Using context for the combination of off-line and on-line learning

Jeffrey F. Queißer

Intelligent Systems - Faculty of Technology

University of Bielefeld

A thesis submitted for the degree of

Master of Science (M.Sc.)

2012 June

Abstract

This thesis proposes a classifier combination system in the context of traversability classification. The goal is to enable a system to utilize on-line training for a fast adaptation to new and changing environments. Moreover the on-line classifier is able to handle additional context information that is only available in the on-line environment. Therefore the learning system is based on two different classifiers: One off-line trained classifier is trained in a pre-processing step and one on-line classifier that adapts its internal representation according to the current task complexity. The off-line classifier allows to use general information that is available for the target application area and can be trained by a slow exhaustive off-line training process. The on-line classifier is accountable for a specialization during operation, therefore it is desirable to achieve fast and efficient learning due to the high costs of the acquisition of labeled training data. To achieve a system that is able to combine different types of classifiers that use different and dynamic input spaces an external probability estimation technique is used. The experimental results highlight the benefits of the proposed classifier combination system in comparison to a solution that is based on a single classifier. It was shown that the classifier combination leads to a fast system adaptation to a new environment that was used for on-line learning. Moreover a specialization of the classifiers in a divide-and-conquer manner to certain data set sub-spaces can be observed. The comparison between systems that used different combinations of input features shows that additional context information in combination with a feature weighting technique can be able to improve the system performance. The introduction of a probability threshold for sample rejection allows a considerably performance increase of the classified samples.

Acknowledgements

I would like to express my appreciation to the Honda Research Institute Europe and my supervisors: Dr. Heiko Wersing and Dr. Britta Wrede. Thanks for giving me the opportunity to be part of the HRI-EU for the research involved with this master thesis. Special thanks to the former HRI-EU employee Dr. Stephan Kirstein for his efforts to make this master thesis possible and Dr. Mathias Franzius for his support in relation to python and the utilized simulation environment. Moreover I want to say thank you to all other employees of the HRI-EU for fruitful discussions and a positive working atmosphere.

Contents

List of Figures

LIST OF FIGURES

List of Tables

1

Introduction

Regardless of the fact that computer vision made huge progress in the past fifty years, current methods and architectures are far away from the performance and robustness of what the human brain is capable to achieve. The Scale-invariant feature transformation (SIFT) introduced by Lowe [1], is one of the newer, powerful and commonly used techniques to realize invariant object recognition that can deal with clutter and partial occlusion. The extracted feature descriptors are robust to different image scales, noise, illumination and geometric distortion. Nevertheless robust features are not sufficient to build a flexible and universal object recognition system. Generalization, categorization, multimodality or life-long learning for example are quite difficult to handle and require strategies with higher cognitive abilities to get solved. Most artificial learning systems completely rely on labeled training data in comparison to animals and humans that can extract meaningful representations from unlabeled data. To overcome this limitation the learning with solely or unlabeled data is beneficial since generation of labeled training data is typically expensive. The combination of using labeled and unlabeled data is commonly termed semi-supervised learning. The basic concept is to enable a system to improve its internal representation based on a large amount of unlabeled data. Thereby semi-supervised learning combines the advantage of a higher reachable performance with less labeled training samples. The usage of only unlabeled data is referred to unsupervised learning.

An interactive on-line learning scenario for example requires an adaptation of the internal representation coupled with a short training time. A human tutor should not have to provide a large amount of supervised training examples. Rather the learning

system itself should use just a small amount of meaningful examples given by a human tutor to get a coarse understanding of the data distribution and a much lager amount of unlabeled data samples to refine the borders and distributions of the represented data.

Unfortunately data samples that can only be labeled unreliably by a certain system improve the representation of the used classifier most, because these samples are commonly not well represented in the network. In comparison to this, data samples that are easy to label are normally already well represented by the current state of the network. This is a strong disadvantage of architectures that utilize self-classification, a technique that forces a classifier to generate its own training data.

In the field of autonomous robots there exists another option for the generation of labeled training data. The exploration of the environment can lead, by trial and error, to labeled training data for the performed actions. But one has to mention that in the majority of cases environment exploration is a slow process that can not provide sufficient amount of training samples. Therefore an autonomous on-line system relies on a learning system that is able to provide good learning results using only a small training set.

This work exploits the capabilities of a learning system that is designed to be used in an unsupervised autonomous robot that has to solve the problem of traversability classification. The proposed learning method is not able to handle unlabeled training data, moreover the estimation of labels is motivated by an exploration phase of the autonomous system. The learning system approach that is discussed here targets the problems of classifier combination and the utilization of additional knowledge. Furthermore the system should be capable of on-line and incremental learning. Therefore the system uses two competing classifiers. One classifier is trained by an pre-processing step and represents the initial configuration of the system. The second classifier is able to handle on-line learning and adapts its model to the current problem complexity. The off-line training process is able to insert general knowledge in a well-trained classifier using many training samples. The on-line training process incorporates the ability of on-line adaptation to new environments with regarding the face that the usage of labeled training samples is expensive. Additional context information that is beneficial for the learning task is only available during on-line learning.

1.1 Scenario

The scenario used for this thesis uses two different environments. The first environment is available during the factory configuration of the learning system and is used to build up the representation of the off-line classifier. The second environment represents the local conditions that can be found in the application area. This environment is used for on-line training. Since the configuration, that can be done in a pre-processing step, is not able to cover all variability that is present in the application area, the data set used for off-line training has less structural complexity than the data set used for on-line training. Moreover additional context information like position, rotation or time information is only available for the on-line environment. The scenarios cover an outdoor area including different non traversable objects like trees, buildings and other obstacles. The traversable ground has a high texture variability that includes flowers and sandy spots. The on-line classifier introduces a third class of ignorable objects, these are objects that do not represent the ground but are traversable like leaves or a wooden veranda. The scenarios are designed in a way that objects that have a similar visual appearance do not always have the same class affiliation, like a wooden wheelbarrow and a wooden veranda or a heap of leaves and leaves that are arranged underneath a tree. This ambiguous environment design raises the importance of using additional context information to be able to solve the classification task.

1.2 Aims

The aim of this thesis is to explore the capabilities of a classifier combination system that is capable of on-line learning. The system should be able to utilize additional feature dimensions. This context information is only available to the on-line classifier, because it is not useful for the off-line environment. Therefore the classifiers have different input spaces. The resulting system should be evaluated by the use of data sets generated in a simulation environment. This artificial environment should model a typical outdoor scenario.
These aims lead to the following tasks that have to be accomplished:

- Adapt and utilize 3D simulator
- Design simulation environments

- Plan and implement experimental framework
- Realize an abstract data simulation
- Implement the proposed classifier algorithm
- Train off-line classifier
- Train and evaluate classification system

1.3 Plan of the Manuscript

The second chapter 2 mentions related work that was done in the field of machine learning, categorization, semi-supervised and unsupervised learning techniques, contextualization, feature weighting and selection, classifier combination techniques and traversability classification. The following chapter 3 presents the learning method used in this thesis and the used classifiers in detail. The subsequent chapter 4 introduces the software system used for data set generation and the experimental setup of the proposed learning system. The following chapter 5 shows the results of the experiments that use artificial data set configurations. Experiments using data from the simulated environment can be seen in the chapter 6. This chapter shows a much more detailed evaluation of the learning system and presents in addition different visualizations of resulting state of the learning system. Thereafter the conclusion in chapter 7 gives an overview of the achieved goals and hints for further work that could be done regarding this system.

2

Related Work

There are several powerful learning methods available, but according to the state of the art only a few of them are capable of dealing with incremental and on-line learning. A crucial capability of incremental learning refers to the ability of a network architecture to adapt its complexity to the intricacy of the current task. Most commonly used classifiers are the Multi-Layer Perceptron (MLP) proposed by Rumelhart et al. [2] and the Support-Vector Machine (SVM) presented by Boser et al. [3]. An MLP network is composed of one input layer, one output layer and a number of hidden layers. They can be seen as universal function approximators, but standard MLPs are not able to fulfill the requirements for incremental learning since the network structure must be predefined. Moreover all connections are changed at each training step. This results in a corruption of older representations if not all data samples are presented repetitively. A SVM defines hyper plane class borders to maximize the distances of those class borders to the given data samples. To handle even non-linear separable data, the feature space gets extended by different non-linear kernel functions, but similar to MLP it can not deal efficiently with incremental learning and a changing training ensemble as shown by Kirstein et al. [4].

Category Learning To face the challenge of visual category learning there were lots of different approaches utilized. For example generative probabilistic models, McCallum & Nigam [5] successfully applied a semi-supervised EM based method on text classification, SVM or boosting by Viola & Jones [6]. Most of the used approaches are off-line batch-learning systems, which cannot deal with incremental- or on-line learn-

ing. To overcome this limitation, several incremental learning techniques have been proposed, like vector quantization that is one common class of learning methods.

Vector Quantization There exist various vector quantization techniques, since the prototype based representation has the benefit that incremental learning can be achieved quite easily without changing the whole representation. Due to the ability of an incremental node insertion into the network. One of the intensively used vector quantization approaches are Self-Organizing Maps (SOM) by Kohonen [7], which are artificial neural networks that reduce the dimensionality of data. Additionally they preserve the topological properties of the input. A standard SOM has a predefined node structure and is not capable of dealing with incremental learning, moreover this type of network can not deal with supervised data and is therefore not applicable for a semi-supervised learning scenario. The Learning Vector Quantization (LVQ) by Kohonen [7] and Neural Gas (NG) by Martinez & Schulten [8] approaches represent the data by single prototypes, the LVQ has no constraints to preserve topological properties. On each training step the best matching prototype (LVQ) or a weighted set of best matching prototypes (NG) get shifted in the direction of the input vector. Prototype-based learning methods have the advantage that learning only affects a small portion of the overall network. This has a positive effect on incremental learning and network stability. LVQ includes a class assignment of each prototype so that supervised learning is possible. Furthermore there exist related methods like the Growing Neural Gas (GNG) by Fritzke [9] that overcome the limitation of a predefined network structure by adding further nodes from time to time. The variant incremental learning vector quantization (iLVQ), as introduced by Kirstein et al. [10] is an enhanced LVQ based approach. There each node has a specific learning rate that guarantees the stability of acquired knowledge. Due to the addition of nodes to the network, incremental learning is possible as well.

Unsupervised and Semi-Supervised Learning Common unsupervised learning methods are clustering algorithms and Self-Organizing Maps by Kohonen [7]. One way to achieve semi-supervised learning are generative models like Gaussian mixture models as shown by McCallum & Nigam [5]. This method assumes a model $p(x, y) = p(y)p(x|y)$. Where $p(x|y)$ covers a Gaussian mixture model, therefore the data distribution is approximated by a set of cluster centers and covariance matrices.

One labeled data sample could then label all other data samples inside the same cluster. This assumes that the model assumption is correct. If this is not the case it could reduce the resulting performance (as shown by Cozman et al. [11]). The mixture components are in practice identified by the Expectation Maximization (EM) algorithm by Dempster et al. [12]. It has the disadvantage that it is likely to converge to a local maximum. This can result in wrong classified unlabeled data that has a negative affect to the confidence of the learning system. Further methods are for example graph-based semi-supervised approaches, where each node represents a labeled or an unlabeled data sample and the edges with optional weights reflect the similarity of those samples. The next step is to estimate a function f that fulfills two constraints, first it should give a good approximation of the graph and second it should be smooth on the whole graph. This can be achieved by optimizing the difference between the approximation of the real graph in combination with regularization to prevent over-fitting. The problem of graph-based methods are, that they strongly depend on the graph structure and the assigned edge weights. There are several methods that deal with the optimization problem of graph-based methods, an expatiate on this and the previous methods can be found in the survey by Zhu [13]. To achieve incremental and on-line learning of categories Kirstein et al. [14] presented a framework that autonomously bootstraps the visual representation. The learning process is divided into two phases, first a supervised training and after this the unlabeled training data gets classified by the internal representation without the need of the original training set.

2.1 Context

With the increasing complexity of tasks that are addressed by cognitive systems, information about the context in which the system operates can lead to a better performance. As mentioned by Calisi et al. [15] different robotic tasks ranging from robot behavior to perception can benefit by this contextualization. Calisi proposes a classification scheme that divides context into several layers of abstraction:

- Environmental Context
- Process Context

The mentioned Environmental Context represents a high level information which represents the abstract structure of the environment like rooms and corridors. The Process Context represents low-level information that can be used to improve the underlying mapping process in terms of speed and robustness. The source of information that can be used for contextualization is not restricted to a certain type. For example Hese & Schmullius [16] evaluated the use of context information for forest change classification. For this task the distance of deforested areas to linear object classes are used as additional information, because they can give a hint for human involved deforestation.

2.1.1 Feature Weighting

Removing irrelevant and redundant features can have positive consequences for the underling classifier. Since a reduced input space can eliminate unnecessary classifier complexity and therefore more cost-effective predictors can be realized, as shown by Guyon and Elisseeff [17]
As mentioned by Kirstein [18], feature selection methods can be divided into three types:

1. Filter Methods
2. Wrapper Methods
3. Embedded Methods

The filter methods are computationally less expensive than wrapper methods and robust against over-fitting, whereas wrapper methods tend to find better solutions for feature subset selection. In comparison to the other methods, embedded methods have the drawback that they can only be used in a complete recognition architecture, but this integration avoids additional computation time and can lead to an improved interaction between the classifier and the feature selection process.

Filter Methods select the appropriate features with no regard to the classifier. They can be seen as a dimension reduction pre-processing step.
One basic filter method is the document frequency. This method uses an occurrence counting of each feature to generate a feature ranking. Then the most frequent or least frequent occurring features are then selected for classification. Other ranking methods

like chi-squared test and Information gain can be also used, which are explained in Forman [19]. He did an extensive empirical study of different feature selection metrics. Another selection can be made on the base of the relevance of each feature. The Relief-Algorithm proposed by Kira & Rendell [20] determines reliability weights w_i by the use of the update rule:

$$w_i = w_i + \left(\frac{x_i - nearmiss_i}{\nu_i}\right)^2 - \left(\frac{x_i - nearhit_i}{\nu_i}\right)^2 \tag{2.1}$$

The term x_i represents the ith dimension of the current input sample, $nearmiss_i$ is the ith dimension of the closest sample vector with the another class label and $nearhit_i$ is the ith dimension of the closest sample vector with the same class. ν is a normalization factor. One feature of a set of m features is selected if the relevance exceeds a certain threshold τ and the relevance is given by:

$$Relevance = \frac{1}{m}W \tag{2.2}$$

The determination of *nearhit* and *nearmiss* is based on the euclidean distance on the data space. Kira & Rendell [20] could show that the Relief-Algorithm employs only a few heuristics which has a positive effect on the robustness of feature selection. Moreover the algorithm is tolerant to noise and has a polynomial complexity. But it is not optimal in the sense that the subset of features that is selected by this algorithm is not always the smallest.

Wrapper Methods In comparison to Filter Methods, the Wrapper Methods utilize the classifier as a Black-Box. As mentioned by Kohavi & John [21] this supervised method has to focus its attention on those features that help to solve a given problem and ignore the other features. The filter Methods wrap around a Black-Box classifier as shown in fig. 2.1 The goal is to maximize the performance of a supervised classifier. The idea is to evaluate the classification accuracy using different feature subsets. Finally the feature subset with the highest classification accuracy is used as the final feature set.

One technique to realize a feature selection is Information Bottleneck as published by Tishby et al. [22]. This method is based on Mutual Information. The amount of

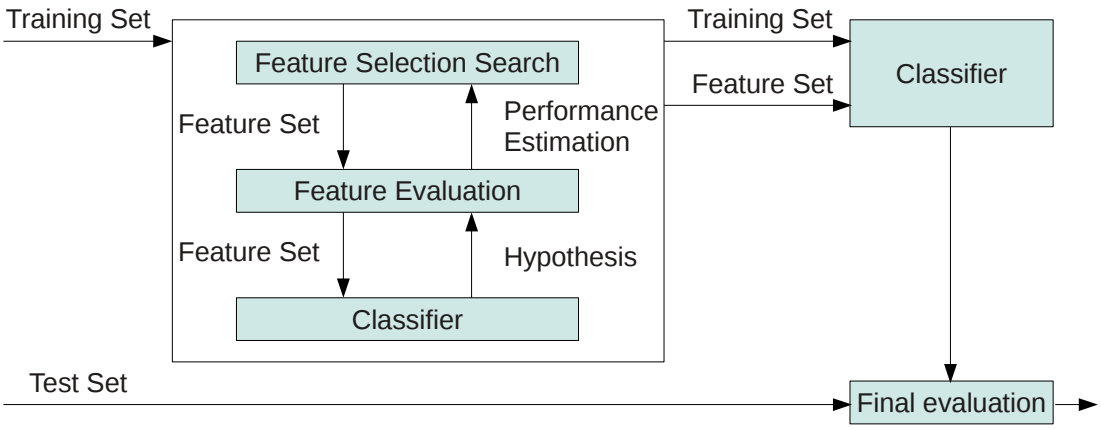


Figure 2.1: Wrapper method concept - The wrapper method has no insights into the functionality of the used classifier. Therefore the classifier is used as a Black-Box. A feature selection tries to select those features that are able to improve the classification result. Adapted from Kohavi & John [21]

information stored by this technique about the data space Y represented in the reduced feature space $\tilde{X}$ is given by:

$$I\left(\tilde{X};Y\right) = \sum_{y\in Y}\sum_{\tilde{x}\in\tilde{X}} p\left(y,\tilde{x}\right) log\frac{p\left(y,\tilde{x}\right)}{p\left(y\right)p\left(\tilde{x}\right)} \leq I\left(X;Y\right) \tag{2.3}$$

Its obvious that the information of the reduced feature set $I\left(\tilde{X};Y\right)$, as given in 2.3, can not exceed the information of the full feature set $I\left(X;Y\right)$. The aim is now to conserve most information about the data space and to minimize the complexity of the feature space. This be interpreted as minimizing $I\left(\tilde{X}:X\right)$, what leads to a minimization of the function:

$$L\left[p\left(\tilde{x}|x\right)\right] = I\left(\tilde{X};X\right) - \beta I\left(\tilde{X};Y\right) \tag{2.4}$$

Another way to obtain a set of optimized features are genetic algorithms as mentioned by Mitchell [23]. A genetic algorithm defines a population of individuals and a fitness function that evaluates the fitness of the individuals. Each individual has a chromosome commonly represented as a binary vector. This chromosome is a candidate solution for the optimization problem. In this case it could represent a binary feature selection. For one evolutionary optimization step a set of individuals is used as parents to create a set of children by recombination and mutation. The probability of being selected as

parent is a function of the fitness of the individual. Therefore better solutions have a higher probability of being selected. One has to mention that genetic algorithms are known for a rapid convergence and their applicability on high search spaces, but it is not guaranteed to find an optimal solution. It is more likely to stick in a local optimum or an arbitrary point.

