algorithm that remain after discarding the low quality keypoints. The features derived from the keypoints are visualized in the image on the right hand side 9b. The length of the feature vectors indicate the scale of the corresponding features.
Figure 10: The object recognition method introduced by Lowe [Low04] finds the best matches between feature points from a query image (on the right hand side) and from a database of images. A set of feature correspondences found in a database image are illustrated on the left hand side. The points in the query image form a two dimensional shape, which is required to be also found in the database image. In this example the corresponding shapes and objects exist in the database image but as the matches are found in different instances of the same object, the shape verification fails.

4 Visual categorization

In the previous sections, we introduced a method for extracting robust features from natural images that are invariant to changes in viewpoint and scale. In this section, we utilize the features for image recognition and more precisely for visual categorization in a grocery environment. Image recognition is the task of finding a similar enough image in a database of images for a given unlabeled input image. One approach is to identify the images in the database by the objects appearing in them, and to recognize objects in a query image with the object recognition algorithm introduced by Lowe [Low04]. The database image with the most matching objects is chosen as the matching image. Unfortunately, as the object recognition procedure is based on verifying the possible matching objects with a two dimensional transformation model, multiple instances of a same object make this verification fail. An example of this is given in Figure 10.

Visual categorization is a more abstract approach to image recognition. Instead of trying to find matching pairs of images, visual categorization aims at assigning a query image into a predefined class of images. For example, we could have classes for cars, bicycles and airplanes, and we want to predict into which of these classes
an unlabeled image belongs to. The characteristics of the classes are learned in the training phase from features extracted from labeled data set of images, which are referred to as training images. Image recognition can thus be considered a special case of visual categorization, where there is only one image per class.

In grocery environments, the shelves contain many instances of the same product and therefore the object recognition approach fails. Another approach is to try to characterize the contents of an entire image into a single representation that can be compared to the representations of other images. An example of this is the color histogram representation of an image, which stores the amounts of different color values within the image. A major problem with this approach is that it ignores the actual contents of images and therefore it can easily consider two significantly different images to be similar. Mobility also renders characterizing an image with a global representation impossible. For example, if we compare two images of the same view from different distances, we would like to consider them similar. However, if we simply look at the images as a whole, they might have completely different global characteristics.

4.1 Bags of Keypoints

To effectively categorize images, we need to have access to high level information about the contents of the image. In the following, we introduce the Bags of Keypoints method, which is motivated by text classification, where a popular approach is to consider word co-occurrence information as features in the classification of documents. In the Bags of Keypoints method, similar local image features are grouped together and frequencies of these groups in images are used to train a classifier. The Bag of Keypoints representation corresponds to the number of features in each group from the images belonging the corresponding category [CDF+04].

The method consists of four steps:

1. Extraction of image features: In our case, this is performed by the SIFT algorithm. We also consider color features.

2. Assigning feature descriptors to a set of predetermined clusters using vector quantization [Hay98].

3. Counting the number of descriptors assigned to each cluster.
4. Applying a multi-class classifier, which treats a bag of keypoints as a feature vector.

**Vector quantization**

As explained in Section 3.3.2, the SIFT algorithm extracts highly distinctive and high dimensional feature vectors that characterize small regions within an image. One way to characterize an image or a set of images is to store all feature vectors extracted from the training images. This however, requires a lot of space and comparing an unlabeled image to the stored images requires comparison of each new vector against each stored vector which is computationally exhaustive [CDF+04]. Moreover, the SIFT algorithm captures very low level information whereas we are interested in learning high level features from the image. To gain higher level information from the vectors, we use vector quantization to create a *visual vocabulary* that is used to characterize each class of images. In practice, this means choosing a set of meaningful vectors or visual words that contain enough information from each class to distinguish relevant differences in images, while leaving out the most low level information.

