CHALMERS

Scene interpretation and object recognition for mobile robots equipped with cameras

Thesis for the Degree of Master of Science in Complex Adaptive Systems

JAKOB ANDERSSON

CAROLINE EBBESSON

Department of Applied Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2010

Thesis for the degree of Master of Science in Complex Adaptive Systems

Scene interpretation and object recognition for mobile robots equipped with cameras

Jakob Andersson and Caroline Ebbesson

Department of Applied Physics
Chalmers University of Technology
Göteborg, Sweden, 2010

Scene interpretation and object recognition for mobile robots equipped with cameras

JAKOB ANDERSSON, CAROLINE EBBESSON

Department of Applied Physics
Chalmers University of Technology
412 96 Göteborg, Sweden
Telephone: +46 (0)31 772 1000

Chalmers reproservice
Göteborg, Sweden, 2010

Abstract

A method consisting of separate steps for object detection and object recognition has been developed for use with digital camera images. The detection step uses double-opponent maps and the Hough transform to find likely object candidates without having to search the entire image. The recognition step is built around a modular neural network (MNN) trained using backpropagation into which colour histograms and eigenimage projection coefficients are fed as main cues along with simple region information. The detection approach was found to be 45 times faster than an equivalent brute force approach and the resulting algorithm is found to be fast enough in order to allow use in real-time robot landmark-navigation. Substantial differences in recognition ability between different types of objects were found when applied on images from a typical office environment. The performance of the algorithm indicates that visual object recognition can be achieved in real-time with only off-the-shelf equipment.

Acknowledgment

We would like to thank our supervisor Mattias Wahde and our friends and family without whom this thesis would not have been possible. We would also like to thank Friends of Chalmers for their financial support.

Contents

Chapter 1

Introduction

For a robot to operate in and interact with a changing and complex environment it has to be able to perceive and interpret its surroundings. Humans heavily rely on vision to achieve this. As cameras are becoming more affordable, the use of visual systems for scene interpretation in robots becomes more common. Scene interpretation is a complex task; the variation in the environment, the variation of objects, and the variation in form, colour, and texture of each object makes it hard to construct a general implementation. In order for robots to be used in everyday tasks, robust scene interpretation is necessary. However, many implementations today use expensive equipment such as laser range finders. Using a visual system built with off-the-shelf equipment such as cameras thus makes the robot more applicable in cost sensitive tasks.

The aim of this thesis is to take some initial steps towards a more general scene interpretation using cameras. Specifically, a robot operating in an office environment can navigate by using visually detected and identified objects as landmarks. In order to design an algorithm capable of solving this detection and recognition task, a literature study was conducted; a summary is presented in the following two chapters. One important design criterion is that detection and recognition can be done in real-time, in order for the robot to navigate while still moving. Another criterion is that it can detect a variety of objects. The implemented method has been selected to try to satisfy these criteria. This thesis attempts to accomplish this task and indicate if visual scene interpretation is suitable for use in navigation.

Chapter 2

Basic image processing and machine learning

In order to understand the concepts discussed in this thesis it is important to have some basic background knowledge of **image processing** and of **machine learning**. In this chapter different methods for information extraction from images will be introduced along with a brief introduction to machine learning in section 2.4. Both image processing and machine learning are huge topics and the aim of this introduction is to comprise the background knowledge needed to understand the rest of this thesis.

2.1 Basic methods for image information extraction

This section will introduce how object information can be retrieved from an image. The focus is on methods working on a local scale, i.e. only using information from close-by pixels, and in the image domain.

2.1.1 Feature detection

Feature detection is used to extract information, called features, from an image. The concept of **features** is very broad. In image analysis, it is used to describe properties, also called feature descriptors or cues, of a point in an image and its surrounding pixels. Common feature descriptors are intensity, orientation, colour, and texture. In cases where several feature descriptors are required, they are organised as elements in a feature vector. Features are used to reduce the amount of data prior to further image processing. It is important that features are robust, i.e. that they are, at least partly, invariant to scale, rotation, translation and illumination changes and thus easily detectable under different conditions regarding illumination, viewing angle, etc. The most basic feature detectors are **edge** and **interest point detectors**; they locate sharp variations of intensity, which correspond to depth discontinuities, surfaces discontinuities, and shadows.

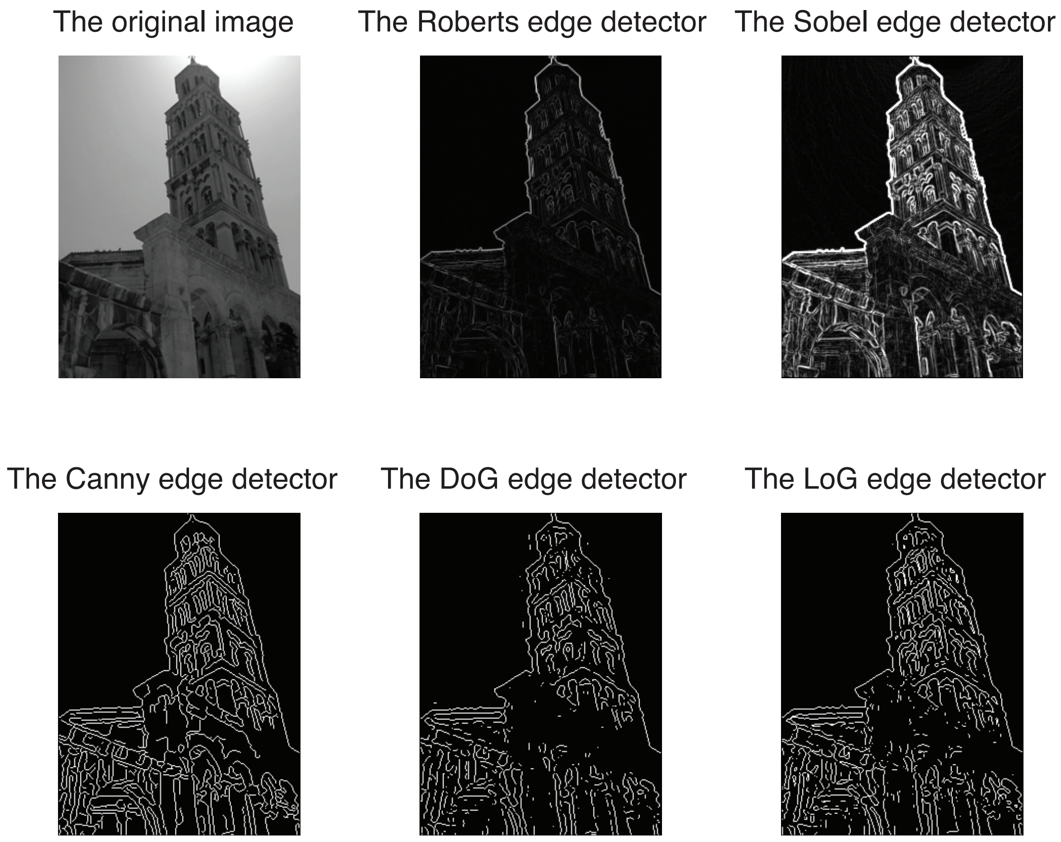

Figure 2.1: *A comparison of five different edge detectors: the Roberts edge detector, the Sobel edge detector, the Canny edge detector, the Difference of Gaussian edge detector, and the Laplacian of Gaussian edge detector. Image by the authors.*

Basic edge detectors, e.g. the **Sobel edge detector** [62] and **Roberts edge detector** [49], measure the intensity gradient near a pixel by applying a set of convolution masks. Some examples of slightly more complex edge detectors are the **Canny edge detector** [6] and detectors searching for zero crossings of the second order derivative over a set of image pixels, like the **Laplacian of a Gaussian** (LoG) method [39] and the **difference of Gaussians** (DoG) method; the latter is an approximation of the LoG method. A comparison of these five edge detectors can be found in figure 2.1.

Most edge detectors output fragmented edges due to noise and variations in illumination. For most applications, it is of interest to complete and link these fragmented edges to object boundaries. This can be done (i) locally, by examining properties like intensity and edge orientation for neighbouring pixels and filling gaps between pixels that have the same properties, (ii) regionally where edge pixels are linked according to their regional membership or (iii) globally using the Hough transform; see section 2.2.2.

Interest point detectors, often called **corner detectors**, localise points at which the intensity varies sharply in several directions. Some examples are the **Harris–Plessey cor-**

ner detector [18] and the **Smallest Univalue Segment Assimilating Nucleus corner detector** (SUSAN) [61]. With modifications, both the Harris–Plessey detector and the SUSAN detector can be used as edge detectors. Edge and corner detectors can be used directly as feature detectors, but are also used as a first step to localise interesting areas in more complex detectors such as the SIFT detector [36] described below.

2.1.2 Scale-invariant feature transform

The **Scale-invariant feature transform** (SIFT) is a feature detector developed by Lowe in 1999 [36]. It transforms an image to a large set of local feature vectors or SIFT-keys that are invariant to scale, rotation, and translation and partly invariant to illumination changes and viewpoint. SIFT uses interest points, called key points. Scale-invariant key point candidates are obtained by examining the image at different resolutions using the difference of Gaussians method. For a more detailed description of how this is done, see Lowe's article [36]. In order to keep only the points that are robust against noise, points with low contrast and points only corresponding to an edge are discarded. The key points left are assigned orientations based on the local image gradient to obtain rotational invariance. For each key point, the local feature vector is computed from the neighbourhood of the key point.

2.1.3 Stereo vision and depth estimation

The human brain uses binocular vision for depth perception. This has inspired the development of **stereo vision**, i.e. the use of two calibrated cameras, separated by a known distance, for depth estimation. The two cameras will have a slightly different view of the scene and objects will thus be projected at slightly different image positions. The difference is called **disparity** and is obtained by a process called **stereo matching**. By knowing the distance between the two cameras, the distance to the object can be calculated from the disparity. An object near the cameras will have a larger disparity than an object farther away. Since disparity depends on distance, depth estimation using stereo vision only works to a specific distance, beyond which the disparity becomes smaller than one pixel.

However, the human brain does not only depend on binocular vision for depth perception. It also uses **monocular cues** such as texture gradients, familiarity with detected objects, perspective, haze, etc. These can be used in addition to stereo vision to obtain a better distance estimate or to obtain a depth estimate from a monocular system [54, 56]. Depth estimation is an important part in object recognition, since it provides information about occlusion and size and since it can be used to create a three-dimensional model of the scene that can be matched to three-dimensional object models.

2.1.4 Template matching

A **template** is a representation of an object, such as a set of features, extracted from an image or from a three-dimensional model of the object. When using features, feature detection is performed on the search image before matching.

In **template matching** [14], an object is considered recognised when correlation between the template and a part of the search image exceeds a certain threshold. The template is shifted over the search image, one pixel at a time. For each position, the correlation is calculated and a correlation map is generated. If the correlation is too small, the object is not present in the image. If, on the other hand, there are multiple positions exceeding the correlation threshold, several matches are found.

This method is easy to implement but computationally expensive, especially, if the method should be able to recognise several objects at different angles and resolution, something that requires a large set of templates. Speed can be improved by the use of an **image pyramid** [5], which consists of a set of images of different resolutions. The pyramid is created by down sampling a high-resolution image creating a set of reduced resolution images. The low-resolution images are used with a down sampled template to obtain areas likely to contain the object. Only these areas are then scanned with the template in the high-resolution image.

2.2 Image processing using transform methods

The previous section introduced information extraction in the image domain; this section will show additional methods for extracting information in a transform domain. There exist, of course, many other transform methods that can be used which are not presented here, such as the **Fourier transform**.

2.2.1 Using wavelets for feature extraction

Frequency analysis of an image can generate information that is not easily obtained in the image domain. A common way to obtain frequency information is to use the Fourier transform; however, it only returns the frequencies in the image and not any spatial information. The **wavelet transform** [38, 42] is designed to return both frequencies and their location.