Embedded Methods The third technique for feature selection and weighting are embedded methods. They embed the selection and weighting of features into the classifier. One way to embed a feature selection into a machine learning algorithm is to extend the error function by a regularization term:

$$E = \underbrace{\sum_{n=1}^{N} \frac{1}{2} \left(y_n - f\left(x_n\right)\right)^2}_{\text{Error Function}} + \underbrace{\lambda \sum_{d=1}^{D} |w_d|^q}_{\text{Regularization Term}} \tag{2.5}$$

The mentioned error function refers to N, the number of training samples, y_n the label of the nth training sample x_n and the classifier is given by function $f\left(x_n\right)$ which gives the estimated label of x_n. The additional regularization abets small feature weights w_d since it summarizes the weights over all dimensions D. The exponent q defines different punishing strategies, for $q = 2$ the method uses a quadratic influence of single weights, for $q = 1$ a linear influence of the feature weights is used and for $q = 0$ all non zero weights have the same influence on the regularization. Another way of fusing feature weighting and classification can be achieved by the Generalized Relevance Learning Vector Quantization (GRLVQ) as proposed by Hammer & Villmann [24]. It extends the Generalized Learning Vector Quantization (GLVQ) by introducing additional weighting factors for each input dimension. This allows a scaling of the dimensions according to their relevance. Thus the GRLVQ provides an adaptive metric whereas most other vector quantization based learning methods focus on the standard euclidean distance:

$$||\vec{x} - \vec{y}||_\lambda^2 = \sum_{i=1}^{n} \lambda_i (x_i - y_i)^2 \tag{2.6}$$

According to [24] the update rule for each weighting factor λ_m is given by:

$$\lambda_m := \lambda_m - \epsilon_1 \left(\frac{d_K}{\left(d_J + d_K\right)^2} \left(x_m^i - w_m^J\right)^2 - \frac{d_J}{\left(d_J + d_K\right)^2} \left(x_m^i - w_m^K\right)^2 \right) \tag{2.7}$$

With lambda adaptation rate ϵ_1, the distance d_J to the next prototype w^J labeled with y^i and the distance d_K to the next prototype w^K not labeled with y^i. To avoid numerical instabilities, all weights are normalized to obtain $\sum_{i=1}^{n} \lambda_i = 1$.

2.2 Classifier Combination

To reach higher classification performances, the idea of using multiple classifiers to improve a single recognition task arose. As Ho [25] mentions, there are different ways how such a combination can be realized. One way would be a divide-and-conquer strategy that leads to a recognition system that selects the best classifier for a given situation. So different classifiers are responsible for different inputs with respect to their confidences. Another way for a combination of different classifiers is a committee approach, this approach uses the results of multiple classifiers to form a combined result. The basic idea of committee votes was already addressed in the 18th century by Condorcet [26]. In this publication it was shown that if each member of a committee has an independent opinion about a subject to be voted on. The probability of a majority vote being correct increases monotonically with increasing numbers of committee members and converges to one, if the chance of being correct for each individual member is greater than 50%. From a technical perspective there are different problems regarding the committee classification: One point is that classifiers based on different operation principles do not necessarily produce comparable classification results and another important argument is that it is may hard to create independent classifiers. As mentioned by Duin [27], the following methods can lead to a higher diversity of a set of different classifiers:

1. Different initializations
2. Different parameter choices
3. Different architectures
4. Different classifiers
5. Different training sets
6. Different feature sets

Since most times training data is expensive, not every classifier can be trained with different data sets. Beside the utilization of classifiers with different operation principles, the variation of classifier parameters can also be used to create a set of different classifiers. But the variation of classifier parameters also does not guarantee a generation of independent classifier configurations.

Name	Combination Method
Product-Rule	$Q_i(x) \sim \prod_j C_{ij}(x)$
Sum-Rule	$Q_i(x) \sim \sum_j C_{ij}(x)$
Maximum-Rule	$Q_i(x) \sim max_j \{C_{ij}(x)\}$
Minimum-Rule	$Q_i(x) \sim min_j \{C_{ij}(x)\}$
Median-Rule	$Q_i(x) \sim median_j \{C_{ij}(x)\}$

Table 2.1: Overview of different classifier combination methods.

Table 2.1 shows several different ways for a fixed classifier combination, the product-rule is suitable for independent classifiers, and could be used if two or more classifiers are completely trained on different feature sets. But it fails if only one of the combined classifiers returns an incorrect result that is close to zero. The sum-rule can be used to average out confidence errors. The maximum-rule selects the classifier with the highest confidence. This can lead to problems if the classifier is overconfident, since it can dominate other classifiers in this case. The minimum-rule is comparable to the maximum-rule with the difference that the classifier that has the least objection against a certain class is selected. The median-rule has the same effect as the sum-rule but leads to more robust results. All the combination rules rely heavily to the quality of the generated confidences. Moreover the outputs of the classifiers could be used as input for a combining classifier that could be trained to adapt the weighting according to the estimated confidences of each classifier.

The problem of selecting a subset of classifiers used for a committee decision is addressed by Aksela [28]. He mentions different selection criteria like mutual information, error correlation and error count and compares them to each other. The outcome of his investigations is that the classifier should as rarely as possible make exactly the same mistakes and that the selection criteria for the classifiers must be appropriate for the given task. Nevertheless the best trade off between accuracy and diversity seems to be the exponential error count criterion. There identical errors made by classifiers are

weighed exponentially. This result again emphasizes the advantage of high classifier diversity.

A famous application area for ensemble classification is face detection based on the AdaBoost method formulated by Freund and Schapire [29]. The AdaBoost method combines many weak classifiers by a weighted summation:

$$sgn\left(\sum_{t=0}^{T}\alpha_t h_t\left(X\right)\right) \tag{2.8}$$

The number of used weak classifiers is given by T, the classifier weight α_t and the binary result of the selected classifier $h_t(X)$ for a given sample X

2.3 Traversability Classification

The vast majority of traverability algorithms and navigation tasks are driven by short range stereo vision that provides a 3D point cloud of the surrounding environment. This generated 3D cloud can be used by heuristics and path finding algorithms to find traversable pathways. But stereo vision can only be utilized for short range path planning since the stereo-based distance estimation is unreliable for huge distances as mentioned by Hadsell et al. [30]. To enable a system to perform a long range path planning additional optical information is beneficial. Therefore Hadsell et al. [30] proposed a navigation system that uses a combination of a near range and far range vision module. The far range module is based on large image patches that are able to fully capture a natural scene element such as a trees or plants. This is motivated by Hadsell et al. [30] by the assumption that a larger context and greater information yields better learning and performance. Using optical features like texture descriptors or other transformation variant features require a set of different pre-processing steps like horizon normalization based on an estimated ground plane or the adaptation of object sizes in relation to their distance to the camera. One way to achieve unsupervised traversability classification is to use the vehicle's navigation experience as presented by Kim et al. [31]. This work proposes an on-line learning system that uses the correspondence of visual terrain data and navigation experiences such as successful traverses, slippages and collisions to generate positive or negative traversability labels. This autonomous label generation allows an adaptation to the wide variety of terrain types such as trees, rocks, tall grass, logs and bushes. The generated labels are assigned to ensembles of

small image patches and a majority voting of the results of all patches of a cluster of patches determines the final classification result. Unsupervised learning of this system is motivated by the fact that the navigation experiences can be assessed automatically by means of on-board sensors such as motor current and bumper switches. So one observation is that most publications targeting traversability classification present a complex system that embeds a multitude of components and information sources like: Stereo map, training data acquisition, hardware components of a real robot, a classifier that is responsible for patch classification in collaboration with depth information or path planning. In comparison to the work of this thesis that targets the classification system core without using depth information nor using a whole path planning scenario. This allows an unaltered evaluation of the classification system performance that could be used in a more complex environment.

3

Learning Method

3.1 Learning Architecture

As mentioned in chapter 2 the performance gain achieved by the combination of multiple classifiers correlates with the diversity of the used classifiers. So a combination of different classifier types is desirable, but this leads to the problem that different classifiers often do not provide comparable results. Moreover adding additional feature dimensions can also lead to different distance metrics that are not comparable. Therefore combination rules as presented in table 2.1 are not applicable. The solution presented here is based on the method proposed by Giacinto & Roli [32]. His learning system uses an input space that is divided into different sub-spaces. For each sub-area an approximated probability, that is generated for each classifier, determines which classification result of the different classifiers is used. The fig. 3.1 shows an exemplary state of the proposed classifier combination system in a 2D feature space. The three prototypes P1-P3 represent the internal state of the on-line classifier. Given a set of

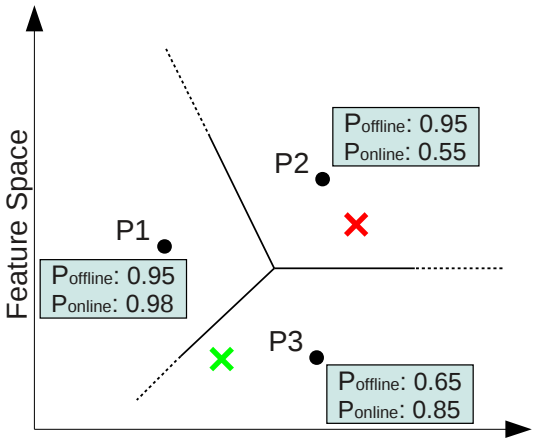


Figure 3.1: Probability estimation - Principle of using Voronoi cells for probability estimation based on feature sub-spaces.

S of prototypes

$$S = \left\{ P_k \subset \mathbb{R}^2 | k = 1, \ldots, K \right\} \tag{3.1}$$

the Voronoi cell $VR(p, S)$ of a prototype p is defined by all points in $\mathbb{R}^2$ that are closer to the appropriate prototype p than to all other prototypes $q \in S \setminus \{p\}$ of the set S:

$$VR(p, S) = \bigcap_{q \in S \setminus \{p\}} D(p, q) \tag{3.2}$$

with $D(p, q)$ defined as:

$$D(p, q) = \{x \in \mathbb{R}^2 : |p - x| < |q - x|\} \tag{3.3}$$

Fig. 3.1 denotes the resulting Voronoi cell borders by black lines and two sample inputs by the red and green cross. The estimated probabilities for the three cells are shown in the blue boxes. In case of the red sample input, the estimated performance of the off-line classifier is higher than for the on-line classifier, therefore the off-line classifier would determine the resulting class label. The cell of the green input sample belongs to the cell of prototype P3. For this cell the on-line classifier has a higher estimated performance than the off-line classifier and determines the system output. Since the Voronoi cells represent the structure of the on-line classifier, the result of the classifier combination system is given by the prototype label of P3.

This leads to the conclusion that a mapping from the input space to the current situation is necessary and that for each situation an approximated probability for each classifier must be determined. The computational cost for the probability estimation and therefore the number of required training views increases with the number of discrete situation states. Using the prototype identifier as situation has different positive aspects: First the number of situations grows with respect to the problem complexity. Another aspect is that an incremental prototype based classifier inserts additional nodes into data regions that contain classification errors. So the density of nodes is related to the local problem complexity. This leads to a more sensitive classifier selection in complex classification areas too. A third point is that no additional computational costs arise for the estimation of the current situation.

For that reason a prototype based learning method was selected as a condition for the on-line classifier architecture. Since the probability for each classifier is determined in a separate process, the system is not restricted in the selection of the classifiers

except that one prototype based classifier is necessary. An overview of the interplay between different components building up the proposed learning system can be seen in fig. 3.2. In the case of this thesis the core of the classification architecture holds two separate classifiers. The first classifier was trained in a pre-processing step by off-line learning. The second classifier is used for the on-line training process. An additional

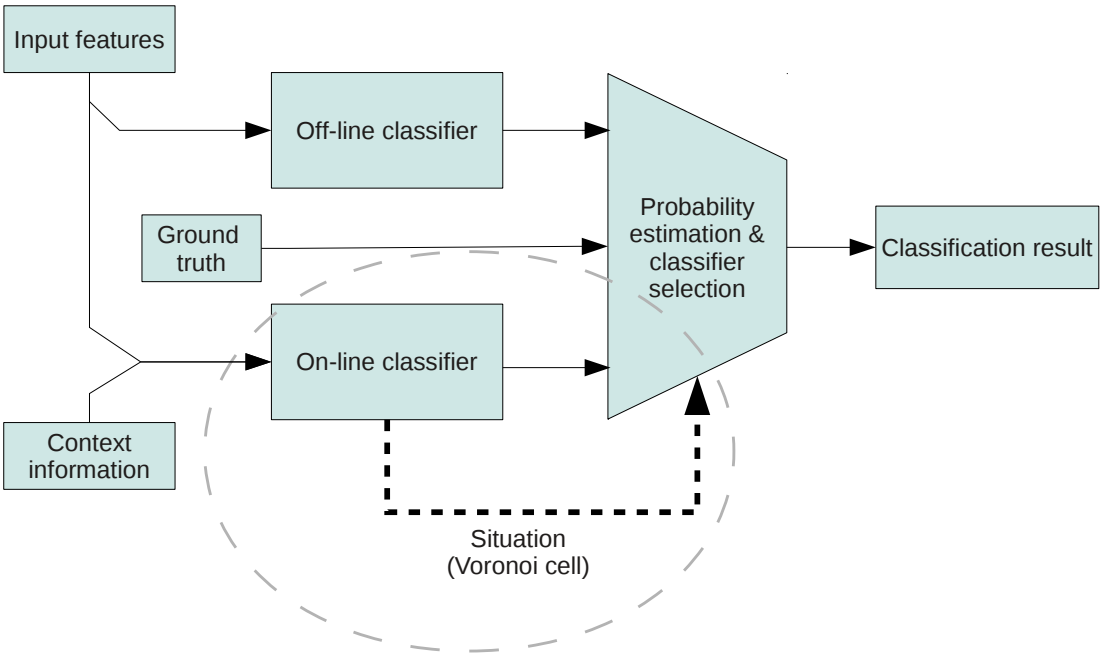


Figure 3.2: Working principle of the used learning system - The shown structure clarifies the working principle of the proposed learning system. This includes the off-line classifier and the on-line classifier. The on-line classifier is able to access additional context information that is not available during off-line training. The state of the on-line classifier is used by the probability estimation for input space partitioning as highlighted by the dashed circle.

selection process selects the appropriate classifier for the current situation. The current situation is given by the active prototype of the on-line classifier. For the initial state of the system, the off-line classifier is pre-trained and is able to achieve generally a better performance than the random initialized on-line classifier. Since the on-line classifier inserts incrementally nodes during the training process, a rising number of Voronoi cells can be used for input space separation and therefore for classifier selection.

3.1.1 Realization

Due to the aims of this thesis, one off-line and one on-line classifier are used. The experiments are based on a Support Vector Machine (Boser et al. [3], LIBSVM [33] implementation) or a Generalized Matrix Learning Vector Quantization and its incremental extension (iGMLVQ) as presented in the following section. The on-line classifier is based on an iGMLVQ for all experiments. The off-line classifier uses a SVM for the experiments that target artificial data distributions. The experiments targeting traversablilty classification utilize an iGMLVQ on-line classifier. Preliminary tests that compared the performance of GRLVQ, GMLVQ and a version that does not use feature weighting showed that the GMLVQ is able to reach the highest classification rates. These test used different artificial data sets and were also performed using gray-scale object images provided by the COIL-100 [34] image library.

The pre-processing step uses the first data set for the training of the off-line classifier. The off-line classifier used for the experiments based on the artificial data set is a SVM using a radial basis kernel function. The experiments evaluating the performance on the data set that was generated by the use of the simulator uses an iGMLVQ. The off-line classifier is based on the same configuration that was used for the on-line classifier, except that this classifier knows just two classes. The off-line training set provides only training patches that belong to the classes of traversable or non traversable objects. The on-line classifier holds three random initialized prototypes, one for each class. Therefore there exist three initial cells that are used for probability estimation. Each new cell is initialized by constant initial values for the probability estimation of the two classifiers. They are selected in such a way that the initial probability values for the off-line classifier are higher than for the on-line classifier. Therefore the off-line classifier it the default option of the initialized system.

During on-line evaluation each given input sample is fed into both classifiers. The prototype id of the on-line classifier is used to determine the active cell and thereby the estimated classifier probability of the off-line and on-line classifier for that cell. The classifier that denotes the highest estimated probability is used to determine the resulting class label. An alternative way of fusing the classifier results was evaluated. There for each classifier and for all three classes probabilities are estimated. They are determined by setting the winner class to $p_{estimation}$ given by the activated Voronoi cell

and the two other classes to $\frac{1-p_{estimation}}{2}$. This leads to the three-dimensional vectors $\vec{p}_{online}$ and $\vec{p}_{offline}$ representing one estimated probability per class for each classifier. Using the Sum-Rule as mentioned in chapter 2 and a normalization to one $\frac{\vec{p}_{online}+\vec{p}_{offline}}{2}$ leads to the probability vector that represents the result of the classifier combination system. This enhanced combination rule does not have an influence on the classification results, since the classification result can only be affected by the changed combination rule if the estimated probabilities for both classifiers select the same class and the probability of one classifier is very low: $\left(\frac{1-p_{loser}}{2}\right) - p_{loser} > p_{winner} - \left(\frac{1-p_{winner}}{2}\right)$. The visualization of the classifier results which can be found in section 6.4.4 show the estimated probabilities acquired by this extended method since there an estimated probability for all classes is shown.

The training process updates the probabilities for each classifier according to:

$$p_{t+1} = p_1 * (1 - \frac{1}{\alpha}) \tag{3.4}$$

if the result of the classifier was not correct or:

$$p_{t+1} = p_1 * (1 - \frac{1}{\alpha}) + \frac{1}{\alpha} \tag{3.5}$$

if the result of the classifier was correct. The term $\frac{1}{\alpha}$ represents the adaptation strength and is defined by

$$\alpha = \begin{cases} 50 & \#\text{samples} < 50 \\ h & \#\text{samples} > h \\ \#\text{samples} & \text{otherwise} \end{cases} \tag{3.6}$$

With a lower bound for the adaptation strength defined by h. The given sample is used as training example for the on-line classifier if at least one of the three conditions is satisfied:

1. the probability of the on-line classifier is higher than for the off-line classifier $p^{on-line} > p^{off-line}$.