In our case, meaningful features correspond to features that occur frequently and are distinctive. After deriving the visual words, each category can be characterized by the amount of occurrences of the visual words within the training images belonging to that category [CDF+04]. As the SIFT features are high dimensional real-valued vectors, it is highly unlikely that two completely similar vectors would ever appear in a finite training data. Therefore we search for dense areas or feature clusters within the high dimensional feature space and consider all features within a cluster similar. The amount of occurrences of visual words correspond to the amount of features within the corresponding cluster. As the distance of two vectors can be measured by Euclidean distance, it is reasonable to consider density according to the Euclidean distance metric. Finding the clusters and assigning the data points to them is referred to as *clustering*. As our training set of feature vectors is relatively large, we require an efficient algorithm for clustering.

For simplicity and computational efficiency, *k*-means [TK04, Chapter 14.5.1] is a good choice as the clustering algorithm. The *k*-means algorithm works as follows: first, choose *k* random feature vectors as the initial cluster centers. Next, assign all feature vectors to a cluster that is defined by their nearest cluster center and choose the centroids of the clusters as the new cluster centers. This process is iterated
until the clusters do not change. It has been shown that the algorithm always converges to a local optimum defined by the maximum distance from each point to the nearest cluster center. A pseudo-code representation of the algorithm is given in Algorithm 1.

**Algorithm 1** $k$-means clustering

**Require:** A set of SIFT feature vectors data with at least $k$ samples.

Let $C$ be the $k$ vectors randomly chosen from data

Let loop = 1 $\triangleright$ The loop condition.

Let $v_c$ be the cluster for each feature vector $v$.

while loop do

    loop = 0

    for Each $v \in$ data do

        $c = \min_{i \in C} |v - i|$

        if $v_c \neq c$ then

            loop = 1 $\triangleright$ Clustering did not converge.

            $v_c = c$.

        end if

    end for

end while

There are two major limitations in the $k$-means algorithm. First, the algorithm only converges to a local optimum. Secondly, the algorithm does not automatically determine the parameter $k$. However, we are not actually interested in a ”correct” clustering in the sense of feature distributions, but rather in the accuracy of the categorization [CDF04]. Therefore, we simply run the clustering several times on the training data with different values for $k$ and with different initial cluster centers. There are also other approaches for optimizing the result of $k$-means; see [BF98, SBY08]. The final result for clustering is chosen according to the principle of lowest empirical risk in the categorization process [Vap98, Chapter 3]. This simply indicates running the categorization with different parameters and choosing the parameter values that yield the highest ratio of correct predictions.

We now have the necessary information for creating the bag of keypoints representation for each image category. As the $k$-means algorithm assigns each feature vector to a cluster and we know to which class each image and its features belong to, we can count the number of features in each cluster for each class. This results in an integer vector representation of length $k$ for each image class where the value of the
The $j$:th element corresponds to the number of features from that class that belong to the cluster $j$.

**Categorization by a Naive Bayes classifier**

The last step of the training phase is to train a multi-class classifier, which is used to predict class labels for unlabeled images [CDF+04]. In this work, we use the Naive Bayes classifier for its simplicity and computational efficiency. The Naive Bayes classifier is a probabilistic classifier based on the Bayes’ Rule [Lew98] given by

$$P(C = c_j | X = x) = P(C = c_j) \frac{P(X = x | C = c_j)}{P(X = x)}.$$  \hspace{0.5cm} (4)

The theorem essentially states that we can calculate the probability of a sample $x$ belonging into a certain class $c_j$ by knowing the prior probability $P(C = c_j)$ of all classes and the conditional probability $P(X = x | C = c_j)$ for each sample and class pair. The prior probability of a class indicates the probability of a random sample belonging to that class when we do not have any information of the sample. For example, if we had a bag of candies with 20 red candies and 30 black candies and we pick one candy from the bag at random without checking the color, we expect the candy to be black. Moreover, we can make a prior assumption that a random candy belongs to the class of black candies with probability $30/50$.

As our training data for the classifier is in the Bags of Keypoints format, a constant length vector consisting of integer counters for each class, it does not make sense to model our data with for example normal distributions, which are models for continuous data. The standard approach to model discrete data is to use a multinomial distribution. This indicates assuming that each word in a document is chosen independently from a multinomial distribution over the words in the visual vocabulary. In general, the occurrence of some word might affect the probability of the occurrence of another word and therefore, the independence is a naive assumption. This can be more easily understood from the case of text documents, where it is obvious that some natural words occur often in pairs or in bigger groups. Regardless of the naive assumption, the Naive Bayes classifier has been shown to produce accurate results [DP97]. The classifier is illustrated in Figure 11.

