ALBERT-LUDWIGS-UNIVERSITÄT
FREIBURG

Department of Computer Science
Autonomous Intelligent Systems
Prof. Dr. Wolfram Burgard

accomplished at the
UNIVERSITY OF BRASÍLIA
Department of Electrical Engineering
Control Systems and Computer Vision Laboratory
Prof. Geovany Borges, D.Sc.

Fault Detection in Aerial Power Lines from Video Images

Diploma Thesis

Joachim Strach

October 2005 - May 2006

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Brasília, den 01. Mai 2006

Joachim H. Strach

Acknowledgements

I would like to thank all the people that helped me to complete this thesis. My first acknowledgements go to **Prof. Geovany Araujo Borges** who was my primary advisor during my time at UnB (Brazil). His working style is outstanding and professional, and he made crucial suggestions to my work based on his well-founded theoretical knowledge with insight into several practical applications. I want to thank Geovany for all the other help he gave to me with processing administrative issues, arranging power station visits, and many things more. Geovany is also a thoroughly enjoyable person and working with him was both a pleasure and an intellectual challange. I also want to thank my co-examiner **Prof. Dr. Wolfram Burgard** (Germany) to give me the great opportunity to do this thesis abroad and for always supporting me when it was possible.

Second, I owe a huge debt of gratitude to members of several groups. I want to thank people of AIS (Germany), namely PhD student **Christian Plagemann**, PhD student **Axel Rottmann** and former graduate student **Daniel Meier** for discussions and all kind of support, **Dr. Kai Arras** for his time he will spend as assessor and **Susanne Bourjaillat** for her administrative help. Furthermore I would like to thank my former advisors **Dr. Maren Bennewitz** and **Dr. Sven Behnke** from NimbRo (Germany) for their discussions even after my time at NimbRo. I also want to thank many people of LAVSI and LCVC (Brazil): **Dr. Adolfo Bauchspiess** for his discussions; **Flávio de Barros Vidal** for his practical advice; graduate student **Carla Aguiar** for her help with Omni; **Alexandre Martins**, **Antônio Padilha**, **Luiz Curado** and all others for the time we spend together in the laboratories.

Third, many thanks to my friends, especially **Alfredo Américo de Freitas** and **Viviane Aguiar**, and all other persons that made my stay in Brazil possible, comfortable and enjoyable.

Above all, I am deeply grateful to my family who supported me during my whole studies. My parents deserve much gratitude and I want also to thank my sisters for their invaluable support they gave to me.

Contents

1 Introduction

The problem of fault detection in aerial power lines from video images involves two main areas, namely Power Line Inspection and Computer Vision.

Over the last years power line maintenance philosophy was redefined from *repairing* equipment to *inspecting* it, and then repairing or replacing it when it is necessary [37]. For inspection on a regular basis especially techniques for camera guidance and controlling of unmanned aerial vehicles (UAV) have been developed. We present a method for fault detection from video images for 3D objects under real-world conditions to analyse the large amount of surveillance video data automatically and to faciliate higher inspection frequencies of power line equipment. Future applications could include, e.g., general inspection tasks and intelligent fault detection in robotics.

The majority of current object recognition methods in Computer Vision agrees on view-based representations using local distinctive descriptors which allow for a sparse object representation. Conversely, fault detection needs some tradeoff between *object-covering* and distinctiveness. We are using biologically-inspired features [72, 71] which are principally suitable to select features in an object-covering manner. Additionally, fault detection deals with the problem that faults can have a *high variation of fault appearance* in the absence of many training examples. This suggests the usage of one-class learning techniques which have been quite unattended in the field of object recognition. Two-class or multi-class classification are frequently used and can solve for selection of distinctive features within objects implicitly which is not feasible any more for an one-class classifier.

Up to our knowledge no method exists that deals with robust fault detection for 3D objects neither in Power Line Inspection nor in Computer Vision. We present an abstract view on the question *"What makes a fault?"* and state that a fault is a variation of object appearance not covered by an *one-class object model*. This perspective on fault detection supersedes the need for difficult to model fault classes. A model only has to be learned once and can detect *unseen faults* of various appearance. We propose a method for fault detection on a limited view that simplifies the notion of fault to an "one-feature-is-missing" task using a linear one-class classifier. The problem of object-covering features is approached in a practical and efficient way similar to the human behaviour of focusing the field of vision on a region of an object. We provide a solution for feature extraction within the object and show that distinctive features can be learned with a background class.

For evaluation we generated a large synthetic data set containing a faultless subset and three different fault subsets. It shows an low-textured rigid object around a specific

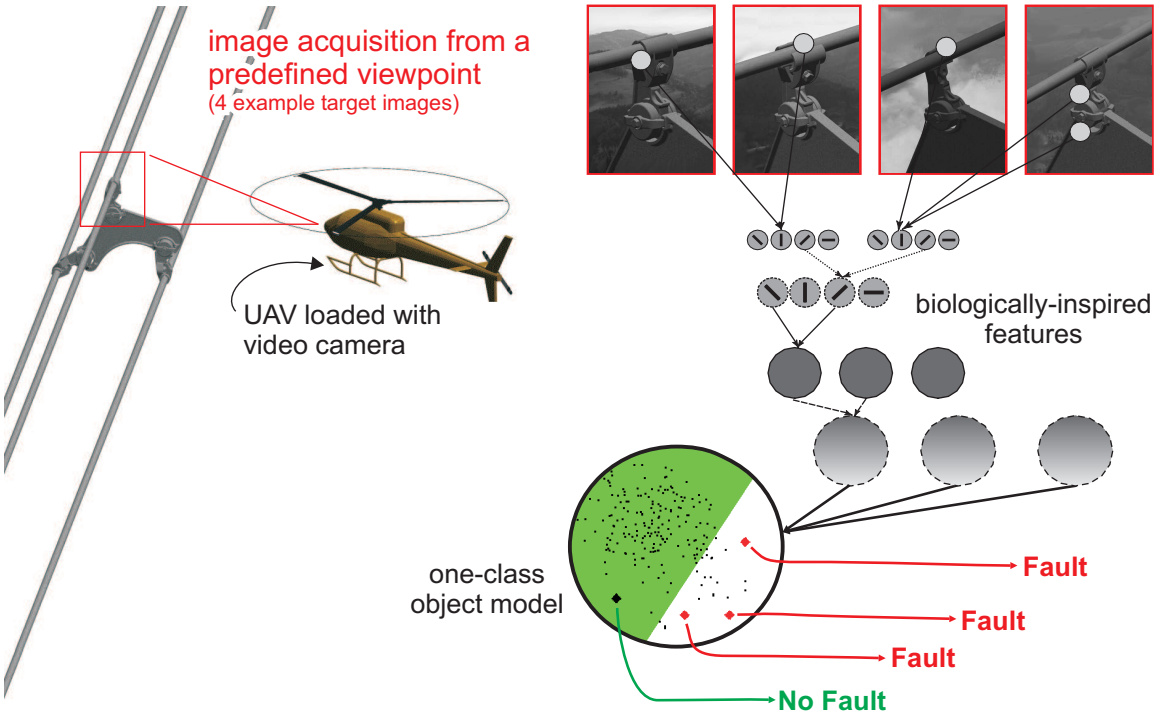


Figure 1.1: Power line inspection by an UAV using the proposed method for fault detection on a limited view.

viewpoint and models strong illumination changes, clutter, scale. We demonstrate on this data set that the use of an one-class model does not necessarily have to lead to worse fault detection performance, give insight into the degree of fault detectable and demonstrate the robustness of the approach to real-world conditions.

Our last contribution is a basic framework for fault detection in aerial power lines for an UAV. It suggests how to integrate our method with the UAV controls system and summarizes several parameters for a future fault detection system. Figure 1.1 illustrates the on-line inspection of a target object from a predefined view, and shows one faultless and three fault example images of our synthetic data set.

1.1 Objectives

The objective of this work is to develop a fault detection method for 3D objects under real world conditions from video images that will be suitable for power line inspection and as general as possible. There are several sub-goals that must be addressed to achieve

this main goal. To our knowledge no method for this special purpose exists and we have to identify an appropriate and extendable object recognition method as basis for fault detection that has state-of-the-art recognition performance for our specific target object. Faults of our problem domain have a high variation in their appearance, and little training data available further impedes a direct modeling of faults. Our intention is rather to learn one object model that will consider a particular degree of aberration as fault. This idea for fault detection is common in other fields. For a specific choice the most important sub-goal is to develop an approach and provide an implementation for evaluation purpose. Our main interests are how good the method will perform for faults of our problem domain and to which degree it is possible to detect unseen faults. The method should provide a solid foundation for further work and give guidance how to procede. The last sub-goal is to provide an interface between our fault detection method and the existing UAV control system.

1.2 Outline of this Work

This works starts with a chapter about related work which summarizes various other methods of object recognition and one-class classification that are worth to be considered for fault detection. Our choice are features inspired by visual cortex [71] and in the first part of Chapter 3 we introduce these features in more detail. Our description covers basic knowledge of Cognitive Sciences, the computational model, which modifications we made and to some degree how features interact with our extension for fault detection in Chapter 5. The second part of Chapter 3 introduces the primal and dual form of the one-class SVM of Schölkopf et al. [68] and its properties.

In Chapter 4, we describe basic framework for fault detection in aerial power lines which uses an UAV and our fault detection method. An introduction presents the projects that are connected to this thesis and the specific inspection task which our work is intended to solve primarily. We propose an inspection procedure that illustrates how our method should be used with an UAV for inspection.

In Chapter 5, we propose our approach to fault detection on a limited view. It starts with an abstract definition of a fault for 3D objects and draws connection between this general notion and our proposed method for fault detection. Section 5.2 describes the techniques used for learning an one-class object model. Subsections show how features are extracted from within the object and a region, how discriminant features are selected and which notion of one-class object model a linear SVM is capable to learn. The chapter ends with the procedure for fault detection using the learned one-class model and concludes with a summary that discusses properties of this method for fault detection.

Chapter 6 first describes our synthetical data set which is the basis for the experiments in the following subsection. In all, five experiments are performed. Chapter 6 concludes with a fault detection experiment made with trained individuals on our synthetical data set. Finally, Chapter 7 concludes our work and identifies subjects for future with respect

to fault detection under real-world conditions.

2 Related Work

This chapter presents current methods and techniques of Object Recognition and different approaches to one-class classification.

The task of fault detection is close to the task of object recognition and we describe how fault detection is related to the objectives and terminology of Object Recognition (Section 2.1.1). Object recognition is approached in different ways and we introduce *model-based* approaches, *view-based* approaches and *template matching* (Section 2.1.2). *Model-based* approaches are less common at this point of time and most recent work is view-based or uses template matching. We give an overview of methods for all three approaches and contiue with a more detailed description of image features for view-based recognition in Section 2.1.3. The majority of state-of-the-art methods of view-based approaches aggree on local descriptors to build sparse object representations. Another important concept is distinctiveness that makes feature matching more robust. A huge body of work about local descriptors exists and a recent comparison was published by Mikolajczyk et al. [47]. Their evaluation does not cover some more recent edge-based and biologically-inspired features. We introduce and discuss the latter and only the best features of Mikolajczyk's evaluation. Some methods model *spatial relationships* between local descriptors. Recent work of this body is presented in Section 2.1.4.

In Section 2.2, we introduce the field of one-class classification which is quite unattended and only recently more comprehensive work is published that connects different approaches to one-class classification. Symptomatically, most of the work in this field does not follow a homogenous terminology and we first summarise the most prominent terms.

The work presented in this chapter is also a summary of our research on methods for fault detection. At the end of major paragraphs we sketch reasons for our decisions that led to the fault detection method as we are proposing it.

2.1 Object Recognition

2.1.1 Objectives and Terminology

The term *Object Recognition* is usually used in two notions. First it is the Recognition of *one* object that has seen previously. Second it is the commonly accepted term for the subfield of computer vision that deals with *object classification* - that is, assigning a given object instance to its appropriate object class. Another term that often falls under

the umbrella term of object recognition is *object detection* - determining the presence and location of an object in an image. Sometimes object recognition is also referred to as *object identification*.

Our method for fault detection takes place after preliminary object recognition. After object recognition our fault detection method performs a task which is close to object classification. An one-class classifier of our fault detection method yields one of two classes. Instead of predicting the presence or non-presence, or the first or the second object class, our method yields a faultless or a fault class.

2.1.2 Approaches to Object Recognition

Model-based approaches

Model-based approaches rely on an explicit 3D mesh model as a representation of a target object. These polygonal surface meshes are usually triangular and sometimes also have normals for a more precise curvature description. Model-based approaches have the challenge of matching the mesh model to the target object in the image, which are two representations that are very different. The 3D models are processed to a different shape or point representation that should be suitable for general shapes, be robust to clutter and occlusion and be efficient [31]. Some of the better approaches simplified their method by making assumptions to the appearance of the object shape. There exist mainly three dominant methods for matching. They are *alignment*, *correspondence* and *registration* of shape or point representations. These methods are capable of inferring the objects pose and the camera parameters from the test images. Prominent methods in this body are Huttenlocher and Ullmann [29], Rothwell et al. [67], and Johnson and Hebert [32]. Forsyth and Ponce offer a survey of model-based vision in Chapter 18 of *Computer Vision: A Modern Approach* [19]. While Hebert still works with 3D model representations [36], model-based approaches are not common any more. Much of the research has been done in the 1980's and early 90's.

Model-based methods have problems to deal easily with texture information. This includes also difficulties with finding suitable answers to illumination. The latter is of particular relevance for fault detection under real-world conditions. The majority of view-based methods outperforms model-based methods significantly in speed and recognition performance. However many of the fundamental problems, e.g. view invariance for descriptors, feature geometry and modeling of spatial relationship, remain the same and model-based literature is a valuable source for those.

We modeled the transmission line spacer that is the target object in our work and hold an usefull 3D model-representation of the object, that could be used also for further tests with model-based approaches.

View-based approaches

Current view-based approaches use local distinctive descriptors. Some methods build a hierarchy of continuously more abstract features. In general an object model is learned in three steps:

1. Extraction of local view-invariant descriptors
2. Matching extracted descriptors over different input images
3. Modeling spatial relationship of descriptors

Matching is primarily done for robustness and rejects not plausible descriptors. Not all methods model the spatial relationship of descriptors. In general the methods are not restricted to input images of special views, rather they learn to find correspondence between features automatically and build an efficient representation by a kind of clustering. However, it is usefull to know about the invariance properties of the local descriptors to use an appropriate view collection for learning, that covers the objects appearance completely. In our case of a rigid single target object we could use randomly distributed views of the object or use an equally spaced view representation as described in [56]. Prominent state-of-the-art object recognition frameworks that can be adapted and used to build a complete abstract 3D object representation based on an complete set of object views in view-based representations are the probabilistc appearance model of Pope and Lowe [59], a sparse representation of the constellation models of Fergus et al. [18], the modeling with affine-invariant descriptors and multi-view spatial constraints of Rothganger et al. [66] and a biologically-inspired system by Serre et al. [73]. The work from Brown and Lowe [5] and from Rothganger et. al [66] are capable of recovering the camera parameters, that includes the point of view per image, from the models.

Our method performs fault detection on a specific object viewpoint. Invariance properties of current local descriptors make our method robust to some degree of pose change.

Template Matching

Another popular approach in recent years has been the use of statistical classifiers finding 2D templates in images by matching. Usually a template search is done over all possible positions and a specific scale range. The 2D template representation and matching varies with the method. Popular approaches deal with detection of faces, pedestrians and road signs. Prominent examples include the Viola and Jones real-time face detector, that uses a cascade of classifiers to improve performance [82], the eigenfaces face recognition algorithm presented by Turk and Pentland [78] and Papageorgiou and Poggio's work on pedestrian detection [53].

2.1.3 Image Features for View-Based Recognition

For an evaluation of some of the current local descriptors see Mikolajczyk et al. [47]. They classified the descriptors into *distribution based* descriptors, *spatial-frequency* techniques, *differential* descriptors and other methods, which helps to understand the variaty of available descriptors. This section discusses and describes, in the following order, features properties, distribution-based features (especially the *SIFT* features), edge-based features, biologically-inspired features and feature combinations. In the end, the description passes to a discussion.

Feature properties The current *state-of-the-art* 3D object recognition frameworks agree on *local descriptors* as low level features. Local descriptors are features that are extracted on specific points of an object, but also can cover larger regions. They behave advantageous for arbitrary occlusion and are more flexible and efficient to describe the appearance of an object from different views. Local features also offer a solution to the need for invariant features. Heisele et al. showed in a comparison of local and componant-based features for face recognition [24], that local features outperform global features in his setting. There is still a discussion about using also global features, especially in combination with local features, for instance Lee et al. [41] and Le and Satoh [40] present work using global and local features and reported some improvements in speed and accuracy. Global features, that are features that cover the entire object, can be included in the set of local features [82] or can be built by organising local features in a hierarchy to increasingly "more global" features [59, 71].

The idea of local features was recently combined with an earlier idea of *interest points*, that can be traced back to the work of Moravec [49], that was improved by Harris and Stephens [23]. Those local descriptores allow for a sparse object representation, that is still *discriminant* between other objects.

Distribution-based features Recently Lowe's work of *SIFT* features [44, 43] gained considerable attention in the field of object recognition. His features are biologically inspired, but use a different computational mechanism. They show good performance for arbitrary pose change, are quite robust against view change and scale good for larger databases. SIFT features are used succesfully for different task. One major drawback is, that these features are especially usefull for textured objects. Experiments with Lowe's publically available demo software showed very bad matching performance for our *low-textured free-form* target object. It seems clear that all similar descriptors of this type will not do better, especially, because Mikolajczyk et al. [47] claims SIFT features to perform best of those he tested.

The local affine-invariant descriptors used by Rothganger et al. [66] are also distribution-based features. For more see the work on feature comparison of Mikolajczyk et al. [47].

Edge-based features This category of features is not evaluated by Mikolajczyk et al. [47], because it consist of feature that are more recently published or do not strongly enough act as an interest point detector. Mikolajczyk et al. [48] presented local edge-based features that can be integrated into Lowe's SIFT framework, Jurie and Schmid [33] proposed local shape features based on salient convex local arrangements, Carneiro and Jepson [9] developed phase-based local features, that are designed for greater illumination changes and Carmichael uses localised, sparse edge density operations [8]. Fergus et al. [17] introduces a type of curvate features. Earlier Pope [58] presented a hierarchy of edge features, that showed to have problems with free-form objects [56] not robustly matching the initial pairings.

Four of them [58, 47, 33, 17] rely on Canny [6] as the underlying edge extractor, Carmichael [8] only mentions that he is using some kind of binary edge images and Carneiro and Jepson [9] used a new low level edge representation. These local edge-based features seem all to be quite comparable and similar. Plagemann et al. [56] reported a problem with Pope's features. The initial pairing was only robust for objects with a very distinctive curvature. We suggest that other edge-based features will have a similar problem.

Features inspired by visual cortex Serre et al. [72] proposes an extension to the *standard model* of Riesenhuber and Poggio [61] that they further improved in [71]. In contrast to the other methods it uses Garbor filters as low level edge detectors. The architecture consists of a hierarchy of features, which are increasing more complex and which have more global visual fields by going up into the hierarchy. Our experiments showed that the C2b response maps behave good for shaped as well as for textured objects. Patches can be learned to obtain same kind of interest points and Serre et al. [72] reports better performance than the SIFT features. The description of the features is not primarily intended to give insight into a runtime efficient implementation and they are computational more expensive. We are using a subset of these features and consider a bypass route of the architecture in [71] that was earlier published in [72]. This choice leads to a more light-weight architecture and is computational faster.

Combination of features and discussion Most of the local descriptors are rather homogenous. Serre's features are one of the few methods that show good performance for a variaty of object appearances. The work of Fergus et al. [18] tries to overcome the heterogenous features by using a combination of three different features, that are somewhat complimentary in their properties. They use Cadir and Brady [34] that favors circular regions, Harris [23] for interest points and Canny-edge-based features [17, 6] for curves. Another way to approach 3D fault detection would have been to use a set of edge-based and distribution-based feature descriptors for representing our target object. However, the sparseness, which is of great importance for performance and scalability of object recognition methods, is a drawback for a fault detection method. We want

to cover the whole area of the object to be able to detect changes at every region of the object. Distinctive local features crowed on salient points. Another property of current local features is that they do overlap in an uncontrolled manner. An unifrom distribution would be favorable for fault detection. Serre's features are response maps over the entire image. A more uniform patch extraction seemed to be easier than for other features.

2.1.4 Spatial Relationships between Local Features

The modeling of a kind of spatial relationships between local features helps to discard false matches and enalbes methods to do pose estimation. Comprehensive descriptions use 3D models of local features, but this approach is computational expensive and complex. Another approach is to approximate exact models with probabilistic models of object parts and their positions. Some methods make a strong assumption to the problem of spatial modeling and assume, that the object is planar. In this case a modified General Hough Transforms can be used to model the spatial relationship. Some work demonstrates that object recognition also can be done with state-of-the-art performance without taking into account any spatial relationships at all.

Systems were proposed that can totally recover the 3D structure of an object from local descriptors [5, 66]. Since several years Rothganger et al. [66] continued to use this approach, despite the fact that 3D representations were quite uncommon the last 10 years. Models using both a 3D representation and local descriptors that were extracted from images are also know as *mixed-representations*. Recently other groups revisited this idea. For instance the appraoch of Brown and Lowe is similar, but they solve camera positions together with the feature matching problem. There approach so far is not capable of recognising objects with this 3D-model representation.

Pope and Lowe [59] introduced a representation called *probabilistic models of appearance*. There approach was based on the extracting of a hierarchy of features from a view-based representation and clustered similar views together. Inside these clusters position and appearance of the features were modeled in a probabilistic manner. Plagemann et al. [57] proposed a method that realised pose estimation on these clusters. For several years Fergus et al. [18] are working on constellation models which are especially usefull for modeling classes of objects. Constellation models model appearance and positions of distinctive local descriptors in a probabilistic manner.

Objects with planar surfaces and planar wiry-objects have been successfully recognised by using modified General Hough Transforms [43, 48]. The underlying Hough Space is extended to pose parameters. General Hough Transforms provide invariance to view to some extend and pose estimation is possibles.

The "standard model" from Riesenhuber and Poggio [61] models findings in primate visual cortex. In the upper regions of inferotemporal cortex, neurons exist which are tuned to different views of an object. Several view-tuned units can model an object appearance for different views. Serre et al. [72, 71] presented a concrete implementa-

tion that follows this approach and performs comparable to state-of-the-art approaches. Their quantitative model is inspired by pathways in the ventral stream of visual cortex. The dorsal stream, which deals with spatial relationships of shape information, is not considered (see more in Section 3.1).

2.2 One-Class Classification

The task of *one-class classification*, classifying a data point as a member of the target class or as an outlier, gained less attentation compared to two-class classification problems. However several one-class approaches emerged which were denoted in various ways. We will shortly clarify the nomenclature, mention different type of one-class methods, and focus on the boundary methods in the last paragraph of this section. The most prominent recent approach to one-class classification is the *one-class SVM* from Schölkopf et al. [68] which is a boundary method. It is the most frequently used method in recent work and we shortly sketch the origin of boundary methods, mention different boundary approaches, note recent applications, and make some performance considerations compared to two-class classifiers.

Nomenclature of one-class, two-class and multi-class classification Several other terms were used for the task of *one-class classification* which itself originated from [51]. Other terms include *outlier detection* [63], *novelty detection* [3], and *concept learning in the absence of counterexamples* [30]. The different terms originate from the different applications to which one-class classification can be applied.

Classification task with more than one class are usually denoted as *multi-class* classification and *two-class classification*. The multi-class classification problem can be decomposed into several two-class classification problems [20]. The two-class classification problem is the basic problem and we always will use this term, as it is common practice, when comparing one-class with multi-class classification problems.

Different approaches to one-class classification Two-class and one-class classification are methodologically very close to each other. Several approaches to one-class classification originate from two-class classification methods. Some [65, 38] follow the most simple approach to generate outlier training data around the target class and afterwards use a two-class classifier, other [4, 45, 64] use more advanced Bayesian approachs, and *density methods* [1, 3, 74, 54, 63] try to model the one-class distribution statistically. A more recent approach to one-class classification are *boundary methods*, that are only considering a closed boundary around the target set and are discussed in the next paragraph. Tax [75] provides a quite comprehensive survey to those methods mentioned above.

Boundary methods, their application and performance Boundary methods are in some way the "pure" one-class classification task. They are mainly a result from Vapnik's principle that we should never solve a problem which is more general than the one that one is actually interested in. A preliminary boundary method was developed by Vapnik et al. in 1962. They proposed an algorithm for seperating unlabeled data from the origin with a hyper-plane [81, 80]. In the following time the originating learning theory and two-class classification gained most of the attention with ground-breaking work of Support Vector Machines (SVM), e.g the C-SVM of Cortes and Vapnik in [12] and Vapnik [79]. Some other work on boundary methods were for instance the attempt of Moya et al. [50, 51] to model boundaries with a neuronal network. In 1999, Schölkopf et al. [68] proposed the *one-class SVM* that was important to the field of one-class classification as their work on SVMs was for two-class classification. Up to our knowledge there exist no comprehensive comparison between one-class classifier performance, but the one-class SVM is prominent in most recent state-of-the-art approaches which make use of one-class classification. Tax [75] proposes another more recent boundary method, the *support vector data description* (SVDD), which is in fact an extension to the one-class SVM and is capable of fitting a more complex decision boundary around the target class. Current applications of the linear one-class SVM include a modified version for content-based image retrieval of Yunqiang Chen et al. [11], a fMRI analysis by Hardoon and Manevitz [22], and an online kernel algorithm for abrupt change detection in music data by Desobry et al. [13].

