DOI:10.15774/PPKE.ITK.2019.005

Representation and recognition methods on parallel and unconventional machines

Ph.D. DISSERTATION

Attila Stubendek

Supervisors: Péter Szolgay, DSc; Kristóf Karacs, PhD

Pázmány Péter Catholic University

Faculty of Information Technology and Bionics

Roska Tamás Doctoral School of Sciences and Technology

Budapest, 2018.

DOI:10.15774/PPKE.ITK.2019.005

Table of contents

DOI:10.15774/PPKE.ITK.2019.005

DOI:10.15774/PPKE.ITK.2019.005

DOI:10.15774/PPKE.ITK.2019.005

List of abbreviations

AD	Averaged distance
AL-NN	Adaptive Limited Nearest Neighborhood
CGD	Cross-group distance
CMOS	Complementary Metal-Oxide Semiconductor
CNN	Cellular Neural Network
CZMD	Complex Zernike Moments Descriptor
EPPSED	Extended Projected Principal Shape Edge Distribution
FFNN	Feed-forward neural network
FN	False negative
FP	False positive
FPGA	Field-programmable gate array
GA	Genetic Algorithm
GFD	Generic Fourier Descriptor
GSPPED	Global Statistical and Projected Principal Edge Distribution
IGD	In-group distance
KNN	K-nearest neighborhood classifier (k-NN)
OCNN	Oscillatory Cellular Nonlinear Network
ON	Oscillatory network
PPED	Projected Principal Edge Distribution
STO	Spin Torque Oscillator
SVM	Support Vector Machine
TN	True negative
TP	True positive
VLSI	Very Large Scale Integration
ZMD	Zernike Moments Descriptor

DOI:10.15774/PPKE.ITK.2019.005

1 Introduction

Designing and building a machine that helps people, is the most important challenge and achievement of our civilization. For thousands of years by the development of tools used for moving and handling, one had to bear less and less burden on its shoulder and could devote more time to art and thinking. As a result, new, better machines were created giving aid to people in the field of arithmetic, logic, and even making inferences. Nowadays the primary indicator of the development of our civilization is not just the capability of the human family, but the capabilities of the machines made and taught by the people.

Today state-of-the-art engineering aims to understand and reproduce the process of the human thought, recognition, and learning – to mimic everything that has put us to the top of the evolution. Meanwhile, the scientific community will meet again and again with the recognition that life and patterns of organizations in nature, the excellent work of God is the perfect inspiration for creating the simplest and most complex devices and alike.

Sight is the most fundamental way of human perception. Therefore, any machine that aims to interact with humans and the human world is required not only to be capable of sensing physical light but also to be able to *interpret* the image.

Artificial vision or *computer vision* is employed in a number of applications that make the daily life easier for millions. Artificial vision provides more accurate and faster healthcare, automated industry and quality checking, safe unmanned vehicles, and exploration of places where humans cannot enter. The quality of life of visually impaired can be improved by orders of magnitude by devices using reliable artificial vision.

Automated *image understanding* utilizes methods of several different disciplines. Image or image flow is first *preprocessed*, and mathematical *descriptions* are generated, composing an object *representation*. If given a specific task regarding the image, descriptions are used for *analysis*, *modeling* or *comparison*. Thus, the key of an effective image understanding is encapsulated in the choice of the features: the representation must highlight those modalities of the image that are discriminative for the given task and neglect other irrelevant aspects.

DOI:10.15774/PPKE.ITK.2019.005

In a mathematical interpretation, *object recognition* is a mapping from an image to a few dimensional output vector. The output is an answer to the actual *visual query* regarding the presence, the position, the category or a numerical feature of the object. To give an answer, a decision machine is required that has the appropriate a priori *knowledge* gained by *learning* from already known *examples*.

Methods and architectures employed in computer vision applications are consciously getting closer to the known mechanisms of the human vision system and perception. Classical architectures and concepts are universal, although in several special cases dedicated architectures are significantly faster and provide lower power consumption by employing multiple processing cores and utilizing the *precedence of locality*. For that very reason, the computational schemes of the available devices should be considered during the design of any algorithm.

The speed of the digital computing blocks used in processors is approaching their physical borders. Following Moore's Law about the development of architectures, the innovation of multi-core processors using parallel architectures began in the last decade. Meanwhile, the interest has increased for alternative technologies as well, breaking out of the Neumann's paradigm. One of these principal branches of research is the development of non-Boolean-based, non-digital computing units implemented in a network of interacting nanomagnets called *Spin Torque Oscillators*. The basic principle of the architecture is to use current to excite the magnets that, depending on the current and the topology, synchronize with a pattern of phases on each oscillator.

In my work, I defined the terms of data and program on oscillatory networks and showed how to use them as computing units. I conducted experiments aiming to find the class of problems that are effectively computed on oscillatory networks.

In my research, I have been striving to create general methods. However, a particular area of usage, the design and the implementation of algorithms employed in the Bionic Eyeglass was the primary source of my motivation. The Bionic Eyeglass is an application collection for visually impaired, realized as a part of a research project of the Hungarian Bionic Vision Center. Functions asked by the potential users of the applications and the limitations of hardware available defined the requirements, aims, and constraints concerning the algorithms. Objects to be recognized – pictograms, banknotes – are all two-dimensional and rigid patches, thereby in my research, I focused on algorithms that describe and recognize shapes this kind of nature.

DOI:10.15774/PPKE.ITK.2019.005

1.1 Classification

A concept of the machine learning was described as the "*field of study that gives computers the ability to learn without being explicitly programmed*" by Arthur Samuel [1]. In Euripides' view learning involves an agent remembering its past in a way that is useful for its future [2].

Learning is essential to an *intelligent agent* as the capability of improving its behavior based on experience [2]. In the *supervised learning* approach, the goal is to *predict* the decision mapping from *inputs* to output *labels*, when a certain number of elements of the mapping are given [3]. The set of these a priori known mappings is called *training set*.

Depending on the output type, the label can be a qualitative or a quantitative variable. According to the naming convention, predictions of continuous, quantitative outputs are called *regression*, and the inference is called *classification* if the output label is discrete qualitative variable. [4]

Classification is a prediction used to determine a nominal, discrete, class value, that can be a category, a semantic label, a discrete number, or a logical value as well.

The simplest classification is binary, in this case, the label is a binomial variable. Binary classifications include mappings where labels take values of true or false, 1 or 0, in or out, a presence or a lack of a particular property, or any set with two discrete elements.

If the class labels take more than two values, thus the label is a polynomial variable, we speak about multiclass classification. [3] In specific tasks the number of the classes is not necessarily limited. In this case, we distinguish relevant and not relevant classes from the aspect of the actual task. The classification process is different if it is known that the input belongs to some of the relevant classes (i.e., boy or a girl), or if the input may be the member of any other classes (voices of birds are recognized, but other noises also may appear on the input record). Real-world objects and their representation often belong to an unbounded number of classes. Depending on the task, out of these classes, the number of relevant ones may be orders of magnitude smaller than the number of irrelevant classes, thus representing each irrelevant class is not efficient, if at all feasible.

The most important distinction of predictors is based on the relationship between the input data size and the number of the prediction model parameters. If the model size changes with the amount of training data, we speak about *non-parametric models*, and the models having fixed number of parameters for a fixed task, independently from the training set, are called *parametric models*. [3][5] In the next paragraphs I will show some examples of non-

parametric models – artificial neural networks, decision trees, SVMs –, and parametric models –K-nearest neighbor models [6], that are later mentioned or compared.

1.1.1 Nearest-neighbor methods

The K-nearest neighborhood model (KNN) is one of the most straightforward classification algorithms, an example of memory-based or instance-based learnings, where a new input is compared to the instances in the training set, specifically the output is the plurality vote of K nearest neighbor template outputs. [4][6] The underlying assumption behind the algorithm is that the feature space is continuous, and the samples in the feature set, except on the boundaries, are surrounded in the majority by other instances from the same class. From the probabilistic aspect:

$$p(y = c\,|x, D, K) = \frac{1}{K} \sum_{i \in N_K(x,D)} \Im(y_i = c)$$

thus the probability p that the input x belongs to the class c, is determined by the output labels y_i of the K closest instances in the training set D, where $N_K(x, D)$ is the set of the indices of the K closest instances in D, and $\Im(e)$ is the indicator function: [3]

$$\Im(e) = \begin{cases} 1, & if\ e\ is\ true \\ 0, & if\ e\ if\ false \end{cases}$$

To design a nearest neighbor model for a specific task, we face making several choices. In order to define closeness, a metric in the feature space is needed. In general, typically a *Minkowski distance*, or L^p *norm* is used: [6]

$$L^p(\boldsymbol{x_i}, \boldsymbol{x_j}) = \left(\sum_q |x_{i,q} - x_{j,q}| \right)^{1/p}$$

In case of real attributes, $p = 2$ is a common choice, resulting in the *Euclidean distance* [3], or with $p = 1$ the *Manhattan distance* is used as well.

Another choice to make is to find the optimal K. For small K, we get many small regions for each class, and with large K we get large class regions with smooth boundaries. If $K = 1$, we speak about *nearest neighbor rule* or *nearest neighbor classifier.* [5]

The K-nearest neighborhood algorithm is simple from the aspect of learning since the training complexity is zero. Furthermore, addition and removal from the model are straightforward as well. Beyond its simplicity, the advantage of the NN is its suitability to be implemented on dedicated VLSI architecture. KNNs are successfully used in several classification problems, such as handwritten digit recognition, image scene classification,

and EKG analysis. From another aspect, KNNs are successful where classes are not necessarily compact, many subclasses, prototypes form the *superclasses*, and the class boundaries are irregular. [4]

The conceptual drawback of the KNN is the lack of generalization, meaning that no interpretable model can be derived from the training set. Another disadvantage is called the *curse of dimensionality*, summarized in a nutshell, for high dimensions the KNN models perform poorly [3][6]

In this thesis I will deal with the following two major drawbacks of the KNN: One is the KNN's sensitivity to the distribution and the size of the training set instances. Second is the lack of reject option that makes the KNN method inappropriate for classification problems, where the number of the classes is large, and only a subset of the classes, the relevant ones are distinguished in the training set. [4][7]

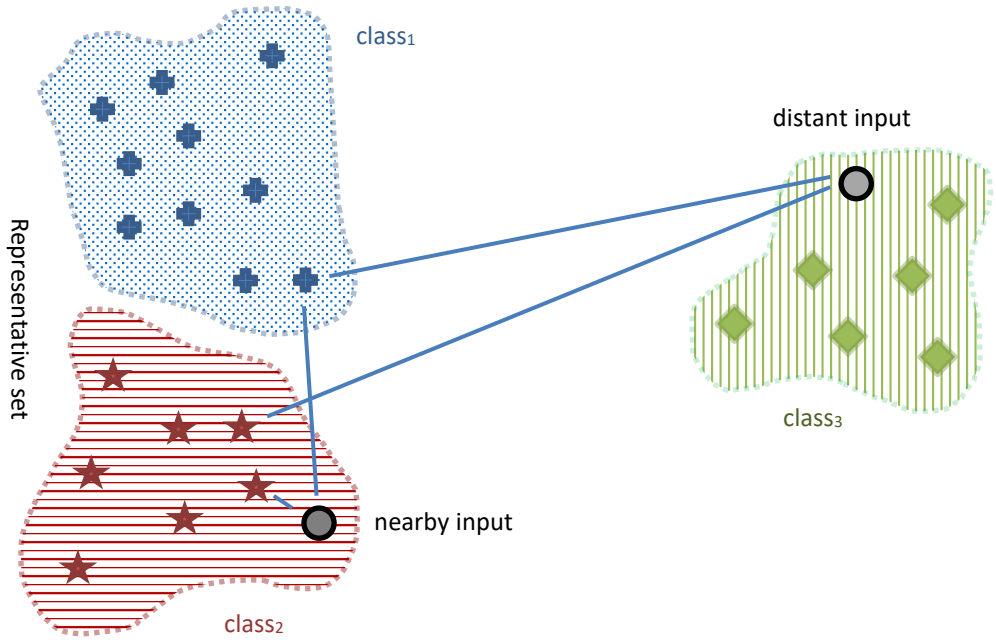


Figure 1.1. The basic principle of the nearest neighborhood classification: the input (black circle) is classified to the closest known instance. In the example class1 and class2 are subsets of the representative set, but class3 is not represented. A nearby input is classified easily, but a distant input is also classified to the closest instance in the representative set, in our case to class1, despite it transparently belongs to class3.

As the classifier name says, the input is classified based on the closest known instances. However, for every point of the feature space is true that there is always a nearest element, even if it is very far away [4], whether because it lies in a region that is covered sparsely, or not covered by a training instance. (Figure 1.1)

I will focus on the following questions: how can we know if the closest known instance is close enough? What could be the criteria for accepting or refusing an input? How can the training set instances be optimized in order to have all the classes covered, but not over-represented?

In my thesis, I propose an extension to the standard nearest neighborhood method that makes the model able to reject inputs by distinguishing relevant and non-relevant classes and a gradual decision algorithm that provides faster classification.

1.1.2 Decision trees

Decision trees are another simple and powerful classifier family, and they are the most widely used logic method.[7][8] The basic principle of classifying by the decision trees is the following: start from the root of the tree, in every non-leaf node i.e. *decision node* use a branching query regarding the attributes of the input to choose between the branches, and continue, till a leaf node is reached, where the output class is assigned based on the leaf.

From another aspect, the root node splits the feature space into two or more subsets. Its children – as long as they are non-leaf nodes – continue splitting the resulted subsets along other attributes – dimensions. If the resulted subset contains instances only from the same class, a leaf node of that class terminates the branch. [9]

Decision tree learning, i.e., the construction of a decision tree for a training set is to determine which attributes to split on in every node, which attribute to choose first, and which to continue. Given a set with a binomial or polynomial attribute, the best is to find an attribute, that splits the space into *pure*, homogenous subspaces form the aspect of the label. However, this kind of attribute is rare; hence a metric is needed to measure the *purity* or *information* from the relation of the subsets and the corresponding label splits. [9][10] Several purity-impurity functions are known. However, the most frequently used decision tree algorithms (ID3, C4.5) employ the *entropy*. For a polynomial label (having l different values), for the (sub)set S, the entropy is defined as follows:

$$I(S) = -\dot{p_1}\log_2\dot{p_1} - \dot{p_2}\log_2\dot{p_2} - \cdots - \dot{p_l}\log_2\dot{p_l}$$

where $\dot{p_\iota}$ is the empirical probability of the output label j in the subset S:

$$\dot{p_J} = n_j/(n_1 + n_2 + \cdots + n_l)$$

The *information gain* of the attribute e is the difference of the entropy of the set S, and the weighted average of the entropies of each split: subsets S_1 to S_k of S, resulted by the splits along the attribute values v_1 ... v_k:

DOI:10.15774/PPKE.ITK.2019.005

$$I_G(e) = I(S) - \sum_{i=1..k} \frac{|S_i|}{|S|} I(S_i)$$

where $S_1 \dots S_k \subset S$, resulted by the splits along the values $v_1 \dots v_k$ of attribute e. [8][9][10]

Decision trees describe the training set, adapted to its weaknesses, overrepresented and poorly covered regions, and noise. This phenomenon is called *overfitting*, and in the case of decision trees is manifested in a very deep, detailed, fragmented tree structure, representing each case of the training set. From another point of view, the more specific the tree is on the actual training set, the less generalization is encapsulated in the model. Thus, the goal is to simplify, to *prune* the tree, even if the error rate increases beneath a reasonable level. Generally, two methods can be employed to reduce overfitting: The first is not to split a subset under some statistical condition and instead of a decision-node represent the set with a leaf of the majority class – this approach is called *preprunning*. In contrast, the *postprunning* removes some subtrees, after the decision tree has been built, based on accuracy criteria. [8][7]

The advantage of the decision trees is the speed and their simplicity, the models are easily readable, interpretable, changeable, and understood by a human observer, making the decision trees a frequently used tool in data mining [7][10], Due to the logical approach, decision tree learning is not sensitive to attribute distribution or attribute dependence. [8]

1.1.3 Artificial Neural Networks

The neural system is one of the wonders of the created world: The brain is a highly sophisticated computer with billions of parallel, nonlinear cores. It processes sensed information from the receptors, provides a representation of the surrounding context, interacts with the environment, and routinely solves pattern recognition problems. It develops with experiences and adapts to changed conditions. [11] Thereby the structures and the mechanisms discovered in the brain repeatedly arouses the interest of mathematicians, software and hardware engineers, researchers of the artificial intelligence, machine learning, robotics, etc. Here I will show some basic methods inspired by the neuron network, as the set of connected, parallel, distributed computational units.

The basic unit of an artificial network is the artificial neuron. Neurons are connected by directed *links*, that transmit the *activation* between the *units*, input and output, and each link has a scalar *weight*. Units have a bias weight as well.

DOI:10.15774/PPKE.ITK.2019.005

For each unit, the weighted sum of all input is computed, and an *activation function* is applied to derive the output:

$$a_j = \varphi\left(\sum_{i=1}^{n} w_{ij} a_i + b_j\right)$$

where a_l is the activation from unit l, w_{ij} is the weight associated to the inputs a_i, and b_j is the bias of the unit j. Other notations denote the bias as w_{0j} with a constant activation $a_0 = 1$. Depending on the activation function φ, the unit is called *perceptron*, if g is a hard threshold, or Heaviside function, and we speak about *sigmoid perceptron*, if φ is a sigmoid function. [2][6][11][12] An example for a sigmoid function is the signum function, the hyperbolic tangent function (tanh), and a logistic function defined as follows:

$$\varphi_l(v) = \frac{1}{1 + \exp(-av)}$$

where v is the weighted sum of input activations, a is the slope parameter.

Depending on the topology, and connection patterns, several types of different neural networks are known.

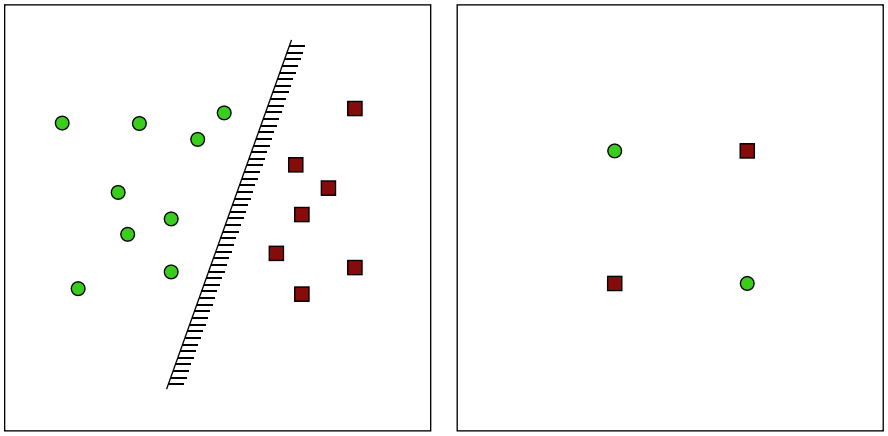


Figure 1.2. Example datasets with two classes. The dataset on (a) is linearly separable. On figure (b) the XOR-like function is shown, which is not separable linearly.

Perceptron

The basic topology consists of one artificial neuron only. Despite its simplicity, the perceptron is a powerful classifier, if the data is *linearly separable*, i.e., there is a *hyperplane – decision surface –* in the feature space that separates the classes (Figure 1.2.a). [5][10]

Perceptron learning is determining the input weights and the bias by iterating through training set. Every time an input is misclassified, the weights are adjusted based on the

misclassified input, multiplied with a learning rate parameter. If the feature space is linearly separable, the algorithm converges. [5][10][13]

Feed-forward neural networks

In most cases of the interesting classification problems the data is not linearly separable (a simple example is the XOR function, Figure 1.2.b), thus one single perceptron is not enough. One of the possible solutions is to employ more artificial neurons and connect them into a network. If the topology does not contain any feedback connection, we speak about a feed-forward network (FFNN). If the neurons are structured in layers, the first one is the input source layer, and the last one is the output layer. If the architecture contains more layers, intermediate ones are called *hidden layers*. [6][9] By boosting the structure with hidden neurons the network becomes able to extract higher order statistics from the input and acquires a global perspective despite its local links. [11]

Training of a FFNN is similar to a perceptron learning, with the difference that the errors found at the output layer, have to take effect in possibly all affecting neurons; thus the error has to be propagated backward. [5][9][13]

Recurrent networks

If the directed graph of the neural network links contains a *feedback* loop, the structure is called a recurrent network. The feedback loops represent a basic memory in time and result in a nonlinear dynamic behavior. Recurrent structures are responsible for some types of episodic memory processing and encoding special representations in the human brain, in the hippocampus, and the cortex. [14][15] Such architectures of artificial neurons are successfully used in time-dependent pattern classifications correspondingly. [11][16]

1.1.4 Support Vector Machines

As mentioned before, the overwhelming majority of classification problems is not linearly separable. Another way to overcome this problem is to transform the input feature space into another space with higher dimension, where the problem is then linearly separable. These transformations are called *kernel functions*. [2]

DOI:10.15774/PPKE.ITK.2019.005

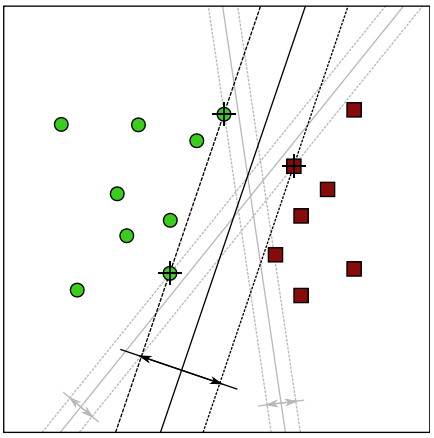


Figure 1.3 Finding the optimal separator (thick black line) between two classes. Support vectors of the optimal separator are data points marked with a cross. The margin is the distance of the dashed lines.

Since transforming to arbitrary large dimension makes any dataset linearly separable; however, the model easily gets overfit and loses its generalization capability, hence the goal is to find the optimal separator that creates the *maximal margin* between the classes (see Figure 1.3). [6][16]

Finding the margin, i.e., finding those (transformed) dataset points that hold the separators – *support vectors* – is a quadratic programming optimization problem. However, the choice of the kernel function is somewhat based on the designer's knowledge about the problem domain and the available kernel functions and their parameter space. The most frequently used kernel functions are polynomial kernel functions, and radial basis functions, such as Gaussian kernel function:

$$K_{poly}(x, y) = (\langle x|y\rangle + 1)^d,$$

$$K_{Gauss}(x, y) = \exp\left(\frac{-\|x - y\|^2}{\sigma^2}\right)$$

where $\langle \cdot \mid \cdot \rangle$ represents the dot product, d is the order of the polynomial, and σ is the kernel width. [8][17][18]

DOI:10.15774/PPKE.ITK.2019.005

1.2 Genetic Algorithms

Computer science, including machine learning, frequently cherry-picks concepts and methods from the created world. Genetic algorithms mimic the principle of the *biological evolution and natural selection* in a stochastic beam search problem, with the difference that the subjects of the evolution are vectors, other mathematical structures, and algorithms instead of living beings.

Genetic algorithms work on a *population*, which consists of *individuals* represented by their *chromosomes*, that are being built up from *genes* or *bits*. The GA starts with a random population of length k, and in every iteration new offspring are created from the previous generation by the *genetic operators*. Parent selection is analog to the natural way: the better, stronger, faster the individual is, the more has the chance of reproduction. The corresponding term for "better" in the world of genetic algorithms is having higher *fitness*. The fitness function estimates the successfulness of the individual, generally with a positive number. [2][6][21][19]

To select a parent for the new generation, two frequently methods are used:

- *Truncation selection*: Simply select f of the best individuals, where $1 \ll f \ll k$, and drop the rest of the population.
- *Fitness-Proportional Selection*: Every individual has the chance to become a parent with a probability proportional to its fitness value – even the worst individuals can survive, and the best ones may not become a parent. The generally used function for individual i with fitness $F(i)$, satisfying the conditions above is the following:

$$p_{simple}(i) = \frac{F(i)}{\sum_{j \in P} F(j)}$$

An extended version of selection probability calculation employs a temperature T (or in [13] a selection strength $s = 1/T$) that might decrease during the algorithm:

$$p_{Botzmann}(i) = \frac{\exp\left(\frac{F(i)}{T}\right)}{\sum_{j \in P} \exp\left(\frac{F(i)}{T}\right)}$$

DOI:10.15774/PPKE.ITK.2019.005

The most widely and generally employed genetic operators are:

- *Crossover*: New individuals are created by mixing of two existing chromosomes. The most common strategy is the single point crossover: For parents $\boldsymbol{p^{(1)}} = [p_1^{(1)}, p_n^{(1)} \dots p_n^{(1)}]$ and $\boldsymbol{p^{(2)}} = [p_1^{(2)}, p_n^{(2)} \dots p_n^{(2)}]$ chose a random point s, where $1 < s \leqq n$, and concatenate the corresponding parts to get the new individual $\boldsymbol{p^{(k+1)}} = [p_1^{(1)} \dots p_{s-1}^{(1)}, p_s^{(2)} \dots p_n^{(2)}]$. Multi-point crossover utilizes more split points, uniform crossover randomly chooses each gene from its parents.
- *Mutation*: For each gene in the offspring, its value is changed with a small probability, often chosen $p_{mut} \approx 1/L$, where L is the length of the chromosome.

[2][6][16][20][21]

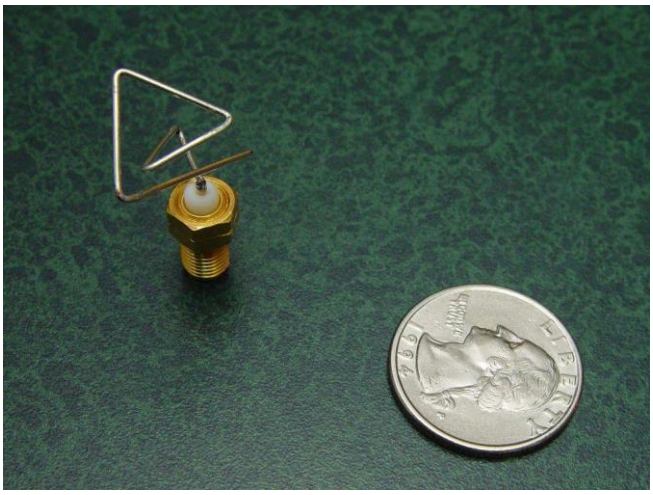


Figure 1.4. Antenna geometry designed with a genetic algorithm by the NASA Evolutionary software. [1]

Genes might be discrete or continuous, and might not only represent numbers, characters, but complex structures, such as operations, functions, graphs, topologies, geometric configurations (see Figure 1.4) [22] and program source codes [23] as well.

In my work, I used a genetic algorithm in finding the optimal filter parameters in the gradual classification (Section 4.), and in designing oscillator topologies (Section 6.)

[1] Source: http://www.nasa.gov/centers/ames/news/releases/2004/antenna/antenna.html

DOI:10.15774/PPKE.ITK.2019.005

1.3 Shape Recognition

Generally, the goal of computer vision is to generate answers to visual queries based on the input image. Depending on the query, several levels can be identified in a vision problem. A typical categorization distinguishes between detection, localization, and recognition.

In the detection part, the presence of an object is examined, localization determines the position of the object, if available, whereas recognition identifies the detected objects, possibly considering their context in the visual scene. However, the definition of an object depends on the task. [24][25] In typical computer vision systems the result is computed from the image through its features, as a verified hypothesis [26][27]. Similar to queries, features may incorporate local details of pixel surroundings as well as global image properties [28]. If patches or complete contours are extracted from the image, shapes of local image parts can be localized, detected re or recognized.

The shape is the most meaningful visual aspect of an object, more than the color, material or texture [29][30]. Consequently, shape recognition is an essential element of artificial vision, especially in understanding digital images and image flows. [31] A broad spectrum of application areas relies on shape recognition, including robotics [32][33], healthcare [34][35], industrial automatization [36][37][38], surveillance systems [39], unmanned vehicle orientation [40], assistance for impaired [41].

In computer vision, the shape generally is binary image distinguishing the inner regions of the shape and the background. Many applications are defined on grayscale shapes or can be extended to process grayscale patches, and shapes might be represented with vector graphics as well, however, in this thesis, I will focus on binary shapes consisting of pixel points.

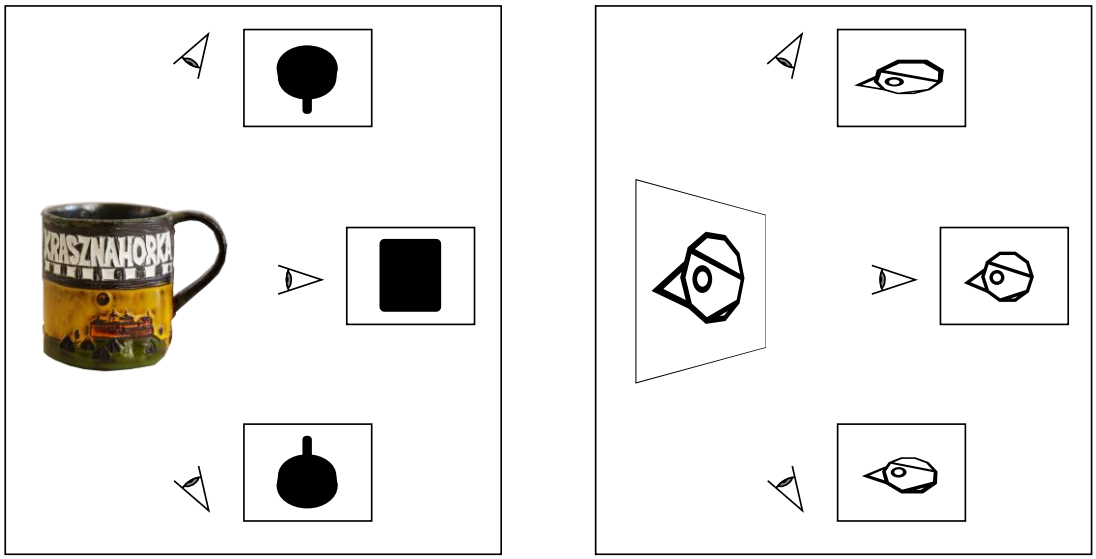


Figure 1.5. The two-dimensional projections of shapes. For an object having an extent in a 3D (a), projections have an only semantic relationship through the original, three-dimensional object. Projections of a flat, two-dimensional patch are transformable to each other.

The way of generation of the shape, the direct source of the binary shape image is a black-box for the task of shape recognition, that will not be opened in this thesis, just in a nutshell, a shape can be a result of an image processing method that generates binary images, such as pattern extraction, segmentation or thresholding.