To categorize a bag of keypoints representation $x$ of an unlabeled image, we apply the maximum a posteriori decision rule, i.e., we calculate the conditional probabilities

$$P(C = c_j | I = x)$$
The Naive Bayes classifier predicts the class label according to features $f_1, \ldots, f_k$, where $k$ is the number of features used to train the classifier. In the bags of keypoints model, value of $k$ corresponds to the number of cluster centroids. The arrows illustrate dependencies, i.e., the class label depends on the values of the features and the features are assumed to be mutually independent.

For each image class $c_j$ and choose the one with the highest probability. Let $V = v_1, v_2, \ldots, v_k$ be the visual vocabulary obtained with the $k$-means algorithm. To calculate the conditional probabilities for each class, we apply the Bayes’ rule giving

$$P(C = c_j | I = x) = P(C = c_j) \frac{P(I = x | C = c_j)}{P(I = x)}.$$  \hspace{1cm} (5)

As we know all values in vector $x$, the probability $P(I = x)$ is a constant over all classes. Therefore we can write Equation 5 as

$$P(C = c_j | I = x) \propto P(C = c_j) P(I = x | C = c_j)$$

$$= P(C = c_j) \prod_{i=1}^{\|V\|} P(v_i | C = c_j)^{N(i, j)},$$

where $N(i, j)$ corresponds to the occurrences of keypoint $v_i$ in class $c_j$. Since we are taking a product of probabilities $P(v_i | C = c_k)$, we wish to avoid multiplying with zero. To accomplish this we apply Laplace smoothing \cite{CDF04} on the counts, which assigns an uniform prior probability on all possible occurrences of features within all classes. In practice Laplace smoothing can be implemented by adding a constant $\alpha$ to each count. In our case we use $\alpha = 1$ giving

$$P(v_i | C = c_j) = \frac{1 + N(i, j)}{\|v\| + \sum_{s=1}^{\|v\|} N(s, j)}.$$  \hspace{1cm} (6)
The probabilities that are multiplied are often small, which indicates that the final probability values can easily cause an underflow. To avoid this issue, instead of multiplying the probabilities directly, the probabilities are transformed onto a logarithmic scale and summed up.

4.2 Removing noise

The Bags of Keypoints method provides us with a method for general visual categorization. As we are restricting ourselves into a grocery environment, we assume that all training and test data consist of images of grocery shelves. A typical example of a grocery shelf with SIFT keypoints is illustrated in Figure 13. The characterizing contents of shelves are the products appearing in them and therefore we wish to remove the features arising from non-interesting objects such as price tags or floor tiles. Even though both SIFT and Bags of Keypoints algorithms have been shown to be robust to background clutter [Low04, CDF+04], objects such as price tags contain relevant structure from the viewpoint of these algorithms.

As the products are usually grouped together, the majority of SIFT features are grouped in dense areas. On the other hand, non-interesting objects such as price tags tend to be far away from similar objects. Therefore we propose a clustering algorithm that groups up the dense areas of SIFT features into clusters. In addition, we require each cluster to contain a significant amount of features to reject small non-interesting objects such as light switches. As we cannot know the number of groups of products appearing in an image beforehand, the most straightforward approaches such as k-means clustering do not work. Furthermore, we are dealing with the SIFT features that contain information about the local intensities, orientation and scale in addition to the spatial coordinates. Therefore, we define the distance between two points in a more sophisticated fashion, which takes the additional information into account.

Distance measures

As the grocery shelves contain multiple instances of the same object and the objects are usually captured from a distance, the scale values are rather uniform within a group of similar objects. Therefore a difference between scale values of different indicates a change of object type. As the convolution operation increases the scale of an image by a multiplicative factor, we measure the differences between scales as
Figure 12: The orientations of feature vectors are illustrated as arrows starting from the locations of the corresponding keypoints. The example images show that the orientations of keypoints occurring from price tags and the structures of the shelves usually have opposite directions. The examples also indicate that the interesting objects (products) contain a lot of keypoints with high range of orientations making it likely that a cluster will be formed on them.

the ratio of two different scale values. The scale distance of $q$ and $p$ is given by

$$d_s(p, q) = \max \left\{ \frac{ps}{qs}, \frac{qs}{ps} \right\},$$

where $qs$ and $ps$ represent the scale values of the corresponding SIFT features.