First, consider a wavelet transform, $X(s,\tau)$, of a one-dimensional time signal, *x(t)*. The wavelet transformation multiplies the signal with window functions, the **wavelets**, to transform the signal into the frequency domain. A wavelet is a short oscillatory function, hence the name wavelet – small wave. The continuous wavelet transformation can be seen in equation (2.1). Each wavelet is derived by scaling and translating a function called the **mother wavelet**, $\varphi(t)$. The **translation factor**, τ, shifts the wavelet over the signal and the **scale factor**, s, dilates or compresses the wavelet, changing the width of the window. With a low scale-factor, the wavelet is compressed, giving a higher resolution in time. With a high scale-factor, the signal is dilated, giving less resolution in time. Since the scale factor is proportional to time, it is inversely proportional to frequency.

$$X_\omega(s,\tau) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} x(t)\varphi(\frac{t-\tau}{s})dt \tag{2.1}$$

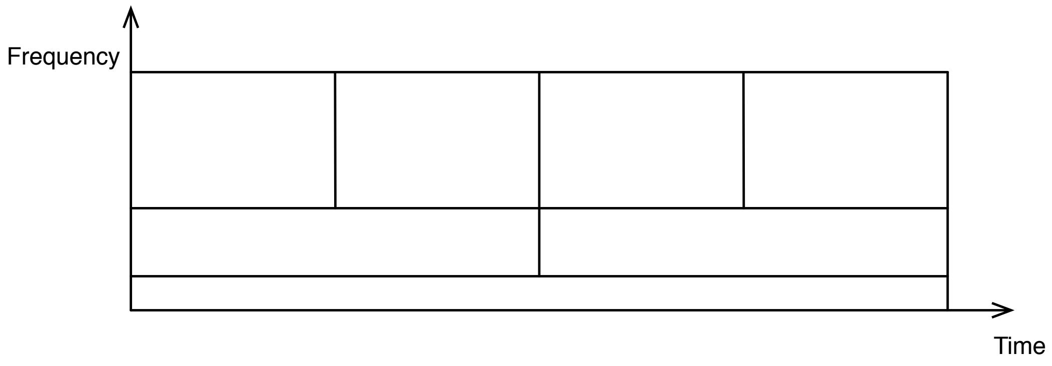

Figure 2.2: *The time frequency resolution for the wavelet transform. As can be seen high time resolution gives low frequency resolution, and low time resolution, high frequency resolution. The boxes have the same area and the minimum size is restricted by the type of wavelets used.*

It is impossible to know the exact frequency at a specific time, cf. Heisenberg's uncertainty principle, and hence full frequency resolution can only be obtained if the signal stretch out to infinity, but then the resolution in time is lost, as for the Fourier transform. The wavelet transform acts only on small windows of the signal, due to the finite length of the wavelet functions, reducing the frequency resolution and increasing the time resolution. The wavelet transform gives high time resolution for low frequencies and low time resolution for high frequencies, see figure 2.2. The product of the size of the time intervals and the size of the frequency intervals is always the same and is restricted by the type of mother wavelet.

The discrete version of the wavelet transform passes the signal through a series of high and low pass filters. The signal is first split into a high frequency part and a low frequency part. Then one of the parts, usually the low frequency part, in order to get better time details, is split again and the process is repeated a predefined number of times.

In image processing, the **two-dimensional spatial domain** is used instead of the time domain but the approach is the same, i.e. giving a series of images with different details. An example of a wavelet transformation, or **wavelet decomposition**, of an image can be seen in figure 2.3. As for the Fourier transform, the wavelet transform, gives the coefficients for a series expansion in wavelets of a signal or image, called wavelet series. These coefficients can be used as scale invariant features for an object [45]. A commonly used family of wavelets in image processing are the **Haar wavelets** [17], mostly used for their simplicity.

2.2.2 Shape extraction using the Hough transform

In the beginning of the 1960, Hough developed the **Hough transform** to identify particle trails in bubble chamber images [25]. It identifies lines by calculating the likelihood that a specific line is present in the image. A line, $y = kx + m$, is normally described by two parameters, the slope, k, and the y-intercept, m. The parameters k and m are then

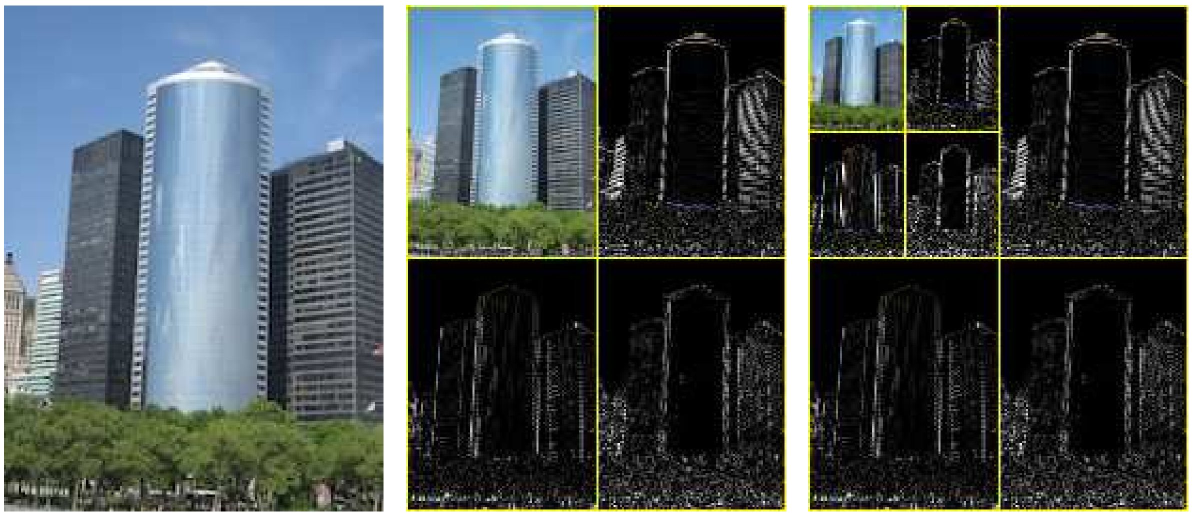

Figure 2.3: *An example of a wavelet transform. The left panel shows the original image. The middle panel shows the first level of wavelet decomposition: the upper left is the approximation coefficient, the upper right the horizontal coefficient, the lower left the vertical coefficient and the lower right the diagonal coefficient. The right panel shows two levels of decomposition, with the approximation transformed in the second step. Photo by the authors.*

unbounded for vertical lines. In the Hough transform, a line is instead described by equation (2.2) with the parameters r and θ, which are bounded for all lines. The definition of r and θ can be seen in figure 2.4. The value of r is the length of the normal vector from the origin to the line and θ is the angle from the x-axis to the normal vector and has positive values in the clockwise direction.

$$y = -\frac{\cos\theta}{\sin\theta}x + \frac{r}{\sin\theta} \quad \text{or} \quad r = x\cos\theta + y\sin\theta \tag{2.2}$$

In parameter space, a line in image space is represented by a point in the (r, θ)-plane. Every line going through a point in image space will belong to the same sinusoidal curve in parameter space; see figure 2.5 and equation (2.2). If two points in image space belong to the same line, their respective sinusoidal in parameter space will cross and the point where they cross gives the parameters for that line.

The Hough transform creates a map of the likelihood for a line to be present in an image. The map is a matrix where the rows correspond to the r-parameters and the columns correspond to the θ-parameters. The value of an element represents the likelihood of the line described by the parameter pair, (r, θ), corresponding to the element. If a pixel in the image corresponds to an edge point, the parameters for all lines that go through that point are calculated. Then the value of the elements representing the parameters for those lines is increased by one. The most likely lines that are present in the image are then given by local maxima in the likelihood matrix, see figure 2.5.

The Hough transform has been generalised to identify arbitrary shapes using the same approach but other parameters [1]. It can be used for object recognition by finding the

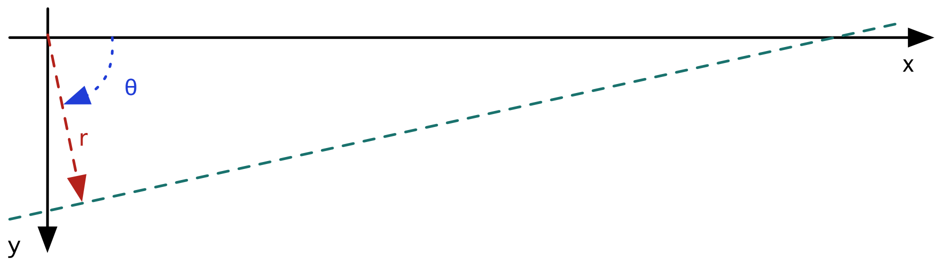

Figure 2.4: *Here the Hough parameters r and θ are shown. The value of r is the length of the normal vector from the origin to the line and θ is the angle from the x-axis to the normal vector, counted positively in the clockwise direction.*

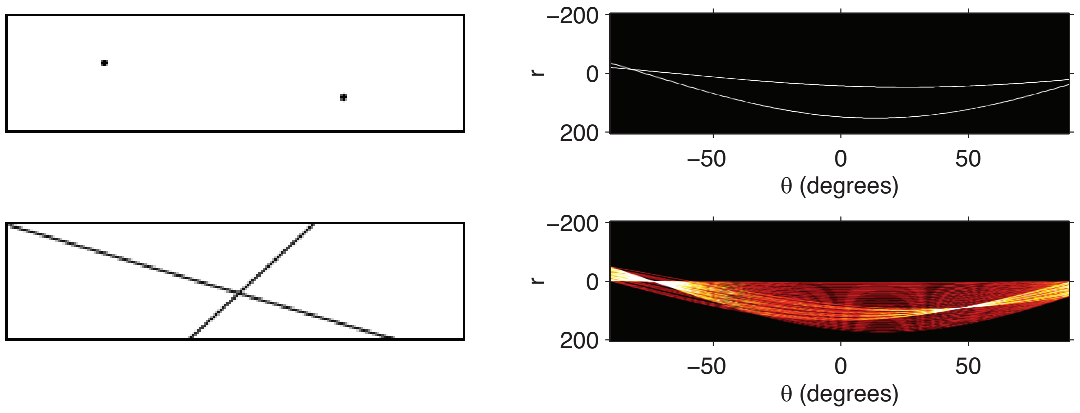

Figure 2.5: *The upper right panel show the Hough transform of the two points in the upper left panel, which is sinusoidal. The lower right panel show the Hough transform of the two crossing lines in the lower left panel. It is easy to distinguish the two maxima corresponding to the parameters for the two lines.*

parameters that maps an object model into the image and using them to determine the objects position in the image.

2.2.3 Feature detection with eigenimages

Images of any instance of an object class can be considered a combination of **eigenimages**. A set of eigenimages consists of eigenvectors selected by performing **principal component analysis** – see section 2.4.5 – on eigenvectors extracted from a large set of images of different instances of the object class. [34, 43]

Eigenimages are mostly used in face recognition under the name **eigenfaces**. Even a small set of eigenfaces gives a fair approximation of a face. However, for the recognition method to work well the face images need to be taken with the faces in the same position and with similar lighting.

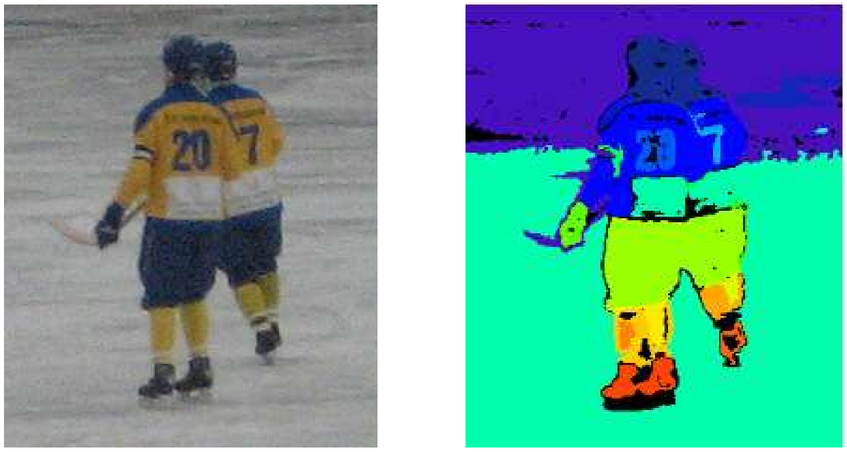

Figure 2.6: *The left panel show an example on a typical difficult segmentation task. The clothing with their distinctive colours will be recognised as different segments by almost any segmentation algorithm. The right panel shows an example of such a segmentation. Photo by the authors.*

2.3 Global image features

Hitherto, local image information cues have been described, but much information remains to be extracted on a global image scale. Such global image features will be introduced in this section.

2.3.1 Segmenting the image

Global image features are often helpful in determining the real world origin of each pixel. Dividing the image into several **segments** for separate classification is also an alternative approach to labelling the image pixels individually. The concept of image segmentation is intuitive in its purpose and benefit, but after a closer inspection, the task of segmenting an image turns out to be far from straightforward. What at first is an obvious segmentation often is both ambiguous and unstable to changes in image data and in the choice of segmentation algorithm. For example, the persons in figure 2.6 are made up from different homogeneous regions, which are hard for an algorithm to cluster. The segmentation is at the same time ambiguous since grouping the regions into one region is correct in some situations and not in other. Ambiguities as these make training of a classifier much more complex since no ground truth segmentation exist. These difficulties can be circumvented by use of multiple segmentation algorithms to form a probability distribution or by limiting the segmentations to basic segment types, e.g. ground, sky, etc. Both these approaches have been applied in object detection tasks in the past. Segmentation is many times used instead of searching the entire image with a moving window. Cleverly chosen, image segments are often easier to analyse than an arbitrary window of pixels and they can aid in object detection by setting features in context.