From the perspective of this thesis, the source of the original image is more important since the shape images suffer from the same geometrical transformations and distortions caused by the camera optics, 3D-to-2D projection, photo acquisition, and sampling. From the aspect of the projection dependencies, we distinguish flat patches, and objects having an extent in all the three dimensions. The two-dimensional projection of a flat patch depends only on the mutual position of the camera and the subject, and the patches from different angular viewports can be transformed to each other. In contrast, the projection of 3D objects depends on the actual positioning of the object – a sculpture or a mug has a completely different shape viewed from different sides, not necesseraly transformable to each other, as seen in Figure 1.5. [42][43].

In my work I was less concerned about 3D shape recognition; however, several descriptions and methods described here can be easily extended from 2D to 3D, and a wide spectrum of researches concentrate on 3D shape modeling, projections, and analysis. [44][45][46][47][48][49]

In my thesis I focus on flat, rigid shape patches obtained from a digital image taken by a standard camera or a cell phone, that has an extent only in 2D, such as pictograms, road signs, patches of banknotes, etc. The challenge in recognition of such shapes lies in the high variety of objects, resulted by various lighting conditions, unambiguous segmentation and uncontrolled camera handling.

The key to efficient shape recognition is to use an appropriate representation that comprises all crucial characteristics of a shape in a compact descriptor. A shape description is considered to be efficient from a recognition point of view, if

- the representation is compact,
- a metric for the comparison of the feature vectors can be efficiently computed,
- the representation is insensitive to minor changes and noise, and
- the description is invariant to several distortions, such as translation, scale, perspective transformation, and rotation.

The most basic classification of shape descriptions distinguishes between contour-based and region-based techniques. Each method extracts specific features that encompass some significant aspects of the information in shape.

The requirement of compactness stands for the maximal level of independence of the feature data that do not go to the expense of the comparison and recognition performance. In other words, redundancy in the feature vector is accepted if it significantly simplifies the subsequent processing of the vector, thus accelerates the classification, and may increase the accuracy of the recognition.

Typical shapes of objects seen in our everyday life are compound, and no unified aspect can categorize all these shapes unambiguously and clear. Human address shapes several properties, as being rough or smooth, fat or skinny, but the most conventional way to identify and describe shapes is by comparing or addressing by prototypes (“shaped like a…”). [50] Hence it is expedient to observe and analyze more modalities, resulting in more robust recognition.

In the following sections I will present the most important shape description methods, and an image descriptor as well, that inspired the GSPPED shape descriptor.

The basic mathematical features of a shape, as perimeter, area, eccentricity consist of only one or few scalar values, though by which they express the feature of the shape in a very compressed way. Complex shape descriptors belong to two groups depending on the part of the shape they describe:

- *Contour-based shape features* describe the shape based on its contour lines in various representations, such as contour moments [51][52], centroid distances and shape signatures [53][54][55][56][57], scale space methods [58], spectral transforms [59][60], and structural representation [61][62][63]. Common drawbacks of contour methods are the complexity of feature matching, representation of holes and detached parts of the shape, and noise sensitivity [64].
- *Region-based techniques* describe the shape based on every point of the shape and represent mainly global features of the shape. Moment invariants are derived as statistical features of the shape points [65]. Orthogonal moment descriptors such as Zernike and Legendre descriptors employ polynomials instead of the moment transform kernels [66][67][68]. Complex shape moments are robust, and matching is straightforward; however, lower order of moments poorly represents the shape, but higher orders are more sensitive to noise and difficult to derive [69]. Generic Fourier descriptor represents the shape as the 2D Fourier transformation of the polar-transformed shape.

In my research, I aimed to create a compound shape descriptor, in which several modalities of the shape are described, and are suitable for fast, robust and reliable recognition.

1.3.1 Basic shape features

The most basic descriptors combine different size parameters to form features independent form position, size and rotation. Dozens of such properties are known and used under several aliases, here the most widely employed are presented:

$$aspect\ ratio = \frac{d_{max}}{d_{min}} \qquad convexity = \frac{P_{conv}}{P}$$

$$solidity = \frac{T}{T_{conv}} \qquad area\ ratio = \frac{T}{T_{\blacksquare}}$$

$$roundness = \frac{4\,T}{\pi\,d_{max}} \qquad formfactor = \frac{4\pi}{P^2}$$

$$compactness = \frac{\sqrt{\frac{4}{\pi}\,T}}{d_{min}} \qquad eccentricity = \sqrt{1-\frac{b^2}{a^2}}$$

where T is the area, P is the perimeter of the shape, T_{conv} and P_{conv} are the area and perimeter of the convex hull, $T_{\blacksquare}$ is the area of the bounding rectangle, d_{min} and d_{max} are the minimal

and maximal diameters respectively, a and b are the semi-major and the semi-minor axes of the fitting ellipse having the same second moments as the shape. [50][70]

Basic features are highly expressive; however, they represent weak discriminative power for classification, although they are suitable for rejection obviously false matches. [65]

1.3.2 Shape moments

The binary shape image shape is a set of pixel points in a Cartesian coordinate system; therefore it can be considered as a statistical set, and statistical moments can be computed. Two-dimensional central moments of the joint density $p(x, y)$ are defined as follows:

$$U(r,s) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \eta_x)^r \, (y - \eta_y)^s \, p(x,y) \, dx \, dy$$

The term "central" refers to the spatial normalization, where $[\eta_x, \eta_y]$ is the mean of $p(x, y)$. m and n are the order of the moment. In case of shapes, the distribution is replaced by the binary (grey) values of the image (or an image function for non-binary images), means are the center of mass $[\langle x \rangle, \langle y \rangle]$, and instead of continuous, a discrete summation is employed:

$$m_{r,s} = \sum_{x,y} (x - \langle x \rangle)^r \, (y - \langle y \rangle)^s$$

Central moments are translational invariant but are dependent on the actual size of the shape. To get rid of scale dependency, the shape is scaled to a unit area. And since the area of a binary image is the $(0,0)^{th}$ moment, and a scale by a factor of α scales the moment to ${m_{r,s}}' = \alpha^{r+s+2} m_{r,s}$, the scale-invariant moments are calculated in the following way:

$$\bar{m}_{r,s} = \frac{m_{r,s}}{m_{0,0}^{(r+s+2)/2}}$$

Only the first few two-dimensional moments are easily understood by the human or at least have standard naming. Moments $m_{1,0}$ and $m_{0,1}$ are the first-order row and column moments, in the case of binary images the center of gravity coordinates – if centralized, then both are zero. Moments $m_{2,0}$, $m_{0,2}$ and $m_{1,1}$ are called the *row- and column moment of inertia*, and the *row-columns moment of inertia* respectively. [28][43][70]

The second order moments receive special attention since an ellipse can be assigned to a shape, which has the same second moments as the shape. The major and the minor axes

of the ellipse can be computed as the eigenvalues of the inertia *covariance matrix*, or *inertia tensor* ***J***:

$$J = \begin{bmatrix} m_{0,2} & -m_{1,1} \\ -m_{1,1} & m_{2,0} \end{bmatrix}$$

The *orientation* of the ellipse is the declination of the major axis, in other words, the object is most elongated in the direction of the major axis. The *orientation* of the ellipse and consequently of the shape then can be expressed with the moments [28][43]:

$$\phi = \frac{1}{2}\arctan\frac{2m_{1,1}}{m_{2,0} - m_{0,2}}$$

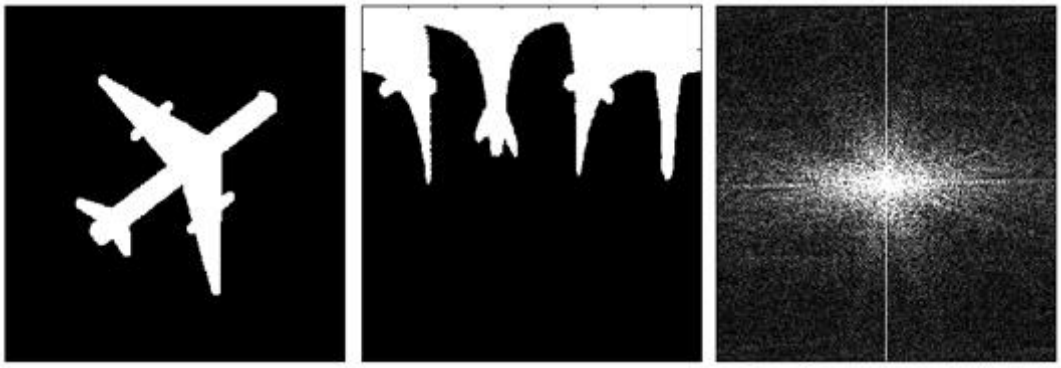
The *eccentricity* is another shape measure describing the elongation of the shape. Its value is 0 for a circle, and 1 for a line [43] [28]:

$$\varepsilon = \frac{(m_{2,0} - m_{0,2})^2 + 4m_{1,1}^2}{(m_{2,0} + m_{0,2})^2}$$

To develop moments that are invariant to the rotation as well, Hu proposed seven moments based on normalized image moments. The construction of these moments will not be presented in the thesis but can be found in [5] and in [28][71].

1.3.3 The Generic Fourier Descriptor

The idea behind the Generic Fourier Descriptor is in characterizing the shape image by a transformation of its two-dimensional spatial frequencies. To ensure rotational invariance, the image is transformed into the polar-space, sampled to a given resolution and transformed by the 2D-Fourier transform.



(a) (b) (c)

Figure 1.6. The generation of the Modified Polar Fourier Transform. The original image on (a) is transformed into the polar coordinates (b); finally, the spatial frequencies are obtained (c)

Fourier transform is successfully employed in several types of pattern analysis. The discrete two-dimensional Fourier transformed coefficients are defined as

$$F(u,v) = \frac{1}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1} F(j,k)\exp\left(\frac{-2\pi i}{N}(uj+vk)\right),$$

where $F(j,k)$ are the pixel values at (j,k), N is the width and height of the image, and $i = \sqrt{-1}$. [28]

The definition below, however, is not rotation invariant; thus the usage of 2D Fourier as features is suitable for describing textures, or shapes, where a characteristic direction is defined, but generally for shapes is not useful. Therefore, the image is transformed into the polar space and then transformed. The Modified Polar Fourier Transform for the ρ^{th} radial frequency and for the ϕ^{th} angular frequency is defined as:

$$PF(\rho,\phi) = \frac{1}{N}\sum_{r}\sum_{j} F(r,\theta_j)\exp\left(2\pi i\left(\frac{r}{R}\rho + \frac{2\pi j}{T}\phi\right)\right),$$

where $0 \le r = ((x-x_c)^2 + (y-y_c)^2)^{\frac{1}{2}} \le R$ and $\theta_j = j\left(\frac{2\pi}{T}\right), (0 \le j < T)$, $[x_c, y_c]$ is the center of gravity of the shape, and R and T are the radial and angular resolutions. In Figure 1.6 the GFD feature generation is demonstrated.

Finally, the Generic Fourier Descriptor is composed of normalized coefficients:

$$GFD = \left[\frac{|PF(0,0)|}{area}, \frac{|PF(0,1)|}{|PF(0,0)|}, \dots, \frac{|PF(0,n)|}{|PF(0,0)|}, \frac{|PF(m,1)|}{|PF(0,0)|}, \dots, \frac{|PF(m,n)|}{|PF(0,0)|}\right],$$

where $area$ is the area of the bounding circle, m is the maximum number of the radial frequencies, and n is the maximum number of angular frequencies.

Papers of the authors of the GFD claim that the GFD performs better than the Zernike moments descriptor based on tests on large shape databases. [72][73]

1.3.4 Zernike Moments Shape Descriptor

As shown before, rotation invariance is easily ensured by the shape moments. However, Hu moments do not fulfill the requirement of compactness; moments involve redundancy. Thus they are not orthogonal from the aspect of information theory. [74] Zernike Moments are orthogonal complex moments on the unit disk, that are rotationally invariant by the magnitude of the moments. [67] Originally, Zernike moments were designed to describe optical aberrations. [75]

DOI:10.15774/PPKE.ITK.2019.005

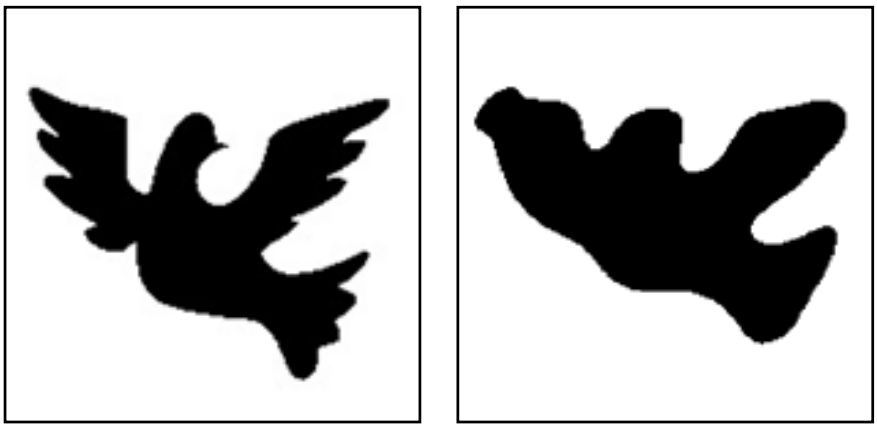
Zernike moments A_{nm}, where n is the order. and m the repetition for a discrete image are defined as

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y V_{nm}^*(r,\varphi) F(x,y)$$

where n is a positive integer, $m = -n, -n+2n, \dots, n$, $x^2 +$ and V_{nm} are the Zernike polynomials, where R_{nm} is the radial function [71][74][76][77]:

$$V_{nm}(r,\varphi) = R_{nm}(r)\exp(im\varphi)$$

$$R_{nm}(r) = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!} r^{n-2s}$$



(a) (b)

Figure 1.7. Birds. On the left side is the original shape, on the right the reconstructed. 15 orders were used.

Radial coordinate r and angle φ in the definitions of the Zernike polynomials are defined on the unit disk, thus the shape has to be transformed onto the disk as well. By a proper transformation scale and translation invariance is provided. A common choice uses the following functions:

$$r = \frac{\sqrt{(x-x_c)^2 + (y-y_c)^2}}{r_{max}}$$

$$r_{max} = \frac{\sqrt{m_{00}}}{2}\sqrt{\frac{M}{N} + \frac{N}{M}}$$

$$\varphi = \arctan\left(\frac{y-y_c}{x-x_c}\right),$$

DOI:10.15774/PPKE.ITK.2019.005

where N and M is the size of the image, r_{max} is the radius of the circular part of the image that will be mapped to the unit disk, and x_c and y_c are the centroid points. [71]

The rotation invariance is ensured by the magnitude of the complex moments since if the image is rotated by an angle α, moments also "rotate" [71][74][76]:

$$A'_{nm} = A_{nm} \exp(-im\alpha)$$

Zernike Moments are suitable not only for detailed representation and thus recognition but for reconstruction as well. (see Figure 1.7.). Due to the orthogonality, no redundancy is involved in the features, making the Zernike moments highly expressive. The original form of the calculation is complex, thus slow, but there exist methods for faster computation. [78] However, employing higher order Zernike moments is less effective due to generalization loss and sensitivity to noise [75].

1.3.5 The Projected Principal Edge Distribution

Projected Principal Edge Distribution (PPED) is a grayscale image descriptor that characterizes principal edges of the 64x64 pixels' moving image window developed for recognizing anatomical regions in X-ray images (Figure 1.8). The reason to highlight the PPED is the motivation behind its construction: to mimic one of the characteristics of the human image processing, the oriented edge description [79], and its design for a dedicated VLSI chip. The PPED was the inspiration of the edge-based part of the compound shape descriptor presented in this work.

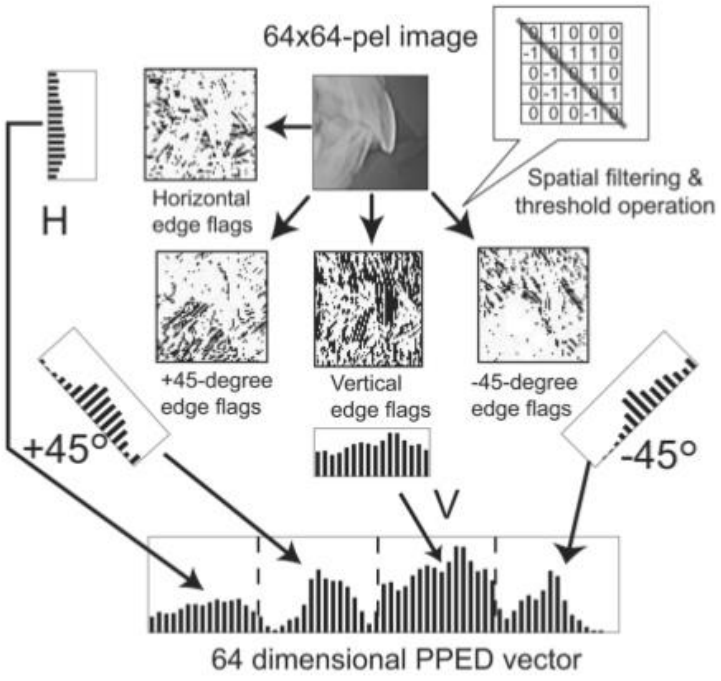


Figure 1.8 The schema of the PPED algorithm

To highlight important edges, for every pixel a local threshold is defined as the median of differences of neighboring pixel values in a 5×5 pixels' window around the pixel. Edges are detected in four directions $(0, \pi, \pi/4, \pi/2, \text{and } 3\pi/4)$ with a convolution resulting four edge maps, where values below the actual pixel threshold (defined above) are set to zero. To select the principal edges only, for every pixel location of the four edge maps, only the largest edge value is kept, and the values of the same pixel location on the other three edge maps are set to zero. Maps are then projected in the same direction as the convolution and normalized to the length of 16 values. Finally, smoothing is applied to reduce noise.

For every window position on the input image, a separate feature vector is computed and then compared to the labeled templates, choosing the closest instance to classify the input. [80]

Note that PPED and relative descriptors are not rotationally invariant, and scale invariance is ensured by using various window sizes and scaling.

DOI:10.15774/PPKE.ITK.2019.005

1.4 Non-Boolean computing architectures

The physical limitations of the state-of-the-art CMOS technique stimulate researchers to find a new solution to keep up with the Moore's law. One principal approach is to multiply the computational elements and use many simple and fast elementary processors [81]. Due to data transfer delay, which is now the bottleneck of the computation speed, the precedence of locality became extremely important. [82][83] The second direction searches for new concepts of computation beyond the CMOS technology, and aims to re-think the basics of information science we were used to in the past decades.[84][85][93][94]

In my research, I investigated both kilo-processor based algorithms and non-CMOS based architectures. In the following section, I will outline the basic concepts of oscillator networks and the Cellular Neural-Nonlinear Networks.

1.4.1 Oscillators

The oscillator is a system producing *oscillation*, a repetitive change of *states*. The oscillation can be characterized by the *period* as the time between the same states, and by the *frequency* as the number of reaching a certain state in a time unit. If a metric can be defined on the states, the *amplitude* is the distance between the extreme states. The actual state of the oscillation in the state space is the *phase*.

From mathematical aspect the oscillation is a state function that is the periodic solution of a differential equation with parameter μ:

$$\ddot{x} = f(x, \mu)$$

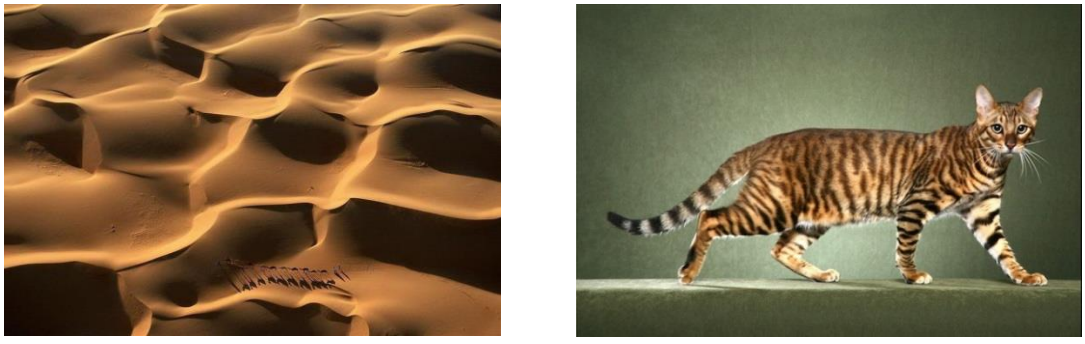


Figure 1.9. Spatial patterns in nature: sand dunes in a desert[2], an ancestral tabby pattern on a cat[3]

[2] Author: Yann Arthus-Bertrand, http://goo.gl/9Njkx7
[3] Credits: Helmi Flick/Science, http://www.scienceupdate.com/2012/10/cat/

DOI:10.15774/PPKE.ITK.2019.005

Oscillations occur in mechanics, electrodynamics, but oscillation is ubiquitous in nature, biology, chemistry, and social sciences as well. The oscillator is a pushed pendulum, a mass hanging on a spring, or an excited RLC circuit. [86][87][88][89]

Examples of spatial patterns are shown in Figure 1.9.

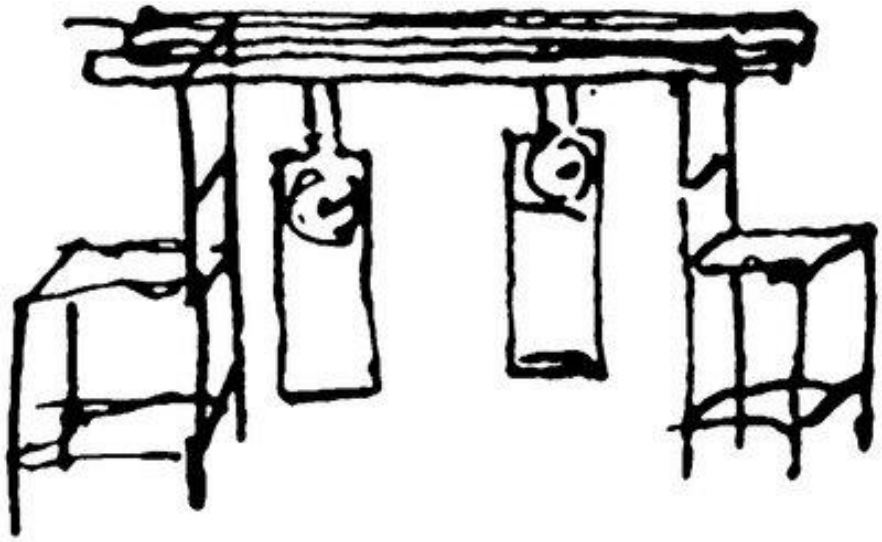


Figure 1.10. Huygens' experiments with pendulum clocks hanged on the same, flexible lath in 1665 [4]

1.4.2 Coupled oscillators

The networks of oscillators, where the nodes are interacting, are called coupled oscillators.

The coupling is defined by the coupling strength, the effect function of the coupling, and the speed and dynamics of the coupling effect.

$$\ddot{x_1} = f(x_1) + \varepsilon \cdot g(x_1, x_2, \dots, x_n)$$
$$\ddot{x_2} = f(x_2) + \varepsilon \cdot g(x_1, x_2, \dots, x_n)$$

...

$$\ddot{x_n} = f(x_n) + \varepsilon \cdot g(x_1, x_2, \dots, x_n)$$

Based on the coupling strength (ε) strong and weak couplings are distinguished. In my research, I investigated systems with weak coupling. A generally used example for weakly coupled oscillators are metronomes placed on a flexible surface (Figure 1.10): After a certain time weakly coupled oscillators may get the same frequency, the phenomenon is called *synchronization*. [86] [88]

[4] Synchronization, Shcolarpedia, http://www.scholarpedia.org/article/Synchronization

In my research, I assume that the state space of every node is the same. The behavior of the oscillatory network is determined by the individual dynamics of the nodes, the coupling effect, and the attenuation. In the following sections I will use models of weakly coupled oscillators, with the assumption that no external event affects the behavior of the system, and the structure and the physical parameters are constant. The interaction of the system and the outer environment is realized only by excitation of several oscillators and by reading the states of the network.

1.4.3 Spin Torque Oscillators

Spin Torque Oscillators (STOs) are nanosized microwave oscillators placed on a flat surface. The oscillation occurs in the spin vector's spatial position (***M***), a direct current injection achieves excitation. During oscillation, spin waves are radiated. The interaction of the oscillators depends only on their relative position, i.e., the distance that directly defines the coupling strength and function. [90][91]

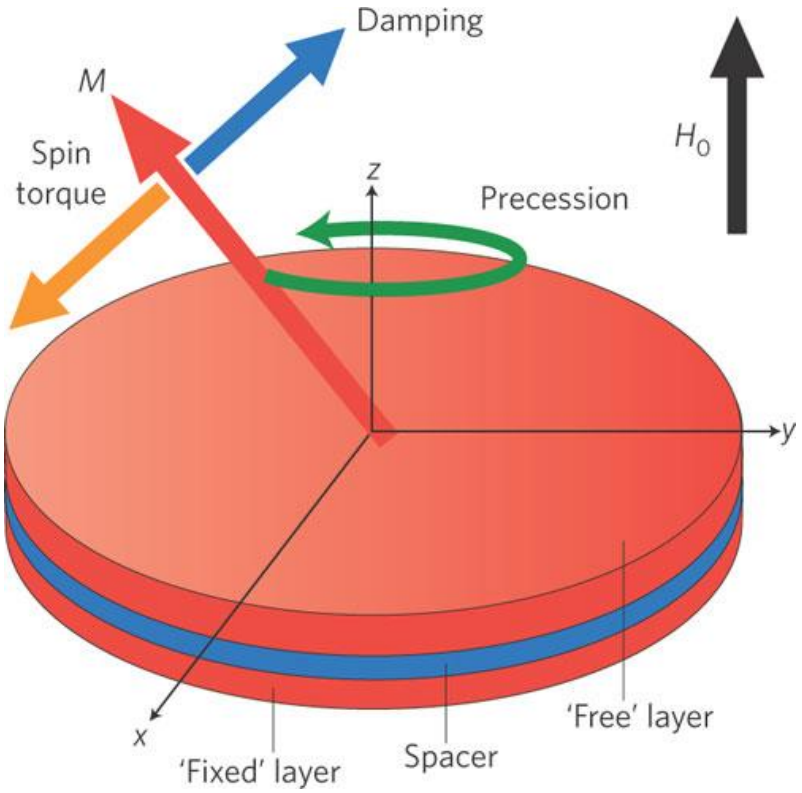


Figure 1.11 The free layer magnetization, M (red arrow), precesses around the direction of an applied magnetic field (H_0) when natural magnetic damping (blue arrow) is compensated by the spin torque (yellow arrow) applied by a spin-polarized current flowing from the fixed layer. [5]

[5] Andrei Salvin: Microwave sources: Spin-torque oscillators get in phase, Nature Nanotechnology 4, 479 - 480 (2009)

The magnetization is defined by the vector:

$$\boldsymbol{M}(t) = \left(M_x(t), M_y(t), M_z(t)\right)^T$$

The equations of the motion of the spin torque nano-oscillator results to be:

$$\frac{d\boldsymbol{M}}{dt} = \gamma\left(\boldsymbol{M} \times \boldsymbol{H_{eff}}\right) - \gamma\alpha\boldsymbol{M} \times \left(\boldsymbol{M} \times \boldsymbol{H_{eff}}\right) - \gamma A\boldsymbol{M} \times (\boldsymbol{M} \times \boldsymbol{S})$$

Ms is a parameter related to the saturation magnetization of the material. Permalloy is characterized by $Ms = 8.6 * 10^5$,

$\boldsymbol{H_{eff}}$ is the resultant magnetic field

$\times$ denotes the cross product between the vectors,

A is the normalized current (This number is proportional to the current of the oscillator and not the actual current itself (for instance A=100 corresponds to 1 mA).) and

γ (gyro-magnetic ratio) and α (magnetic efficiency) are physical constants with values $\gamma = 2.21 \cdot 10^{-5}$ and $\alpha = 8 \cdot 10^{-3}$, respectively. [92]

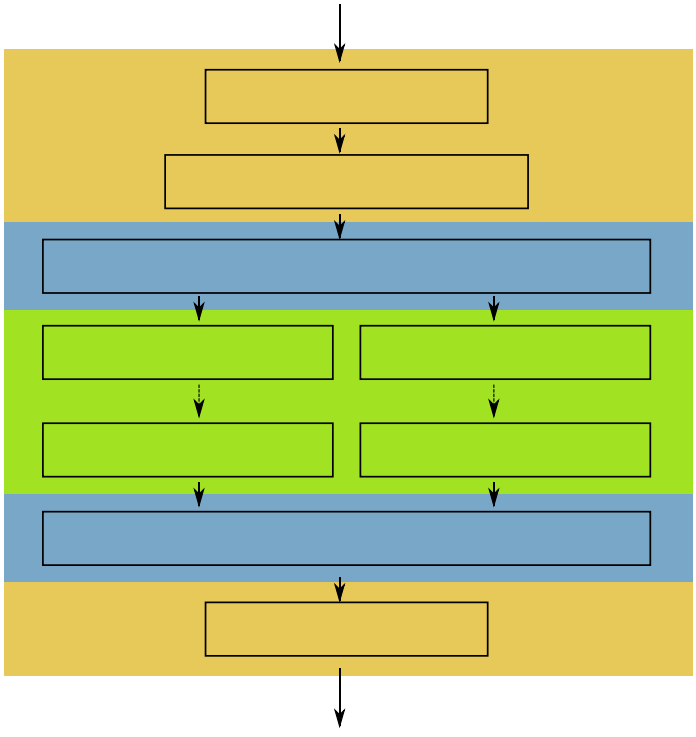


Figure 1.12. The structure of a CMOS-STO network processing unit. The input is preprocessed, then a conversion layer translates the digital data into analog currents interpretable by the STO network layers. Finally, the outputs of the STO layer has to be converted back to the CMOS level as well. The output is the result of a classifier.

Communication between the STO network and the CMOS layer is through a conversion layer (Figure 1.12): the input, as mentioned above, is realized via current excitation, the output is the array of the phases read out by a circuit loop. When my research was performed, by the state-of-the-art method only the edge of the system could be reached, so the excitation and the phase read-out could only be applied on the oscillators placed at the border of the network. [A5]

The STO network synchronizes after a particular time. In the case of a time-varying excitation, it can happen that the synchronization does not occur or breaks up and only smaller clusters remain synchronized. These partial synchronization patterns may encode important information also, but in my research, I only investigated fully synchronized cases. [95]

One computational cycle on an STO network corresponds to synchronization that is in order of milliseconds. Therefore, one cycle is significantly longer than a step of a basic CMOS unit. However, I will show that a program that is solved in one cycle on an STO can be performed on a standard CMOS architecture in substantially longer time. Therefore, even if basic tasks are also solvable on STOs, their usage is unnecessary. In the case of a certain class of complex tasks, oscillators may return the solution faster.