2. the classification result of the off-line classifier is wrong.

3. the estimated performance of the best classifier is below a certain threshold $p^{winner} < Q_min$.

3.2 GRLVQ and GMLVQ Learning

The generalized Matrix learning vector quantization (GMLVQ) as proposed by Schneider et al. [35], [36] is based on the Generalized Relevance Learning Vector Quantization (GRLVQ) as proposed by Hammer & Villmann in [24].

Like all supervised vector quantization based learning methods, a finite training set $X = \{(x_i, y_i) \subset \mathbb{R}^n \times \{1, \ldots, C\} \,|i = 1, \ldots, M\}$ is used to determine a set of K prototypes $W = \left\{w^1, \ldots, w^K\right\}$ and their attached class labels $u^k \in \{1, ..., C\}$ that represent the training set. Fore each input sample x_s the winner prototype $w_{k,min}$ having the smallest distance $\text{dist}\,(x_i, w_k)$ is determined by $k_{min} = \underset{k}{\text{argmax}}\,(\text{dist}\,(x_s, w_k))$. The classification result is given by the label of the winner prototype u_k.

The GRLVQ and GMLVQ minimize the cost function proposed by Sato & Yamada [37] that is given by:

$$Cost_{GLVQ} = \sum_v \Phi\left(\mu_\lambda\left(\vec{v}\right)\right) \text{ with } \mu_\lambda\left(\vec{v}\right) = \frac{d^\lambda_{r_+} - d^\lambda_{r_-}}{d^\lambda_{r_+} + d^\lambda_{r_-}} \tag{3.7}$$

The function Φ is a monotonic function like a logistic function or the identity $\Phi\left(x\right) = x$ as used here. The term d_{r_+} is the weighted Euclidean distance to the closest prototype with the same label and d_{r_-} is the weighted euclidean distance to the closest prototype with another class label. The GRLVQ uses the weighted euclidean metric $d^\lambda = \sum_{i=1}^{D_v} \lambda_i \cdot (v_i - w_i)^2$ with $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$. Therefore the distances used in equation 3.7 are defined by:

$$d^\lambda_{r+} = \sum_{i=1}^{D\vec{v}} \lambda_i \cdot \left(\vec{v}_i - \vec{w}_{r+,i}\right)^2 \tag{3.8}$$

and

$$d^\lambda_{r_-} = \sum_{i=1}^{D\vec{v}} \lambda_i \cdot \left(\vec{v}_i - \vec{w}_{r-,i}\right)^2 \tag{3.9}$$

Minimizing of equation 3.7 can be achieved by taking the derivatives with respect to the weighting parameters $\Delta\lambda$, the best matching correct prototype Δw^{r+} and the closest prototype of another class than the training input Δw^{r-}, as shown by Biehl et al. [38]. This results in the update rules:

$$\Delta\vec{w}_{r+} = \epsilon^+ \cdot \Phi'\left(\mu_\lambda\right) \cdot \frac{4 \cdot d^\lambda_{r-}}{\left(d^\lambda_{r+} + d^\lambda_{r-}\right)^2} \cdot \Lambda\left(\vec{v}_j - \vec{w}_{r+,j}\right) \tag{3.10}$$

for the best matching correct prototype,

$$\Delta\vec{w}_{r-} = -\epsilon^{-} \cdot \Phi'(\mu_\lambda) \cdot \frac{4 \cdot d^\lambda_{r+}}{\left(d^\lambda_{r+} + d^\lambda_{r-}\right)^2} \cdot \Lambda\left(\vec{v}_j - \vec{w}_{r-,j}\right) \tag{3.11}$$

for the closest wrong prototype and

$$\Delta\lambda_k = -\epsilon \cdot \Phi'(\mu_\lambda) \cdot \frac{2 \cdot d^\lambda_{r-}\left(v_k - w_{r+,k}\right)^2 - 2 \cdot d^\lambda_{r+}\left(v_k - w_{r-,k}\right)^2}{\left(d^\lambda_{r+} + d^\lambda_{r-}\right)^2} \tag{3.12}$$

for the update of the feature weighting.
In comparison to the GRLVQ the GMLVQ is based on an enhanced distance measurement. There a matrix is used for distance estimation:

$$d^\Lambda(\omega, \xi) = (\xi - \omega)^T \Lambda (\xi - \omega) \tag{3.13}$$

Minimizing the same cost function as for the GRLVQ given in equation 3.7 but based on the distance metric given in equation 3.13 results in the modified update rules:

$$\Delta\vec{\omega}_{r+} = \epsilon^{+} \cdot \Phi'\left(\mu\left(\vec{\xi}\right)\right) \cdot \frac{2 \cdot d^\lambda_{r-}}{\left(d^\lambda_{r+} + d^\lambda_{r-}\right)^2} \cdot \Omega\Omega\left(\vec{\xi} - \vec{\omega}_{r+}\right) \tag{3.14}$$

for the best matching correct prototype,

$$\Delta\vec{\omega}_{r-} = \epsilon^{-} \cdot \Phi'\left(\mu\left(\vec{\xi}\right)\right) \cdot \frac{2 \cdot d^\lambda_{r+}}{\left(d^\lambda_{r+} + d^\lambda_{r-}\right)^2} \cdot \Omega\Omega\left(\vec{\xi} - \vec{\omega}_{r-}\right) \tag{3.15}$$

for the closest wrong prototype and

$$\begin{aligned}\Delta\Omega_{lm} = &-\epsilon \cdot \Phi'\left(\mu\left(\vec{\xi}\right)\right) \cdot \\ &\left(\frac{2 \cdot d^\lambda_{r-}}{\left(d^\lambda_{r+} + d^\lambda_{r-}\right)^2} \cdot \left(\left[\Omega\left(\vec{\xi} - \vec{w}_{r+}\right)\right]_m \left(\xi_l - \vec{w}_{r+,l}\right) + \left[\Omega\left(\vec{\xi} - \vec{w}_{r+}\right)\right]_l \left(\vec{\xi}_m - \vec{w}_{r+,m}\right)\right) \right. \\ &\left. - \frac{2 \cdot d^\lambda_{r+}}{\left(d^\lambda_{r+} + d^\lambda_{r-}\right)^2} \cdot \left(\left[\Omega\left(\vec{\xi} - \vec{w}_{r-}\right)\right]_m \left(\xi_l - \vec{w}_{r-,l}\right) + \left[\Omega\left(\vec{\xi} - \vec{w}_{r-}\right)\right]_l \left(\vec{\xi}_m - \vec{w}_{r-,m}\right)\right) \right)\end{aligned} \tag{3.16}$$

for the update of the feature weighting matrix Λ that is used for the distance estimation given in equation 3.13. It is assumed that Λ is given by $\Lambda = \Omega\Omega$, as shown in Biehl et al. [38].
This enables the classifier to perform a scaling and rotation of the input space, which leads to ellipsoidal clusters that do not need to be aligned to the axes.

3.3 iGMLVQ Extension

As introduced by Kirstein & Wersing [39], an incremental insertion of nodes enables a prototype based network to extend its network complexity to the current problem complexity. Analogous to the implementation of Kirstein & Wersing [39], Kietzmann et al. [40] used this technique to extend the GRLVQ classifier. For the learning system presented here, the same technique was applied for the GMLVQ classifier, resulting in the incremental iGMLVQ classifier.

If there occurs a classification error $d_{r+} < d_{r-}$ during the training process, an internal error counter g is increased. If the erroneous sample has a smaller distance d_{r-} to the wrong prototype than all erroneous samples of that class before it gets saved as a prototype candidate for that class. If a threshold of the number g of occurred errors $g > g_{max}$ is exceeded, new nodes are inserted. Therefore all prototype candidates that are collected on classification errors get inserted as new nodes. If there exists a class that had no classification error since the last time g_{max} was reached, no new prototype gets inserted for this class. After the node insertion process a reset of the current error counter $g_{max} = 0$ is performed and the lists of the stored prototype candidates is cleared.

Therefore the samples having the largest classification error are used as new prototypes.

4

System

This chapter presents the software architecture used for the proposed learning system and the corresponding experiments. The system is partitioned into four sections. The simulation part realizes the generation of data sets and the storage into a database.

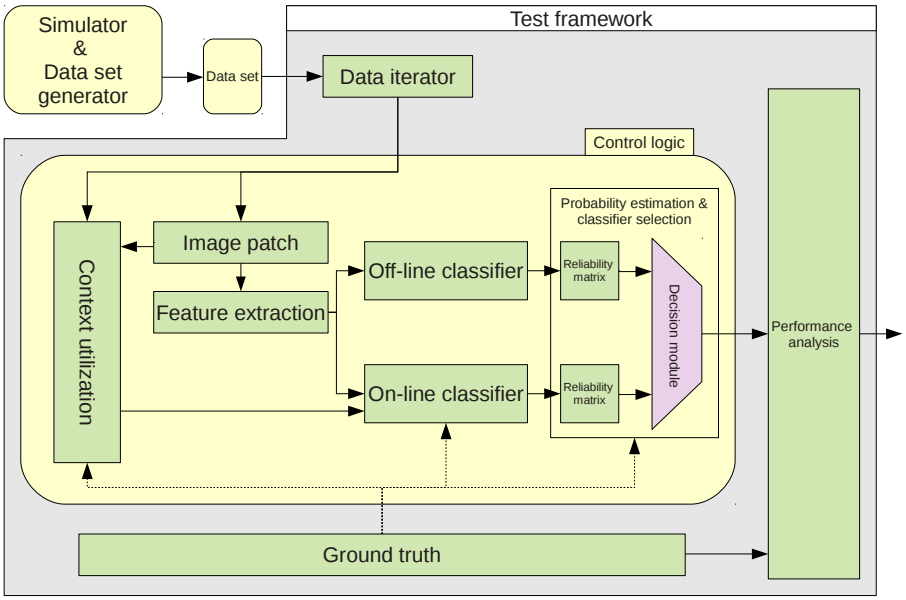


Figure 4.1: Software architecture - The schematic view of the used software architecture. Exchangeable control logic modules allow to test different experimental setups using the same test framework. The data sets generated by the simulator are stored for later usage by the experiments.

The second important section of the system contains the infrastructure to execute different test runs and to save and visualize the results of the performed tests. This test environment provides an interface to a control logic unit that holds the current experimental configuration. By changing the control logic module, different test configurations can be realized. The fourth crucial element are the classifier that are hidden inside the control logic. The whole software architecture except the classifier cores are implemented in *python* using the famous extensions *matplotlib* and *scipy*.
Fig. 4.1 shows a schematic overview of the software architecture. It can be seen that the classifier combination system is embedded into the control logic module. In addition to the classifier combination, the control logic handles the image patch generation, the feature extraction and the context utilization. Therefore all processing steps that are relevant for a given experimental configuration are integrated into this module. The surrounding test framework handles the access to the data sets and the extraction of the ground truth. In addition the test framework evaluates the performance of a given control logic element by comparing the ground truth and the estimated class labels. Since the simulator is an external process and the generated data sets are stored externally they are not included in the test framework module. The following sections give an overview over all components that are mentioned in fig. 4.1.

4.1 Simulator

The simulator is used to generate a collection of data sets in a batch process. Each data set entry consists of the following entities:

1. Visual representation
2. Depth map
3. ID map
4. Context information

The context information contains the position, rotation and the time information.

4.2 Data Sets

All data sets are stored and can be selected as training or evaluation data set input for the test framework. The samples are presented in the same order to the classifier as they had been recorded during the simulation. This simulates the training data presentation during on-line learning since the acquired training data cannot be shuffled during a real on-line training process.

4.3 Test Framework

The test framework collects all elements that are necessary to build a complete classification framework. For each experiment the same test framework is used. It controls the iteration through the training and test data sets. The resulting functionality is mainly defined by the referenced control logic. The framework controls all inner units to realize the desired system behavior.

4.3.1 Data iterator

The data iterator iterates over all data set entries. Therefore it takes an iteration step width which defines the number of used training samples. The recorded data sets used for the experiments presented in this thesis consist of 3000 recorded samples.

4.3.2 Performance analysis

The Performance analyzer collects all generated output values and is able to visualize them as a graph or store them in a log file. Sample patches that show multiple objects that belong to different classes are neglected and not used for the estimation of the system performance.

4.3.3 Ground truth

The ground truth module determines the correct class labels. Therefore it uses the depth-map and the ID-map provided by the data iterator. The result is a three-dimensional output matrix holding binary values that carry the class affiliation of each patch. This module selects only patches that have a clear class affiliation. This is realized by ignoring patches that represent two or more different classes. Additionally

the module tries to balance the number of training views between the traversable and the other classes since traversable ground patches dominate the recorded data. This is realized by selecting the same amount of patches of the traversable and the ignorable obstacle classes.

4.3.4 Control logic

The control logic module implements the core of the desired learning system behavior. Different configurations allow the simulation of different combination systems. Additionally a PCA-estimator module was realized to determine the principal components of a given data set used by the feature extractor module.

4.3.4.1 Image patcher

The image patcher is used to split the input images into rectangle image patches. These image patches are not overlapping and the desired size can be configured by adjusting external parameters.

4.3.4.2 Feature extraction

The feature extraction is based on the PCA dimensionality reduction. Therefore a pre-processing step is used to determine the eigenvectors and eigenvalues of the input patches of a training data set. The three eigenvectors having the biggest eigenvalues are then used to transform the high-dimensional image patch into a three-dimensional feature vector.

4.3.4.3 Context utilization

A further pre-processing step is used to select the desired input features. That can include the three-dimensional visual features, position information, rotation information and time information that was used during simulation of the ambient light. The selected features are then transformed to get a variance of one and zero mean.

4.3.4.4 Off-line classifier

The off-line classifier only uses the visual features for training. A pre-processing step is used to train this classifier with respect to an off-line training set. The resulting

performance of the off-line classifier represents the baseline of the whole classification system since the initial configuration prefers the result of the off-line classifier.

4.3.4.5 On-line classifier

The on-line classier is only used during the on-line training process. As described in chapter 3 it is based on a prototype representation that incrementally inserts new nodes. The id of the closest prototype is used by the decision module to determine the estimated probabilities of both classifiers for different sub-spaces of the input data.

4.3.4.6 Reliability matrix

This module estimates the probabilities for each classifier and generates an reliability matrix including a three-dimensional vector for each given input patch of one input frame. For the experiments used here just the winning class is interesting. So the dimension of the winner class is set to the estimated probability $p_{estimation}$ whereas the other two dimensions are set to $\frac{1-p_{estimation}}{2}$. This guarantees that the sum of the probability of all classes equals one.

4.3.4.7 Decision module

The decision module selects the entry having the highest estimated probability. The result of the selected classifier represents the final output of the classifier combination system.

5

Abstract Data Set

This section explains the experiments that have been realized using two different artificial data sets. Each data set is based on three classes to simulate the required classes: Traversable, non traversable and ignorable obstacles. For the first experiment the off-line classifier is trained on the traversable and non traversable data and only the on-line classifier makes use of the additional ignorable class. The first experiment compares classification results between 2D and 3D data. This was used to simulate the availability of additional feature dimensions to the on-line classifier. In contrast to the first experiment, the second experiment refers a data set that has a more complex data structure for the on-line and the off-line classifier. This experiment was done to highlight the benefits of the proposed classifier combination concept.

5.1 Artificial Data Set I

The data set used for the first experiment was generated by five different multivariate Gaussian distributions. Fig. 5.1 shows the arrangement of all three classes. All data points are divided into two subsets, one for training and one for evaluation. The class that is additionally used for the on-line training process is color coded by blue. The two other classes are used for on-line and off-line training. It can be seen that the blue class is not separable from the other classes in a 2D space. Therefore two versions of the data set have been used, the first uses the non separable 2D configuration and a second version uses an additional dimension that represents the context information. As shown in fig. 5.2 the blue class is separable in the third dimension since there are

no overlapping distributions in this space. This configuration was chosen to simulate contextual information if non separable input data can be separated by the use of an additional feature dimension.

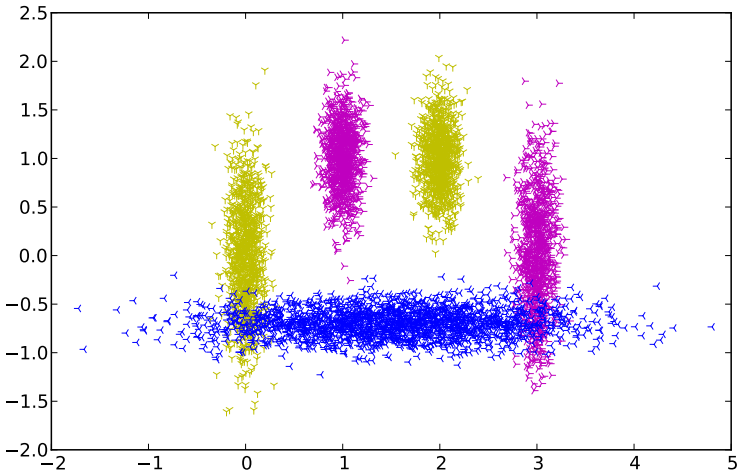


Figure 5.1: Artificial data set I visualization XY - Each color codes one class. The off-line classifier is only trained on class one(yellow) and two (magenta). The off-line classifier operates also on one additional third class(blue). It can be seen that the third class cannot be separated from the other classes in the 2D space due to the overlapping areas at [0,-0.75] and [3,-0.75]

The final state of the on-line classifier at the end of a training process that used 2900 samples is shown in fig. 5.3. The comparison of the training results achieved by using the 3D data set to the results of the 2D data set reveals that the network contains fewer nodes in the case of the 3D data set. Especially the sub-areas of the overlapping regions of the classes have a much denser node distribution. This effect results by the fact that it is not possible for the classifier to separate the two classes. Therefore the classification error rate cannot decrease any more at a certain time and this leads to more incremental node insertions.

