Politechnika Łódzka
Instytut Informatyki

PRACA DYPLOMOWA MAGISTERSKA

SEMANTIC IMAGE SEGMENTATION WITH USE OF CONDITIONAL RANDOM FIELDS

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej
Promotor: dr inż. Arkadiusz Tomczyk
Dyplomant: inż. Aneta Andrzejewska
Nr albumu: 215126
Kierunek: Informatyka
Specjalność: Computer Science and Information Technology

Łódź, 18.02.2019

Instytut Informatyki
90-924 Łódź, ul. Wólczańska 215, budynek B9
tel. 042 631 27 97, 042 632 97 57, fax 042 630 34 14 email: office@ics.p.lodz.pl

Abstract

Semantic image segmentation is one of the tasks of computer vision that is crucial for understanding and interpreting images. There are a lot of different methods aimed to perform it, with the majority being based on machine learning. The primary purpose of this study was to examine the possibility of developing an image semantic segmentation system with use of Conditional Random Fields. Multiple different algorithms were studied and tested, in order to provide a method capable of fulfilling the thesis purpose. In this study, three main procedures were developed. Firstly, images were modelled with the use of factor graphs, which represented image features and labelling outcomes, as well as the relations between them. Feature selection was obtained by a stepwise regression process, while a feature function was based on an estimation of the probability density function of all features and available labels. For parameter learning Stochastic Gradient Descent was chosen, which included few simplifications aimed to make the computations feasible and less resource consuming. The final prediction of labelling of unknown images was performed by an inference process with use of Loopy Belief Propagation. The developed method correctly segmented objects in an image that they were differing only by colours if CIELAB colour space was used. By incorporating the contextual data to the model it was also possible to distinguish objects by shape. A large improvement of segmentation, especially in the presence of noise, was obtained by including pairwise relations between regions in a prediction process.

Keywords – semantic image segmentation, Conditional Random Fields, Structured Prediction, factor graphs, Stochastic Gradient Descent, Loopy Belief Propagation

Many thanks to the supervisor of this thesis, dr. Arkadiusz Tomczyk, for his enormous knowledge, invaluable help and understanding throughout the whole work on this thesis.

Contents

Chapter 1

Introduction

Due to the rapid increase in the number and quality of camera systems, which can be now found literally everywhere, the field of computer visions thrives like never before. Digital images and videos are usually used to be viewed by humans, however, they can be also interpreted automatically and the demand for systems capable of doing it is constantly rising. One of the categories of tasks that can be performed by such systems is semantic image segmentation and this dissertation will provide a detailed description of how it can be achieved with the use of Conditional Random Fields.

1.1 Purpose and scope of the thesis

The topic of this thesis lies in the area of machine learning, which is a subset of artificial intelligence aimed to analyse known data, identify its patterns and trends, and

by their generalisation make predictions about new data, without being programmed in an explicit way. The aim of machine learning is to mimic the learning ability of a human brain and apply it for various tasks which require a decision-making process. Applications of machine learning are numerous, and many of them are present in our daily life. An example would be Social Media Services which use machine learning for targeted advertisements, proposals of new friends or for recommendations of potentially interesting groups, sites and products. It is also used in traffic predictions, spam filters, online customer support bots or virtual personal assistants like Siri or Alexa, and in a number of different applications that have hundreds of thousands of daily users. Apart from purely commercial usages, machine learning has also a huge impact on currently performed research in a number of different scientific areas such in a medical field, to better adjust treatments or locate tissue pathologies, or physics, chemistry and astronomy for performing simulations that aid in finding distant planets or new particles [1].

Regardless of an application, any machine learning task needs data to learn from. The training data may be expressed as simple numbers but very often it has a more structured form of for instance images. Semantic image segmentation is one of the main tasks in automatic image recognition systems. It is aimed to group together parts of an image that belongs to the same object class, which carry some semantic meaning. Scene understanding is a necessary part of many technologies that may now be still under development but presumably will become common in a near future, with the most prominent example of autonomous driving. Semantic image segmentation can be tackled with various different methods, among others by using Conditional Random Fields. Thus, the main purpose of this thesis is to develop a system that will be capable of performing semantic image segmentation using this method.

1.2 Goals of the thesis

The first goal of the thesis was to perform theoretical research on the different algorithms that may be useful for semantic image segmentation with the use of Conditional Random Fields. After a thorough analysis of those algorithms, it was necessary to choose such that may be applicable for the dataset used in this thesis and at the same time will not require excessive computational resources. The second main goal was to create a system that will perform a semantic segmentation of the chosen images. This required implementing of the selected algorithms and checking which ones are in fact capable of fulfilling the topic of this thesis and which would require too many resources. Another objective was to test the performance of the system and adjust the details and parameters of the developed algorithms in order to increase the precision of the semantic segmentation.

1.3 Thesis contents

This dissertation will be divided into two main parts, firstly all the theoretical concepts needed to understand the process of semantic image segmentation with the use of Conditional Random Fields will be provided, and next there will be a description of the system that was developed for this thesis, and the performed experiments.

An explanation of the main goal of the thesis will be presented in *Chapter 2: Image Segmentation*. This chapter will include a description of what the semantic image segmentation is, with which methods it can be achieved and what are its possible

applications. Next, *chapter 3: Structured Prediction* will provide the theory behind the core of the created system in terms of procedures and algorithms required to fulfil its purpose. This chapter will begin with an explanation of how the input data are modelled for further processing. Then, the second part of the topic of this thesis will be described, which are Conditional Random Fields. Next section will be devoted to providing information on how the final prediction on an unknown sample is obtained. After that, there will be an explanation of the machine learning part of the system which is a parameter training of the model, which is based on images from the known dataset. The last section will present a way in which images are transformed into features representation that is understandable by the created model.

Chapter 4: Semantic image segmentation system will provide an introduction to what are the goals of the developed system. Moreover, the implementation details in terms of the technological stack will be included and the key components of the system will be briefly described. *Chapters 5: Colour-based semantic image segmentation* and *6: Shape-based semantic image segmentation* will be devoted to the experimental part of this dissertation. They will include a description of what the system is supposed to do and what were the steps needed for implementation. Furthermore, those chapters will contain information on which algorithms were chosen for the specific tasks, how the image dataset is transformed into meaningful data and finally what are the results of the semantic segmentation process. Last chapter, *Discussion and conclusions* will be devoted to the summary and comparison of the presented results, as well as, a description of possible improvements and extensions to the created system.

Chapter 2

Image Segmentation

As the main goal of the thesis is to perform semantic segmentation of images, this chapter will be focused on providing an explanation of this task and description of methods that can be used to achieve it.

In order to present the concept of semantic image segmentation, there is a need to define a more general term, which is image segmentation [2]. It is a process aimed to divide an image into regions that share similar properties, which usually means finding distinct objects and boundaries between them. Depending on an application, those partitions will differ, as the goal is to identify regions that provide relevant data for a given problem and this relevance is specific to the given applications. Segmentation is often conducted at the preprocessing stage as by grouping pixels into larger regions, further analysis is much simpler and less costly in terms of computations. There are many techniques to tackle the problem of segmentation, however, most of them can be described by one of three categories: threshold, edge-based and region-based methods [3]. Thresholding is the simplest approach to image segmentation in

which individual pixels are assigned to a certain region depending on whether their intensity is in a range of intensities of this region. Edge-based methods are aimed to detect pixels that can be identified as edges between objects, which are characterised by a noticeable change in intensity between neighbouring regions. With all edges detected, it is possible to specify boundaries between objects, giving a full segmentation of an image. The last set of techniques, which are region-based methods, are aimed to group pixels that have similar properties into sets called regions, which correspond to objects or parts of objects present on an image. This may be obtained by region splitting, which means dividing regions into subregions until all the pixels in a region are homogeneous in terms of their features. An alternative process is named region growing, which works by joining neighbouring subregions if they share similar properties in order to obtain larger regions. Very often also a hybrid of both algorithms is used.