Keypoints occurring near the edges of objects commonly have orientations that are nearly orthogonal to the edges and point outwards from the object. This indicates that two keypoints occurring near a shared edge of distinct objects tend to have opposite directions. To take this observation effectively into account, we have chosen to measure the difference of orientations (radians) of vectors $p$ and $q$ by the following formula

$$d_\theta(p, q) = |\pi - (p_\theta - q_\theta)|,$$

which gives high distances to opposite directions orientations while giving low distances to parallel orientations. Examples of situations supporting this approach are illustrated in Figure 12.
Density-and-Join based clustering

To remove unwanted keypoints, we propose to use the Density-and-Join based clustering algorithm DJ-cluster [ZFL+07], which derives clusters based on the local density of points. The idea of this algorithm is to calculate a local neighborhood for each input point and to combine neighborhoods with non-empty intersections. For each point, we calculate a neighborhood which consists of points that are within a predefined distance threshold \( \text{Eps} \). If the number of points within this neighborhood exceeds another threshold, \( \text{MinPts} \), the neighborhood is considered as a potential new cluster and all the points within the neighborhood are marked as processed. If there are existing clusters that have a non-empty intersection with the new cluster, they are merged. Otherwise, a new cluster containing the whole neighborhood is formed. A pseudo-code representation for the DJ clustering algorithm is given in Algorithm 2 that uses a neighborhood query given by Algorithm 3 as a subroutine.

**Algorithm 2** Density-and-Join based clustering

**Require:** Threshold for the maximum distance \( \text{Eps} \) and the minimum number of points within a neighborhood \( \text{MinPts} \). The current feature point \( p \) and data set \( \text{Data} \) containing all feature points.

\[
\text{Clusters} = \emptyset \quad \triangleright \quad \text{A set of clusters.}
\]

while An unprocessed point \( p \) is left in the data set \( \text{Data} \) do

\[\text{N}(p) = \text{EpsNH}(p)\]  \quad \triangleright \quad \text{The neighborhood query given by Algorithm 3.}

if \( |\text{N}(p)| < |\text{MinPts}| \) \& \( p \) has not previously been added to a cluster then

Label \( p \) as noise.

end if

if \( |\text{N}(p)| \geq \text{MinPts} \) then

for each cluster \( \in \text{Clusters} \) do

if \( \text{N}(p) \cap \text{cluster} \neq \emptyset \) then

\[\text{N}(p) = \text{N}(p) \cup \text{cluster}\]  \quad \triangleright \quad \text{Combine joint clusters.}

\[\text{Clusters} \setminus \{\text{cluster}\}\]  \quad \triangleright \quad \text{Remove duplicate points.}

end if

end for

\[\text{Clusters} \cup \{\text{N}(p)\}\]

end if

end while
Figure 13: Figure 13a shows the keypoints extracted by the SIFT algorithm. The DJ clustering is applied to these points and the ones labeled as noise are left out. The result of this process is illustrated in Figure 13b.
Algorithm 3 neighborhood query

Require: Threshold for the maximum distance $E_{\text{ps}}$, $E_{\text{ps}}$, and $E_{\text{po}}$ for Euclidean distance of coordinates, scale and orientation respectively. The current feature point $p$ and data set $Data$ containing all feature points.

$E_{\text{ps}}NH = \emptyset$ \quad $\triangleright$ The $E_{\text{ps}}$-neighborhood

for each $q \in Data$ do
  if $d_s(p, q) < E_{\text{ps}} \parallel d_{\theta}(p, q) < E_{\text{po}} \parallel \text{Euclidean}(p, q) < E_{\text{pe}}$ then
    $E_{\text{ps}}NH = E_{\text{ps}}NH \cup q$
  end if
end for

return $E_{\text{ps}}NH$

4.3 Extending the local features with color information

In addition to noise removal, we propose an extension for visual categorization in a retail environment that utilizes the color values of the images. For example, milk products are often packed in white cartons and beer is typically sold in brown bottles. We utilize this observation by storing the RGB color values from within the 1-neighborhood of the keypoints and using them in classification. Examples of typical milk and beer shelves are given in Figure 14.