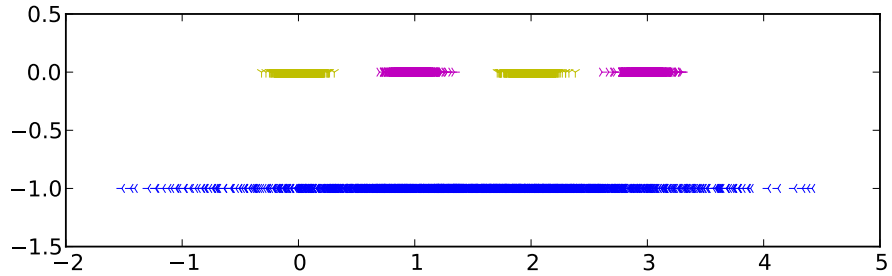
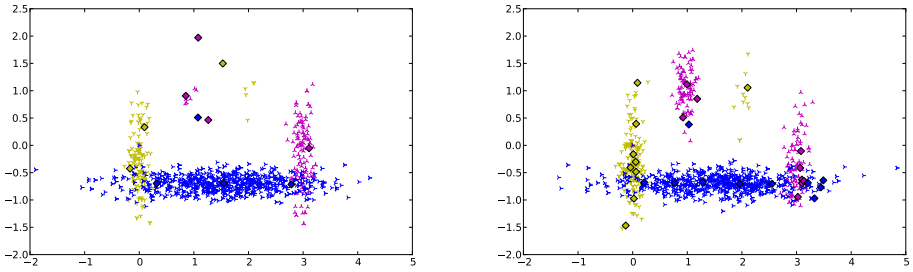


Figure 5.2: Artificial data set I visualization XZ - Each color codes one class. The off-line classifier is only trained on class one(yellow) and two (magenta). The off-line classifier operates also on one additional third class(blue).

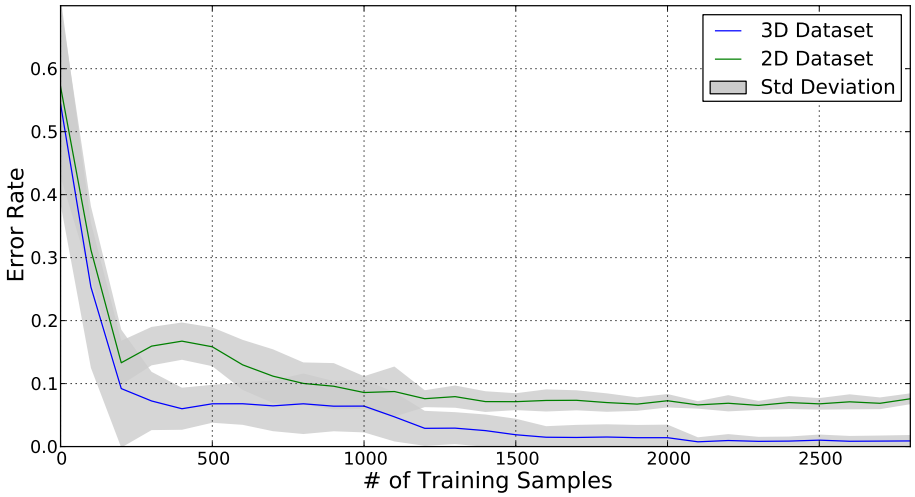


(a) **(b)**

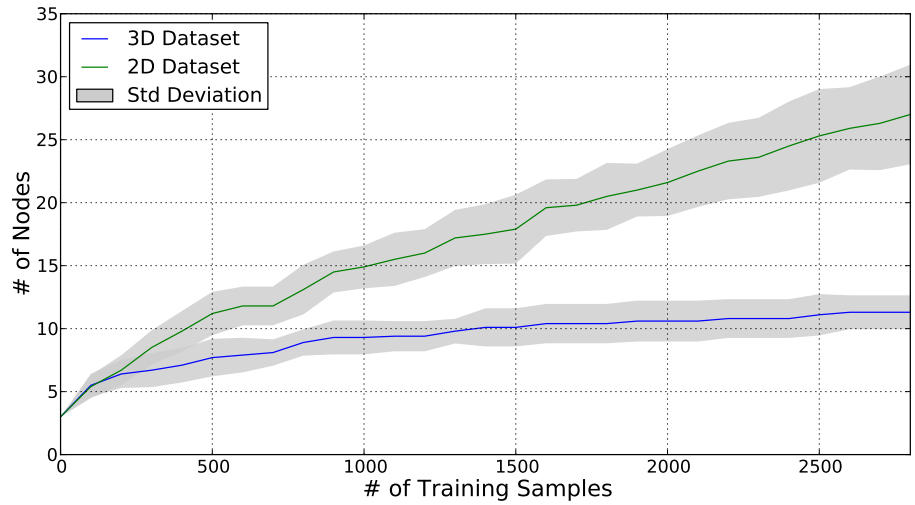
Figure 5.3: Artificial data set I node distribution - Both figures show the distribution of nodes at the end of 2900 training steps. The left figure (a) shows the result of the training using the 3D data set and the right figure (b) shows the result of the training using the 2D data set(third dimension is zero for all data points). The visualized data points represent those samples that were used for the training of the on-line classifier.

Since fig. 5.3 shows only those data points that are used for on-line classifier training, it can be seen that the on-line classifier uses a smaller potion of the two classes centered at [1,1] and [2,1] of the 3D data set. But due to the simple structure of the data set, the on-line classifier has seen samples of all distributions and has inserted sufficient nodes to represent the whole data set. This leads to the conclusion that this experiment does not show the benefits of an additional off-line classifier.

The evaluation of the error rate reveals that the classifier system using the 3D data set for on-line learning is able to reach a higher classification rate. The results are shown in fig. 5.4(a) This shows that the classification system is able to benefit from the additional feature dimensions. The error rate at the beginning of the training process (0 presented training samples) marks the base line and the performance reached by the off-line training.
For this experiment the error rate after off-line training is over 0.5, this is caused by the blue class that is not available to the training process of the off-line classifier.
The investigation of the development of the number of nodes during the training process confirms the assumption that the problem of class separation leads to an unlimited node insertion of the incremental training process in case of the 2D data set. This can be seen in fig. 5.4(b), the graph shows the number of inserted nodes of the on-line classifier. The comparison between a classifier training using the 2D and the 3D data set shows that the node insertion rate is not declining during training using the 2D data set.



(a)



(b)

Figure 5.4: System evaluation during training with artificial data set I - The figure (b) shows the number of incremental inserted nodes of the on-line classifier. The blue graph shows the results of the training using the 3D data set and the green graph represents the results using the 2D data set. The gray area surrounding the graphs indicates the standard deviation of the results. The results represent the average of $n = 10$ runs.

5.2 Artificial Data Set II

The second experiment is based on a more complex data set configuration. The data set is based on 21 different multivariate Gaussian distributions. They are arranged into two symmetric star like configurations. The additional blue class has much less overlap and is located at the center of the right star as shown in fig. 5.5. In comparison to the first experiment there exists no 3D version of this data set.

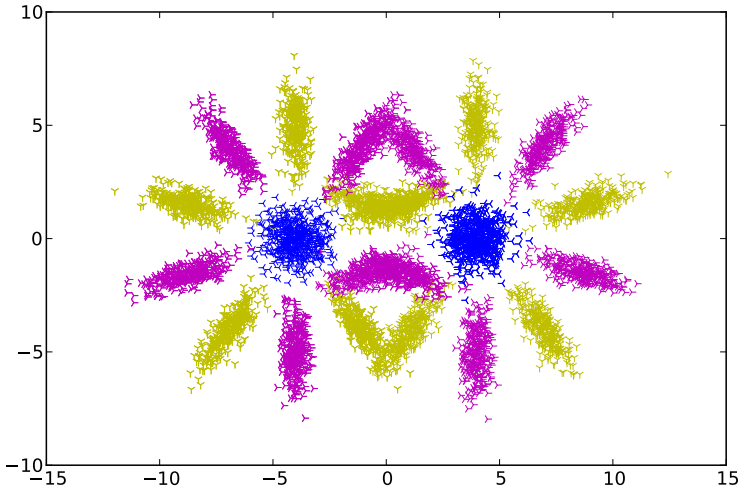


Figure 5.5: Artificial data set II visualization - Each color codes one class. The left star configuration was used for off-line classifier training. This includes all three available classes in comparison to the first experiments that used only the classes one(yellow) and two (magenta) have been used for off-line classifier training. For on-line learning the complete data set was used.

The left star configuration was used for off-line classifier training and for on-line training the whole data set was used. The data set was divided into two separate chunks for training and evaluation in the same way as for the first experiment. Since the used classifiers are prototype based, a symmetric arrangement of distributions has no effect on the classifier performance. The results of the evaluation of the error rate during the

training process is shown in fig. 5.7(a). It can be seen that the on-line training process is able to improve the classification rate for this more complex scenario in the same way as for the much simpler data set used for experiment I.

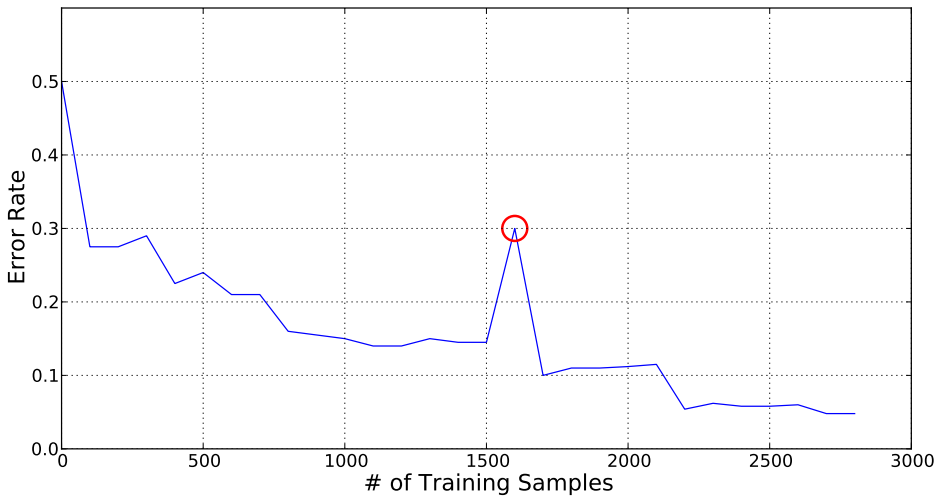
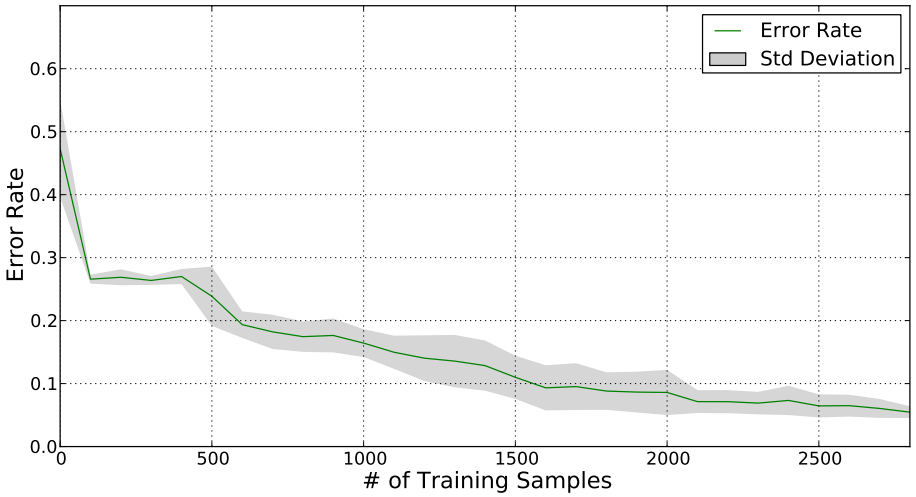
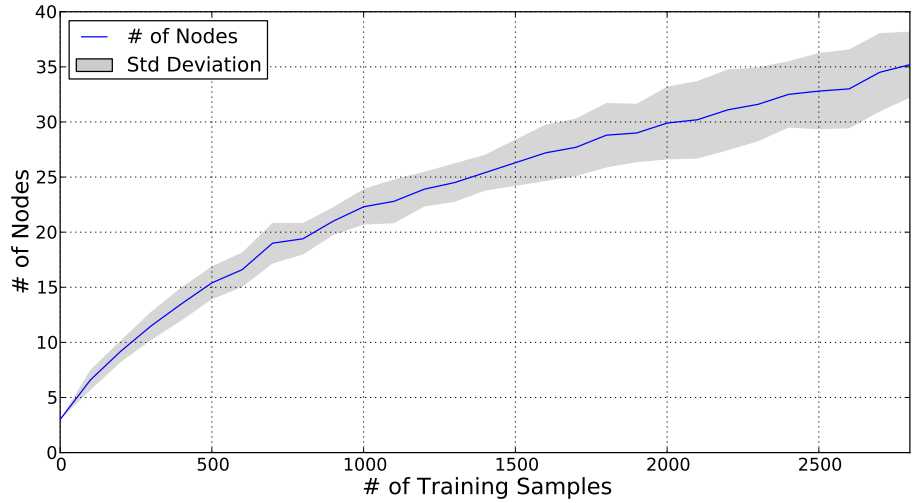


Figure 5.6: Single run system evaluation - This figure shows the development of the error rate during one training process. The insertion of nodes can have a negative effect on the performance development. The red circle marks an error rate peak between 1500 and 1750 presented samples.

The development of number of nodes during training is shown in fig. 5.7(b), where it can be seen that the node insertion rate declines over time. Due to the complex structure of the data set and that it is not possible to find a perfect separation of all distributions (overlapping) there are still node insertions until the end of the training process. The investigation of the evaluation of one single run reveals that it is possible to achieve a performance degradation during training, as shown in fig. 5.6. The short performance decrease between 1500 and 1750 presented training samples (marked by red circle) is caused by the node insertion process. Since the system assigns default performance values to newly inserted nodes it takes some training samples adjacent to that newly inserted node to adapt the estimated performance values. The configuration of the nodes at the time of the performance decrease is shown in fig. 5.8(d). In comparison to the system configuration before the performance decrease occurs in fig. 5.8(c) it can be seen that a new yellow node was inserted at the position [0.2, -0.25]. The selection

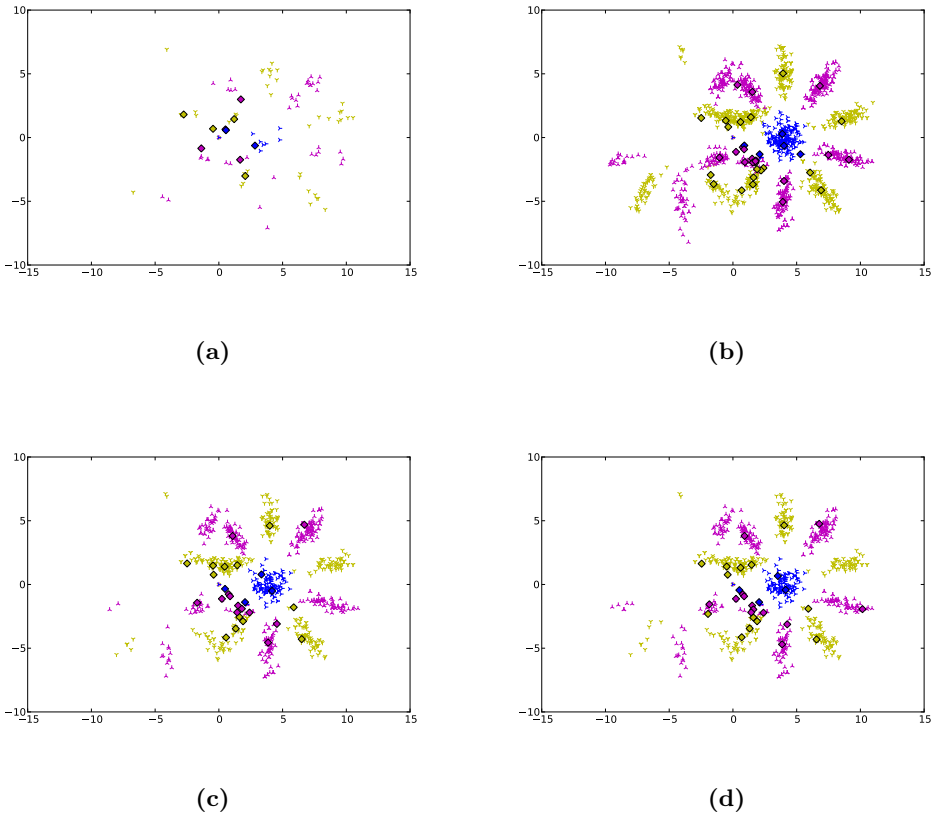


(a)



(b)

Figure 5.7: System evaluation during training with artificial data set II - The top figure (a) shows the development of the system performance during training. The bottom figure (b) shows the number of incrementally inserted nodes of the on-line classifier. The gray area surrounding the graphs indicate the standard deviation of the results. The results represent the average of $n = 10$ runs.



(a) **(b)**

(c) **(d)**

Figure 5.8: Prototype representation of the artificial data set - The four images illustrate the prototypes of the on-line classifier at different training states, moreover the given training samples are plotted. The color indicates the class label of each training sample. The first image (a) shows the state at the beginning of the training process, at this time the system had processed only 200 presented samples. (b) shows the state at the end of the training and after presenting 2900 samples. The lower graphs clarify the state of the training system after 1500 (c) and 1600 (d) presented samples. At this time a performance decrease of the overall system occurs. The visualized data points show those samples that were used for the training of the on-line classifier.

of the wrong classifier for this node leads to a misclassification of a huge portion of the left star configuration, since the Voronoi cell covers a huge region of the input space. The prototype configuration of the on-line classifier at the end of 2900 training steps is shown in fig. 5.8(b). It can be seen that only small portions of the data set used for the off-line training was used for training. Moreover the newly presented star configuration on the right is represented by the nodes of the on-line classifier. The figure reveals also that a higher node density is present in the area of the overlapping distributions of both stars. This is caused by an overlap between the yellow and magenta class of the two star configurations.

5.3 Conclusion

The experiments using the artificial data set showed that the classifier combination concept presented in this thesis is able to create a separated input space and also is able to create specialized classifiers for different feature sub-spaces. In particular the benefits can be seen if each individual classifier is not able to cover the whole data space. Experiments targeting the context information could also show that an additional dimension that leads to a clear class separability is able to improve the classification performance and the network structure. The visualization of the error rate development during the training process reveals that due to the incremental node insertion short performance decrease peaks can occur. This is caused by an unsure probability estimation for newly inserted nodes. Further investigations could address the question if the neighboring nodes could give a bias for the initialization of the performance estimation. The experiments revealed that the used node insertion technique causes an unnecessary amount of node insertions into data regions having an ambiguous class affiliation. But this problem is not related to the classifier combination method nor the effect of the additional feature dimensions. Moreover the experiment does not show if the proposed system can be used to achieve an enhanced system performance in comparison to its underling single classifier. This question is addressed by the following experiments targeting the simulation environment.