The presented three methods are the most basic ones when it comes to simple segmentation tasks. Even though, for more complicated tasks they are usually not applicable, they can be partially incorporated into more sophisticated algorithms.

2.1 Semantic Image Segmentation

For some applications segmenting an image is not enough, as it may be important to know what each segmented region present. The aim of segmentation is just to partition a given image into regions, without giving information on what those regions represent. The more complicated task of assigning meaning to image regions is named semantic segmentation. It goes a step further into understanding of

an image, as apart from dividing it into regions, which correspond to objects, the algorithm also recognises what those regions depict, by assigning one of the possible classes to them. Another way of giving a semantic meaning to the image is pixel-wise classification, which means classifying each pixel separately instead of assigning a class to already segmented regions. Either way, the effect is the same. To visualise the objective of semantic image segmentation an example comparing it to ordinary segmentation is presented in Figure 2.1. On the left side, there is a test image presented. It depicts some people standing and other people on bikes in a forest. The middle image shows a segmentation of this image into objects. As visible, all foreground objects, namely people and bikes, were detected. However, only the right image shows what is the meaning of segmented regions. Regions marked in pink were classified as people, while green areas were labelled as bikes.

Figure 2.1: *Difference between instance segmentation and semantic segmentation. Example image (left), instance segmentation (middle), semantic image segmentation (right).*

Source: *The PASCAL Visual Object Classes [4]*

Understanding of which objects are present on an image is a key issue in a large variety of practical applications which require image processing. One of the possible uses of semantic image segmentation is in the automotive industry. Thanks to road segmentation, which include obstacle detection and classification, autonomous driving may be possible. It is also applicable to traffic control systems, or smart parking solutions. Similarly, it is needed for robot vision in order to allow the decision-making process that is based on what is in a field of view of a robot. Furthermore, semantic image segmentation is required for recognition systems, for example, to provide access to a restricted area based on people biometric features. Similarly, a large set of applications for object recognition and classification is connected with medical imaging, as scanning results are in a form of a substantial number of images, which then have to be analysed by a doctor. Thanks to automatic detection tools, for example to locate tissue pathologies from images, diagnosis can be much easier and less time-consuming. Those are only a few examples of possible applications for semantic image segmentation, however, all of them have one thing in common – they require high precision and accuracy of the results, which can be achieved using various techniques.

2.2 Semantic Image Segmentation methods

For semantic image segmentation, far more complex methods are required than for ordinary segmentation. As in case of any segmentation, there is a need of feature extraction and selection that will allow the distinction between objects. However, apart from specifying object boundaries, those features have to carry enough information to differentiate between objects on a higher level, in order to assign the

semantic meaning to them. A conventional approach to this task would be based on manual selection of features, which may include previously mentioned colour intensities or detected edges. However, in practice more sophisticated algorithms for feature extraction are preferred. Instead of describing an image as a whole, some areas that are significantly different from others are detected. Those areas are named regions of interest, as they are the ones on which the segmentation process is focused. Then, for every such region a vector of important features is constructed with the use of feature extraction algorithms. The aim of those algorithms is to take an image as an input and encode information from regions of interest into a series of numbers describing a specific region. They take into consideration features like local spatial information, gradients, histograms, object orientation or texture. Only with a proper selection of features, it will be possible for the segmentation algorithm to not only differentiate one region from the other but also to assign a proper class to each region. For natural images, it is an extremely difficult task as the same object may come in various forms, sizes, shapes, colours etc [5].

Basing on the extracted feature vectors classification algorithms conduct semantic image segmentation. Traditionally, algorithms like Support Vector Machines or Random Decision Forests were involved in this task. Though there are relatively old algorithms, as both were proposed around the 1990s [6, 7], they are still applicable and used for both, classification and regression tasks. Support Vector Machine is a supervised learning algorithm that performs classification based on defining decision boundaries between classes. When it comes to Random Decision Forests, they as well involve supervised machine learning to solve classification and regression tasks. This algorithm is based on composing a classifier ensemble from a set of decision trees which are independent of each other in a decision making process. It is done in order to obtain a more accurate and robust prediction comparing to the

case of a single and complex classifier. Even though those traditional methods are still applicable for classification tasks, lately more advanced, deep learning methods have become dominant in this field.

Though, initial approaches to deep learning for the described task date back to 2000s [8], a breakthrough of using Deep Neural Networks for image segmentation happened in 2014 [9] when such a method of using Fully Convolutional Networks was introduced that allowed image inputs of arbitrary size as opposed to first implementations that required the size to be fixed. Since then, most of the algorithms used for semantic image segmentation have been based on this method. Later in the same year [10], a method combining Fully Convolutional Networks with Conditional Random Fields was proposed, which improved the results by better boundary detection and capturing of details. However, Conditional Random Fields, which will be further explained in this dissertation, can themselves act as an independent classifier that can be used to perform semantic image segmentation.

Chapter 3

Structured Prediction

In various tasks of classification it is not enough to propose an output as a simple scalar value but instead, a complex structure or multiple related outputs like sequences, trees or graphs are needed. One of the examples of such problems is natural language processing, for instance aimed to assign parts of speech for each word in a sentence [11]. A different example would be a prediction of a structure of biological sequences like proteins or genes [12]. Furthermore, such tasks are common in computer vision to detect and categorise objects present in an image or a video [13]. What is characteristic of all these kinds of applications is that very often the dimensionality of inputs is very large. As a result, a number of potential outcomes can be enormous, making the classification computationally highly demanding or even impossible. A framework that allows solving such problems is called Structured Prediction. It is a technique involving supervised machine learning algorithms in which the output is treated as one object instead of a set of individual values. Such an object has a complex structure containing a number of

related entities, which can involve constraints or dependencies. Taking into account those relations is a key concept of Structured Prediction.

In this chapter, an explanation of definitions and terms connected with Structured Prediction will be provided, which are necessary to understand this dissertation. First, the concept of probabilistic graphical models will be described, followed by an overview of the factorisation process. In the next section, Conditional Random Fields, which are a part of the topic of this dissertation, will be introduced. After that, a process of inference in factor graphs and parameter training will be described. Last, but not the least an overview of different methods of energy formulation will be provided.

3.1 Probabilistic Graphical Model

In machine learning very often there is a need to construct a mathematical model that represent real-life objects in terms of data essential to a given problem. An example of such representations is a graphical model, which may reflect not only relations between observations and quantities of interest but also between different observations. Such models offer a compact and intuitive way of representing a large set of related variables, which is of high use in Structured Prediction.

Depending on an application, elements of a graph represent different objects, however, in the most basic form a graphical model can be presented as such a model that is composed of a set of nodes being a reflection of each random variable. If two or more variables are related, then their associated nodes are connected with an edge and with each edge, there may be a vector of parameters, or weights assigned that

describe how much one variable is depended on the other. Also, a graph can model other information than relations between nodes. Giving an example, for image segmentation each node represents a pixel in an image, and edges describe relations between individual pixels. Apart from a layer of interconnected pixel nodes, there can be an additional layer created, with each node representing a class label. Then, between a pixel node and its label node, there is also an edge that describes in a clear way the relations between a pixel and its label.

The goal of constructing graphical models is to propose a data representation that will allow finding a solution for the specified problem in a clear and direct way. However, for some applications, especially for classification, it is impossible to propose a well-defined model that will give an answer with certainty. In such situations, it is beneficial to propose a representation of an object that encodes not direct, but statistical dependencies between observations and quantities of interest and probabilistic graphical models offer such functionality. By using them it is possible to obtain not a single, definite answer but a full probability distribution of answers, which is a mathematical function representing each possible result with a probability of its occurrence.