Besides acceleration, the significantly lower power distribution of systems with STOs is also an important aspect.

The advantage of the architecture is the size and the computational power, which require different programming method than in case of CMOS programming, and the ability to process static and spatial-temporal input due to the interaction of the oscillators. [95]

1.5 The Cellular Neural/Nonlinear Networks

The CNN is *an array of analog dynamic processors* or cells that mimic the structure and behavior of sensory and processing organs in nature and extend it with the additional capability of programmability. Cells are arranged in a grid and are *connected only locally* with neighboring cells in their *sphere of influence* with the radius r_d. (see Figure 1.13)

DOI:10.15774/PPKE.ITK.2019.005

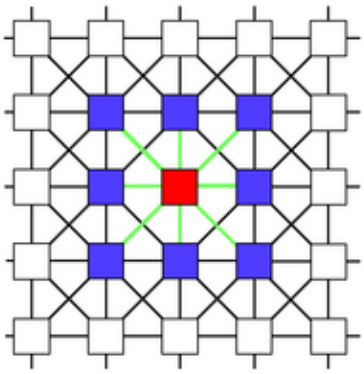


Figure 1.13 Cells and connections in a CNN with r_d=1: A cell is connected only with its neighbors.

The computational model of a single layer CNN (the Chua-Yang model) is described by a nonlinear differential equation system of a state variable of a cell *x*:

$$\dot{x}_{ij} = -x_{ij}(t) + \\ + \sum_{|k-i|\le r_d} \sum_{|l-j|\le r_d} A(i-k, j-l)\, y_{kl}(t) + \\ + \sum_{|k-i|\le r_d} \sum_{|l-j|\le r_d} B(i-k, j-l)\, u_{kl}(t) + \\ + z_{ij}$$

A and ***B*** is the feedback and the input synaptic matrix, *z* represents the threshold, the *template* (***A***, ***B***, ***Z***) determines the exact function of the actual CNN array.

U is the input. Note that the input is directly processed after its acquisition.

The output *y* is typically defined as a piecewise-linear function [96]:

$$y = f(x) = \frac{1}{2}(|x+1| - |x-1|)$$

The CNN paradigm rewrote the paradigm of the classic Boolean software-based thinking. The CNN technology successfully provided new and efficient solutions in many fields, mainly in image understanding, topological applications, but also non-topological problems [96][97]. Complete library of basic and complex templates can be found in [98].

The design both of the Adaptive Limited Nearest Neighborhood classifier and the GSPPED shape descriptor was motivated by the principle of parallel computing on kilo-processor architecture with low power consumption with real-time processing on a standard architecture of a PC or a cell phone as well.

DOI:10.15774/PPKE.ITK.2019.005

1.6 The Bionic Eyeglass

The Bionic Eyeglass is an abstract portable device, to help blind and visually impaired people in everyday navigation, orientation and recognition tasks that require visual input. [99][100][101][102][103][104] The device is being developed continuously, finalized algorithms are now implemented on different platforms, like Android, iOS and FPGA. The Bionic Eyeglass assembles several functions required by visually impaired, namely banknote recognition, crosswalk detection (see Figure 1.14) [105][106][107], public transport number reader and others.

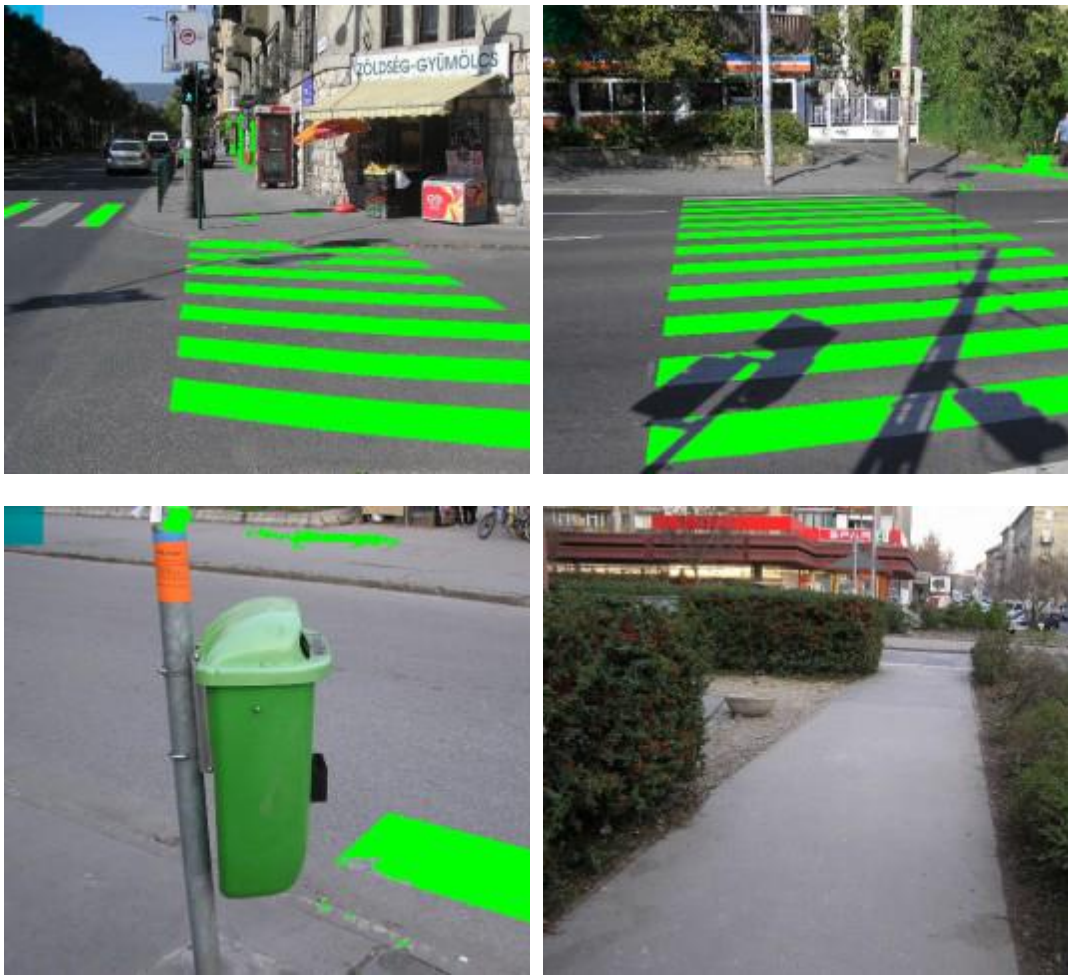


Figure 1.14. Examples of white crosswalk detection with the Bionic Eyeglass

Since the Bionic Eyeglass aims to be a handful, reliable, fast and easy tool, it is required to employ real-time algorithms with low power consumption. Hence most of the visual algorithms used by the device are based on CNN technique.

DOI:10.15774/PPKE.ITK.2019.005

Figure 1.15 The banknote recognition module in use. The module was designed to function in different lighting conditions and backgrounds fully.

Figure 1.16 The morphological processing of the colored input image. Extracted blobs, as numbers and portraits are then classified based on their shape or pattern.

The flagship of the Bionic Eyeglass is the banknote recognition module developed to identify Hungarian Forint banknotes in a cell phone image stream (Figure 1.15), incorporated in the Android[6] and iOS[7] application "LetSeeApp"[8]. The core process flow of the recognition is the following: The input is obtained from a single cell phone camera; the image is segmented, and interesting blobs are selected employing a hierarchical peeling algorithm (see Figure 1.16). [108][109][110][A4] Then the blobs are classified based on the shape, color, and their relative position. [A3] Finally an assembled classifier selects the output class, and the device provides voice feedback to the user. [A5][A7]

I participated in the development of the Bionic Eyeglass in designing and implementing a fast and robust shape descriptor and a classifier. I present the details of the algorithms in Sections 3, 4, 5.

[6] https://play.google.com/store/apps/details?id=com.letseeapp.letseeapp&hl=hu
[7] https://itunes.apple.com/hu/app/letseeapp/id1170643143?mt=8
[8] http://letseeapp.com/

DOI:10.15774/PPKE.ITK.2019.005

1.7 Structure of the thesis

In Section 1., I introduced the most important concepts of the machine learning, shape recognition and oscillatory networks. In Section 2., I investigate open-world object recognition and classification, from a general aspect, founding the basic concepts of my work. In Section 3. the Global Statistical and Projected Principal Edge Distribution description are presented, while in Section 4. the Adaptive Limited Nearest Neighborhood Classifier is introduced as a part of a two-level classifier. The relevant shape descriptors and classifiers presented in the Introduction are compared with the developed methods in Section 5. Oscillatory networks, including the formalization of the OCNN architectures, and classification of both static and dynamic input, are investigated separately in Section 6. I summarize the results and my thesis points in Section 7.

DOI:10.15774/PPKE.ITK.2019.005

2 Object recognition in an open-world environment

In open-world multiclass recognition problems, only a relatively small subset of the classes is considered relevant for the given task. This is similar to a binary classification scheme with only positive and negative labels, with the difference that inside the positive class we need to be able to differentiate between several "positive" labels, which are considered relevant by themselves, as opposed to the irrelevant ones, among which no differentiation is necessary. More precisely, the relevancy attribute partitions the set of classes into the relevant and the irrelevant subsets.

2.1 Performance evaluation in multiclass classification

For an appropriate evaluation performance, metrics need to be adapted to this nature. Due to the prevalence of the positive-negative property for this multiclass case, it makes sense to rely on traditional binary performance metrics, including recall and precision. To be able to use them, we need to extend the binary confusion matrix scheme of positive and negative decisions. Since we do not differentiate between irrelevant classes, all decisions from and into irrelevant classes are counted as true-negative (*TN*). True-positive (*TP*) counts all correct positive, that is, relevant, classifications; false-negative (*FN*) refers to the number of decisions where a relevant input was classified as irrelevant. False-positive decisions are split into two categories: FP_{Rel} indicates the number of false classifications between relevant classes, while FP_{NRel} counts decisions where an irrelevant input is classified as a relevant one.

Using this extended taxonomy, precision and recall can be defined as follows:

$$precision = \frac{TP}{TP + FP_{Rel} + FP_{NRel}}$$

$$recall = \frac{TP}{TP + FN + FP_{Rel}}$$

For F_β, being a weighted average of precision and recall, the definition does not need to be changed, with recall being more important for $\beta > 1$, and precision weighted more important for $\beta < 1$:

$$F_\beta = (1 + \beta^2) \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

As I primarily target real-time recognition tasks on video sequences, type II errors have a much lower cost than type I errors. Hence, I have used the value $\beta = 0.05$, which reflects this preference. [5][10]

DOI:10.15774/PPKE.ITK.2019.005

2.2 The role of description and classification

I investigate classic machine learning decomposition and the role of edges and their appropriate and efficient representation. The estimation of the ground truth is based on limited sensing, resulting in a different representation of essentially same objects. The key point of the recognition is a model that draws boundaries of output classes. However, classes may differ based on various traits; thus the selection of discriminative features is also essential. From this point of view, I will divide recognition to feature extraction and classification.

In this section I investigate shape recognition that models the decision based on supervised learning, where the model is built up based on previously labeled inputs denoted as templates; the set of already known inputs are denoted as the training set. Independently from the exact type and behavior of the classifier, the classification is a comparison of the input to labeled elements from the training set (or a model built up from the set), where the decision is a function of the objects. The difference is a result of various distortions that occur during the image acquisition and preprocessing. Note that distortions may also affect the elements of the training set.

The input shape S_i^* is a result of a T transformation of the original shape S_i, where γ denotes the parameter(s) of the transformation and P is the set of all possible parameters of the transformation:

$$T_{\gamma_i}(S_i) = S_i^*, \gamma_i \in P$$

The input shape S_t^* is a result of a T transformation of the original template shape S_t:

$$T_{\gamma_t}(S_t) = S_t^*, \ \gamma_t \in P$$

The output class of S_i^* is the result of a decision function $\widehat{D}$, depending on one or more labeled shapes S_{t1}^* .. S_{tn}^*, comprising the representative set R:

$$\widehat{D}(S_i^*) = D_R(S_i^*)$$

$$\bigcup_{i=1}^{n} S_{t1}^* = R$$

The task of the recognition is not the reconstruction of the original shape by mathematical operations but to classify independently from transformations that distort the original and the template shapes and thus to estimate the ground truth C.

$$\widehat{D}(S_i^*) \approx C(S_i^*)$$

From this aspect the transformation can also be considered as noise and noise is considered as a transformation.

In the next paragraphs, I give an overview of possible distortions of a shape in an object recognition problem and formalize deviations mathematically. Then I try to define the ability to represent similarity by formalizing tolerance and invariance generally and especially for the target shapes. Finally, I give an overview of possible solutions of ensuring invariance and tolerance in a description-based recognition system.

2.2.1 Distortions in a shape description problem

To find the all possible deviations of shape I go along the process where the binary shape is generated from a real-world object. However, shape generally can be defined as a multidimensional set of points; in this thesis, as I mentioned, I only focus on 2D shapes that are projections 2D, flat objects in a 3D space and characteristic silhouettes of 3D images.

Applying the constraints above, during image acquisition by a camera, where the 3D-2D transformation and the sampling takes place, the following geometric and pixel-level deviations may occur:

a) Rotation of the object on its plane compared to the camera axes
b) Position difference of the object relative to the camera that can be split to
    ba) distance difference between the camera and the object
    bb) position difference of the projected camera origin and the object
c) Angular deviation of the object plane normal-vector and the camera projection direction
d) The appearance of noise due to sensing limitations and sampling errors
e) Some part of the shape is missing or the shape being joint with another pattern

Note that, from practical considerations, geometric variances can be represented in other spaces too. If we consider the characteristic motives of the shape to be larger than the sampling rate, the deviance in (d) is limited only to the sensing noise. However, inappropriate focusing may also cause loss of details of the shape which in most of the cases exceeds the sampling error. [28][42][43]

The shape is generated from the input image by various image processing algorithms, such as segmentation, patterns extraction, and morphological operations. As mentioned before, here, I will not investigate these preprocessing phases. Generally, it can be stated that the shape generation is a binarization of some characteristic pattern of the image; thus the deviation (e) may befall due to the various lighting condition and unambiguous shape edges.

DOI:10.15774/PPKE.ITK.2019.005

Summarizing the deviations variations can be named, of which shape recognition may be independent, or the similarity index should be proportional to the deviation. From the aspect of the shape, the distance variation appears in different scales of the shape. Positioning variance results in a different location of the shape on the image canvas; rotation of the image in its plane also results rotated shape. Angular deviation of the image plane together with positioning difference results in perspective variance. Not only do binarization ambiguity and noise result in misplaced edge pixels on the desired shape but also both of them may lead to detached shape parts or holes in the original shape.

2.2.2 Decoding shape similarity

Variance in the appearance of an object can be modeled in a mathematical sense as noise. We call shapes to be similar if the difference is due to different observation properties and processing noise. If the shape is rigid, observation property is reduced only to geometrical transformations. To achieve classification consistency across various distortions, we identify two different aspects, invariance, and tolerance, concerning these distortions.

Invariance of a recognition engine for a particular type of deviation is defined as the ability to return the same result for all inputs that only differ in the given deviation.

$$\widehat{D}\big(T_\gamma(S)\big) = \widehat{D}(S)\ for\ \forall\gamma \in P$$

We speak about tolerance to an effect if a difference in the input causes no difference in the output to a certain limit L_T:

$$\widehat{D}\big(T_\gamma(S)\big) = \widehat{D}(S)\ for\ \forall\gamma \in P, \|\gamma\| < L_T$$

Note that the norm for the transformation parameter is substantially an abstract function, which cannot be measured directly, but only can be estimated based on the transformed shape. Similarly, the limit L_T also represents an abstract value. Both the norm and the limit are determined by the actual interpretation of the similarity.

Tolerance can be defined as a limited, local invariance, and vice versa, invariance is a global tolerance. From this reason invariance with respect to an effect implies tolerance to the whole domain, while overlapping regions of tolerance can achieve invariance.

The human similarity metric highly depends on the actual task; thus no general statement can be defined which deviations should be eliminated and which should be tolerated during a shape recognition. The environment in some cases does provide some references regarding the projection details. Some of the parameters described above might be fixed, previously adjusted (e.g., relative orientation or position of the camera and the

object) or can be derived from the image metadata (e.g., the distance of the focused subject of an image, the angular difference from the horizontal plane). In these cases deviations in the given parameters result in a different shape; thus invariance is needed only if the human notion of the shape is not dependent of the distortion, and only tolerance is required if the given parameters are not exact or the human perception does tolerate deviations with a certain limit.

The transformations above can be characterized by the possible outputs applying the transformations. The range Q of transformation T is defined as the set of all possible results of transformation T on a shape S:

$$Q_T(S) = \{U, U = T_\gamma(S), \gamma \in P\}$$

In the case of reversible transformation $T_\gamma(\cdot)$, the inverse transformation is denoted here as $T_{\gamma^{-1}}(\cdot)$. To represent noise as transformation, I chose the parameter γ as a shape, and the noise transformation $T_\gamma(S) = S \oplus \beta$, and where $\oplus$ stands for the logical X-OR operation. By using this formalization, the random property of the noise transformation is ensured in a random selection of parameter γ. This annotation allows us to represent the noise as a reversible operation, where $\gamma^{-1} = \gamma$.

We denote shapes S and U to be separated by transformation T if there are no parameters γ_1 and γ_2 of T which transform S and U to the same shape:

$$\nexists \gamma_1, \gamma_2 \in P. T_{\gamma_1}(S) = T_{\gamma_2}(U)$$

$$Q_T(S) \cap Q_T(U) = \emptyset$$

If the transformation is reversible then S and U are separated by transformation T:

$$\nexists \gamma. T_\gamma(S) = U$$

$$S \notin Q_T(U)$$

If we assume that output classes are separated by transformation T, and no reference system is given, the recognition should be invariant to transformation T. If the classes are not separated, the recognition should only tolerate the difference caused by transformation T.

Without any assumptions about the noise, adding sampling and preprocessing noise to a shape (noise transformation) may result in an arbitrary distortion; no shapes are separated by noise transformation, and thus the recognition should only be tolerant to the noise transformation. If the noise is bounded, the result space is limited.

Adding sampling and preprocessing noise (noise transformation) theoretically may result in an arbitrary shape. The geometric transformations, except for the 90-degree

perspective distortion, are closed transformations; thus invariance with respect to rotation, scale, and transition and tolerance to perspective distortion are standard requirements in case of shape recognition. However, distortions affecting a shape cannot be handled separately. Sampling noise when doing a low-resolution scale or a flat perspective view can be significant. Hence, scale invariance and perspective tolerance are limited to scales where essential details of the shape are still present.

2.2.3 The role of feature extraction and classification

Invariance and tolerance regarding different distortions can be ensured in various ways. Feature extraction generalizes the shape from the specific aspect independently from those effects that are irrelevant for the classification, and classification performs a decision based on a complex distance. Hence, feature extraction is generally responsible for ensuring invariance and classification for tolerating difference to a specific limit. However, as I described in Section 2.2.2, invariance can be achieved by continuous tolerance and tolerance is a partial invariance; thus encoding similarities may occur in different parts of the recognition unit. Besides, many classifiers also include generalization power (i.e., kernel functions).

DOI:10.15774/PPKE.ITK.2019.005

3 The Global Statistical and Projected Principal Edge Distribution (GSPPED) descriptor

3.1 Motivation

Shapes have different properties depending on several aspects, and the distinctive characteristics may be encoded in a different aspect. Using only one feature type thus limits the description power of the descriptor in terms of discriminative power and classification performance. [62] Combining different descriptors include information about different essence of the shape and may contain redundant data, but increase robustness [111][112][113][114][115]. However, employing compound feature vectors require a decision method that suits the different parts of the description. In machine learning several ensemble classifiers are known that handle compound features, like boosting, bagging or stacking [9][69][116][117][118]

Representations of the same real-world object may differ due to several effects such as lighting conditions, camera settings, position, and noise. The major challenges of object detection are to ignore the differences in the representation resulting by sensing and preprocessing and to recognize if the difference is caused by different input objects. Several invariance requirements are often standard expectations to shape recognition methods, but the exact group of requirements has to be defined to each individual task, considering other parameters as well, such as hardware ones.

The description and the classification method presented in this thesis was developed in the framework of the Bionic Eyeglass Project that aims to help visually impaired by recognizing certain objects and patterns using cell phones, smart devices or dedicated architectures. The requirements towards the application also outlined the specifications of the used algorithms including shape recognition parts. The patterns to be recognized – figures on banknotes, pictograms, indoor and outdoor signs, etc. – are mainly rigid objects, but due to various image acquisition conditions and poor image quality significant amount of noise has to be handled, and several invariance requirements have to be fulfilled. Since the application is valuable only if it is reliable, false answers easily can cause a dangerous situation, thus minimizing false-positive errors has priority over maximizing cover ratio. Finally, that kind of algorithms is preferred that are appropriate for dedicated VLSI architecture but provide real-time processing even on standard cell phone CPU and GPU.

I suggest a shape description denoted as Global Statistical and Projected Principal Edge Distribution description (GSPPED) that combines shape features in order to represent

different aspects. To cover most of the potential aspects optimally and avoid excessive redundancy, independent features are utilized.

The descriptor consists of global statistical features and principal edge descriptors representing local characteristics. Structurally the descriptor is divided into three parts – see also Figure 3.1:

a) A highly powerful general header including eccentricity and area fill ratio.

b) A region-based feature set with histogram moments representing global shape properties.

c) A contour-based edge description employing the Extended Projected Principal Shape Edge Distribution description (EPPSED)

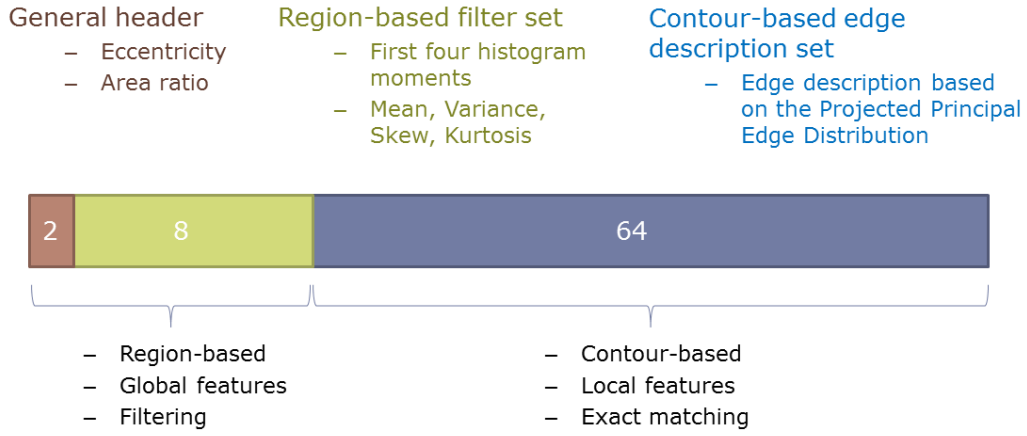


Figure 3.1 The schematic structure of the GSPEED descriptor

3.2 General region-based global features

As I highlighted in the Introduction, moments and general statistical features derived from moments are frequently used descriptors in shape and pattern recognition. A series of moments express the properties of a shape from basic features to details; however, moments of higher orders are more vulnerable to noise and variances in shape. Thus, in vision applications, where patterns belonging to the same class may vary due to camera position or segmentation, using higher-order moments is less effective.

The header part of the proposed description aims to depict the shape in the most compressed and expressive way. That kind of combinations are searched that are perceptually linear but may be calculated by nonlinear operations from easily measurable operands. Eccentricity and area ratio represents the basic outline of the shape however, they are only suitable to use as primary features at the first phase of the classification in filtering obviously false matches.[65] Besides they are simple scalars encompassing understandable

and most characterizing information for a human. The smaller the eccentricity is, the closer is the shape to a circle, while shape with eccentricity value of one is a line. The area ratio is the ratio of the area occupied by the shape and the area of the minimal rectangle covering the shape.

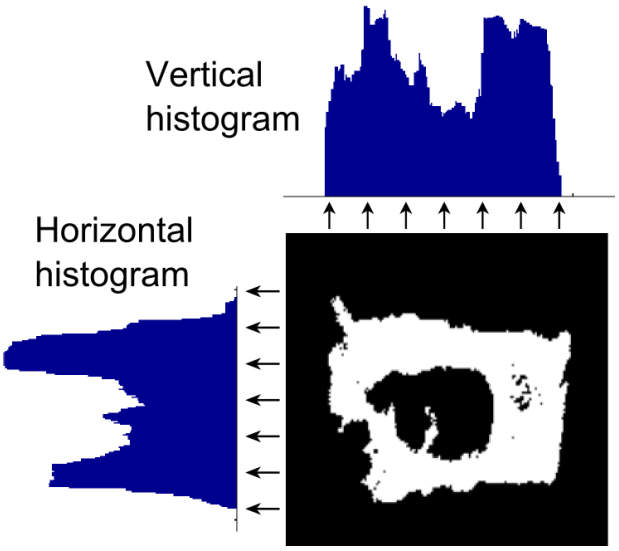


Figure 3.2. Vertical and horizontal histograms of a shape

Using more moments would enable us to describe the shape in more detail, but we would lose the general recognition ability. Thus, I used the first four moments: mean, variance, skewness, and kurtosis. For the sake of simplicity but not losing dimensional information, the moments are computed from the histograms of the shape (Figure 3.2). This solution reduces computational complexity compared to the two-dimensional moment calculation and provides advantages when the descriptor is computed on VLSI architecture.

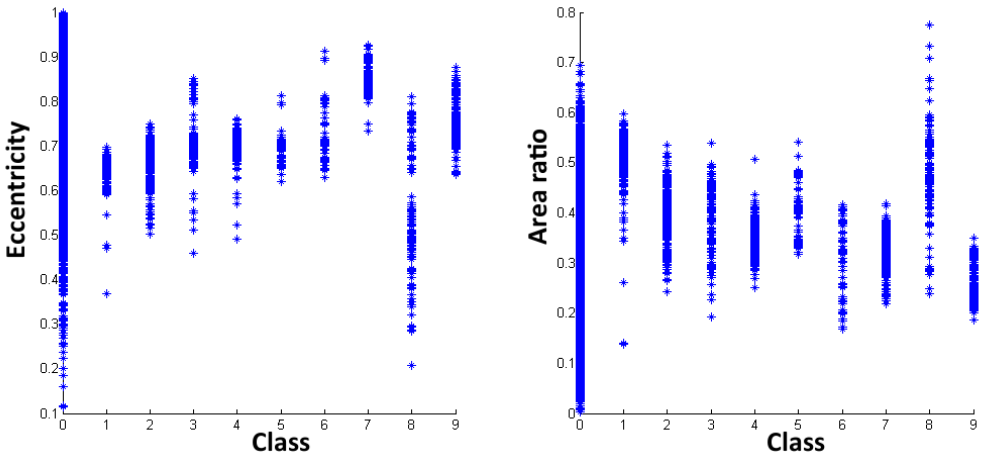


Figure 3.3. The distributions of the eccentricity and the area ratio on the Hungarian Forint Banknote pattern dataset (for details see Section 5.). Classes 1-9 represent different patterns from banknotes, and other irrelevant shapes are denoted as class 0.

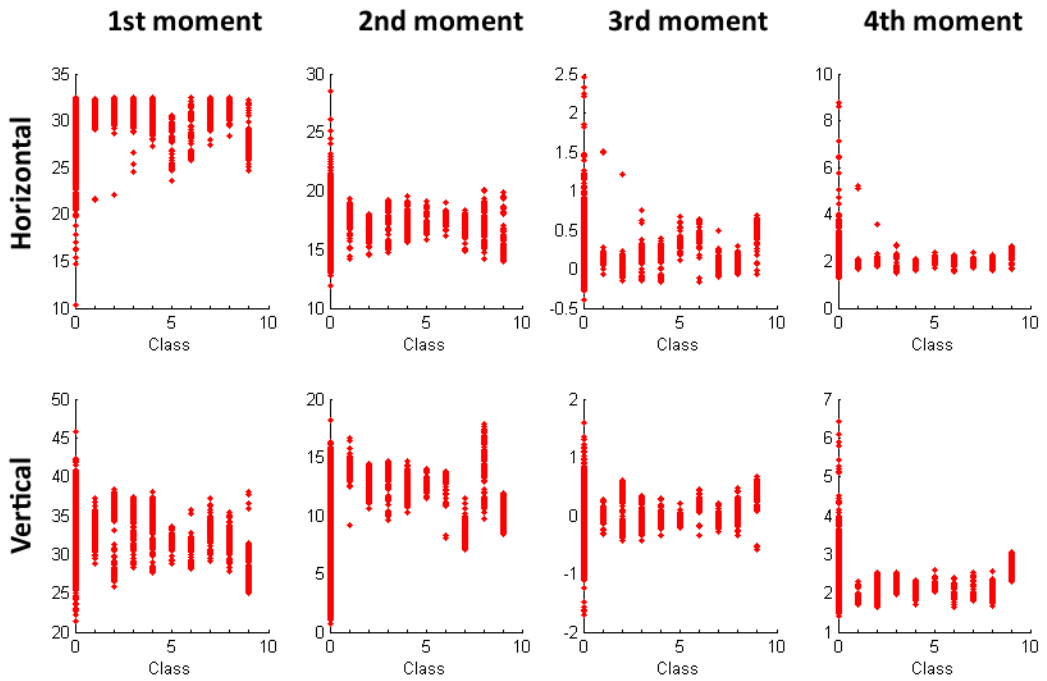


Figure 3.4 The distributions of the vertical and horizontal moments on the Hungarian Forint Banknote pattern dataset (for details see Section 5.). Classes 1-9 represent different patterns from banknotes, and other irrelevant shapes are denoted as class 0. The figure shows that these values do not contain enough discriminative power to classify the patches but provide a good guide to filter and reject obviously different shapes.

Figures 3.3 and 3.4 show the distribution of the global features of the banknote portrait set. The features do not represent enough discriminative power to make accurate classification but are feasible to filter out obviously different templates.

3.3 The Extended Projected Principal Shape Edge Distribution (EPPSED)

The core of the contour-based edge description is based on the principle used by the PPED presented in Section 1.3.5. The edge values are detected in four directions; principal edges are selected and then projected and concatenated; the result for one shape is a 64-element feature vector. The essential difference between the methods is in selecting the object, calculating the thresholds and the maxima of the four edge maps, and ensuring scale and rotation invariance.

3.3.1 Thresholding

An essential difference between the EPPSED and the PPED lies in the thresholding method and in choosing the maximal edge value. The aim is to design a cross-architecture

algorithm, where architecture-dependent computing does not influence the output significantly.