6

Simulator Based Data Sets

To evaluate the system behavior in a more realistic scenario a data set that was generated in a simulated environment was used for the evaluation of the classifier combination system. To achieve a more realistic result an outdoor scenario is selected that includes different types of objects and high texture variances for the same object types. The following experiments present the used simulator and the results that could be achieved using this artificial environment.

6.1 Simulation Environment

The used simulation environment was built on top of a 3D-Simulator tool. The environment consists of different elements:

- Ground Planes
- Object (Planes)
- Environment
- Light Source
- Mobile Robot

In the following the elements will be presented in more detail:

Ground Planes These objects build the ground surface of the outdoor environment, the planes define potential traversable areas. To enhance the visual diversity a set of different ground textures was used, they are chosen randomly.

Object (Planes) In relation to ground planes, the object planes define non traversable and ignorable obstacles. Traversable obstacles change the class id of that region but are traversable by the robot. The obstacles represent objects that trigger an orientation randomization of the robot on a collision. This avoids an intersection of the simulated robot and the non traversable obstacles.

Environment The environment is based on an environment map simulating a sky and a surrounding scenery. This is mainly used for beautification of the visualizations since the used image patches are only selected as training samples, if they are close to the robot position.

Light Source The light source defines the environment light characteristic and the generation of the shadows that can be seen inside the simulated environment. The variation of the light color was used to simulate light variances over the time of day.

Mobile Robot This element is used to capture visual streams. During simulation the robot is moved step by step and after a specified step-width the next image is captured. If the robot would collide with a non traversable object, it is rotated around a random angle. This avoids the collision and intersection with other elements and generates a random path inside the simulated environment.

6.2 Scenario Environment

The scenarios used for the following experiments are based on an outdoor environment. Two different environments have been modeled. The first was used for the training of the off-line classifier, the second for the on-line training process. Two independent random routes for each environment generated two independent data sets for each configuration for training and evaluation. In general the environment used for off-line training is less complex than the environment used for on-line training.
The environment used for off-line training as shown in fig. 6.1 has less structured elements on the traversable ground than the environment used for on-line training, as visualized in fig. 6.2. Additional yellow flowers and darker green grass patches enhance the variability of the grass plane. The buildings have different locations as well as different color schemes. Objects that appear in both scenarios are a garbage can and a

Figure 6.1: Environment I - The outdoor environment used for off-line classifier training.

park bench, both located outside the traversable area. The trees used in both scenarios have also a different coloring of the leafs, but the foliage has the same appearance for both scenarios.

Additional objects that have been introduced in the on-line scenario are a motor scooter located beside the traversable area, a hand barrow, a heap of leaves, a wooden veranda and a wooden garden fence. The appearance was chosen in a way to enforce visual ambiguities. Therefore the wooden veranda, wooden garden fence and the hand barrow have the same texture but different class affiliations, since the veranda is traversable and belongs to the class of ignorable objects. Moreover the foliage beneath the trees has the same texture as the heap of leaves that belongs to the class of the non traversable objects. The surrounding area is constructed by a random arrangement of trees onto a dark green ground plane. The traversable area is limited by grey curb stones, they belong to the class of non traversable objects.

During data acquisition the shadow generation is enabled. Additionally it is possible to change the lightning source position and the emitting light. This leads to a simulation of different time of days and creates more difficult data sets since the position of the shadows and the visual appearance changes in relation to the simulated time.

Figure 6.2: Environment II - The outdoor environment used for training the on-line classifier and the system based on the classifier combination

6.2.1 Data sets

For the creation of data sets, a simulated autonomous robot was moved around the two outdoor environments. The robot was equipped with a virtual camera that delivers depth information and a color image into the frontal direction. The robot was moved step by step until it would hit a non traversable object. In that case the robot moved into a random direction to be able to contain with a forward movement. This leads to different random pathways for the different runs. After each step, one snapshot was added to the database. Therefore the movement of the robot is represented in the recorded data bases. A data base entry of one time stamp contains the following elements:

1. Color image (256x256x3 Pixel)
2. Depth image (256x256x1 Pixel)
3. Position (xy-position)
4. Rotation (radian measure)

5. Simulated time (if light source simulation is active)

6.3 Feature Extraction

The feature extraction uses 5x5x3 RGB image patches. The off-line training set was used to determine the eigenvectors of the presented samples. The three eigenvectors having the biggest eigenvalues are used to transform the 5x5x3 image patch into a three dimensional feature vector. The graph of fig. 6.3 shows the estimated reconstruction distance in relation of the number of used eigenvectors. For this figure the reconstruction distance is defined by the Euclidean distance between the original input image patch and the reconstructed image patch after feature reduction. For the experiments

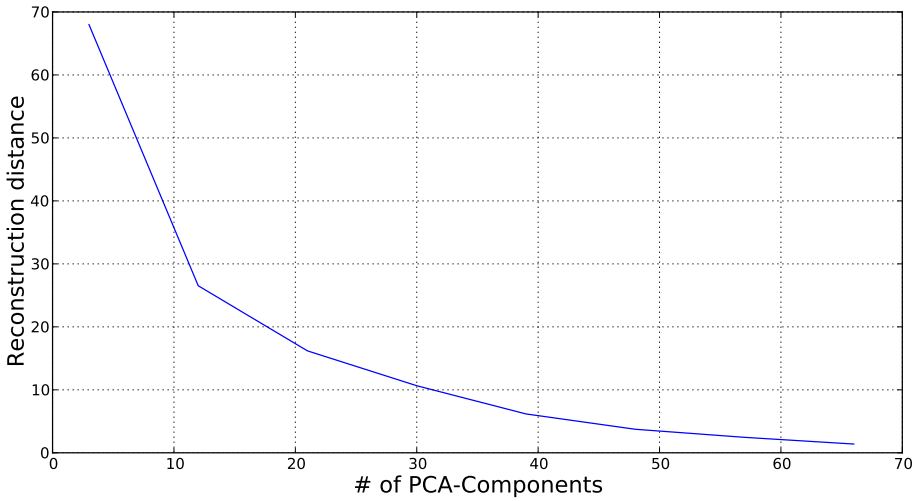


Figure 6.3: Feature reconstruction error - Reconstruction distance in relation to the number of used prototypes. Using an input feature dimensionality of 5x5x3 components leads to a loss-less reconstruction.

performed in this thesis the three eigenvectors having the biggest eigenvalues are used. The principal components have been estimated for the off-line classifier data set. They do not necessarily represent the whole color space that is present in the on-line data set. But this is not a problem in our case as fig. 6.7 shows, there the reconstruction of a scene of the on-line data set can be seen. The yellow flowers show clearly visible reconstruction artifacts that are caused by the fact that the off-line data sets do not

contain this texture. But nevertheless they remain distinguishable. It can be seen that the usage of only three eigenvectors leads to a high reconstruction error. But as fig. 6.4 shows, three eigenvectors provide enough information to reach a sufficient image recovery. The fig. 6.4 shows that it is not possible to recover all structures of the environment that are visible in the selected view by using two eigenvectors. Moreover the usage of additional PCA-components lead to better reconstruction results, but they do not provide necessary information that can be used for the discrimination of class labels.

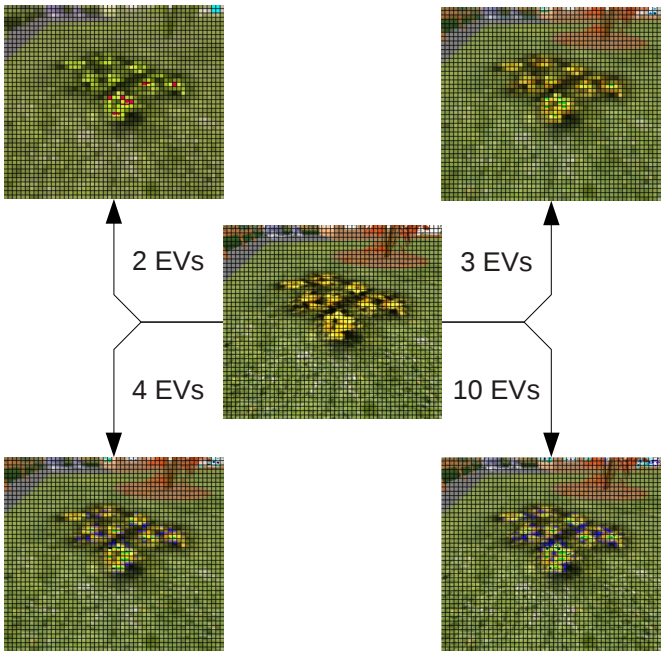


Figure 6.4: Reconstruction visualization - Image reconstructions using different numbers of eigenvectors (having biggest eigenvalues) are shown. For all further experiments the three eigenvectors having the biggest eigenvalues are used. Although the blue artifacts (that can be seen in the figure showing the reconstruction using 10 eigenvectors) seem to have less similarity to the original input image than the those shown in the figures using 2 or 3 eigenvectors their Euclidean distance is smaller for all patches.

6.4 Classifier Combination

The first experiment using the simulation environment targets the comparison between the system using additional context information and a system that has no access to these additional feature dimensions. Fig. 6.5 shows the results of this experiment, it plots the resulting error rate of both systems during the presentation of training samples. The baseline indicates the error rate that was achieved by the training of the off-line classifier. Since all random initialized nodes are initialized with a higher probability for the off-line classifier the baseline also represents the performance of the initial classifier combination system. It can be seen that the system using additional feature dimensions is able to reach a lower error rate than the system not using these dimensions.

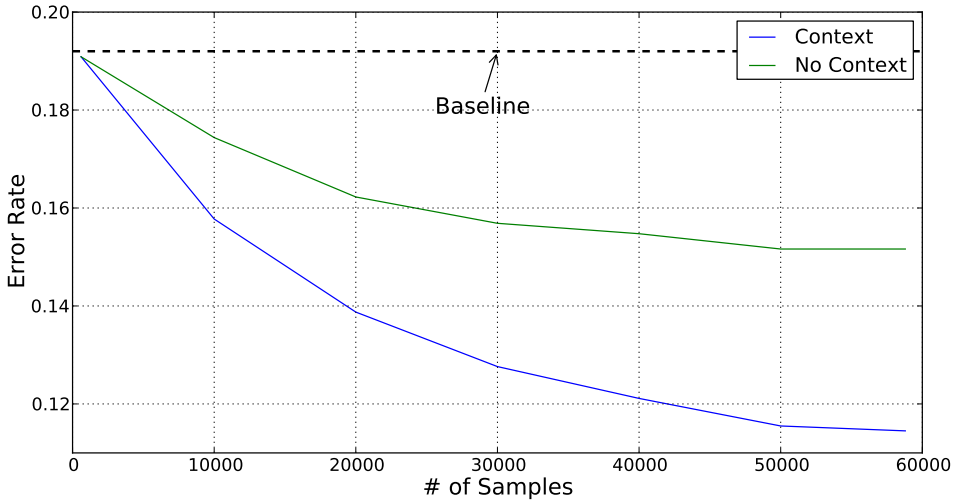


Figure 6.5: System performance - The comparison of the resulting error rates during training of the classifier combination system. The baseline represents the error rate achieved by the off-line classifier training. The performance of the system that uses additional context information exceeds the performance of the system not using additional features. The results represent the average of $n = 10$ runs.

To address the question if the combination of both classifiers gives a certain advantage over the use of just the on-line classifier the combination gain is introduced. It represents a measure of the quality of the classifier combination and is defined by the

performance reached by the combination of the classifiers divided by the performance of the on-line classifier:

$$gain_{combination} = \frac{\text{performance of combination system}}{\text{performance of on-line classifier}} \tag{6.1}$$

This leads to values greater one, if the combination of the classifier leads to a better performance than the on-line classifier on its own. The results during training using context information are shown in fig. 6.6. It can be seen that at the end of the training the performance gain is slightly above the value of 1.1. That means that the combination of the classifier leads to a 10 percent higher classification rate in comparison to the performance of the on-line classifier. Furthermore it is obvious that a considerably higher performance gain is visible at the early stages of the training process, this leads to the conclusion that the combination of the classifier leads to a much faster decline of the error rate. Therefore systems that have only few training samples available, or have to adapt quickly to new situations can benefit from this configuration in a particular way.

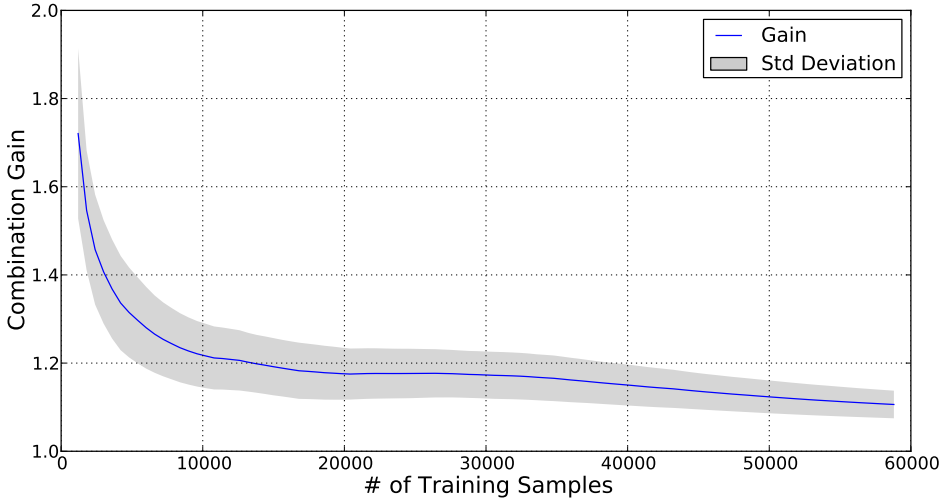


Figure 6.6: Combination gain during training process - The shown combination gain, given by $gain_{combination} = \frac{\text{performance of combination system}}{\text{performanfe of on-line classifier}}$ indicates an advantage of the combined classifiers over the on-line classifier. The results represent the average of $n = 10$ runs.

6.4.1 Parameter optimization

Further experiments target the optimization and evaluation of the sensitivity of parameters. The first experiment addresses the g_{max} parameter.

This parameter defines the number of errors that must occur until a new node is inserted into the on-line classifier network. The results of this experiment are shown in fig. 6.7. The graph given in fig. 6.7(a) reveals that a higher g_{max} threshold leads to a higher deviation of the resulting overall classification rate of the system. Additionally fig. 6.7(b) shows the number of nodes that have been added during the training process. As expected the number of network nodes rises in correlation with a lower g_{max} threshold.

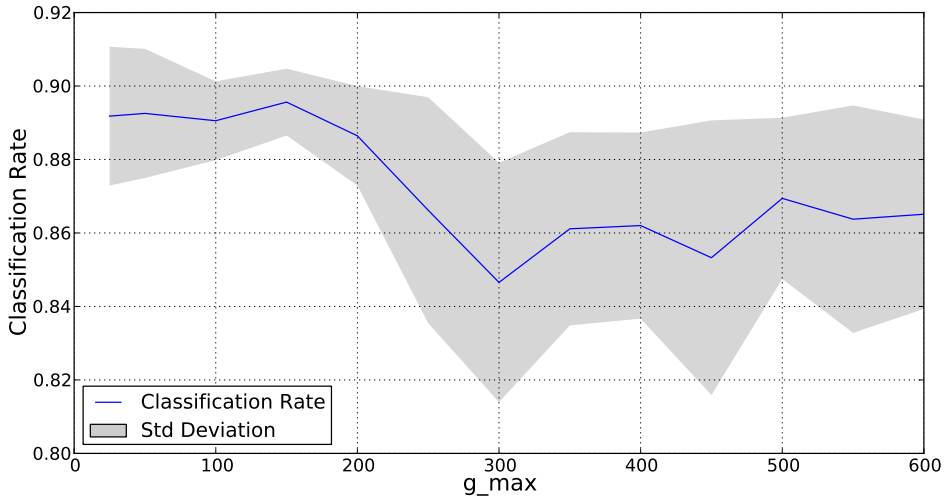
To achieve a compromise between high classification rate having a low deviation and a low number of used prototypes a g_{max} of 300 was chosen for all experiments. A further evaluation to examine one extreme case was done using a modified classifier. This modified classifier inserts a new prototype for each training sample. This would imply to a g_{max} parameter of zero. The resulting classification performance of this configuration reaches 81.2% and is most likely caused by over-fitting. A large number of nodes would lead to a high node density in overlapping regions of the data space, since most classification errors are done in these regions. Moreover large number of nodes also increases the number of probability estimations since for each node the system estimates a probability for the on-line and the off-line classifier. Therefore it requires more samples until a newly inserted node has a reliable performance estimation. This increases the probability of single performance peaks as shown in section 5.2.

The variation of the probability history h is not very critical as it can be seen in fig. 6.8. The probability update rule is defined by:

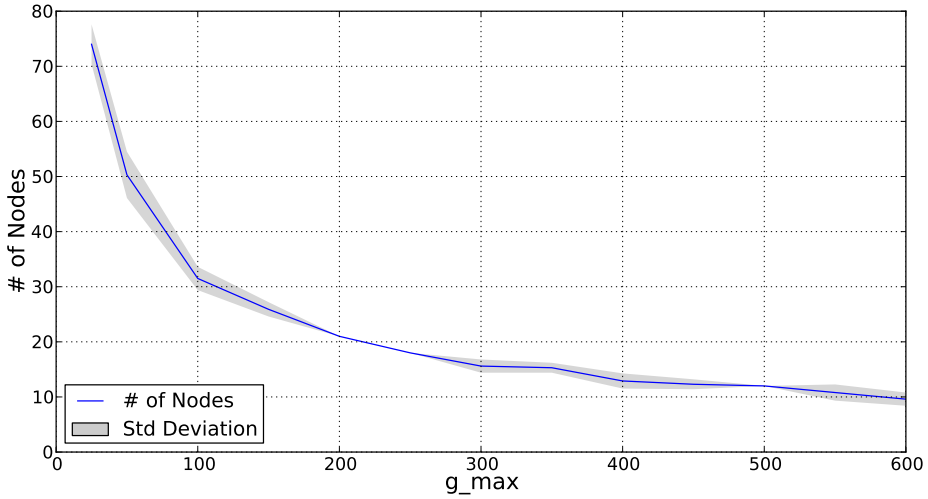
$$prob_{t+1} = prob_t * (1 - \frac{1}{\alpha}) + res * \frac{1}{\alpha} \tag{6.2}$$

The term res is zero if the classification was wrong and one if the classification was correct. The factor α representing the adaptation strength and if defined by

$$\alpha = \begin{cases} 50 & \#\text{samples} < 50 \\ h & \#\text{samples} > h \\ \#\text{samples} & \text{otherwise} \end{cases} \tag{6.3}$$



(a)



(b)

Figure 6.7: Node insertion threshold sensitivity - g_{max} optimization. Figure (a) shows the achieved performance and (b) illustrates the number of used prototypes for the on-line classifier and therefore the number of regions for the performance estimation. The results represent the average of $n = 10$ runs.