Depending on a kind of a graph this probability distribution is modelled in a different way. There are two main types of graphical models, directed and undirected. Directed graphs, also known as Bayesian networks, have edges that have a direction from one node to the another, meaning that there is a one-way dependence between variables. Using those graphs causality between different variables can be modelled in a clear way. Bayesian networks encode conditional probability of every label of a node given that its parent nodes have a given label assigned. However, in many applications, especially in computer vision, variables do not have this kind

of interactions. This is also a case in image processing, as there is no hierarchy between pixels in an image. Such problems can be modelled with undirected graphs named Markov Random Fields, which form a base of representing a problem domain in Structured Prediction. In this case, instead of having conditional probability based on parent nodes, a joint probability distribution is modelled, which encodes the probability of every label of each variable occurring simultaneously. Unfortunately, for large input space, this involves a lot of data processing. In order to store information about it in an explicit way, for example in the form of a table, an enormous number of cells would be needed as every combination of all the variables' possible values should be stored in a separate cell. Because of the fact that in Structured Prediction a quantity of variables is typically large, there is a need to propose different methods of encoding joint probability distributions that just a table and graphical models offer such functionality. In Markov Random Fields, all not connected variables are assumed to be conditionally independent, which means that variables depend only on their closest neighbours. Because of this, it is possible to decompose large distribution functions into smaller distributions containing only related variables, which significantly reduces the number of data needed to be stored and therefore the number of computations needed [14].

3.2 Factorisation

A process of decomposing a joint probability distribution is named factorisation. It is performed to obtain a such a set of factors which product is equal to a given joint probability distribution. Therefore, factorisation can largely decrease the complexity of calculations as it is only required to calculate the outcome of each factor

within its scope that is usually much smaller than the whole input domain. A base for defining a factor in undirected graphs is a clique which is defined as a subset of nodes in which each variable is connected to every other variable in a set. A clique which is not a subset of any other clique, and therefore cannot be extended by adding any more nodes, is called a maximal clique [15]. Given an undirected graph $G = (V, \mathcal{E})$ modelled with a set of nodes, also called vertices V, and a set of edges $\mathcal{E}$ a family of joint probability distributions can be obtained as a result of the factorisation process. For a single realisation over all random variables in a model that is denoted as y, a joint probability $p(y)$ is understood as a normalised product of factors ψ_c of every clique c from a set of all maximal cliques C found in the graph G. This is presented in equation 3.1, in which y_c denotes all variables nodes from a given clique c.

$$p(y) = \frac{1}{Z} \prod_{c \in C} \psi_c(y_c) \tag{3.1}$$

In those graphs, factors also called clique potentials, do not represent conditional probability. Instead, they are non-negative functions with a property that values with higher potentials are more probable to occur. Thus, a family of joint probabilities does not need to sum up to 1, and therefore, a normalising constant Z is introduced. This constant, also known as a partition function, is expressed as a summation over joint probabilities of all states y in an overall output domain $\mathcal{Y}$.

$$Z = \sum_{y \in \mathcal{Y}} \prod_{c \in C} \psi_c(y_c) \tag{3.2}$$

Such factorisation can be presented on a fully connected graph, however, it is not very convenient for further processing. A far more intuitive way of modelling factorisation of a joint probability distribution is to use factor graphs, which are a type of undirected, probabilistic graphical models. Such graphs contain two kinds of nodes – variable nodes, and factor nodes, with each factor node being connected to

variable nodes that depend on it. A sample factor graph created for six random variables is depicted in Figure 3.1, where variable nodes are marked as green circles, and factor nodes as black rectangles. As presented, between every pair of factor and variable nodes there is an edge marked.

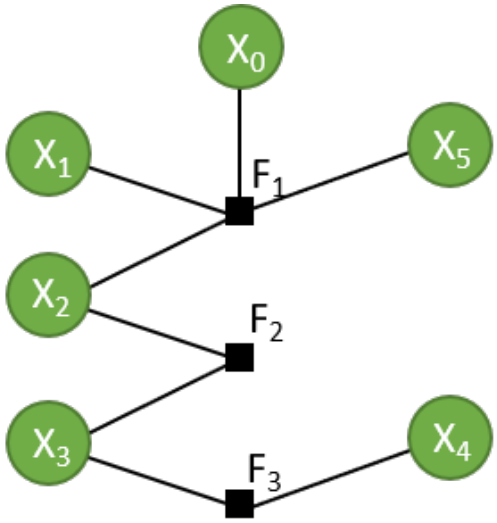


Figure 3.1: *Sample scheme of a factor graph with variable nodes marked as green circles and factor nodes as black rectangles.*

Then to calculate joint probability $p(y)$ it is enough to compute a product of each factor $F \in \mathcal{F}$ and normalise it with normalising constant Z as in equations 3.3 and 3.4. The scope of the factor F is denoted as $N(F)$ and it represents the set of all variable nodes that are adjacent to this factor.

$$p(y) = \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(y_{N(F)}) \tag{3.3}$$

$$Z = \sum_{y \in \mathcal{Y}} \prod_{F \in \mathcal{F}} \psi_F(y_{N(F)}) \tag{3.4}$$

Hence, for the sample factor graph from Figure 3.1 the joint probability $p(y)$ is equivalent to product of the factor F_1 dependent on nodes x_5, x_0, x_1, x_2, then the

factor F_2 with a scope of x_2, x_3, and factor F_3 which is dependent on x_3, x_4, as in equation 3.5.

$$p(y) = \frac{1}{Z}(\psi_{F_1}(x_5, x_0, x_1, x_2, y_{N(F_1)}) \cdot \psi_{F_2}(x_2, x_3, y_{N(F_2)}) \cdot \psi_{F_3}(x_3, x_4, y_{N(F_3)})) \quad (3.5)$$

For directed graphs it was clear that factors represent conditional probabilities, however, for undirected graphs, the meaning of factors is not so strictly defined. The only requirement for them is to be non-negative as probability cannot be lower than 0. One of the easiest ways of ensuring that factors will output a non-negative value is to present a problem in an exponential domain, which is obtained with the use of energy function. Energy is such a function which exponential represents a factor potential as in equation 3.6.

$$\psi_F(y_{N(F)}) = \exp\Big(-E_F(y_{N(F)})\Big) \quad (3.6)$$

Then, the joint probability $p(y)$ can be expressed as a summation of energies for each factor as in formula 3.7.

$$\begin{aligned} p(y) &= \frac{1}{Z}\prod_{F\in\mathcal{F}}\Big(\exp\Big(-E_F(y_{N(F)})\Big)\Big) \\ &= \frac{1}{Z}\exp\Big(-\sum_{F\in\mathcal{F}} E_F(y_{N(F)})\Big) \end{aligned} \quad (3.7)$$

For the presented calculations, the normalising constant Z has the following form:

$$Z = \sum_{y\in\mathcal{Y}} \exp\Big(-\sum_{F\in\mathcal{F}} E_F(y_{N(F)})\Big) \quad (3.8)$$

Similarly to a factor potential, energy represents how accurate is a given solution [16]. The evaluation is done by assigning a scalar value to each configuration of random variables with an interpretation that the lower an energy is, the more probable is a given configuration. Then to obtain a state $y \in \mathcal{Y}$ that has the highest

probability, it is enough to specify such a configuration which will have the lowest energy, as presented in formula 3.9.

$$\arg\max_{y \in \mathcal{Y}} p(y) = \arg\min_{y \in \mathcal{Y}} \sum_{F \in \mathcal{F}} E_f(y_{N(F)}) \tag{3.9}$$

Factorisation is a highly convenient process for supervised learning in which the number of inputs and possible outputs is extensive, as it largely reduces the complexity of the generated model. Furthermore, factor graphs increase the efficiency of algorithms further described in this dissertation, which are needed to solve prediction problems.