The goal of thresholding in the PPED algorithm family is to highlight principal edges values and hide less important ones. Global thresholding removes all values from the edge maps which are considered to be noise.

$$\forall i,j,d\ \bar{M}_{i,j}^{d} = t^{\theta_{global}}\left(M_{i,j}^{d}\right)$$

where i and j are pixel coordinates, $d \in directions = \{\leftarrow, \downarrow, \swarrow, \searrow\}$, $M_{i,j}^{d}$ is a pixel of the raw edge map at the $(i,j)^{th}$ position and from the d direction, $\bar{M}_{i,j}^{d}$ is the globally thresholded edge map, and $t^{\theta_{global}}(x)$ is the threshold function with the threshold value θ_{global}.

Comparative thresholding aims to select the most substantial value amongst the four edge maps and to neglect others.

$$\forall i,j\ \hat{M}_{i,j} = \max_{d \in directions}\left(\bar{M}_{i,j}^{d}\right)$$

$$\forall i,j,d\ M_{i,j}^{d} = t^{\hat{M}_{i,j}}\left(\bar{M}_{i,j}^{d}\right)$$

where $\hat{M}_{i,j}$ is the maxima of the four edge map values in the $(i,j)^{th}$ pixel location.

As described in the Introduction, the input of a shape recognition task is the shape: a binary image or a grayscale image with a binary mask, where the borders, the edges of the shapes are detected by the pattern extractor or the segmentation algorithm. Therefore, the selection of the principal edges is the task of the preprocessor, not the shape descriptor. From another aspect, the differences between neighboring pixel gray-values in a binary image are 0 or 1 (pixel value 1 for in-shape pixels and 0 for others); consequently, the median value is also 0 or 1. Hence using the median of differences as a global threshold is unnecessary.

I experimented with different threshold values, and I concluded that in case of hard thresholding, the best results can be achieved by using a global threshold value of $\theta_{global} = 2$.

DOI:10.15774/PPKE.ITK.2019.005

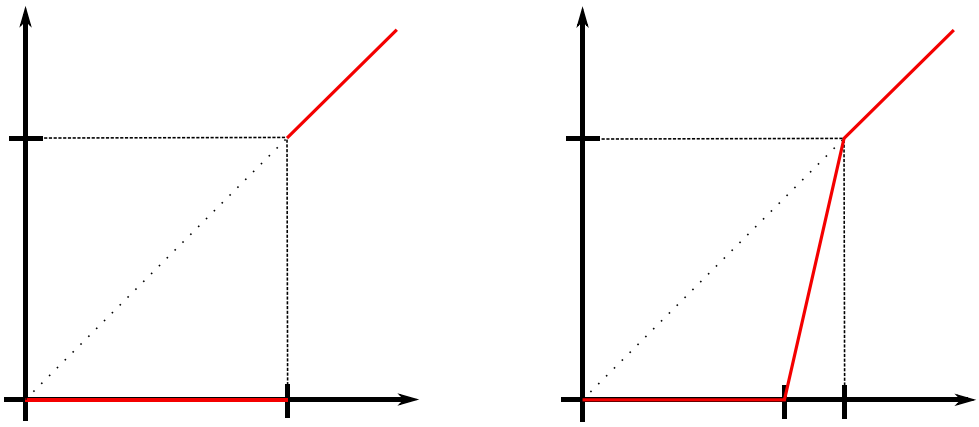


Figure 3.5, Thresholding functions: a) hard-thresholding, b) soft-thresholding.

Using hard-thresholding (t_{hard}) may result in ambiguous behaviors near the threshold value for almost identical edge values; thus I use a soft-thresholding (t_{soft}) method with no discontinuity (see Figure 3.5).

$$t_{hard}^{\theta}(x) = \begin{cases} 0, & if\ x < \theta \\ x, & if\ x \geq \theta \end{cases}$$

$$t_{soft}^{\theta}(x) = \begin{cases} max\left[0, \frac{\theta}{b}x + \left(1 - \frac{\theta}{b}\right)\right], & if\ x < \theta \\ x, & if\ x \geq \theta \end{cases}$$

where θ is the threshold value or the maximal edge value and b is the tolerance bound.

The width of the linear cutoff a can be set as a fixed value or as a portion of the threshold value θ. The advantage of having a fixed cutoff width is that the global threshold is also fixed, and the same difference is tolerated linearly under the threshold values. However, proportional cutoff width $b = l \cdot \theta$ provides the same steepness for every threshold. Note that $l = 1$ corresponds to an identical function in all cases, and for fixed cutoff width $b = 2$ corresponds to an identical function for the global thresholding.

DOI:10.15774/PPKE.ITK.2019.005

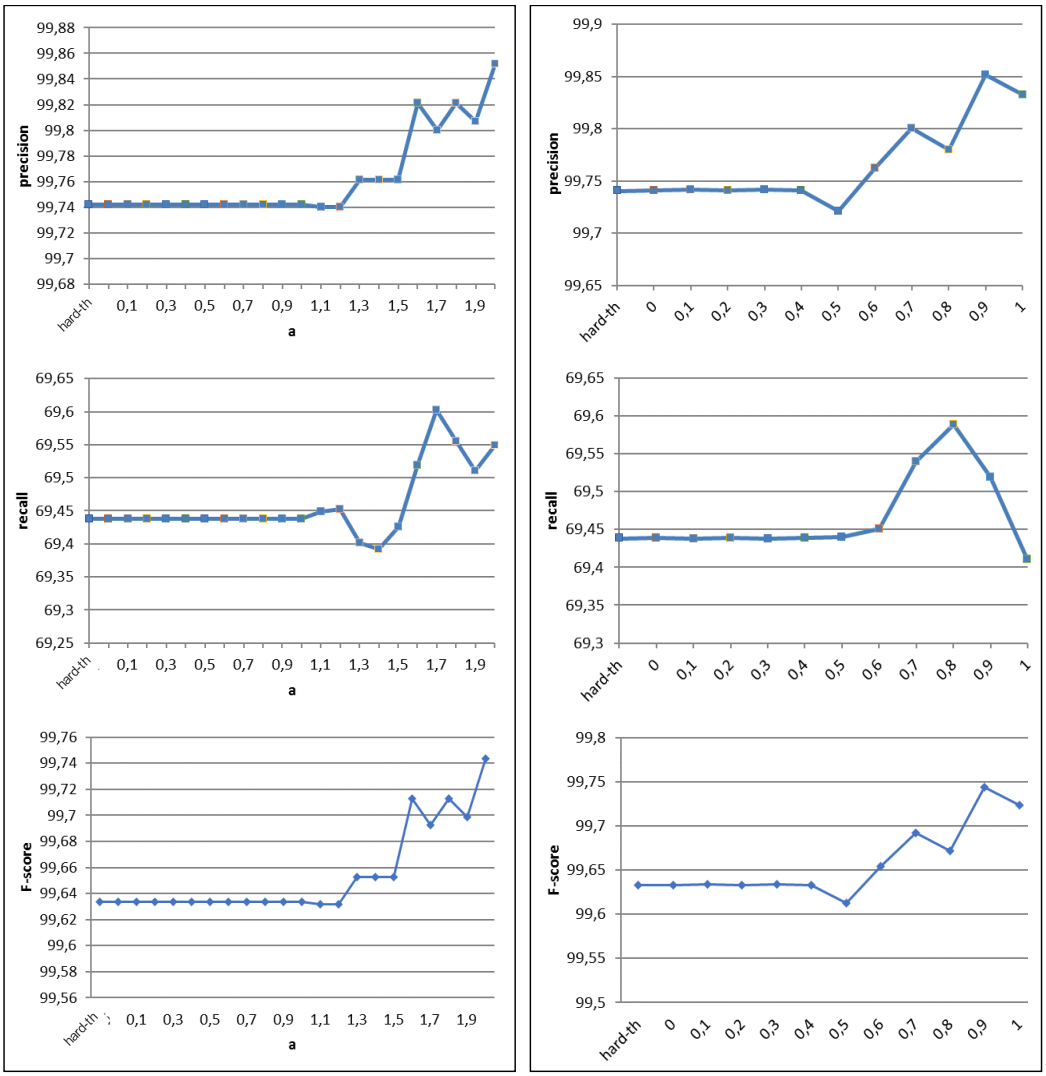


Figure 3.6. The performance of the classification depending on the thresholding. (a) shows results for fixed cutoff width, (b) for relative cutoff width. The first column marked as "hard-th" stands for the results reached with hard-thresholding.

Fixed cutoff width

Since the global threshold is set to *2*, the fixed cutoff was tested between *(0, 2)*. Results are summarized in Figure 3.6.a. Results show that employing soft-thresholding does not cause a decrease in the classification characteristics, except for the recall on a small interval that can be interpreted as noise. In my simulations, the F-measure value increases to 99.74% with a fixed cutoff width of $a = 2$, compared to 99.63 achieved by the hard-thresholding algorithm. For $b = 2$ both the precision and the recall increases significantly.

Relative cutoff width

I tested the global classification performance depending on the relative cutoff ratio l between *(0, 1)*, where $b = l \cdot \theta$. Results are summarized in Figure 3.6.b. Similarly to the

previous results, a narrow cutoff width does not cause any change neither in the precision and in the recall. Except for the precision on a small interval, the classification results do not decrease compared to the hard-thresholding. The best F-score value of 99:74% was achieved with cutoff ratio $l = 0.9$, where the soft-thresholding outperforms the hard-thresholding in the recall and the precision as well.

3.3.2 Normalization

|One significant deficiency of the PPED image descriptor is that it is not invariant with respect to rotation. Rotation invariance can be ensured at various phases of feature-based object recognition. One approach generates rotation-invariant feature vector applying adequate mathematical transformation while generation the description. Another possibility is to solve the rotation-invariance in classification technique whether by using rotationally redundant training set or by applying a preprocessing on the feature vector [119][120][121]. Since the PPED algorithmically is not rotation-invariant, only angular normalization or employing a rotationally redundant template bank may provide invariance. The latter solution can easily result in a vast and complicated database. To achieve rotation invariance, I chose to detect a characteristic angle and normalize the shape angularly. The orientation of the shape (defined as the declination of the major axis of the ellipse having the same second moment) serves well as a characteristic angle since it is consistent in the sense that orientation values of similar shapes are close to each other.

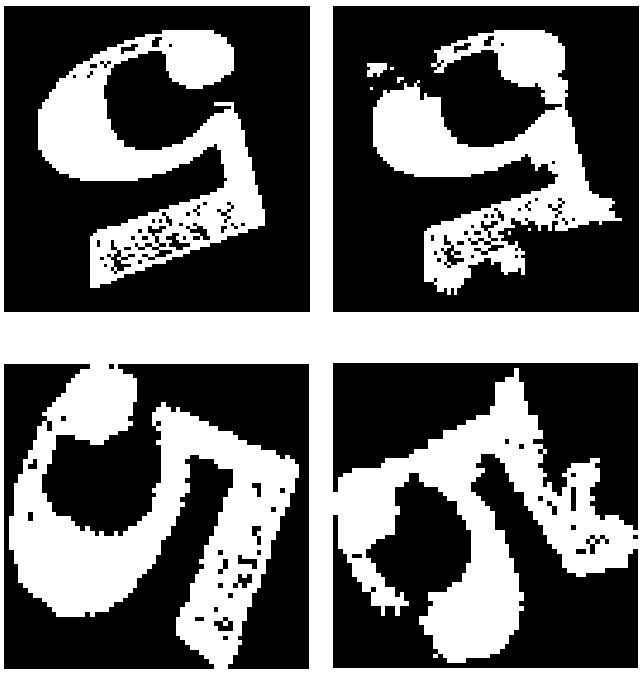


Figure 3.7. Angular normalization by the orientation of shapes having eccentricity near 0. The almost identical shapes on (a) and (b) are rotated to (c) and (d) respectively.

Orientation is a value within $(-\pi, \pi)$; thus, rotation by the orientation provides invariance to rotation by $k * \pi$, resulting in two distinct possibilities. To make the rotation unambiguous, the shape is rotated by π if the center of mass of the shape is located on the right side. Note that mathematical orientation may significantly deviate from the orientation value estimated by a human observer.

The orientation provides ambiguous results if the eccentricity is close to 0, thus for rounded shapes. In these cases, a slight difference in the shape can result in a significant difference in the orientation, as shown in Figure 3.7.

Another angular basis could be considered as well. Most of the graphic patches have a natural, human-estimated orientation; however, that is not a computable value in most cases. Another characteristic angle is the declination of the line connecting the centroid of the shape with the farthest point of the shape, i.e., the maxima of the shape signature defined as the distance from the centroid to the edge points depending on the angle. [122]. This method provides ambiguous results for the rounded object as well, and its more sensitive to noise compared to the mathematical orientation. In [28], several other orientation descriptors are presented.

To achieve scale-invariant shape analysis, the shape is normalized to fit in a window sized 64 × 64 pixels, preserving the original aspect ratio. It has been shown earlier that using larger sizes is unnecessary. Due to angular normalization, the shape entirely fills the horizontal space; thus positioning is limited only to the vertical alignment, where the shape is moved to have the same distance between the borders and the square box on the two sides.

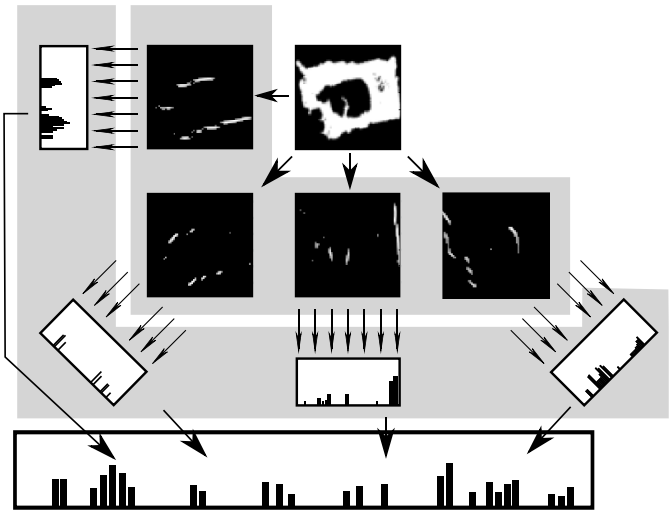


Figure 3.8. Construction of the EPPSED feature vector. Edges are detected in four directions; then thresholding and maxima selection are applied; finally, projections are concatenated and normalized.

DOI:10.15774/PPKE.ITK.2019.005

I summarized the construction of the EPPSED feature vector by Pseudocode 1 and Figure 3.8.

PSEUDOCODE 1

$$EV(\rightarrow)=\begin{bmatrix}0&0&0&0&0\\1&1&1&1&1\\0&0&0&0&0\\-1&-1&-1&-1&-1\\0&0&0&0&0\end{bmatrix}\qquad EV(\searrow)=\begin{bmatrix}0&1&0&0&0\\-1&0&1&1&0\\0&-1&0&1&0\\0&-1&-1&0&1\\0&0&0&-1&0\end{bmatrix}$$

$$EV(\uparrow)=\mathrm{rot90}(EV(\rightarrow))\qquad EV(\nearrow)=\mathrm{rot90}(EV(\searrow))$$

```
function EPPSED(S)
      N := 64
      % preprocessing
      rotate(S, -orientation(S))
      resize(S, [N, N], fit)
      if horizontal_mass_center(S) > N/2 then
            rotate(S, 180)
      end

      directions := [↑, ↗, →, ↘]
      % generate edge maps (EM) for every direction
      for dir in directions
            EM(dir) := convolution2d(S, EV(dir))
      end
      % for every location threshold the edge maps
      for i in [1..N], j in [1..N]
            for dir in directions
                  θ := max_dir2 in directions(EP(dir2)[i, j])
                  EMT(dir) := t_soft(EM(dir), θ)
            end
      end

      % project thresholded edge maps and scale them
      for dir in directions
            PR(dir) := histogram(EMT(dir))
            scale(PR(dir), N/4)
      end
      return EPPSED := [PR(↑), PR(→), PR(↗), PR(↘)]
end
```

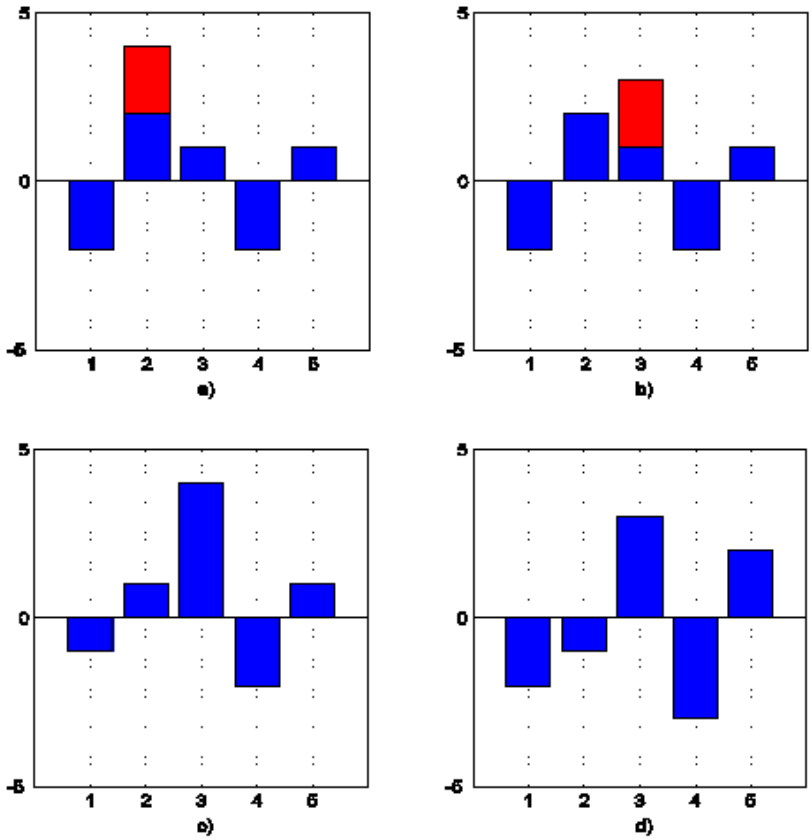


Figure 3.9 Vectors a) and b) differ only in values highlighted in red color, while vectors on c) and d) differ in all dimensions. Difference between a) and b) is typical for topological features, where a small difference in the input results in an offset in the feature vectors. Manhattan and Euclidean distance between pairs a)-b) and c)-d) are the same, but the topological distances are smaller for the first pair.

3.4 Metric for the EPPSED space

Metric defined on feature set plays an important role in classification, an inappropriate metric selection may significantly reduce classification accuracy. I present a new metric designed for topological and chronological features, and globally for that kind of vectors, where a small difference in the input results in an offset between close dimensions of the feature vector. In these cases, a local difference represents a lower global error if a difference with opposite sign is also present in close dimensions (see Figure 3.9).

For the vector class described above, I developed a new metric group. Algorithms are based on the standard Euclidean metric, but local errors are summed up to the global error. The essence of the developed metrics is that a difference in one dimension (thus in case of the shape description a pixel location) generates an error offset that is inversely signed and declines exponentially with respect to the distance of the location. The local error is the minima of the standard difference and the difference corrected by the offset. Finally, adjusted

DOI:10.15774/PPKE.ITK.2019.005

error values are summed up to the global error. The difference between developed algorithms is in the way of generating and cumulating the error offset. Pseudo codes of the algorithms are shown in Pseudocode 2.

PSEUDOCODE 2. TOPOLOGICAL METRICS

```
gd = 0;
off = 0;
for i=1:n
    d = fv1(i) - fv2(i);
    e = d - off;
    if abs(e) <= abs(d)
        err = e;
        off = -e;
    else
        err = d;
        off = l*off;
    end
    gd = gd + err^2;
end
return sqrt(gd);
```

a) reset error

```
gd = 0;
off = 0;
for i=1:n
    d = fv1(i) - fv2(i);
    e = off - d;
    off = l*e;
    gd = gd + err^2;
end
return sqrt(gd);
```

b) simple fading

```
gd = 0;
off = 0;
for i=1:n
    d = fv1(i) - fv2(i);
    e = d - off;
    if abs(e) <= abs(d)
        err = e;
        off = -e;
    else
        err = d;
        if abs(d) <= abs(l*off)
           off = - d;
        else
           off = l*off;
        end
    end
    gd = gd + err^2;
end
return sqrt(gd);
```

c) conditional reset

```
gd = 0;
off = 0;
for i=1:n
    d = fv1(i) - fv2(i);
    e = d - off;
    if abs(e) <= abs(d)
        err = e;
        off = -d;
    else
        err = d;
        off = l*off - d;
    end
    gd = gd + err^2;
end
return sqrt(gd);
```

d) reset difference

Code 1. Pseudo codes of topological metric computing algorithms. Input vectors are `fv1` and `fv2`, `gd` stands for global distance, `d` for local distance in one dimension, `e` for offset distance in the dimension, `err` is the local error. `off` denotes local offset, `l` is the fading constant. Note that computed distances are signed.

I tested the developed metrics and compared to the Euclidean metric, and results are shown in Table 3.1. The tests showed that no significant change appears in the values of the precision. The higher recall values could reach the *simple fading* and the *reset difference*

DOI:10.15774/PPKE.ITK.2019.005

algorithms. Since the *simple fading* algorithm is more straightforward than the *reset difference,* I prefer the usage of the first one.

I performed several tests with different fading constants, and the best accuracies were achieved with fading constant *l=0.3*.

TABLE 3.1. PERFORMANCE OF TOPOLOGICAL DISTANCE METRICS

	Test set A		***Test set B"***		***Test set D***		***Test set E***	
	precision	*recall*	*precision*	*recall*	*precision*	*recall*	*precision*	*recall*
euclides	99.88%	61.11%	100%	66.38%	99.7%	58.26%	99.75%	90.72%
reset error	99.88%	60.83%	100%	66.03%	99.59%	57.98%	99.95%	92.37%
simple fading	99.78%	62.41%	100%	66.97%	99.71%	58.89%	99.95%	92.94%
reset difference	99.88%	61.59%	100%	66.88%	99.6%	59.23%	99.89%	92.87%
conditional reset	99.88%	60.83%	100%	66.03%	99.59%	57.98%	99.95%	92.37%

The deficiency of methods shown above is that they are not symmetric. My measures showed that global results are not influenced by reversing the order of computation, but minimal differences appeared in distances measured in reversed order.

DOI:10.15774/PPKE.ITK.2019.005

4 The Adaptive Limited Nearest Neighborhood classification in a two-level classification

Adapted to the structure of the GSPPED descriptor I propose a two-step classification method that adapts to the compound characteristics of the GSPPED descriptor, but it can be used in general as well.

Nearest neighborhood classifiers are typical when using PPED-type descriptors. The GSPPED, as a compound descriptor, enables us to use a special comparison method, since the parts of the vector represent different features. Compound classifiers are frequently used techniques to handle separate parts, but generally, they do not exploit the meaning of each part of the vector.

I suggest a two-step classification scheme that allows using the different parts of the descriptor individually. Shape classification is performed by comparison of the descriptor to labeled points in the feature space denoted as templates. In the first step global and statistical features are compared, then, if a satisfactory match is achieved, the final decision is computed from the differences between the contour features.

I call the set of templates used for comparison as the *representative set*, which is a subset of the training set. Every template in the training set is labeled by its semantic class. Depending on the task, several classes are chosen as relevant ones, whereas every input vector outside of these is considered non-relevant (non-relevant classes). Although the non-relevant subset typically comprises many classes, it can be handled as a single class due to the lack of need to differentiate among them.

4.1 Filtering

The first phase of the decision selects candidates from the representative set for the second phase by rejecting obviously dissimilar template vectors. An input descriptor matches the labeled template vector if the number of elements with a difference higher than the threshold is under a certain limit.

$$comparison(f,t) = \begin{cases} match, & if\ E(f,t) \le th_G \\ reject, & if\ E(f,t) > th_G \end{cases}$$

$$E(f,t) = \sum_{i \in filters} e_i(f,t)$$

$$e_i(f,t) = \begin{cases} 1 & if\ |f_i - t_i| \le th_i \\ 0 & if\ |f_i - t_i| > th_i \end{cases}$$

f is the input shape feature, t is the template vector, th_G is the global filtering threshold, and th_i is the threshold for the i^{th} feature used for filtering.

Other definition of $e_i(f,t)$ can also be considered with continuous error values, for the sake of simplicity and simple computation I chose a discrete function. The threshold values th_G and $\boldsymbol{th}$ were determined based on preliminary measurements and genetic algorithm results. The fitness value of a filter-vector $\boldsymbol{z}$ was chosen as follows:

$$f(\boldsymbol{z}) = \sum_{x \in R} -penalty(x, \boldsymbol{z})$$

$$Penalty(x, \boldsymbol{z}) = \begin{cases} 0 & if\ C(x) = \widetilde{D}(x, \boldsymbol{z}) \\ 1 & if\ C(x) \neq \widetilde{D}(x, \boldsymbol{z})\ and\ \widetilde{D}(x, \boldsymbol{z})\ is\ not\ relevant \\ P & if\ C(x) \neq \widetilde{D}(x, \boldsymbol{z})\ and\ \widetilde{D}(x, \boldsymbol{z})\ is\ relevant \end{cases}$$

$$\widetilde{D}(x, \boldsymbol{z}) = \widehat{D}_R(x)\ using\ filters\ \boldsymbol{z}$$

$C(x)$ is the class of x, and $\widetilde{D}(x, \boldsymbol{z})$ is the predicted class of element x from parameter set R using the filter vector $\boldsymbol{z}$. The false-positive penalty value P represents the priority between the precision and recall. If P>1 the precision is prioritized, and if P<1 the recall is maximized.

The resulting filter values are denoted $\boldsymbol{z}^*$, and were computed using a genetic algorithm, with fitness function defined above with P=50, in 200 epochs and population size of 100 individuals, on parametrization set P and test set T, and compared to other set tests as well. The details of the shape image sets are described in Section 5.

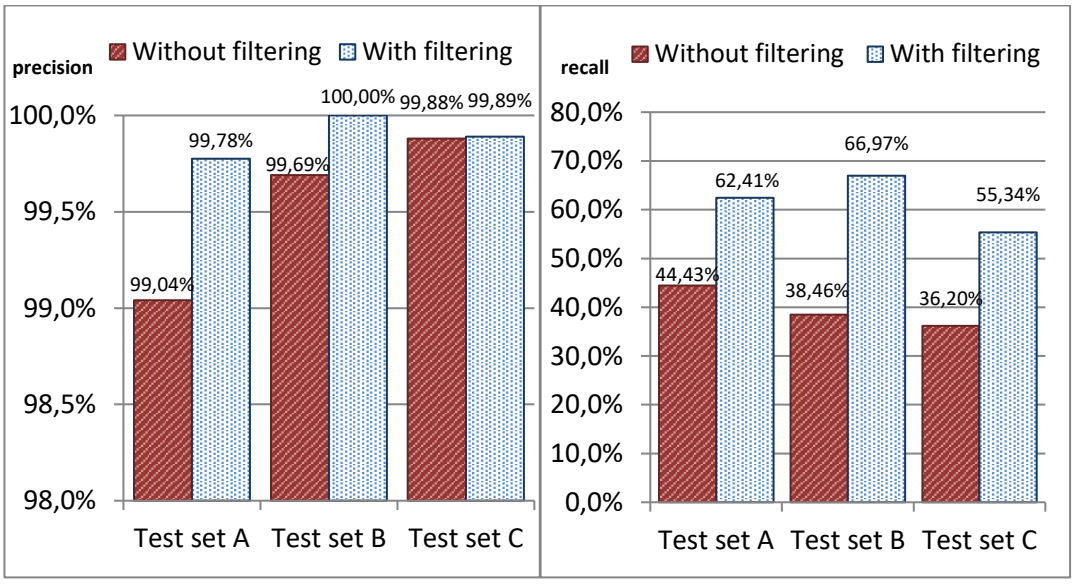


Figure 4.1. Classification precision (left) and recall (right), tested on three different test sets A, B, and C, depending on the usage of filtering phase.

However, filtering not only results in a slight increase in precision, but I could achieve significantly higher recall rate (see Figure 4.1).

The explanation of the anomaly is the consequence of the second phase of the classification explained in Section 4.2. The Adaptive Limited Nearest Neighborhood model learns the limits of acceptance also on filtered results and maximizes the classification precision. Assuming that filtering is based on a data orthogonal to the data used in the second phase, it might filter out templates that in the second phase would determine a lower acceptance radius for some instance. To verify the hypothesis, the frequency of acceptance radius lengths was measured in the function of the usage of filtering on the same representative set.

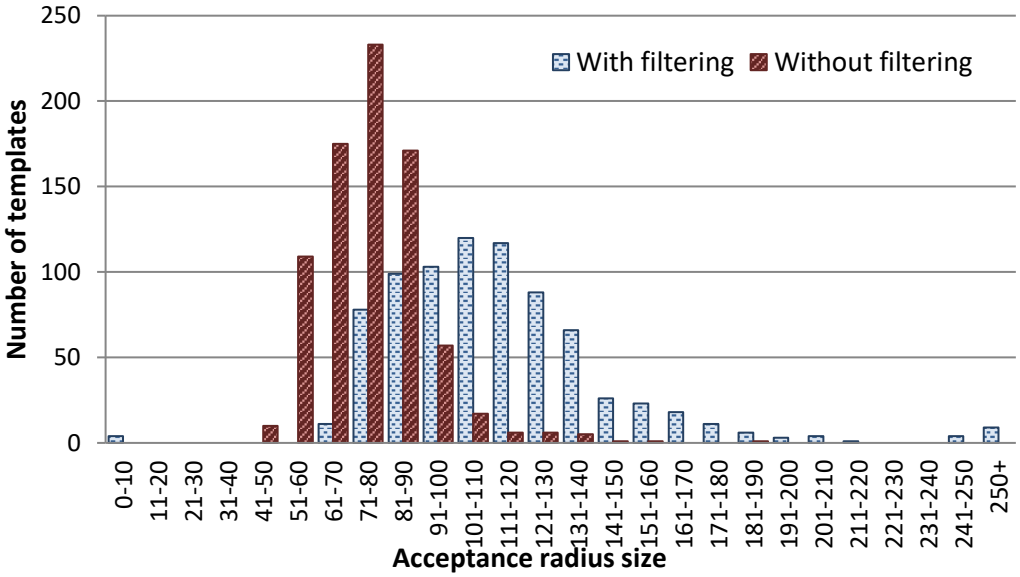


Figure 4.2. The frequency of the acceptance radiuses in case of filtering (blue dotted) and without filtering (red lined). Filtering allows the acceptance radiuses to take higher values in average, but even zero (if no other comparable elements remain after filtering) and overly high values also (if only a few and distant templates remain to compare)

As it is seen in Figure 4.2, filtering allows higher acceptance radii resulting in better recall in the final classification. The mean of the acceptance radii is 113.4 if filtering is applied. Without filtering the mean is reduced to 75.35, and only a few representative instances have higher radius than 110. (Radii shown in this paragraph are distances in the EPPSED feature space. Typically, the values of each dimension are in the interval from 0 to 200.)