Therefore the α factor has a lower bound of 50 and limited by an upper bound of h. The growing α in relation to the number of given samples $\#samples$ for this node, is used as a decay term for the adaptation of the performance estimation (equation 6.2) during training. Small probability history values cause a performance decrease as shown in fig. 6.8, This can be explained by unstable performance estimation since a small α leads to a huge influence of new training observations. This can lead to an alternating selection of the best classifier for one region. On the other hand, the upper bound does not seem to play an important role for the resulting performance.

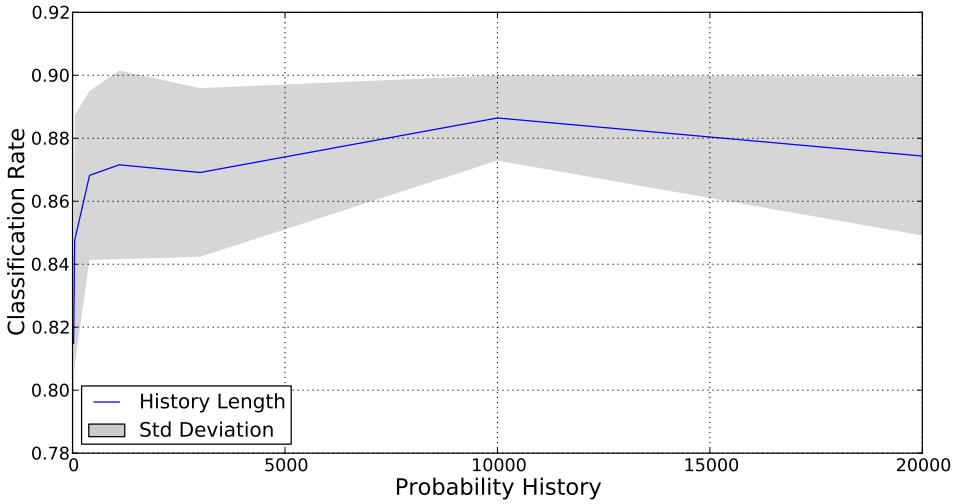


Figure 6.8: Probability history - The resulting performance during variation of the probability history threshold. A low threshold leads to faster adaptation of the probability estimation and lowers the influence of older training samples. The results represent the average of $n = 10$ runs.

6.4.2 Contextualization

The following results present a detailed evaluation of the learning system. Further a comparison between the training using additional context and not using additional context is addressed. Table 6.1 shows a detailed overview of the classification rates.
It can be seen that the resulting error rate of the system using additional context information reaches with 10.44% a better performance in comparison to the system that uses no additional context information. In addition the error rates of the single

Error Rates in %	Context (data set share)	No Context (data set share)
Combined System	10.44(100%)	15.53(100%)
Off-line on all data	19.33(100%)	19.32(100%)
On-line on all data	16.78(100%)	23.04(100%)
Off-line on own assigned data	07.26(81.23%)	16.50(52.94%)
On-line on own assigned data	23.74(18.77%)	20.02(47.06%)

Table 6.1: System Evaluation using context information - Performance comparison of different classifier combination system configurations to evaluate the influence of additional context information. The first column shows the result of the system that uses additional feature dimensions(position, orientation). The second column shows the result of the system not using these additional information. The results represent the average of $n = 10$ runs.

off-line and on-line classifiers can be seen. The given values represent the error rate on the data samples that are assigned to the respective classifier. Moreover the error rate on the complete classifier is given, too. The error rate of the off-line classifier on the whole data set is equal for both systems(19.32%) and represents the baseline, since both system use the same off-line acquired start configuration. The off-line *data set share* and on-line *data set share* entries specify the amount of samples that are processed by the on-line respectively the off-line classifier. They sum up to 100% since there is always one classifier responsible for a given input sample. Comparing the context and non context version of the learning systems reveals that the off-line classifier of the non context version reaches an error rate that is over twice as high (16.5%) as the error rate of the off-line classifier of the system using context information (7.26%), but the system using no context information assigns less samples (52.94% compared to 81.23% if using context) to the off-line classifier. That shows that the samples that are no longer processed by the off-line classifier must be easy cases, otherwise the classification rate of the off-line classifier should decrease. Investigating the performance on all samples shows that the error rate rose from 16.78% up to 23.04%, but the error rate on the assigned samples dropped slightly from 23.74% downto 20.02%. This is caused by the fact that the on-line classifier of the system that uses no context information is

responsible for more easy data samples that where processed in the system using context information by the off-line classifier. To get a detailed view of the performance on easy and difficult image patches, the table 6.2 and table 6.3 give a more detailed view of the evaluation.

Error Rates in %	Context (data set share)	No Context (data set share)
Off-line on confusable objects	53.45(24.29%)	69.20(39.97%)
On-line on confusable objects	07.99(75.71%)	11.65(60.03%)

Table 6.2: Detailed system evaluation on confusable objects - The table compares the error rates of the classifier combination systems using additional context information(left) and not using context information(right). The given results are related to a sub data set only containing confusable objects. Those objects that have similar visual appearance for different class affiliations. The results represent the average of $n = 10$ runs.

Error Rates in %	Context (data set share)	No Context (data set share)
Off-line on non confusable objects	04.92(94.23%)	07.87(55.90%)
On-line on non confusable objects	79.48(05.77%)	29.13(44.10%)

Table 6.3: Detailed system evaluation on non confusable objects - The table compares the error rates of the classifier combination systems using additional context information(left) and not using context information(right). The given results are related to a sub data set only containing non confusable objects. Those objects that are already mainly present in the off-line data set. The results represent the average of $n = 10$ runs.

The error rates on the data set were computed for two different sub data sets. The sub data set of confusable objects contains all patches that are hard to classify. This includes the foliage, the heap of leafs, the wooden veranda, the wooden wheel barrow

and the wooden garden fence. Those objects represent new objects introduced only in the on-line training environment and have similar visual appearance for different class affiliations. All other objects are represented in a second sub data set containing the non confusable objects. Each table shows the performance achieved by the on-line and off-line classifier on the samples that are assigned to the classifiers by the selection process driven by the performance estimation. Additionally the on-line *data set share* and off-line *data set share* specify the rate of samples that are processed by the on-line and off-line classifier again. Analyzing these results for the system using context information reveals that the vast majority (75.71%) of samples belonging to the confusable class are processed by the on-line classifier. In comparison to the non confusable samples that are mostly processed by the off-line classifier (94.23%). Moreover it can be seen that a specialization of the classifiers occurs. The off-line classifier reaches an error rate of 4.92% and therefore a low error rate for the non confusable objects. But it reaches a high error rate error rate of 53.54% for the data set of the confusable objects. This is reasonable since most non confusable objects are also present in the data set used for the training of the off-line classifier. The on-line classifier shows a specialization on the confusable objects, analyzing the error rates shows that the on-line classifier reaches a low error rate of 7.99% for the confusable objects but has an unacceptable error rate of 79.48% for classifying non confusable objects. Comparing these results with the values given by the evaluation of the system not using additional context information, one can see that they show much less classifier specialization and predominantly higher error rates. One exception is the error rate of the on-line classifier of non confusable objects, but this results in a larger number of training views of the non confusable class that were presented to the on-line classifier.

Table 6.4 shows the matrix Λ used for the general distance metric. It shows the mean of 10 training results. The diagonal determines the weights for each component and the other values are related to weights between correlated dimensions. It can be seen that the second and third principal component, the x and y position and the sine function of the rotation angle seem to be most interesting for classification. On the contrary the first PCA component and the cosine function of the rotation do not seem to play such a big role. Moreover it can be seen that the second and third dimensions have a strong correlation. To get an impression of variability of these values, fig. 6.9(a-d) show visualizations of single runs. Fig. 6.9(e) shows the visualization of the mean matrix Λ

	PCA Component						
	#1	#2	#3	X	Y	$sin(\theta)$	$cos(\theta)$
#1	0.0888	-0.03389	0.01395	0.01237	-0.01572	0.03044	-0.02303
#2	-0.03389	0.25081	-0.11559	-0.00865	0.02916	-0.03971	0.02268
#3	0.01395	-0.11559	0.207	0.00108	0.01718	-0.0125	-0.01827
X	0.01237	-0.00865	0.00108	0.1331	-0.00049	0.04328	-0.02475
Y	-0.01572	0.02916	0.01718	-0.00049	0.11673	-0.05541	0.02011
$sin(\theta)$	0.03044	-0.03971	-0.0125	0.04328	-0.05541	0.13783	-0.02796
$cos(\theta)$	-0.02303	0.02268	-0.01827	-0.02475	0.02011	-0.02796	0.06569

Table 6.4: GMLVQ Λ-Matrix used for distance estimation - This table shows the matrix used for the similarity measure of the GMLVQ learning method. The distance is defined by $d^{\Lambda}(\omega, \xi) = (\xi - \omega)^T \Lambda (\xi - \omega)$ as given by Schneider et al. [36]. The table caption indicates the content of each dimension starting with the three eigenvector components having the biggest eigenvalues, the position coded by the x-y-position and the orientation coded by the *sin* and *cos* of the robot angle.

given in table 6.4. This figure shows that there is no clearly visible correlation between other feature dimensions than the first and second PCA component.

(a) **(b)** **(c)** **(d)** **(e)**

Figure 6.9: Λ matrix visualization - The sub-figures (a)-(d) show single Λ matrix configurations at the end of the training process of the system using additional context information. The sub-figure (e) shows the average weights of $n = 10$ runs. The labels for the dimensions correlate to those shown in fig. 6.4

6.4.3 Comparison to solely on-line training

One hint for the advantage of the classifier combination system is the performance gain as introduced in section 6.4. The fig. 6.10 shows a comparison between the classifier

combination system and the training using just the on-line classifier. It can be seen that the performance of the combination system has a lower starting point due to the baseline of the off-line classifier. With a large number of training data it is possible for the on-line classifier to reach the performance of the combination system. If this is the case the on-line classifier represents the whole data set and the off-line classifier is not needed any more. For these experiments the same classifier type was chosen for on-line and off-line classification. Using different classifiers could lead to a performance that is not reachable by just one classifier, since the different architectures may have better abilities to represent different sub data sets. It can be seen that the combination of the classifier has clearly advantages in terms of learning speed.

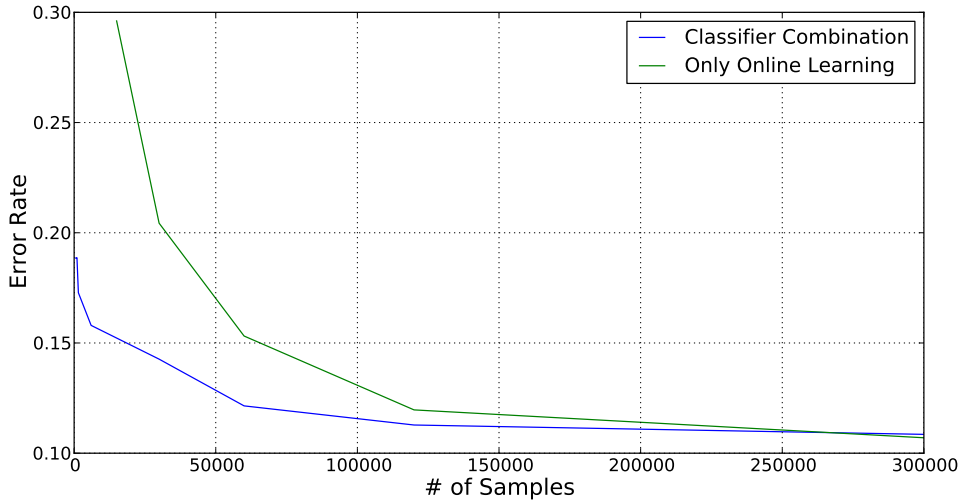


Figure 6.10: Combination system in comparison to on-line classifier - The graph shows the comparison of the classifier-combination system and a single on-line classifier. It can be seen that the on-line classifier is able to reach the same error rate as the classifier combination system if many training samples can be used. The result given here show the average of $n = 10$ runs.

6.4.4 Visualization of classification results

The following figures 6.11-6.12 show visualizations of typical results acquired by the classifier combination systems either using or not using context information. And show

the benefits of the system that uses additional context. The top left sub image shows the given input image, the top right image shows the ground truth class affiliation of the generated 5x5 image patches.

The lower image represent the classifier results, they do not show both the results for exactly the same patches since the patches are selected randomly for each new run. The lower left image visualizes the results of the classifier using context information and the lower right image shows the classification results of the classifier system not using context. The results are visualized by miniature pie charts. The pie represents the probability estimation for each class. Each of the three classes is coded by an unique color. Red represents the traversable class, green the non traversable objects and blue indicates ignorable objects. The probabilities are estimated by the extended Sum-Rule based probability estimation presented in section 3.1.1.

All figures show that the system that uses no context information is not able to deal with patches that belong to the class of ignorable objects. In fig. 6.11(top) and fig. 6.11(bottom) the foliage underneath the tree is classified almost completely as a non traversable object(green) by the system that uses no context information, in comparison to the system that uses context information that is able to assign the correct label traversable(blue) to these patches. The same comparison for fig. 6.12(top) shows the same behavior for the patches that represent the wooden veranda and fig. 6.12(bottom) shows patches of the foliage and the wooden veranda.

The system that uses context information is not able to discriminate the orange leafs of the trees and the orange leafs of the foliage on the ground in all cases. As shown in fig. 6.11(top) the result of the orange leafs of the tree are labeled correctly as obstacle(green) and in fig. 6.11(bottom) these leafs are labeled as ignorable objects(blue). It can be seen that the classification errors related to the leafs are caused by the similar looking foliage underneath the trees. According to the viewing angle the leafs change their illumination so that they can get confused with the traversable leafs on the ground. Both system using either context or not are able to classify nearly all of the patches that cover the traversable ground plane correctly.

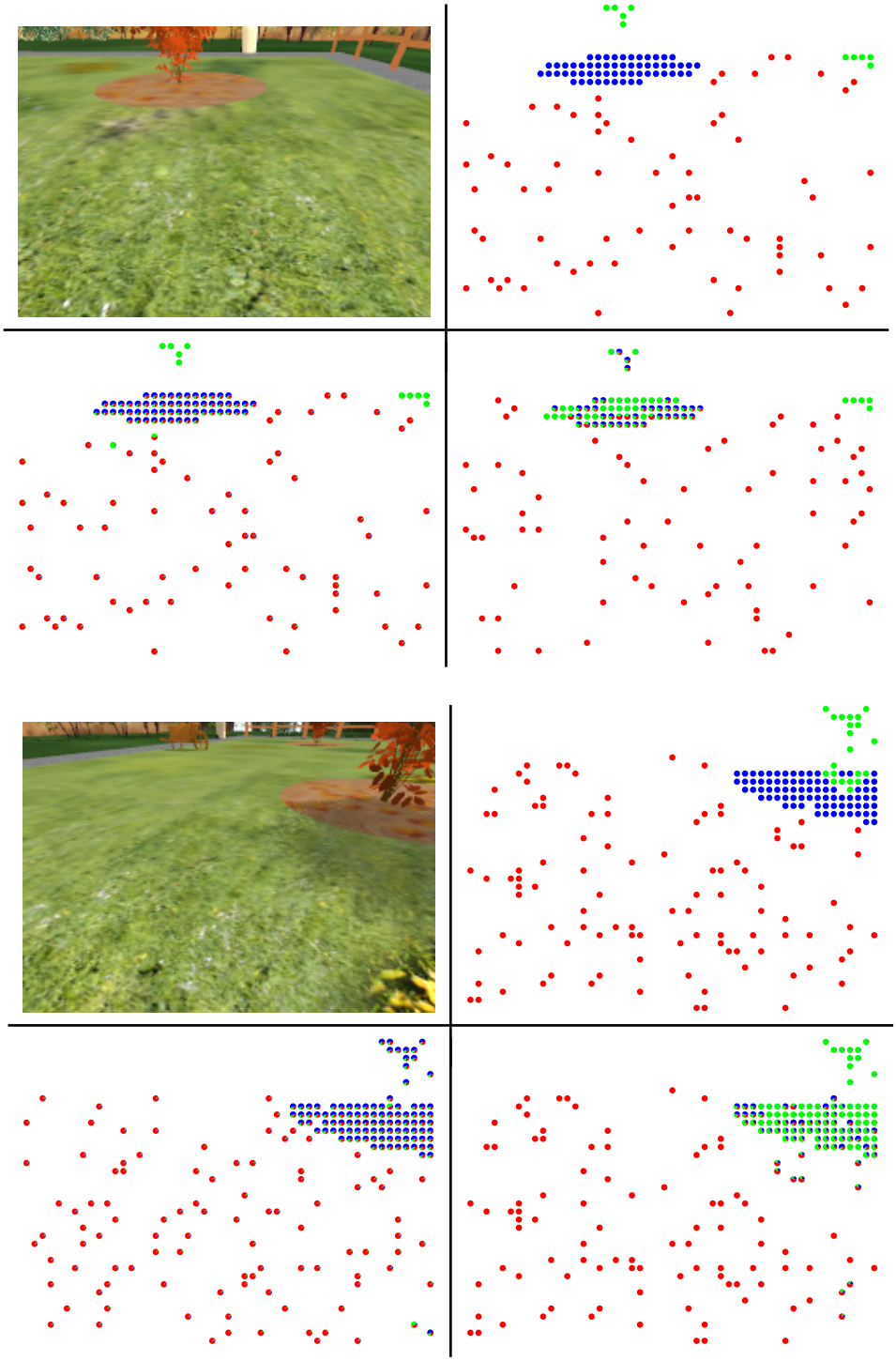


Figure 6.11: Visualization of the classification result: View 160(top) & view 2980(bottom) - The input image is shown in the top left, the ground truth can be seen in the top right, the lower images represent the classification results of the classifier systems using context information (left) and using only visual features (right). For discussion see section 6.4.4.