In the majority of applications, one-class SVMs are outperformed by comparable two-class classifiers. Hardoon and Manevitz [22] report that the linear one-class SVM performed worse for their fMRI compared two a two-class method. Tax [75] evaluates and compares SVDD with a SVC with polynomial kernel of degree three. His experiments are controlled, in the sense that synthetical data with known distribution is used, and his SVDD method performs worse compared to SVC, but fairly little. In fact, only the yeast gene data set of Raskutti and Kowalczyk [39] is reported to yield arbitrary higher performance for an one-class SVM. Their data has high dimensionality with a majority of "noise" features and only very little meaningful but highly-discriminant genes. This setting is comparable to our problem. First we learn distinctive features and afterwards, some before-hand unkown features will indicate a fault. The remaining features do not contribute to detection and become "noise" features. This similarity, together with our good results, is remarkable, since we did not know about the work of Raskutti and Kowalczyk [39] until the end of this project. This similarity further motivates to neglect two-class classification in favor of an one-class object model for fault classes with high variation of appearance.

3 Foundations

This chapter introduces features inspired by visual cortex of Serre et al. [72, 71] (Section 3.1) and theoretical foundations of the one-class SVM proposed by Schölkopf et al. [68] (Section 3.2).

Our method for fault detection under real-world conditions makes use of the good invariance and specificity properties of the biologically-inspired features. We present neurophysiological theory (Section 3.1.1), the computational model architecture (Section 3.1.2), and how model and theory are connected. Feature computation is strictly feedfoward and is related to findings in Cognitive Science that primates are capable of "immediate recognition" within approx. 150 ms. Serre et al. proposed recently a basic [72] and an enhanced (January, 2006) [71] version of their features. Our method incorporates the basic version of these features. Section 3.1.3 presents the implementation of the computational model in detail. The implementation is based on the free available MatLab source code of Serre et al. [72]. This section is restricted to what we are using in our method. Section 3.1.5 summarises relationships and modifications of our implementation and of the versions of Serre et al.. Several parameter adaptions and one major adaption has been made to the basic version. We changed substantially the learning procedure of the object-specific features. Section 3.1.4 is where our fault detection method of Chapter 5 interacts with the original features. We propose shortly learning and extracting features with respect to our method.

In Section 3.2, we introduce the theoretical foundations on how an one-one-class SVM of Schölkopf et al. [68] is learned and which properties the learned model has.

3.1 Biologically-Inspired Features

In Section 3.1.1 we introduce neurophysiological theory and key concepts about primate visual cortex. Neurophysiological experiments proove the ability of primates for "Immediate Recognition". Primates are capable of performing object recognition rapidly and without the need for attention. This finding suggests that a path in the ventral stream of visual cortex exists that is feedforward. At the end of this section, we state the visual areas of the ventral stream and their function. This section and the computational model disregard the role of feedback and are restricted to a bypass route of the feedforward path in the ventral stream of visual cortex. At the very end of Section 3.1.1 four points summarise the mostly accepted theory about primate visual cortex.

In Section 3.1.2 the general model architecture is stated and the mapping between the-

ory and model is summarised (see Figure 3.2). The architecture is strictly feedforward and is based on a former model of Hubel and Wiesel [26, 27]. It consists of alternating simple and complex cell layers. Simple cells provide selectivity and are tuned to *different preferred* stimuli, and complex cells provide invariance and are tuned to the *same preferred* stimuli. The model has two main operations which are designed for invariance and selectivity.

Section 3.1.3 describes the layers and features and summarises parameters of features in Table 3.1. Some parameters were modified and others are combined from the older [72] and the more recent [71] work of Serre et al.. In this work layers S1, C1, S2b and C2b are used and the final features consist of a set of C2b Euclidean responses.

In Section 3.1.4 an algorithms for learning features and one for extracting learned features are proposed. Learning and extracting differs from the procedures proposed by Serre et al. [72, 71]. Both algorithms are specific to our fault detection method and connect theory presented in this chapter with our techniques for fault detection on a limited view presented in Chapter 5.

Section 3.1.5 summarises modifications on the original features and how our setting is related to features of Serre et al. published in their older [72] and their more recent [71] work.

3.1.1 Feedforward Path in the Ventral Stream of Visual Cortex

Primate visual cortex covers the areas in the brain that account for the ability of primates to see. Cognitive Science distinguishes two main cortical areas. The dorsal stream, sometimes refered to as the "Where Pathway", is associated with motion, representation of object location and the control of eyes and arms, whereas the ventral stream or also know as the "What Pathway" is mostly concerned with pattern recognition and object detection.

Findings suggest, that there exists a rapid feedforward path for object detection in the ventral stream of visual cortex. The features we are using are primarily inspired by a bypass route [52] in this path. Even the path is likely not to be exclusively feedforward, top-down connections seem to play a less important role in the main direction of flow than in other areas. The bottom-up architecture can be organised into several visual areas. The overwhelming amount of theories for primate vision is still fragmentary, but for some visual areas, such as in primate visual cortex (V1) already a detailed understanding exists.

"Immediate" recognition in the ventral stream EEG studies of Thorpe et al. [77] investigated that human vision if capable of detecting an animal in natural scenes within 150 ms. Immediate recognition [71] is possible without eye movements or shifts in attention [60]. More recent results of FeiFei [15] and FeiFei et al. [16] suggest that immediate recognition seems to be biased towards natural scenes. Individuals show a tendency to classify indoor scenes rather as out-door scenes, if they are confronted with

a rapid classification task. A possible conclusion would be that immediate recognition uses an universal set of features, which are evolutionary tuned to natural stimuli and are less suitable to differenciate non-natural stimuli. Most recent results point towards the fact that the information flow across visual areas is characterised by a feedforward architecture in the first 150 ms of visual perception.

About the role of feedback Anatomical work suggests that feedback connections are more abundant than feedforward connections [2]. Top-down connectivity is seems to be important for controlling information flow, but it is also often local and seems not to play a major dynamic role in the first 150 ms of visual preception. Theory assumes that feedback is setting-up subnetworks for task-specific usage by selecting and modulating connections. These top-down connections are loosely speaking kind of "programs" for tasks as "Is this an animal?" or "Which size does it have?" and are, e.g., responsible for shifts of attention. They seem to be located mainly in inferotemporal cortex (IT), but reach down to the lowest level V1. Also LGN recieves strong feedback connections from V1. Besides controlling attention, local feedback loops almost certainly have additional key roles, see [55].

Architecture and function of the ventral stream The feedforward path starts at primary visual cortex (V1) and has connections up to prefrontal cortex (PFC). The main path runs from V1, over V2/V4 and inferotemporal cortex (IT) to PFC. V2 and V4 are extrastriate visual areas, TEO and TE (see Figure 3.2) are two areas in IT and anterior inferotemporal cortex (AIT) is a region of IT at the top of the ventral stream. A simplified depiction of the ventral stream is shown in Figure 3.1. A bypass route [52] exists that directly projectes from V1/V2 to TEO in IT thus bypassing V4 (see Figure 3.2, blue arrow). The features [72] in this this work correspond to this bypass route.

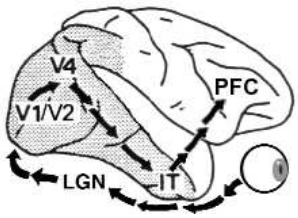


Figure 3.1: The ventral stream of visual cortex. (modified from [62])

Recordings by Hubel and Wiesel [28] in V1 of the cat and the macaque monkey were a breakthrough for the understanding of visual cortex. However little comprehensive theories of function and architecture exist beyond V1. This is mainly due to the complexity of the brain, the need for specialists from different areas and heterogenous data

of specialists. Hubel and Wiesel showed that cells in V1 respond preferably to oriented bars in receptive fields and developed a simple (V1) and complex cell (V2) model for visual cortex. Simple cells provide selectivity for some preferred stimuli and complex cells provide invariance to translation (to some degree), scale and view-change. In V1 invariance properties are rather small. A receptive field of a sensory neuron is a region of the visual field that is determining the neurons firing [25, 35].

Receptive field size and complexity increases along the ventral stream. An increase of complexity can be, e.g., considering additional motion for a specific oriented bar. Units in upstream direction store more complex shape information. From V4 to IT preferred stimuli of units are learned in a task-independend manner depending on visual experience tuning. Preferred stimuli of units in IT represent shape information and it is assumed that something like an overcomplete dictionary of shape features exist [62], which is modulated and combined depending on the specific tasks.

AIT represents the highest level of shape abstraction in the ventral stream, where units are tuned to specific object-views (such as faces) and lightning conditions. AIT units have the property to be robust against arbitrary view-change, translation and scaling. Logothesis et al. [42] trained monkeys to paperclips and proposed that cells in IT show invariance of 20° to view-change, of 2° of visual angle to translation and of two octaves to scale. View-invariant units in AIT have are learned object-specific, but achieve those invariance properties immediately without additional training or experience. Curcuits running between AIT and RFC are linking memory and action and represent category signals. RFC recieves high level shape information for abstract decision making.

According to Serre et al. [72] we can summarise the properties of ventral stream, that are mostly accepted, in four points:

1. Invariance increases in upstream direction, from small translation and scale invariances in V1 to more complex invariances of lightning conditions, view-change and larger invariance to scale and translation.

2. Higher level units are connected to more subunits and have bigger receptive field sizes of higher complexity of the optimal stimulus.

3. The core architecture is feedforward.

4. Plasticity and learning probably at all stages with a time scale that decreases from V1 to IT and PFC.

3.1.2 Feedforward Computational Model of the Ventral Stream

This section introduces the modeling of the ventral stream as it is proposed by Serre et al. in [72, 71]. We focus on the main architecture of their quantitative framework, on how the model evolved and how the model architecture maps to the primate visual cortex. For details on the implementation see the Section 3.1.3 and for a description,

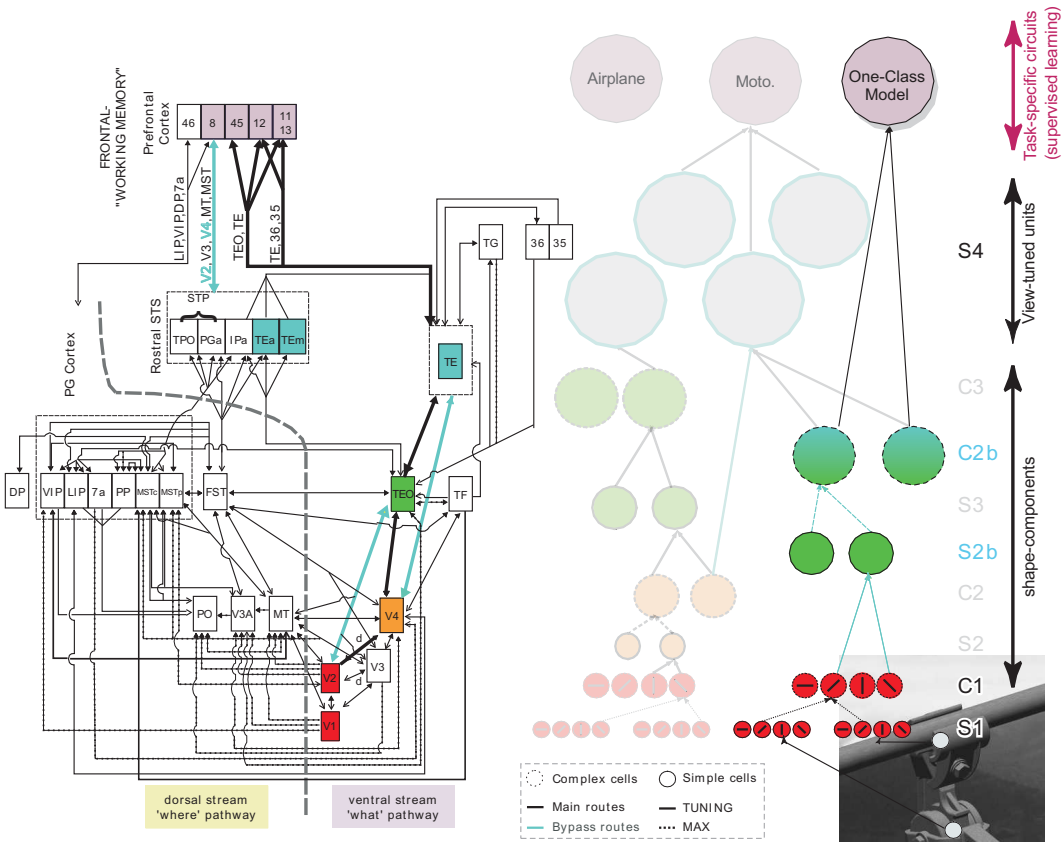


Figure 3.2: Tentative mapping of areas in the ventral stream of visual cortex (left) to the computational model architecture (right). The modeled pathway of this work bypasses V4 and view-tuned units and ends in the one-class model (). (modified from [70])

which modification were made to the frameworks proposed in [72, 71], see Section 3.1.5. We want to quote literally a passage from [71], which gives a good impression, of what the authors themself expect from their model:

"The theory may well be incorrect but at least it represents a skeleton set of claims and ideas that deserve to be either falsified or further developed and refined ..."

In fact their results are ourperforming most of the state-of-the-art object recognition system, but it should be kept in mind, that this is one of the first quantitative theories of the ventral stream which was primarily focused on explaining visual recognition.

Feedforward model architecture

The model builds up on the model of simple and complex cells proposed by Hubel and Wiesel [26, 27] and mainly makes use of the four broadly accepted anatomical and

biological findings summarised at the end of the last section.

The model is strictely feedforward assuming that back-projections are likely to be inactive and models an ultra-rapid object recognition. It is build from alternating simple and complex cell layers that compute step-wise more complex and invariant features for bigger receptive fields. Simple cells are tuned to *different preferred stimuli* and provide for object selectivity, while complex cells are tuned to the *same preferred stimuli* and provide invariance to some amount of scale and translation.

Two main operations for selectivity and invariance Two *main operations* model selectivity and invariance for simple and complex cells. Serre et al. discuss three different operations for selectivity, a multivariate Gaussian, a dot product, and a normalised dot-product. The dot-products outcomes are further processed by a sigmoid transfer function for tuning. They all have in common that have a operation for selectivity and afterwards a operation for sharpness of *tuning*. The normalised dot-product is given as

$$y = g\left(\frac{\sum_{j=1}^{n} w_j x_j^p}{k + \left(\sum_{j=1}^{n} x_j^q\right)^r}\right) \tag{3.1}$$

and its sigmoid transfer function g is

$$g(t) = 1/(1 + e^{\alpha(t-\beta)}). \tag{3.2}$$

The multivariate Gaussian as it was used in [72, 61] is

$$y = \exp^{-\frac{\sum_{j=1}^{n}(x_j - w_j)^2}{2\sigma^2}}, \tag{3.3}$$

which uses an Euclidean distance as operation for selectivity and Gaussian tuning for sharpness. Indeed both are very similar and the normalised dot-product can be approximated by a multivariate Gaussian in a high-dimensional space [46]. The normalised dot-product is biologically more plausible, because its calculation can be explained by a simple model neuron.

The specificity operation in Equation 3.2 is a more general version of the invariance operation for *complex* cells. It is in its general form

$$y = g\left(\frac{\sum_{j=1}^{n} x_j^{q+1}}{k + \left(\sum_{j=1}^{n} x_j^q\right)}\right) \tag{3.4}$$

and will be further parameterised with $r = 1$, $p = 2$ and $q = 1$.

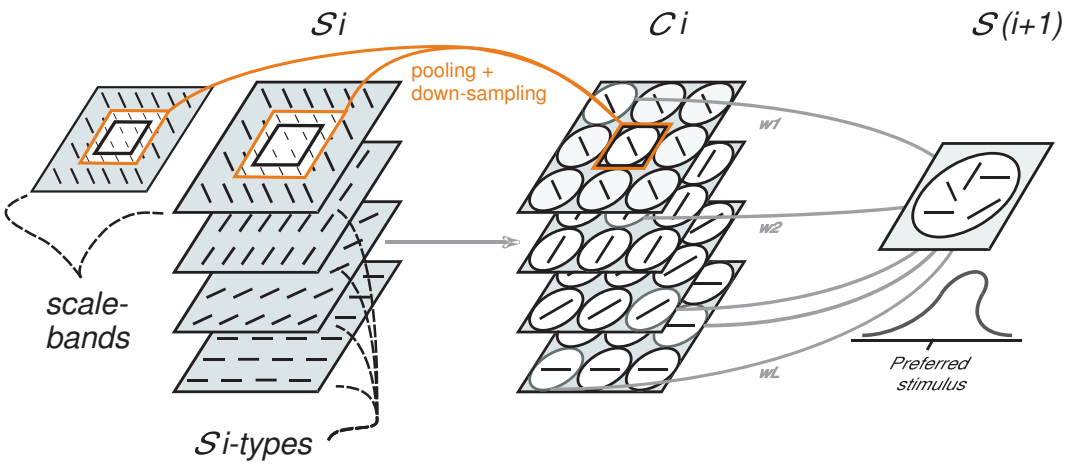


Figure 3.3: Simple and complex cell layer connectivity (modified from [71]).

It is called the *softmax* operation due to its behaviour of a smoothed *max* function, that not only considers the max-value, but also to some extend all remaining values. In former work [72, 61] it was approximated with a simple max-function.

$$y = max_{j \in 1,\ldots,n} \; x_j \tag{3.5}$$

Despite the fact that the softmax function appears to be quite different from the max function, the softmax becomes a max function for $q \to \infty$.

Simple and complex cell layer computation In the following we want to state the computation for complex cell response maps C_p^s and simple cell response maps S_p^s in a general manner. We present formulas for the "normal" case and we will note all aberrations from it for special layers in the Section 3.1.3.

Every simple cell layer follows a complex cell layer and the latter one again a simple cell layer. The main operation for invariance depends on unweighted stimuli x_j of the former layer, the operation for selectivity uses additional weights w_j that represent preferred stimuli. As for now, we consider all w_j as already known and constant. We donate an arbitrary layer that can be either a complex or simple cell layer as L.

Each layer L consists of $K_L \cdot S$ two-dimensional response maps. S denotes the number of scale maps per layer and K_L denotes the number of all S-types which are rectangular patches of different sizes which represent the preferred stimuli of a simple cell layer. The following C-layer has the same number of K_L S-types, because it does not change the preferred stimuli. Imagine an arbitrary simple cell. It has afferents from the former C-layer only in a local neighborhood, but always from *all* former S-types of the former C-layer. The size of the local neighborhood corresponces to the respective patch size. A

patch stores all w_j that correspond to the afferent weights. The recent work of Serre et al. [71] consider also sparse connections which do only have a specific subset of afferents to the simple cell. For each patch the response is calculated for all image positions over all scale bands using the selectivity operation. Consequently the response y of an arbitrary simple cell with input x and patch weights w computes to

$$y = selop(w, x) \tag{3.6}$$

where *selop* is the selectivity operation. See Figure 3.3 for an illustration of cell connectivity. The different layers in this figure correspond to the S-types for layer Ci. The simple cell has afferents from all those S-types from the same rectangle local neighborhood. The S-type or patch for layer $S(i+1)$ shown here has a size of 3×3 and the former layers Si and Ci have 4 S-types. All Si and Ci maps depicted are from one scale band.

A complex cell from Ci does only have afferents x from *one* response map in Si with one specific preferred stimulus. It does not have connections between Si response maps with different preferred stimuli. Its afferents are also from a rectangle local region which side length is called the pooling or grid size N^s_{Ci}. But instead of having connections between different Si-types the complex cell has additional afferents between some scale bands of similar size. The afferents x are from a local neighborhood with regard to translation and scale. The complex cell's response y computes to

$$y = invop(x) \tag{3.7}$$

where *invop* is the invariance operations. Additionally to the pooling over a local neighborhood the response map is down-sampled by the sampling size ϵ_{Ci}. The pooling size is always larger than the sampling size. Down-sampling is essential to avoid a combinatory explosion in higher layers. Figure 3.3 illustrates the pooling with an orange rectangle and the down-sampling with a black rectangle. It shows how the complex cell pools over two scale bands.

The preferred stumuli for every layer are learned from trainings data. Serre et al. [71] learn an universal dictionary of features as well as task-specific features. The process of learning also can be seen as an imprinting of the patches from training data. How we learn features is described in Section 5.2.1 and for more on learning that is not covered in our work see Serre et al. [71].

Mapping model layers to the ventral stream

Figure 3.2 shows the tentative mapping from visual areas of the ventral stream to computation model layers. The left part of the figure is a simplified schema of the information flow in visual cortex. The layers used in our method are S1, C1, S2b and C2b. All

	S1 parameters							
$RFsize$	7 & 9	11 & 13	15 & 17	19 & 21	23 & 25	27 & 29	31 & 33	35 & 37
effective width σ	2.8 & 3.6	4.5 & 5.4	6.3 & 7.3	8.2 & 9.2	10.2 & 11.3	12.3 & 13.4	14.6 & 15.8	17.0 & 18.2
wavelength λ	3.5 & 4.6	5.6 & 6.8	7.9 & 9.1	10.3 & 11.5	12.7 & 14.1	15.4 & 16.8	18.2 & 19.7	21.2 & 22.8
orientation θ	$0°$;$45°$;$90°$;$135°$							
	C1 parameters							
Bands S	1	2	3	4	5	6	7	8
grid size N_{C1}^S	8	10	12	14	16	18	20	22
sampling ϵ_{C1}	3	5	7	8	10	12	13	15
	S2b parameters							
num. S2b-types K_{S2b}	X×4 (for each patch size)							
patch sizes band 1	4×4 ; 8×8 ; 12×12 ; 16×16 ($\times$ 4 orientations)							
patch sizes band 2	4×4 ; 7×7 ; 10×10 ($\times$ 4 orientations)							
patch sizes band 3	4×4 ; 5×5 ; 7×7 ($\times$ 4 orientations)							
num. afferents n_{S2b}	(each patch fully connected)							
	C2b parameters							
Bands S	1 & 2 & 3 & 4 & 5 & 6 & 7 & 8							
grid size N_{C2b}^S	full image size							

Table 3.1: Parameter summary of the feedforward computational model.

other layers, that are not used are toned down. This subset of layers is a fast bypass route [52] and was also used in a former work of Serre et al. [72]. Additionally we are bypassing view-tuned S4 units directly to the one-class classifier. In terms of visual areas the architecture starts at cortial areas V1/V2 (S1/C2), runs over TEO and partially over TE in inferotemporal cortex (S2b/C2b) directly to PFC (one-class classifier), thus bypassing extrastriate cortial area V4 and view-tuned units in AIT (S4). The bypass route cells are trained to specific objects and view-tuned units in AIT are candidates for further improvment of our method of fault detection as well as for viewpoint estimation (see Section 4.2 and Algorithm 4.1). We skipped those parts of the architecture, that we did not use in our work (see [71]).

3.1.3 Detailed Implementation and Parameters

We want to thank Serre et al. [73] and Riesenhuber and Poggio [61] for providing the source code to "A new biologically motivated object recognition system". Our implementation is based upon this implementation. See Section 3.1.5 for a detailed comparison.

S1 Layer

The S_1 layer consists of $4 \times 16 = 64$ response maps of indentical size as the input image. Each map is computed as response to a gabor filter of orientation θ and receptive field size $RFsize$. A slightly modified version of the normalised dot-product (see Equation 3.1) performs the selectivity operation.