3.3 Conditional Random Fields

For the task of classification, in which observations are globally known, it is not necessary to model joint probability distribution over all possible outputs and observed values. Instead, a conditional distribution $p(y|x)$ is modelled, which represent a distribution of output variables on condition that input variables take an observed form. One of the methods that aim to find the conditional distribution between variables is named Conditional Random Fields [17]. This is a type of Markov Random Fields that can be viewed as undirected graphical models, which represent both, observations and output labels. With the use of factor graphs, factorisation of a probability distribution can be modelled. Then, those graphs contain two different types of variable nodes, inputs and outputs [18]. The first ones are observed input variables x_i, which definition is dependent on an application. For semantic image segmentation, x_i would denote individual pixels from a single image x taken from the set of all available images X. Input nodes are represented in terms of feature vectors

containing observed properties of an individual image pixel expressed in a numerical form. A great advantage of Conditional Random Fields is that input nodes can involve a large variety of different features as long as they can be transformed into a form understandable by the algorithm. Input variable nodes are connected via factors to the second type of nodes, which are known as hidden nodes, as they cannot be directly obtained from an observed object. Those nodes model a configuration of output variables denoted as y. For the task of semantic image segmentation, an output variable y_i would represent a label assigned to a given pixel from a set of all possible labels $\mathcal{Y}$, and y would be a labelling for a whole image. Output nodes are as well connected with each other with the use of factors making it possible to take into consideration relations between neighbouring entities like pixels in an image. Figure 3.2 shows how such a graph can be constructed on an example of an image with size 3×3 pixels.

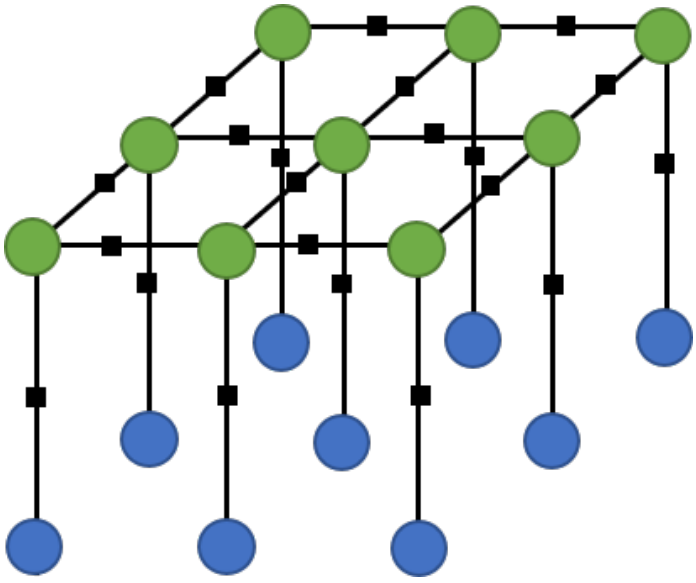


Figure 3.2: *Factor graph representing an image* 3×3 *pixels with input nodes marked in blue, and output nodes in green.*

On the graph, blue circles represent input nodes with observed variables, and green circles a hidden layer of output nodes. Factors, which join each output node to the corresponding input node as well as every neighbouring output nodes are marked as black squares. Between each pair factor – node there is an edge presented as a line, which connects them. This is the simplest type of the graph which is modelled on a regular grid with relations only between input and output nodes, and neighbouring output nodes. However, it is also possible to introduce other dependencies, for example between an output node and observation nodes related to neighbouring output nodes. For the sake of clearer explanation in the following equations it will be assumed that there are relations between one input and one output node, and between two neighbouring output nodes only.

Conditional Random Fields model distribution $p(y|x)$ of a single labelling y based on observed features of an image x. This distribution can be expressed in terms of energy as in formula 3.10, with constant Z computed as in equation 3.11.

$$p(y|x,w) = \frac{1}{Z(x,w)} \exp\left(-E(x,y,w)\right) \tag{3.10}$$

$$Z(x,w) = \sum_{y \in \mathcal{Y}} \exp\left(-E(x,y,w)\right) \tag{3.11}$$

Then, assuming an underlying factor graph, the energy of the whole system can be expressed as a summation of energies of individual factors, as it has already been presented in formula 3.7. However, for Conditional Random Fields a factor graph is composed of two types of nodes, input and output nodes. As a result, there are two sets of factors, a set $\mathcal{F}_1$ of factors between an output node and an input node, and a set $\mathcal{F}_2$ of factors between two neighbouring output nodes. Hence, the energy of the whole system can be expressed as two summations, one over the set of factors $\mathcal{F}_1$ and the second one over a set $\mathcal{F}_2$. As each factor $F \in \mathcal{F}_1$ is bound exactly to one input node and one output node the first summation over

the set $\mathcal{F}_1$ may be expressed as a summation over all output nodes $i \in V$, which will compute energies for factors that are between the current output node an input node assigned to it. Similarly, the summation over factors from $\mathcal{F}_2$ is equivalent to summation over every pair of output nodes $i, j \in V$ that will compute energy for a factor that is between them. Hence, the formula for energy for a factor graph modelling Conditional Random Fields is presented in equation 3.12.

$$\begin{aligned} E(x,y,w) &= \sum_{F \in \mathcal{F}_1} E_F(x_F, y_F, w) + \sum_{F \in \mathcal{F}_2} E_F(y_F, w) \\ &= \sum_{i \in V} E_1(y_i, x_i, w) + \sum_{i,j \in V} E_2(y_i, y_j, w) \end{aligned} \tag{3.12}$$

Factor energy is defined differently depending on the factor type. Energy E_1 of factors $\mathcal{F}_1$ is called a unary potential. It models a relationship between observed features and a predicted label. It should assign high values of energy if a wrong label was assigned to an output node of a given input node, and low energy values for proper assignments. It is used to promote a situation in which two nodes which are similar in terms of their feature vectors have the same label. Energy E_2 for factors from the set $\mathcal{F}_2$ is named a pairwise potential and it reflects a relation between labels of neighbouring output nodes. It introduces a penalty if neighbouring output nodes have a different label by assigning high energy values to such configurations, which as a result lower probability of their occurrence. What is more, factor energy can be parametrised with a set of weights w and then, energy is dependent on three types of variables, inputs, outputs, and parameters [19]. A more detailed explanation of energy function and its formulation will be provided in *section 3.6: Energy formulation*.

In order to make predictions about an unobserved object, it is necessary to find such a combination of output variables in a factor graph that will give the lowest

energy, as this would be a most probable setting. However, it is computationally intractable to calculate graph energy for every possible label configuration especially for problems involving large data structures. On an example of semantic image segmentation, even for a small problem of segmenting an image with size 100×100 pixels into two classes, it would require to calculate energy for $2^{100 \times 100}$ combinations, which is already impossible to compute, and the real-life problems are much more complicated. Because of this, in most cases, it is impossible to obtain an exact solution for classification problems based on factor graph energy. Though, it is possible to approximate a solution.

3.4 Inference in factor graphs

A correctly defined model, either by training or by expertise, is used to make predictions about the unobserved data in a process called inference. There are two most popular approaches to this task named marginal and maximum a posteriori (MAP) inference. Marginal inference is aimed to compute marginal distributions, which give a probability of a variable taking a given value as opposed to every other possibility. An example of it would be to segment an image into foreground and background, where one object is considered foreground and all other objects are assigned as background. On the other hand, the goal of MAP inference is to find the most probable configuration of unobserved variables' values basing on the values of observed data. Both types of inferences are used in factor graphs and consist of establishing observed data and finding such values of output variables that minimise the total energy. In the thesis for the task of minimising the energy MAP inference was used. The reason behind this choice is that MAP inference outputs directly

a state $y^* \in \mathcal{Y}$ that has the maximal probability of occurrence given an underlying factor graph, observation x, and learned weight vector w and this is the goal of semantic image segmentation.