2.3.2 Colour as a cue for segmentation and object recognition

Using **colour** as a cue for object recognition can be very powerful. Many human-designed objects have distinctive colours, e.g. signs, fire extinguishers, etc., and are thus easily located by such cues. It is also powerful in characterising different scenes or image parts such as vegetation, sky, or water. Colour alone is, however, not enough except in very limited applications since many common objects are not characterised by it, e.g. cars, clothing, and furniture.

Colours can be represented in many different ways, not only by the commonly used RGB colour space. In fact many other representations such as HSV, L*u*v*, and YCbCr exist with their own strengths and weaknesses. Colour information is usually quantified as a histogram for a part of the image. Histograms contain no spatial information and are thus invariant to rotational, translational, or elastic transformations. Colour statistics is, however, sensitive to changes in illumination. The illumination sensitivity can be reduced by utilising different techniques, e.g. energy normalisation [57] or weighted histograms [19].

2.3.3 Information from texture

Some objects, e.g. spotted cats and brick walls, are almost perfectly described from their **texture** alone. Other objects are on the other hand not characterised by texture at all, e.g. cars and human clothing. This ambiguity makes the decision on whether to use texture information or not in a classifier dependent on the context in which the classifier is applied, i.e. even if the algorithm can be made good enough for all cases it may not be worth the extra computational cost. In using texture information, one has to account for information at different scales dynamically, since textures at different distances will be represented at different scales. One also has to avoid misclassification at true image boundaries. Textural information can be characterised by colour histogram or by a filter response. This characterisation is usually applied on small patches, either from a segmentation algorithm or from a moving search window.

2.3.4 Improving detection by contextual information

Image **context** is a powerful cue for object identification. Context information can be of different types; it can be inter-object relation where a computer screen indicates a high probability for a keyboard close by, or it can relate object candidates to their position within the scene, e.g. that cars are not found on rooftops. Context can also communicate information of the entire scene, i.e. the information on where the image is taken, e.g. in an office or at a tennis court. Regardless, the detection algorithm receives knowledge of probabilities for object occurrence and thus improves both detection scores and algorithm speed. The relationship between objects can be found from analysing a large set of pictures, and thus be well tuned for a specified environment, or from some other source such as semantic relationships.

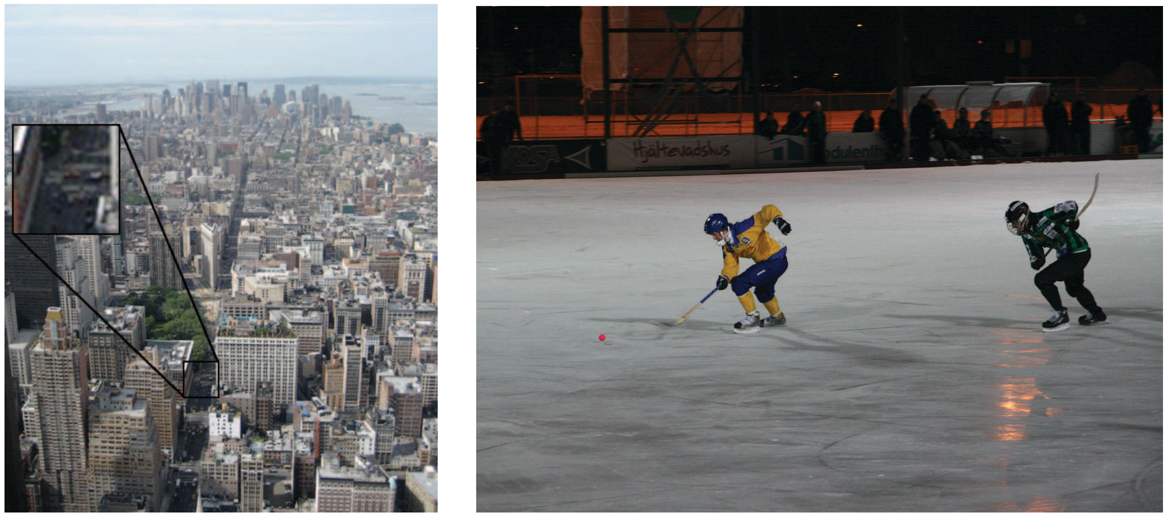

Figure 2.7: *In the left panel, a view of New York from 2009 is shown. For a human it is not difficult to see that the pixels on the streets represent cars, but when the pixels is enlarged and removed from their context the resemblance is less clear. In the right panel, it is equally easy for a human to identify the ball, but without context, that task would be next to impossible. The left photo by the authors and the right photo by M. Ahlberg. Reproduced with permission.*

Using contextual information is important in object-detection software aiming at handling many different object types in versatile environments. For a human, it is easy to see that the pixels in the streets in figure 2.7 represent cars or that the pink pixels in a picture from a bandy game are a ball and not a fruit, but for a computer with no contextual information that task is almost impossible. It should be noted that context information has the drawback of requiring true objects in unusual locations to be more salient than what would otherwise be necessary for detection. This however agrees with psychophysical results in humans, e.g. both humans and algorithms using contextual information will have more difficulties detecting a person standing on a rooftop than on a sidewalk. [24]

2.4 Machine learning and related techniques

In this section, some main concepts in machine learning and related techniques important for understanding the notions of this thesis are briefly introduced. It should, however, be noted that this selection is limited and that many more concepts exist which will not be considered in this text. A review of common techniques in pattern recognition is given by Jain *et al.* [28].

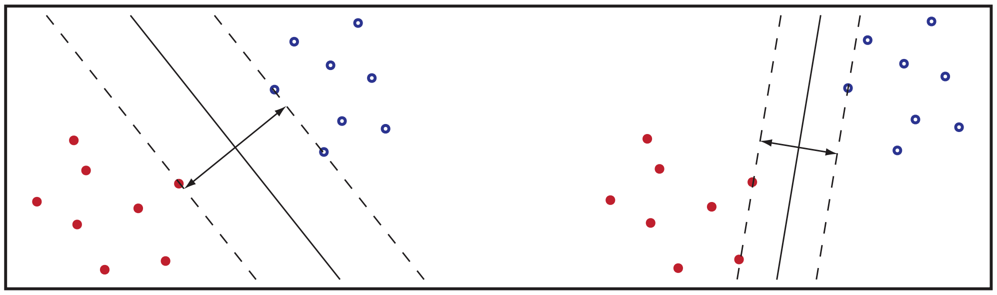

Figure 2.8: *Two different data classes (the dots and the circles in the image) separated by a hyperplane. In both the left and the right view, the two classes are divided by the hyperplane. However, in the left view the margin between the two data classes is maximised, i.e. the generalisation error is minimised.*

2.4.1 Probably approximately correct

Probably approximately correct (PAC) is a framework for theoretical analysis of machine learning theory introduced by Valiant [64] in 1984. If a **learner** is able to learn a concept approximately, i.e. with a low generalisation error, with a high probability from an arbitrary distribution of examples, then the concept is said to be PAC learnable. The PAC framework gives an upper bound on the number of samples needed for learning the concept with an error smaller than ε and a probability of at least $1 - \delta$.

2.4.2 Support vector machines

Support vector machines (SVM) introduced by Vapnik are methods for supervised learning in high dimensional space popular in machine learning and classification tasks. The concept of SVM will only briefly be introduced here and a more thorough introduction can be found in Vapnik's own book on the subject [65].

Consider a set of data points belonging to two classes, a hyperplane that maximises the **margin** between the two classes is then constructed; see figure 2.8. By maximising the margin, the generalisation error is minimised. Most of the data points will be redundant and thus not contribute to the solution. The points, which do contribute, are called **support vectors**. However, many real life problems cannot be solved by a linear classifier and the original SVM is therefore extended to use any positive definite, symmetric **kernel function**. Depending on the choice of kernel, other learning algorithms such as radial basis functions and two layer neural networks can be constructed.

2.4.3 Neural networks

The human brain consists of a large complex network of neurons. The artificial neural network or **neural network** emulates its biological counterpart with artificial neurons. Neural networks are a huge topic and will only be introduced briefly here, for a more thorough introduction see a textbook on the subject [21]. A neural network consists of one or more layers of neurons linked to each other by weighted connections. The **weights** are adaptively assigned during **training** and they determine how data propagate through the network. Each neuron takes several weighted inputs, either from direct input or from another neuron, and passes their sum through a, usually, non-linear activation function to form an output. In simple neural networks, information only propagates forward, but in incarnations that are more complex, neurons are allowed to connect in loops. Many different forms of neural networks exist, e.g. Hopfield and Kohonen networks, each with different strength and weaknesses.

2.4.4 Boosting techniques

Boosting is a set of techniques for supervised learning in the PAC framework. The development of boosting algorithms originated from the work by Kearns [29]. He questioned whether a set of **weak learners**, each only slightly better than random guessing, could be combined to form a single **strong learner**. The concept, which gives a positive answer to the question of Kearns, is to build an array of weak learners that each carve off some of the candidates not belonging to the class. Such weak learners need to have a very high detection rate, but can afford a high false positive rate since both the detection rate and false positive rate multiply, see equations in 2.3.

$$P_{detection} = \prod_{n=1}^{N} P_{detection}^{n} \quad , \quad P_{false} = \prod_{n=1}^{N} P_{false}^{n} \tag{2.3}$$

Boosting comes in many forms, on which extensive literature can be found [13, 66]. The most used boosting method today is **AdaBoost** introduced by Freund [12]. The difference between various boosting techniques is most commonly their way of weighting training data. Analysis made by Brubaker *et al.* [3] show that different boosting techniques generally give comparable detection rates, but have, depending on the weak learners applied, significantly varying running speed.

2.4.5 Principal component analysis

Principal Component Analysis (PCA) is a mathematical technique for dimensionality reduction with minimal loss of information. In PCA, the data is decomposed according to the eigenvectors corresponding to the eigenvalues of the data covariance matrix. These eigenvectors are orthogonal to each other and are called principal components; the vector corresponding to the largest eigenvalue, i.e. the first principal component, accounts for most of the variation in the data. Each successive component then accounts for less information. Thus, dimensionality can be reduced with minimal loss of information by throwing away all but the first few components. For a more detailed explanation on PCA, see a textbook on linear algebra [31].

2.4.6 Methods inspired by human visual cortex

Even after recent advances in image understanding techniques and increase in computational power, humans and primates outperform the state-of-the-art detection systems in almost every measure. Building a system mimicking the features of the visual cortex is thus an attractive idea. Serre *et al.* [60] aimed at imitating higher complexity processes in the visual cortex in combination with more established techniques inspired from the first stages of the visual cortex, i.e. Gabor filters. Their results showed both that such features could perform comparably with state-of-the-art systems and that they could be used in conjunction with standard image processing techniques.

Chapter 3

Related work

A great deal of work has been carried out on scene interpretation in the past. Research has been done in classification algorithms, cue extraction, and cue combination to name a few topics. Today one has many well-explored techniques to choose from when constructing a scene interpretation system. However, the literature does not show any signs of convergence; there are no signs indicating any best method or any most-important cue. The systems built in the past use many combinations of cues and learning algorithms; to synthesise what has been done, as is the aim for this chapter, is therefore not trivial. The first section will describe the different aspects of work done on scene interpretation generally. Specific approaches in indoor scene interpretation for robots, i.e. the focus of this thesis, will be discussed in the second section.

3.1 Different aspects on scene interpretation

Scene interpretation is a broad topic and it has been approached in many ways in the past. A large part of the literature focuses on developing techniques [10,11,66], rather than applications. Of the part that focuses on identifying objects in real-life applications some main areas can be recognised. Image understanding has made an important contribution in automotive applications [16, 33, 44, 46] and in surveillance [3, 8, 35, 50, 58] where enough demand and applicability have spurred development. Much work has also been done with image understanding techniques in robotics. Applications range from grasping objects [48] and opening doors [30] to **human-robot interaction** (HRI) [51] and navigation [7,54,68]. Scene interpretation can be applied in many more areas, such as image compression where important objects can be preserved in better quality than unimportant objects [47]; the description of those topics is however beyond the scope of this text.

3.1.1 Automotive applications

The extent of the work done with support from the automotive industry or with the aim of detecting pedestrians and vehicles is natural since the demand for innovations improving safety is high. Understanding of the scene in front of a vehicle is important in several aspects; detecting moving objects to aid the driver avoid collisions is maybe the most important of them. Leibe *et al.* [33] approached the problem by combing several different cues, e.g. interest points in Hough space, colour, and edges, to identify objects in images taken from a moving vehicle and to track the objects in order to predict and prevent collisions.