Four different orientations of gabor filters are used. Greenspan et al. [21] state that the response to a set of eight edge filters, i.e., an orientation bandwidth of 45°, is sufficient to reconstruct the original edge orientation with more than 99% accuracy. Four orientations are not sufficient to recover the original edge orientation. Determined by experiments, the set of four gabor filters looses about 50% of orientation information. However, the choice of four orientations is sufficient to provide good results and are in agreement with recordings from AIT [61]. The filters provides some degree of rotation invariance for higher layers, while providing still sufficient sensitivity to orientation. Garbor filters are computed to

$$G(x,y) = \exp\left(-\frac{X^2+\gamma^2Y^2}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda}X\right), \tag{3.8}$$

where $X = x\cos\theta + y\sin\theta$ and $Y = -x\sin\theta + y\cos\theta$.
Values are chosen identical to Serre et al. [72] with

$$\sigma = 0.0036 \cdot RFsize^2 + 0.35 \cdot RFsize^2 + 0.18\lambda = \frac{\sigma}{0.8} \tag{3.9}$$

and γ fixed to 0.3. See Table 3.1 for computed parameters for all receptive field sizes and Figure 3.4 for a visualisation of some filter weights. Implicitely, the absolute sum of all negative values equals the sum of all positive values for any gabor filter w:

$$\left|\sum_{w_i<0} w_i\right| = \left|\sum_{w_i>0} w_i\right| \tag{3.10}$$

In the following we start with the version of the normalised dot-product in Equation 3.1 and transform it stepwise to the version used in the implementation. Parameters are chosen to $r = 1/2$, $p = 1$, and $q = 2$ and a gabor filter is denoted as w:

$$y = g\left(\frac{\sum_{j=1}^{n} w_j x_j}{k + \left(\sum_{j=1}^{n} x_j^2\right)^{1/2}}\right).$$

According to Serre et al. [71] this parameter setting with $p \leq q \cdot r$ already has a tuning-like behaviour and it is sufficient to use the identitiy function

$$g(t) = t$$

as tuning function in S_1. It is correct to assume $k = 0$, as long as we are checking for division by zero additionally. This yields

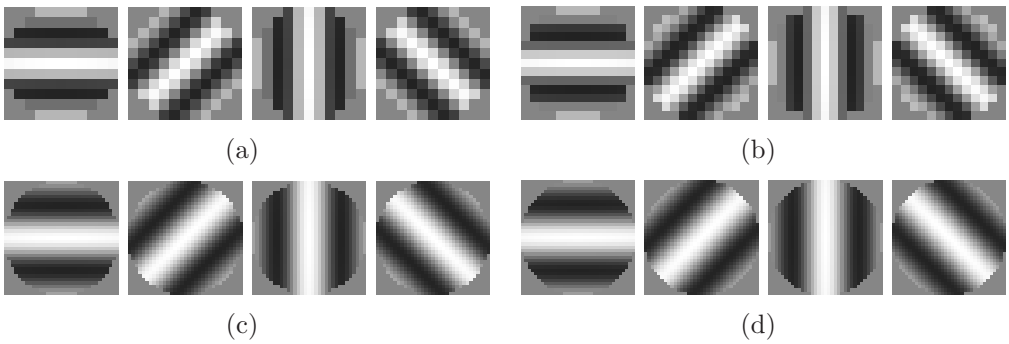


Figure 3.4: Gabor filters for band 2 (row 1) and band 7 (row 2). Each band in C1 is computed as the maximum response over two different receptive field sizes. We use orientations of 90°,45°,0° and 135°. For illustration only two bands are shown, for a comprehensive overview of parameters used in our implementation see Table 3.1. (a) Band 2, size 11. (b) Band 2, size 13. (c) Band 7, size 33. (d) Band 7, size 35.

$$y = \frac{\sum_{j=1}^{n} w_j x_j}{\left(\sum_{j=1}^{n} x_j^2\right)^{1/2}}.$$

Negative and positive correlation are considered equal, and the response computation in the implementation becomes

$$y = \left|\sum_{j=1}^{n} w_j x_j\right| \cdot \frac{1}{x_{norm}}. \tag{3.11}$$

The normalised dot-product is reduced to a convolution over the image with weights w, normalising it with the Euclidean norm of the receptive field image intensities, and taking the absolute value of the result. The values x_j correspond to the receptive field image intensities for a specific position in the image.

C1 layer

In C_1 maps positions pool over a spatial neighorhood and over two receptive field sizes for each output band S (see Table 3.1). They do not pool over different orientations θ. Pooling is performed for every position of two receptive field sizes with the same orientation. The invariance operation is chosen as max operation due to runtime performance. Experiments showed that the $softmax$ operation (see Equation 3.4) was arbitrary slower and the detection performance was only slighlty more accurate. Pooling takes the maximum over a rectangular area of side length $N_{C_1}^S$ and two receptive field sizes that are

joined to one band S. Finally the result is down-sampled with sampling ϵ_{C1} to increase runtime performance. The sampling is usually smaller than the pooling size. For all absolute values refer to Table 3.1. In Figure 3.3 pooling and down-sampling is shown for one position. The pooling grid size $N^S_{C_1}$ is denoted in orange and the smaller inner down-sampling grid in black. Pooling regions of vinical down-sampling regions do overlap and compared to sampling they get relatively smaller.

The sampling ϵ_{C_1} is chosen from Serre et al. [71]. It can be calculated as

$$\epsilon_{C_1} = round\left(\frac{RFsize^S_1 + RFsize^S_2}{2}\right),$$

for receptive field sizes $RFsize^S_i$ in S_1 pooled together to band S in C_1.

S2b layer

Patches in C_{2b} have afferents from all four orientations θ and a local neighborhood of C_1. For experiments we extracted patches from bands 1, 2 and 3, but they were never considered together. All bands were evaluated independly in different runs. For every band different patch sizes are defined (see Table 3.1). Note that the three biggest patch sizes for every band are chosen to be similar in their receptive field sizes. The smallest patch size in band 1 would be too small in band 2 and 3 and is skipped in band 1 and 2. The number of extracted patches X should be chosen as high as computational possible (see Experiment 3 in Section 6.2.2), but only little filtered patches are favorable (see Section 5 for more details how patches are extracted and filtered). Each patch is fully connected to all pixels in its local neighborhood of all orientations.

The selectivity operation in S_{2b} is based on the multivariate Gaussian (see Equation 3.3). S_{2b} is the last layer that contributes to the final features response quantitatively, the exponential function is strictly monotonic, and runtime can be saved choosing the selectivity operation in S_{2b} as the Euclidean distance between patch w and afferent responses x:

$$y = \sum_{j=1}^{n}(x_j - w_j)^2 \tag{3.12}$$

There will be only a small difference in performance for two classification methods that use the multivarite Gaussian and the Euclidean distance.

C2b layer

The invariance operation in layer C_{2b} pools for one patch in an over-simplified manner over all bands and positions yielding one response value for each patch of S_{2b}. Instead of the max function the min function is the invariance operation in this layer due to the

invers monotonic relationship between the Euclidean distance (see Equation 3.12) and the multivariate Gaussian (see Equation 3.3) in S_{2b}.

3.1.4 Feature Learning and Extraction

Feature learning procedure

The learning procedure learns object-specific patches and extracts features in one feedforward computation. Algorithm 3.1 illustrates how feature extraction of this section and patch learning in Section 5.2 interact. First S_1 and C_1 layers are computed for all training images. Afterwards for each region the patches are extracted according to Algorithm 5.1, S_{2b} and C_{2b} layers are computed, and a subset of discriminent features is selected with Algorithm 5.2. The procedure returns the C_{2b} euclidean distance as well as the set of learned patches for all regions.

Note that S_1 and C_1 layer computation also could have done inside the region loop. This could be usefull for relatively small regions in large training images. We preferred to calculate S_1 and C_1 layers for the whole images in order to be more flexible with choosing different or additional regions for patch extraction.

Algorithm 3.1 Learning Biologically-Inspired Features

1: compute layers: Training Images $\longrightarrow$ S1 $\longrightarrow$ C1
2: **for** $r = 1$ to $length(regions)$ **do**
3: $p[r]$:= **extract C1 patches** {see Algorithmus 5.1}
4: compute layers: (C1, $p[r]$) $\longrightarrow$ S2b[r] $\longrightarrow$ C2b[r]
5: $p[r]$:= **select discriminant patches from** $p[r]$ {see Algorithmus 5.2}
6: **end for**

7: return(C2b, p)

Feedforward feature extraction

First S_1 and C_1 layers are computed. In contrast to Algorithm 3.1 this computation has to be done for the entire input image, because we do not know about the object location. The patches of all regions are loaded and S_{2b} and C_{2b} features are extracted. The algorithm returns C_{2b} euclidean distances for all filtered patches of all regions.

3.1.5 Comparative Summary

Serre et al. summarise, how their recent work [71] envolved from [61]. This does not cover exactly the differences to their previous work in [72], which our work is based on. The earlier work [72] describes mainly a subcomponent of [71]. However there are some minor changes between the old and new subcomponent, which we will summarise in the

Algorithm 3.2 Feedforward Feature Extraction

1: compute layers: Images $\longrightarrow$ S1 $\longrightarrow$ C1
2: **load learned patches** p
3: **for** $r = 1$ to $length(regions)$ **do**
4: compute layers: (C1, $p[r]$) $\longrightarrow$ S2b[r] $\longrightarrow$ C2b[r]
5: **end for**

6: return(C2b)

following. [72] is a more technical description, whereas [71] gives a more detailed insight into the relationship between the primate visual system and the model implementation. We transfer some new published concepts on Serre's earlier method to our implementation. Additionally there are several modifications that we made to match our needs. We summarise these modifications afterwards.

Comparison between old and new version of Serre et al.

- **Model architecture**: The S1, C1, S2 and C2 layers presented in [72] correspond to layers S1,C1, S2b and C2b in [71]. S2b and C2b are actually a fast bypass route and not the main object recognition pathway (see Section 3.1.2). There are several new layers in [72] and S2b and C2b are not any more necessary for object recognition in [71], but still have a biological justification. The new architecture is much more comprehensive, for details of all improvements see [71].

- **Two key operations:** Complex cells in [73] use a more realistic modeling of invariance with a softmax operation. Simple cells in [73] were modeled by a biologically more plausible normalised dot-product, followed by a sigmoid-like nonlinearity. In [72] multivariate Gaussian and a simple max operations for simple and complex cells respectively are used, which are more common in the field of computer science.

- **Fully connected receptive fields in simple cells:** In [71] simple cells only use a limited number of all possible connections to input neurons inside their receptive fields. The earlier method in [72] uses fully connected simple cell receptive fields.

- **Learning object-specific features:** The preferred stimuli of simple cells have either to be learned once, building an universal set of features, or learned more often for each object-specific task. We intend to become an expert (see [71]) and learn all patches object-specific.

Our modifications to the older version

- **Sampling of C1:** We modified the sampling of all bands in C1 accordingly to [71] to achieve more equally tuned band properties.
- **Learning preferred simuli in S2:** We developed a learning procedure for preferred stimuli in S2, which is appropriate for our purpose of fault detection. Learning of the features is not covered in this chapter. Our learning procedure is presented in Section 5.2.
- **Number of patches in S2:** We distinguish between the number of extracted patches for learning and the number of filtered patches for fault detection. We do not report a constant number of patches here, but refer for a better understanding of our learning procedure to Section 5.2.
- **Patch sizes and location of patch extraction:** Patch extraction is done from only one band, instead of using all bands as possible feature locations. For evaluation purpose we extracted patches seperately for bands 1,2 and 3, and compared their performance in Chapter 6. Patch sizes for band 1 are chosen as in [72], for band 2 and 3 they are chosen to be comparable to band 1. Band 2 and 3 only use 3 different band sizes.

3.2 One-Class SVM

This work uses the one-class SVM of Schölkopf et al. [68] for the task of one-class classification. We state their solution for the decision border and some of its properties.

The one-class SVM is a direct transfer from the two-class SVM to the one-class domain and has to modify the objective to maximise the margin between to classes which is not any more possible with only a single class. Instead, the one-class SVM separates the single class from the origin with maximum margin. Schölkopf et al. [69] proposed an extension of the two-class SVM which incorporates a smooth and more noise-robust decision border by accepting some degree of error for every training instance. In the case of one class learning, the same technique allows the one-class SVM to draw a smooth border around the unique class. A higher accepts higher fraction of outliers. This solution to the one-class problem is also an estimate for the support of a high-dimensional distribution.

For a threshold $\nu \in (0, 1]$, training vectors $\mathbf{x}_i \in R^n, i = 1, ..., l$, the Mercer kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_i) \tag{3.13}$$

with Φ being the corresponding transformation of feature space, the one-class SVM computes a decision function by solving for the following problem in its primal form [68]:

$$\min_{w,\xi,\rho} = \frac{1}{2}w^T w - \rho + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i. \tag{3.14}$$

w is the normal of the decision hyperplane, ρ influences its distance to the origin and the slack variables ξ_i allow for a smooth decision border. Equation 3.14 is subject to

$$w^T\Phi(\mathbf{x}_i) \geq \rho - \xi_i \quad and \quad \xi_i \geq, i = 1, ..., l, \tag{3.15}$$

assuring that data points lie on one side of the hyperplane. Its dual is

$$\min_{\alpha} \frac{1}{2}\sum_{i,\ j}\alpha_i\alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{3.16}$$

subject to

$$0 \leq \alpha_i \leq \frac{1}{\nu l} \quad and \quad \sum_i \alpha_i = 1. \tag{3.17}$$

All those $\mathbf{x}_i$ that have $\alpha_i > 0$ are called support vectors. The decision function becomes

$$f(x) = sgn(\sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) - \rho). \tag{3.18}$$

The one-class SVM comes with two properties of the two-class SVM. It can be used with different kernels which gives it an arbitrary flexibility and power to model different types of data distributions, and it incorporates the parameter ν which yields a maximum smooth decision boundary in a sense depending on the kernel [68]. The usage of a manual threshold ν for outliers is mainly due to the lack of a comprehensive theory for this problem. If the solution of Equation 3.14 satisfies $\rho \neq 0$ the threshold has the following two properties

- ν is an upper bound on the fraction of outliers.
- ν is an lower bound on the fraction of SVs.

Additionally if all data was drawn independently from one distribution which does not contain discrete components and the kernel is analytic and non-constant it holds:

- ν equals both the fraction of the outliers and the fraction of the outliers.

4 Fault Detection in Aerial Power Lines

This chapter relates our method on fault detection to the task of UAV power line inspection. Our work is part of an project for power line inspection. The project did not reach its final stage yet. The UAV control system is capable of performing pose estimation. We introduce the rough UAV flight procedure, error bounds on pose estimation and the faults in a power line spacer which is subject to inspection (see Section 4.1). In Section 4.2 we propose a procedure for power line inspection by an UAV. Suggestions are made how to extend the biologically-inspired features for pose estimation and object detection. Enhanced accuracy of pose estimation contributes to more robust fault detection performance. Object detection can help to crop images and increase runtime performance. Finally, we summarise parameters of our fault detection method and of the UAV platform that will have to be adjusted for a coming integrated fault detection system, and suggest reasonable parameter usage.

4.1 Introduction

This section presents projects involved in our work, the power lines that are subject to coming inspection, the rough UAV flight procedure and the error bounds for pose estimation of the UAV control system. Section 4.1.2 introduces the spacer that is the main target object in power line inspection of our project. We describe frequent faults and categorise them into four different types.

4.1.1 UAV Surveillance Project

This thesis is part of a research project at the University of Brasília (UnB) done in collaboration with the electricity company Expansion. The UnB research project "Carcarah" aims to develop an integrated UAV solution for exploration, inspection and security tasks. The UAV will be used for power line inspection and this work contributes an automatic approach for fault detection from video images that will be integrated with the UAV control system.

Expansion is a company owned by four spanish electicity companies, holds power networks connecting Samambaia-Itumbiara, Samambaia-Emborcação and Itumbiara-Marimbondo and runs 4 substations in Samambaia, Itumbiara, Emborcação and Marim-

Pose parameter	Error bound
Rotation	6°
View-Change	10°
UAV-line distance	$\|2 - d\| \leq 0.2\ m$

Table 4.1: Error bounds for pose estimation of the UAV control system.

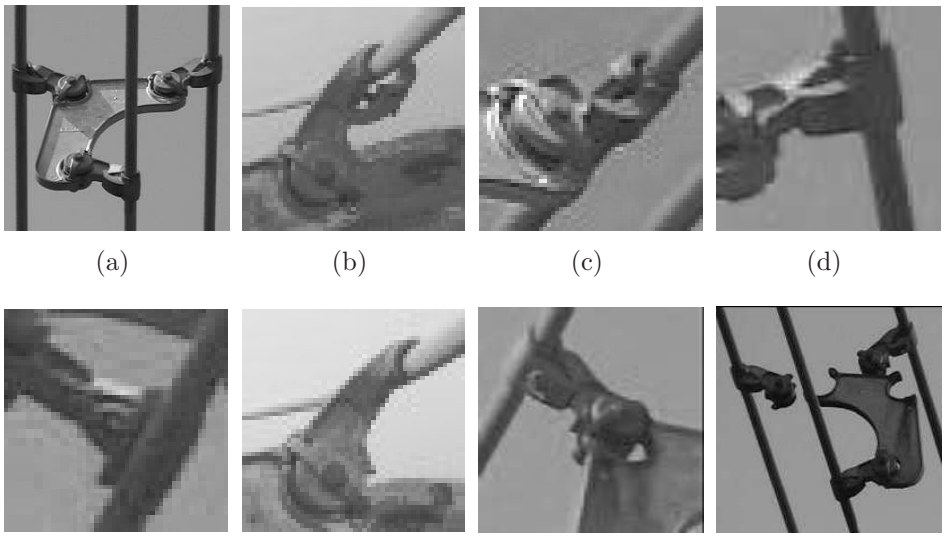
bondo. Operation and maintenance is not conducted by Expansion itself, but by the Argentinean company Transener. Inspection is primarily intended to be done for one type of power line spacer (see Figure 4.1 and 4.2) that keeps bunches of three power lines apart. The 500kV network consists of about 800 km of power lines with about 10 spacers every kilometer, which in total are about 8000 spacers, that has to be inspected on a regular basis.

The UAV is based on a X-Cell 60-based helicopter platform (see Appendix A.1), which is equipped with an Image Source DFK 41BF02 Firewire camera (see Appendix A.2) for video capturing. The helicopter has a remote control for trained technicians and will be guided along the transmission lines based on GPS, line-following techniques and live broadcasted video streams. The final flight procedure is still under development, but at this point it is clear that the helicopter will fly slightly above the lines to avoid hazardous collision and that the procedure will have slow-down regions or stop points close to the spacers. In this way the UAV can gather more detailed video data about the spacer conditions.

The helicopter system will be able to estimate its relative pose towards the power lines within known error bounds. Table 4.1 shows temporary error bounds which are expected for the final UAV control system. We will use an UAV-line distance of 2 m for flights along the lines. Final error bounds will differ slightly. See Section 6.1 and Table 6.1 to compare the synthetic spacer pose variations and error bounds.

4.1.2 Faults in Aerial Power Lines

We introduce faults in a power line spacer that are of importance for power lines inspection by our UAV surveillance project (see last section). The faults have different characteristics and for convenience we divide them into four named classes of which three of them will be modeled in an idealised way as synthetic data in Section 6.1. The inspected aerial powerlines all follow one architecture and consist of bunches of three powerlines that are kept apart by spacers (see Figure 4.1). Unlike the non-symmetric spacer geometricy could suggest, the three lines inscribe an equilateral triangle with sides length of 45.8 cm (see Figure 4.2). The spacer consists of a large non-symmetric body and three equal gripper constructions, which are connected to the spacer body by a hard rubber spring. The spring allows little rotational movements of the gripper



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.1: One faultless and seven fault images of the spacer. (a) Faultless spacer holding the three equally spaced powerlines properly. (b)-(d) Three loose-claw faults. (e)-(f) Two missing-claw faults. (g) An exhausted-spring fault. The spring moved slightly upwards. (h) A released-spring fault.

towards the spacer body and absorbs powerline movements. Each gripper has an integrated claw that together with a seperate claw holds one powerline. The seperate claw is only connected to the gripper body by a bolt.

All images of the spacer shown here are shot by hand-held digital cameras. They are only given for explanation purpose and they all have partially sideways inclined bottom views as they are shot from ground level. In contrast to these pictures, pictures shot by the UAV will have more strongly cluttered background due to appearance of trees, houses and all kind of ground objects.

We distinguish four fault class that appear in the spacer. The most frequently fault is the *loose-claw* fault. The bolt loosened and the separate claw does not embrace the powerline firmly any more (see Figure 4.1, (b)-(d)). After a time the bolt departs, the claw falls, and the loose-claw fault moves on to a *missing-claw* fault (see Figure 4.1, (e)-(f)). The *exhausted-spring* fault (see Figure 4.1, (g)) is a damage at the connection between gripper spring and the body of the spacer. The spring is still connected to the body. Both connections can be deformed. Sometimes one connection breakes and the other is only deformed. When the gripper construction breaks apart from the spacer body we observe a released-spring fault (see Figure 4.1, (h)). Latter can be detected in the same way as the exhausted-spring fault and is not modeled separately in Section 6.1.

4.2 A Power Line Inspection Procedure

In this section we propose a procedure for UAV power line inspection using our method for fault detection (see Section 5). The UAV is able to acquire images from a specific *predefined viewpoint* of the target object through estimation of the UAV pose relative to the power lines. UAV pose estimation may not be always reliable, especially due to the indirect pose estimation of the target object. We suggest an additional mechanism that could assure precise image capturing of the target object from a specific view by using view-tuned S4 units (see Figure 3.2) for pose estimation. Considerable runtime improvement can be achieved when cropping the image to the target object size before feature extraction. The target object is likely to cover only a region in the captured image.

Procedure details The predefined viewpoint *vp* is identical to the central viewpoint (see Table 6.1) our fault detection method was trained to. Lines of the procedure are annotated by the UAV control system *csys* and the biologically-inspired vision system *biosys*. The vision system *biosys* denotes the system that embraces our fault detection method and the additional mechanism. Annotation makes the interaction between both systems more transparent. The output *survey* covers fault detection results for all target object inspected.

At the beginning of the UAV flight procedure the UAV starts (line 1) and afterwards follows a predefined inspection route until the *endpoint*. The control system guides the UAV with a constant distance along the power lines (line 4) and stops when it detects a target object for inspection (line 5). The UAV estimates its pose towards the power lines and the target object, and steers to the predefined viewpoint *vp* (line 6). Depending on the system setup it may be necessary to fly closer to acquire images of higher resolution. The images are acquired by the UAV (line 7), the vision system extracts biologically-inspired features and does pose estimation (line 8 and paragraph below). The procedure will refine the UAV pose, acquire images and estimate object pose repeatedly until the current object pose differs less then the accepted error ϵ from the predefined viewpoint *vp*. The last image is used for fault detection. For on-line fault detection our method will be called at inspection time (line 15) and stores the results to the fault survey. If the method is set up for off-line fault detection the target object image will be stored (line 18) and only processed after landing at the ground station. Commands from lines 8-13 can be skipped if UAV pose estimation is accurate.

Pose estimation with view-tuned S4 units The method for fault detection can be naturally extended by S4 units corresponding to view-tuned units in AIT of visual cortex (see Section 3.1). Each S4 unit is imprinted by an image of the target object from a specific view. To make pose estimation applicable, training images should be chosen approximately equally spaced (e.g., according to [56]) around the predefined viewpoint

Algorithm 4.1 Power Line Inspection Procedure

Input: *endpoint* : end point for inspection
vp : best viewpoint for fault detection
csys : UAV control system
biosys : biol.-inspired vision system

Output: *survey* : fault survey

```
1: UAV takeoff (csys)
2: repeat
3:    repeat
4:       fly along power lines (csys)
5:    until UAV arrives at a new object for inspection (csys)
6:    steer UAV approximately to view vp to object (csys)
7:    img := acquire object image (csys)
8:    vp2 := estimate viewpoint from S4 (biosys)
9:    while (vp2 − vp) > ϵ do
10:      refine UAV viewpoint with vp, vp2 and pos (csys)
11:      img := acquire object image (csys)
12:      vp2 := estimate viewpoint from S4 (biosys)
13:   end while
14:   if ON-LINE then
15:      result := fault detection for img (biosys)
16:      add result to survey
17:   else if OFF-LINE then
18:      add img to survey
19:   end if
20: until arrived at endpoint (csys)
21: UAV landing (csys)
22: if OFF-LINE then
23:   survey := results of fault detection from image survey (biosys)
24: end if
```

vp and each S4 unit has to be labeled with the corresponding view-point of the training image. Pose estimation can be performed approximately by choosing simply the view of the unit with highest response or by learning an interpolation function between S4 unit responses. All features up to S2b only have to be extracted once for pose estimation and fault detection. Only little additional computation time for pose estimation is required for calculation of S4 unit responses. See Serre et al. [71] for more details on S4 units.

Cropping images before feature extraction Cropping images to the target object size before feature extraction can safe arbitrary computation power, but requires the object to be detected in the image first. We did not integrate cropping into the procedure due to the absence of knowledge about the procedure implementation details. However, we want to give some suggestion how cropping images fits into the procedure. Cropping for view estimation in line 8 of Algorithm 4.1 only makes sense if the UAV control system can provide a fast object detection mechanism. A rough detection would be sufficient. For iterations over lines 9 to 13 the vision system *biosys* itself can provide more accurate object detection. Serre et al. mentions, that by tracking the feature extraction paths throughout the layers, object localisation should be easily to be realised. Detection can be integrated despite the originally focus on modeling the "what" pathway. Tracking will be computationally cheap and the feature extraction of the next iteration can be accelerated considerably.

4.3 Parameters of a Fault Detection System

This work is the foundation for a coming fault detection system of an UAV. The UAV flight procedure and the UAV equipment is in devolpment and a final system setup is not possible at this point in time. This section summarises parameters of our method and the UAV which will have to be chosen adequately for a properly working fault detection system. There exist several competing objectives for a fault detection system configuration. It is not possible to configure parameters in a way so as to achieve all of these objectives maximally. The dependencies between parameters are high, a good setting is complex due to the high number of parameters, and some of the parameters act as tradeoffs between other parameters. The user has to prioritise his goals and to find appropriate tradeoffs accordingly. There can be manifold "good" settings depending on the task in mind. This could be an off-line or on-line method, emergency line inspection with the need for rapid and accurate results, or a method for surveillance on a regular basis, just to name a few of them.

In the following we present the parameters, make annotations to each parameter and provide absolute parameter values (see Table 4.2) for some of them. Given values are temporary and they are likely to be adapted in subsequent work.