DOI:10.15774/PPKE.ITK.2019.005

I also measured the speed of classification that depends on the number of comparisons to the template vectors. Filtering reduces the average lookup time by 85-95%.

TABLE 4.1. RECALL DEPENDING ON THE FILTERING.

	*z**	*std*	*std/2*	*std*2*	no filtering
Test set A	**62,41%**	50,82%	54,23%	47,61%	44,43%
Test set B	**66,97%**	60,29%	62,72%	59,02%	38,46%
Test set C	**55,34%**	48,91%	47,66%	47,47%	36,20%

Filter values were computed to fit one actual shape set; thus these values may not be suitable for other sets. Since generated values are in the same order of magnitude with the standard deviation of the measured moments on the training set, I tested the standard deviation (as well as the half and the double of the standard deviation) on the same test sets. Results are summarized in Table 4.1, where z^* denotes the filter vector obtained from the genetic algorithm, ***std*** denotes the standard deviation of the relevant classes. Precision does not depend on the filter values; recall is significantly lower using the standard deviation compared to the z^*; however they are clearly higher than without filtering.

4.2 The Adaptive Limited Nearest Neighborhood classification

The second phase of the classification defines the final class of the input employing the Limited Nearest Neighborhood classification method. The reason to choose nearest neighborhood classifier (NN) is due to its suitability to be implemented on dedicated VLSI architecture; it can be easily learned and extended with new knowledge by inserting new representative instances.

As described in the Introduction, an essential property of the nearest neighborhood method is that there is always a nearest element to every input vector if no additional constraints are specified. This can be a disadvantage in some cases if the distance between the closest element and the input is high.

The inability to reject a hypothesis results in a type I error when remote parts of the input space are not covered with training instances. To make the classifier able to maximize the precision, I propose an Adaptive Limited Nearest Neighborhood method (AL-NN) that allows the rejection of irrelevant inputs Y by defining an acceptance radius individually for all training instances. ($C(Y) \in \mathcal{C}_{NRel}$, where $\mathcal{C}_{NRel}$ is the set of irrelevant classes and $\mathcal{C}_{Rel}$ is the set of relevant classes.) By setting an upper bound for the distance of an input and a representative instance, we can limit the set of inputs that may get classified to the

corresponding class to a hypersphere in the feature space, that we call the acceptance region of the instance.

Acceptance regions of different shapes can also be considered. If the dimensions can be typically regarded independent, and the noise is not significant, the usage of a hypercube as an acceptance region is justifiable. Employing a hyper-ellipsoid allows different limits in different dimensions, but it is effective only in case of low dimensional spaces. Since we work with a 64 dimensional feature vector, the degree of freedom would be impractically high. Additionally, in the proposed edge-based shape description, dimensions are typically equivalent as the amount and distribution of noise is the same in all dimensions, dimensions are weakly dependent of each other, and we expect a similar tolerance in all dimensions; thus we opted to use hyperspheres as acceptance regions.

Using the same radius for all representative instances would be computationally easier, but it would result in a disproportionately large representative set to represent in-class regions, and also boundary regions with the same radius. Furthermore, irregular boundaries might increase the inefficiency of the cover if the radius is determined based on the radius of the highest curvature.

4.3 Learning

The acceptance radius indicates the extension of the class, the region in the feature space where the characteristics of the instance are valid. The clues in determining the acceptance radius as a boundary measure for a representative instance are the closest known instances that belong to another class and the instance with the maximal distance that belongs to the same class. In case of a relevant sample, it is worthwhile to differentiate relevant instances from another classes and irrelevant instances to make the representation more flexible.

I define the set of all irrelevant examples (N), and for every example, I define the set of other instances of the same class ($SP(x)$), and the set of instances of all other relevant classes ($OP(x)$):

$$OP(x) = \{y \mid y \in R, C(y) \in \mathcal{C}_{Rel}, C(y) \neq C(x)\}$$

$$N = \{z \mid z \in R, C(z) \in \mathcal{C}_{NRel}\}$$

$$SP(x) = \{w \mid w \in R, C(w) = C(x)\}$$

DOI:10.15774/PPKE.ITK.2019.005

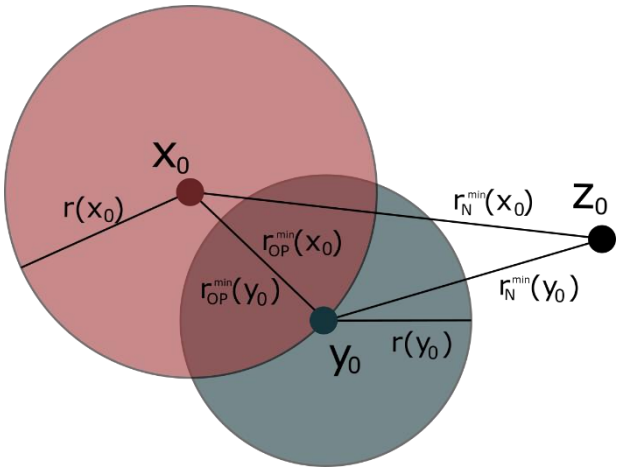


Figure 4.3. Definition of the acceptance range. x_0 and y_0 represent relevant elements from different classes, z_0 the closest irrelevant element. Acceptance threshold for y_0 ($r(y_0)$) is set to the half of the distance to the closest irrelevant element ($r_N^{min}(y_0)$). Acceptance threshold for x_0 ($r(x_0)$) is chosen as the distance to the closest element of another class ($r_{OP}^{min}(x_0)$). Since z_0 is an irrelevant template, it is not included in the representative set, thus no acceptance region is defined for it.

To be able to handle the three cases in a unified manner a partial acceptance region function is introduced ($r_A^\lambda(x)$), that expresses the threshold for a given set A and a given threshold function λ:

$$r_A^\lambda(x) = \begin{cases} \lambda(\{\, d(x,v) \mid v \in A(x)\}) & if\ A(x) \neq \emptyset \\ \infty & if\ A(x) = \emptyset \end{cases}$$

The final acceptance radius will be the smallest of the partial acceptance radiuses. An example x from the training set R, from the class $C(x) \in \mathcal{C}_{Rel}$ will get an acceptance radius $r(x)$ (Figure 4.3):

$$r(x) = \nu \cdot \min(\eta_{OP} \cdot r_{OP}^{min}(x), \eta_N \cdot r_N^{min}(x), \eta_{SP} \cdot r_{SP}^{max}(x))$$

where ν serves as a shared vigilance parameter, which affects how cautious do we want to be, and can be used to move on the precision-recall trade-off curve.

In the experiments I used $\eta_{OP} = \eta_{SP} = 1$, because this safely excludes other relevant samples from the acceptance region but still tries to include as many samples from its own class as possible. For the irrelevant classes, I have set the threshold parameter η_N more conservatively to 0.5, so as to enable the omission of the irrelevant elements from the representative set, as this choice does not allow acceptance regions to intersect with acceptance regions of irrelevant samples. Thus if an input is not within the acceptance region of any relevant elements, then it is refused as being a non-relevant input. With these

parameter values, setting $\nu = 1$ results in a strong preference for a high precision over a high recall rate. In this thesis, where it is not specified, I used $\nu = 1$.

Finding the optimal representation set in general is hard; hence it is important that slight overrepresentations do not degrade the recall rate and thus the generalization capability of the model does not change considerably. This is satisfied by the formula proposed above, as it does not let the radius of the acceptance region decrease if a new instance of the same class is added to the representative set.

4.4 Representative set optimization

Another major disadvantage of the Nearest Neighborhood classification is that the manually built training/representative set might be disproportionately large, making the classification very slow.

The representative set can be optimized by eliminating unnecessary points so that the resubstitution results do not change significantly on the training set. Omitting points may lead to a small decrease of the cover, but most of the omissions can be regarded as noise filtering, thus making the model eventually more robust.

Selection of unnecessary points can be carried out based on the analysis of the representative set, by minimizing the set size, while preserving approximately the same cover. A template **Y** is unnecessary (U denotes the set of unnecessary elements) from the aspect of classification if the classification result remains the same for all the points of the space (i.e., for an arbitrary input) if Y is removed from the representative set:

$$U = \{\boldsymbol{x} \in F: D_R(\boldsymbol{x}) = D_{R\setminus\{Y\}}(\boldsymbol{x})\}$$

where F is the feature space, and $D_R(\boldsymbol{x})$ is the decision for feature vector $\boldsymbol{x}$ using the model learnt by representative set $\mathcal{R}$.

In the Nearest Neighborhood model, the class is determined by the nearest labeled point. A representative instance Y is unnecessary if, for every point in the feature space that is classified to Y as the closest template point, the second closest template point belongs to the same class as Y.

$$Y \in U$$

$$\leftrightarrow$$

$$\forall \boldsymbol{x} \in \boldsymbol{F}\ \exists \boldsymbol{Z} \in \boldsymbol{R}\setminus\{\boldsymbol{Y}\}\ \forall \boldsymbol{w} \in \boldsymbol{R}\setminus\{\boldsymbol{Y}\}:$$

$$d(\boldsymbol{x},\boldsymbol{Y}) \leq d(\boldsymbol{x},\boldsymbol{w}) \ \rightarrow\ d(\boldsymbol{x},\boldsymbol{Z}) \leq d(\boldsymbol{x},\boldsymbol{w})$$

where $d(\boldsymbol{x},\boldsymbol{y})$ is the distance between points **x** and **y.**

DOI:10.15774/PPKE.ITK.2019.005

The boundary surface *B* between classes is the set of points that are equally distant from support vectors of different classes. A template point **Y** is unnecessary if it does not influence the boundary surfaces. In an n-dimensional feature space such a boundary surface is n-1 dimensional, and apart from the singular cases when points lie in one hyperplane, complete shadowing of **Y** can be achieved with at least n necessary points of the same class. Therefore, if the number of representative points of a class and the dimension of the feature space is the same order of magnitude, only a negligible portion of the representative set can be unnecessary.

In the Adaptive Limited Nearest Neighborhood method proposed in Section 4.3, acceptance regions of each representative instance provide a reasonable estimate of their contribution to the global cover. (see Figure 4.4).

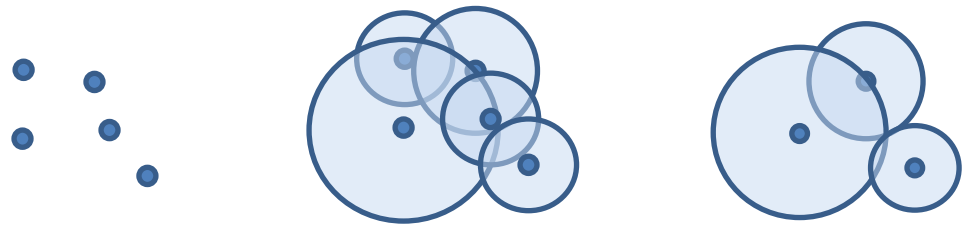


Figure 4.4 Representative set optimization in the AL-NN model. a) Unnecessary elements of a representative set in feature space can hardly be selected without additional information. b) Acceptance regions outline the covered parts of the feature space and redundantly represented regions as well. c) After optimization, the recall may reduce.

I propose an iterative optimization algorithm for the Adaptive Limited Nearest Neighborhood classification that reduces the mutual cover of the representative set elements. As initialization the points of the representative set are ordered in a queue P. The set S is initialized as empty:

$$P := (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots \boldsymbol{x_n}),$$

$$S := \emptyset$$

The first element of *P* is taken out from *P* and moved to *S*, and all other instances are removed from *P* that are covered by it.

$$p := P[1]$$

$$S := S \cup \{p\}$$

$$remove\ \{x \in P \mid C(p,x) = 1\}\ from\ P$$

The iteration ends when *P* is empty, and the *S* will be the reduced representative set. Note that *P* is an ordered set; thus different permutations will result in different performance. In the next paragraph, I will show three permutations and the corresponding performance results.

- **Ordering based on the number of representative covers**

$$P \coloneqq (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots \boldsymbol{x_n}), \qquad \forall x_i, x_j \;\; i < j \;\rightarrow c(\boldsymbol{x_i}) < c(\boldsymbol{x_j})$$

$$H_m(\boldsymbol{x_i}) = \sum_{\substack{j=1 \\ j \neq i}}^{n} h_m(\boldsymbol{x_i}, \boldsymbol{x_j})$$

$$h_m(x_i, x_j) = \begin{cases} 1, & if\ d(\boldsymbol{x_i}, \boldsymbol{x_j}) \leq m \cdot r(\boldsymbol{x_i}) \\ 0, & if\ d(\boldsymbol{x_i}, \boldsymbol{x_j}) > m \cdot r(\boldsymbol{x_i}) \end{cases}$$

$r(\boldsymbol{x_i})$ is the acceptance radius of $\boldsymbol{x_i}$, and *m* is the cutoff ratio.

- **Ordering based on the acceptance radius**

$$P \coloneqq (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots \boldsymbol{x_n}), \qquad \forall x_i, x_j \;\; i < j \;\rightarrow r(\boldsymbol{x_i}) > r(\boldsymbol{x_j})$$

- **Ordering based on covered sum-of-hypersphere-intersection-ratios**

$$P \coloneqq (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots \boldsymbol{x_n}), \qquad \forall x_i, x_j \;\; i < j \;\rightarrow v(\boldsymbol{x_i}) < v(\boldsymbol{x_j})$$

$$v(\boldsymbol{x_i}) = \sum_{\substack{j=1 \\ j \neq i}}^{n} Vr(\boldsymbol{x_i}, \boldsymbol{x_j})$$

$$Vr(x_i, x_j) = \frac{V(G(x_i) \cap G(x_j))}{V(G(x_i))}$$

$V(G)$ is the volume of hyperbody G,

$G(x)$ is a hypersphere with centroid *x* and radius of $T(\boldsymbol{x_i})$

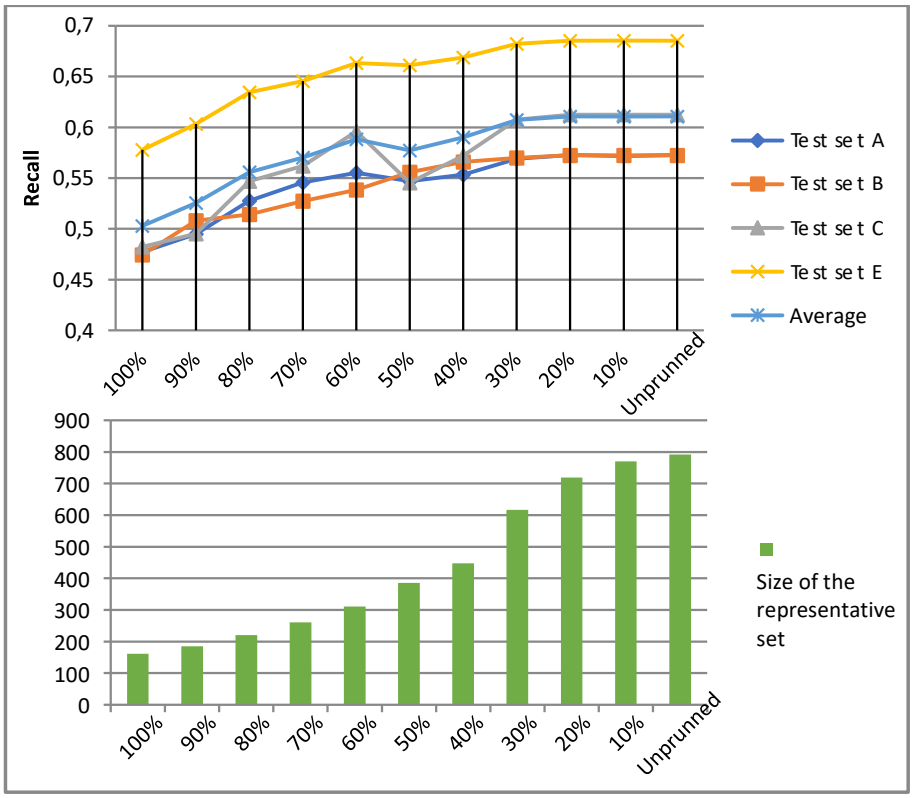


Figure 4.5 Recall and the size of the representative set in the function of the cutoff size for the ordering based on the representative cover

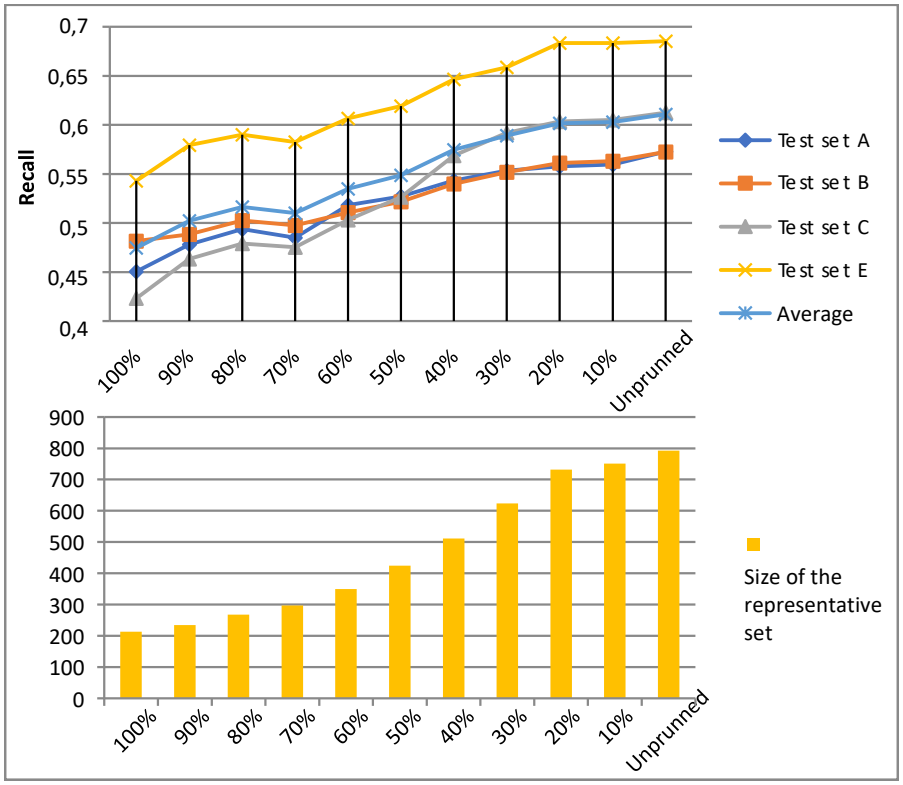


Figure 4.6 Recall and the size of the representative set in the function of the cutoff size for the ordering based on the acceptance ratio

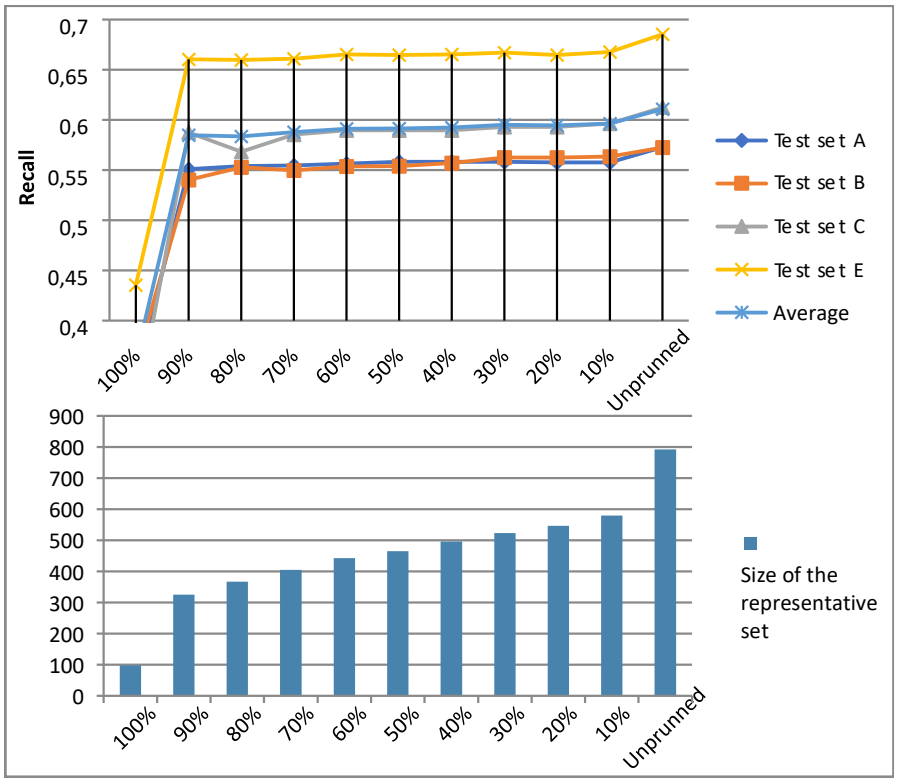


Figure 4.7 Recall and the size of the representative set in the function of the cutoff size for the ordering based on the sum-of-hypersphere-intersection-ratios

I tested the optimization algorithm from cutoff ratio m=1 to m=0, where m=1 stands for deleting any covered representative element, and m=0 stands for the unpruned model where even identical elements may remain in the set.

Results show that pruning based on representative cover (Figure 4.5) and the one based on the sum-of-hypersphere-intersection-ratios (Figure 4.6) overperform the pruning based on the acceptance radius (Figure 4.7), and since computing volumes of the hyperspheres and their intersection is orders of magnitude more complex than to run a single classification, I suggest to use the first one.

In more details, in the case of pruning based on the representative cover, from m=1 to m=0.5 the recall increases from 0.5 in average to 0.6 nearly linearly, and from m=0.6 to 0 only a small increase can be noticed. Almost parallel to that the size of the reduced representative set increases slightly to m=0.4 and after a significant increase saturates after m=0.2.

DOI:10.15774/PPKE.ITK.2019.005

4.5 Extending the representative set

Adaptive extension of a learned AL-NN model can be carried out efficiently by inserting a new point to the representative set and setting the acceptance radius of the inserted element based on the original training set. The extension can bring higher recall rates by covering previously uncovered regions of the feature space.

The primary challenge in extending the representative set is to select new instances to be inserted adequately. On the one hand, to cover new areas, a candidate has to be far from the existing representative elements. On the other hand, an automatic update should only insert elements that are classified correctly with reasonably high confidence, that is, ones close to a labeled point. If both conditions hold, I declare the insertion of the new instance to be safe.

As I showed in Section 4.2, the acceptance regions clearly bound the coverage in the representative set; thus both conditions can be formalized based on acceptance thresholds. The real challenge in selecting new instances is that the two conditions are contradictory. To resolve the contradiction that the distance of the candidate sample should be low (for high confidence) and high (to gain significant coverage) at the same time, we rely on temporal information.

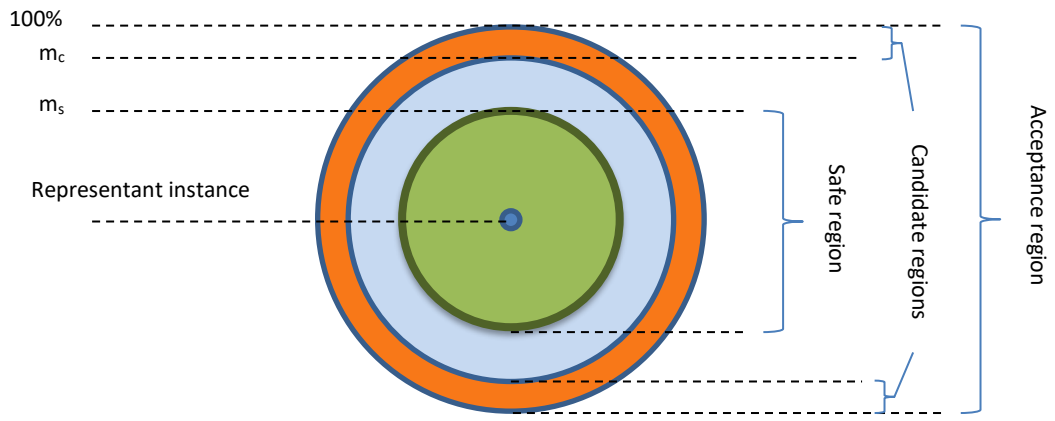


Figure 4.8 Acceptance region of a representative instance in the feature set

I developed an automatic extension algorithm for the AL-NN model, which uses temporal information in the update method, if available. A decision is defined to be safe if a test set element is closer to the representative example than the half of its acceptance threshold.

$$d(\boldsymbol{t}, \boldsymbol{c}) \leq m_s r(\boldsymbol{t})$$

$$m_s = 0.5$$

DOI:10.15774/PPKE.ITK.2019.005

The choice of the value $m_s=0.5$ was based on the quick decision (presented in Section 4.2) threshold.

An element is chosen as a candidate for insertion if it is at the edge of the acceptance region.

$$d(\boldsymbol{t}, \boldsymbol{c}) \geq m_s r(\boldsymbol{t})$$

A candidate is only inserted to the representative set if neighboring frames contain patches that were classified in the same class with a safe decision. (Figure 4.8)

TABLE 4.2. RESULTS OF THE ONLINE LEARNING ALGORITHM BASED ON THE NEIGHBORING FRAMES

$\mathbf{m_c}$	**# of added instances**	**# of new recognized instances**
0.5	91	35
0.75	24	15
0.9	8	15
0.95	4	6

The radius of neighboring frames is chosen based on the processing framerate and the median translation of the image. In the shape set I used, the total processing time is between 0.1 to 0.3 second, while the images were taken by a cell phone camera moving slightly upon a table; thus the frame radius was set to involve only directly neighboring frames.

I tested the extension with $m_c=0.5$ to $m_c=0.95$. I added new elements from three different test sets (test sets A, B, and C) and measured the improvement on an independent test set (test set D). Results are summarized in Table 4.2. Details of the test sets are described in Section 5.

Results show that the most valuable new instances are the distant ones, with $mc \leq 0.9$. However, candidate instances that neighbor a safe instance is very rare.

DOI:10.15774/PPKE.ITK.2019.005

5 Experimental results

The GSPEED and the multi-level classifier including the Adaptive Limited Nearest Neighborhood classifier presented in this thesis have been tested in the framework of the Bionic Eyeglass, on shape patches obtained from Hungarian Forint banknotes.

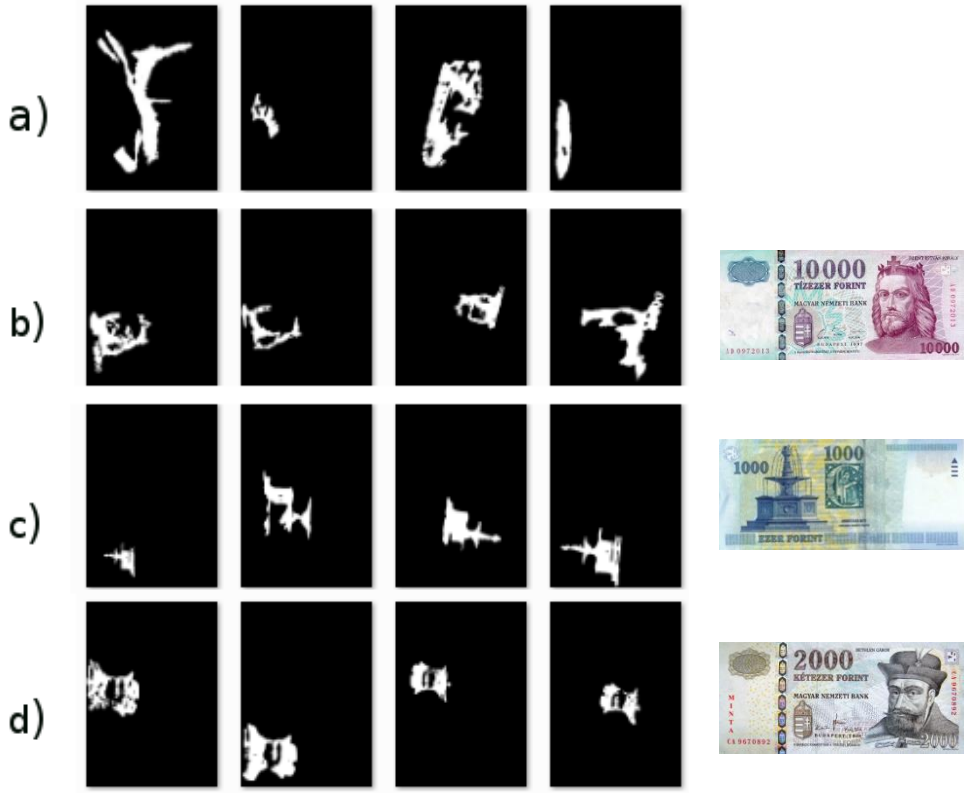


Figure 5.1 Fragment of the test sets. In the row a) are shapes from irrelevant classes, rows b)-d) show relevant shapes of banknotes, the source banknotes are shown next to the shape images.

The five shape datasets contain several thousands of shape images, including irrelevant inputs that do not belong to any class. The GSPPED was extracted in an average of 29.5 milliseconds on a standard computer (Core2 Quad CPU @ 2.66 GHz, 4 GB memory). The properties of the test sets are indicated in Table 5.1.

DOI:10.15774/PPKE.ITK.2019.005

TABLE 5.1. TRAINING AND TEST SETS

database name	number of instances	number of classes	source
Test set A	7008	9	live test with visually impaired
Test set B	6482	9	live test with visually impaired
Test set C	6171	9	live test with visually impaired
Test set D	13895	9	live test with visually impaired
Test set E	13113	9	live test in the laboratory
Test set F	1500	17	live test with visually impaired and generated abstract patches
Parametrization set P	1500	17	live test with visually impaired and generated abstract patches

The representative set was produced from a training set; the initial representative set contained 792 shapes. Shapes in the test sets represent characteristic graphical patches (portraits and drawings) extracted from banknotes, sets F and P contained digits and noised abstract patches as well. All datasets included non-relevant patches also, consisting of shadows, joined patterns, and other noise from the background. The average lookup time was 1.8 ms. Examples of input images are shown in Figure 5.1. Test results are summarized in Table 5.2.

TABLE 5.2. EXPERIMENTAL RESULTS OF THE SHAPE RECOGNITION

database name	F_β	Precision	Recall
Test set A	99,63%	99,78%	62,41%
Test set B	99,88%	100%	66,97%
Test set C	99,69%	99,89%	55,34%
Test set D	99,54%	99,71%	58,89%
Test set E	99,93%	99,95%	92,94%

DOI:10.15774/PPKE.ITK.2019.005

5.1 A comparative test of the AL-NN classifier

Results achieved on test sets A–D (excluding test set E) were compared with other shape descriptions (Complex Zernike Moments and Generic Fourier Descriptor) and classification methods. To allow for different weights for prioritization of precision over recall, AutoMLP, FFNN, and SVM models were trained using a cost matrix with false-positive to false-negative penalty rates ranging from 5 to 100; in the case of the AL-NN, I changed the vigilance parameter from 1.0 to 1.25, with an appropriate adjustment of the filter limit vector $\boldsymbol{z} = \nu^2 \boldsymbol{z}^*$.