State y^* represents the most optimal configuration of pixel labels for a given image. Finding the optimal state would require calculating energy for each possible configuration of outputs, which is too computationally demanding to be performed. However, there are methods to provide the final result without a need of extensive calculations. Two most popular types of methods that are used for inference are Monte Carlo methods and variational algorithms [18]. Monte Carlo algorithms are a type of stochastic algorithms which instead of estimating conditional probability distributions for a given test object, base the approximation on a repetitive generation of random samples from a distribution of interest. The idea is that given an enough number of samples their mean will approximate the desired output. On the other hand, variational algorithms are aimed to find a single configuration that is an estimation of a most probable output by changing inference into an optimisation problem. Though Monte Carlo methods are more accurate with a large number of iterations they will converge to the real result, variational algorithms tend to be much faster.

One of the most popular approaches towards variational inference in probabilistic graphical models is named Belief Propagation. It is an iterative framework based on message passing that exchange data between variable and factor nodes [20]. Those messages, also named beliefs, are vectors that can be understood as a measure of certainty of related nodes that other nodes belong to some state. It is based on a concept of dynamic programming, which instead of doing calculations for each factor independently uses a recursive procedure to propagate beliefs throughout the

whole graph. Recursive nature reduces the number of computations because of reusing previously computed data. Belief Propagation was first used to find exact marginals on tree [21], tough it is also applicable in a general case of graphs that may contain loops, however, without a guarantee of finding a global optimum. This method is applicable to both of the previously mentioned inference problems. Sum-product algorithm is used for computing marginal distribution, and max-product algorithm for solving MAP inference.

3.4.1 Sum-Product algorithm

The sum-product algorithm is a method used to compute marginal distributions of a given test sample and a normalising constant Z with the use of a trained model. Those outcomes can be described by formulae 3.13 and 3.14.

$$p(y|x,w) = \frac{1}{Z}\exp\left(-\sum_{f\in\mathcal{F}} E_f(y_f)\right) \tag{3.13}$$

$$Z = \sum_{y\in\mathcal{Y}} \exp\left(-\sum_{f\in\mathcal{F}} E_f(y_f)\right) \tag{3.14}$$

In the Sum-Product algorithm, factor energy can be computed based on information sent from adjacent nodes in a form of messages, which are aimed to propagate observed data throughout the whole model. They are sent through each edge of a factor graph in both directions. The messages sent from variable nodes to factors are denoted as $q_{Y_i\to F}$ and factor-to-variable message can be expressed as $r_{F\to Y_i}$, where Y_i denotes an i^{th} variable node [22]. Variable-to-factor messages depend on previously calculated factor-to-variable messages that are associated with factors adjacent to the given variable. They can be formulated as a summation of messages

from each factor F' from the variable neighbourhood M with an exception of the factor F to which the initial message is directed. Such a belief is then calculated for each variable node i and for every label y from the output domain of a given problem as in formula 3.15.

$$q_{Y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to Y_i}(y_i) \tag{3.15}$$

Similarly, when it comes to computing messages from factors to variables there is a summation of messages from each variable j from the neighbourhood N of the factor F apart from a variable i which is a recipient of the initial message. Furthermore, factor-to-variable messages take into consideration factor potentials expressed in terms of energy. These messages are again calculated for every label y separately. Given the fact that one component of factor energy is a pairwise potential, which is dependent on labels of every variable connected to the factor, while computing these messages there is an extra summation over all possible states of neighbouring variables. This is needed to calculate energy for every factor state y'_F. This state can be understood as a label configuration of variables adjacent to the factor knowing that the variable to which the message is directed has been labelled with y. A formula for a factor-to-variable message is presented below.

$$r_{F \to Y_i}(y_i) = \log \left(\sum_{\substack{y'_F \in \mathcal{Y}_F \\ y'_i = y_i}} \exp \left(-E_F(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{Y_j \to F}(y'_j) \right) \right) \tag{3.16}$$

Both of those kinds of messages are depended on previous computations, however, for trees, there are two situations in which they can be calculated without prior knowledge. For leaf nodes that are only connected to one factor, a variable-to-factor message will be equal to 0, as the neighbourhood $M(i) \setminus \{F\}$ will have no elements. Message from factor to variables can be also calculated for factors that are only

connected to one variable. In such a situation neighbourhood $N(F) \setminus \{i\}$ will have 0 elements and consequently only an energy term is needed for the message to be calculated. If a graph contains no cycles, using those calculated beliefs recursively other messages can be computed [23]. For tree-like structures, message computations start with choosing an arbitrary node as a root node. Then, there is a two-step procedure, which is forward and backward propagation of beliefs. Initially, starting from leaf nodes, messages are computed and passed to the root node. They are then are transferred from every node to their parents after all the messages from the node's children are received. They carry information about an optimal label of the root node. After all messages are computed the propagation starts again but in reverse direction, which is from the root to all leaves. In this step, optimal labels for all non-root nodes are predicted. In those two steps, messages of every edge of the tree-like graph are established and can be used to calculate marginals. Then, for every variable node marginals can be computed based on factor-to-variable messages from all adjacent factors. Again, normalisation with a partition function Z is needed for the marginals to carry probabilistic information. It is defined based on the summation of factor-to-variable messages directed towards the root node r for every possible label y. Formulae below are used to compute both marginals and their normalising constant.

$$p(y_i) = \exp\left(\sum_{F \in M(i)} r_{F \to Y_i}(y_i) - \log Z\right) \tag{3.17}$$

$$\log Z = \log \sum_{y_r \in \mathcal{Y}_r} \exp\left(\sum_{F \in (r)} r_{F \to Y_r}(y_r)\right) \tag{3.18}$$

To obtain a final result, which is the most probable configuration of labels for a set of variables $i \in V$ it is enough to calculate marginals for each variable and every possible label y and to chose the one with the highest probability.

Tough Belief Propagation was designed to be used for non-cyclic, tree-like structures it is also possible to apply its concepts to the general case of graphs that may contain cycles, which are the most common type in computer vision tasks [24]. With graphs that contain loops it is impossible to determine a root node, hence, a previously stated algorithm for exact determination of marginals is inapplicable, however, it can be modified to allow approximating marginals for undirected graphs, which is also a case of Conditional Random Fields. In this algorithm, named Loopy Belief Propagation, messages are first initialised with random variables and then iteratively updated until they converge. This method is based on a concept that the local neighbourhood of nodes can be viewed as a tree, and as a result, message passing algorithms are applicable to it. A transfer of those messages can be done in parallel or in a sequential scheme, in a random or predefined order [25]. For parallel processing, computation of messages for every edge is done simultaneously and after that, they are propagated to neighbours. A sequential scheme is more similar to the original Belief Propagation method as initially messages for one node are computed and then propagated to neighbouring nodes and used for further computations. Though formula for factor-to-variable message $r_{F\to Y_i}$ is the same as in an original algorithm, the equation for messages from variable to factors is modified to contain a normalising factor as in 3.19.

$$\bar{q}_{Y_i\to F}(y_i) = \sum_{F'\in M(i)\setminus\{F\}} r_{F'\to Y_i}(y_i) \tag{3.19}$$

$$\delta = \log \sum_{y_i\in\mathcal{Y}_i} \exp\left(\bar{q}_{Y_i\to F}(y_i)\right) \tag{3.20}$$

$$q_{Y_i\to F}(y_i) = \bar{q}_{Y_i\to F}(y_i) - \delta \tag{3.21}$$

In Loopy Belief Propagation the exact marginal distribution is not known, however, it is possible to compute approximate marginals. Furthermore, as in this algorithm

a notion of a root node does not exist it is not possible to calculate the exact partition function. That is why local normalising constants are used. The final equation for approximate marginals is also normalised according to formulae presented below.

$$\bar{\mu}_i(y_i) = \sum_{F' \in M(i)} r_{F' \to Y_i}(y_i) \tag{3.22}$$

$$z_i = \log \sum_{y_i \in \mathcal{Y}_i} \exp\left(\bar{\mu}_i(y_i)\right) \tag{3.23}$$

$$\mu_i(y_i) = \exp\left(\bar{\mu}_i(y_i) - z_i\right) \tag{3.24}$$

Using the presented method of loopy message propagation it is possible to find the most probable label for each node of a factor graph even for non-tree-like structures.