UAV-line distance The UAV-line distance d_{uav} is the horizontal distance between UAV and the closest power line. The UAV is normally flying along the power lines with a predefined UAV-line distance. The UAV can fly closer to the power line to acquire more detaild images of the object, but it will always keep a minimal security distance to avoid hazardous collision with power lines. So far we plan to use an UAV-line distance of 2 m. The distance can be estimated by the UAV control system with an error bound of approx. <0.2 m (see Section 4.1.1).

Variable	Value
View scale for training (band of patch extraction)	1 px/mm (band 1) 2/3 px/mm (band 1) $\Leftrightarrow$ 1 px/mm (band 2) 1/2 px/mm (band 1) $\Leftrightarrow$ 1 px/mm (band 3)
Target size	w_{target} >200 mm, h_{target} >260 mm
Camera CCD size and resolution	1/2" with $w_{ccd} = 6.4$ mm, $h_{ccd} = 4.8$ mm 1280×960 $pixels$
Scale-invariance of biol.-insp. features	approx. 4.5 *octaves*
UAV-line distance	1.8 to 2.2 m
Camera-object distance	3.55 to 4.26 m

Table 4.2: Data summary of available parameter values.

Camera-target distance At a point of time in the flying procedure the UAV will aim its camera at a target, that has to be inspected. The camera-target distance d_{ct} will always be higher than the UAV-line distance. In our case we assume an augmented 45° view point to one gripper with respect to the power lines (see Table 6.1). Our camera-target distance calculates to a range of approx. 3.55 to 4.26 m.

Target size The target is a rectangular bounding box parallel to the image plane and its sides are aligned with the camera sides. The center of a target object or a constellation of target objects is inside the bounding box plane and constraint its size. The target size is defined by its width w_{target} and its height h_{target}. Values in Table 4.2 equal the minimal view bounding box for synthetic training image creation (see Table 6.1). Optimally the UAV should be able to aim its camera directly to the center of the target object. If this is not possible, the focal length will have to be smaller and the final resolution for fault detection is lower. Figure 4.2 shows a sketch of the spacer that is subject to inspection. This figure provides measures which can be used for estimating minimal target sizes.

Camera parameter We consider the CCD width w_{ccd}, its height h_{ccd} and the pixel resolution of the camera. Our camera supports values as in Table 4.2.

Focal length of lens An central parameter is the choice for the focal length of the camera lens. We use a lens with fixed focal length and fixed focus. An application-specific focal length value can be calculated according to the camera manufacturer [76] as follows: A suitable choice of the focal length depends on the camera-target distance d_{ct}, the CCD dimenions w_{ccd} and h_{ccd}, the target width w_{target} and height h_{target} and the orientation of the target towards the camera. If the camera is being rotated by 90°, consequently w_{ccd} and h_{ccd} have to be switched for use in 4.1. The focal length l is calculated as

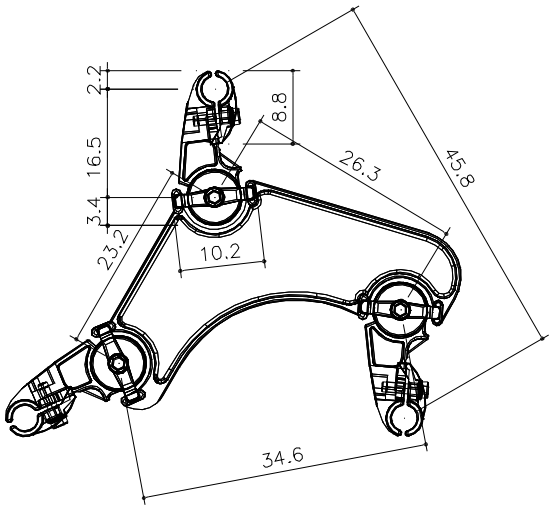


Figure 4.2: A sketch of the power line spacer of our UAV surveillance project. The spacer was modeled in AutoCAD. All measures are in *cm* and have error bounds of < 0.2 *cm*.

$$l = \min\left(\frac{d_{ct} \times w_{ccd}}{w_{target} + w_{ccd}}, \frac{d_{ct} \times h_{ccd}}{h_{target} + h_{ccd}}\right). \tag{4.1}$$

A smaller focal length will yield a higher field of view.

View scale for training Pixel scale is a term usually used in photography and astronomy. It is measured in *arcsec/pixel* and denotes the amount of sky falling on one pixel of the digital camera. To avoid confusions we define the *view scale* measured in px/mm which incorporates the camera-target distance. The view scale denotes the resolution of an object in the image. This notion is quite intuitive giving a fast impression of how many pixel represent one millimeter. The definition is inspired by the notion of map scale and can be used analogously. The view scale is only an approximated measure, which is far of being precise. Even if we assume the object to be flat and identical to the target bounding box, there would be an error due to distortion caused by a small focal lenghts. It is sufficient to assume that the object depth is small compared to the camera-target distance. However, this measure faciliates predictions about the fault detection performance. We can *control performance* by building a real training image data set with a preselected view scale. Mention that the synthetic data set is created using parallel projections and real data has to deal with projective projections. For the latter, there is an unique relation between focal length, camera-target distance and the

resolution of the camera. Neglecting the difference between projections, view scale will be still an usefull measure.

For experiments (see Section 6.2.1) three different bands are evaluated separately. For each band the same training images with the same view scale were used. To make the view scale measure valid for performance prediction, we have to assume that patches are always extracted from band 1. Analogous values of band 1 for band 2 and 3 are given in Table 4.2.

Outlier threshold The one-class SVM provides the threshold ν to control the upper bound on the fraction of outliers. This indirectly gives control over the false or true positive rate, and an adequate threshold can be chosen using the ROC curves. A higher threshold makes the classifier more sensitive to faults, but accepts a higher false positive rate. A lower threshold predicts less images as faults, but misses a higher fration of faults.

Amount of scale-invariance The biologically-inspired features provide invariance from 1 to approx. 4.5 octaves. If this amount of scale-invariance is really needed or how much should be available has to be determined.

Computing power The UAV acquires image data that is send back to a ground station. Image analysis is done remotely, which allows for arbitrary scalability of computing power and therefore it mainly relies on available computer platforms.

5 A Method for Fault Detection on a Limited View

This chapter proposes our approach to fault detection from video images for 3D objects under real-world conditions. We consider fault detection to take place after preliminary object recognition and restrict fault detection to a predefined viewpoint. Our method is designed to be robust to some degree of pose change. Experiments demonstrated that it is robust to about 24° of view-change and to about 12° of rotation in plane. The object model is learned with an one-class classifier independently from any preliminary fault knowledge. This method is capable of detecting unseen faults. Essentially every type of fault can be detected, as long as its degree is large enough to be detected with our method. We consider an object defective if its appearance is different to the object model we learned.

In Section 5.1, we first explore the problem of fault detection from an abstract perspective and present our notion of fault detection afterwards. We reduce the abstract notion of a *space of variablitiy sources* to our approach with features inspired by visual cortex. These features do not model any spatial relationship. We present a notion of fault that is based on the *discriminance* of features with respect to a background class and identify two major problems which are related to the choice of an one-class classifier. Features has to be extracted or selected from within the object and this can not be done implicitly as with a two-class classifier. Fault detection that does not know about specific faults beforehand has to "observe" the entire object and features have to be *object-covering*. In Section 5.2 we propose the learning procedure for the one-class object model used in fault detection. The procedure involves patch extraction (Section 5.2.1), discriminant patch selection (Section 5.2.2) and one-class learning (Section 5.2.3). Patches are extracted from within the object using segmentation masks and some kind of object-covering is supported through regions. At the end of this chapter we propose the procedure for fault detection using the previously learned one-class object model.

5.1 What Makes a Fault?

This question is investigated from an abstract perspective and a first practical answer is given in the second part of this section with the description of our fault detection method.

5.1.1 An Abstract View of Fault Detection

We introduce sources of variability, develop a notion of fault detection for an idealised object model and present a Cognitive Science perspective on how humans could perform fault detection.

Source of variability We restate Carmichaels perspective on object recognition [7]. In his nomenclature a target object can be any arbitrary abstract class of objects. This class can be defined by allowed changes in its *sources of variability*. Carmichaels perspective has two favorable properties, it is general enough to deal with any kind of target classes and it provides a level of abstraction, that allows for an intuitive and better understanding of the problem of fault detection in 3D objects. Sources of variability of an 3D object in a 2D image can be manifold, e.g., illumination, view-point, scale, translation, rotation or *object identity*. Object identity embraces changes in the object molecular structure itself, allowing for more abstract classes as "tables", "human" or "grippers with an opened claw up to 3°".

Fault detection for an idealised object model Any target object is only defined precisely if we have perfect knowledge about allowed qualitative and quantitative changes of its sources of variability. Imagine we would have this perfect knowledge, then the corresponding model of the target object is a subspace in the *variability space*. This subspace is our object model. We introduce the notion of a variability space as the space which dimensions are all possible sources of variability. So far let us continue with this idealised notion of a target object and assume that we already know that we are looking at the target object. In this framework, fault detection becomes a simple task: *Is this instance of the target object an element of our object model or not?*. Furthermore the correct tradeoff between variance and specificity to specific sources of variability is known and defined exactly by the decision border of the object model.

Fault detection by an individual From a more realistic perspective a model of the target object will be hard to be determined exactly. If a human looks towards a car and has to perform fault detection, he profits from two major sources for his model of a car. Primate visual cortex will provide the human with immediate invariance to sources of variability as translation, scaling, rotation, view-point and also partially object identity and illumination (see Section 3.1). Additionally it can use attention to improve its decision. The individual will have no problems to responde rapidly to a car without a fault, that is presented him at any pose or scaling. It could claim a fault, because the back door of the car is opened, but it does not know that the car is used for transportation of a large objects, that blocks the door. A human can *learn* a better model by, e.g., looking towards an object from different viewpoints and tuning its view-tuned units in visual cortex, or by gathering additional information about the problem.

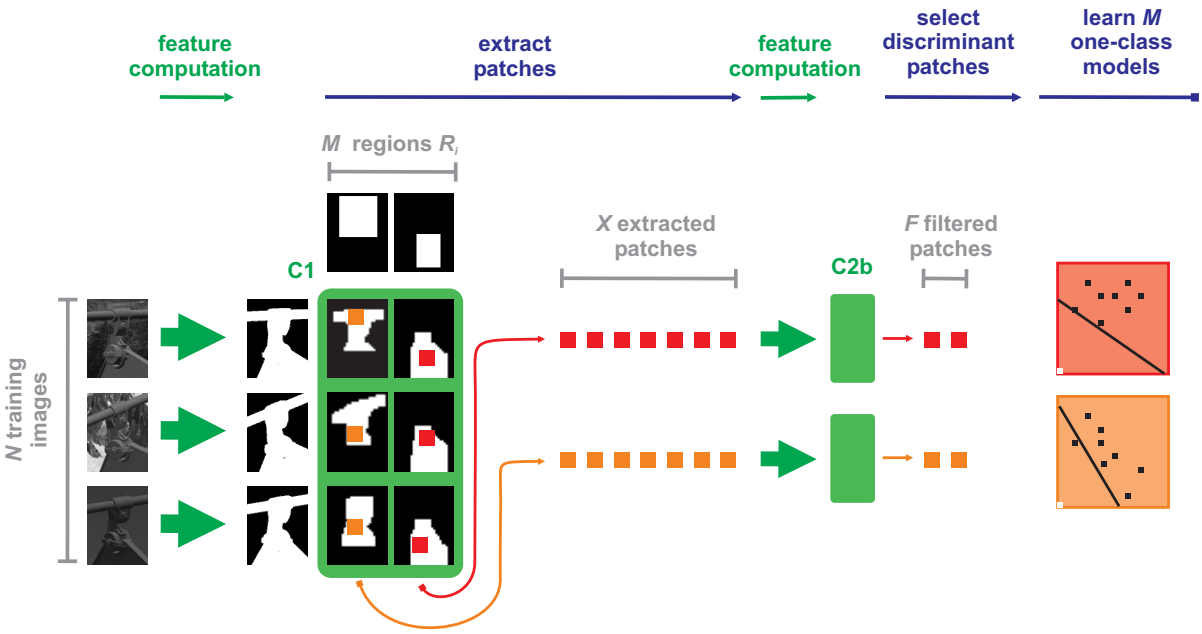


Figure 5.1: Learning the one-class object model. X patches are extracted from M regions R_i, F most discriminant patches are selected, and F-dimensional hyperplanes are learned from N training images for all M regions. Feature response maps C1 and S2b are computated before and after patch extraction, respectively (green).

5.1.2 Our Fault Detection Method

State-of-the-art object recognition features are not able to compute clear noise-free signals from sources of variability. In general the sources of variablitiy are not any more independent from each other, an invariance to one source will lead to a loss of specificity for another source, inaccuracies are high, and so far most methods do not provide at all invariances to all sources of variability. Consequently, machine learning methods can only roughly estimate decision boundaries for object models and, in case of doubt, simpler models should be prefered over complex models due to Ockham's razor - everything else being equal, prefer the simpler hypothesis. Additionally, a good choice of a tradeoff between invariance and specificity has now central importance.

However, the biologically-inspired features of Serre et al. [72] provide several invariance properties and are specific to a target object. They are fully translation invariant, are in same range scale-invariant, and are to some degree invariant to rotation, view-point, object identity and illumination. For last sources, higher invariance always means higher loss in specificity. Their features are working already good for real-world task, but contain still arbitrary noise and inaccuracies.

The transformation of the idealised space of variability sources to our feature space is noisy, lossy and complex. The biologically-inspired features are inprecise, do not consider spatial information of object identity and represent only some regions of the object. Each feature dimension in our space of variability implicitely accounts to some extend for several different source of variability. A features is tuned to some view-point, rotation in plane, illumination, some sample of object identity and a specific region of the object.

Our method is an over-simplification of the abstract notion of fault detection in several ways. The object model for fault detection is built only on one predefined view-point including a specific rotation in plane of the object. Features provide some invariance to this view-point and rotation. Illumination can be learned depending on what is provided in the training data, our object is rigid having low variability in object identity, and we learn only some regions of the overall object surface. Each feature acts as a radial-basis function to a specific constellation of sources of variability.

A fault detection method for unseen faults has to "observe" the entire object surface and optimally analyse features of the entire object region. In fact, selecting those local features and rejecting noise matches without spatial information is difficult. Our method uses predefined regions of the objects where features has to be extracted. Those Regions allow to incorporate spatial domain knowledge to some extend. They are also a computational mechansim to model the human behaviour of focusing the field of vision on a region of a object. The accuracy of selecting object-covering features is controlled by the number of regions defined. The unavailability of spatial information causes another problem. Features, that have only little distinctiveness, provide no means to be assigned to the object or the background. Our method selects features that are most discriminant between a background and object class. This approach biases feature selection towards salient regions instead of object-covering regions. The number and size of regions can be used as a tradeoff between object-covering and discriminance.

Discriminance is central to our notion of fault detection. Highly discriminant features have a larger margin between the background and the target object class and are less likely to be confused due to false matches. Our notion of fault is similar to the question: "Is an object feature missing which was expected to be there?". The use of a linear on-class SVM instead of single feature thresholds gains some more descriptive power and what is considered as missing for one feautre depends also in a linear manner on the response of the other features.

The tradeoff between invariance and specificity is determined by learning with the linear one-class SVM from Schölkopf et al. [68]. A threshold ν enables to control the fraction of outliers accepted by the learned model on the training data. The one-class SVM draws a smooth decision border around the data and finds those data records that should be rejected first when accepting some outliers due to their highest aberration to the overall object appearance.

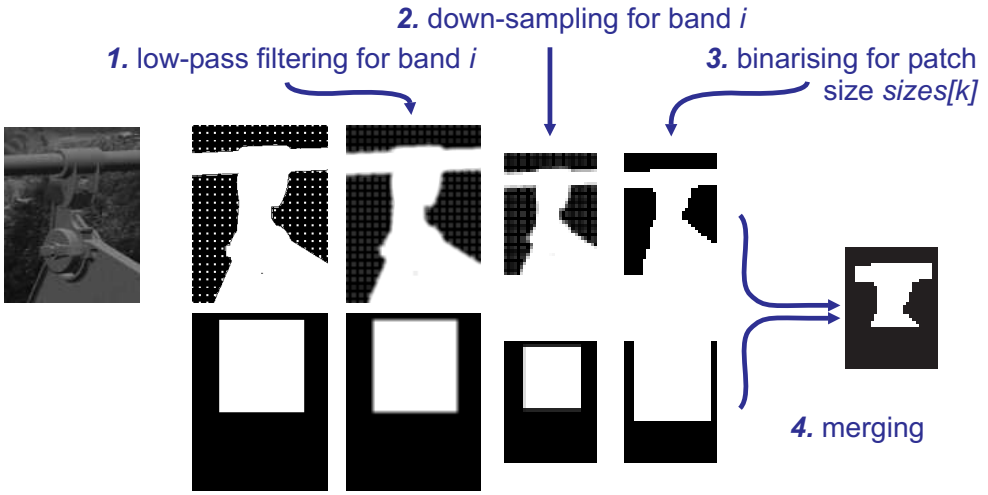


Figure 5.2: Preprocessing and merging of an object segmentation mask and a region mask. Patches are only extracted from within the merged mask region.

5.2 Learning the Object Model

This section proposes the learning procedure of an one-class object model for fault detection. Learning this model is the central technique in our fault detection method. Only patches that were extracted from within an object will probably be usefull for a represention of our target object. Feature extraction from Serre et al. [72, 71] is done from arbitrary positions. In contrast to their method we can not select patches implicitely while learning object classes. A method for fault detection can only detect a fault if the model covers the appearance of this fault region. If we use sparse representations, non-distinctive object regions could be defective without notice of the model. We introduced a simplified notion of fault for our method (see Section 5.1.2) that relies on the concept of discriminance. If background or other objects respond in the same intensity to a patch than the object itself, the patch will not be able to report a fault. If the corresponding object region is defective the outer object will respond instead. In this setting, there are three main properties our learning procedure has to achieve:

1. Patches in the model have had to be extracted from within the object.
2. Patches should be object-covering.
3. Patches should be discriminant.

Properties one and two are approached in the patch extraction stage and number three is done afterwards. We solve problem one by providing segmentation masks for every training image. Our approach can not garantee object-covering patches, but it garantees that patches are extracted from all regions uniformly. The training images are

additionally labeled with region masks. Both masks are merged and the remaining mask defines possible locations for patch extraction. After patch extraction those patches are selected that are most discriminant between the object and a background class. Finally, the one-class classifier is trained on the selected set of patches.

Figure 5.1 illustrates the object model learning procedure. Compare this figure with Algorithm 3.1 for learning of the biologically-inspired features. Both are nearly the same, the figure does not show feature computation and the algorithm does not learn the one-class SVM after feature computation.

5.2.1 Extraction of C1 Patches in Object Regions

This section introduces the procedure which extracts patches in the learning stage (see Section 3.1.5 and Algorithm 3.1). Serre et al. [72] provide a simple patch extraction procedure which is not sufficient for fault detection on a limited view. Following our approach to fault detection using discriminant features, two further problems arise for selecting good patches. Both problems deal with patch extraction position relative to the object and therefore are handled in the extraction stage. Shortly speaking, the first is involved with object segmentation and the second with the need for object-covering patches. We first discuss our solutions to these problems and afterwards give a more detailed description of their implementation.

Segmentation for more robust patch extraction

When using cluttered training images, there has to be a mechanism that will select patches from within the object shape. This requires solving for object segmentation in some way. It is very unlikely that patches outside the object are being usefull for representing the target object. Serre et al. [72] implicitly solve segmentation eliminating outlier patches by learning two or more object classes on labeled training data. In a similar way, we could have learned patches with a two-class classifier on the object and background class. Instead, we require the training set to embrace additional segmentation masks for every training image. This choice contributes to the robustness of our approach, so that even a misleading background class will not confuse patch selection and patch extraction is more exact. Note that segmentation is only required for the learning of patches. Segmentation mask can be provided manually or object segmentation could be done automatically in line 7 of Algorithm 5.1. Our proprocessing of masks will eliminate errors of automatic segmentation to some extend.

Regions and object-covering features

Most state-of-the-art object recognition approaches make use of local descriptors, that are most distinctive (see Section 2.1.2). Our method selects discriminant patches (see Section 5.2.2). While distinctive features enable sparse object representations which

Algorithm 5.1 C1 Patch Extraction Procedure

Input: $c1[1..nImg][1..nBand][1..nOrient]$: C1 response maps
m : number of patches per patch size
$sizes[1..n]$: patch sizes
i : band for patch extraction
$imgRegion$: region for patch extraction

Output: $patches[1..n][1..m]$: extracted patches

1: $mapSize$:= size of $c1[1][i]$ {size equal for all images}
2: $m1$:= mask of $imgRegion$
3: $m1$:= low-pass filter $m1$ with Gaussian filter
4: $m1$:= down-sample $m1$ to $mapSize$
5: **for** $i = 1$ to m **do**
6: j := choose random training image
7: $m2$:= segmentation mask of image j
8: $m2$:= low-pass filter $m2$ with Gaussian filter
9: $m2$:= down-sample $m2$ to $mapSize$
10: **for** $k = 1$ to n **do**
11: $m1tmp$:= binarise mask1 with $sizes[k] \times sizes[k]$ overhanging by $\leq x$
12: $m2tmp$:= binarise mask2 with $sizes[k] \times sizes[k]$ overhanging by $\leq x$
13: $mask$:= merge $m1tmp$ and $m2tmp$
14: $patches[k][i]$:= extract patch from $c1[j][i]$ at random position within $mask$
15: **end for**
16: **end for**

are advantageous for object recognition, the sparseness is not an appropriate property for fault detection. It is obvious that some regions of the object can not be covered in the representation and as a consequence fault detection would only be possible in regions with high distinctiveness, which is unfounded bias. We think of two ways to approach this problem. A more generally approach would have to find a tradeoff between the distinctiveness of the local descriptors and an uniform object covering. The latter approach would be preferable for fault detection, that does not know anything about possible faults. However in practice it is often known, where faults will appear, even though the appearance of a fault can vary arbitrary. In this setting a labeling of some specific fault regions of the objects improves runtime and detection performance. Runtime is improved when removing patches of object regions not needed for fault detection and detection can be improved by reducing noise, that comes from useless patches. Distinctiveness can be incorporated by choosing most discriminant patches for each labeled region separately. Our fault detection method on a limited view of the target object

makes use of a time-efficient region labeling. A region is defined by just one rectangular for all training images. This is only possible due to limited pose change and position alignment for the target object, and equally sized training images. This kind of labeling is usefull for larger regions, but not any more practical for smaller regions for which object pose changes cause relatively high label errors. It seemed to be a reasonable simplification, but it is straight-forward to apply our method with a more general labeling. Arbitrary shaped region masks can be individually defined for every training image easily. Conversely this labeling yields good results and it seems to be unlikely that a more sophisticated labeling will contribute substantially to performance.

Implementation details

The procedure for extracting patches is illustrated in Algorithm 5.1. How the procedure interacts with feature computation while learning can be seen in Algorithm 3.1. S1 and C1 features do not have to be learned and learning takes place on computed C1 response maps. C1 layers are exracted for all $nImg$ training images, for all $nBand$ scale-bands and for all $nOrient = 4$ orientations (see Section 3.1.3). Patches are extracted in different sizes, which are stored in $sizes$. For parameterisation of $sizes$, see Table 3.1. The number of extracted patches per patch size can be specified by m. In all we will extract $m \cdot n$ patches ($length(sizes) = n$). The procedure extracts only patches for a specific region $imgRegion$ of the object. The region label is approximate and used for all training images. Section 3.1.3 and Algorithm 3.1 describe, how features are learned for different regions. The extraction procedure outputs m patches for all n sizes, which are one input paramter for Algorithm 5.2. Values for parameters m and i influence performance. In the following we make annotations to Algorithm 5.1 as appropriate. Masks are preprocessed and merged, otherwise patches could not be extracted close to borders, on slightly to thin object shapes, or when inside the object a small amount of pixels are marked as outliers. The last point makes patch extraction robust to errors in automatic object segmentation. Preprocessing of masks is illustrated in Figure 5.2.

Region and segmentation masks We implemented masks as binary, 1 denoting pixels inside the region or object and 0 pixels outside. $imgRegion$ is given in vector rectangle representation and is converted to a binary masks first in line 2. $mask1$ and $mask2$ have an equal size as the training images. Before they can be used as constraints for patch extraction in band i, they have to be preprocessed and merged, see lines 2-4, 7-9 and 11-13, and the next two points. Mask preprocessing stages are done distributed over the procedure to improve runtime. If we want to use a more detailed region labeling for every training image, lines 2-4 should be moved inside the outer for-loop and specific labels $mask1$ should be loaded together with $mask2$ for every training image individually.

Mask preprocessing Both masks are preprocessed separately in three steps illustrated in Figure 5.2. Patch extraction takes place in the C1 layer of the features in band i.