First I compared the AL-NN classifier to a feed-forward neural network, an AutoMLP, a KNN model, and a SVM on the shape feature vectors obtained from the GSPPED.

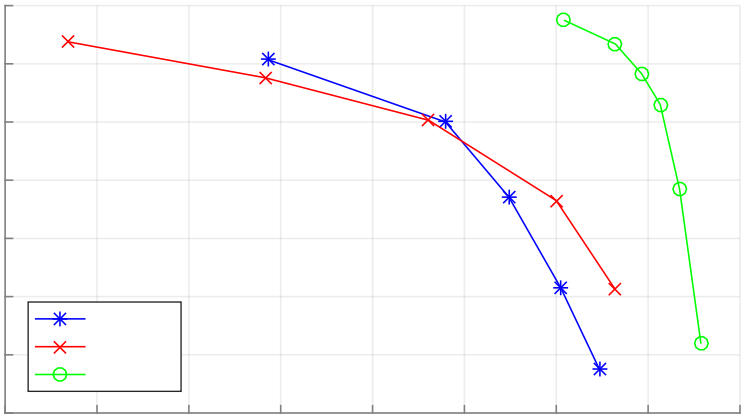


Figure 5.2. Precision and recall of the shape classification by FFNN, AutoMLP and the presented Adaptive-Limited Nearest Neighborhood (AL-NN) classification. The source data is constructed by the GSPPED shape descriptor.

The best results were reached by the neural networks, FFNN and AutoMLP. I tested the FFNN containing 2 to 5 hidden layers and trained from 100 to 1000 epochs. AutoMLP was trained for 20 cycles of 10 generations and 5 MLPs per ensemble in the RapidMiner environment. The best performance achieved by the models are shown in Figure 5.2. Since the SVM (with radial basis function and polynomial kernels) and the KNN (for K from 1 to 10) models could not achieve precision rate above 90% with any parametrization, they are not included in Figure 5.2.

DOI:10.15774/PPKE.ITK.2019.005

5.2 A comparative test of the GSPPED descriptor

In order to investigate the efficiency of the GSPPED shape descriptor, I compared it with the Generic Fourier Descriptor (GFD) and with the Complex Zernike Moments Descriptor (CZMD), trained on the same train set, and using the AutoMLP classifier.

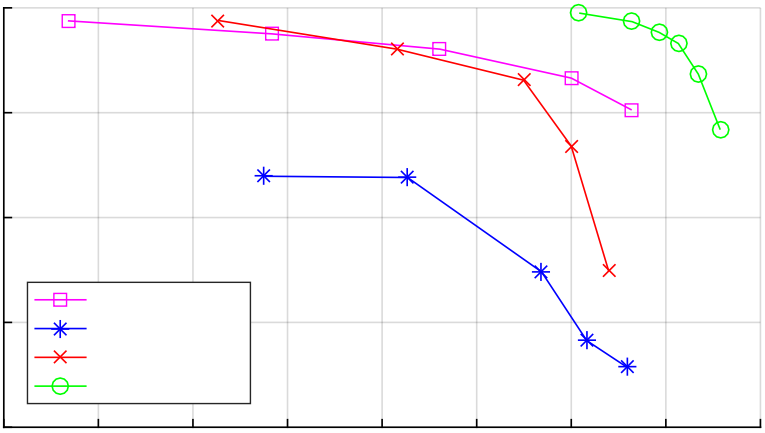


Figure 5.3. Precision and recall of the shape classification, comparing the performance of the Complex Zernike Moments Descriptor, the Generic Fourier Descriptor, and the GSPPED descriptor

The feature vector of the GFD contained 85 elements with an angular frequency of 16 and radial frequency of 4. The CZMD contained 121 feature elements with the highest order of 20. Results are shown in Figure 5.3.

The GSPPED and the Complex Zernike Moments Descriptor evidently outperform the Generic Fourier Descriptor. When classified by the AutoMLP, the GSPPED slightly outperforms the CZMD, however, with high penalty coefficient, the CZMD provides better recall.

I also compared the descriptors based on the McNemar's test. CZMD and GSPPED with AL-NN differ significantly ($p = 1.27e\text{-}4$), and with GSPPED with AL-NN also exceeds the performance compared to GFD significantly ($p < 1e\text{-}15$).

DOI:10.15774/PPKE.ITK.2019.005

5.3 Effect of noise

To measure the sensitivity of the developed shape description and classifier, I repeated the test on noisy images and compared the results with the results obtained with the Complex Zernike Moments Descriptor and the Generic Fourier Descriptor. Based on our datasets I observed that deviations in the extracted shape images do not occur in pixel-level additions or removals, but in joining with other blobs or in the removal of some parts of the shape (also see Figure 5.1). To model this kind of noise, I added and removed several randomly generated blobs to and from the original shape. The total area of the blobs is given as a ratio (w) to the shape area. An example of an artificially added and removed noise is demonstrated in Figure 5.4.

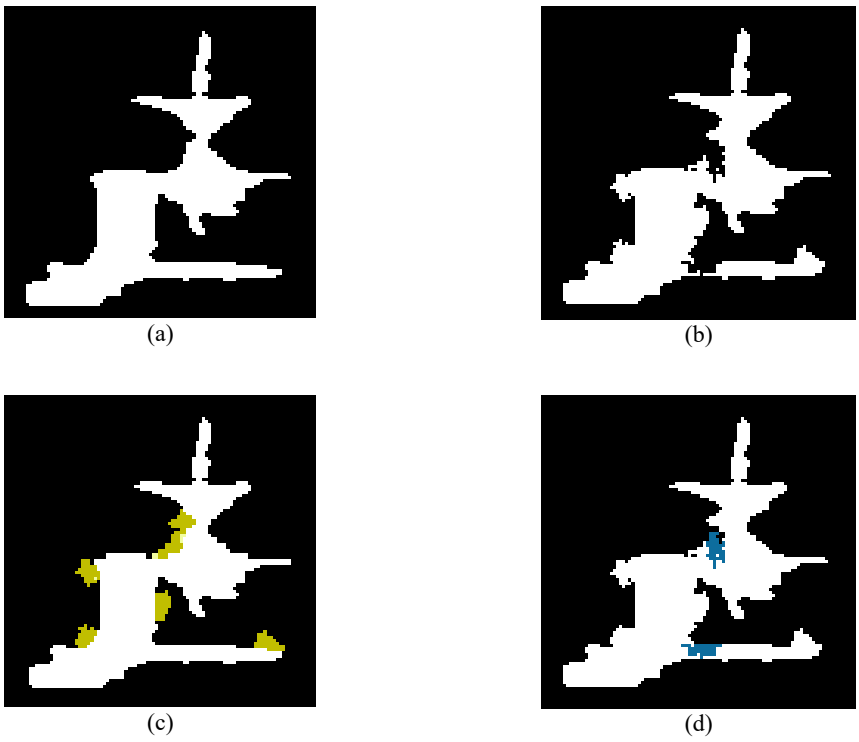


Figure 5.4. Example of blob-level shape noise with manipulation ratio w = 0.2. In a) the original shape is shown, in c) the additions, in d) removals are highlighted, and the final noisy shape is shown in b).

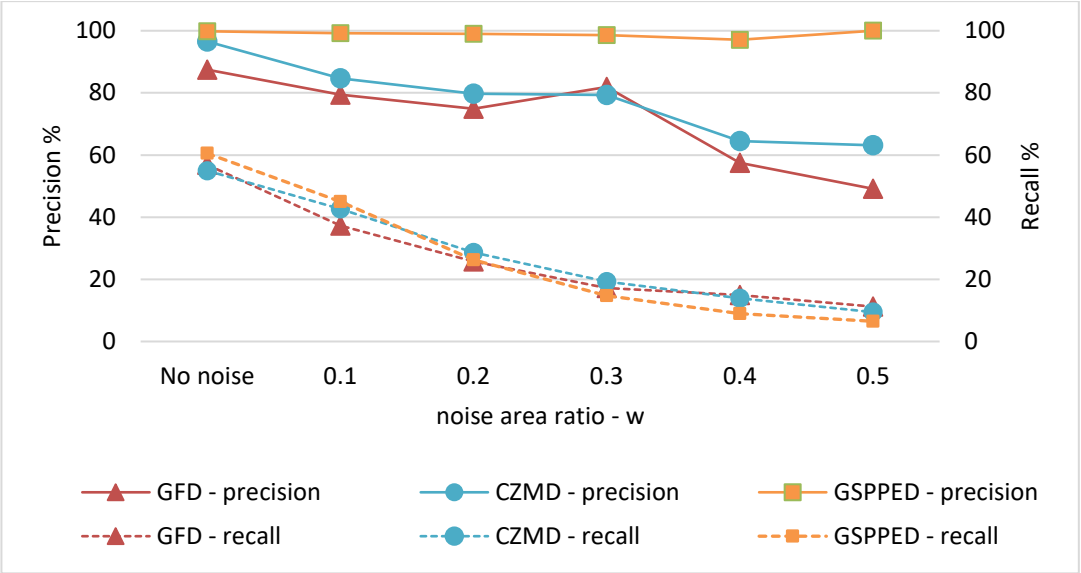


Figure 5.5. Classification recall and precision of the GSPPED, the Complex Zernike Moments Descriptor, and the Generic Fourier Descriptor, depending on the noise ratio w that represents the ratio of the number of manipulated pixels compared to the total area of the shape. The CZMD and GFD features were classified by the AutoMLP algorithm, while GSPPED features were classified with the AL-NN method.

In the case of the CZMD and GFD, results show consistent decrease both in recall and in precision. The GSPPED provides lower recall on high noise ratio than the other two descriptors. However, the precision is significantly higher compared to the CZMD and the GFD (Figure 5.5). These results might highlight the nature of the GSPPED and the AL-NN: its generalization capability is limited, but still comparable to other methods, at the same time it provides outstanding discriminative power.

5.4 Summary

The core idea behind the method presented in this thesis is the two-level description and classification: For an input shape a low-level, global statistical information is extracted to roughly select the set of similar objects and to reject obviously different templates. In the second stage local edge information is investigated to find the closest known shape, but with the ability to reject the match. The refusal is based on the acceptance radius that is specified individually for every item in the representative set according to the properties of the local proximity in the feature set.

Results demonstrate a high precision rate (99.83%) and an acceptable recall rate (60.53%), which fulfill the requirements for a safety-oriented visual application processing an image flow. The reason to have lower cover is that input frames contain highly deformed shapes, which for the sake of reliability, are classified as non-relevant inputs. The recall is

acceptable, as long as a continuous input is available. Compared to other classifiers, none of the tested ones could outperform the AL-NN in precision, and the same recall could only be reproduced with significantly lower precision. If a final decision is made based on multiple input frames and multiple clues, the false positive error can be minimized to practically negligible.

The computation time of the descriptor (~30ms) and the classification time (~2ms) allow real-time recognition even on standard CPUs in computers and phones, and the architecture core of the algorithm is easily adaptable to locally connected cellular array processors. The proposed algorithms were implemented on cell phones and FPGAs with the purpose to provide a reliable vision aid for blind and visually impaired people. One of the drawbacks of the GSPPED descriptor I have found is the high sensitivity to positioning and scaling, depending on minor variations. I will focus on designing and employing a more robust translation and scale normalization method. Together with my research group, I also plan to investigate the possibility of taking more training elements into account when defining the acceptance threshold, similarly to the KNN method. [A1]

DOI:10.15774/PPKE.ITK.2019.005

6 Spin Torque Oscillator Networks

In this section, the programming basics of cellular oscillatory networks are shown. An oscillatory network is a non-Boolean architecture consisting of only interacting oscillators placed on a plain surface, where the program is the function of the input current and the topology of the oscillators. I show that in this environment complex problems can be solved in one program cycle, with significantly lower power consumption and computation time compared to standard architectures.

I present a general topology and two basic ones. Both layouts have the advantages of the cellularity, the precedence of the local neighborhoods. The OCNN (Oscillatory Cellular Nonlinear Network) is a grid-like layout and is used to enhance classification results by performing a transformation in the feature space. The oscillatory network with the pyramidal layout is a binary classification architecture including the preprocessing as well. The topology of the network is determined by genetic algorithms. To show the capabilities of the network, I present a solution for two static classification problems and one for spatiotemporal input.

6.1 Programs in cellular oscillatory networks

Oscillatory networks (ON) generally consists of several oscillators with given physical parameters: the oscillator-dynamics and the coupling interaction between the oscillators. I assume that the position and the physical properties of the oscillators do not vary during the experiment; the interaction with the outer environment is possible only by excitation and measuring of actual state.

When programming an abstract **ON** of n oscillators, the data, the input, output interface, and the program need to be defined. Since the frame-constraints are strict, the next solution seems reasonable:

Oscillatory network **ON** is a set of oscillators on a plain:

$$ON_n\left(o_n, \Theta_n, T_n, \imath_k, o_l\right), \text{ where}$$

o_n are the oscillators of the network,

Θ_i is the type (dynamics) of the o_i oscillator,

T_n is the topology of the oscillators,

DOI:10.15774/PPKE.ITK.2019.005

$\iota_k \subset o_n$ are the input oscillators,

$o_l \subset o_n$ are the output oscillators.

Data $D(ON_n)$ is considered as the individual and relative state of the oscillators: the unsynchronized oscillation (*us*), and if the oscillation is fully synchronized, then the phase-difference. The phase difference is given as the difference of consecutive oscillators in a predefined order, from what all differences can be computed.

$$D(ON_n) = \begin{cases} us \text{ if unsynchronized} \\ \Phi \text{ if synchronized} \end{cases}$$

where

$$\Phi_i = |\varphi_i - \varphi_{i+1}|, i \in [1, n-1]$$

- Input I_k is considered as an initial oscillation frequency on input oscillators ι_n resulted from the excitation on these nodes
- The output O_l is considered as the data of output nodes o_l.
- The program is considered as the T_n topology of the **ON** and the types of oscillators Θ_i determining the dynamics of the individual oscillators

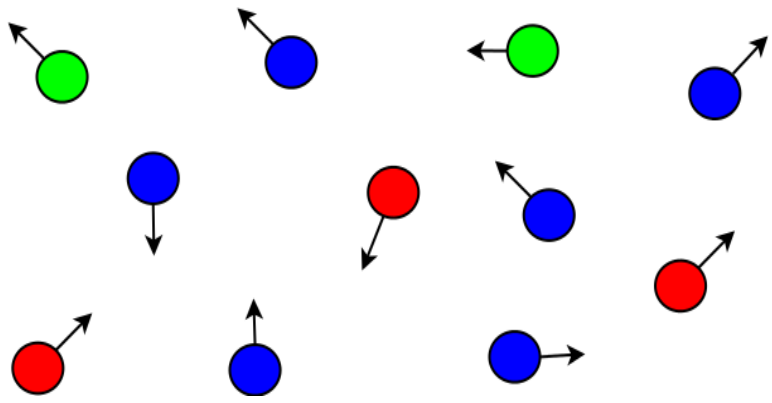


Figure 6.1. A two-dimensional oscillator network. The red (light) nodes are dedicated oscillators for input, the blue (dark) nodes for output

Note that the definition of the system allows static and dynamic, continuous input in time as well. In case of static input, the nodes of the oscillator network will synchronize after a

certain time. In case of continuous variable input in time the synchronization might not occur, or it might be disrupted, the presence of relatively constant synchronization, the groups of synchronized partitions also encode information.

During my work, I used the Spin Torque Oscillator (STO) model. STOs are placed on a plain surface, while the interaction is defined only by the distance between the oscillators. The positions of the STOs have to fulfill the rules of geometry. The excitation of the oscillators is applied by current input, and nodes have random initial states. Conversion between CMOS and STO level is limited, and only the oscillators at the edge of the network can be accessed. [A2]

In the case of static input, a basic step time (the time needed to achieve synchronization in **ON)** takes more time than an elementary operation on CMOS. However the dynamics of the STO network encapsulates rich computational capacity, and for a class of complex problems, only one synchronization step is enough on STO network with very low power consumption, while using CMOS long series of steps and much higher power is required. One of the most important questions will be to define the class of problems, where the STO networks are more efficient than software using CMOS architectures.

To write a program on STO network, to find the correct topology of oscillators requires an entirely novel approach. To solve the problem, it is advisable to apply some restrictions on the topology, which adapt to the properties of the ONs, and which were successfully used with other technologies also. Since the **ON**s are cellular, the interaction between closer nodes is stronger than between distant nodes; a CNN-type topology seems to be appropriate.

In the following two sections I will present two cellular (mainly locally connected) topologies. In the first example the grid-type layout, the OCNN is discussed, then a pyramid-like topology is presented, based on the OCNN, but with some relaxation of the geometrical constraints, still retaining the locality, i.e., the cellular structure of the system.

DOI:10.15774/PPKE.ITK.2019.005

6.2 Improving classification results with OCNN arrays

I proposed a grid-like topology denoted as OCNN (array). The rows of the grid are denoted as layers. An $n \times m$ grid OCNN can be characterized by the *n-1* distances between the layers and $m - 1$ distances between the columns. Thus; an OCNN program is determined by $n - 1 + m - 1$ positive values. The nodes of the first (top) layers are denoted as input and the last (bottom) layer as output. In the case of $1 \times m$ chain network, the input and the output nodes are the same.

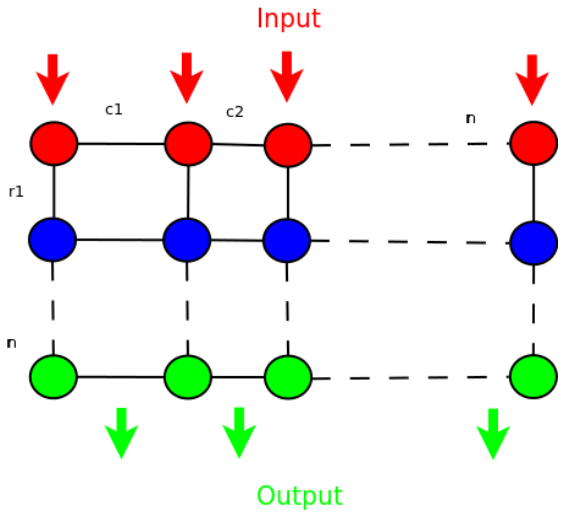


Figure 6.2. Structure of an Oscillatory CNN. The input is present as the current flow in the top layer; the output is the phase differences of the last layer

The hypothesis I raised was that the link between the input and the output could be formalized as a transformation between the two spaces, where the transformation is determined by the network topology. To demonstrate this capability of the OCNN array I proposed a modified classification architecture that also employs an OCNN array. Compared to the classic method where the feature vector of the input is classified to get the final result, in the architecture I propose, the feature vector is transformed by the OCNN array, and the resulted signature is classified (see Figure 6.3).

DOI:10.15774/PPKE.ITK.2019.005

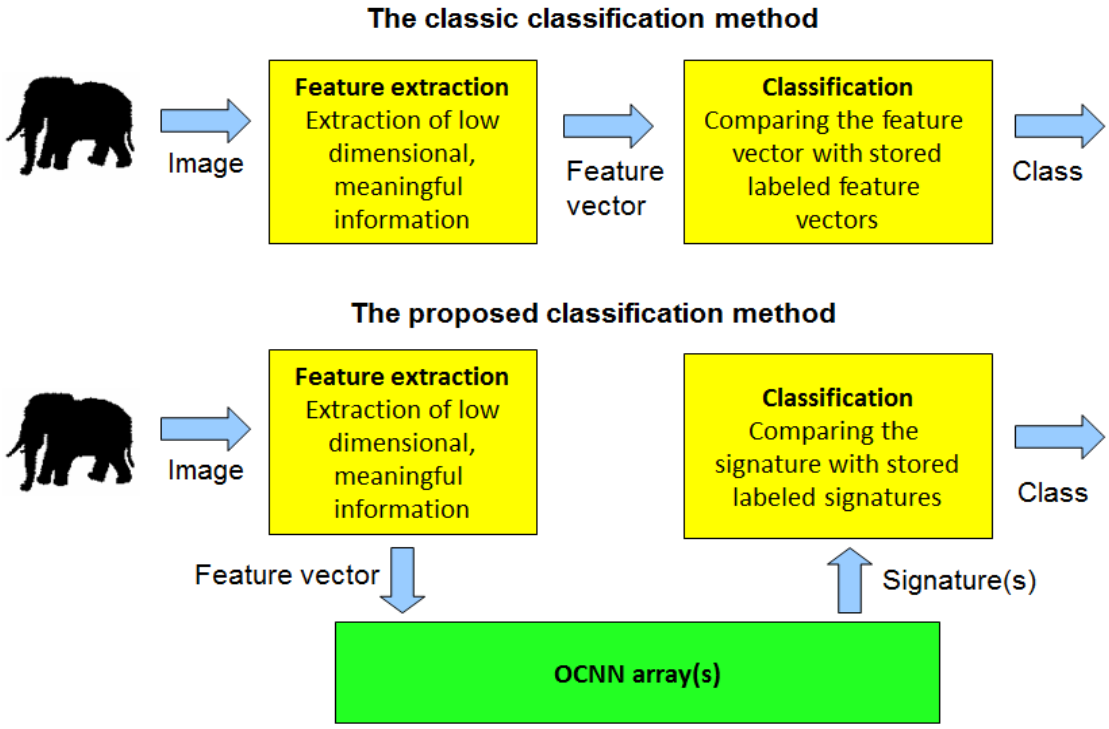


Figure 6.3 The classic and the proposed classification method using OCNN arrays for preprocessing the classification data

In this example, the program is to improve the classification performance. Increased accuracy can be achieved by a transformation, where the test set elements get more separable, the regions of the same classes get closer to each other, and regions of different classes get further. The method is similar to the kernel functions employed in the SVM classifiers, where the input is translated to another space, where in contrast to the original space, the classes are separable, as demonstrated in Figure 6.4.

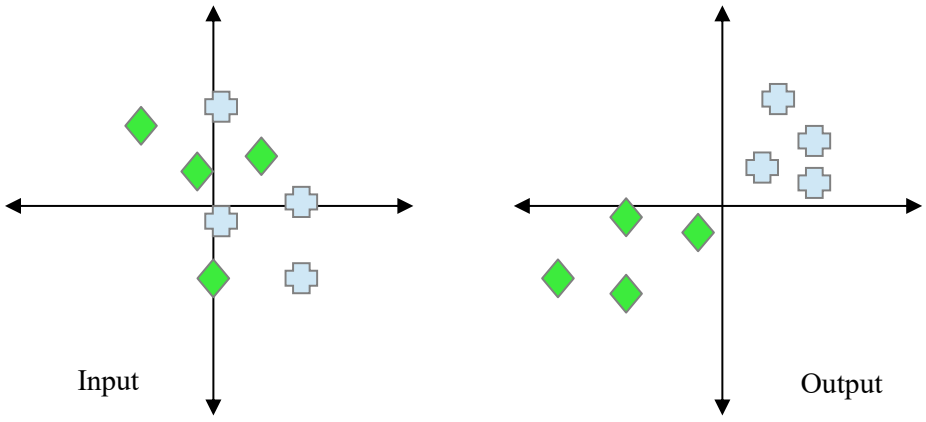


Figure 6.4 The effect of a 1D oscillator chain. Different points illustrate different classes in the input and the output space

DOI:10.15774/PPKE.ITK.2019.005

As a test data, I used reduced EPPSED shape feature vectors generated from the images of public shape databases. Examples are shown in Figure 6.5. The reduced feature vector consisted of the first and every fourth element of the original feature vector. This modification significantly shortened the simulation time and resulted in lower classification performance, thus provided a broader area to demonstrate the improvements of the OCNN array in the increase of the accuracy.

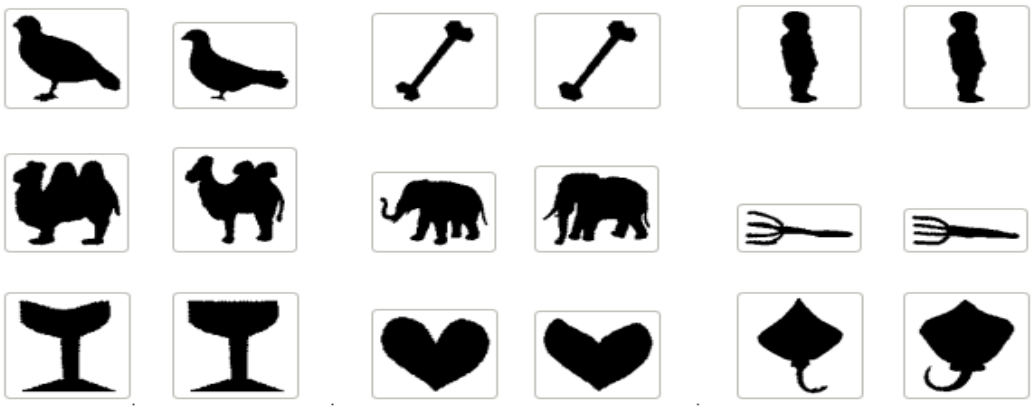


Figure 6.5. Examples of input shape images of 9 classes, images are from the Shape Database of the Vision Groups at LEMS, Brown University

In the experiment, I employed a one-dimensional OCNN array (chain). To find the proper topology that increases the separability, and hence the classification rate I used a genetic algorithm, where the chromosomes corresponded to the distances between neighboring nodes and the fitness function returned with the classification accuracy achieved by the topology determined by the distances:

$$f(T_n) = \sum_{i=1}^{k} \frac{1}{k} a_i \text{ , where}$$

$$a_i = \begin{cases} 1 & \text{if } i^{th} \text{ test element was classified correctly} \\ 0 & \text{if not} \end{cases}$$

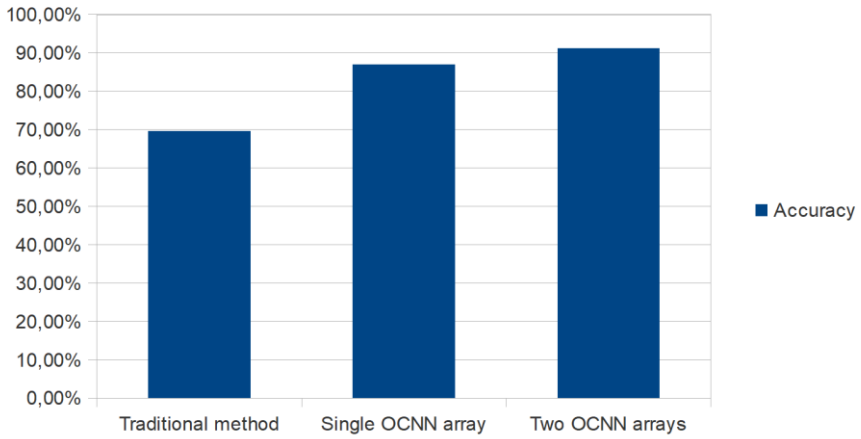


Figure 6.6. Learning improvement with a one-dimensional OCNN array.

On the shape set using reduced EPPSED features, the accuracy from 69.6% was increased to 87%, using two chains the result increased to 91.3% (Figure 6.6).

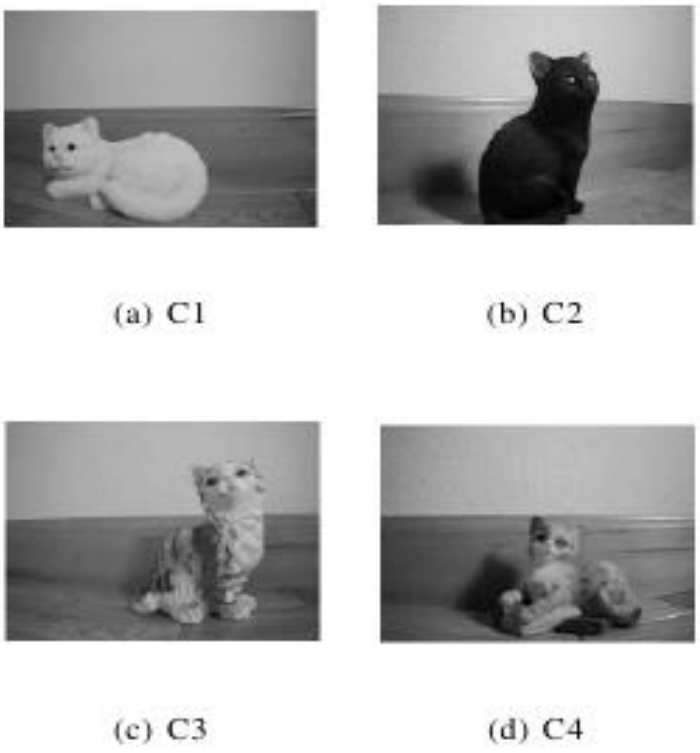


Figure 6.7. Example images of the four different image classes of figurines, the sources for the H-MAX data

To verify the results above, I performed a learning test on a different data set with HMAX features and measured the relationship of the classes in the signature and feature

space. The HMAX feature aims to describe the input image based on biological methods, but result in a significantly larger feature vector counting 8150 elements. The HMAX algorithm contains four different layers (S1: Gabor Filter Layer C1: Local invariance Layers S2: Intermediate Feature layers C2: Global invariance layers) and generates a different number of features in each layer. This way it is capable of generating scale-invariant image representations based on the predefined features. [123] Example input images are shown in Figure 6.7.

Since the size of the network increase with the dimension of the input, the number of the free parameters of the network is also proportional to the input dimension. Searching in such a large parameter space is complex; thus I reduced the input by averaging and vector quantization.

To measure the topology of the feature and the signature space I defined a metric that indicates the compactness and the separation of the classes.

The in-class distance or in-group distance of the class is the average of distances of all instances belonging to the class:

$$IGD(C_i) = \frac{\sum_{l,k \in C_i, l \neq k} |l,k|}{\frac{n(n-1)}{2}}$$

where C_i denotes the class, l and k are the members of the class, n is the number of elements int he class

The cross-class distance is defined as the average of distances of every instance of the class and every other class elements:

$$CGD(C_i) = \frac{\sum_{l \in C_i, k \in C_1 \cup C_2 \cup \ldots \cup C_{i-1} \cup C_{i+1} \cup \ldots \cup C_m} |l,k|}{n \cdot o}$$

where C_i denotes the class, l is the member of the class C_i, k is the member of all other classes, n is the size of C_i, and o the number of all other elements.