3.4.2 Max-Product algorithm

Finding the most probable label for each node individually may not result in the most optimal label configuration of the whole structure. MAP inference, with an example of Max-Product algorithm of Belief Propagation, addresses this problem. It is similar to the Sum-Product algorithm, as it is also based on message propagation between nodes and is exact for trees and gives an approximate result for general graphs. When the Max-Product algorithm is transformed to logarithmic space by using energies instead of factor potentials it is called a Max-Sum algorithm, or a Min-Sum if the problem is expressed as a negative log-likelihood. The difference between those algorithms and the Sum-Product method lies in the computation of factor-to-variable messages as instead of marginalisation, a maximisation over fac-

tor states for Max-Sum, or minimisation for Min-Sum variant, is performed. For Max-Sum inference, the formula for messages from a factor to a variable can be presented as in formula 3.25.

$$r_{F\to Y_i}(y_i) = \max_{\substack{y'_F\in\mathcal{Y}_F \\ y'_i=y_i}} \left(-E_F(y'_F) + \sum_{j\in N(F)\setminus\{i\}} q_{Y_j\to F}(y'_j) \right) \tag{3.25}$$

Similarly, in a Min-Sum variant of this algorithm, factor-to-variable messages can be computes as in equation 3.26.

$$r_{F\to Y_i}(y_i) = \min_{\substack{y'_F\in\mathcal{Y}_F \\ y'_i=y_i}} \left(E_F(y'_F) + \sum_{j\in N(F)\setminus\{i\}} q_{Y_j\to F}(y'_j) \right) \tag{3.26}$$

When it comes to computations of variable-to-factor message the only difference is in a formulation of normalising factor, which instead of computing logarithm of a sum of messages in exponential form, takes the average value of those messages just like in formulae below.

$$\bar{q}_{Y_i\to F}(y_i) = \sum_{F'\in M(i)\setminus\{F\}} r_{F'\to Y_i}(y_i) \tag{3.27}$$

$$\delta = \frac{1}{|\mathcal{Y}_i|} \sum_{y_i\in\mathcal{Y}_i} \bar{q}_{Y_i\to F}(y_i) \tag{3.28}$$

$$q_{Y_i\to F}(y_i) = \bar{q}_{Y_i\to F}(y_i) - \delta \tag{3.29}$$

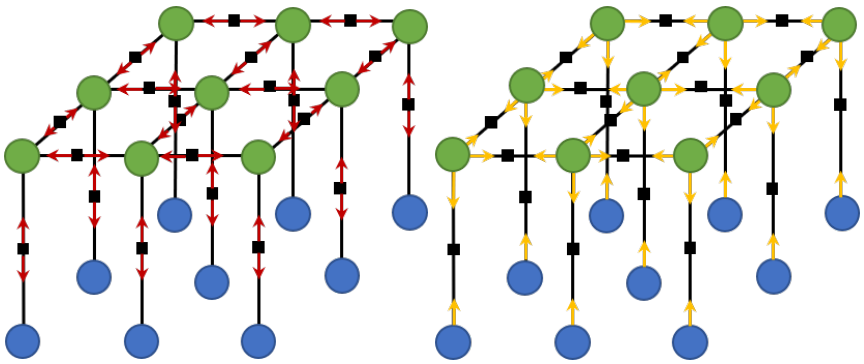
Computing of all messages for every edge of the graph allows finding such a configuration of labels, which has the highest probability of occurrence. The optimal prediction y^* is obtained by determining a state that has the maximal belief for every variable, which is equivalent to finding the state of lowest energy:

$$y_i^* = \arg\max_{y_i\in\mathcal{Y}_i} \mu_i(y_i) \tag{3.30}$$

$$\mu_i(y_i) = \sum_{F'\in M(i)} r_{F'\to Y_i}(y_i) \tag{3.31}$$

3.4.3 Optimal labelling prediction

Inference is a message passing algorithm which is conducted iteratively. Initially, all variable-to-factor messages are set to 0 and because of that factor-to-variable messages can be computed. In the first iteration, those messages are calculated basing on the energy of the factors only. Then, variable-to-factor messages can be computed. This process is exactly the same for factor graphs composed of two types of nodes, which is the case in Conditional Random Fields. Figure 3.3 depicts both kinds of messages for a factor graph constructed on an image that is to be segmented. After all messages are transferred, maximal beliefs for each factor are



(a) *factor-to-variable-messages* ***(b)*** *variable-to-factor messages*

Figure 3.3: *Message passing algorithm on a factor graph created for an image of size* 3×3 *pixels.*

computed and a winning label is chosen, and the whole process is repeated in the next iteration. Inference is said to be completed when the state of convergence is achieved, which means when a change of maximal beliefs is smaller than a chosen convergence tolerance coefficient.

Presented inference algorithms allow predicting the optimal configuration of labels for unclassified test samples given the observed data and an established model in a form of a factor graph. The model can be defined by expertise, however, usually it is trained based on known train samples by a machine learning process.

3.5 Parameter training

Parameter estimation, also called parameter training, is a process that adjusts weights of the model so that they reflect real dependencies between input and outputs. In Conditional Random Fields properly estimated weights are necessary to compute factor energy during an inference process. In order to train parameters of the model supervised learning is used, which is a machine learning task that uses a ground truth to optimise an objective function so that it will be able to correctly map input into outputs. For the task of semantic image segmentation, ground truth is a set training samples which is composed of properly labelled images. Having them it is possible to construct such a factor graph in which not only features for observed variable nodes X are known, but also labels for hidden nodes Y. Similarly to the task of inference, there are two main ways of parameter learning, first involving computation of marginal distributions named Maximum Likelihood Learning and the second requiring only MAP labelling [20]. The latter method known as Max-Margin Learning has recently gained recognition as an alternative towards Maximum Likelihood Learning, however, it is based on Support Vector Machines and not on Conditional Random Fields, which are the key topic of this dissertation and therefore will not be described in this dissertation.

Maximum Likelihood Learning is a standard and the most popular approach to parameter learning in Conditional Random Fields. In order to perform any training process to find optimal parameters of the model, an objective function is required. This is a function that evaluates the ability of the model to properly map inputs into outputs given a chosen set of parameters. In Maximum Likelihood Learning the objective function is called likelihood and it is used to maximise conditional probability $p(y|x,w)$ based on a set of training objects denoted as N with known inputs x and outputs y, and an unknown weight vector w. Usually, the concept of log-likelihood is used instead of likelihood, as computations in logarithmic scale tend to be less complex. Furthermore, knowing that the state with the lowest energy of a factor graph is, in fact, the most probable state, the task of maximising probability is equivalent to minimising the energy of the system. Hence, the objective function for Maximum Likelihood Learning is expressed as a negative log-likelihood denoted as $\mathcal{L}$. Hence, basing on the definition of conditional probability $p(y|x,w)$ presented in equation 3.10 that can be transformed into logarithmic scale, the objective function $\mathcal{L}$ needed for the process of supervised learning can be expressed as in equation 3.32, where x^n and y^n are inputs and outputs of the n^{th} training sample.

$$\mathcal{L}(w) = \lambda \|w\|^2 + \sum_{n=1}^{N} \Big(E(x^n, y^n, w) + \log Z(x^n, w) \Big) \tag{3.32}$$