As the color values are not distinctive in the same way as the SIFT features, we do not perform any vector quantization or count the occurrences of a certain color within an image. Instead, we model the color values with normal distributions. As we are dealing with the RGB color model, we fit a separate normal distribution for each color channel. As in the Bags of Keypoints method, we use a Naive Bayes classifier, which assumes that the color values are mutually independent. The only difference is the distribution according to which the conditional probabilities are calculated.

Finally, the posterior probability vectors from SIFT vector classification and color classification have to be combined. In both cases all the classes are considered and therefore the probability vectors can be added together to form a new posteriori vector. However, the color values are much weaker than SIFT vectors and more sensitive to changes in illumination and to the quality of the camera sensor. Therefore we wish to give less emphasis to the color vector in comparison to the Bags of Keypoints vector. This can be performed by multiplying the color posteriori vector
Figure 14: The color values in the milk shelf illustrated in the image 14a are generally lighter than the ones in the beer shelf shown in the image 14b. The difference is greatest on objects from which most of the SIFT keypoints are extracted, and for this reason we use histogram values calculated around each keypoint.
by a weight scalar before addition. After addition, we use the maximum a posteriori decision rule.

In the following Section, we perform an experimentation of visual categorization with the Bags of Keypoints method in a real-life environment. We also apply the proposed extensions and analyze the yielded changes in the classification results.
5 Results

In this section, the visual categorization is applied to a problem of vision based indoor positioning in a grocery environment. This indicates that we wish to determine our location within a grocery store using visual information. We experimented the methods introduced in the previous sections in K-Citymarket Ruoholahti, which is a medium size grocery store. In this section, we first evaluate the results yielded by the Bags of Keypoints method without any extensions. Then we apply the proposed noise removal and color classification extensions and study their effects on the accuracy of the categorization. Before evaluation, we briefly explain the measurements used in the evaluation. As the location information is aimed to be used to offer the user information for example about the nearby products, special offers and suggest optimal routes for completing a shopping list, indoor positioning is sufficient to work on shelf level accuracy. This indicates that a prediction is considered correct, if we can predict the nearest shelf. Therefore we consider the grocery shelves as categories in the visual categorization.

Our domain consists of 18 grocery shelves containing various products. The contents of each class are presented in Table 1. Our training database contains 171 images with roughly 1000 SIFT features per image and each shelf contains 10 training images on average yielding over 150000 SIFT features. While the training images were taken by a systems camera, we created a test image database using Nokia N900 mobile phone to better suit a real world situation. The test images are taken from similar viewpoints and distances as the training images, covering random areas of the shelves. The test database contains 83 images taken evenly from all shelves covered by the training images. As suggested in [CDF+04] the $k$-means clustering was performed on 5000 features taken randomly from each class to hasten the calculations and to avoid biases. The classifiers are trained on the whole training data and the testing of the classifier was performed on all of the features extracted from test images. Furthermore, the clustering is repeated three times when a classifier is trained and the clustering result that yields the lowest sum of distances from features to cluster centers is chosen. As suggested in [CDF+04], we perform the $k$-means clustering using values 500, 1000, 1500 and 2500 for $k$. 


<table>
<thead>
<tr>
<th>Class</th>
<th>Contents</th>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Juice, butter, ice cream and alcohol beverages such as beer and cider.</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Both light and dark colors providing no significant color characteristics.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Breads and baked goods. Mainly dark colors and brown color dominates.</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Candies. Contains a wide range of colors.</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Pet food, pet toys and vitamins.</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>Cheeses. Yellow color dominates.</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>Fresh meat products, special sauces and canned fish. Red color dominates.</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>Conveniences foods. Red color dominates.</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>Milk products and hams. Light colors dominate.</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>Canned food, dry pasta and sauces.</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>Spices and rice. Light color dominates.</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>Flours and cereals. No dominating colors characteristics.</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>Cafe, tea and juices. Dark colors dominate.</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>Cookies, crisp breads, jams and dip sauces. No distinctive colors.</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>Soft drinks and chips. Lots of yellow and black colors.</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>Soft drinks, energy drinks, beer and sparkling water.</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>Special beers and other alcoholic beverages. Brown and blue colors dominate.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>Chips and chip sauces. Yellow colors dominate.</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>Vegetables. Green colors dominate.</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 1: The database contains image data from various locations within a grocery store. The index for each shelf is given in the first column and a short explanation of the contents of the corresponding shelf is given in the Contents column. The values of the last two columns denote the number of training images and test images for the corresponding class, respectively.
5.1 Plain Bags of Keypoints