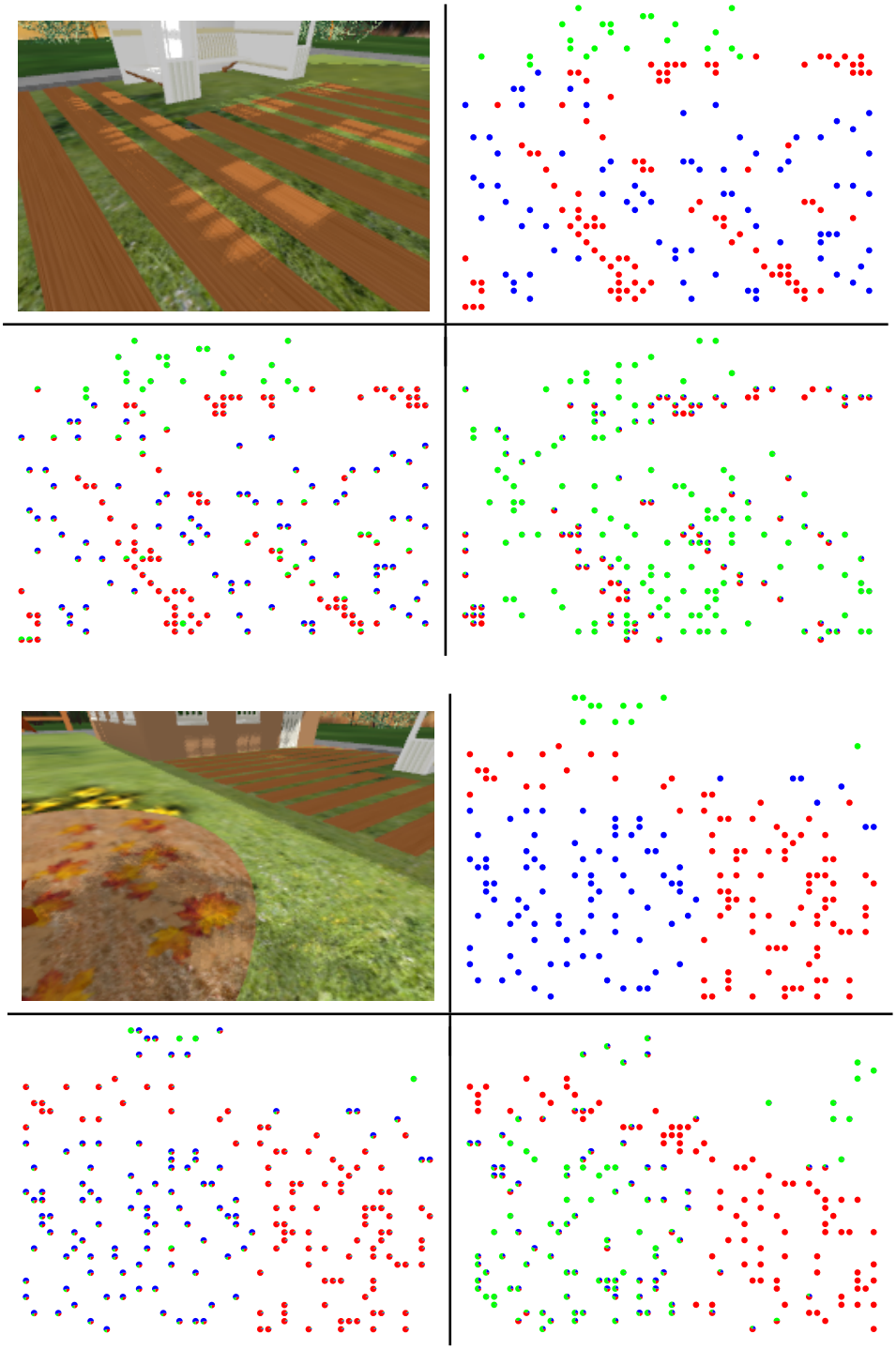


Figure 6.12: Visualization of the classification result: View 140(top) & view 20(bottom) - The input image is shown in the top left, the ground truth can be seen in the top right, the lower images represent the classification results of the classifier systems using context information (left) and using only visual features (right). For discussion see section 6.4.4.

6.4.5 Prototype visualization

Since the used classifier is prototype based it is possible to visualize the given network state in a certain way. Due to the fact that only the state of the on-line classifier is shown, nodes that have a lower estimated performance than the off-line classifier do not necessarily represent the data set. This is caused by proposed learning method explained in section 3.1.1. Training samples are only used for training of the on-line classifier if this classifier is responsible for the samples, a performance threshold is not reached or an classification error occurred. Therefore the components representing the first three eigenvectors can be used to reconstruct the visual appearance of a given prototype. Moreover the prototype dimensions represent a position and orientation of a given node. Based on this data the fig. 6.14 was generated. Fig. 6.13 shows a detailed view of two prototypes for the purpose of explanation. All prototypes given by the on-line classifier are mapped into a top down view of the environment used for on-line training. Each filled circle indicates a prototype position and the related box holds extra information related to that prototype. This information includes:

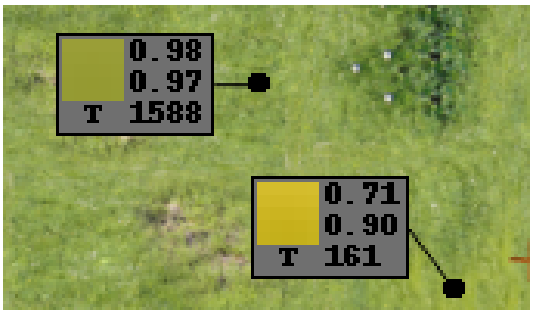


Figure 6.13: Explanation of classifier visualization - This image shows a part of the visualization of the internal state of the on-line classifier shown in fig. 6.14.

1. Visual reconstruction
2. Class label of the given prototype
3. Probability estimation of the off-line classifier
4. Probability of the on-line classifier.
5. Number of training samples assigned to this prototype

The acronyms for the label affiliation are I for ignorable, T for traversable and O for non traversable objects. The upper left node visualized in fig. 6.13 shows a node that

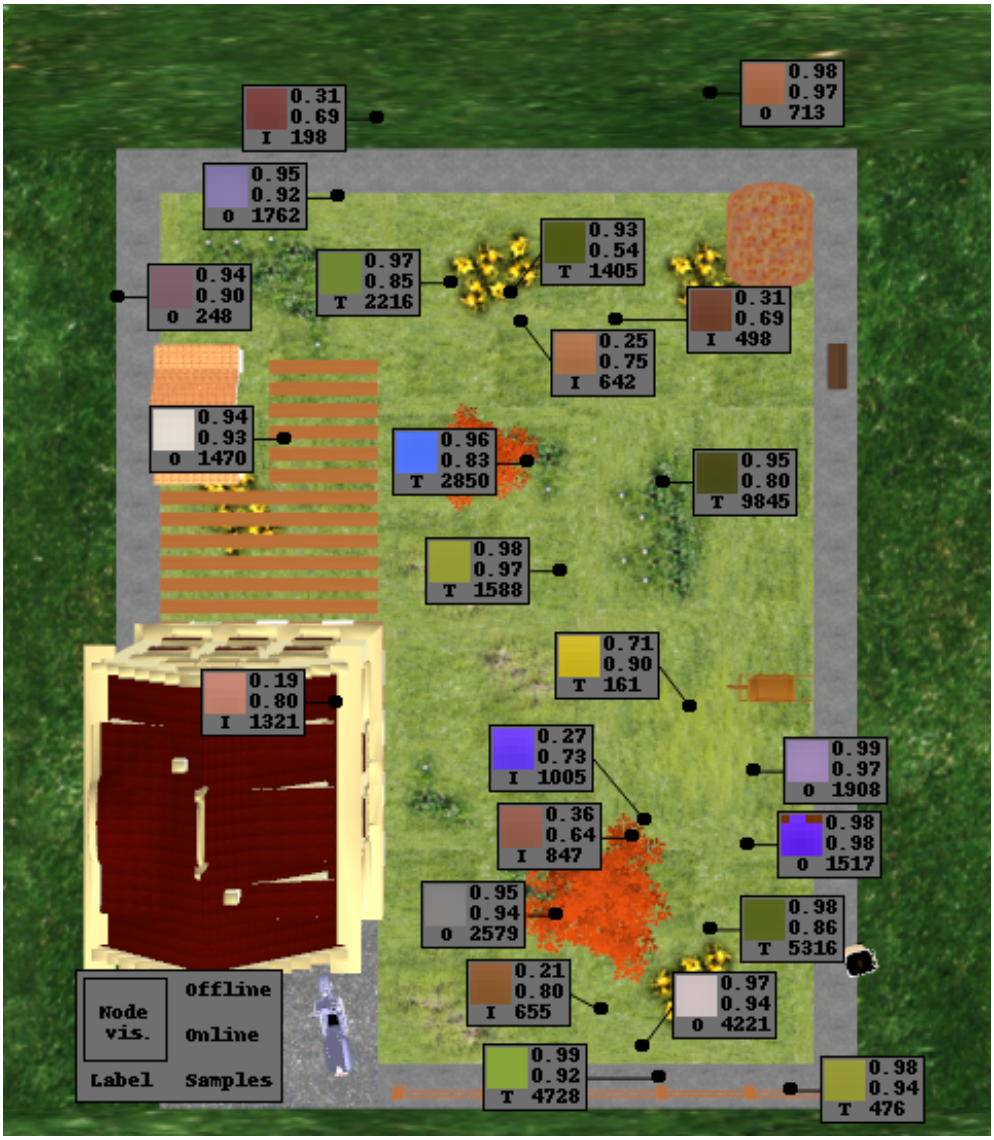


Figure 6.14: Node visualization of the on-line classifier - The visualized nodes represent the state of the on-line classifier embedded in the classifier combination system that uses additional context information. Each dot indicates the position of one prototype. The related box shows additional information. This includes the visual appearance of the node that was reconstructed using the principal components. The class affiliation that is shown below the node visualization (O-Obstacle, T-Traversable or I-Ignorable). The estimated probabilities for the on-line and off-line classifier are shown in the top right corner of the box. The number of samples that are assigned to the node during the training process are shown in the lower right corner.

prefers the off-line classifier. It can be seen that it represents the traversable ground, although it is not selected for classification. The lower right node represents yellow colored traversable objects. It can be seen that the on-line classifier reaches a much higher performance than the off-line classifier. That indicates that this node represents patches that beong to objects that where not available in the off-line classifier. Moreover the node has a much lower probability of appearance than samples that belong to the upper left node. It is obvious that this node represents the yellow flowers that can be seen in fig. 6.14. But such an analysis is not possible for all prototypes. Since nodes that prefer the off-line classifier do not need to represent the data, there are more than just the visual feature dimensions that effect the distance estimation and the feature weighting changes the equal importance of all dimensions.

6.4.6 More complex data set and additional features

The following experiments target a more complex data set. Therefore the scene illumination was changed in relation to a simulated time value. This time value was also used as an additional context dimension. The rest of the simulation environment was kept unchanged. Tab. 6.5 shows the classification results in the same way as table 6.1 for the former experiments targeting additional context information.
The comparison of the resulting error rates at the end of the on-line training to the baseline given with 33.9%, achieved by the off-line classifier training, reveals that the system was able to enhance the classification results for all three experiment configurations. But the use of additional feature dimensions leads to a degradation of the data representation in this case, since the error rate of the system using no additional context dimensions reaches with 20.82% the lowest value.
The comparison between the results of the experiment using the same context information as in the former experiments and the experiment using time information as an additional feature dimension shows that this context has the ability to improve the classification. But it seems that most context dimensions do not provide useful information for this scenario. Therefore the adaptation of the distances leads to a more complex optimization process that results in a lower performance compared to the system not using context information.
Tab. 6.6 and tab. 6.7 show the results of the on-line and off-line classifier on the two sub data sets representing the confusable and non confusable objects as previously

introduced in section 6.4.2. It can be seen that for the more difficult data set there is less specialization of the classifiers. This is caused by the fact that the off-line classifier is not able to handle data patches that do have a much higher variance regarding the visual appearance. This is caused because the global lightning shifts its color to red as shown in fig. 6.15.
Regarding tab. 6.6 and tab. 6.7 it can be seen that especially the off-line classifier processes for all three configurations a much smaller part of the non confusable objects. But it can also be seen that the off-line classifier yields better results on the non confusable objects although the on-line classifier yields better results on the confusable objects.

Error Rates in % (Data set share)	No Context	Context	+ Time
Combined System	20.82(100%)	25.16(100%)	24.29(100%)
Off-line on all	33.92(100%)	33.89(100%)	33.88(100%)
On-line on all	22.62(100%)	27.27(100%)	26.26(100%)
Off-line on own assigned data	16.17(38.79%)	13.68(30.52%)	14.57(34.45%)
On-line on own assigned data	24.26(61.21%)	30.39(69.48%)	30.34(65.55%)

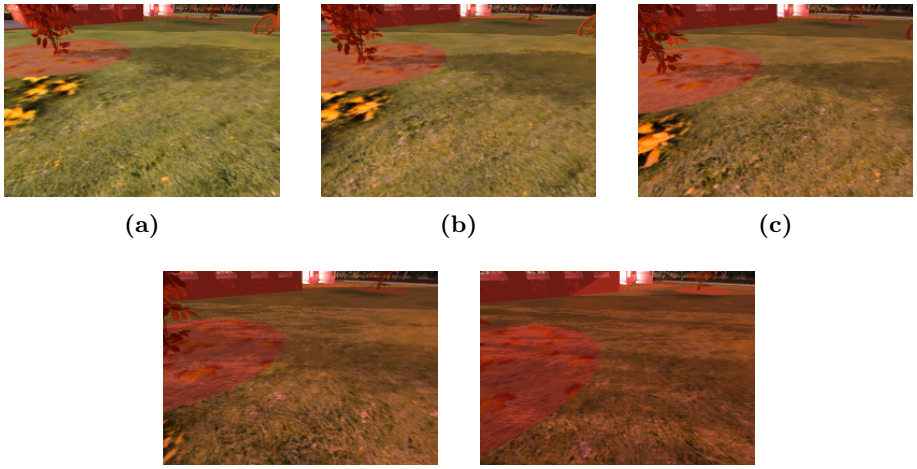
Table 6.5: System Evaluation using context information and time simulation - Performance comparison of different classifier combination system configurations to evaluate the influence of additional context information. The first column shows the result of the system that uses the same additional feature dimensions than in the previous experiments(position, orientation). The second column shows the result of the system using an additional dimension that represents the current time. The results represent the average of $n = 10$ runs.

Error Rate in % (Data set share)	No Context	Context	+ Time
Off-line on confusable objects	79.48(30.50%)	63.90(23.75%)	66.67(25.82%)
On-line on confusable objects	49.46(69.50%)	40.40(76.25%)	39.54(74.18%)

Table 6.6: Detailed system evaluation on non confusable objects - The table compares the error rates of the classifier combination systems using position and rotation as context information(left) and using additional time information(right). The given results are related to a sub data set only containing non confusable objects. Those objects that are already mainly present in the off-line data set. The results represent the average of $n = 10$ runs.

Error Rate in % (Data set share)	No Context	Context	+ Time
Off-line on non confusable objects	07.33(40.29%)	06.87(31.73%)	07.54(36.00%)
On-line on non confusable objects	19.37(59.71%)	28.57(68.27%)	28.55(64.00%)

Table 6.7: Detailed system evaluation on non confusable objects - The table compares the error rates of the classifier combination systems using position and rotation as context information(left) and using additional time information(right). The given results are related to a sub data set only containing non confusable objects. Those objects that are already mainly present in the off-line data set. The results represent the average of $n = 10$ runs.



(a) (b) (c)

(d) (e)

Figure 6.15: Simulation environment using time information - The sub-figures (a)-(e) show the effect of the time simulation. They illustrate the higher texture variance that shifts to red.

6.4.7 Rejection rate

Since the classifier system is able to generate probability estimates for the classifier results, a threshold could be used to reject unsure classification results. Due to the fact that this system uses small image patches and the proposed scenario would use an on-line system with a continuous stream of images, a majority voting of patches that belong to a given region could be used for a more stable classification. Therefore it would not be necessary to have a classification result for all image patches. To evaluate the effect of a probability threshold fig. 6.16 shows the probability distribution of all classified patches. The visualization in fig. 6.16(a) shows the probability distribution of the correctly classified patches and fig. 6.16(b) shows the probability distribution of the wrongly classified patches.

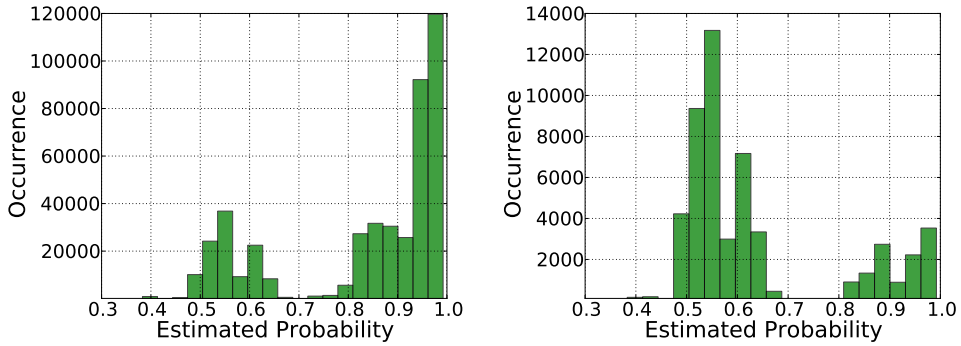


Figure 6.16: Distribution of the estimated probability - Both histograms show the probability distribution of the classifier combination system. The left figure (a) shows the estimated probabilities occurring on correct labeled patches. The histogram on the right (b) shows the estimated probability values of patches that are not labeled correctly.

It can be seen that the estimated probabilities of the correct classified patches have a higher density on higher estimated probabilities. However, the wrongly classified patches lead to probability estimates that have mainly lower estimated values. The impact of a patch rejection based on a probability threshold can be seen in fig. 6.17. This figure shows the resulting classification performance in relation to a patch rejection rate. For a rejection rate of zero the resulting error rate complies the error rate given

by the results in section 6.4. By increasing the probability threshold the number of rejected patches rises. That leads to a decrease of the error rate as long as more wrong classified patches are ignored by the used threshold than correct classified patches. It can be seen that a rejection rate of 0.3 improves the error rate roughly by factor three. This shows that the classification system is able to benefit from patch rejection as long as a patch rejection is feasible for the given task.