3.1.2 Applications in surveillance

In addition to automotive applications, building systems able to detect persons and vehicles in images is common. However, most work [3, 8, 58] describing such an approach do not make any connection to surveillance applications, instead, they use the approach to test algorithms. However, some work aim directly for surveillance applications; Roth *et al.* [50] designed a system able to detect people in a video stream and Levin *et al.* [35] designed a system able to count vehicles in road surveillance videos. Regardless of the aim, the requirements are the same.

3.1.3 Other aspects on scene interpretations

The largest part of the literature in object recognition and scene interpretation focuses on developing the techniques rather than on building serviceable systems. Pre-existing image sets are commonly used for simplicity, but mostly because of comparability. Such image sets usually consist of a broad range of motives; the systems using them can therefore not be labelled as some specific application. In addition, the broad range of features and classification algorithms available results in almost as many possible combinations as attempts.

The focus of this thesis is, however, in robotics and in particular in indoor scene interpretation. Object recognition is, of course, important in such applications and there have been many implementations. The next section will give a review of the work done in that domain.

3.2 Indoor scene interpretation for robots

In indoor robotics, there are three main applications where scene interpretation and object recognition are essential: **human–robot interaction** (HRI), **robot–object interaction**, and **navigation**. These different applications have different approaches to object recognition as the information is used for different purposes.

3.2.1 Human–robot interaction

Interaction between humans and machines is becoming increasingly important, since robots today operate closer to humans and are able to perform tasks that are more complex. Many such HRI applications build on robots, or other machines, being able to detect, track and recognise human faces. For **face detection**, most methods use skin detection, segmentation and facial features; two examples are Hsu *et al.* [26] and Sandeep and Rajagopalan [53]. Of course, there are other methods; surveys can be found in, Yang *et al.* [70] or Hjelmås and Low [23]. Hsu *et al.* [26] found that using eye and mouth maps together with skin detection and segmentation drastically lowered the number of false detections, though it also decreased the detection rate. Sandeep and Rajagopalan [53] used edge information to group skin pixels into regions and used a width and height ratio to determine which regions were faces. They concluded that using edge information increased the detection rate while not heavily affecting the computation speed.

In **face recognition**, the aim is to identify to whom a detected face belongs. Two common approaches for face recognition are to use eigenfaces [63] – see section 2.2.3 – or to use facial feature positions [69]. Turk and Pentland [63] proposed an algorithm where eigenfaces were used for real-time face recognition. Wiskott *et al.* [69] represent faces as grids with facial features at the nodes and the distances between the nodes on the edges. Both methods have shown high recognition rates on frontal views; the latter was also tested on profile views, but with large decrease in the recognition rate compared with the frontal views. For a survey of facial recognition methods, see Lu [37] or Zhao *et al.* [71]. These methods are however not only used in robotics. For example, Rowley *et al.* [51] built a system using neural networks to identify faces and concluded that such a system could be used in media applications, e.g. search engines, to find faces automatically.

3.2.2 Robot–object interaction

In order to operate in many indoor environments and in order to accomplish many tasks, a robot needs to be able to grasp and manipulate objects. For robot–object interaction, it is not only necessary to identify an object, but also to identify its pose and position. Saxena *et al.* [55] and Rusu *et al.* [52] have proposed methods to determine object orientation; the latter was able to identify and orient common tableware with high accuracy. Their robot was equipped with a stereo camera and used clouds of point-features to create a three-dimensional model of the objects from which the object orientation could be determined. Saxena *et al.*, on the other hand, used symmetries to determine object orientation. While they found the problem difficult to solve completely, they identified the orientation correctly enough to let a robotic arm grasp several different types of objects.

An important task for a robot supposed to navigate an office or a hospital, is to identify and open doors. Ng and colleagues [30, 48] constructed a robot able to open doors by manipulating the handles. Their robot used a laser scanner to find the doors and a visual detection system to detect and recognise the handles. The robot used SVM and PCA – see section 2.4.2 and 2.4.5 – to locate and classify the object features; it also used contextual information by taking into account that most door handles are located in waist height. When a handle was found, the robot determined the action based on a classification of the handle type. They tested the robot in a new environment with previously unknown doors and it managed to open most of the doors. However, it had difficulties with glass doors, doors with number pads and dim lighting conditions.

3.2.3 Visual robot navigation

In robot navigation and localisation and in object avoidance it is common to use laser range finders, sonars, or infrared sensors to determine the robot's position and plan a route. However, since digital cameras have become very cheap, more research is being done in usage of visual cues. For localisation, such a visual approach is to use landmarks, found either by object recognition [7,20,40] or by matching features directly [59,68]. When using objects as landmarks, generating landmarks is a more supervised process than using features directly, i.e. when using object recognition, the possible object types are chosen *a priori*.

Making robots localise themselves can be approached in several different ways; Werner *et al.* [68] used a robot with a panoramic camera to obtain colour histograms and Se *et al.* [59] used a trinocular camera system to find SIFT features. Werner *et al.* had their robot determine its position by finding a match among colour histograms connected to positions from a previous mapping; similarly, Se *et al.* had their robot matching SIFT features. Both groups achieved good results on local localisation, i.e. they improved odometry, but Se *et al.* also achieved good results in global localisation, i.e. their robot could find its position from just a map without any other previous knowledge.

Mata *et al.* [40], Celaya *et al.* [7] and Hayet *et al.* [20] taught robots to recognise both artificial landmarks, specially made to aid in robot navigation, and natural landmarks, such as office signs. Mata *et al.* used a pattern recognition method that segmented the image into regions of interest and then used a set of genetic algorithms to determine if the regions were real landmarks or not. Similarly, Hayet *et al.* [20] used natural quadrangular landmarks, e.g. posters, found using edge detection. To aid landmark recognition, Hayet *et al.* used the Harris interest point detector to extract features within the landmark and then compared – by Hausdorff distance – detected landmarks to landmarks in a database. Both these methods managed to detect landmarks accurately, even at large viewing angles.

These were just a few examples of approaches in visual landmark detection. Robot navigation is a large topic and visual landmark navigation is just a part of it. For a more detailed survey on visual robotic navigation, see DeSouza and Kak [9].

Chapter 4

Method

A computer program for scene interpretation has to find objects in cluttered environments as well as determine the type of each object. This section will describe the algorithm implemented in this project along with some background theory regarding the employed methods. The selected structure revolves around two main parts: one part that finds regions of interest and one part that decides if the regions contain any object or not. However, before the method can be described in more detail, the environment in which the program should be applied and the tasks it is intended to solve has to be discussed in more detail.

4.1 Selection of landmarks

The kind of object that should be located is a key factor in the method design. Different objects are distinguishable by different kinds of features. In this project, the objects recognised are intended for indoor landmark navigation, which limits the set of available objects. For an object to suit as a landmark, it has to have a fix position and be easy to distinguish from the background. In indoor environments, emergency equipment is one such group of objects, since they are designed with distinct colours made to stand out. Except for their colour and usually fixed position, they also have the advantage of being homogeneous in colour and texture, which makes them easy to segment.

There are also objects that do not stand out in colour, that still are interesting to use as landmarks. In this project two such groups of objects has been chosen, doors and objects that stand out in intensity with a homogeneous texture. Doors are of great interest in navigation; there exists doors that are distinguishable in colour and intensity, but they are almost as often made to blend in with the background and doors are thus treated as a separate group. A list of the chosen landmarks and the feature used to locate them can be found in table 4.1.

Table 4.1: *A list of the chosen objects and the features used to locate them.*

Objects	Feature
Emergency exit signs	Colour
Fire alarm buttons	Colour
Red signs	Colour
Card readers	Intensity
Doors	Vertical lines

4.2 Locating regions of interest

The camera takes an image of a large part of the room and it is therefore necessary to locate interesting parts of the image in order to isolate regions with a potential object that can be used as input to the object recognition algorithm. The simplest way to do this is to go through the image from top to bottom and around each pixel extract a rectangular region as an image segment, for different region sizes. This method is guaranteed to find image segments containing each object completely. The problem with this approach is that it results in a great amount of image segments, most of which does not contain any object. An image with n^2 pixels and a region size of m^2 can result in $n^2 - 2mn$ image segments for one region size. For example, an image with 10,000 pixels, a region size of 100 pixels will give 8,100 image segments. Even if only every hundredth position is used and if only ten different region sizes are used, the image will result in approximately 800 image segments for this small image. This will of course severely damage performance. It is therefore interesting to locate regions where the probability for an object to exist is large, i.e. **regions of interest**. By only extracting image segments from these regions, the number of image segments are reduced and thus also the computation time. It should be noted that there is a possibility that an object will not be included in the regions; however, the performance advantages outweigh the disadvantages and the navigation task can still succeed without finding every landmark.

4.2.1 Generating saliency maps

There are several ways to locate regions of interests; Mata *et al.* [40] use a segmentation algorithm based on thresholding in the HLS colour space as well as space morphological transformations[1], where regions of interest are found as blobs with previously defined geometries. Both the thresholds and predefined geometries are optimised by training the segmentation algorithm. Gould *et al.* [15] create a visual attention map by calculating the probability that a pixel belongs to an unknown object in any of the known object classes.

[1]Morphological transformations are a part of morphological image processing which in essence is built on set theory. It is mostly used for binary images. Some examples are erosion, dilation, hole filling and border extraction. For more information, see a book on image processing, for example *Digital Image Processing* by Gonzalez and Woods [14].

In this project, regions of interest are found by using saliency maps, inspired by the work of Itti *et al.* [27] on visual attention. They use feature maps, which are combined to a single saliency map. Their method is inspired by **double opponent cells** in the primary visual cortex; they detect colour and intensity contrast, thus areas that stand out in colour or intensity. There are three types of double opponent cells: two for colour, viz. red–green and blue–yellow, and one for intensity. The colour double opponent cells compare the relative amount of each colour in the centre of their receptive field with the amount of the opponent colour in the outer rim of the receptive field, generating the largest response for local colour contrast, e.g. green adjacent to red. The intensity double opponent cells work essentially the same way, but measure the intensity instead.

To generate the colour double opponent maps, Itti *et al.* use four colour channels, corresponding to red (R), green (G), blue (B) and yellow (Y), calculated from the RGB channels (r,g,b), pixel by pixel according to:

$$\begin{aligned} R &= r - (g+b)/2 \qquad (4.1)\\ G &= g - (r+b)/2 \qquad (4.2)\\ B &= b - (r+g)/2 \qquad (4.3)\\ Y &= (|r+g| - |r-g|)/2 - b \qquad (4.4) \end{aligned}$$

Negative values are set to zero. To generate the double opponent colour maps Itti *et al.* use Gaussian image pyramids – see section 2.1.4 – made from the colour channels, by comparing the colour channel values in the same area at different levels of the pyramid.

In the implementation in this project, contrary to Itti *et al.*, the two feature maps are used separately as saliency maps, one for finding red and green contrasts and one for blue and yellow. A simpler method is also used to generate the colour double opponent maps, by taking different sized regions around each pixel instead of using Gaussian pyramids. The algorithm is described below using the red and green double opponent map as an example; the blue and yellow map is calculated in the same way. Two different square region-sizes are used, one small with side s and one large with side $s+p$, where s and p take several different values. For each region, the colour difference is calculated according to:

$$\Delta RG_{region}(pixel) = \sum_{region\ pixels,\ q} R(q) - G(q) \qquad (4.5)$$

The difference between the colour differences in the small region and in the large region is then calculated according to equation (4.6). To get the double opponent value a summation over s and p is done.

$$RG_{sp}(pixel) = \Delta RG_s(pixel) - \Delta RG_{s+p}(pixel) \qquad (4.6)$$

$$RG(pixel) = \sum_s \sum_p RG_{sp}(pixel) \qquad (4.7)$$

To create a saliency map for the intensity contrast, the same method is used, but the double opponent values are then calculated with the intensity mean in each region instead