The energy of a single sample can be expressed as a linear relation between a current weight vector w and a feature function φ that is dependent on inputs and outputs of the factor graph constructed for this sample, as presented in formula 3.33. Then, the partition function Z can be expressed as in equation 3.34.

$$E(x^n, y^n, w) = \langle \varphi(x^n, y^n), w \rangle \tag{3.33}$$

$$Z = \sum_{y \in \mathcal{Y}} \exp\Big(- \langle \varphi(x^n, y^n), w \rangle \Big) \tag{3.34}$$

Apart from an energy term calculated for each of N training samples and their normalising constants Z, in the objective function there is also a regularisation term λ. Regularisation is one of the methods used to prevent overfitting of the model. Its aim is to penalise large values of weights so that less complex models are learned. As λ is a hyperparameter of an objective function, it should be additionally chosen using for example cross-validation. Then, the process of parameter learning is aimed to find such a weight vector w^* for which the objective function will be minimal, as in equation 3.35.

$$w^* = \arg\min_w \left(\lambda \|w\|^2 + \sum_{n=1}^{N} E(x^n, y^n, w) + \sum_{n=1}^{N} \log Z(x^n, w) \right) \tag{3.35}$$

Finding a minimum of a function is a typical task in machine learning and can be solved with various methods. The fundamental and most straightforward method of solving optimisation problems is named Gradient Descent [26]. It is an algorithm that updates parameters of an objective function in such a way to minimise an error between expected outputs of training examples and the actual output of the trained function. It is done by iteratively moving into a direction specified by a negative of a gradient of the objective. The algorithm starts by assigning arbitrary values to the parameters of the function and changing them slightly in each iteration, according to the calculated gradient. At each iteration $t+1$ parameters are updated by subtracting part of a value of the gradient. The size of this part is defined by a hyperparameter α named step size, which can be either constant or changeable throughout a process of learning. Weight update is done according to the formula 3.36.

$$w_{(t+1)} = w(t) - \alpha_t \nabla \mathcal{L}(w_t) \tag{3.36}$$

The process of weight updating continues until the error is small enough to state that a local minimum, or global in case of convex functions, was reached. To apply

the Gradient Descent method in an optimisation problem, an objective function has to be differentiable. Fortunately, a regularised conditional probability specified that was presented in formula 3.32, is a smooth and convex function, hence Gradient Descent method can be used to find a global minimum of it.

During the training process, in each iteration value of gradient $\nabla\mathcal{L}(w)$ needs to be computed in order to update weights of the model. This requires summation over all training samples, which for large training sets is highly time-consuming. Furthermore, to obtain the part of the gradient that is associated with a partition function also a summation over all possible configurations of labels for every pixel is needed, which even for small images with only a few classes requires a lot of calculations. Formula for gradient calculation is presented in 3.37.

$$\nabla_w\mathcal{L}(w) = 2\lambda w + \sum_{n=1}^{N}\left(\varphi(x^n,y^n) - \log\sum_{y_z\in\mathcal{Y}}\exp\left(\varphi(x^n,y_z^n)\right)\right) \tag{3.37}$$

Fortunately, it is not necessary to directly compute the gradient, as there are methods with which it is possible to approximate it. Similarly as in the case of the inference problem, a Loopy Belief Propagation can be used to estimate the marginal probability and therefore the gradient needed for parameter update. However, this would mean that in each iteration the Loopy Belief Propagation has to be performed, which require a lot of time and resources. Instead, the second type of algorithms that have already been introduced in *section 3.4: Inference in factor graphs*, namely Monte Carlo methods, can be applied. One of such methods is a stochastic variant of the Gradient Descent algorithm which is used when the number of training examples is too high to compute the gradient efficiently [27]. The key idea of Stochastic Gradient Descent is to update the parameters of the objective function not after processing the whole training set but after a random batch of training examples, or only

after just one example. In this way, the gradient is not computed exactly but only estimated. Tough this estimation is usually very rough and more iterations are required to converge, the simplification introduced by this method largely reduce the time of each iteration making the whole algorithm much faster. However, even if the sum over all training examples is not required, still in order to obtain the partition function the summation over all configurations of labels is needed, which is computationally infeasible. Fortunately, similarly as in the case of the gradient, the partition function does not need to be computed directly. Instead, another estimation is proposed, which is based on an assumption that large populations usually have a lot of redundancy as they tend to contain very similar objects. Hence, instead of taking into account the whole population it is enough to propose such a set of samples which will reflect the probability distribution of the population. This concept is based on the notion of Monte Carlo integration [28], which states that for sufficiently large number of independent samples S summation of outputs for each sample is roughly equal to the integral of the function, as in equation 3.38, with $p(x)$ denoting distribution of chosen samples.

$$\int_a^b f(x)p(x)dx \cong \frac{1}{S}\sum_{s=1}^{S} f(x_s) \tag{3.38}$$

The main issue of this idea is that it is not known what is the required number of samples for the distribution to reflect the whole population. However, there are techniques named Markov Chain Monte Carlo methods to generate such samples that will fulfil the Monte Carlo integration principle. One of the most popular examples of these methods is named Gibbs Sampling. It is an algorithm that is used to create mutually dependent samples from the probability distribution of an underlying model. It works by constructing a Markov Chain based on a conditional distribution of random variables. A Markov Chain is a model representing a sequence of states,

through which it is possible to move freely with a probability depending only on one, previous state. Gibbs Sampling is an iterative algorithm which initially creates a random Markov Chain representing a configuration of labels for a given object. Then, in each iteration, every variable is replaced by the most probable state, given the conditional probability on all the other variables. Hence, for every possible label of a given variable, an energy of the whole configuration is calculated and a label with the lowest energy, meaning the highest probability of occurrence, is chosen to be involved in the final sequence. After every variable is processed, the resulting Markov Chain forms one sample, which is then used to initialise variables in a sequence for the next sample. This forms a set of Gibbs samples created in such a way that they can be used to estimate the part of the gradient associated with the partition function Z. Thus, the formula for gradient estimation during a Stochastic Gradient Descent after the process of Gibbs Sampling may be expressed as in 3.39.

$$\nabla_w \mathcal{L}(w) = 2\lambda w + \varphi(x^n, y^n) - \log\left(\frac{1}{S}\sum_{s=1}^{S}\exp\left(\varphi(x^s, y^s)\right)\right) \tag{3.39}$$

The process of computing sampling distribution is far less demanding than the computation of full conditional distribution, as instead of taking into consideration every possible configuration of labels it is enough to process distribution of only one random variable at a time provided a set of samples that fulfil Monte Carlo integration principle. However, if a chosen set of samples S is large, in every iteration still a lot of calculations are needed to be performed. Fortunately, in this step yet another simplification can be introduced. For a set of samples containing only one element the number of computations is largely reduced. As an effect, again an estimate of a gradient will be very rough, however, as each Gibbs sample is created based on the previous sample with every iteration it is more close to the approximation of the true distribution. Hence, given a large number of iterations, the algorithm will eventually reach a direction which is an approximation of the true gradient. This

method of gradient estimation based on only one sample is known as contrastive divergence training [22]. Thus, the final formula to compute the gradient needed for parameter update is presented in equation 3.40.

$$\nabla_w \mathcal{L}(w) = 2\lambda w + \varphi(x^n, y^n) - \varphi(x^s, y^s) \tag{3.40}$$

Typically Gradient Descent Methods tend to choose the shortest path towards the minimum of the objective function. The key idea behind all the presented simplifications is not to find the shortest path, but instead to make more steps that are less accurate but easier to compute. Both approaches are aimed to converge near the global minimum, though the second one is faster are more applicable for problems with a lot of training data and large output space. Having defined the objective function and a method to estimate the gradient it is possible to efficiently perform a supervised parameter learning and to define weights of the system that are required for the inference process.