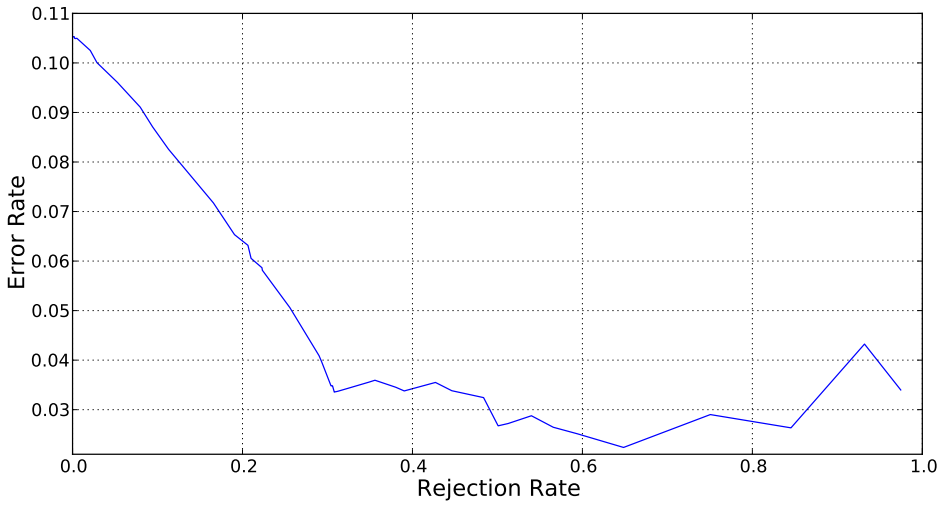


Figure 6.17: Error rate in relation to the rate of rejected patches - The rejection rate is given by a probability threshold that defines a minimum estimated probability for accepted patches

6.5 Conclusion

The experiments using data generated in an artificial environment confirmed the results of chapter 5. Additional context information was able to improve the classification result. It could be seen that especially the early stages of the training process are able to benefit from the use of a classifier combination system. Further it was shown that this method is interesting for systems that do not have access to large training data sets, so that interactive on-line systems can use the advantages of a faster learning process. Since the architectures of the on-line and off-line classifier are based on the same classifier architecture, the on-line classifier is able to reach the same performance

as the combination system. But this requires much more training data. Considering the Λ matrix of the GMLVQ learning system it can be seen that except a correlation between the second and third dimension no further data dependency can be detected. The detailed comparison between confusable objects and those that have a much simpler structure reveals that the combination system leads to a classifier specialization. In comparison to the off-line classifier that deals with the simpler patches the on-line classifier handles those patches that have similar visual appearance but different class affiliations. Since an on-line system usually detects multiple small image patches of one object and that it is very likely that multiple image frames show the same objects, the idea of a patch rejection mechanism arose. The evaluation of the estimated probabilities showed that the estimated probabilities of the wrongly classified patches have a higher density on lower values while the correct classified patches have mostly high estimated probabilities. The introduction of a probability threshold used for class rejection was able to lower the error rate to one-third by rejecting around one-third of all input patches.

7

Discussion

For the evaluation of the proposed classification system, different virtual 3D environments were designed to generate training and test data sets. The modified simulator is able to generate random paths along the traversable area of these environments. This allows the creation of a more realistic on-line learning scenario. Due to the implementation of an experimental test framework, different experimental configurations could be realized easily. Therefore different classification methods and experimental setups were evaluated. Moreover different experiments using artificial data sets have been used to evaluate the system behavior. This helps to understand and visualize the working principle of the used classifier selection system.

The experiments showed that the implemented classifier is able to handle additional feature dimensions due to its weighting strategy. This allows the proposed system to use a combined feature space and to select the appropriate features. If the used feature dimension is much larger than the seven to eight dimensions used for the experiments that are presented here, using the GRLVQ instead of the GMLVQ can help to reduce the number of required training samples. The GRLVQ requires a higher amount of samples since it has to adapt the squared Λ-matrix in comparison to the GRLVQ that only has to adapt on weight for each dimension. By a feature space pruning that is based on the weights λ_i of the feature dimensions, as proposed by Kietzmann et al. [40], it is possible to reach a further reduction of the computational cost and the problem complexity.

It was shown that it is possible by the use of additional context information to improve the system performance. But it was also evaluated that this is not possible for all

cases, since for an experiment that uses an increased scene complexity with additionally higher similarities between the different classes, it was not possible to show benefits of the usage of additional context information. This could be caused by the fact that the problem complexity is far too high for the used classifier, since this data set contains many overlapping classes. Such a complex scenario may require the use of an enhanced feature weighting process or an additional feature pruning.

The combination of the off-line and the on-line classifier is realized by an external probability estimation module. Therefore the system does not rely on the classifier type. This is a significant advantage in comparison to typical combination methods that require comparable classification techniques and comparable input spaces. Since the probability estimation uses the Voronoi cells of the on-line classifier to realize a partitioning of the input space, the probability estimation is adapted to the local problem complexity of the input space. But the incremental insertion of new Voronoi cells of the on-line classifier can lead to short error peaks. These peaks are triggered by the probability estimation for newly inserted cells, since the system can select the wrong classifier for a cell until not enough samples for a reliable classifier selection were presented. The estimated probabilities allow to use additional post-processing steps to increase the quality of the classification results. This is shown by the usage of a probability threshold. The threshold allows the system to reject sample inputs that do not have a high confidence. This ability allows the system in this case to achieve a three times lower error rate compared to the system not using any kind of post-processing. Over this, the proposed classification system shows that a combination of specialized classifiers can be beneficial for the given scenario. The specialization of the classifiers requires less training time compared to a system that is only using one on-line classifier. This is caused by the fact that the off-line classifier was trained on a data set that shares partial similarities with the used on-line scenario. This is a great advantage for autonomous systems that have to adapt to a changing environment. Since it is very expensive to generate training data for custom environments. One particular application purpose can be a robotic system that is equipped with a factory configuration. The proposed method would allow this robotic system to adapt its internal factory configuration fast to the requirements defined by the consumer.

The experiments showed that the combination system is not able to deal with overlapping data distributions, since the incremental node insertion rule adds new nodes

until all classification errors are resolved. This leads, in the case of overlapping data distributions, to an unnecessary insertion of nodes since the system will never be able to resolve all classification errors.

7.1 Further Work

Further investigations could target the combination of more then two classifiers. According to the problem complexity the usage of multiple classifier architectures could be beneficial.

The used initialization for the probability estimation of newly inserted nodes can lead to short error peaks that are caused by wrong classifier selections. One way to avoid those peaks could be to use an initialization that involves the neighboring probability estimations. Another way could be to ignore this newly inserted region for the generation of the classifier result until a reliable probability estimation of this node is available.

To address the problem of unlimited node insertions for overlapping data, one could consider the pre-condition for the activation of incremental node insertion. If additional node insertion does not lead to a further improvement of the error rate it is most likely that there are sufficient nodes available.

Appendix A

Data Sets

Table A.1: Database visualization of the simulated environment used for off-line learning.

Table A.2: Database visualization of the simulated environment used for on-line learning.

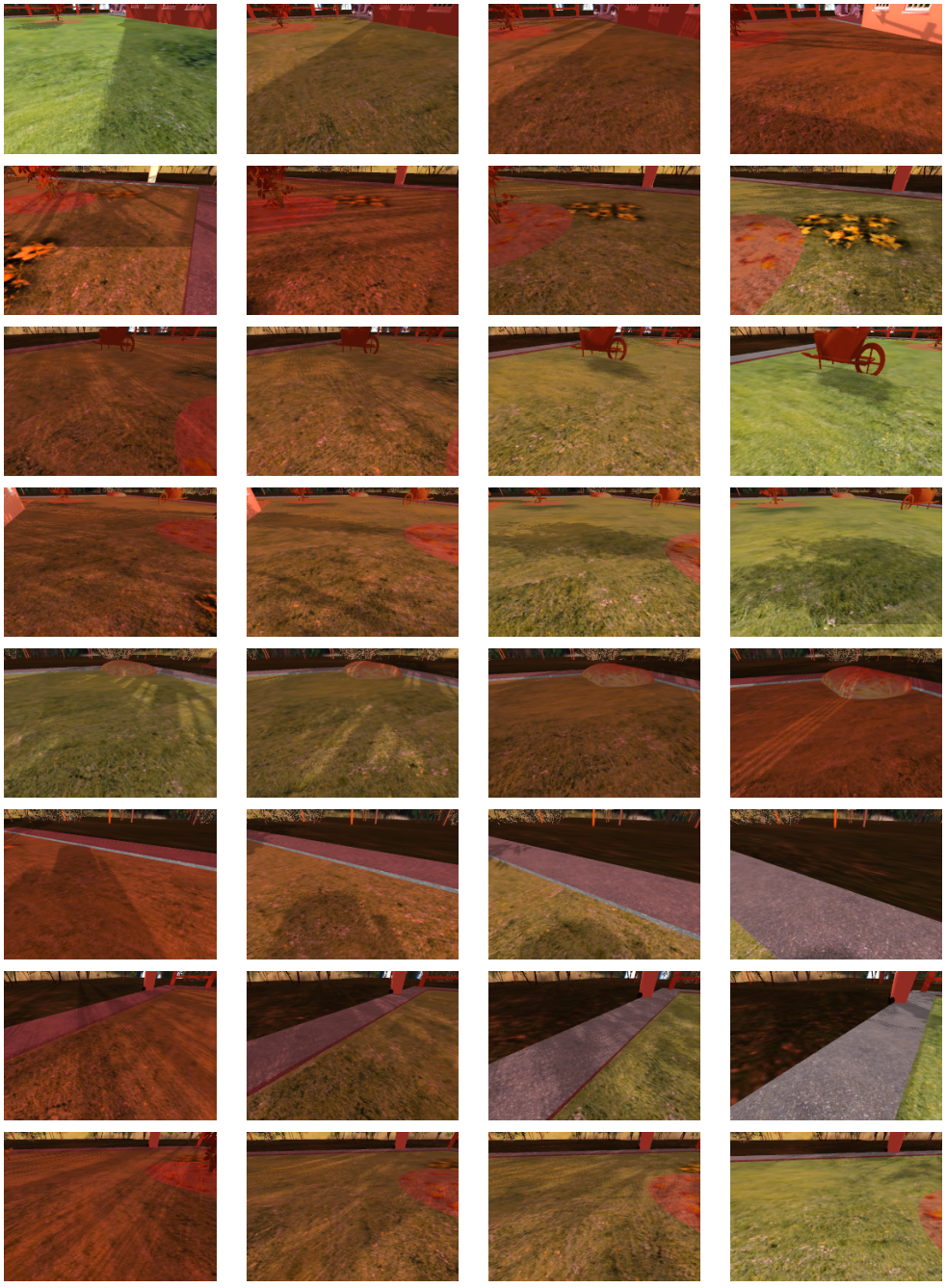


Table A.3: Database visualization of the simulated environment used for on-line learning including additional time simulation.

References

[1] David G. Lowe. **Distinctive Image Features from Scale-Invariant Keypoints**. *International Journal of Computer Vision*, **60**(2):91, November 2004. 1

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. **Learning internal representations by error propagation**. In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing vol. 1*, chapter 8, pages 318–362. MIT Press, Cambridge, Mass., 1988. 5

[3] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. **A Training Algorithm for Optimal Margin Classifiers**. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992. 5, 20

[4] Stephan Kirstein, Heiko Wersing, Horst-Michael Gross, and Edgar Körner. **A life-long learning vector quantization approach for interactive learning of multiple categories**. *Neural Networks*, **28**:90–105, 2012. 5

[5] Andrew Kachites McCallum and Kamal Nigam. **Employing EM in pool-based active learning for text classification**. In *In Proceedings of the 15th International Conference on Machine Learning*, pages 350–358. Morgan Kaufmann, 1998. 5, 6

[6] Paul Viola and Michael Jones. **Rapid Object Detection using a Boosted Cascade of Simple Features**. In *Computer Vision and Pattern Recogntion*, pages 511–518, 2001. 5

[7] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer Series in Information Sciences, Springer-Verlag, third edition, 1989. 6

[8] T. Martinetz and K. Schulten. **A "neural-gas" network learns topologies**. *Artificial neural networks*, **1**:397–402, 1991. 6

[9] Bernd Fritzke. **A Growing Neural Gas Network Learns Topologies.** In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *NIPS*, pages 625–632. MIT Press, 1994. 6

[10] Stephan Kirstein, Alexander Denecke, Stephan Hasler, Heiko Wersing, Horst-Michael Gross, and Edgar Körner. **A Vision Architecture for Unconstrained and Incremental Learning of Multiple Categories**. *Memetic Computing*, **1**(4):291–304, 2009. 6

[11] Fabio Gagliardi Cozman, Ira Cohen, Marcelo Cesar Cirelo, and Escola Politécnica. **Semi-Supervised Learning of Mixture Models**. In *ICML-03, 20th International Conference on Machine Learning*, pages 99–106, 2003. 7

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin. **Maximum likelihood from incomplete data via the EM algorithm**. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, **39**(1):1–38, 1977. 7

[13] Xiaojin Zhu. **Semi-Supervised Learning Literature Survey**. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005. 7

[14] Stephan Kirstein Heiko Wersing and Edgar Körner. **Towards autonomous bootstrapping for life-long learning categorization tasks**. In *Proc. Int. Joint Conf. on Neur. Networks.*, IJCNN (2010), Barcelona, 2010. 7

[15] Daniele Calisi, Alessandro Farinelli, Giorgio Grisetti, Luca Iocchi, Daniele Nardi, Stefano Pellegrini, Gian Diego Tipaldi, and Vittorio A. Ziparo. **Contextualization in Mobile Robots**. In *Proceedings of the ICRA '07 Workshop on Semantic Information in Robotics*, Rome, Italy, 2007. 7

[16] S. Hese and C. Schmullius. **Object context information for advanced forest change classification strategies**. In *OBIA06*, 2006. 8

[17] Isabelle Guyon and Andre Elisseeff. **An introduction to variable and feature selection**. *J. Mach. Learn. Res.*, **3**:1157–1182, March 2003. 8

[18] Stephan Kirstein. **Feature Selection and Weighting Techniques**. Technical report, HRI-EU, 2011. 8

[19] George Forman. **An extensive empirical study of feature selection metrics for text classification**. *J. Mach. Learn. Res.*, **3**:1289–1305, March 2003. 9

[20] Kenji Kira and Larry A. Rendell. **The feature selection problem: traditional methods and a new algorithm**. In *Proceedings of the tenth national conference on Artificial intelligence*, AAAI'92, pages 129–134. AAAI Press, 1992. 9

[21] Ron Kohavi and George H. John. **Wrappers for Feature Subset Selection**. *Artificial Intelligence*, **97**(1-2):273–324, 1997. 9, 10

[22] Naftali Tishby, Fernando C. Pereira, and William Bialek. **The Information Bottleneck Method**. In *The 37th annual Allerton Conference on Communication, Control, and Computing*, pages 368–377, 1999. 9

[23] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998. 10

[24] Barbara Hammer and Thomas Villmann. **Generalized relevance learning vector quantization**. *Neural Networks*, **15**(8–9):1059–1068, 2002. 11, 22

[25] Tin Kam Ho. *Multiple Classifier Combination: Lessons and Next Steps*, **47** of *Series in Machine Perception and Artificial Intelligence*, chapter 7, pages 171–198. World Scientific, 2001. 12

[26] Marie-Jean-Antoine-Nicola de Caritat Marquis de Condorcet. *Essai sur l'application de l'analyse à la probabilité des decisions rendues à la pluralité des voix*. L'Imprimerie Royale, Paris. Chelsea Publishing Company, 1785. 12

[27] Robert P. W. Duin. **The combining classifier: to train or not to train?** In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, **2**, pages 765–770 vol.2, 2002. 12

[28] Matti Aksela. **Comparison of classifier selection methods for improving committee performance**. In *Proceedings of the 4th international conference on Multiple classifier systems*, MCS'03, pages 84–93, Berlin, Heidelberg, 2003. Springer-Verlag. 13

[29] Yoav Freund and Robert E. Schapire. **A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting**, 1995. 14

[30] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. **Learning long-range vision for autonomous off-road driving**. *J. Field Robot.*, **26**(2):120–144, February 2009. 14

[31] Dongshin Kim, Jie Sun, Sang Min Oh, James M Rehg, and Aaron F Bobick. **Traversability classification using unsupervised on-line visual learning for outdoor robot navigation**. *Proceedings 2006 IEEE International Conference on Robotics and Automation 2006 ICRA 2006*, **2006**(May):518–525, 2006. 14

[32] Giorgio Giacinto and Fabio Roli. **Methods for Dynamic Classifier Selection**. In *Proceedings of the 10th International Conference on Image Analysis and Processing*, ICIAP '99, pages 659–et seqq., Washington, DC, USA, 1999. IEEE Computer Society. 17

[33] Chih-Chung Chang and Chih-Jen Lin. **LIBSVM: A library for support vector machines**. *ACM Transactions on Intelligent Systems and Technology*, **2**:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 20

[34] S. A. Nene, S. K. Nayar, and H. Murase. **Columbia Object Image Library (COIL-100)**. Technical report, Feb 1996. 20

[35] P. Schneider, M. Biehl, and B. Hammer. **Adaptive relevance matrices in learning vector quantization**. *Neural Computation*, **21**(12):3532–3561, 2009. 22

[36] Petra Schneider. **Relevance Matrices in LVQ**. In Michael Biehl, Barbara Hammer, Michel Verleysen, and Thomas Villmann, editors, *Similarity-based Clustering and its Application to Medicine and Biology*, number 07131 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2007. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. 22, 55

[37] A. Sato and K. Yamada. **Generalized Learning Vector Quantization**. In *Advances in Neural Information Processing Systems*, **7**, pages 423–429, 1995. 22

[38] Michael Biehl, Barbara Hammer, and Petra Schneider. **Matrix Learning in Learning Vector Quantization**. Technical Report IfI-06-14, Clausthal University of Technology, 2006. 22, 23

[39] Stephan Kirstein and Heiko Wersing. **A Biologically Inspired Approach for Interactive Learning of Categories**. In *International Conference on Development and Learning*. IEEE, 2011. 24

[40] Tim C. Kietzmann, Sascha Lange, and Martin Riedmiller. **Incremental GRLVQ: Learning relevant features for 3D object recognition**. *Neurocomputing*, **71**(13-15):2868–2879, August 2008. 24, 69