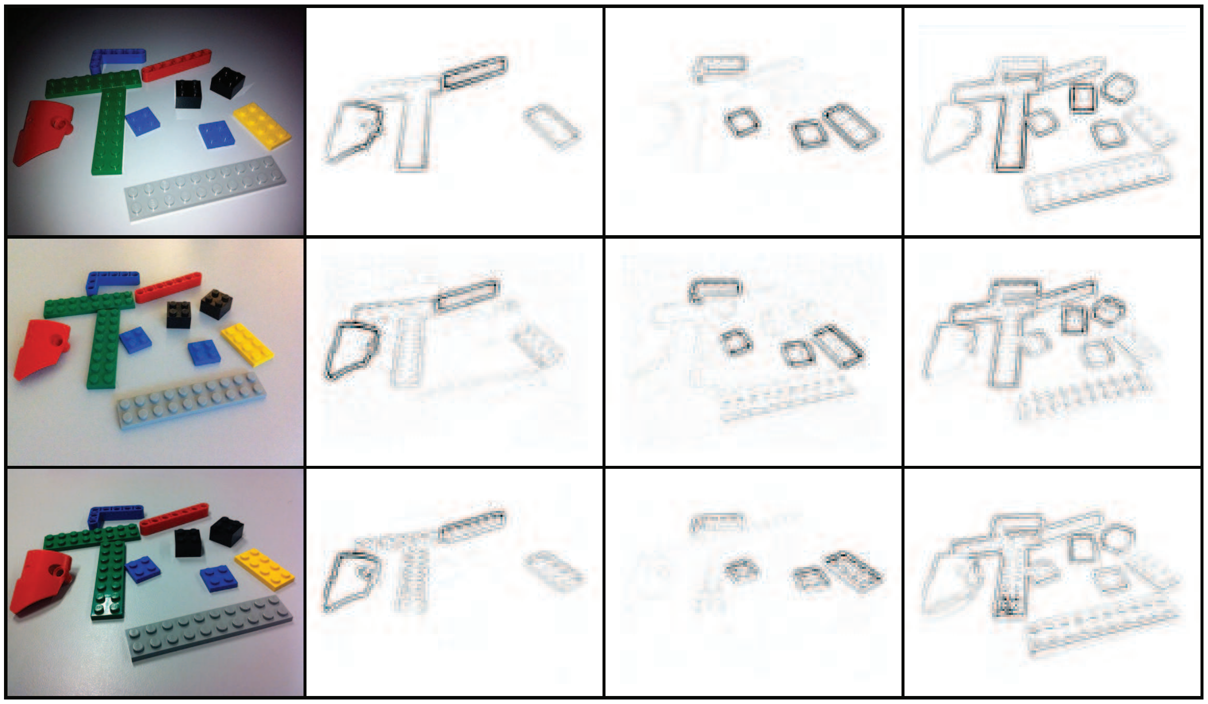

Figure 4.1: *An example of the saliency maps generated with the double opponent method and their performance under different lighting conditions. In the left column are the original images, in the second the red and green double opponent maps, in the third the blue and yellow double opponent maps and in last the intensity double opponent maps. Photo by authors.*

of the colour difference, ΔRG_s. An example of the saliency maps and their performance under different lighting conditions can be seen in figure 4.1.

In order to limit the performance impact only a rough position of the regions with high double opponent values is needed, the double opponent values are therefore only calculated for every nth pixel, where n is small relative to the image size.

4.2.2 Extracting salient regions

From the saliency maps the regions of interests are found near saliency maxima; to determine the size of the region so that it contains the entire potential object, some more processing is however needed. This can be done by segmenting the image – see section 2.3.1 – and letting the location and extension of the segment to which the maxima belongs to determine the size and location of the regions. There are many methods for image segmentation, some that are more complex than others are, and each with different result. Here **region growth** is used as it is easy to implement and gives a reliable result. In the region growth algorithm, pixels, called seed pixels, are used as starting points for the regions. The eight nearest neighbours of the seed pixel are examined, if they are similar to the seed pixel, they are added to the region. Then the nearest eight neighbours of each of the new region pixels are examined and if they too are similar to the seed pixel, they are added to the region, and so on.

In this case, the pixels corresponding to the maxima in the saliency maps are used as seed pixels and the maxima will therefore be sure to belong to the regions. As similarity measure, the Euclidian distance in the RGB colour space, between the potential region pixel and the seed pixel, is used.

This method is unable to handle objects with a lot of texture or objects built from several parts of different colours. To solve that task, a more complex segmenting algorithm is needed. For the purpose of this project, where the objects chosen are homogeneous in colour and texture, this method has a good balance between computation time and result.

4.2.3 Locating non-salient regions

The method described above is able to find objects that stand out in colour or in intensity. However, as mention in section 4.1, doors are sometimes made to blend in with the background; an extra method is thus needed to find the image regions that may contain doors. The method used in this project and the method used by Lee *et al.* [32] and Hensler *et al.* [22], is to locate long vertical lines in the image using the Hough transform; see section 2.2.2. As doors usually are smooth, door candidates are located in the region between a line and its closets neighbour. Very thin regions usually do not correspond to doors; a minimum distance between the lines is thus used. If the distance between the line and its closest neighbour is smaller than the minimum distance, the region between the line and its second-closest neighbour is used instead, if that distance is large enough, and so on.

4.3 Object recognition

To identify an object automatically, some type of object recognition algorithm is required. Such a computer program will typically contain one part for extracting the cues and one part consisting of a learner that draws conclusions; see section 2.4. The selection of the type of learner to implement is far from trivial. Different learners, such as neural networks (section 2.4.3) or SVM (section 2.4.2), will manage the task differently and the true performance of each learner is not apparently visible *a priori*. If implementation and comparison of these, with respect to the implementation, fundamentally different approaches are impossible, as in this case due to time limits, then the choice has to be done somewhat arbitrary. In this case, neural networks were chosen based on the knowledge of their successful utilization in other classification tasks and based on the subjective preferences of the project group.

4.3.1 Cue selection

In chapter 2, different cues were introduced. In designing the neural network, the cues were divided into two groups: (i) **main cues** and (ii) **minor cues**. The main cues are the most

Table 4.2: *The cues used in this project, both main cues and minor cues.*

Main cues	Colour Eigenimages
Minor cues	Horizontal position Vertical position Object width Object height Object proportion

important cues, but also the cues with most data. The minor cues are, on the other hand, cues that by themselves do not give enough information to make a prediction, but still can help the network make its decision. These minor cues normally consist of only a few data points each. The selection of cues and the division of them can be seen in table 4.2. The collection of cues is intended to capture different salient variations between the object types. Each cue is briefly described in the following sections.

The colour cue

As mentioned previously, colour is an essential feature when humans design objects and it is thus a natural cue to use in trying to identify salient objects, e.g. warning signs, in office environments. Different aspects on using colour as a cue was briefly discussed in section 2.3.2. In this project, the chosen implementation is to describe the colour by three histograms, one per colour channel. These histograms are then used as one single main cue. Each histogram consists of only a few bins in order to capture the basic appearance of the object while being, at least somewhat, robust to different lightning conditions.

Eigenimages as a cue

In addition to colour, the perhaps most important cue for humans to differentiate an object from another object is the shape of the object. Eigenimages, introduced in section 2.2.3, is one possible choice of cue describing the shape of objects. In this project, eigenimages were chosen because of their straightforward implementation. The input data for the eigenimage cue is generated by projecting a rescaled greyscale candidate-image onto the **limited eigenimage space**. The eigenimage space is spanned by the eigenimages calculated from the set of all non-background training images. The projection of the image gives a factor in each eigenimage. The number of eigenimages is large if the training set is large and to limit the amount of data, PCA – see section 2.4.5 – is used to select only the first few major eigenimages, i.e. the limited eigenimage space.

The minor cues

Many object properties are not discriminative, i.e. they cannot be used on their own to, even vaguely, guess the object type. However, these properties are still important in limiting the set of possible object types between which the object recogniser algorithm has to decide. The size of an object is such a cue; it can be used in object recognition to make a conclusion less probable or even exclude it completely, i.e. a two meter high object is not a cup. However, to estimate the size of the object, it is necessary to know the distance to the object; if the distance is not known, then the object's proportions can be used. The size can also be used as a vaguer cue, i.e. a cue where an object that takes up a large part of the image probably is a large object or, possibly less probably, a close-by object. Object size is in this project implemented as a vague minor cue.

Likewise, the position of the object is a quick aid in object recognition used by the human brain, i.e. an object in an unfamiliar position is at first harder to identify [24]. Position estimation suffers from the same problem as above and the position has to be implemented in such a vague way when no three-dimensional information is known.

4.3.2 Designing the neural network

Neural networks can be structured in many ways, ranging from simple feedforward networks to arbitrary recurrent neural networks. The simple feedforward neural network is straightforward in its implementation, but suffer from two main problems; (i) local minima can attract the learning algorithm very differently based on the random starting weights and (ii) the number of weights can get very large since the number of connections typically grow as $O(n^2)$ with the number of input neurons. The second problem is especially significant in this project since the input stem from images and thus naturally can be numerous in order to represent the variations in the images, each image possibly consisting of tens of thousands of data points. It is essential to avoid building networks with too many weights since more weights will result in a larger computational effort required for both training and utilisation, as well as require more training data to learn all weights correctly.

One solution to the above problem, and the solution used in this project, is to build a **modular neural network** (MNN). An MNN can schematically be described as a simple feedforward neural network where each neuron is replaced by a network of neurons. Generally, of course, the MNN can be designed in any way that neural networks can be designed. The particular design chosen in this project is based on a first layer of networks with one network per object type and per main cue. In the second layer, a network per object type takes the output from the networks specialised on finding that object type from different cues. The third and final layer consists of one network that weighs together the output from the different object-specialist networks as well as the minor cues. An illustration of the design is given in figure 4.2. The modular network-design makes it simple to add or remove cues, both main cues and minor cues, to the network and it is thus well suited for experimentation and extension.

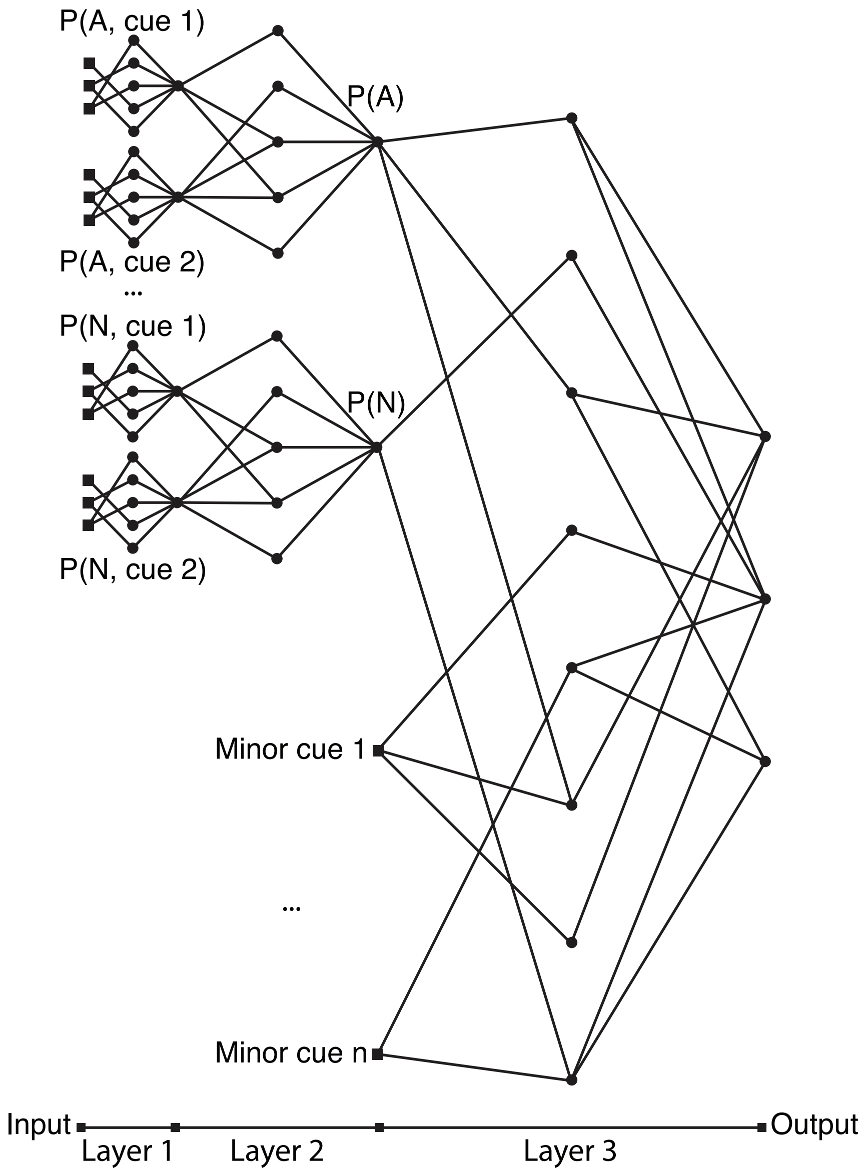

Figure 4.2: *An illustration of the modular neural network deployed in this project. In the first layer, a simple feedforward neural network is specialised on each combination of object type and main cue. The networks in the second layer weigh together the output from the networks in the first layer to form specialists on each object type. In the third and final layer, the conclusions from the object specialists are combined with the minor cues to form the input to the final decision network, which draws the conclusion on what object is present in the image, if any.*

4.3.3 Training the network

In order to decide between different object types correctly, the neural network needs to learn the differences from a **training set**. There exist different types of learning algorithms, but the details thereupon are out of the scope of this text. In this project, the **backpropagation** algorithm – first introduced by Bryson and Ho [4] – were selected. The choice of training algorithm and the tuning of the selected algorithm to best suit the problem at hand have great effect on the result. An accessible introduction to backpropagation was given by Wahde [67], here it will only be described briefly in order to give a background to the specific choices made in this project.