First, we categorized the training images using plain Bags of Keypoints method, i.e. without taking colors into account and without removing noise. The experiment was conducted several times to avoid effects from randomness. The classifier with the average performance was selected and the result of the classification is given as a confusion matrix in Figure 15. The confusion matrix has the input labels plotted on x-axis and the predicted classes on y-axis, which indicates that the correct predictions are found on the diagonal. Even though the $33/83 \approx 40\%$ accuracy of the classification seems very low, we must take into account that as the highest amount of test images in one class is 7, which means that the probability of randomly guessing correct label for a query image is only $7/83 \approx 0.084 = 8.4\%$. Furthermore, some of the classes have identical or almost identical products, which makes distinguishing them extremely difficult. An example of this is given in Figure 16, where small bottles of soft drinks are incorrectly classified as a shelf containing similar soft drink bottles with different sizes.
On the other hand the classifier performs extremely well on some of the shelves. For example all the test images from shelves 9 and 18 are classified correctly and the samples from shelf 3 and 4 are classified correctly. Examples of the correctly classified samples are given in Figure 17. These shelves contain products with extraordinary appearances such as vegetables, dog bones and all sorts of bags of candies, which therefore yield characterizing distributions of SIFT features. The shelves containing common looking products, such as cereal boxes or cookie boxes, that are hard to distinguish yield similar characteristics making the classification often fail. A good example of this is shelf 10, which contains generic rectangular items with vapid labels that can easily be confused with many other products in the store.

5.2 Extensions

Our second experiment combines the noise removal method with the Bags of Keypoints approach and uses the same training and test datasets. By experimentation, the best results were achieved when choosing $E_{ps} = 2, E_{po} = \pi/4, E_{pe} = 0.08$. Unfortunately, the noise removal yields a slight decrease of 7% in accuracy, which indicates that five more test images were classified incorrectly in comparison to the plain approach. We can see from the confusion matrix given in 18 that in comparison to the plain approach, the incorrect classifications seem to spread more evenly
Figure 17: Examples of correctly classified test images. The image containing vegetables is easily distinguishable from other classes due to the unique visual characteristics of vegetables in general. The common looking labels on the products in the sample on the right hand side make the classification a bit more difficult, still providing enough distinctive information for the classifier to succeed.

to all classes. This suggests that in addition to removing noise, the clustering also removes a significant part of the meaningful feature vectors. This results in less distinctive power for the Bags of Keypoints, which again leads to additional false classifications. An example of this is given in Figure 19.

While reducing the accuracy of the classifier as a whole, noise removal does help in cases where lots of the features have appeared from non-interesting objects. An example of this kind of situation is given in Figure 20. The main problem with the noise removal is that it ignores most of the high level information of images such as resolution and the distance from the camera to the contents of the image. As the Euclidean distance threshold is constant over all possible viewing distances, the local neighborhoods are highly sensitive to the viewing distance. Furthermore the minimum amount of points needed for all clusters is constant over all resolutions. Lower resolution practically indicates less keypoints than from a similar image with high resolution making the clustering sensitive to image resolution. Solutions for these problems are discussed in Section 6.

Finally, we combine the Bags of Keypoints classifier and the color classifier introduced in Section 4.3. Unfortunately, adding the color information does not improve the accuracy of the classification either. The classification was performed with weights 0.1, 0.5, 1 and 2 for the color classifier. As giving more weight to the color classifier gives little emphasis to the more sophisticated Bags of Keypoints method, it is unsurprising that the performance began to decrease swiftly when giving a lot
Figure 18: The confusion matrix resulted from applying the noise removal on the categorization. The result was obtained using $k = 2500$ yielding 27 correct predictions out of 83.

of weight to the color classification.