Masks come in the image size s_I and have to be down-sampled to the response maps of a size s_I/ϵ_{C1}. Before down-sampling the masks are convoluted with a Gaussian filter which is of size appropriate to down-sampling factor ϵ_{C1}. Masks are grayscale after down-sampling. Binarising is performed in step three. Binarising in this case is an advanced operation and is designed to have two properties. The masks are binarised in a way that x percent of the a patch area can be outside the mask. Binarising is done depending on the patch size $sizes[k] \times sizes[k]$. Down-sampled masks consists of contiuous values in [0..1] denoting the fraction of inliers voting "inlier" for the down-sampled pixel. This gives the succeeding binarising additional information and yields more precisely smoothed object borders. The percentage nature of threshold x for outlier pixels makes binarising comparable for different patch sizes. Experiments suggested to choose a value of 1/10 as a threshold. This threshold will provide tolerance to errors of automatic segmentation. When computing the absolute number $\hat{x}$ of pixels allowed to be outliers, we rounded $\hat{x} = round(sizes[i] \cdot nOrient)$.

Mask merging Preprocessed region and segmentation masks are merged in line 13. Both masks are binary and they are combined by the point-wise logical $\wedge$ to

$$mask = m1tmp \wedge m2tmp$$

.

The merged mask is refined allowing only those points that can be covered by a patch of size $sizes[k]$.

Both masks are combined to one mask, that only allows for patches inside of $imgRegion$ and on the object. See line 11 and step 4 in Figure 5.2.

Random patch extraction with a restriction mask Serre et al. [72] extracted patches from arbitrary random positions of randomly chosen training images. We follow this approach, but constraint available positions by a restriction mask which is obtained by region and segmentation mask preprocessing. A patch can only be extracted from positions, where the patch is completely inside of the restriction mask.

5.2.2 Selection of Discriminant C1 Patches

This section introduces an approach for selecting discriminant features which is described in Algorithm 5.2. An one-class classifier is only to same extend capable of selecting discriminant features from a high-dimensional feature space. Two-class classifier can learn discriminant features implicitly while learning object classes. High performance in our setting is only possible with preliminary patch selection. We are using the Pearson correlation coefficient as it is common practice, e.g., in Bioinformatic to select discriminant

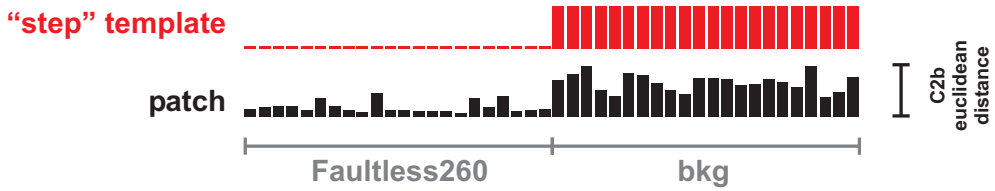


Figure 5.3: Correlation of a patch with a “step” template.

genes. Discriminance is defined with a background class. Selected features are most discriminant between object and background class.

Implementation Details

We first describe the parameter of Algorithm 5.1 and later make annotations to the procedure for more detailed understanding. The patch extraction procedure in Algorithm 5.2 passes the argument *patches* with $nPE = m \cdot n$. Selection of patches is based on the C2b euclidean response for training image and all extracted patches (see Algorithm 3.1). Additionally C2b is computed for the background image class (see Algorithm 3.2). The threshold n_{ps} controls the number of patches to select from the extracted patches and is a critical parameter for performance. The procedure outputs selected patches, that are most discriminent between the background and training image class. A threshold for the p-value pv assures that no non-discriminent patches will be selected. Patches are selected by the *Pearson's product moment correlation coefficient*. It can be defined as

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{5.1}$$

or in a more intuitive form as the scalar product of the standards scores of the two measures divided by the degrees of freedom:

$$r = \frac{\sum_{i=1}^{n} z_x z_y}{n - 1} \tag{5.2}$$

The standard score is the standardised version of the measures and calculates to

$$z = \frac{X - \mu}{\sigma}. \tag{5.3}$$

X denoting one random variable, μ the mean of X and σ the standard deviation of X. Note that two formulas for the standard deviation exists, that are used frequently. They

Algorithm 5.2 Select Discriminant C1 Patches

Input: $c2b[1..nImg][1..nPE]$: C2b response maps for training images
$c2bBkg[1..nBkgImg][1..nPE]$: C2b response maps for background images
$patches[1..nPE]$: extracted patches
n_{ps} : number of patches to select
pv : threshold for p-value

Output: $distinctivePatches$: selected patches

1: $n_{img} := \min(nImg, nBkgImg)$
2: $template := [zeros(n_{img}),\ ones(n_{img})]$
3: $c2b := c2b[\text{“}random\ permutation''][\ ]$
4: $c2bBkg := c2bBkg[\text{“}random\ permutation''][\ ]$
5: **for** $i := 1$ to nPE **do**
6: $\quad c2Patch := [\ c2b[1 : n_{img}][i], c2bBkg[1 : n_{img}][i]\ ]$
7: $\quad r[i] := corrcoef(c2Patch, template)$
8: **end for**
9: $distinctivePatches :=$ choose n_{ps} patches i with highest coefficient $r[i]$
10: remove all patches i with $pvalue(r[i]) < pv$ from $distinctivePatches$

only differ in the denominator, one using n and the other $n-1$. The latter version is used here which is

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}}. \tag{5.4}$$

It is common practice in bioinformatic to use class labels as template for pearson correlation selecting candidates out of several thousands of genes. As alternative sometimes also the spearman correlation coefficient is used, that does a preliminary ranking of the continuous values and is more robust to outlier and noise. On the other hand the spearman is not sensitive for selecting more discriminent patches on an absolute scale which is of central importance for fault detection. The template in line 2 consists of the class labels for the faultless and background class, with 0 denoting the faultless and 1 the background class (line 2 and 6, see Figure 5.3). The n_{ps} most discriminent patches are selected based on the highest positive coefficients (line 9). Our selection is not based on the p-value, but on the correlation coefficient itself. As long as we choose a fix number n_{ps} of patches and constant degrees of freedom for all correlations it makes no difference, because the computation of the p-value from r is strictly monotonic. A threshold for the p-value pv assures that no non-discriminent patches will be selected (line 10). However the procedure allows even for not very discriminante patches which

is reasonable as discussed in more detail in Section 5.2.1. Unbalanced data would yield coefficients that give more importance to the more abundant class, therefore we assure equally-sized classes in line 1 and 3. Line 4 assures a random subset from the bigger class. This procedure selects patches that have most significant "step"-character for the two classes and looks solely for patches that have higher C2b euclidean distance for the background class than for the faultless class.

5.2.3 One-Class Classification

One-class classification is done using the one-class SVM of Schölkopf et al. [68] with a linear kernel. In this way the features will not be transformed to a new space and the kernel is

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j. \tag{5.5}$$

We are using the freely available implementation of Chang and Lin [10] with the OSU MatLab Interface of Ma and Zhao.

The one-class SVM works on shape-based object-specific features that itself working as RBF function for object appearance. Their output is a tuned or in our case an un-tuned distance measure of similarity of appearance. The linear one-class SVM learns a smooth decision border around the training image responses. The parameter ν controls which fraction of training instances will be at most outside of the learned one-class model. For increasing ν, those image instances, which have feature response that differ strongest from all other image feature responses, will be first subject to moving out of the class. Some patches maybe will not be capable at all to cover some common appearance of the training images. Those patches will have a threshold very close to the origin. Other patches may be better capable to describe the variance of appearance. The existance of the latter kind of patches is crucial for a good detection performance. For a new instance the latter kind of patches will contribute most to classification accuracy. Patches with a threshold close to the origin will vote in the majority of cases for inside the class.

The choice of a linear kernel is according to Ockham's razor. We chose the simpler kernel because up to our knowledge no other more complex kernels would have had a strong justification. We tried also experimentally one-class SVMs with more powerful kernels as it is common practice. Experiments showed that, e.g., with a RBF kernel fault detection performance is worse. One reason for this is that indeed only one threshold for a S2b feature is needed and a RBF kernel draws a decision border that corresponce to two thresholds for a feature. The RBF kernel led to overfitting instead of increasing prediction accuracy.

However one-class SVM with a linear kernel disregards additional structure in the training data. Imagine several subgroups of the training images that have a representation in feature space with less variance. This could be for instance images of the object

Algorithm 5.3 Fault Detection Procedure

Input: img : image
$models[1..nRegion]$: one-class models for all regions
$p[1..nRegion][1..nP]$: selected patches for all regions

Output: $result \in \{\text{“}fault\text{”},\text{”}nofault\text{”}\}$: fault detection result
$where[1..nFault]$: fault region indexes

1: $result :=$ “$nofault$”
2: $where := [\,]$
3: $C2b :=$ compute C2b for img and p] {see Algorithmus 3.2}
4: **for** $r := 1$ to $nRegion$ **do**
5: $result2 := models[r](C2b[r])$
6: **if** $result2 =$ “$fault$” **then**
7: $result :=$ “$fault$”
8: $where := [where,\ r]$
9: **end if**
10: **end for**

from a similar viewpoint and similar lightning conditions. The classifier should draw a decision border for each subset separately and should classify a new instance as outlier if it does not lie in any one-class subset region. The linear kernel takes the maximum over all thresholds for a feature and is loosing specificity this way.

In contrast to other applications which want to find a ν for optimal performance, a manual threshold ν is favorable in our setting. ν indirectly controls false positive rate and helps to set up the fault detection system appropriately for different tasks such as off-line or on-line fault detection.

5.3 Fault Detection Procedure

After learning the one-class object model, fault detection can be performed following Algorithm 5.3. The result of fault detection is the combination of results for all regions. If one model of any regions reports a fault, this is already sufficient for the overall method to report a fault. However we track all faults, so afterwards also the fault regions for all faults are known.

The fault detection procedure outputs “fault” or “nonfault” for one image img. As input the learned one-class region model $models$ and the patches $patches$ are required. The output variable $where$ tracks in which regions faults were detected. For every region first the C2b responses (line 4) are extracted and afterwards the learned one-class model classifies the response pattern (line 5).

	Fault prior											
	0			0.1			0.2			0.3		
No. of	**False positiv rate**											
regions	0.1	0.2	0.3	0.1	0.2	0.3	0.1	0.2	0.3	0.1	0.2	0.3
1	0.10	0.20	0.30	0.09	0.18	0.27	0.08	0.16	0.24	0.07	0.14	0.21
2	0.19	0.36	0.51	0.17	0.33	0.47	0.15	0.29	0.42	0.14	0.26	0.38
3	0.27	0.49	0.66	0.27	0.45	0.61	0.22	0.41	0.56	0.20	0.36	0.51
4	0.34	0.59	0.76	0.31	0.55	0.72	0.28	0.50	0.67	0.25	0.45	0.61

Table 5.1: Overall false alarm rates for different number of regions, fault prior and false positive rates. Values for a false positive rate of zero are all zero. Values are calculated from Equation 5.11

This procedure is quite simple, but the influence of the *number of regions nRegion* to the *fault detection performance* is not obvious. In the following we examine this relationship in detail:

Number of regions and fault detection performance If we have fault models that have best fault detection performance for all regions, the number of regions will not make a difference to the overall performance. For a more realistic situation where the models are imprecise, parameter ν can be used to control the tradeoffs between sensitivity and precision for the models. For every model we denote the *false alarm rate* $p_{fa(r)}$ as the probability that the model will report a fault, when there is no fault. It can be computed from the prior $fprior_r$ for a fault in region r and the false positive rate fpr_r as

$$p_{fa(r)} = fpr_r \cdot (1 - fprior_r) \tag{5.6}$$

Normally the parameter ν_r will be chosen according to the false and true positiv rate in the ROC-curve of the training results, which makes this parameter needless in this setting.

It seems to be approximately correct to assume the false rate alarms of the regions to be independent from each other. If we know about one false alarm in one region that will in general not influence the probability of the other regions to claim false alarms. Then the probability p_{nofa} that the overall system claims no false alarm becomes

$$p_{nofa} = \prod_{r=1}^{nRegion} (1 - p_{fa(r)}) \tag{5.7}$$

and the overall false alarm rate p_{ofa} of the system becomes respectively

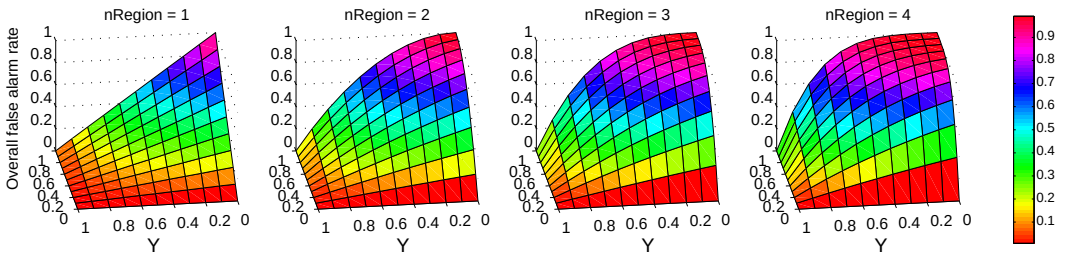


Figure 5.4: The false alarm rate for the overall system. Four different number of regions are illustrated. The y-axis denotes the *fault prior* and the x-axis the *false positiv rate*.

$$p_{ofa} = 1 - \prod_{r=1}^{nRegion} (1 - p_{fa(r)}). \tag{5.8}$$

Whereas the false alarms are independent from each other, the false alarm probabilities for the individual regions are not. The false positiv rates fpr_r are normally set in similar ranges depending on the false positiv rate accepted in a specific setting. If we want to set up the system for off-line fault detection, we are likely to accept high false positive rates to achieve high sensitivity. In the following we assume all fpr_r to be identical in all regions for a specific application. We also assume that the priors of a fault in all regions are identical, so that

$$(fpr = fpr_r) \;\; and \;\; (fprior = fprior_r), \qquad \forall r \in 1, ..., nRegion \tag{5.9}$$

These assumption make the false alarm rates p_{fa} for all models identical (see Equation 5.6). They furthermore simplify Equation 5.8 to

$$p_{ofa} = 1 - (1 - p_{fa})^{nRegion} \tag{5.10}$$

and allow us two express the overall false alarm rate p_{ofa} of the system by two parameters, the false alarm rate p_{fa} for a model and the number of regions $nRegion$. By substitution with Equation 5.6 and 5.9 this yields

$$p_{pfa} = 1 - (1 - fpr \cdot (1 - fprior))^{nRegion} \tag{5.11}$$

which is visualised in Figure 5.4 for four different values of $nRegion$. An Increasing number of regions in the fault detection system yield higher overall false alarm rates

for constant false alarm rates p_{fa} of the models. As a conclusion the number of regions should be as small as possible. In table 5.1 we show some overall fault detection results for reasonable ranges of $fprior$ and fpr.

For a real system with slightly different fault priors and false positive rates for all regions, the overall false alarm rate computes to

$$p_{pfa} = 1 - \prod_{r=1}^{nRegion} (1 - fpr_r \cdot (1 - fprior_r)) \tag{5.12}$$

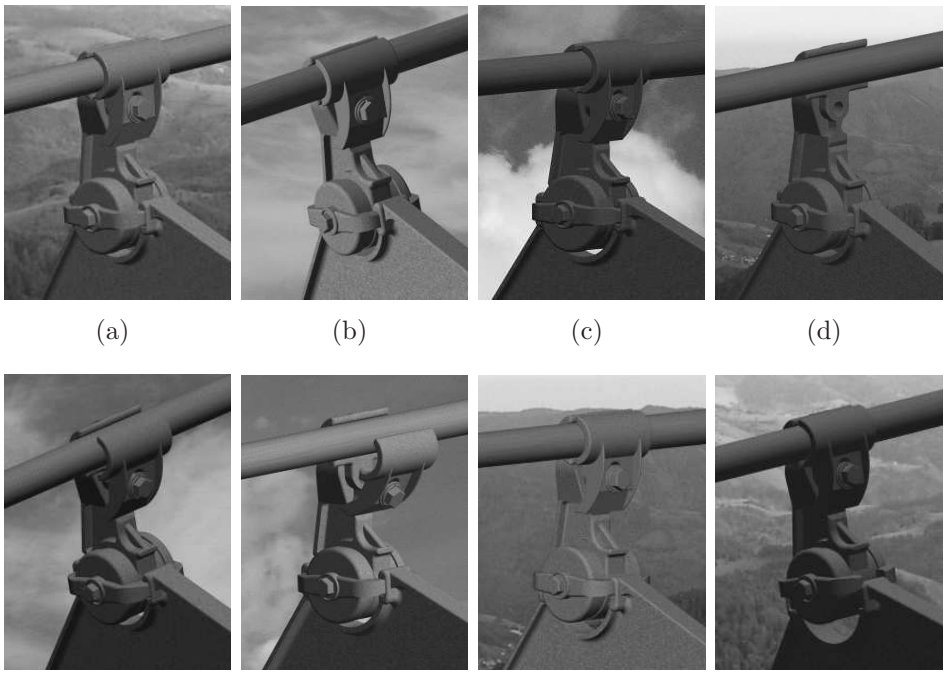
6 Experiments

Fault detection for 3D-objects under real-world conditions is quite unattended and we do not have access to data sets that would have been usefull for evaluation of our fault detection method. In Section 6.1 we present an intensily labeled synthetic data set. Images show one gripper of the power line spacer which is intended to be inspected in our UAV project. The entire data set covers one subset with a faultless gripper and three subsets with erroneous grippers. We modeled missing-claw, loose-claw and exhausted-spring faults. All images are taken around a central viewpoint limited to 12° pose change. This data sets enables us to make detailed experiments and present performance estimates for all faults. However, we also gain insight into more general properties, such as the *degree of fault* detectable with our method, or how parameters of our method should be chosen. In Section 6.2 we present five main experiments with our automatic method for fault detection. They were all based on one experimental setting (see Section 6.2.1). The first three experiment investigate the influence of the number of filtered patches, the number of extracted patches and the number of training images for the missing-claw fault. The remaining two experiments are made with the loose-claw and exhausted-spring fault data. These experiments show the influence of the degree of claw rotation and spring shift on fault detection performance. Finally, we present an experiment with three trained individuals on the same synthetic data as for our automatic procedure.

6.1 Synthetic Spacer Fault Data

In this Section we propose a synthetic data set which we created for evaluation puprose using AutoCAD. All images show a power line spacer (see Section 4.1) from one *specific view* that is subject to limited and equally distributed pose changes. Changes in pose from the central viewpoint are due to rotation in plane, view change and scale. Images vary in illumination and background and subsets of the data cover possible spacer faults (see Section 4.1.2). Each image has a record listing its complete configuration of pose, illumination, degree of fault, and background. The spacer was created as 3D model and images were rendered with different parameters.

The spacer consists of rigid object parts, but the power line is flexible. We do not model the flexible nature of the power lines. Our method is very likely not disturbed by potential curvate changes due to the local characteristic of the object patches and due to the fact, that our method makes no use of spatial relationships between the patches.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 6.1: Synthetic spacer image examples. All images are scaled to equal size for visualisation purpose. (a)-(c) Three faultless images with central viewpoint, viewpoint change of 11.6° and 9°. (d) A missing-claw fault. (e)-(f) Two loose-claw faults with 9.6° and 29.7° claw rotation. (g)-(h) Two exhausted-spring faults with a spring shift of 0.59 and 1.49 *cm*.

The 3D model has correct dimensions and proportions. There are same minor inaccuracies of the edges and forms in detail. We do not recommend to use synthetic training images for the purpose of detecting faults in real-world images.

Figure 6.1 shows three faultless images, one missing-claw fault, two loose-claw faults, and two exhausted-spring faults. All images are equally scaled for visualisation purpose. Note that the loose-claw and exhausted-spring fault embrace images that have a very little degree of fault and it is not easy to detect this fault for a human or this fault is sometimes not even considered a fault. We investigate how trained individuals classify these synthetic images in Section 6.3. More images are shown in Appendix B.

6.1.1 Faultless, Fault and Background Subsets

All images show one gripper of the spacer with same part of the spacer body. The data set embraces mainly four subsets of images. The Faultless subset contains only faultless representation of the spacer, the other three subsets contain three different faults. In the

MissingClaw subset the claw is missing. In contrast to a real fault appearance, there is no shift of the power line towards the gripper. The remaining two fault subsets describe faults which have a *degree of fault*. The LooseClaw subset models the opening of the claw towards the gripper and the ExhaustedSpring subset models a shift of the spring away from the spacer body. The latter covers actually two faults, the exhausted-spring fault as well as the released-spring fault. Real faults have a higher variations of fault appearance. For a description of the real faults see Section 4.1.2.

There are two additional subsets for the faultless set. The Faultless260 is identical to the Faultless set, but has a uniform object and image size. The Faultless segmentation set contains a binary segmentation mask for the spacer parts in every Faultless image. The Faultless260 and Faultless segmentation subset are for learning and the Faultless, MissingClaw, LooseClaw and ExhaustedSpring for testing. We implicitly cover the description of the Faultless260 set by describing the Faultless set. The Faultless260 and Faultless subset contain of twice the amount of images than the fault image sets. The background set is included in the description here, but contains real background images.

6.1.2 Parameters for Image Synthesis

Figure 6.1 summarises all parameters and values used in images synthesis. Pose and illumination for each single image is randomly created in an uniform manner. In the Faultless, Faultless260 and Faultless segmentation set each image of equal ID number has identical pose and illumination configuration. The background is not randomly selected, but rather in a deterministic cyclic fashion.

The pose of the central view is denoted in a coordinate system, where the power lines run parallel to the y-axis and the plane defined by the lower two power lines is parallel to the x-y plane. The z-axis points upwards to the third power line and the set of axis form a right hand coordinate system. The zero point of the system is at the center of the equilateral traingle formed by the power lines and is located in the middle of the spacer body. The polar description of the coordinate systems follows the terminology of sun positions. Azimuth is the angle along the horizon, with zero degrees corresponding to the x-axis, increasing in a clockwise fashion. The x-y plane denotes the horizon. Altitude is the angle up from the horizon. These two angles are sufficient to describe the position of the object. Viewpoint changes are limited by 12° to all directions. The rotation in plane varies up to 6° around the central view-point. Consequently the maximum viewpoint difference between two images in the data set is 24° for the viewpoint and 12° for rotation.

All images have a minimum size of 200×260 pixels. Images inside the Faultless260 of the background class are all equally-sized. All other image sizes are scaled with a random factor between 1 and 1.53. The view scale (see definition in Section 4.3) is constant for all training images in the Faultless260 subset. Depending on the scale s and for all other subsets, the view scale v calculates to $v \cdot s$.

Illumination is modeled with a distant light from different directions. Distant light is

Parameters	Values for data sets					
	Fault-less260	**Fault-less**	**Missing-Claw**	**Loose-Claw**	**Exhausted Spring**	**Bkg**
Number of images	1000		500			91
Central viewpoint	135° azimuth, 15.6° altitude					-
Viewpoint change	0° to 12° (to all directions)					-
Rotation	-6° to 6°					-
Min. image size	200×260					400×400
View scale	$1 \cdot px/mm$					-
Scale s	1	1 to 1.53				1
Distant light	40% to 100%					-
Azimuth	90° to 180°					-
Altitude	30° to 80°					-
Ambient Light	0.2 to 0.3					-
Missing claw	No	No	Yes	No	No	-
Claw rotation	0°	0°	-	0° to 30°	0°	-
Spring shift	0 cm	0 cm	0 cm	0 cm	0 to 1.5 cm	-
Background	91 alternating background images					-

Table 6.1: Parameter values for all synthetic data sets and the background class. For all parameters with value intervals, values are drawn randomly and uniformly from within these interval.

light with parallel rays and is comparable to sunlight. Light directions are limited to those that will lighten the object side we are looking at. The ray directions are given in azimuth and altitude measures analog to the central view. Ambient light linearly influences the intensitiy of the image.

Before rendering, the claw was removed for the MissingClaw subset, the claw was randomly rotated between 0° and 30° for the LooseClaw subset, and the spring was shifted randomly by 0 to 1.5 cm. Afterwards images were rendered indentically to the Faultless subset.

Our lightning model yields quite good and realistic looking images. However, there are some strong limitations compared to realistic images. We did not model shadows and object appearance is quite sharp through strictly parallel light. Shadows for one distant light were very sharp and strong and seemed not to be realistic neither. The attempt to include several distant lights of different intensity and equally-spaced directions was not successfully due to a runtime explosion. Real spacer images show less sharp object shapes and evaluation performance with this data is slightly biased towards higher performance.

6.2 Automatic Fault Detection on a Limited View

6.2.1 Experimental Setup

In this section we describe the experimental setup which was used for experiments. Computing the spacer model requires first learning and computing features for training images (see Algorithm 3.1), computing features for test images (see Algorithm 3.2) and learning the object model with the one-class classifier 5.2.3. Learning the classifier is fast, but feature extraction is time-consuming. Consequently we used only a subset of all images for evaluation, precalculated features layerwise, and stored intermediate results to files.

For all images, S1 and C1 response maps can be computed before learning anything (compare Algorithm 3.1 and 3.2). Afterwards patches were extracted from the Faultless260 data set for two regions and bands 1, 2 and 3. S2 and C2b layers are computed for all patches and images and response maps are stored together with their corresponding patches. A more detailed description of the regions and patch extraction is given below.