In the beginning of the learning process, the network is assigned small random weights. These weights describe a surface in weight space and to minimise the error in the network output the weights needs to be optimised. The network output error is propagated backwards in the network, hence the name backpropagation, to find the local error in the output from each neuron. Each weight is then updated according to the local gradient. This process is repeated over the training set in a random order and one pass through the set is called a training epoch. After each epoch, the **root mean square error** (RMS error) is computed on a separate, but similar, set called the **validation set** to determine if the result has improved on unknown data. Training is aborted when the minimum error on the validation set has been found, i.e. when further training will increase the error because the network will be to specifically fitted to the training data and thus lose the ability to generalise. The true performance is then measured on a third, completely unknown, data set, i.e. the **test set**.

One common addition to the learning algorithm used in this project is weight decay. By letting the weights decay, if not reinforced by the training, the impact of unimportant input neurons will diminish with time. Weight decay will also penalise large weights and thus lead to an overall smaller L^1-norm[2] for the weights. The latter is important since large weights will lead to rough changes in the network output on small variations in the output data and thus hurt the network's ability to generalise [2].

Local minima are a difficulty in most optimisation problem and backpropagation is no exception. In order to address the problem with local minima in this project, a small random term was stochastically added to the weight changes.

A related problem faced, is the ratio between **positive and negative examples**, i.e. the ratio between examples of object and non-objects in the training set. The negative examples are typically much more numerous, and needs to be so in order to comprehend the much larger variations between non-objects. The difficulties lie with the fact that if the negative examples are allowed to dominate the learning process then the network is inclined to become very pessimistic, i.e. give the correct answers to the numerous negative examples by always determining that no object is present. To counter this tendency, a compensation factor that makes the negative and positive examples equally important even while the negative examples are typically ten times more common were introduced in the project.

[2]The L^1-norm is the sum of the absolute values of all the elements.

Chapter 5

Results

In order to evaluate the algorithm, its ability to detect objects as well as its ability to recognise objects and reject non-objects has to be measured. Another important factor, since the robot is meant to recognise objects in real-time, is the image processing speed, i.e. how many images it can process per second. It is also of interest to compare the speed of the algorithm with the speed of an algorithm using the brute force method described in section 4.2. In this chapter, the result of these measurements will be presented, but first a description of the image data set used to generate these results is in order.

5.1 A description of the data set

The data set, on which the algorithm is evaluated, is decisive in what results can be achieved. The set used in this project consists of 400 images taken in similar office corridors at Chalmers University of Technology; figure 5.1 show an example image. Each image can, and often does, contain several different objects, all of which are manually labelled. These ground truth object regions are complemented with dynamically found salient regions that pose as negative examples. The negative examples, or background examples, are selected by the same algorithm that selects the regions of interest, thus giving specifically relevant examples of the same kind as the non-object regions found when running the program. The background regions are filtered to avoid fuzzy training examples that overlap the true object regions. All in all the data set consists of about 7,700 image regions. These image regions are equally divided into the training, validation and test sets. Different types of objects are however diversely common; table 5.1 shows the exact distribution.

5.2 Object detection ability

The ability to detect an object, i.e. the **detection rate**, is measured by comparing the ground truth object regions in the data set with the regions found by the object detection algorithm. If all regions given in the data set are detected, the detection rate will be one. To evaluate how well the regions match, an overlap threshold is used. The overlap threshold

Table 5.1: *The different object types and the number of image regions in the data set of each type.*

Object type	Examples count
Background	6,595
Card readers	54
Doors	495
Emergency exit signs	147
Fire alarm buttons	69
Red signs	351

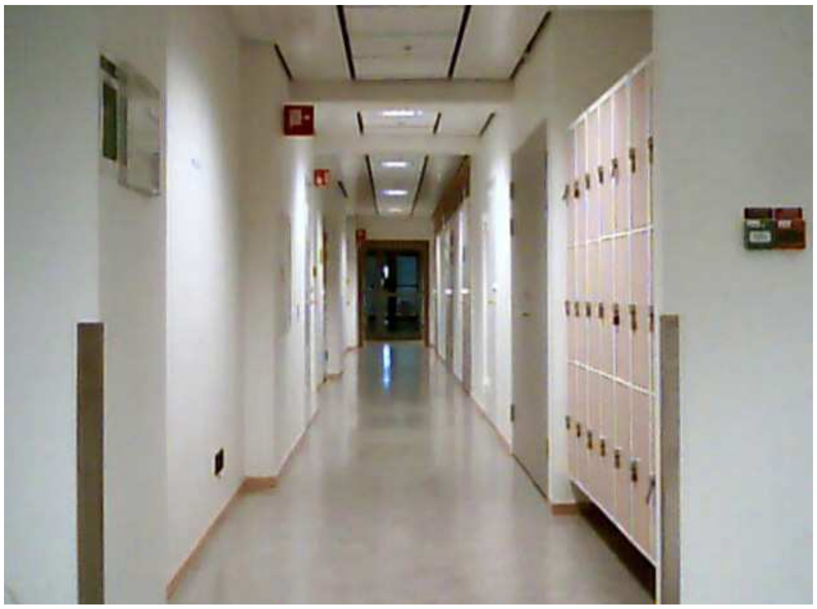

Figure 5.1: *An example of a typical image in the training set. This particular image contains several red signs on the left wall, a fire alarm button to the right and some hardly recognisable doors. Photo by the authors.*

sets the minimum region overlap, i.e. the area percentage that the larger region has in common with the smaller region, required for the regions to be considered a match. A small minimum overlap results in a higher detection rate, but then the algorithm consider a region detected even if the regions hardly overlap or are of very different size. A large minimum overlap gives a lower detection rate, but better matches. Since the recognition algorithm can handle objects that do not match perfectly, the minimum overlap has been chosen to 0.70.

As two separate detection algorithms are used to find regions of interest, the double opponent method (section 4.2.1) and the Hough transform method (section 4.2.3), the methods were first tested separately. The most interesting parameter to study is the num-

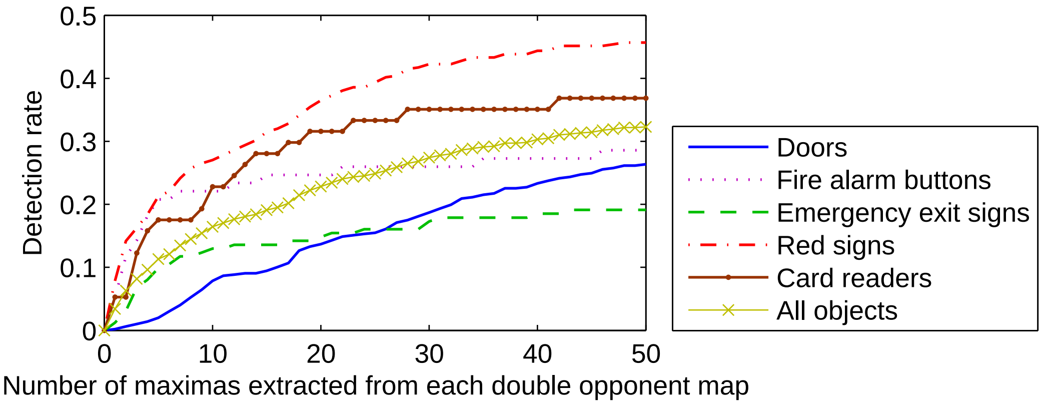

Figure 5.2: *The detection rate for the different object types using only the double opponent method with the number of double opponent maxima,* $m_{do} = 0, 1, ..., 50$.

ber of regions extracted as it affects the detection rate and the speed of the recognition algorithm. The maximum number of regions extracted by the double opponent method is the number of maxima extracted from each double opponent map, m_{do}, times three, since there are three double opponent maps. For the Hough transform method, the maximum number of regions extracted will be the number of Hough maxima, m_{ht}, extracted plus one.

Figure 5.2 shows the detection rate using the double opponent method only. The detection rate for each object type is plotted against the number of maxima for each double opponent map, $m_{do} = 0, 1, ..., 50$. As can be seen, the detection rate is increasing with the number of maxima, but the curve levels out. This is due to that the most easily detected regions are detected first. Another problem is that objects might have been detected but the segmentation is not good enough to pass the overlap threshold. The object types red signs, fire alarm buttons and card readers are easiest to detect, while doors and emergency exit signs are harder.

In figure 5.3, the detection rate of the Hough transform method is shown for the number of Hough maxima, $m_{ht} = 0, 1, ..., 60$. Only door regions were detected and thus only the door detection rate is shown. Figure 5.4 shows the detection rate for the different object types using both methods with the number of double opponent maxima as $m_{do} = 0, 1, ..., 50$ and the number Hough maxima, $m_{ht} = 25$. As can be seen the detection rate for the doors increased, compared to when using only the double opponent method. The detection rate seem low, but to get a total detection rate of 0.22, which the algorithm achieves by extracting 56 regions per image, a brute force method – see section 4.2 – need approximately 27,500 regions. The brute force method then uses 25 different template regions that are shifted 10 pixels each step. A comparison of the object detection rate between the methods for the different object types with the total detection rate being 0.22 can be seen in table 5.2. The brute force method is good at detecting doors but less successful with the other object types.

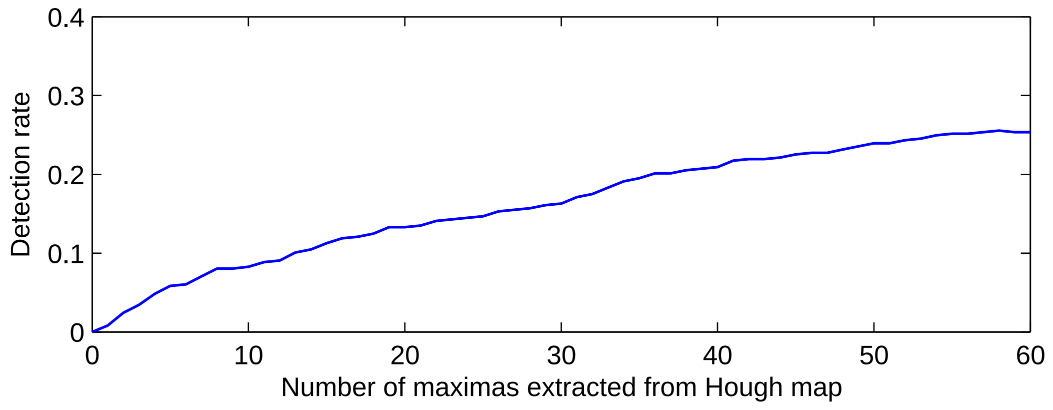

Figure 5.3: *The detection rate of the Hough transform method, used to find non-salient large rectangular regions, plotted against the number of Hough maxima,* $m_{ht} = 0, 1, ..., 60$.

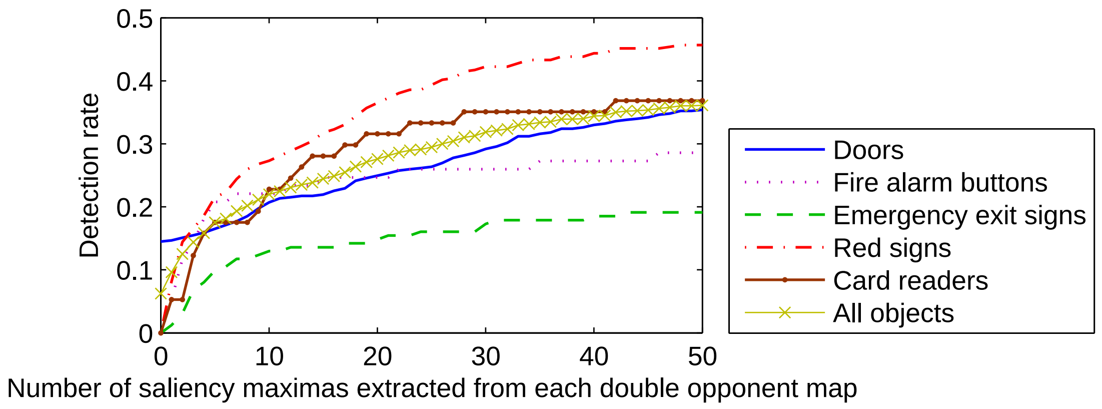

Figure 5.4: *The detection rate for the different object types using both methods with the number of double opponent maxima as* $m_{do} = 0, 1, ..., 30$ *and the number Hough maxima as* $m_{ht} = 25$.