One reason behind the weakness of the color classification is that the posterior distribution from the Bags of Keypoints classification is rarely smooth, indicating that one class is clearly favored. The posterior distributions from the color classification on the other hand are closer to uniform, as each image generally contains various different colors even if some color appeared more often than others. Therefore the color classifier requires a lot of emphasis to make a difference in the classification process. Another problem with our approach is that the multimodal color distributions are modeled with one dimensional Gaussian distributions yielding loss of information.
Figure 19: Many of the SIFT features removed as noise contained relevant features indicating that some of the characterizing information was lost and therefore the features remaining after noise removal are classified incorrectly.
Figure 20: Majority of the removed SIFT features have appeared from background structures and price tags. Therefore after the noise removal, the unlabeled data fits the corresponding class better and results in a correct classification.
6 Discussion and summary

In this thesis, we presented a method for tracking mobile devices in a retail environment by using visual information. We captured the visual characteristics of a Finnish supermarket by using both a systems camera and a Nokia N900 mobile phone’s camera. All the images were first processed by the Scale-Invariant Feature Transform SIFT algorithm which finds blobs from natural images that are invariant to linear transformations and changes in illumination. The blobs were located on multiple levels of scale to provide scale invariance and the errors caused by linear illumination and image noise were reduced by using the Difference-of-Gaussians representation. Gaussian derivatives of the image signal were calculated within the location of the blob to detect if the blob was incorrectly detected along an edge or from background clutter. The derivatives were also used to find a local orientation of intensity changes in order to provide invariance to image rotation. Finally the intensity changes within the blob were stored as a high dimensional real-valued vector from which the highest values were removed to gain invariance to non-linear illumination.

The SIFT features were captured from a wide collection of images from a medium size Finnish supermarket K-Citymarket Ruoholahti and were grouped up according to which shelf each image belonged to. To relate new feature vectors from unlabeled images to the ones in our database, we created a visual vocabulary using the $k$-means algorithm and connected each feature to the nearest cluster center. We modeled the shelves with the Bags of Keypoints representation, which corresponds to the number of features in each cluster from the set of features extracted from the corresponding shelf. We trained a Naive Bayes classifier with the Bags of Keypoints representations from the images taken from each shelf with the systems camera and used it to predict the labels of the images taken with the camera of the mobile phone.

We achieved a ratio of approximately 40% of correct predictions against incorrect predictions, which is unfortunately too small for most real world applications. We also introduced a noise removal procedure which removes keypoints that occur from non-interesting objects such as price tags. The idea is to detect groups of products using a density based clustering algorithm that leaves out outliers occurring from isolated objects. Unfortunately the noise removal did not generalize well and did not improve the general performance of the classification. We showed that noise removal performed well in cases where a lot of background was included and the distances of groups of points were high. On the other hand, the noise removal performed
badly on images where the features were distributed evenly throughout the whole image. Therefore we claim that the noise removal procedure could be improved by analyzing the image contents in beforehand and by deciding the minimum distance and minimum points parameters individually for each image.

Another attempt to improve the accuracy was to use the color histogram representation of images to give more emphasis to categories with similar color features. However, the color histograms are often multimodal and therefore we lose important information when modeling the with a single dimensional Gaussian distribution. Furthermore the Bags of Keypoints classification rarely ended up in a situation where two or more classes were evenly favored, which yields the color classification futile in our domain. However in easy cases, the color classifier yielded positive results which indicates that the approach could be applied in a different setting. For example if we tried to categorize objects in a retail environment where the wrappings and the packages of objects are quite similar and therefore the initial classification would result in more uniform distribution, the color classification would make a difference.

Combining our approach with existing methods is left as future work. For example WiFi based indoor positioning approaches could be used to restrict the possible locations from all shelves to a small group of nearby shelves. The vision based approach could then be used to choose the best candidate among the restricted group of shelves. With fewer classes, the complexity of classification would be significantly decreased, which would likely lead to higher accuracy.

References