Depending on the experiment in mind, different subsets of the features were used for evaluation, one-class models were learned and experiments were made by varying some desired parameters. The ratio of training to test data size were chosen different as usually and ROC-curves together with the ROC area are used for measuring performance. See the end of this section for more on the last two points.

Data subsets for evaluation The synthetic spacer fault data (see Section 6.1) was used for all experiments. See Table 6.1 for detailed information about images. Features we extracted for only a subset of the available data (see Table 6.3). Faultless260 images were used for training and all other subset for testing. Note that the Faultless260 and Faultless data sets only differ in their image size (see Section 6.1) and therefore the image subset for testing has to be distinct from the Faultless260 subset.

Region labels We defined two regions R_1 and R_2, both are of rectangle shape, and their positions and sizes are given in Table 6.2. Position and sizes are in pixel and the region position is represented by the upper left corner (x,y) of the recangular. Figure 6.2 illustrates the regions for a faultless image from the central view (see Table 6.1). Region R_1 is used to detect the missing-claw Fault and the loose-claw Fault. Region R_2 is used to detect the exhausted-spring Fault. Only one rectangle is used as region label for all training images of size 200×260 *pixels*. This is an approximated and fast labeling and is sufficient for fault detection on a limited view with aligned training images.

Patch extraction for evaluation For every region and scale patches are extracted and stored separately. A slightly different and constrained version is used for patch extraction. Compared to Algorithm 5.1 in line 6 the number of the currently extracted

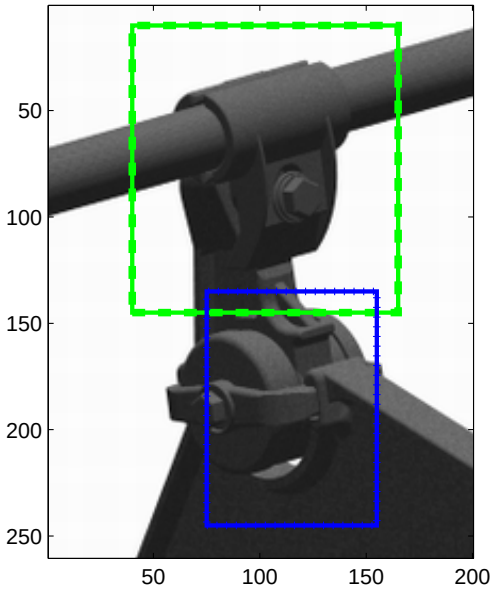


Figure 6.2: Two regions used for experiments. R_1 is shown as green dashed and R_2 as blue dotted rectangle.

Region	x	y	*width*	*height*
R_1	40	10	125	135
R_2	75	135	80	110

Table 6.2: Position and size of regions used in experiments. The region position is represented by its upper left corner (x,y).

Data set	**Subset images**
Faultless260	1 to 200
Faultless	501 to 700
MissingClaw	1 to 200
LooseClaw	1 to 200
ExhaustedSpring	1 to 200

Table 6.3: Subset of images used for experiments. Only 200 of all 500-1000 images per synthetic data set were used due to runtime limitation.

patch constraints the training image selection. The first patches are extracted randomly from images 1 to 50, the second from 51 to 100, and so one. The four different subsets of patches are used later in different runs to determine the variance of the procedure. Due to the random order of the training images there is nearly no difference to the original procedure. Depending on the band a different number of patches is extracted. Lines 10 to 15 in Algorithm 5.1 show that for any randomly drawn image one patch for each patch size $sizes[k]$ is extracted. Band 1 has four different patch sizes and bands 2 and 3 have three (see Section 3.1.3). We choose 50 random images from each image subset, yielding 200 patches per subset for band 1 and 150 patches for bands 2 and 3. In all 800 patches were extracted for band 1 and 600 for band 2 and 3. Over all experiments the same patches sizes are used (see Section 3.1.3).

Ratio of training to test data size For evaluation usually 66% of images are used for learning and the remaining 33% for testing. In our case for an one-class model this should be handeled differently. We use 33% of the faultless training images for learning, 33% for the faultless test images and 33% for the test images containing faults. One major reason was that the fautless class should be reasonable large to compute statistically reliable results in some experiments. A second reason was that we not even need as much training images as we were using. This was indicated by some preliminary runs. We assume balanced classes for testing only for evaluation purpose. This will not make a difference to the detection performance measure (see [14] and next paragraph).

Fault detection performance measure In all experiments we use the ROC area as performance measure which is the area under a ROC-curve. ROC-curves provide two nice properties that are advantageous for evaluation of our fault detection method. They are entirely insensitiv to changes of the class distributions which are not known for the faults evaluated. Second when learning an one-class model, the parameter ν has to be chosen and its choice is critical for fault detection behaviour. Beforehand any choice would have been quite ungrounded. Usually ROC-curves are generated by thresholding continuous outcome variables. Sometimes a modification has to be made to methods to provide some kind of continuous output. In our case we are using different values of ν for calculating ROC-curves. The ν parameter is defined as the expected percentage of false outliers for its respective one-class model. A ν parameter of zero corresponds to an expected false positiv rate of zero and a ν parameter of one to an expected false positive rate of one. Compared to thresholding we can control the false positiv rate more directly, but not exactly. We use 50 equally spaced values of ν to compute a ROC-curve. This value was determined experimently and showed good results. Fawcett [14] provides a valuable and practical introduction to ROC-curves.

	no. of filtered patches							
Band	1	3	5	8	11	14	24	34
1	0.9896	0.9785	0.9664	0.9822	0.9850	0.9850	0.9798	0.9793
2	0.7800	0.9414	0.9695	0.8660	0.9191	0.9376	0.9712	0.9663
3	0.8082	0.6673	0.7432	0.7732	0.8346	0.6846	0.6230	0.5967

Table 6.4: ROC areas for sub-experiment 1(b).

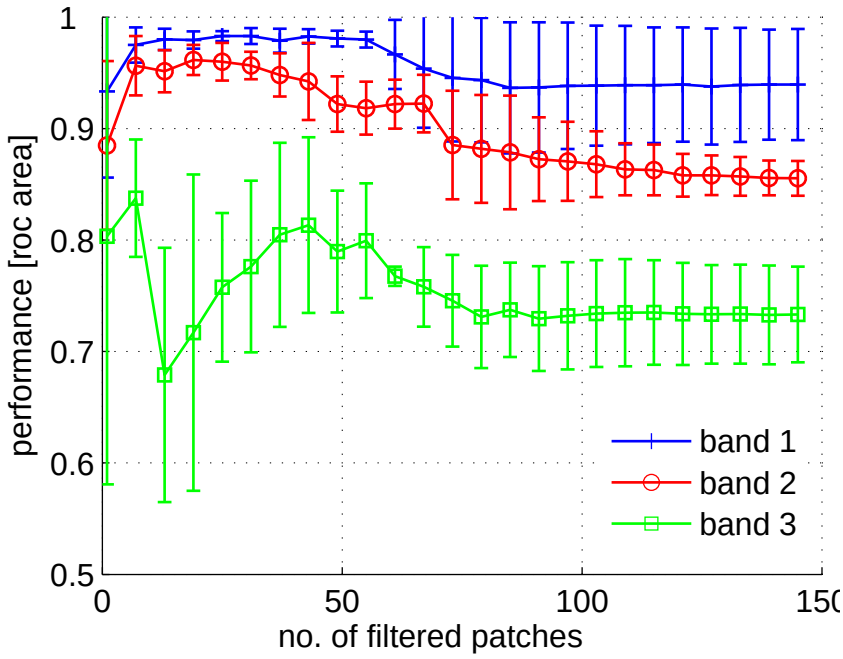
6.2.2 Fault Detection Performance

This section presents results of five experiments with the synthetic data set and are based on the experimental setup of the last section. Experiments one, two and three evaluate the influence of the number of filtered patches, the number of training images, and the number of extracted patches using the FaultLess260 images for learning and the FaultLess and MissingClaw images for testing. Experiments four and five illustrate performance dependent on the degree of claw rotation and the degree of spring shift, and use the LooseClaw and ExhaustedSpring data instead of the MissingClaw for testing.

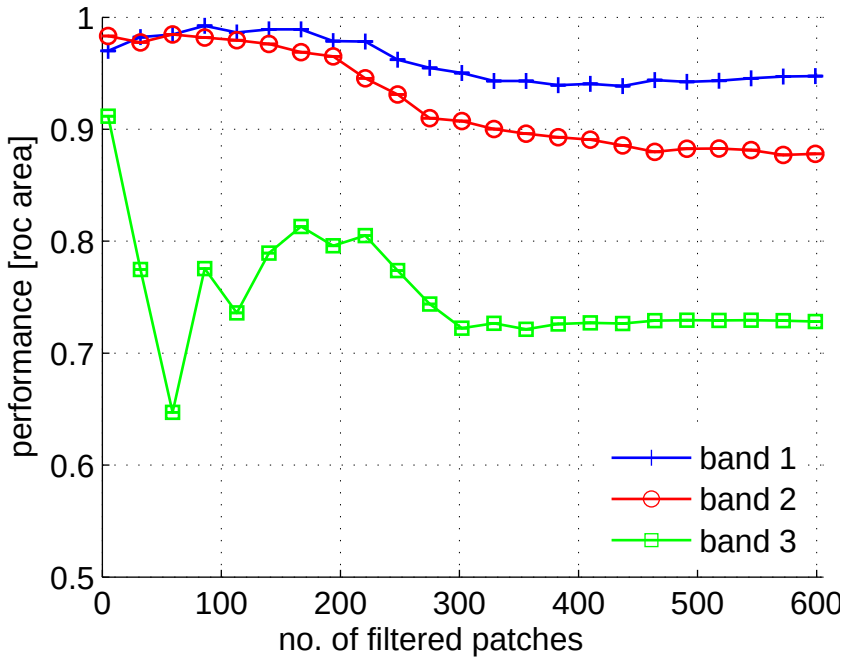
Experiment 1: Variable number of filtered patches This experiments investigates how the number of filtered patches used for fault detection influences detection performance. We performed two slightly different sub-experiments. The sub-experiment 1(a) (see Figure 6.3) computes 95% confidence intervals over three independent runs with a reduced number of extracted patches per run. The sub-experiment 1(b) uses all extracted patches. Two sub-experiments were made to present both the variance and the highest performance of the method. Table 6.4 reports the ROC areas and Figure 6.6 depicts-ROC curves for sub-experiment 1(b).

The number of filtered patches were varied over the sub-experiments and the sub-experiments differ in the number of extracted patches used. The number of images for learning and testing remained constant. The one-class classifier was learned and tested with C2b responses from subsets as in Table 6.3. For learning 200 of the Faultless260 images were used. The method was tested with 200 of the Faultless and 200 of the MissingClaw images. We evaluated bands 1, 2 and 3 with patch sizes according to Table 3.1.

Sub-experiment 1(a varies the number of filtered patches starting from 1 in steps of 6 up to 145. Each run in band 2 and 3 relies on 150 extracted patches from 50 images that are distinct from the images of the other runs (see Section 6.2.1). For band 1 each run relied also on 150 extracted patches but only from 38 images. In sub-experiment 1(b) for bands 2 and 3 the number of extracted patches was 600 and for band 1 it was 600. The number of filtered patches in Figure 6.3 (b) run from 5 to 599 in steps of 27. Table 6.4 has a smaller step size of 9. Figure 6.6 shows the first three columns of Table 6.4 in more detail as ROC curves.

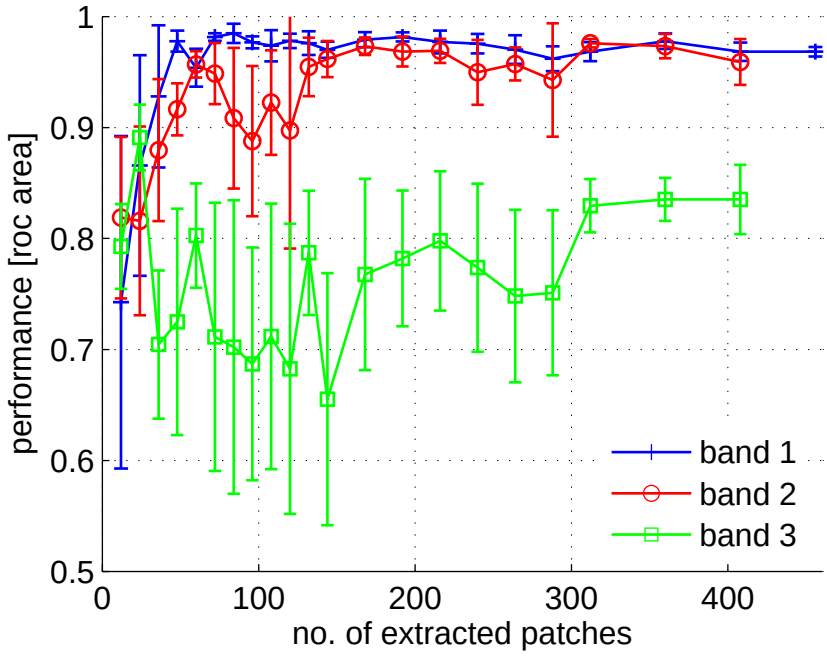


(a)

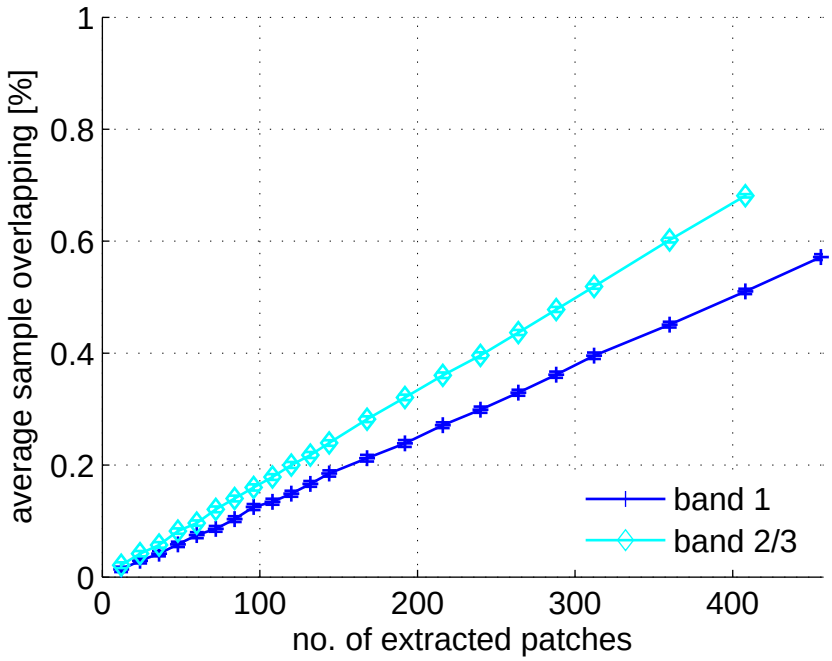


(b)

Figure 6.3: Experiment 1: Performance for a variable number of filtered patches. (a) Error bars are 95% confidence intervals from four independent runs with 150 extracted patches each. (b) Performance for one run with 600 extracted patches.



(a)



(b)

Figure 6.4: Experiment 2: Performance for a variable number of extracted patches. (a) Error bars are 95% confidence intervals from six runs with randomly chosen patches without replacement. (b) The expected overlapping of two randomly chosen patch subsets.

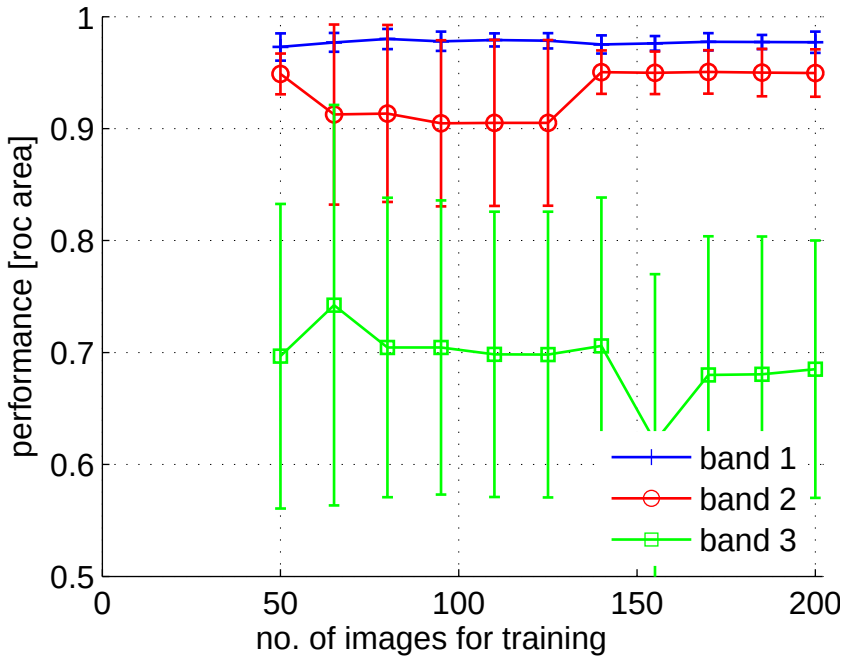


Figure 6.5: Experiment 3: Performance for a variable number of training images. Error bars are 95% confidence intervals from four runs.

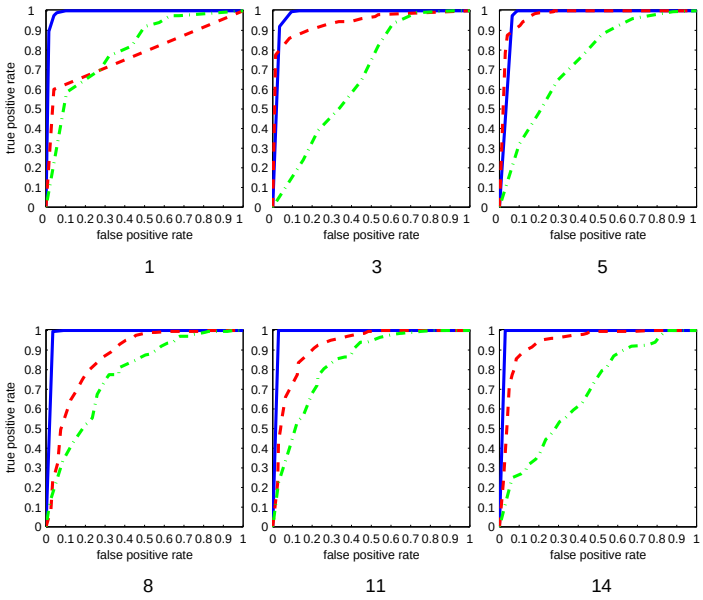


Figure 6.6: ROC curves of sub-experiment 1(b) for 1, 2, 3, 5, 14 and 23 filtered patches.

We show absolute values of the ROC-areas for sub-experiment 1(b) in Table 6.4. Choosing a threshold for the number of filtered patches between 1 to 41, choosing the highest number for extracted patches and the highest number for training and testing images, and based on sub-experiment 1(b) we can report a ROC area higher than 0.9673 for band 1. And we could even choose only one or two patches to achieve high performance. Figure 6.3 (a) shows for one selected patch that all bands have high variance. Interestingly for all bands exist at least some patches that are selective enough to achieve high performance, but in contrast to band 1, band 2 is less tolerant for many non-informative patches, and band 3 is only very little. For less patches we have a higher risk that we will not find a good one, but with more patches our performance suffers from many noise patches.

The missing-claw fault is a fault that does not have a degree of fault. Despite the fact that this fault could have learned with a two-class classifier our results suggest that in this case an one-class classifier is probably favorable.

Experiment 2: Variable number of extracted patches In this experiment the number of extracted patches is used as variable. Figure 6.4 (a) illustrates performance. Error bars are baised to be small for an increasing number of extracted patches.

All 200 images are used for training the one-class classifier. The number of filtered patches was chosen fix to a value of 10. Extracted patches are randomly chosen from all 600 available patches for band 2 and 3, or for all 800 patches for band 1. Curves start at 12 extracted patches and values are computed in steps of 12, 24 and of 48 patches. For band 1, 200 patches more exist (see Section 6.2.1). Six runs for every band were performed, but due to limited number of patches available for a higher number of patches the runs are not any more independent from each other. Random selection was done without replacement. Figure 6.4 illustrates the expected overlapping between two subsets of patches of two different runs.

Patch learning relies on randomly selecting patches from regions. Filtering of discriminant patches does not help as long as the random procedure draws a patch from a good position for fault detection. High variance for lower number of patches can be explained by randomly choosing good or bad patches for fault detection. For a higher number of extracted patches, the high performance suggests that our filtering approach is successfully filtering those patches that are usefull. For band 1 already for 100 patches high performance with little variance is shown. Mention that the size of the region will influence the number of extracted patches required. Larger regions will need more patches to achieve the same results.

Experiment 3: Variable number of images for training This experiments itends to understand the relation between performance and number of images provided for training (see Figure 6.5). The number of training images was varied from 50 to 200 in steps of 15 images. Four runs were made to compute 95% confidence intervals. For every

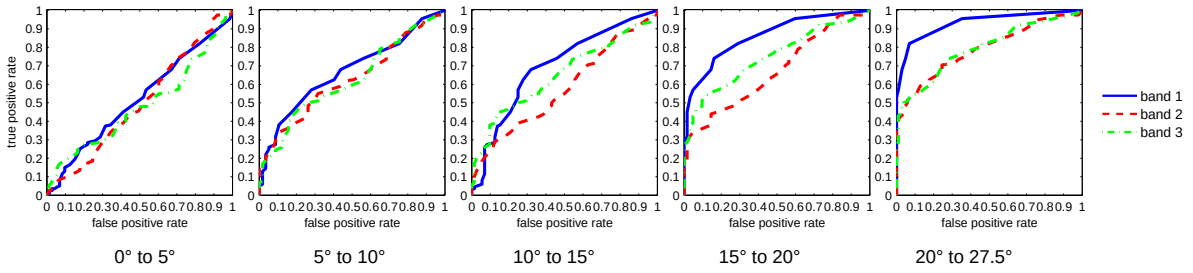
run patches were extracted from one of the 50 image subsets (see Section 6.2.1). All bands used the same number of $3 \cdot 50 = 150$ extracted patches. The number of filtered images was set fix to 10. The number of training images does not significantly influence performance in the range from 50 to 200 images. It suggests that very little training images are required for training at all. This is in agreement with results from Serre et al. [72]. Even smaller training sets seem to be plausible for learning.

Experiment 4: Variable degree of claw rotation This experiments explores which degree of fault our fault detection method is capable to detect. Two sub-experiments were performed to show cumulative performance results for degree of faults greater than a claw rotation value (see Figure 6.8 (a)) and a result for a degree of faults within a specific interval (see Figure 6.8 (b)). Figure 6.7 further illustrates results from sub-experiment 4(b) with ROC curves. For training 600 extracted patches and 200 images were used for all bands. For testing all 200 Faultless responses are considered. The number of filtered patches were chosen to a value of 10.

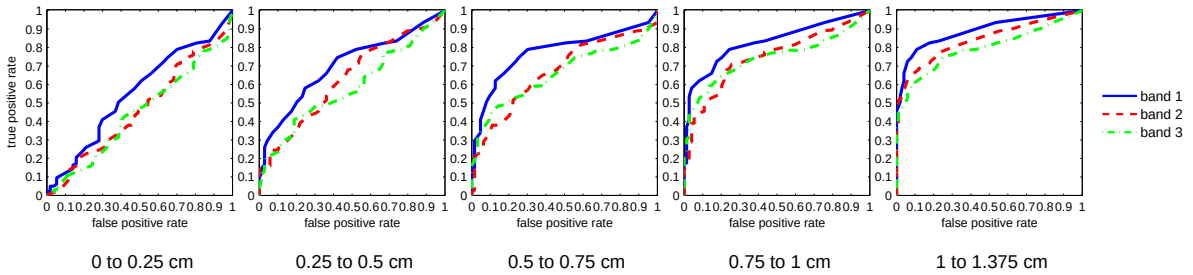
Sub-experiment 4(a) constrains the available fault testing data to only those images that are above a specific claw rotation. It starts with all fault images for claw rotation greater than 0° and decreases the amount of training data constantly by increaseing the constraints in steps of 2.5°. The last performance measure for a claw rotation greater than 27.5° only considers about 17 fault images for testing. Sub-experiment 4(b) parts all 200 available LoowClawFault data into 12 nearly equally sized test sets. There size is about 17 images each. Every of those small test set embrace faults with a degree d with $x \leq d < (x + 2.5)$. Figure 6.8 (b) illustrates this sets as steps of size 2.5°.

Sub-experiment 4(b) gives insight into the sensitivity of our method for images with some small degree of fault. Band 1 is able to detect faults with a good performance for a claw rotation greater than about 15°. Band 2 and 3 have problems to detect even larger faults and perform badly. The gap in performance between bands is considerable and if we imagine a higher view scale than band 1 it will likely detect even arbitrary smaller faults. Mention that the testing and training images are around the central viewpoint (see Table 6.1) of about 45° towards the claw rotation plane and therefore rotation can not be considered absolute. The relation between rotation and performance is further distorted for other objects through possible different length of rotating parts.