5.3 Object recognition ability

To measure the recognition ability, i.e. the algorithm's capability to recognise the objects in the detected image regions as well as the ability to discard regions without objects, **sensitivity** and **specificity** are two common statistical measures. However, as the following section will show, neither of the two measures can alone give a correct picture of the algorithm's ability. Together they however give information on the recognition ability of the algorithm. Common techniques, such as the ROC curve, will be introduced in the following section together with an introduction to the sensitivity and specificity measures before applying them on the algorithm.

Table 5.2: *A comparison between the algorithm and the brute force approach. The detection rate is shown for the different object types and the total detection rate is set to 0.22.*

Object type	Double opponent method & Hough transform method	Brute force
Card readers	0.2281	0.0526
Doors	0.2072	0.4427
Emergency exit signs	0.1296	0
Fire alarm buttons	0.2208	0.0390
Red signs	0.2730	0.0919
Total	0.2197	0.2223

5.3.1 Sensitivity and specificity measures

Sensitivity and specificity are statistical measures that describe an algorithms achievement in a binary classification task. Sensitivity is a measure for an algorithm's ability to identify an object correctly. The sensitivity is defined as:

$$\text{Sensitivity} = \frac{\text{Number of true positives}}{\text{Number of ground truth positives}} \tag{5.1}$$

A high sensitivity can, however, always be achieved by lowering the **discrimination threshold** of the recogniser, i.e. the threshold the neuron response has to surpass before considered positive. For example, a classifier presented with the task of distinguishing an object from the background can easily achieve a high sensitivity by simply classifying everything as objects; such a classifier would however be pointless.

Specificity, on the other hand, measures an algorithm's ability to avoid classifying a non-object as an object; it is defined as:

$$\text{Specificity} = \frac{\text{Number of true negatives}}{\text{Number of ground truth negatives}} \tag{5.2}$$

Similar to sensitivity, a high specificity can be gained by altering the discrimination threshold. In the above example: a classifier that never finds an object would not make any false calls, but it would be equally pointless as the classifiers that find objects everywhere. However, there is a relationship between the sensitivity and the specificity; e.g., a threshold that forces the classifier to be extra certain before making the call would at the same time force the classifier to miss correct classifications because it not is sufficiently certain about them. An example of the relationship between the sensitivity and the specificity can be seen in figure 5.5. In binary classification tasks, these two measures are easily understood; however, in a multi-class classification task[1], the specificity definition becomes unclear. In that case, a true negative both can be a correctly identified negative of another class and be

[1]A multi-class classification task is a classification task with more than two options.

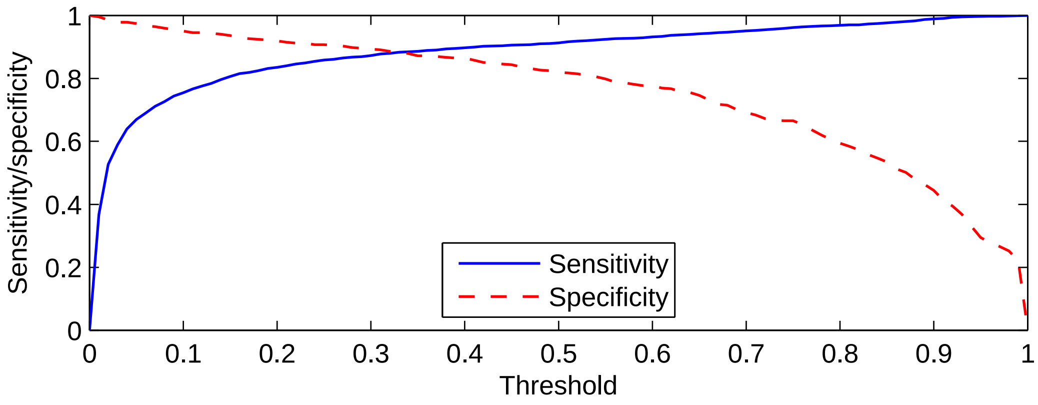

Figure 5.5: *The sensitivity and specificity for the background object class plotted together. For the background class, a high discrimination threshold gives many detection and vice versa. The data is generated on the test set.*

a negative correctly identified as a negative but incorrectly classified among the remaining classes. One definition – the definition used in this text – is to ignore such misclassifications between the remaining objects when calculating the specificity.

Because of the dependency of the discrimination threshold, sensitivity and specificity are not commonly used directly to describe the ability of a classifier. For that purpose, an **ROC curve** is better suited. In an ROC curve, the true positive rate, i.e. the sensitivity, is plotted against the false positive rate, i.e. one minus the specificity. The ROC curve weighs together the sensitivity and the specificity where the area under the curve describes the classifiers ability to make correct assessments. A numerical measure of the algorithms ability is **Matthews's correlation coefficient** (MCC), which was introduced by Matthews [41]. The MCC is closely related to the correlation between correct and predicted classifications. However, the individual measures are still of interest, especially in order to optimise the applied discrimination threshold.

5.3.2 The ability to separate objects from the background

Even if this algorithm is tasked with identifying multiple object types, the task of separating objects from background is distinct. The sheer number of background regions in the images make that task crucial in order to maintain an overall low rate of false detections. Fortunately, the task can be considered binary, thus allowing simpler analysis.

An ROC curve, describing the characteristics of this object recogniser's ability to discard background image regions correctly can be seen in figure 5.6. The explicit sensitivity and specificity can also be seen in figure 5.5. In order to give a numerical score on the ability the MCC is computed to 0.68. Another common numerical measure is the **area under the ROC curve** (AUC), which in fact is equal to the probability that the classifier rates a random positive higher than a random negative, and it is calculated to 0.96.

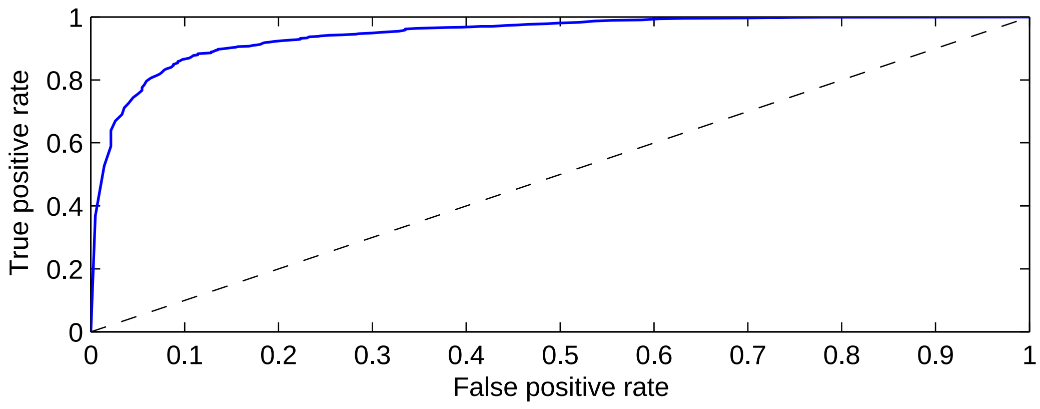

Figure 5.6: *This ROC curve describes the object recognition algorithm's ability to difference between background, i.e. examples that not belong to any object class, and objects. The diagonal line is the line of no-discrimination, i.e. the score of completely random guesses. The data is generated on the test set.*

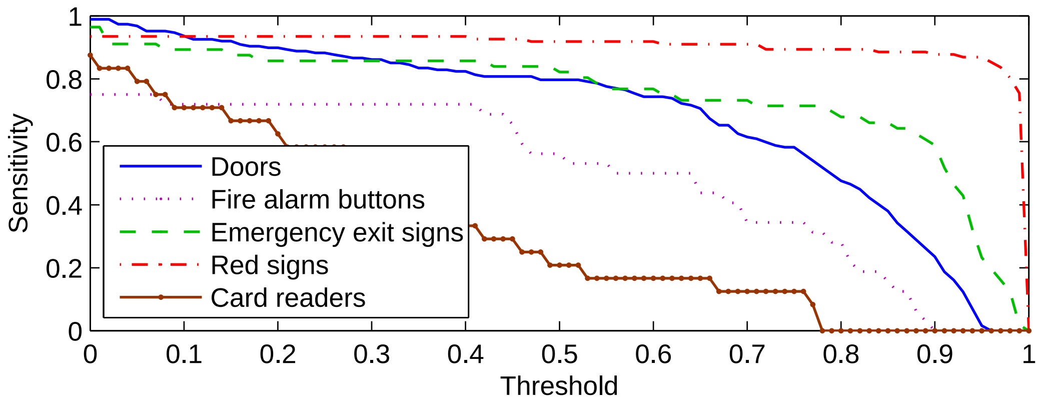

Figure 5.7: *The sensitivities of the recogniser computed on sets of positives for the different object types. The sensitivities are computed on previously unseen examples only, i.e. the test set.*

5.3.3 The ability to recognise different object types

The task of differencing between the multiple object types is non-binary and measures intended for binary problems are thus no longer suited. The sensitivity can still be measured since it is computed only on the set of positives and thus on a binary problem; the sensitivities for the different object types can be seen in figure 5.7. The sensitivities vary greatly between the different object types. It can be noted that the curves for the different object types meet the sensitivity axis at different positions. The position on the axis where the meet occur give the percentage of the positive examples for each class that the classifier incorrectly assign to other object types.

The specificity, however, cannot be computed for this multi-class task. The set of all

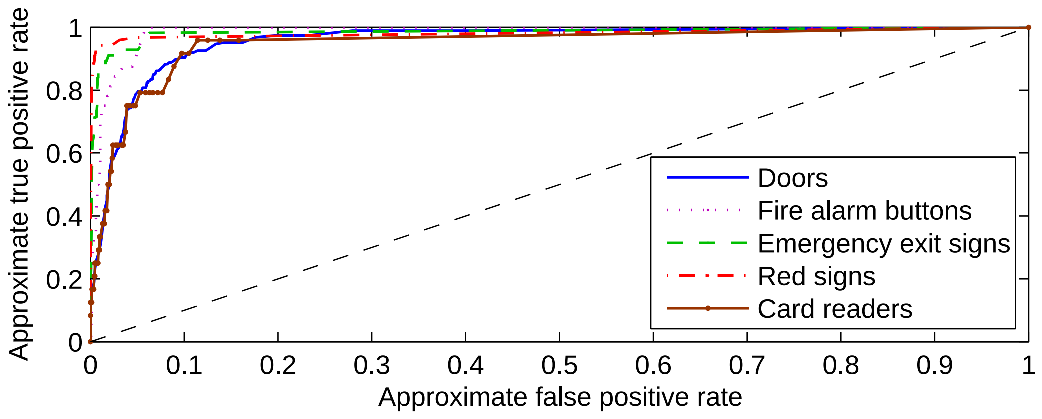

Figure 5.8: *The ROC curves for the approximate sensitivities and specificities of the recogniser computed for the different object types when ignoring misclassifications among the object types. The data is generated on the test set.*

Table 5.3: *The confusion matrix of the object recognition algorithm. The columns represent the actual objects and the rows represent the classified results, thus the diagonal entries are the true positives and the off-diagonal entries are the misclassifications. Data is generated on the test set.*

	Background	Card readers	Door	Emergency exit signs	Fire alarm buttons	Red signs
Background	2,110	12	44	8	6	5
Card readers	2	1	0	0	3	0
Doors	105	0	100	0	0	0
Emergency exit signs	20	0	0	45	0	0
Fire alarm buttons	8	1	0	0	12	0
Red signs	8	0	0	0	2	118

negatives contain correct answers on the same side of the discrimination threshold as the examined class, thus the answer will not only depend on the threshold but also on the network response for each object type. An approximated specificity can be computed by ignoring the misclassification between the object classes. However, that misclassification is important – as can be seen in figure 5.7; such an approximate specificity, however, permits a comparison between the ROC curves for the different object types. The approximated ROC curves can be compared in figure 5.8.

The true ability of the algorithm is only encompassed by the **confusion matrix**, see table 5.3. However, the information is not easily grasped on that form.

Table 5.4: *The execution times for the selected algorithm and the reference brute force approach.*

Method	Average execution time per image	Images per second
Brute force detection	75,997 ms	0.013
The algorithm	1,676 ms	0.60

5.4 Performance

One major assumption in this project has been that a brute force approach to object detection would be too slow for a real-time application. To test that assumption and to evaluate the speed of the algorithm a simple test were run. As seen in section 5.2, 27,500 equally spaced regions of varying size gives a roughly equal object detection rate as the implemented algorithm with its selected parameters. The average time it takes the program to analyse each image in a small set when using each method were sampled; each test were repeated several times in order to average out noise in the background load on the computer[2]. The resulting figures are displayed in table 5.4 and as can be seen the difference is truly significant.

[2]The test system used ran Windows 7 64-bit on an Intel Core 2 Duo E6600 processor at 2.6 GHz with 4 GB of RAM.