Since the topology of the classes may vary a lot in extension and distribution also, I measured the compactness as the ratio of cross-class and in-class distances.

$$AD(C_i) = \frac{CGD(C_i)}{IGD(C_i)}$$

DOI:10.15774/PPKE.ITK.2019.005

TABLE 6.1 SEPARABILITY METRIC (VECTOR AVERAGING)

Cross Group/ In Group	*Class I*	*Class II*	*Class III*	*Class IV*
Without the OCNN array	3.60	2.87	1.58	1.83
50 long vectors	15.02	3.58	3.26	9.16
163 long vectors	12.28	3.46	3.23	7.92
815 long vectors	20.10	5.30	2.47	8.56

TABLE 6.2 SEPARABILITY METRIC (VECTOR QUANTIZATION)

Cross Group/ In Group	*Class I*	*Class II*	*Class III*	*Class IV*
50 long vectors	29.70	8.82	9.24	9.57
163 long vectors	96.55	10.80	30.74	52.54
815 long vectors	175.56	23.81	56.04	59.28

Results of the learning are shown in Table 6.1 (average) and 6.2 (quantization). The compactness could be significantly improved in all four input classes. Based on the performance increase experienced on the reduced EPPSED data and the increased compactness of the H-MAX data it can be stated that OCNN arrays are programmable to perform a transformation in space, based on a limited number of training elements. [A8]

6.3 Classification using pyramid-like cellular oscillatory network

To write in the input and read out the output a specific conversion between the oscillators and the CMOS level a specific converter circuit is assumed. Hence, the learning-classification algorithms are running on the CMOS level. This method is improved here, implementing the preprocessing and classification part also on the oscillatory level.

In the method mentioned above to increase the efficiency of machine learning by using OCNN arrays, the classification is done after conversion taking place on CMOS. This means that after synchronization the phase differences are read out from the last layer of the OCNN to perform the nearest neighborhood classification on CMOS. To avoid reading out

DOI:10.15774/PPKE.ITK.2019.005

a vector of phases, an oscillatory classification network is proposed, where only the final result is read and converted. Thereby, implementing the classification on oscillatory level increases the speed and decrease the size and consumption.

Since the properties of the oscillators are the same, the behavior of a network depends only on the initial phase differences and the topology. Without the possibility of wiring, no complex algorithms (steps, branching, etc.) can be done; only those computations can be performed that follow from the physics of the coupled oscillators used. Therefore a binary classifier is an obvious solution, where the sign of the phase difference of a fixed pair of oscillators is considered as the output of the classification. The output-pair can be the part of an OCNN array with more layers, i.e., a 1-D OCNN chain, retaining the locality of the interactions.

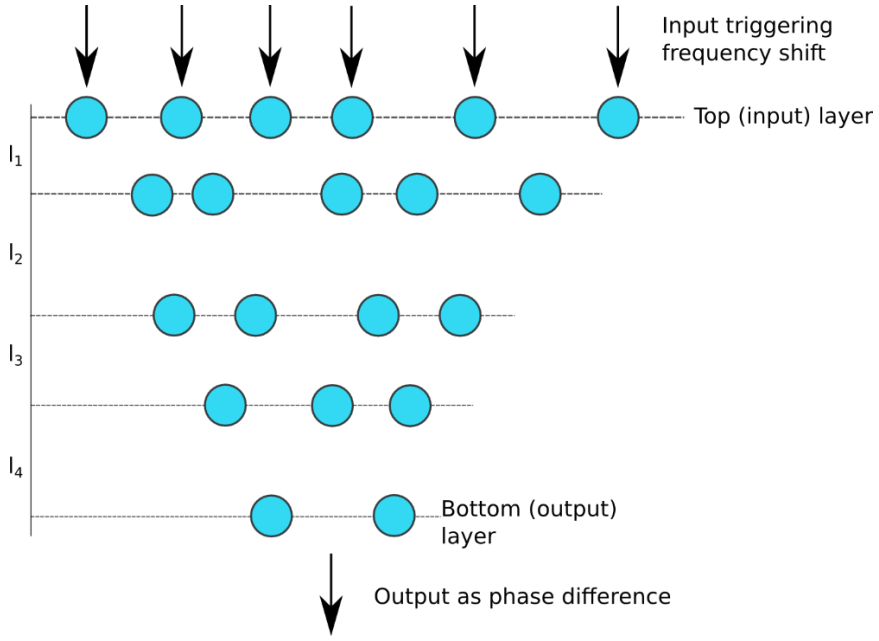


Figure 6.8 The structure of the pyramid-like oscillatory network

To achieve higher complexity, a pyramid-like architecture is proposed. The regular grid-like topology is modified in a way that the number of the oscillators in the layers are decreasing from top to bottom, where only two oscillators can be found. (The reason to have two oscillators at the bottom, and not one, is that the output is not the phase but the phase difference of two oscillators.) In each layer, the distances between the oscillators are also varying. Note that all other constraints of the regular OCNN grid are still satisfied: the oscillators are homogeneous, and the interaction between node cells only depends on the distance between them. (Figure 6.8.)

To determine the optimal distances between the oscillators, a genetic algorithm is used, where the fitness value is equal to the accuracy of the classification defined as above. Every chromosome of given length represents a real and unique network topology.

- The first $[g_1, \dots, g_{k-1}]$ genes determine the horizontal position of cells in the first (input)
- The next $[g_k, \dots, g_{2k-3}]$genes determine the distances of the layers (vertical position)
- The rest $[g_{2k-2}, \dots, g_n]$ genes determine the horizontal positions of the non-top-layer cell nodes

6.3.1 Static input classification

To demonstrate the functionality of the pyramid-like classifier I constructed two basic examples.

The first example solves a summation and threshold classification by learning (applicable also forAND and OR logical functions by biasing). The input is a vector of real numbers on the interval of $[0, \dots, 100]$, the desired output is $true$ if the sum of the input is higher than a fixed threshold ($100n$ for logical OR operation, $100n(n-1)$ for logical AND operation), $false$ otherwise. This class of problem is easily solvable by, e.g., neural networks, single perceptrons, but with decision trees, no perfect solution can be given. I tested the learning on a four-long vector, where I could reach 100% of accuracy on few steps of the genetic algorithm. Results are shown in Figure 6.9 and 6.10.

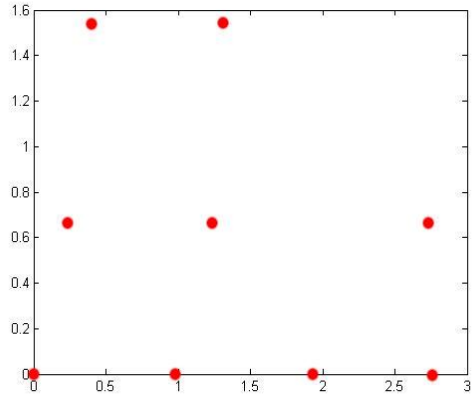


Figure 6.9 The topology of the summation and threshold example. The input oscillators are at the bottom of the figure, output nodes are at the top.

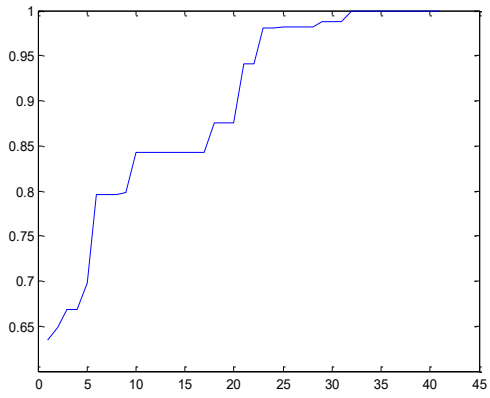


Figure 6.10 The fitness value (the classification accuracy) of the best instance in the function of the epochs during the evolutionary learning of the OR-AND example

DOI:10.15774/PPKE.ITK.2019.005

The second example demonstrates a XOR-type classification by learning. The expected input is a vector of real numbers representing logical values, $false$ as $[0, 10]$ and $true$ as $[95, 105]$. The desired output is true if the number of true inputs is odd, false otherwise. This problem-class can be solved in two steps by a decision tree, but it cannot be solved by a single perceptron because the input set is not linearly separable.[2] In a hundred steps I could achieve 95% accuracy on random test sets. Results are shown in Figure 6.11 and 6.12.

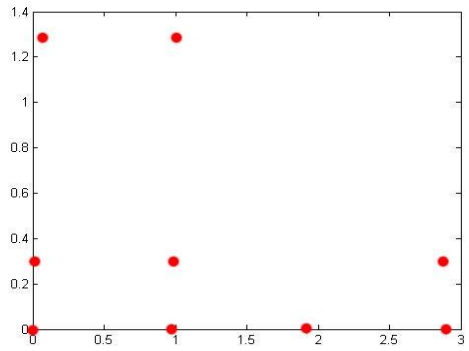


Figure 6.11 The final topology of the XOR example.

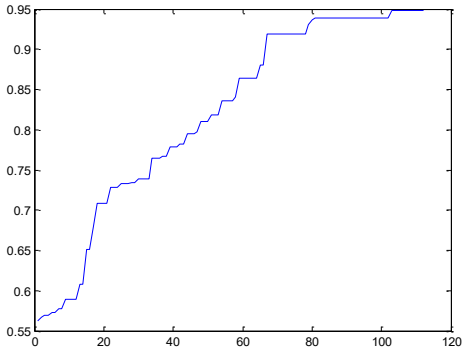


Figure 6.12. The fitness value (the classification accuracy) of the best instance in the function of the epochs during the evolutionary learning of the XOR-example

The presented examples show that the two elementary learning problems are solvable on the proposed architecture.

6.3.2 Classification of spatial-temporal inputs

Besides the advantages already mentioned, the oscillator network is able to handle not only constant but also continuous input in time. Since the oscillation represents memory in time, the possibility is given to classify spatial-temporal inputs by the proposed architecture at the oscillator-level.

To test this ability of the oscillator networks I investigated a classification problem of primitive spatiotemporal signals.

In my first experiment, I learned the system to detect the direction of a moving signal. All the examples were the members of the “left” or the “right” class, but noise was added to the signal in different levels. In the test environment, I used five input oscillators with 100 time steps (Figure 6.13 c and 6.13 d).

DOI:10.15774/PPKE.ITK.2019.005

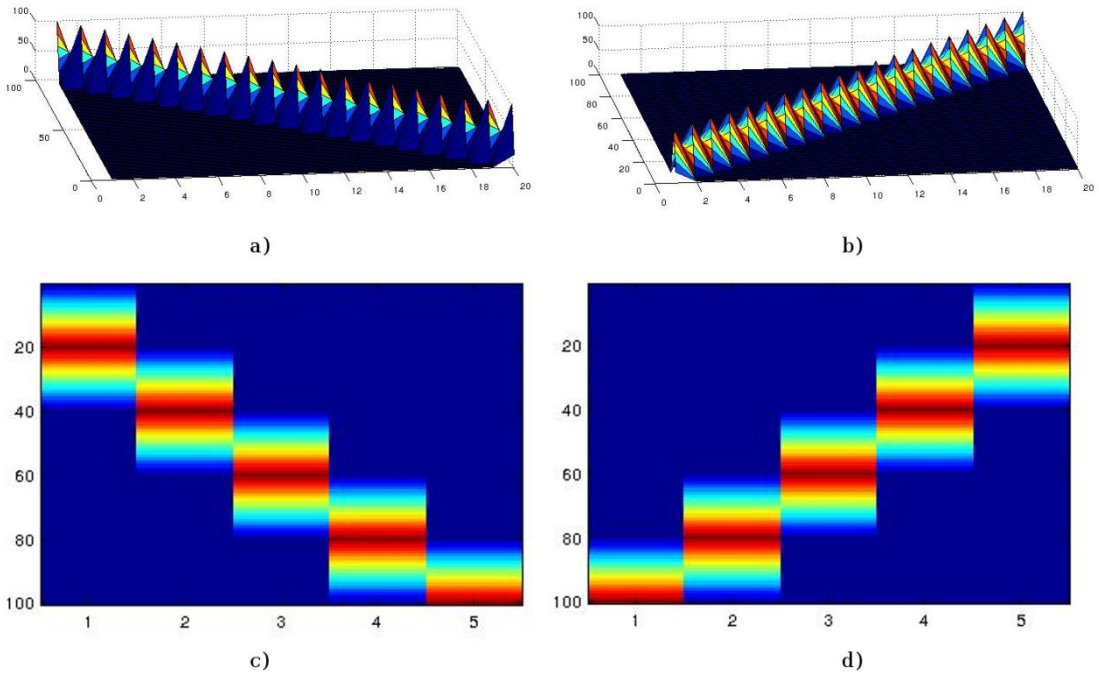


Figure 6.13 Input examples of moving signal in time with 20 (a and b) and 5 (c and d) input oscillators. Blue (dark) color represents low excitation, red (light) corresponds to a high value

In the first experiment, I used the nearest neighborhood classification on the signature output from a 1-D array, where the oscillator array functioned as a feature generator also. I investigated the response for noisy events with different noise levels. The train elements for the nearest neighborhood classification were the signatures from the oscillator-system with the noise-free input. The distances of the consecutive oscillators in the chain were the same. The noise ratio is the ratio of the noise amplitude and the total input amplitude:

$$n_i = \frac{A_{noise}}{A_{signal} + A_{noise}}$$

DOI:10.15774/PPKE.ITK.2019.005

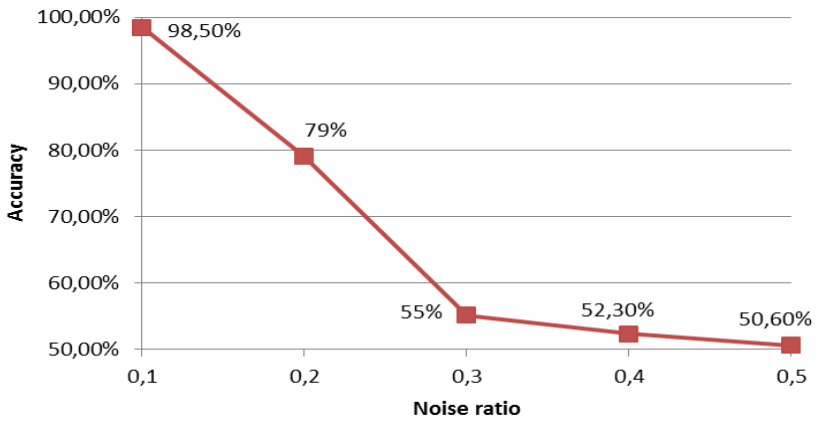


Figure 6.14 The accuracy of the NN classification on noisy input

Figure 6.14 shows the results for the NN classification on the 1-D oscillator chain signatures. The accuracy is decreasing continuously, on the level where the noise amplitude and the signal amplitude is equal, the accuracy reaches the 50%, that is level of uncertainty. In Figure 6.15 few examples of inputs and output signatures are shown for different noise levels.

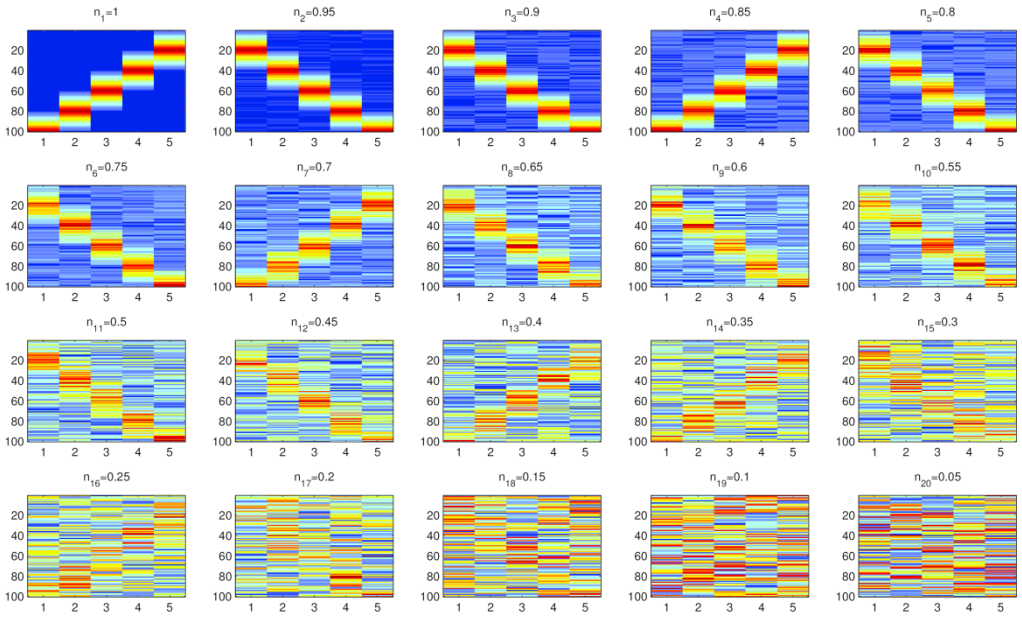


Figure 6.15 Example inputs for detection of the moving signal direction

DOI:10.15774/PPKE.ITK.2019.005

The motion direction classification was also tested on the pyramid-classifier using the generic learning technique showed above. The training set contained patterns with different noise levels. To avoid overfitting and ensure that the classifier will learn the basic pattern without noise the fitness function *f* of the trainer was modified to tolerate mistakes on higher noise level:

$$f(T_n) = \sum_{i=1}^{k} \frac{1}{k} a_i ,$$

where

$$a_i = \begin{cases} w_i & \text{if } i^{th} \text{ test element was classified correctly} \\ 0 & \text{if not} \end{cases}$$

and

$$w_i = 1 - n_i = \frac{A_{signal}}{A_{signal} + A_{noise}}$$

The genetic algorithm to build the pyramidal topology was performed with 30 instance limit for 250 epochs in five families. The resulting oscillator network showed high classification accuracy on low noise and acceptable performance even with higher noise levels. Accuracy in function of the noise is shown in Figure 6.16, whereas Figure 6.17 shows the corresponding topology of the oscillator network. The evolution of the learning is shown in Figure 6.18.

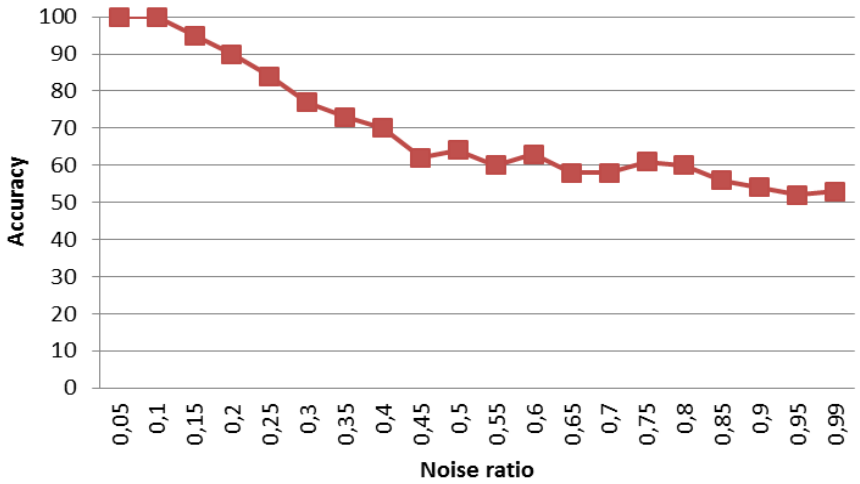


Figure 6.16 The classification accuracy in function of the noise level. The vertical axis shows intervals of the noise level

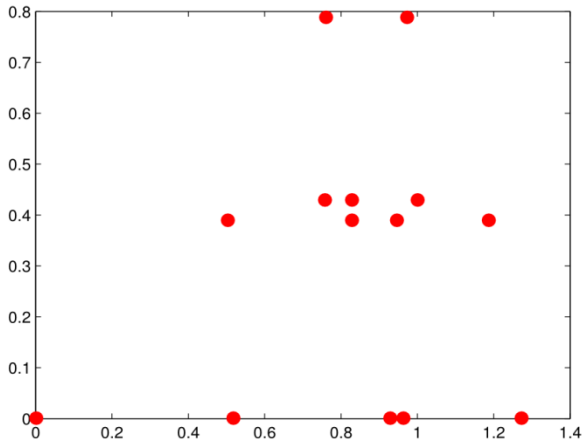


Figure 6.17 Topology of the pyramidal ON with the highest fitness value resulting in the accuracy in Figure 6.18. The input row is at the bottom of the network; the output is computed as the difference between the phases of the top oscillators

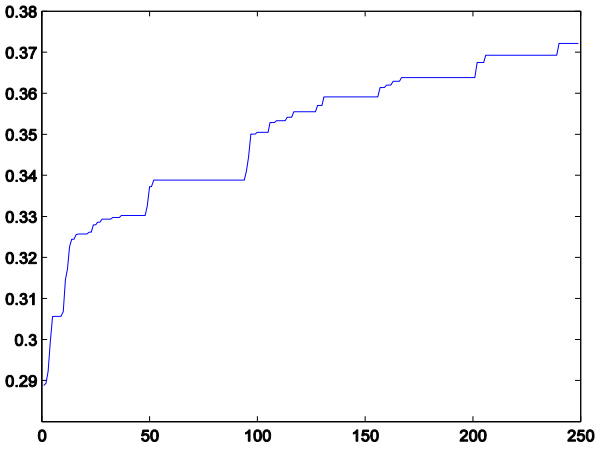


Figure 6.18 The best accuracy of the learning epochs. The longer plateaus show stuck in the local minimas. The maximal value is 0.5245 due to the noise-level corrected fitness function, while the minimal value of 0.26225 corresponds to the 50% accuracy (the zero level of a binary classification)

To validate the results, I performed a reversed test also since a reflected input should result in an inverted output. The classification performance remained the same as well as in the function of the noise. I also measured the ratio of the intersection of the correctly classified input instances and got the result $a_c = 79\%$. A ratio near to $a_c^{max} = 100\%$ would correspond to a total overfitting, since the learning was independent of the directional information. On the other hand, since the global accuracy on the test set regardless the noise level was 69.4%, the minimal possible value of the intersection ratio, $a_c^{min} = 38.8\%$ would indicate total overfitting as well.

DOI:10.15774/PPKE.ITK.2019.005

The goal of the experiments was to verify the basic applicability of the STO networks in identifying spatiotemporal signals. I succeed to prove the hypotheses in all cases; however further improvements are needed for any application in the future. [A2]

DOI:10.15774/PPKE.ITK.2019.005

7 Summary

7.1 New Scientific Results

Thesis I.

I designed a new general shape descriptor called the Global Statistical and Projected Principal Edge Distribution descriptor that is based on different modalities. It combines the advantages of the global statistical and local edge-based descriptors. I implemented the descriptor and verified its efficiency through experiments.

Related publications: [A1][A3][A4][A7]

The descriptor consists of global mathematical shape features, and an edge based local feature set. The method of assembling more shape descriptors into one aims to describe different modalities independently from each other, and to speed up the recognition process.

The first part of the descriptor contains statistical shape features that describe the shape as whole. Employing these features resulted in a higher cover ratio, and accelerated decision accuracy in case of a comparison-based classification.

The first property is the eccentricity of the shape, which characterizes the elongation of the shape with one scalar value. The second property is the ratio of the shape area and the area of its bounding box. The following eight features are the first four statistical moments of the vertical and horizontal histograms.

The edge-based shape description part characterizes the local properties of the shape. The algorithm was inspired by the PPED image descriptor, which selects and projects principal edge values.

The edges are detected in four directions, resulting in four edge maps. For every pixel location in the four edge maps, the principal value is highlighted, and the other values are weakened or totally neglected. Highlighting is performed by a soft-thresholding function, which decreases the effect of overfitting and provides the ability to use the algorithm on different architectures with different number representations. I measured the efficiency of the soft-thresholding, and it improves the performance of the shape description.

The rotation invariance can be ensured at two points of the recognition process. The first approach incorporates the invariance in the description directly or by a preprocessing including an angular normalization. The second point is at the classification phase, on the employing an invariant metric or by storing (all) the different rotations of one instance in the training set.

Descriptors of the family of PPED are not rotation invariant. Therefore the rotation invariance is ensured by rotationally redundant training set. Although this is closer to the human sense, it would result in a vast database. From this consideration, I normalize the shape angularly, based on the orientation of the shape. The blob is rotated to have its major axis parallel to the horizontal axis. Finally, the shape is rotated by 180 degrees if its center of gravity is on its right side.

The scale-invariance is ensured by normalization, by resizing the blob to fit into a 64×64 pixel window. The position-invariance in the vertical dimension is ensured by moving the shape in the middle, thus in the other dimension, the shape touches both sides of the frame edge.

The shape descriptor algorithm I designed is optimal for dedicated VLSI architecture and for Cellular Wave Computer.

I compared the GSPPED shape descriptor with the most widely used shape descriptors on a shape set containing patches from the Hungarian Forint Banknotes. Results show that the performance of the GSPPED is higher than the performance of the Complex Zernike Moments and the Generic Fourier Descriptor.

Thesis II.

I designed and implemented a reliable and robust, two-level, parametric, memory-based classification method. The classifier is able to handle multiple classes, it has the ability to reject distant inputs, and can handle compound features.

Related publications: [A1][A3][A4]

The two-level decision adapts to the semantics of the descriptor and to the discriminative power of each descriptor parts. First, the mathematical and statistical shape-descriptors are being compared for fast filtering, then by employing the Adaptive Limited Nearest Neighborhood algorithm, the edge-based properties are being compared.

The essence of the method is a multi-level application of the compound descriptor that speeds up the decision and increases the cover of the recognition. When comparing with representative instances, first only simple mathematical properties are analyzed to examine the presence of a basic similarity. Although the final decision cannot be expected from this primary comparison, it is proven to be efficient to filter and reject elements that obviously differ from the instances of the representative set.

II.1 I demonstrated that in a two-level, comparison-based classification, filtering based on simple but highly expressive features can both speed up the

classification and can significantly increase the recall, if the goal of the second level is to maximize the precision.

In case of a comparison-based classification, the complexity of the decision increases with the size of the training set. Thus, if the difference exceeds a certain threshold level during the comparison, the input can be rejected without continuing the evaluation. The first properties of the GSPPED descriptor contain highly expressive features, that are suitable for rejection.

Applying the proposed filtering, the average decision time decreases to 12% compared to the decision time without filtering. Besides the significant acceleration, filtering has a secondary effect, namely that the cover of the recognition increases by 17% in average in case when filtering is applied. This seemingly paradox phenomenon can be explained as follows: features applied for filtering are orthogonal to those applied in the second phase of comparison in information theoretical sense. To maximize the precision in the second phase, the acceptance radius is set as a function of the distance of the closest element of other classes. As a result, acceptance regions increase since instances closer to the representative elements might be rejected. The explanation is verified through measurements, where acceptance regions are 25% larger with filtering. Since the second phase focuses on maximizing the precision, the precision remains independent of the filtering.

II.2 I designed and implemented the Adaptive Limited Nearest-Neighbor (AL-NN) classifier. The presented method is capable of rejecting non-relevant (zero class) instances, that is one of the main benefits against standard KNN methods.

Nearest-neighbor classifiers compare inputs to already known, labeled instances of representative sets. To overcome the one of the main drawbacks of the traditional nearest-neighbor classifiers - namely the lack of rejecting non-relevant elements - I extended the algorithm by introducing limits to each stored instance.

Limits are set individually to each element of the representative set based on the irrelevant and out-of-class instances included in the training set. For this very reason I introduced a combined training set including not only relevant instances but non-relevant ones as well. Although it is not discussed in the present thesis, the model is capable of making decisions in case of overlapping classes on the instance-level.

The model is built up as follows: The training set contains labeled instances, including non-relevant (also called as zero class) elements as well. To all relevant instances of the training set a limit is defined based on the surrounding instances. In case of having only non-relevant neighbors, the limit is set to the half of the distance between the closest

neighbor and the current trainable element. When only an instance from a different class is within the current filter region, the limit is set to the distance of the nearest outlier and the current example. If all elements within the filter region are of the same class, the limit is defined as the distance of the farthest instance of the class. However, in the most cases more instances of more classes fall close to the example, thus the limit will be the minima of the limits defined above. Finally, if the corresponding filter region of the current training instance is empty, we reject the instance due to the lack of information about its surrounding. The result of the training is the representative set, containing only relevant class instances. The limit can be fine-tuned with a vigilance parameter, that can adjust the recall-precision relation as well.

I compared the classifier with other widely used methods, like KNNs, AutoMLPs, FFNNs and SVMs. The AL-NN shows higher performance compared to the other methods.

II.3 I designed a representative set optimizer and an adaptive online update method for the AL-NN classifier. By using the optimizer, the size of the representative set can be decreased without a significant decrease of the cover. The online update of the representative set provides the ability to include new instances from test sets automatically.

The usability of the decision methods is significantly better if the model can be automatically or semi-automatically updated and optimized during tests or measurements. A special scenario of such use cases is when the initial training set contains only a few instances at the beginning of the first test, and the model continuously increases by performing tests.

During Adaptive Limited Nearest-Neighborhood classification not only the output class of each sample can be determined but also the confidence of the output, based on the ratio of the measured distance and the acceptance radius. In case a new instance that is classified is located at the border of a class, it can be added to the representative set to extend the cover of the current class, or to increase the precision of the borderline between classes. To ensure that the representative set is extended with proper instances, I used the temporal order to verify the classification result.

By estimating the size of the intersections of acceptance regions of the instances in the representative set, redundant elements can be dropped. I suggested an iterative optimizer, where representative instances are ordered based on the number of other instances they cover, and in each step the instance with the larger cover is kept, and other instances covered by it are removed. By employing the optimizer, the representative set can be reduced to the 30-50% of the original size, while the cover decreases only by 5-15%.

DOI:10.15774/PPKE.ITK.2019.005

Thesis III.

I formalized the network of weakly coupled oscillators as a computational unit, and I defined the concepts of a program: the data, the input and the output on the system. I provided an experimental proof that the oscillator networks are capable of resulting in a desired signature on response to a given input excitation, thus the unit is suitable to be used as a computational unit.

Related publications: [A2][A3][A8]

Two or more oscillators are interacting and form a coupled oscillator network. The Spin Torque Oscillators and the networks built from STOs are the subjects of recent researches of non-Boolean computational units on several levels.

The main emphasis today is on investigating the feasibility of using the STOs and other non-Boolean units by simulations. In my research an STO network model is used, where the interaction is determined only by the topology of the network. The atomic operation is considered as a synchronization, where the individual oscillators might be in a different phase. I formalized the program as the topology and the oscillator dynamics. The data is implemented as a phase-difference array, the input is the excitation, and the output is represented as the data of the output nodes.

I designed and implemented a method that creates an OCNN topology with a genetic algorithm, that, as an OCNN, transforms an input to a separable output space. The applicability of the method is proven through experiments.

In classical, two level object detection models the multidimensional input is first reduced in dimensionalities through special descriptors, and the final decisions are made through these compressed representations. Trainability and correctness of the decision highly depends on both compactness and discriminative power of the descriptors according to the output classes.