Experiment 5: Variable degree of spring shift This experiment is analog to Experiment 4 but with the ExhaustedSpring fault data as fault test set. Figure 6.9 shows both sub-experiments 5(a) and (b) and Figure 6.7 visualises some steps of sub-experiment 5(b) in more detail as ROC curves. We only state differences to Experiment 4. The continuous degree of fault in this fault data set is a shift of the gripper spring. It is varied from 0 to 1.5 cm in steps of 0.125 cm. Each step also corresponds to about 17 images. For band 1 or for a view scale of 1 px/mm an object part has to shift about 1 cm to be detectable as fault. The performance is a ROC-area of approx. 0.85 for this



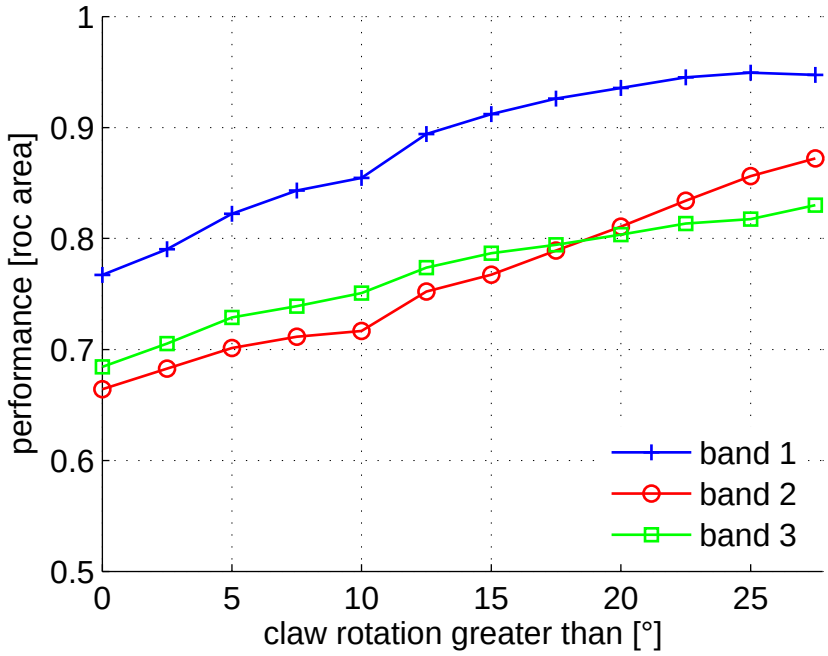
(a) Sub-experiment 4(b)



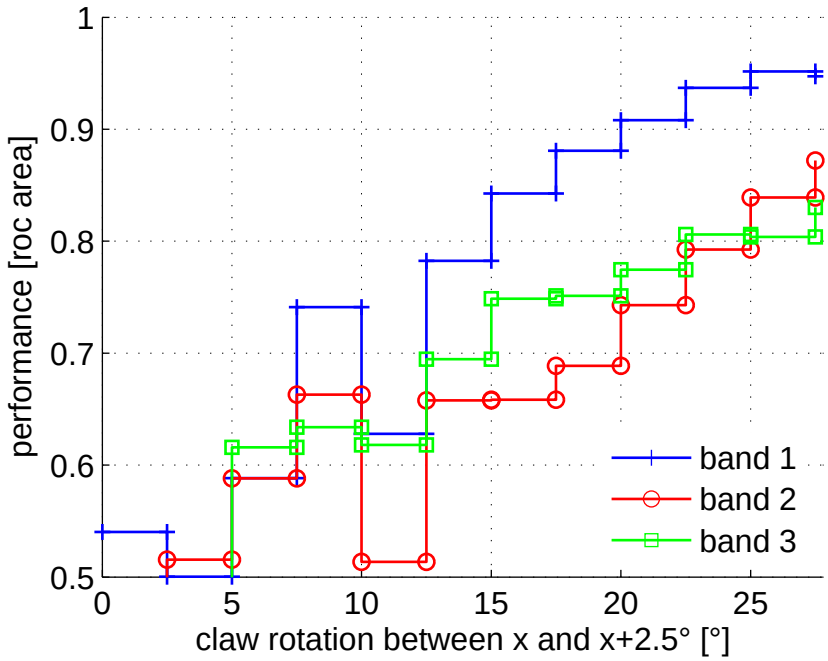
(b) Sub-experiment 5(b)

Figure 6.7: ROC curves of sub-experiment 4(b) and 5(b) for 5 subsets with different degree of claw rotation and different degree of gripper shift, respectively.

task. Also here a higher view scale can improve performance.

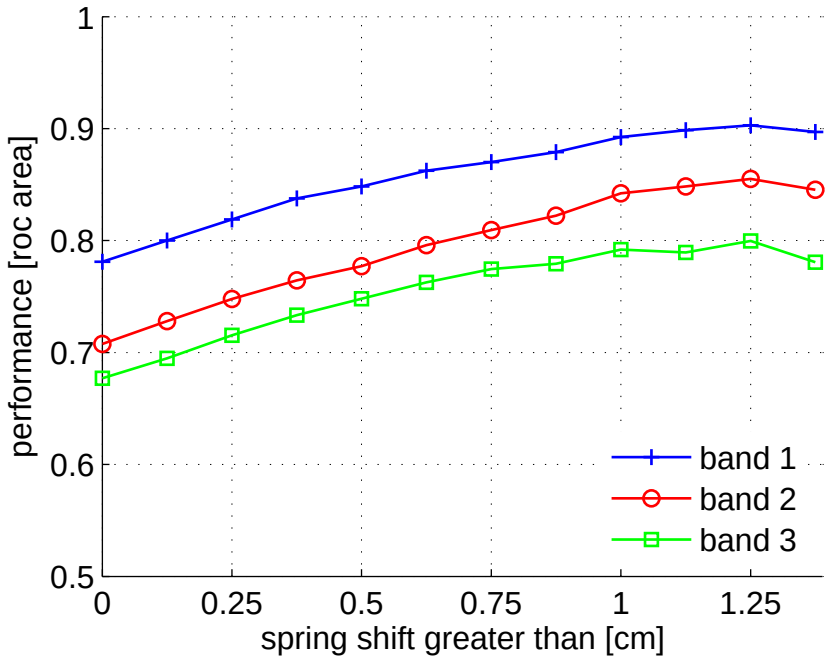


(a)

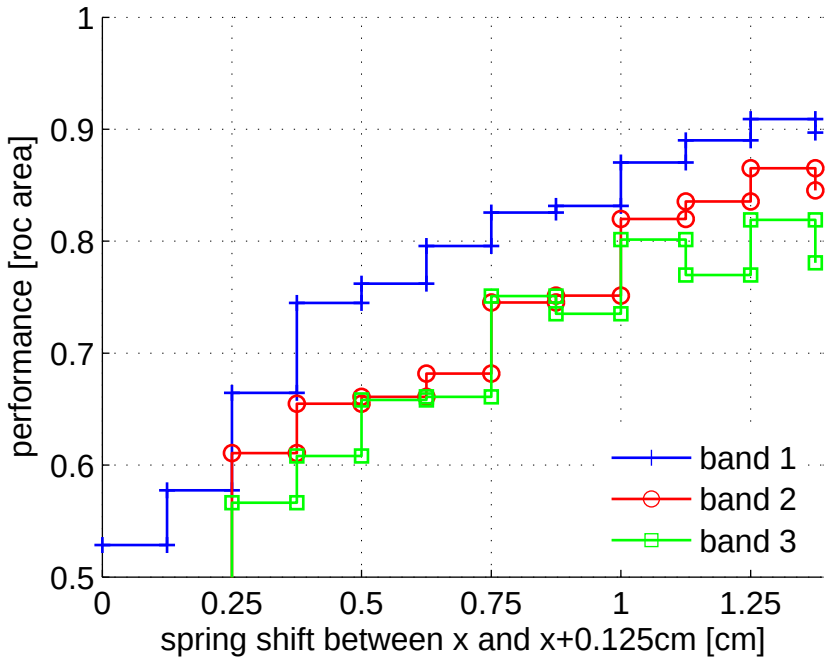


(b)

Figure 6.8: Experiment 4: Performance for a variable degree of claw rotation. (a) Cumulative performance for a degree of fault greater than a specific value. (b) Performance for different degrees of fault in steps of 2.5°.



(a)



(b)

Figure 6.9: Experiment 5: Performance for a variable degree of spring shift. (a) Cumulative performance for a degree of fault greater than a specific value. (b) Performance for different degrees of fault in steps of 0.125 *cm*.

6.3 Fault Detection by Trained Individuals

This section presents and discusses an experiment made with three individuals that were trained for fault detection in aerial power lines. Fault detection performed by one individual for many spacers is time-consuming, needs elevated concentration and is cost expensive. How long would one trained individual need to do fault detection and how accurate would it be? The following tries to give first answers to this question and is also intended to explore the degree of preciseness of fault detection achieved by a trained individual. Subjects will be confronted with a random sequence of fault and non-fault images and they have to perform classification with only little attention permitted.

It follows the description of the method, observed results and a short discussion.

6.3.1 Method

We used a survey form which covered 32 images of fault and non-fault grippers. They were presented individuals in a size of of approx. 4.6×5.3 cm per image. Images were taken from the synthetic data sets LooseClaw and ExhaustedSpring, covering the two faults, which have a degree of fault. The MissingClaw set was considered to be too simple to make a classification experiment reasonable. Subjects were instructed to remain at most about 3 sec at one image and furthermore to mark the fault or non-fault check box of the image within this time. This is an overall time needed fault detection of about 96 sec. Note that experiments with individuals could not be supervised to garantee that the procedure is performed in compliance with our rules.

The first 150 ms of visual perception are known to be used for rapid object recognition in near absence of attention. Humans are able to detect an animal in a natural scene within this time (see Section 3.1.1). FeiFei [15] used intervalls between 27 ms and 500 ms to investigate the influence of attention to categorisation performance. Three seconds are a relatively long time and allow for relatively high attention.

Subjects were trained in the sense that they had a good understanding of the faults in power lines and for the spacer in this work (see Section 4). They have well-founded experience of detecting faults from images, but they were neither familiar with the synthetic data set nor did they have knowledge about the specific modeling of faults.

All 32 gripper images were presented in a random manner and the number of fault images were chosen non-intuitively. A threshold for dividing each fault set into “easy” and “difficult” fault were determined, by using the performance results of our automatic method (see Figure 6.9(b) and 6.8(b)). We considered a ROC-area of 0.8 as a good performance threshold, which corresponds to a threshold of approx. 13.5° for the loose claw data set and approx. $0.8cm$ for the exhausted spring data set. Table 6.5 shows the main distribution of the images with respect to their class and degree of fault. Inside the thresholded fault classes the images were selected to have an approximate uniform distribution of the claw opening angle and the exhausted spring shift.

Image set	Fraction of all images	Abs. no. of images
Faultless	25.0%	8
LooseClaw, $\alpha \leq 13.5°$	28.1%	9
LooseClaw, $\alpha > 13.5°$	9.4%	3
ExhaustedSpring, $s \leq 0.8cm$	28.1%	9
ExhaustedSpring, $s > 0.8cm$	9.4%	3
All images	100%	32

Table 6.5: Class distribution of images presented to trained individuals for fault detection. α denotes the claw rotation of the loose-claw fault, s the spring shift for the exhausted-spring fault (see Table 6.1).

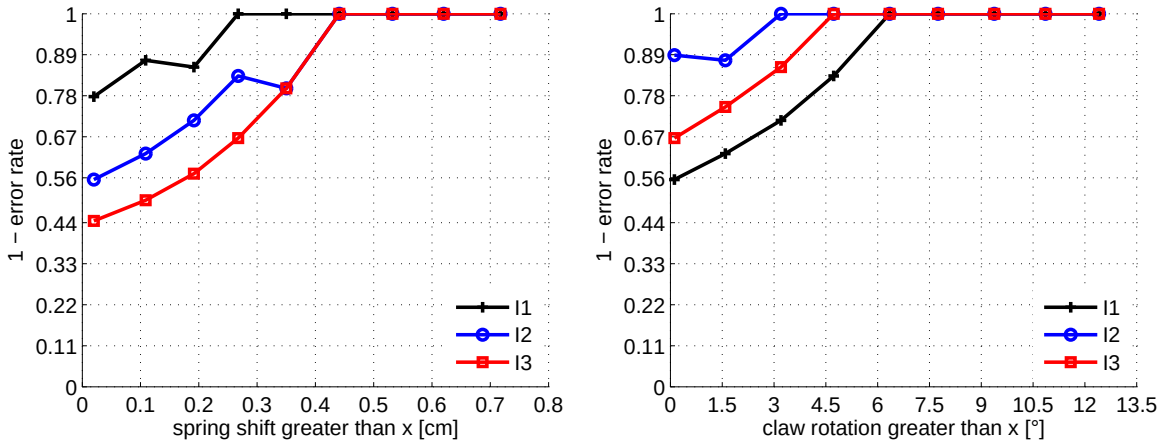
Image set	Error rate of individual I_1	I_2	I_3	Avg. error rate
Faultless	0.13	0.00	0.00	0.04
LooseClaw, $\alpha \leq 13.5°$	0.44	0.11	0.33	0.29
LooseClaw, $\alpha > 13.5°$	0.00	0.00	0.00	0.00
ExhaustedSpring, $s \leq 0.8cm$	0.22	0.44	0.56	0.41
LooseClaw, $\alpha > 13.5°$	0.00	0.00	0.00	0.00

Table 6.6: Fault detection performance of three individuals on data sets as summarised in Table 6.5.

6.3.2 Results and Discussion

Survey results indicate that individuals did not adapt to the distribution of faults or the simplified modeling of the spacer faults. They could not use this simplified setting to their advantage in fault detection.

Table 6.6 summarise the results of the experiments made with individuals $I1$, $I2$ and $I3$. Only error rates on all image set separately are shown. An overall error rate would be highly misleading due to the artificial image set distribution which was designed for testing purpose only and is not realistic for faults. As expected, all individuals classified the faultless set as well as the easier fault image sets, LooseClaw $\alpha > 13.5°$ and LooseClaw $\alpha > 13.5°$, nearly perfectly. Both more difficult fault sets had high error rates of about 29% and 41%. Figure 6.10 allows for a more detailed understanding of the classification errors made. Each curve shows the accuracy for a subset of the fault image set. Each subset is restricted to images with a degree of fault greater than a specific x. Note that every step corresponce exactely to one test image and the curves can only give an rough impression of the accuracy. Note further that the accuracy is



(a) (b)

Figure 6.10: Accuracy of three individuals on two fault image sets for a degree of fault greater than x. (a) On the LooseClaw, $\alpha \leq 13.5°$ image set. (b) On the ExhaustedSpring, $s \leq 0.8cm$ image set.

measured only on each "pure" fault image set. However accuracies would not change due to a false negative rate of approx. zero for trained individuals.

We observe that a spring shift of less then approx. 0.35 cm and a claw rotation of less than approx. 4.5° are difficult to detect for trained individuals or are not considered as fault. More precise statements seem not to be reasonable from the limited amount of data we considered in this experiment. But we can see another effect that occured strongly with individual $I1$ and $I2$. $I1$ focused more on the spring part of the spacer, performed relatively good for the exhausted spring fault, and relatively bad for the loose claw fault. The same observation can be made for Individual $I2$. In contrast to $I1$, $I2$ focused more on the upper part of the spacer. Time for inspection was not sufficient to pay attention to several regions of the spacer. In this sense worse performance on the exhausted-spring fault for $I1$ and on the loose-claw fault for $I2$ are rough performance measure for fault detection without attentation. If individuals know where to focus in the image, they can attend to the region and yield better classification performance.

7 Conclusion

In this work we presented a general method for fault detection from video images for 3D objects under real-world conditions and show how it can be used und integrated with a framework for fault detection in aerial power lines by an Unmanned Aerial Vehicle (UAV). The fault detection method is restricted to a limited view, is designed for usage after object recognition and is robust to strong illumination changes, clutter, translation, scale, about 24° of view change and 12° of rotation in plane. We used a state-of-the-art biologically-inspired framework [72] to learn object-specific features and selected discriminant features based on a background data set. A heuristic of the trade-off between object-covering and discriminant features is proposed that requires regions to be defined for feature extraction. Additionally segmentation masks for training assure feature extraction within the object. In contrast to most other prominant work in Object Recognition we used an one-class SVM for learning.

Our approach is widely applicable for other tasks and all kind of faults for instance changed object part constellations, broken object parts or in general object appearances not covered by the one-class object model. The features are suitable for textured as well as low-textured free-form objects. Our results demonstrate that our method is indeed capable of detecting unseen faults. In this way for one region our method scales *constant* with the number of different faults. Experiments with the missing-claw fault had a ROC area of 0.98 which suggests that an one-class SVM performs at least comparable to a two-class classifier in this task. Note the similarity to Kowalczyk and Raskutti's work [39] on yeast regulation prediction data which also has highly discriminant features. We report a good, but significant less performance for two other fault data sets which have a variable degree of fault. Our explanation of this result is that the regions of interest are highly non-planar, the features only achieve little invariance to view-change and no discriminant features for this regions exist.

Patch extraction with segmentation masks worked properly. The heuristic of using region masks to assure some degree of object-covering proofed to be applicable and yielded good results. However object-covering is not garanteed inside the regions and regions have to be defined beforehand which is an incorporation of fault-specific knowledge to some degree. The number of regions should as small as possible to avoid loss of performance. This work primarily focused on developing a method for maximal fault detection performance. The implementation of our fault detection method was done in MatLab and time needed for learning and testing was high. We consider fault detection as a task that will take place after preliminary object recognition and detection performance relies strongly on accurate pose estimation within tolerance bounds. Our notion of fault

detection uses discriminant patches for fault detection. Experiments showed that several discriminant patches exist all over the target object, but in general discriminance is a limitation and should be overcome by incorporating spatial information.

The synthetic data set has a realistic appearance. Its limitations are mainly the lack of strong shadows and a relatively sharp object appearance. Each image is labeled intensely with object pose, background image, illumination configuration, scale and the degree of fault. The large size allows for detailed and reliable evaluation.

The outlier threshold ν of the one-class SVM indirectly controls the true positive rate and makes our method suitable for off-line as well as on-line fault detection in aerial power lines.

7.1 Future Work

There are several points that are worth to be followed up by future work.

S4 (AIT) view-tuned units of Serre et al. [71], which were proposed in a later stage of this thesis, could be used to overcome two major restriction of our approach. They are one layer above S2b and imprint patch responses for specific training images. S4 units output a clearer response than S2b units, they could help with non-planar object regions and are the natural extension of our fault detection method to several viewpoints of the object. In this way, the selection of discriminant features will have to be on the level of S4 units. The biologically-inspired features of Serre et al. [72, 71] are subject to future and rapid chance and a new method for fault detection should take advantage of it.

The implementation of our fault detection was done in MatLab, it was designed for evaluation purpose and should be reimplemented. A runtime-optimized version for instance in C/C++ is a candidate for real-time fault detection. Runtime can be further improved by a suitable choice for the number of scale bands and the number of filtered patches. In Section 4.2 we suggested how cropping images could be done using our architecture for fault detection.

The need for a tradeoff between object-covering and discriminant features was approached in a heuristic manner. Future work should focus on how to control the relationship between object-covering and discriminance more accurately. Optimally we are thinking of a threshold that guarantees some degree of object-covering while selecting most discriminant features automatically. Additionally the degree of multiple overlapping has to be incorporated to allow for a variety of object appearances. A selection of patches that span the space of patch appearance uniformly could be a solution that circumvents the problem of pose estimation.

We consider fault detection as a task that will take place after preliminary object recognition. This work only deals with fault detection and a further investigation is necessary to understand how object recognition and fault detection could be done efficiently if both are using the biologically-inspired features of Serre et al.. Probably this task is the contrary to our notion of fault detection and a central question could be “How many

features do we have to see at least to consider something a target object?". This task is slightly different to current object recognition in the sense that we do have to allow for changes due to faults which we did not see before and are not covered by the training images.

A Technical Specifications

A.1 Unmanned Aerial Vehicle (UAV)

Series: Miniature Aircraft X-Cell .60

Figure A.1: Image as provided by the manufacturer.

Configurations	**Value**
Manufacturer	Miniature Aircraft USA
Series	X-Cell .60
Main rotor diameter	1455 mm
Length	1360 mm
Height	413 mm
Mass	4.43 kg
Powerplant	0.61 cu. in. (10 cc) two-stroke glowplug engine

Table A.1: Specifications as provided by the manufacturer.

A.2 Camera

Model: Image Source DFK 41BF02

Figure A.2: Image as provided by the manufacturer.

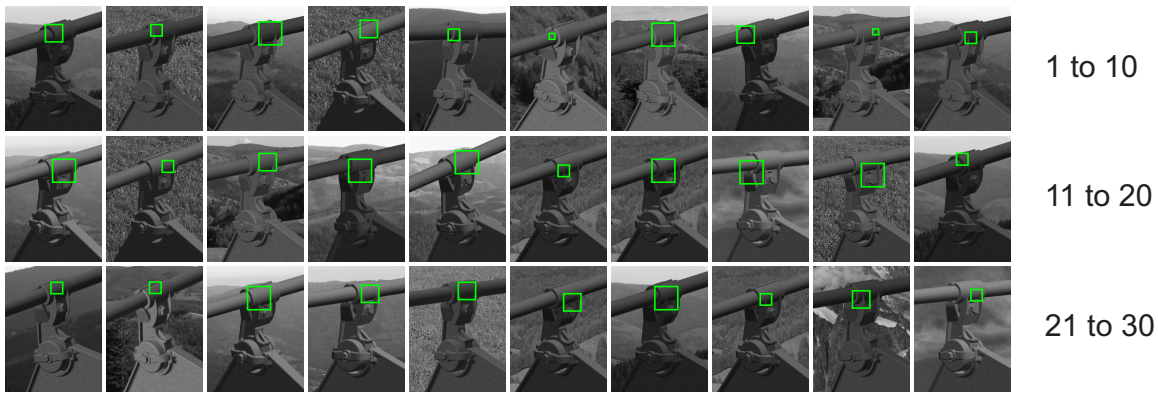
Configurations	Value
Manufacturer	The Imaging Source Europe GmbH
Model	DFK 41BF02
Sensor type	1/2" CCD, progressive scan
Sensitivity	0.5 lx at 1/7.5s, gain 20 dB
Connectivity	IEEE 1394/Firewire
Protocol	DCAM 1.31
Video formats @ Frame rate	1280×960 YUV (4:2:2) @ 7.5, 3.75 fps 1280×960 Y (Mono) @ 15, 7.5, 3.75 fps
Dimensions	H: 50 mm, W: 50 mm, L: 50 mm
Mass	165 g
Additional lens	4 mm or 6 mm

Table A.2: Specifications as provided by the manufacturer. The lens is not included with the camera.

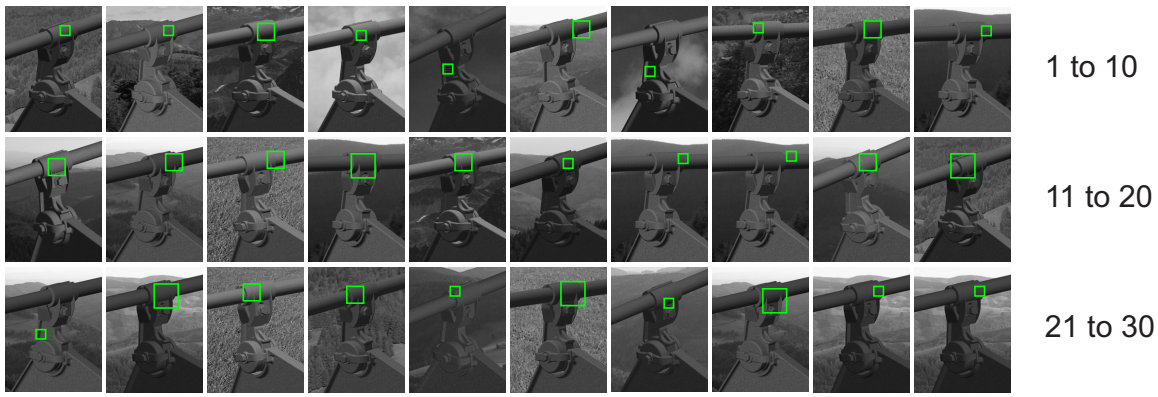
B Learned Patches

This appendix illustrates receptive fields and C2b response of the top-30 patches for all regions and bands. Patches were filtered from 450 extracted patches. 200 Faultless260 images and the Bkg data set were used to select discriminant patches. See more about the experimental setup and in Section 6.2.1.