Chapter 6

Discussion and conclusion

6.1 Discussion

6.1.1 Evaluation of the algorithm

The object detection algorithm does not give a very high detection rate. It should however be noted that it is highly dependent of the manually labelled data set. However, a higher detection rate would have been an improvement. One reason for the low detection rate is that the camera has difficulties with lighting differences, making shadows darker and illuminated areas lighter, thus spoiling the contrast of the image. In many cases, this results in objects not being detected. Even if the objects are salient enough, this may still cause incorrect segmentations. Another challenge is the environment where the robot operates in which the objects often are small. However, these challenges are solvable and the essential concept, to locate regions of interest and thus minimise the number of regions analysed by the recognition process, succeeds. The number of processed regions needed for the same detection rate is much lower compared to the brute force method. The algorithm also shows a consistent detection rate over the different types of objects, as can be seen in table 5.2.

That the applied object recognition algorithm is capable of correctly distinguishing objects from background with high probability can be seen from the ROC curve in figure 5.6. However, the algorithm has more difficulties separating the different object types; see figure 5.7. A comparison between the results for different object types in figure 5.7 and the available amount of training data, shown in table 5.1, makes it probable that shortage of training data is the key to these difficulties. Tests with more object types have shown that the algorithm is scalable with the capacity to learn more than these five object types.

The current setting, i.e. office corridors, is not ideal in object recognition aspect. Objects are typically viewed at a small angle, thus limiting the image detail in the data. The navigation task at the same time imposes a wide viewing angle to encapsulate several objects simultaneously and thus allows each object to seize a smaller part of the image. The human visual system has the ability to focus its attention to one object of interest at a given time and thus gain high resolution and accurate data about that object in order to identify it. In fact, many of the examples in the training set are hard for a human to recognise from the images only; often the brain needs to use more information, such as wall and floor position and heavy contextual information, than what is available for the algorithm from the implemented cues.

6.1.2 Suggested improvements

There are several ways to improve both detection and recognition. A change to a more modern and high-end web camera with faster auto focus and better light handling is maybe the most obvious way. A major problem in recognition is that many objects are viewed at a wide angle and at a large distance. By loosening the constraint that several landmarks should be detected at the same time, a more beneficial and easily recognised image of the object can be gained. One way of accomplishing this is to use a combination of a high-resolution view of the object with a low-resolution view of the scene, similar to a method introduced by Gould *et al.* [15].

The algorithm has been implemented with the idea that it should be easy to add new feature detectors for the detection algorithm as well as more cues for the recognition algorithm. By adding the Hough transform method, the detection of doors was greatly improved and other such extensions will make similar improvements. One such possible addition is to use more context information, e.g. detecting walls, floor and ceiling, or use a stereo camera to retrieve depth information. Any additions will of course result in a loss of speed and any extension of the network will require more training examples.

With improvements of frameworks such as CUDA and OpenCL, graphic cards have been made available for easier utilisation in parallel computations. By enabling faster parallel calculations for the image processing by using such frameworks, the speed of the algorithm would be vastly improved thus allowing for extensions to algorithm. The speed could also be improved by using multi-thread computations, easily accomplished by utilising improvements in the .NET 4 framework or by reworking the code.

6.2 Conclusion

In this thesis, cameras have been used for detection of objects to be used in landmark navigation. The algorithm describe above locates and recognises objects in real-time. The question is, however, if full object recognition is needed for landmark navigation. There are probably more efficient ways to use cameras to navigate without recognising objects first, such as using SIFT features directly.

However, a few steps have been taken to a more general scene interpretation, by achieving real-time object detection and recognition in office environments. Even though the algorithm in this thesis has been trained on emergency equipment and doors, it can easily be retrained to detect and recognise other object types. It has been shown that object recognition using cameras are possible to achieve in real-time and that it in the future should be possible to achieve a more general scene interpretation.

Bibliography

[1] Dana H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[2] Peter L. Bartlett. For valid generalization the size of the weights is more important than the size of the network. In *Neural Information Processing Systems*, pages 134–140, December 1996.

[3] Charles S. Brubaker, Jianxin Wu, Jie Sun, Matthew D. Mullin, and James M. Rehg. On the design of cascades of boosted ensembles for face detection. *International Journal of Computer Vision*, 77(1):65–86, May 2008.

[4] Arthur E. Bryson and Yu-Chi Ho. Applied optimal control: optimization, estimation, and control. *IEEE Transactions on Systems, Man and Cybernetics*, 9(6):366–367, June 1979.

[5] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, April 1983.

[6] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[7] Enric Celaya, Jose-Luis Albarral, Pablo Jiménez, and Carme Torras. Visually-guided robot navigation: From artificial to natural landmarks. In *Field and Service Robotics*, pages 287–296, December 2007.

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, pages 886–893, June 2005.

[9] Guilherme N. DeSouza and Avinash C. Kak. Vision for mobile robot navigation: A survey. *IEEE transaction on pattern analysis and machine intelligence*, 24(2):237, February 2002.

[10] Gyuri Dorkó and Cordelia Schmid. Object class recognition using discriminative local features. Rapport de recherche RR-5497, INRIA - Rhone-Alpes, February 2005.

[11] Pedro F. Felzenszwalb. Learning models for object recognition. In *Computer Vision and Pattern Recognition*, pages 1056–1062, December 2001.

[12] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, September 1995.

[13] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[14] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson Prentice Hall, Upper Saddler River, NJ, USA, third edition, 2008.

[15] Stephen Gould, Joakim Arfvidsson, Adrian Kaehler, Benjamin Sapp, Marius Meissner, Gary R. Bradski, Paul Baumstarck, Sukwon Chung, and Andrew Y. Ng. Peripheral-foveal vision for real-time object recognition and tracking in video. In *International Joint Conference on Artificial Intelligence*, pages 2115–2121, January 2007.

[16] Grant Grubb, Alexander Zelinsky, Lars Nilsson, and Magnus Rilbe. 3d vision sensing for improved pedestrian safety. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 19–24, June 2004.

[17] A. Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371, September 1910.

[18] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, August 1988.

[19] Chie Hashizume, Vasudevan V. Vinod, and Hiroshi Murase. Robust object extraction with illumination-insensitive color descriptions. In *International Conference on Image Processing*, pages 50–54, October 1998.

[20] Jean-Bernard Hayet, Frédéric Lerasle, and Michel Devy. A visual landmark framework for mobile robot navigation. *Image and Vision Computing*, 25(8):1341–1351, August 2007.

[21] Simon Haykin. *Neural Networks: a comprehensive foundation*. Prentice Hall, Upper Saddle River, NJ, USA, second edition, 1999.

[22] J. Hensler, M. Blaich, and O. Bittel. Real-time door detection based on adaboost learning algorithm. In *International Conference on Research and Education in Robotics*, May 2009.

[23] E. Hjelmås and B.K. Low. Face detection: A survey. *Computer vision and image understanding*, 83(3):236–274, September 2001.

[24] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. In *Computer Vision and Pattern Recognition*, pages 2137–2144, June 2006.

[25] Paul V. C. Hough. Method and means for recognizing complex patterns, 1962. US Patent 3,069,654.

[26] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, and Anil K. Jain. Face detection in color images. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):696–706, May 2002.

[27] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, November 2002.

[28] Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, January 2000.

[29] Michael Kearns. Thoughts on hypothesis boosting. Unpublished manuscript, December 1988.

[30] Ellen Klingbeil, Ashutosh Saxena, and Andrew Y. Ng. Learning to open new doors. In *Robotics: Science and Systems*, June 2008.

[31] David C. Lay. *Linear algebra and its applications*. Addison Wesley, Boston, MA, USA, third edition, 2003.

[32] J. Lee, N.L. Doh, W.K. Chung, B. You, and Y.I. Youm. Door detection algorithm of mobile robot in hallway using pc-camera. In *Conference ISARC 2004*, 2004.

[33] Bastian Leibe, Nico Cornelis, Kurt Cornelis, and Luc J. van Gool. Dynamic 3d scene analysis from a moving vehicle. In *Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[34] Ales Leonardis and Horst Bischof. Robust recognition using eigenimages. *Computer Vision and Image Understanding*, 78(1):99–118, April 2000.

[35] Anat Levin, Paul A. Viola, and Yoav Freund. Unsupervised improvement of visual detectors using co-training. In *International Conference on Computer Vision*, pages 626–633, October 2003.

[36] David G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, September 1999.

[37] Xiaoguang Lu. Image analysis for face recognition. *Personal Notes*, 5, May 2003.

[38] Stéphane G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, July 1989.

[39] Daniel Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 207(1167):187–217, March 1980.

[40] Mario Mata, Jose M. Armingol, Arturo de la Escalera, and Miguel A. Salichs. Mobile robot navigation based on visual landmarks recognition. In *Proc. 4th IFAC Symposium on Intelligent Autonomous Vehicles*, September 2001.

[41] B.W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta*, 405(2):442–451, October 1975.

[42] Jean Morlet and Alex Grossman. Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM Journal on Mathematical Analysis*, 15(4):723–736, 1984.

[43] Hiroshi Murase and Shree K Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, January 1995.

[44] Michael Oren, Constantine Papageorgiou, Pawan Sinha, Edgar Osuna, and Tomaso Poggio. Pedestrian detection using wavelet templates. In *Computer Vision and Pattern Recognition*, pages 193–199, June 1997.

[45] Constantine Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *International Conference on Computer Vision*, pages 555–562, January 1998.

[46] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, June 2000.

[47] Claudio M. Privitera and Lawrence W. Stark. Algorithms for defining visual regions-of-interest: Comparison with eye fixations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):970–982, September 2000.

[48] Morgan Quigley, Siddharth Batra, Stephen Gould, Ellen Klingbeil, Quoc V. Le, Ashley Wellman, and Andrew Y. Ng. High-accuracy 3d sensing for mobile manipulation: Improving object detection and door opening. In *International Conference on Robotics and Automation*, pages 2816–2822, May 2009.

[49] Lawrence G. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1963.

[50] Peter M. Roth, Horst Bischof, Danijel Skočaj, and Aleš Leonardis. Object detection with bootstrapped learning. In *10th Computer Vision Winter Workshop*, pages 33–42, February 2005.

[51] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, January 1998.

[52] Radu B. Rusu, Andreas Holzbach, Gary Bradski, and Michael Beetz. Detecting and segmenting objects for mobile manipulation. In *International Conference on Computer Vision Workshops*, pages 47–54, September 2009.

[53] Kanumuri Sandeep and A. N. Rajagopalan. Human face detection in cluttered color images using skin color and edge information. In *Indian Conference on Computer Vision, Graphics & Image Processing - ICVGIP*, December 2002.

[54] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. 3-d depth reconstruction from a single still image. *International Journal of Computer Vision*, 76(1):53–69, January 2008.

[55] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. Learning 3-d object orientation from images. In *International Conference on Robotics and Automation*, pages 794–800, May 2009.

[56] Ashutosh Saxena, Jamie Schulte, and Andrew Y. Ng. Depth estimation using monocular and stereo cues. In *International Joint Conference on Artificial Intelligence*, pages 2197–2203, January 2007.

[57] Bernt Schiele and James L. Crowley. Object recognition using multidimensional receptive field histograms. In *European Conference on Computer Vision*, pages 610–619, April 1996.

[58] Henry Schneiderman and Takeo Kanade. A statistical method for 3d object detection applied to faces and cars. In *Computer Vision and Pattern Recognition*, pages 1746–1759, June 2000.

[59] Stephen Se, David Lowe, and Jim Little. Local and global localization for mobile robots using visual landmarks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 414–420, October 2001.

[60] Thomas Serre, Lior Wolf, Stanley M. Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):411–426, March 2007.

[61] Stephen M. Smith and J. Micheal Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, May 1997.

[62] Irwin E. Sobel. *Camera models and machine perception*. PhD thesis, Stanford University, Stanford, CA, USA, 1970.

[63] Matthew Turk and Alex Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition*, pages 586–591, June 1991.

[64] Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[65] Vladimir Vapnik. *The nature of statistical learning theory*. Springer, New York, second edition, 2000.

[66] Paul A. Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, pages 511–518, December 2001.

[67] Mattias Wahde. *Biologically Inspired Optimization Methods*. WIT Press, Southampton, UK, first edition, 2008.

[68] Felix Werner, Joaquin Sitte, and Frédéric Maire. Visual topological mapping and localisation using colour histograms. In *International Conference on Control, Automation, Robotics and Vision*, pages 341–346, December 2008.

[69] Laurenz Wiskott, Jean-Marc Fellous, Norbert Kruger, and Christopher van der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, July 1997.

[70] Ming-hsuan Yang, David J. Kriegman, and Narendra. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern analysis and Machine intelligence*, 24(1):34–58, January 2002.

[71] Wen-yi Zhao, Rama Chellappa, P. Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys - CSUR*, 35(4):399–458, December 2003.