I proved the OCNN's capability of performing the desired transformation with a learning example. I proposed a new classification method by inserting an OCNN array between the feature extractor and the classifier, thus the desired function of the array was to transform the feature to improve the final classification results. To design the proper topology of the OCNN array, I used a genetic algorithm that maximized the classification performance. I tested the proposed system with the classification of EPPSED data and verified the results with H-MAX vectors.

During the experiment I succeeded in improving classification performance on all the test sets. I also proved by measurement that the compactness of the classes in the feature space increases significantly.

I designed and implemented a method that, based on input-output pairs, defines the topology of a pyramidal oscillator network built up of STOs, performing the desired classification. The applicability of the method for classification of spatio-temporal signals is proven through experiments.

I proved the feasibility of using a STO network as a computational unit by building a classifier. Classification encompasses not only a transformation but also separation, thus the task is more complex. I tested the classifier on static and spatiotemporal data as well.

I proposed a multi-layer topology for classification. The number of the oscillators in the top layer equals the dimension of the output. In the next layers the number of the oscillators is decreasing until the last layer with only two oscillators.

The topology had been tested for static and spatio-temporal inputs. A static sample performing a summation and a XOR was examined with a success. A moving signal function was used as a dynamic input with changing directions and noise. In each of the experiments the capability of teaching topology was proved.

7.2 Methods

The designed algorithms, tests and experiments were mostly implemented in Matlab[9] environment. On lower levels of visual processing (matrix operations, convolution, morphology - propgen, torque computing) I employed the built-in functions of Matlab, but on higher levels I created my own implementations for better understanding and to allow higher flexibility to the methods.

The WEKA[10] and RapidMiner[11] machine learning implementation collections helped me a lot in exploring the basic applicability of methods. Based on these results I made my own implementations of the nearest neighborhood and neural network model, and also implemented a framework for genetic algorithms.

I tested the banknote recognition algorithm of the Bionic Eyeglass with people with visual impairments. At the beginning of the tests, the participants were given detailed information about the operation of the device. After that, they had to identify banknotes without any external help, relying on the device only. During the tests, we stored every

[9] http://www.mathworks.com/products/matlab/
[10] http://www.cs.waikato.ac.nz/ml/weka/
[11] https://rapidminer.com/

DOI:10.15774/PPKE.ITK.2019.005

processed frame with the corresponding decisions, so the experiments can be completely reproduced later in a simulation environment. With these steps, the team was able to track the development of algorithms. During the live tests we have written records and notes to collect all remarks according to the behavior of the algorithms, the usability of the device or to other environmental circumstances. Test sets presented in this thesis are compound of these live tests.

7.3 Application of the results in practice

The first and second thesis was developed in the framework of the Bionic Eyeglass project, actively utilizing experiences collected from the live tests. By employing the shape description and classification I proposed, a complex classifier has been created that is suitable for the recognition of Hungarian Forint banknotes. The application identifies the banknote based on more classifiers, resulting in a robust, reliable device.

Beyond banknotes, the system might be used for other visual tasks as well, which require the classification of rigid, flat shapes. Within the Bionic Eyeglass project algorithms might be used for recognition of other banknotes and pictograms, reading of LED displays and identification of information boards.

The developed descriptor and classifier are already implemented on FPGA as well, providing faster processing compared to standard CPUs. (The implementation on FPGA was not the part of my research.)

In the future, oscillator networks may exchange the part of present-day classic CMOS components. Application areas are examined in many research institutes, this thesis is also part of this process.

DOI:10.15774/PPKE.ITK.2019.005

Acknowledgment

First of all, I would like to thank my supervisor Tamás Roska that he involved me into the scientific community, and launched my scientific career, helped me with his knowledge, enthusiasm, and respect. I would like to thank him for allowing me to visit research institutes in Tokyo and in Portland.

I am especially grateful to Kristóf Karacs for being my patient mentor during my doctoral studies. I am grateful for his ideas and incredibly detailed, constructive critiques.

I am thankful to the leaders of the Pázmány Péter Catholic University, Faculty of Information Technology and Bionics, especially to Péter Szolgay, Judit Nékyné Gaizler, Ágnes Bérczesné Novák for creating and managing such a professional and value-oriented research center. I acknowledge the kind help of my supervisor in Tokyo, Tadashi Shibata, that he helped me to broaden my horizons and to make progress in my researches. The project was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002). I am the most grateful to my family for teaching me to respect our values and for the inspiration to improve my knowledge continuously.

I am thankful to my bride Timi for her patience and love.

I am grateful to my colleagues and friends, András Horváth, Mihály Radványi, Miklós Koller and Tamás Zsedrovits and Tamás Fülöp for the discussions and support.

References

Author's publications

[A1] A. Stubendek, K. Karacs, "Shape Recognition Based on Projected Edges and Global Statistical Features", Mathematical Problems in Engineering, vol. 2018, Article ID 4763050, 18 p, 2018.

[A2] A. Horváth, M. Koller, A. Stubendek, T. Roska, "Spatial-temporal Event Detection via Frameless Cellular Wave Computing - a Review", Nonlinear Theory and Its Applications, IEICE, 2014, vol 5 (3), pp. 391-408, 2014.

[A3] A. Stubendek, K. Karacs, T. Roska, „Shape Description Based on Projected Edges and Global Statistical Features", International Symposium on Nonlinear Theory and its Applications (NOLTA 2014), Luzern, Switzerland, 2014.

[A4] K. Karacs, M. Radványi, A. Stubendek and B. Bezányi, "Learning Hierarchical Spatial Semantics for Visual Orientation Devices," in Proc. of the IEEE/CAS-EMB 2014 Biomedical Circuits and Systems Conference (BIOCAS 2014), Lausanne, Switzerland, pp. 141–144, 2014.

[A5] A. Horváth, A. Stubendek, T. Roska, "One dimensional cellular array of in-plane spin torque oscillators," 13th IEEE International Conference on Nanotechnology, Beijing, China, 2013.

[A6] Z. Solymar, A. Stubendek, M. Radvanyi, K. Karacs, "Banknote Recognition for Visually Impaired" in Proc. of the European Conference on Circuit Theory and Design (ECCTD'11), Linköping, Sweden, Aug 2011.

[A7] M. Radvanyi, Z. Solymar, A. Stubendek, K. Karacs, „Mobile Banknote Recognition – Topological Models in Scene Understanding", in Proc. of the 4th International Symposium on Applied Sciences and Communication Technologies (ISABEL 2011), Barcelona, Spain, 2011.

[A8] T. Roska, A. Horvath, A. Stubendek, F. Corinto, Gy. Csaba, W. Porod, T. Shibata, G. Bourianoff, "An Associative Memory with Oscillatory CNN Arrays using Spin-Torque Oscillator Cells and Spin-Wave Interactions Architecture and End-to-end Simulator," 13th IEEE international workshop on cellular nanoscale networks and applications, Turin, Italy, 2012.

[1] P. Simon, “Too Big to Ignore: The Business Case for Big Data”. Wiley. p. 89. ISBN 978-1-118-63817-0, 2013.

[2] D. Poole, A. Mackworth, “Artificial Intelligence: Foundations of Computational Agents”, Cambridge University Press, ISBN 978-0-521-51900-7, 2010.

[3] K. P. Murphy, “Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)”, The MIT Press, ISBN-13: 978-0262018029, 2006.

[4] T. Hastie, R. Tibshirani, J. Friedman, “The Elements of Statistical Learning – Data Mining, Inference, and Prediction”, Springer Series in Statistics, ISBN: 978-0-387-84857-0, 2011.

[5] Ch. M. Bishop, “Pattern Recognition and Machine Learning”, Springer Science + Business Media, LLC, ISBN: 978-0387-31073-2, 2009.

[6] S. Russel, P. Norwig, “Artificial Intelligence: A modern approach”, Pearson Education, ISBN 963545411-2, 2010.

[7] W. Ertel, “Introduction to Artificial Intelligence”, Springer Science + Business Media, LLC, ISBN: 978-085729-298-8, 2011.

[8] M. Kantrardzic, “Data Mining: Concepts, Models, Methods, and Algorithms”, Institute of Electrical and Electronic Engineers, 2nd ed., ISBN: 978-0-470-89045-5, 2011.

[9] I. Witten, E. Frank, “Data Mining: Practical Machine Learning Tools and Techniques”, 2nd ed., Morgan Kaufmann series in data management systems, ISBN: 0-12-088407-0, 2005.

[10] P. Flach, “Machine Learning: The Art and Science of Algorithms that Make Sense of Data”, Cambridge University Press, ISBN: 978-1-107-09639-4, 2012.

[11] S. Haykin, “Neural Networks and learning machines”, 3rd edition, Pearson Education Inc, ISBN: 978-0-13-129376-2, 2009

[12] M. L. Minsky, S. A. Papert, “Perceptrons: An Introduction to Computational Geometry”, MIT Press, Cambridge, ISBN: 0-262-63022-2, 1969.

[13] S. Marshland, “Machine Learning: An Algorithmic Perspective”, Chapman & Hall / CRC machine learning & pattern recognition series, ISBN: 978-1-4200-6718-7, 2009.

[14] J. W. Rudy, “The neurobiology of Learning and Memory”, Sinauer Associates, ISBN: 978-0-87893-669-4, 2008

[15] E. Cherubini, R. Miles. “The CA3 region of the hippocampus: how is it? What is it for? How does it do it?”, Frontiers in cellular neuroscience, vol. 9 19. 5, 2015.

[16] R. O. Duda, P. E. Hart, D. G. Stork, “Pattern Classification”, A Wiley-Interscience Publication, John Wiley & Sons, 2nd edition, ISBN: 0-471-05669-3, 2001.

[17] S. Schölkopf, A. Smola, “Learning with Kernels: Support Vector Machines, Regularization, Optimization, and beyond”, MIT Press, ISBN: 978-0-262194754, 2002.

[18] E. Alpaydin, “Introduction to Machine Learning”, MIT Press, Cambridge, ISBN: 9780262012119, 2004.

[19] D. E. Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning”, Addison-Wesley, Reading, ISBN: 0-201-15767-5, 1989.

[20] W. Banzhaf, P. Nordin; R. Keller, F. Francone, “Genetic Programming – An Introduction”, San Francisco, CA: Morgan Kaufmann, ISBN 978-1558605107, 1998.

[21] M. Mitchell, “An Introduction to Genetic Algorithms” Cambridge, MA: MIT Press, ISBN 9780585030944., 1996.

[22] J. D. Lohn, Gregory S. Hornby, Derek S. Linden, “An Evolved Antenna for Deployment on Nasa’s Space Technology 5 Mission”, Chapter Genetic Programming Theory and Practice II, vol. 8 of the series Genetic Programming pp 301-315, 2005.

[23] J. R. Koza, “Hierarchical genetic algorithms operating on populations of computer programs”, IJCAI'89 Proceedings of the 11th international joint conference on Artificial intelligence - vol 1, pp. 768-774, 1989.

[24] A. Andreopoulos, John K. Tsotsos, “50 Years of object recognition: Directions forward”, Computer Vision and Image Understanding, vol. 117 (8), pp. 827-891, 2013.

[25] Andreopoulos, S. Hasler, H. Wersing, H. Janssen, J. K. Tsotsos, E. Korner, “Active 3D Object Localization using a humanoid robot”, IEEE Transactions on Robotics, vol. 27 (1), pp. 47–64. 2011.

[26] J. Tsotsos, “The Encyclopedia of Artificial Intelligence”, chap. Image Understanding, John Wiley and Sons, pp. 641–663, 1992.

[27] S. Dickinson, “What is Cognitive Science?”, chap. Object Representation and Recognition, Basil Blackwell Publishers, pp. 172–207, 1999.

[28] W. K. Pratt, “Digital Image Processing”, John Wiley & Sons, ISBN: 0-471-37407-5, 2001.

[29] I. Biederman, G. Ju, “Surface versus edge-based determinants of visual recognition”, Cognitive Psychology, vol. 20, pp. 38–64, 1988.

[30] L. K. Samuelson, L. B. Smith, “They call it like they see it: Spontaneous naming and attention to shape”, Developmental Science, vol 8 (2), pp. 182–198, 2005.

[31] D. K. Prasad, “Survey of the problem of object detection in real images.“ International Journal of Image Processing (IJIP), vol. 6.6, pp. 441. 2012.

[32] R. Singh, R.M. Voyles, D. Littau, N.P. Papanikolopoulos, “Shape recognition and vision-based robot control by shape morphing”, Information Intelligence and Systems. Proceedings, International Conference on IEEE, pp. 188-195. 1999.

[33] M. Franzius, H. Wersing, U.S. Patent No. 8,731,719. Washington, DC: U.S. Patent and Trademark Office, 2014.

[34] X. Chen, J, K, Udupa, U. Bagci, Y. Zhuge, J. Yao, “Medical image segmentation by combining graph cuts and oriented active appearance models”, IEEE Transactions on Image Processing, vol. 21 (4), pp. 2035-2046. 2012.

[35] A. K. Jain, H. Chen, “Matching of dental X-ray images for human identification”, Pattern Recognition, vol. 37(7), pp. 1519-1532. 2004.

[36] J. Du, D. Huang, X. Wang, X. Gu. “Shape Recognition Based on Radial Basis Probabilistic Neural Network and Application to Plant Species Identification”, Advances in Neural Networks, vol 3497. Springer, Berlin, Heidelberg, 2005.

[37] M. Saadat, P. Nan, “Industrial applications of automatic manipulation of flexible materials”, Industrial Robot: An International Journal, Vol. 29 (5), pp. 434-442, 2002.

[38] Ch. Teutsch, D. Berndt, E. Trostmann, M. Weber, “Real-time detection of elliptic shapes for automated object recognition and object tracking”, Proc. SPIE 6070, Machine Vision Applications in Industrial Inspection XIV, 60700J, 2006.

[39] S. Vishwakarma, A. Agrawal, “A survey on activity recognition and behavior understanding in video surveillance.” The Visual Computer, vol. 29. 10, pp. 983-1009. 2013.

[40] I. Sebanja, D. B. Megherbi, “Automatic detection and recognition of traffic road signs for intelligent autonomous unmanned vehicles for urban

surveillance and rescue", Technologies for Homeland Security (HST), 2010 IEEE International Conference on, pp. 132-138, IEEE., 2010.

[41] Rougier, J. Meunier, A. St-Arnaud, "Robust video surveillance for fall detection based on human shape deformation." IEEE Transactions on circuits and systems for video Technology, vol 21.5, pp. 611-622. 2011.

[42] S. L. Tanimoto, "An Interdisciplinary Introduction to Image Processing", Massachusetts Institute of Technology, ISBN: 978-0-262-01716-9, 2012.

[43] Jähne, "Digital Image Processing: concepts, algorithms, and scientific applications", Springer-Verlag Berlin Heidelberg, ISBN: 3-540-56941-3, 1991.

[44] J. W. H. Tangelder, R. C. Veltkamp, "A survey of content based 3D shape retrieval methods," Proceedings Shape Modeling Applications, 2004., Genova, Italy, 2004, pp. 145-156., 2004.

[45] S. Zhang, "Recent progresses on real-time 3D shape measurement using digital fringe projection techniques.", Optics and lasers in engineering, vol 48.2, pp.149-158, 2010.

[46] H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, "Multi-View Convolutional Neural Networks for 3D Shape Recognition", The IEEE International Conference on Computer Vision (ICCV), 2015, pp. 945-953, 2015.

[47] F. Wang, L. Kang, Y. Li, "Sketch-Based 3D Shape Retrieval Using Convolutional Neural Networks", The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1875-1883, 2015.

[48] M. Attene, F. Robbiano, M. Spagnuolo, B. Falcidieno, "Characterization of 3D shape parts for semantic annotation", Computer-Aided Design, vol. 41 (10), pp. 756-763, 2009.

[49] M. C. Chang, B. B. Kimia, "Measuring 3D shape similarity by graph-based matching of the medial scaffolds", Computer Vision and Image Understanding, vol. 115 (5), pp. 707-720, 2011.

[50] J. C. Russ, "The Image Processing Handbook", CRC Press, 1995, ISBN: 0-8493-2516.

[51] S. A. Dudani, K. J. Breeding, R. B. McGhee, "Aircraft identification by moment invariants," IEEE Transactions on Computers, vol. 26 (1), pp. 39–46, 1977.

[52] L. Gupta and M. D. Srinath, “Contour sequence moments for the classification of closed planar shapes,” Pattern Recognition, vol. 20, no. 3, pp. 267–272, 1987.

[53] C. Chang; D. J. Buehrer; S. M. Hwang, “A shape recognition scheme based on relative distances of feature points from the centroid”, Pattern Recognition, Vol: 24, Issue: 11, pp 1053-1063, 1991.

[54] R. Davies, “Machine Vision: Theory, Algorithms, Practicalities“, vol. 54, Academic Press, New York, NY, USA, 1991.

[55] P. J. van Otterloo, “A Contour-Oriented Approach to Shape Analysis“, Prentice-Hall International (UK) Ltd, New Jersey, NJ, USA, 1991.

[56] A. C. Evans, N. A. Thacker, and J. E. W. Mayhew, “Pairwise representation of shape,” in Proceedings of the 11th IAPR International Conference on Pattern Recognition, vol. 1, pp. 133–136, IEEE, The Hague, Netherlands, 1992.

[57] M. Sonka, V. Hlavac, and R. Boyle, “Image Processing, Analysis and Machine Vision”, Chapman and Hall Computing, Boca Raton, Fla, USA, ISBN: 978-0-412-45570-4, 1993.

[58] H. Asada and M. Brady, “The Curvature Primal Sketch,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 1, pp. 2–14, 1986.

[59] Eichmann et al., “Shape representation by Gabor expansion,” in Proceedings of the Hybrid Image and Signal Processing II, vol. 1297 of SPIE, pp. 86–94, 1990.

[60] Q. M. Tieng and W. W. Boles, “Recognition of 2D object contours using the wavelet transform zero-crossing representation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 8, pp. 910–916, 1997.

[61] Freeman, “On the encoding of arbitrary geometric configurations,” IRE Transactions on Electronic Computers, vol. EC-10, no. 2, pp. 260–268, 1961

[62] W. I. Grosky and R. Mehrotra, “Index-based object recognition in pictorial data management,” Computer Vision Graphics and Image Processing, vol. 52, no. 3, pp. 416–436, 1990.

[63] S. Belongie , J. Malik , J. Puzicha, “Shape Matching and Object Recognition Using Shape Contexts”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 24., pp 509-522, 2001

[64] D. Zhang, G. Lu, “Review of shape representation and description techniques”, Pattern Recognition 37, pp.1 – 19, 2004

[65] M. K. Hu, “Visual pattern recognition by moment invariant,” IRE Transactions on Information Theory, vol. 8, no. 2, pp. 179–187, 1962.

[66] Shin Kim and Heung - Kyu Lee, “Invariant Image Watermark Using Zernike Moments”, IEEE Trans. on Circuits and System for Video Technology, 2003

[67] A. Khotanzad and Y.H. Hong, “Invariant image recognition by zernike moments”, IEEE Trans. on Pattern Anal. and Machine Intell., 1990.

[68] R. B. Yadava, N. K. Nishchalb, A. K. Guptaa, V. K. Rastogi, “Retrieval and classification of objects using generic Fourier, Legendre moment, and wavelet Zernike moment descriptors and recognition using joint transform correlator”, Optics & Laser Technology, vol. 40 (3), pp. 517-527, 2008

[69] C.-H. Teh and R. T. Chin, “On Image Analysis by the Methods of Moments,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, pp. 496-513, 1988.

[70] P. Rosin, J. Zunic, “Handbook of Applied Algorithms: Solving Scientific, Engineering, and Practical Problems, 2D Shape Measures for Computer Vision”, Wiley-IEEE Press, ISBN: 978-0-470-04492-6, 2008.

[71] Flusser, T. Suk, B. Zitova, “Moments and Moment Invariants in Pattern Recognition”, John Wiley & Sons, ISBN: 978-0-470-69987-4, 2009.

[72] D. Zhang and G. Lu, “Shape-based image retrieval using generic Fourier descriptor,” Signal Processing: Image Communication, vol. 17, no. 10, pp. 825–848, 2002.

[73] D. Zhang, G. Lu. “Generic Fourier descriptor for shape-based image retrieval.” ICME (1). 2002.

[74] S. Li, M.-C. Lee, and C.-M. Pun. “Complex Zernike moments features for shape-based image retrieval”, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol 39. (1) , pp. 227-237., 2009.

[75] M. Vorobyov, “Shape classification using zernike moments”, Technical Report, iCamp-University of California Irvine, 2011.

[76] M. R. Teague, “Image analysis via the general theory of moment invariants”, Journal of the Optical Society of America, vol. 70. (8), pp. 920-930, 1980

[77] A. Tahmasbi, F. Saki, and S. B. Shokouhi, “Classification of benign and malignant masses based on Zernike moments,” Computers in Biology and Medicine, vol. 41, no. 8, pp. 726–735, 2011.

[78] S.-K. Hwang, W.-Y. Kim, “A novel approach to the fast computation of Zernike moments”, Pattern Recognition, vol 39. (11), pp. 2065-2076, 2006.

[79] E. R. Kandel, J. H. Schwartz, T. M. Jessel, S. A. Siegelbaum, A. J. Hudspeth, (Ed.), “Principles of Neural Science”, E. R. Kandel (Ed), McGraw-Hill Companies, ISBN: 987-0-07-139011-8, 2013.

[80] M. Yagi, T. Shibata, “An Image Representation Algorithm Compatible with Neural-Associative-Processor-Based Hardware Recognition Systems,” IEEE Trans. Neural Networks, Vol. 14, No. 5, pp. 1144-1161, 2003.

[81] D. Geer, “Chip makers turn to multicore processors“, Computer, IEEE, vol 38. (5), pp 11-13. 2005.

[82] T. Roska, P. Giovanni, W. W. Chai. “Cellular wave computing via nanoscale chip architectures.“ International Journal of Circuit Theory and Applications 40.12, pp. 1187-1189. 2012.

[83] Cs. Nemes, Z. Nagy, M. Ruszinkó, A. Kiss, P. Szolgay, “Mapping of high performance data-flow graphs into programmable logic devices.“ Proceedings of International Symposium on Nonlinear Theory and its Applications,(NOLTA 2010). 2010.

[84] M. S. Kulkarni, Ch. Teuscher. “Memristor-based reservoir computing.“ Proceedings of the 2012 IEEE/ACM International Symposium on Nanoscale Architectures. ACM, pp. 226-232. 2012.

[85] Y. Zhang, W. Zhao, J. O. Klein, W. Kang, D. Querlioz, Y. Zhang, “Spintronics for low-power computing.” Proceedings of the conference on Design, Automation & Test in Europe. European Design and Automation Association, pp. 303, 2014.

[86] D. Morin, Introduction to Classical Mechanics, Chapter 4. – Oscillations, p. 101-119. Cambridge University Press, ISBN 978-0-521-87622-3, 2008.

[87] P. Tomcsányi (ed), Fizika – Mechanika, Chapter 10., Mechanical oscillations (10. fejezet – Mechanikai Rezgések), p 227-265. Műszaki Tankönyvkiadó, ISBN 963-16-2275-4, 2015.

[88] R. N. Chaudhuri, “Waves and Oscillations”, New Age International (P) Limited, ISBN: 81-224-1291-2, 2001.

[89] U. Ingard, “Fundamentals of Waves & Oscillations”, Cambridge University Press, ISBN: 0-521-32734-2, 1988.

[90] D. Landau, E. M. Lifshitz, “Theory of the dispersion of magnetic permeability in ferromagnetic bodies“, Phys. Z. Sowietunion, 8, 153, 1935.

[91] A. Slavin: “Microwave sources: Spin-torque oscillators get in phase”, Nature Nanotechnology 4, pp. 479-480. 2009.

[92] S. E. Russek, W. H. Rippard, T. Cecil, R. Heindl, “Spin-transfer nano-oscillators“, Handbook of Nanophysics: Functional Nanomaterials, 2010.

[93] G. Bourianoff, “Towards a Bayesian processor implemented with oscillatory nanoelectronic arrays“, Keynote lecture at IEEE International Workshop on Cellular Nanoscale Networks and Applications, Turin, Italy, 2012.

[94] Gy. Csaba, Á. Papp, W. Porod, “Perspectives of using spin waves for computing and signal processing“, Physics Letters A, vol. 381 (17), 2017.

[95] A. Horváth, F. Corinto, Gy. Csaba, W. Porod, T. Roska, “Synchronization in Cellular Spin Torque Oscillator Arrays,” 13th IEEE International Workshop on Cellular Nanoscale Networks and Applications, Turin, Italy, 2012.

[96] O. Chua and T. Roska, Cellular Neural Networks and visual computing. Cambridge, UK: Cambridge University Press, 2002.

[97] T. Roska and L. O. Chua, “The CNN universal machine: an analogic array computer,” IEEE Trans. Circuits Syst. II, vol. 40, pp. 163–173, Mar. 1993.

[98] Kék, K. Karacs, and T. Roska. (2007) Cellular wave computing library, version 2.1 (templates, algorithms and programs). [Online]. Available: http://cnn-technology.itk.ppke.hu/Library v2.1b.pdf visited on 02-12-2009.

[99] K. Karacs, A. Lázár, R. Wagner, D. Bálya, T. Roska, and M. Szuhaj, “Bionic Eyeglass: an Audio Guide for Visually Impaired,” in Proc. of the First IEEE Biomedical Circuits and Systems Conference (BIOCAS 2006), London, UK, Dec. 2006, pp. 190 – 193.

[100] K. Karacs, A. Lázár, R. Wagner, B. Bálint, T. Roska, and M. Szuhaj, “Bionic Eyeglass: The First Prototype, A Personal Navigation Device for Visually Impaired,” in Proc. of First Int’l Symp. on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2008), Aalborg, Denmark, 2008

[101] T. Roska, D. Bálya, A. Lázár, K. Karacs, R. Wagner, and M. Szuhaj, “System aspects of a bionic eyeglass,” in Proc. of the 2006 IEEE International Symposium on Circuits and Systems (ISCAS 2006), Island of Kos, Greece, May 21–24, 2006, pp. 161–164. 2006.

[102] K. Karacs, Á. Kusnyerik, M. Szuhaj, T. Roska, “Situation-specific Scene Interpretation in a Bionic Navigation Device for Visually Impaired, International Conference on Vision in 3D Environments, Toronto, Canada, Jul, 2009.

[103] K. Karacs, M. Radványi, Á. Kusnyerik, T. Roska, “Basic Scene Understanding and Navigation with a Bionic Camera,” International Conference on The Eye and The Auto, Detroit, MI, Sept, 2009

[104] K. Karacs, Á. Kusnyerik, M. Radványi, T. Roska, M. Szuhaj, “Towards a Mobile Visual Navigation Device”, CNNA2010, 2010.

[105] Radvanyi, G. E. Pazienza, and K. Karacs, “Crosswalk Recognition through CNNs for the Bionic Camera: Manual vs. Automatic Design,” in Proc. of the European Conference on Circuit Theory and Design (ECCTD’09), Antalya, Turkey, Aug, 2009.

[106] K. Karacs, M. Radványi, M. Görög, T. Roska, “A Mobile Visual Navigation Device: New algorithms for crosswalk and pictogram recognition,” in Proc. of 2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2009), Bratislava, Slovakia, Nov, 2009.

[107] M. Radvanyi, K.Karacs, “Navigation through Crosswalks with the Bionic Eyeglass” in Proc. of 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2010), Rome, Italy, 2010.

[108] M. Radványi, K. Karacs, “Autonomous Detection of Information Patterns Through Hierarchical Peeling,” in Proc. of the 21st European Conference on Circuit Theory and Design (ECCTD 2013), Dresden, Germany, Sept. 2013.

[109] M. Radványi, K. Karacs, “Locating Pattern Groups in Segmented Color Images,” in Proc. of the 14th IEEE International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2014), Notre Dame, IN, July 2014.

[110] K. Karacs, M. Radványi, Z. Nagy, “Gestalt principle based multipart object and object group detection on FPGA,” in Proc. of the 14th IEEE International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2014), Notre Dame, IN, July 2014.

[111] M. Fritz, B. Schiele, “Towards Integration of Different Paradigms in Modeling, Representation, and Learning of Visual Categories”, Object Categorization: computer and human vision perspectives, edited by S. J. Dickinson, A. Leonardis, B. Schiele, M. J. Tarr, Cambridge University Press, ISBN: 978-0-521-88738-0, 2009.

[112] E. Pekalska, R. P. W. Duin, “The Dissimilarity Representation for Pattern Recognition: Foundations and Applications”, World Scientific Publishing Co. Pte., ISBN: 981-256-530-2, 2005

[113] J. Iivarinen, M. Peura, J. Srel, A. Visa, “Comparison of Combined Shape Descriptors for Irregular Objects, 8th British Machine Vision Conference,

http://www.cis.hut.fi/research/IA/paper/publications/bmvc97/bmvc97.html, 1997

[114] M. Hasegawa, S. Tabbone: “A Shape Descriptor Combining Logarithmic-scale Histogram of Radon Transform and Phase-only Correlation Function”, International Conference on Document Analysis and Recognition, Beijing, pp.182 – 186, 2011.

[115] S. Khanam, S. Jang, W. Paik, “Shape Retrieval Combining Interior and Contour Descriptors”, International Conference FGCN, Jeju Island, pp.120-128, 2011.

[116] T. G. Dietterich, “An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization”, Article Machine Learning, vol. 40, (2), pp 139-157, 2000.

[117] T. G. Dietterich, “Multiple Classifier Systems”, Chapter Ensemble Methods in Machine Learning, Volume 1857 of the series Lecture Notes in Computer Science pp 1-15, 2000.

[118] D. Opitz, R. Maclin, “Popular Ensemble Methods: An Empirical Study”, vol. 11, pp. 169-198, 1999.

[119] M. Yang, Kidiyo Kpalma, Joseph Ronsin. “A Survey of Shape Feature Extraction Techniques”, Peng-Yeng Yin. Pattern Recognition, IN-TECH, pp.43-90, 2008.

[120] G. Carneiro, A. B. Chan, P. J. Moreno, N. Vasconcelos, “Supervised Learning of Semantic Classes forImage Annotation and Retrieval”, IEEE T. on Pattern Analysis and Machine Intelligence, 2007.

[121] B. Kotsiantis, Sotiris, I. D. Zaharakis, P. E. Pintelas. “Supervised machine learning: A review of classification techniques.” vol. 3-24, 2007.

[122] R. C. Gonzalez, R. E. Woods. “Digital Image Processing”. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, ISBN: 978-0131687288, 2001.

[123] M. Riesenhuber, T. Poggio, “Hierarchical models of object recognition in cortex,” Nature Neuroscience, vol. 2, no. 11, pp. 1019-1025, 1999.