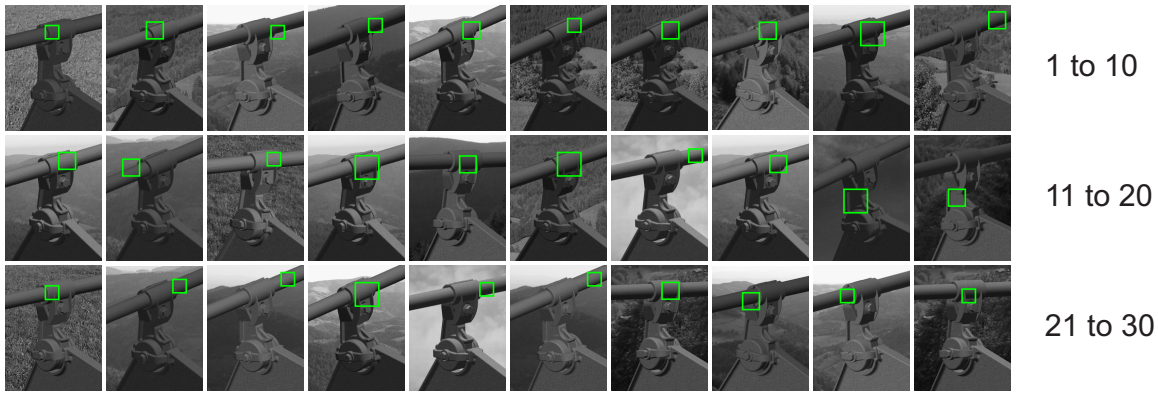
B.1 Extraction Positions



(a) Band 1



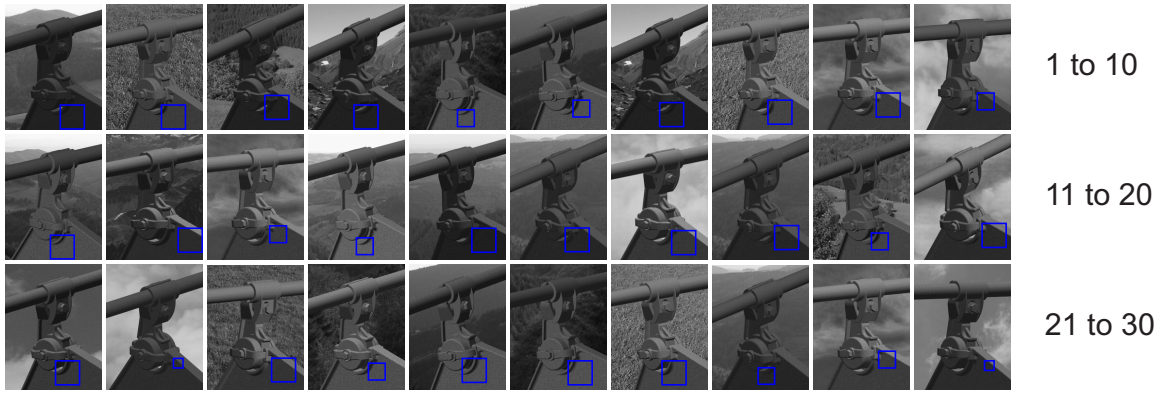
(b) Band 2



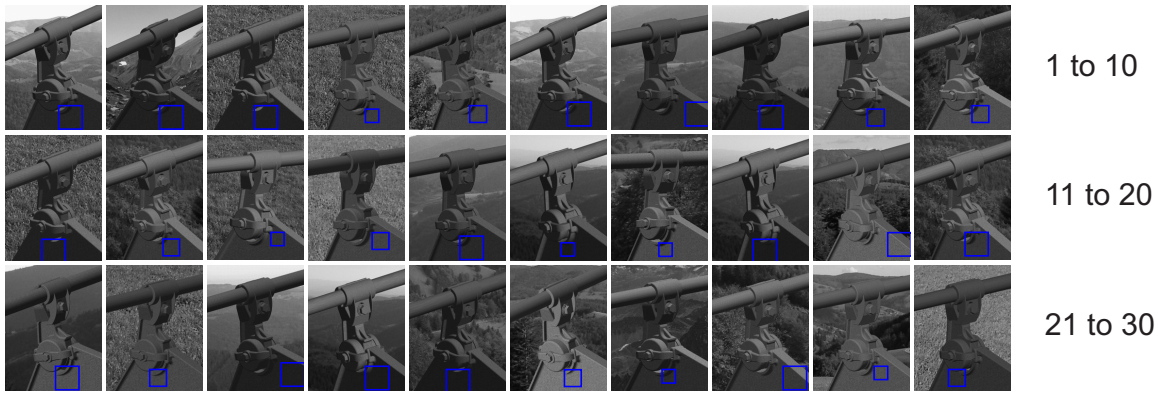
(c) Band 3

Figure B.1: Extraction positions of the top-30 filtered patches extracted from region R_1 in band 1, 2 and 3. The receptive fields of the patches are shown as green rectangles in the images.

(a) Band 1



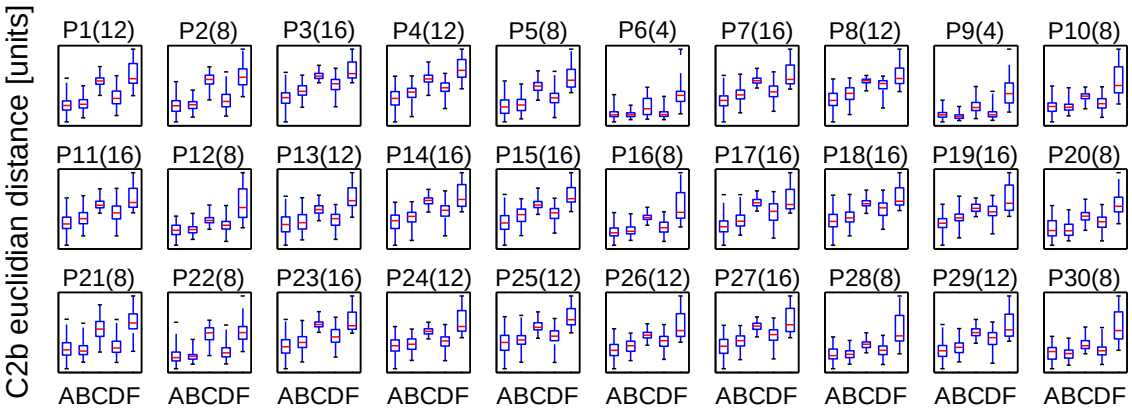
(b) Band 2



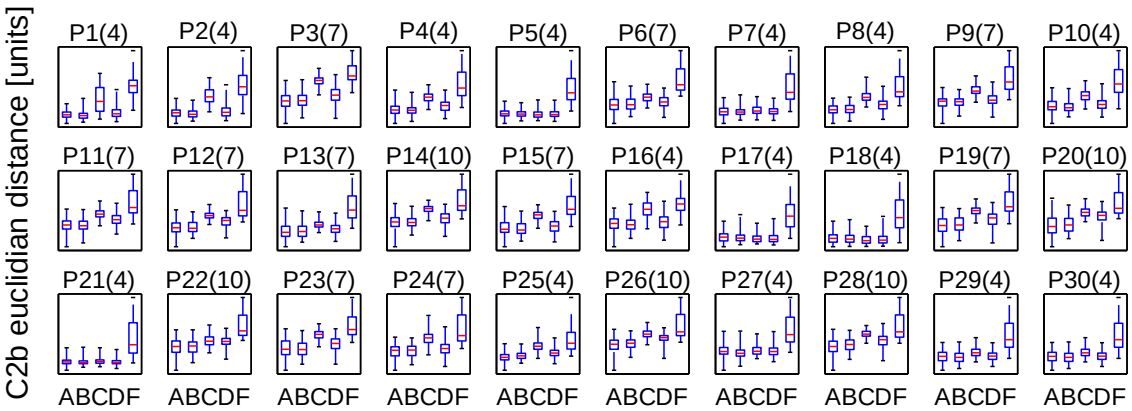
(c) Band 3

Figure B.2: Extraction positions of the top-30 filtered patches extracted from region R_2 in band 1, 2 and 3. The receptive fields of the patches are shown as blue rectangles in the images.

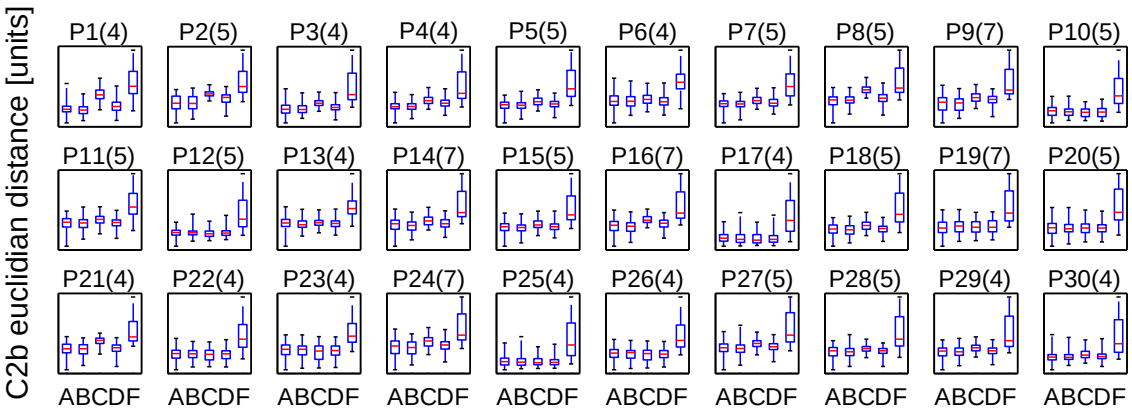
B.2 Response (C2b) Box Plots



(a) Band 1

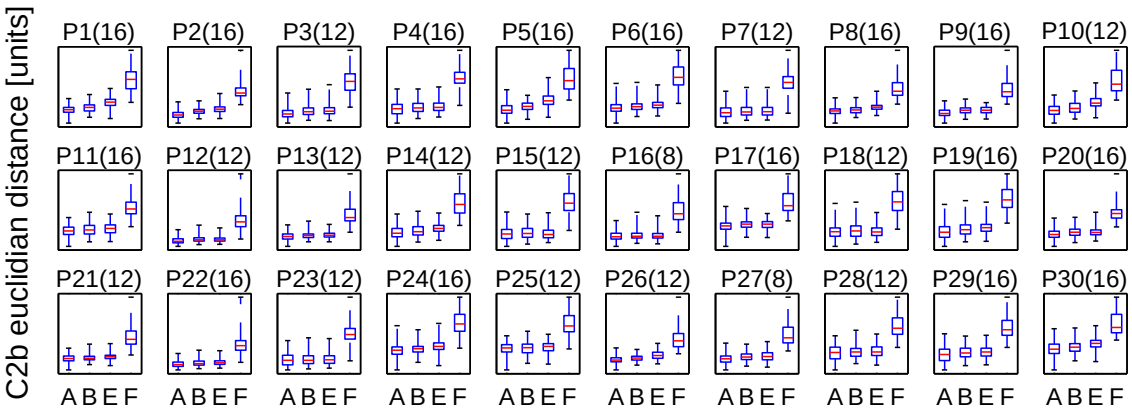


(b) Band 2

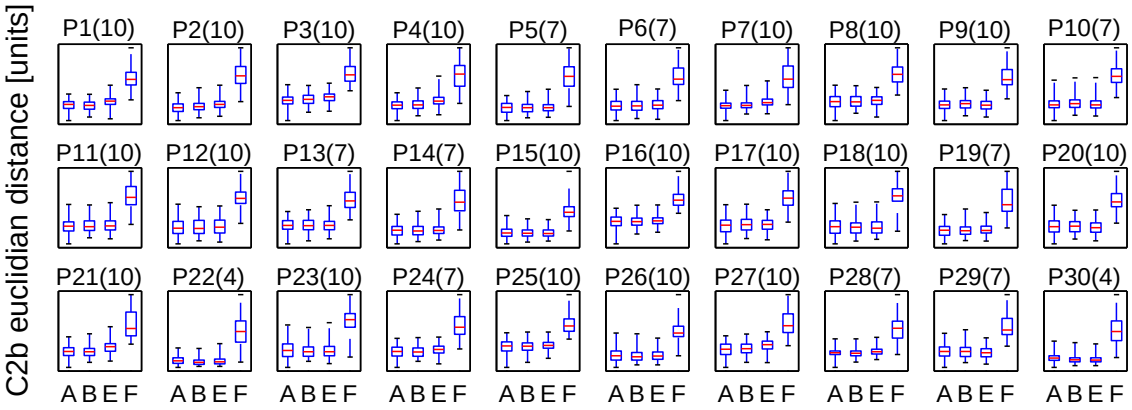


(c) Band 3

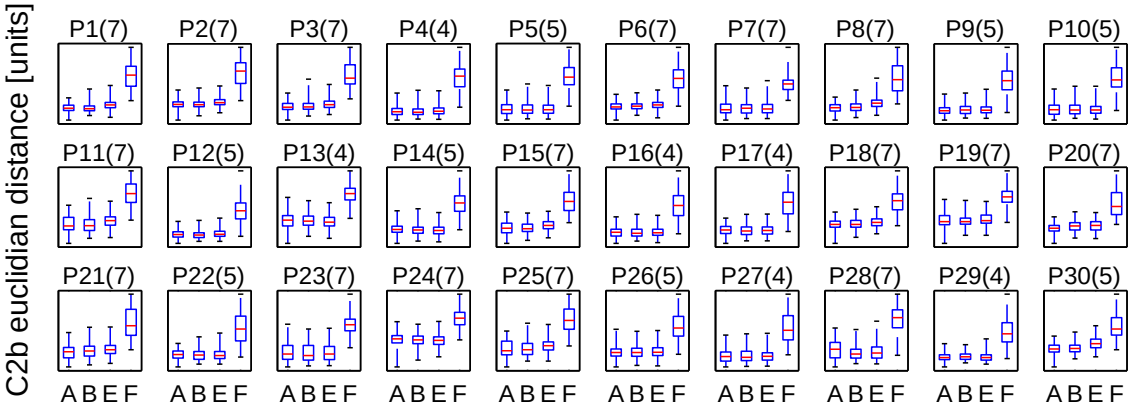
Figure B.3: Response (C2b) box plots of the top-30 filtered patches extracted from region R_1 in band 1, 2 and 3. Each patch P$i(s)$ has rank i and size s. Boxes correspond to data sets A=Faultless260, B=Faultless, C=MissingClaw, D=LooseClaw and F=Bkg.



(a) Band 1



(b) Band 2



(c) Band 3

Figure B.4: Response (C2b) box plots for the top-30 filtered patches extracted from region R_2 in band 1, 2 and 3. Each patch P$i(s)$ has rank i and size s. Boxes correspond to data sets A=Faultless260, B=Faultless, E=ExhaustedSpring and F=Bkg.

Bibliography

[1] V. Barnett and T. Lewis. *Outliers in statistical data. Wiley series in probability and mathematical statistics*. John Wiley & Sons Ltd., 1978. 2nd edition.

[2] T. Binzegger, R.J. Douglas, and K.A.C. Martin. A quantitative map of the circuit of cat primary visual cortex. *J. Neurosci.*, 24(39):8441–8453, 2004.

[3] C. Bishop. Novelty detection and neural network validation. In *IEE Proceedings on Vision, Image and Signal Processing. Special Issue on Applications of Neural Networks*, volume 141, page 217222, 1994.

[4] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Walton Street, Oxford OX2 6DP, 1995.

[5] M. Brown and D.G. Lowe. Unsupervised 3D object recognition and reconstruction in unordered datasets. In *3DIM*, pages 56–63, 2005.

[6] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[7] O. Carmichael. *Discriminative Techniques For The Recognition Of Complex-Shaped Objects*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 2003.

[8] O.T. Carmichael and M. Hebert. Shape-based recognition of wiry objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(12):1537–1552, 2004.

[9] G. Carneiro and A.D. Jepson. Multi-scale phase-based local features. In *CVPR*, volume 1, pages 736–743, 2003.

[10] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[11] Yunqiang Chen, Xiang Sean Zhou, and T.S. Huang. One-class svm for learning in image retrieval. In *2001 International Conference on Image Processing*, 2001.

[12] C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.

[13] F. Desobry, M. Davy, and C. Doncarli. An online kernel change detection algorithm. In *IEEE Transactions on Signal Processing*, volume 53, pages 2961–2974, Aug. 2005.

[14] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical Report HPL-2003-4, HP Labs Tech Report, 2003.

[15] L. Fei-Fei. *Visual Recognition: Computational Models and Human Psychophysics*. PhD thesis, California Institute of Technology, 2005.

[16] L. Fei-Fei, R. VanRuellen, C. Koch, and P. Perona. Why does natural scene categorization require little attention? exploring attentional requirements for natural and synthetic stimuli. *Visual Cognition*, 12(6):893–924, 2005.

[17] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for google images. In *8th European Conference on Computer Vision (ECCV)*, page 242, 2004.

[18] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, 2005.

[19] D.A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, August 2002.

[20] K. Fukanaga. *Statistical pattern recognition*. Academic press, 1990.

[21] H. Greenspan, S. Belongie, R. Goodman, P. Perona, S. Rakshit, and C. H. Anderson. Overcomplete steerable pyramid filters and rotation invariance. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '94*, pages 222–228, 1994.

[22] D.R. Hardoon and L.M. Manevitz. fMRI analysis via one-class machine learning techniques.

[23] C. Harris and . Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference*, pages 147–151, Manchester, UK, 1988.

[24] B. Heisele, P. Ho, and T. Poggio. Face recognition with support vector machines: Global versus component-based approach. In *ICCV*, pages 688–694, 2001.

[25] D.H. Hubel. The visual cortex of the brain. *Scientific American*, 209(5):54–62, 1963.

[26] D.H. Hubel and T.N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cats visual cortex. *J. Phys.*, 160:106–154, 1962.

[27] D.H. Hubel and T.N. Wiesel. Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *J. Neurophys.*, 28:229–289, 1965.

[28] D.H. Hubel and T.N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *J. Physiol.*, 195:215–243, 1968.

[29] D. Huttenlocher and S. Ullman. Object recognition using alignment. In *First International Conference on Computer Vision*, pages 102–111, London, UK, 1987.

[30] N. Japkowicz. *Concept-Learning in the absence of counterexamples: an autoassociation-based approach to classification*. Phd thesis, New Brunswick Rutgers, The State University of New Jersey, 1999.

[31] A. Johnson and M. Hebert. Efficient multiple model recognition in cluttered 3-d scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '98)*, pages 671–677, June 1998.

[32] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):433–449, 1999.

[33] F. Jurie and F. Schmid. Scale-invariant shape features for recognition of object categories. In *CVPR*, volume 2, pages 90–96, 2004.

[34] T. Kadir and M. Brady. Scale, saliency and image description. *IJCV*, 45(2):83–105, 2001.

[35] E.R. Kandel. *Principles of Neural Science*. McGraw-Hill, New York, fourth ed. edition, 2000.

[36] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *Tenth IEEE International Conference on Computer Vision (ICCV'05)*, volume 1, pages 166–173, 2005.

[37] N. Kilroe. Line inspections - eyes and ears of RCM. In *10th IEEE International Conference on Transmission and Distribution Construction, Operation and Live-Line Maintenances*, pages 18–24. Altalink LP, April 2003.

[38] M. Koch, M. Moya, L. Hostetler, and R. Fogler. Cueing, feature discovery and one-class learning for synthetic aperture radar automatic target recognition. *Neural Networks*, 8(7/8):1081–1102, 1995.

[39] A. Kowalczyk and B. Raskutti. One class svm for yeast regulation prediction. *SIG KDD Explorations*, 4(2):99–100, 2002.

[40] D.D. Lea and S. Satoha. Fusion of local and global features for efficient object detection. to appear, 2005.

[41] Y. Lee, K. Lee, and S.B. Pan. Local and global feature extraction for face recognition. In *AVBPA*, pages 219–228, 2005.

[42] N.K. Logothetis, J. Pauls, and T. Poggio. Shape representation in the inferior temporal cortex of monkeys. *Curr. Biol.*, 5:552–563, 1995.

[43] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[44] D.G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

[45] D. MacKay. *Bayesian methods for adaptive models*. Masters thesis, California Institute of Technology, Pasadena, California, 1992.

[46] M. Maruyama, F. Girosi, and T. Poggio. A connection between grbf ane mlp. A.i. memo 1291, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[47] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *CVPR*, volume 2, pages 257–263, 2003.

[48] K. Mikolajczyk, A. Zisserman, and C. Schmid. Shape recognition with edge-based features. In *Proceedings of the British Machine Vision Conference*, volume 2, pages 779–788, 2003.

[49] H.P. Moravec. Rover visual obstacle avoidance. In *International Joint Conference on Artificial Intelligence*, pages 785–790, Vancouver, Canada, 1981.

[50] M. Moya and D. Hush. Network contraints and multiobjective optimization for one-class classification. *Neural Networks*, 9(3):463–474, 1996.

[51] M. Moya, M. Koch, and L. Hostetler. One-class classifier networks for target recognition applications. In *Proceedings world congress on neural networks*, pages 797–801, Portland, OR, 1993. International Neural Network Society, INNS.

[52] H. Nakamura, R. Gattass, R. Desimone, and L.G. Ungerleider. The modular organization of projections from areas v1 and v2 to areas v4 and teo in macaques. *J. Neurosci.*, 13(9):3681–3691, 1993.

[53] C. Papageorgiou and T. Poggio. A trainable system for object detection. *Int. J. Comput. Vision*, 38(1):15–33, 2000.

[54] L. Parra, G. Deco, and S. Miesbach. Statistical independence and novelty detection with information preserving nonlinear maps. *Neural Computation*, 8:260–269, 1996.

[55] D.I. Perrett, P.A. Smith, D.D. Potter, A.J. Mistlin, A.S. Head, A.D. Milner, and M.A. Jeeves. Neurones responsive to faces in the temporal cortex: Studies of functional organisation, sensitivity to identity, and relation to perception. *Human Neurobiology*, 3:197–208, 1984.

[56] C. Plagemann. Ansichtsbasierte Erkennung und Lokalisierung von Objekten zur Initialisierung eines Verfolgungsprozesses. Master's thesis, University of Karlsruhe, 2004. in German.

[57] C. Plagemann, T. Müller, and W. Burgard. Vision-based 3D object localization using probabilistic models of appearance. In *DAGM-Symposium*, pages 184–191, 2005.

[58] A.R. Pope. *Learning to Recognize Objects in Images: Acquiring and using Probabilistic Models of appearance*. PhD thesis, University of British Columbia, Canada, 1995.

[59] A.R. Pope and D.G. Lowe. Probabilistic models of appearance for 3-d object recognition. *International Journal of Computer Vision*, 40(2):149–167, 2000.

[60] M.C. Potter. Meaning in visual search. *Science*, 187(4180):565–566, 1975.

[61] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, November 1999.

[62] M. Riesenhuber and T. Poggio. Models of object recognition. *Nature Neuroscience*, 3 Supp.:1199–1204, November 2000.

[63] G. Ritter and M. Gallegos. Outliers in statistical pattern recognition and an application to automatic chromosome classification. *Pattern Recognition Letters*, 18:525–539, 1997.

[64] S. Roberts and W. Penny. Novelty, confidence and errors in connectionist systems. Technical report tr-96-1, Imperial College, London, 1996.

[65] S. Roberts, L. Tarassenko, J. Pardey, and D. Siegwart. A validation index for artificial neural networks. In *Proceedings of Int. Conference on Neural Networks and Expert Systems in Medicine and Healthcare*, page 2330, 1994.

[66] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. Object modeling and recognition using local affine-invariant image descriptors and multi-view spatial contraints. *International Journal of Computer Vision. To appear*, 2005.

[67] C.A. Rothwell, A. Zisserman, J. L. Mundy, and D. A. Forsyth. Efficient model library access by projectively invariant indexing functions. In *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 109–114, Champaign, Illinois, 15-18 June 1993. IEEE Computer Society Press.

[68] B. Schölkopf, J.C. Platt, J.C. Shawe-Taylor, A.J. Smola, and R.C. Williamson. Estimating the support of a high-dimensional distribution. Technical report 99-87, Microsoft Research, 1999.

[69] B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1083–1121, 2000.

[70] T. Serre. *Learning a Dictionary of Shape-Components in Visual Cortex: Comparison with Neurons, Humans and Machines*. PhD thesis, MIT, April 2006.

[71] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. A theory of object recognition: Computations and circuits in the feedforward path of the ventral stream in primate visual cortex. AI memo 2005-036 / CBCL memo 259, MIT, Cambridge, MA, December 2005.

[72] T. Serre, L. Wolf, and T. Poggio. A new biologically motivated framework for robust object recognition. AI memo 2004-026 / CBCL memo 243, MIT, Cambridge, MA, 2004.

[73] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 994–1000, Washington, DC, USA, 2005. IEEE Computer Society.

[74] L. Tarassenko, P. Hayton, and M. Brady. Novelty detection for the identification of masses in mammograms. In *Proc. of the Fourth International IEE Conference on Artificial Neural Networks*, volume 409, pages 442–447, 1995.

[75] D.J.M. Tax. *One-class classification. Concept-learning in the absence of counter-examples*. PhD thesis, Technische Universiteit Delft, 2001.

[76] The Imaging Source. Lenses - Selection and setup. whitepaper, August 2005.

[77] S.J. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, 1996.

[78] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 586–591, June 1991.

[79] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, 1998.

[80] V. Vapnik and A. Chervonenkis. Theory of pattern recognition [in russian]. Nauka, Moscow, (German Translation: W.Wapnik & A. Tscherwonenkis, Theorie der Zeichenerkennung, Akademie-Verlag, Berlin, 1979), 1974.

[81] V. Vapnik and A. Lerner. Pattern recognition using generalized portraits. *Avtomatika i Telemekhanika*, 24:774–780, 1963.

[82] P.A. Viola and M.J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, volume 1, pages 511–518, 2001.