3.6 Energy formulation

A basis of both inference and parameter training is a proper formulation of an energy function, which reflects a probability of occurrence of a given configuration of variables in a system. In semantic image segmentation, without correctly defined energy function it will not be possible to differentiate objects of different classes. As it has already been presented, energy is composed of two terms: unary potential and pairwise potential.

$$E(y,x) = \sum_{i \in V} E_1(y_i, x_i) + \sum_{i,j \in V} E_2(y_i, y_j)) \tag{3.41}$$

Proper formulation of both of those terms is needed for Conditional Random Fields successfully perform their assigned tasks.

3.6.1 Unary potential

In semantic image segmentation, the unary component of the energy function predicts a label of a given pixel or region, based on some observed features of this part of an image. Though it can be modelled with a number of different methods it is always responsible for assigning the same label to input nodes that are characterised by similar features. A feature can be described as a numeric representation of row data that is fetched from an image. The process of extracting and selecting those representations is a challenging task as without properly chosen features that provide all the required information to the model, it would not be possible to perform the assigned task. What is more, feature engineering usually accounts for the vast majority of the time that is needed to perform a machine learning task [29]. Proper formulation of a feature vector is a crucial step in machine learning as it reflects not only the level of the model complexity but also its performance. However, the relevance of available features is strictly bound to data and to the chosen model and due to to the large diversity in both data and models it is extremely difficult to generalise the process of feature engineering. Depending on the task some models can perform well given a large number of features while others require less, but more informative features. Though there exist some automatic or semiautomatic tools for feature engineering [30, 31, 32], that aim to make feature extraction less problem specific, still the vast majority of machine learning tasks use the traditional approach of manual feature selection [33].

Features used for any task within the area of image processing can be either low- or high-level [34]. Low-level features are such features that can be automatically extracted from an image such as pixel intensity, gradients, colours, edges or textures. They do not give any information about spatial relations between regions of an image. On the other hand, high-level features correlate data obtained from low-level features with a content of an image to provide contextual information about shapes and objects, thus having a semantic meaning. Feature engineering is a time-consuming and demanding task of its own and it is not a part of this dissertation. Therefore, to stay within the scope of the thesis, feature extraction is limited only to features that are based solely on colour-related data.

The unary potential of a given factor between an input node x_i and an output node y_i is generally specified by a linear relation between feature function $\varphi(x)$ and learned parameters $w(y_i)$ [22] as in equation 3.42.

$$E_1(y_i, x_i, w) = \langle w_1(y_i), \varphi(x_i) \rangle \tag{3.42}$$

In the most basic form, feature function outputs a feature vector ϕ that is composed of colour values of each individual pixel. A colour value can be expressed in a numerical form with the use of a chosen colour space, which is a mathematical model representing a colour value as a tuple of typically three or four numbers. The most popular colour space being based on the physiology of a human eye is an RGB model, in which an arbitrary colour value is denoted as a combination of values of three primary colours, which are red, green, and blue. A different colour space, named CIELAB, was created in other to mimic a human perception of colours. The model is constructed on three axes, first being axis L* representing lightness with values between 0 and 100. Axes a* and b* encode green-red and blue-yellow components respectively. There are many other colour spaces such as

CMYK, HSV, HSL or YUV, each of them being more suitable for certain tasks [35]. Nevertheless, the choice of colour space in which colour features will be encoded in the prediction model can influence its performance. For simplicity, in this chapter RGB colour space will be assumed. Hence, a feature vector φ for a given pixel is composed of three features, each representing a different component of the chosen colour space as in equation 3.43.

$$\varphi(x_i) = \begin{bmatrix} \phi_R \\ \phi_G \\ \phi_B \end{bmatrix} \tag{3.43}$$

When it comes to parameters of the unary potential they are expressed in terms of weights that are learned in the training process. With every label y there is a separate weight vector w associated, as in as in equation 3.44, and to compute unary potential a concatenation of weight vectors for each label is needed.

$$w_1(y_i) = \begin{bmatrix} w_R \\ w_G \\ w_B \end{bmatrix} \tag{3.44}$$

This distinction between labels is required as each object class can be best characterised by different features. For example, given an object with label *grass* a green component of the feature vector should be prioritised, while for label *sky* it should be a blue component. As the task of image segmentation, meaning finding an optimal prediction y^* for each pixel in an image, is equivalent to the problem of energy minimisation, weights which are associated with prioritised features should have relatively small values. Hence, with the same example of two labels *grass* and *sky*

proper configuration of weights would be as in equation 3.45.

$$w_{grass} = \begin{bmatrix} 1.0 \\ 0.0 \\ 1.0 \end{bmatrix} \qquad w_{sky} = \begin{bmatrix} 1.0 \\ 1.0 \\ 0.0 \end{bmatrix} \tag{3.45}$$

Then, for example for fully blue pixels, with feature vector $\phi = [0.0, 0.0, 1.0]$, the unary component of energy will be equal to 0.0 for label *sky* and 1.0 for label *grass*. Thus, label *sky* will be chosen for such pixels, as the energy for this label is smaller than for other labels. Hence, to obtain the unary component of the energy function for a given pixel x_i with a given label y_i from a set of labels L, a corresponding weight vector needs to be multiplied by the feature vector as in equation 3.46.

$$E_1(y_i, x_i, w) = \begin{cases} \left\langle w_{1,0}, \varphi(x_i) \right\rangle, & \text{if } y_i = 0 \\ \left\langle w_{1,1}, \varphi(x_i) \right\rangle, & \text{if } y_i = 1 \\ \ldots & \ldots \\ \left\langle w_{1,L}, \varphi(x_i) \right\rangle, & \text{if } y_i = L \end{cases} \tag{3.46}$$

Presented formulation of the unary potential is the most trivial one, however, it can also be applied also for more complex models and the only difference is in a way the feature function is defined. The goal of energy analysis is to provide a measure of how well pixels in an image are labelled and any feature function that outputs a number, or a vector of numbers can be applied for the unary potential. A feature function does not need to be based on low-level features extracted from an image. Instead, a higher level of abstraction can be used to make features of a given pixel dependent on other classifiers. One example would be a Decision Tree Field approach in which factors that form unary potential are dependent on Decision Trees [36]. Another and a very popular and widely described approach to semantic

image segmentation is to model a feature vector for unary potential by utilisation of outputs of a Convolutional Neural Network [19, 37, 38]. However, incorporating an additional classifier into the unary potential of Conditional Random Fields is not the only possibility to improve its performance. Another way is to treat the feature function as a probability density function representing a relation between labels and features. Hence, instead of operating on a feature vector that directly represents some properties of a given pixel, or on outputs that were assigned by an auxiliary classifier, such feature function is introduced that will be able to provide a conditional probability of assigning a label to the current pixel given its features. This probability is expressed in terms of log-likelihood as in equation 3.47.

$$\varphi(x_i, y_i) = -\log p(y_i|x, i) \tag{3.47}$$

A given pixel i is represented as a set of features, hence, computation of unary potential is equivalent to finding probability distributions of every label y_i conditioned on a set of features f_i, which can be parametrised with additional parameter θ.

$$p(y_i|f_i, \theta) \tag{3.48}$$

Conditional probabilities of $p(y_i|f_i)$ are not known, however, according to Bayes theorem, they can be found if there is a prior knowledge on a reverse probabilities $p(f_i|y_i)$ as in equation 3.49.

$$p(y_i|f_i) = \frac{p(f_i|y_i) \cdot p(y_i)}{p(f_i)} \tag{3.49}$$

As probabilities of all possible labellings for a given pixel needs to sum to 1, a term $p(f_i)$ acts as a normalising constant in this equation. Therefore, it can be obtained by equation 3.50.

$$p(f_i) = \sum_{y_i \in \mathcal{Y}_i} (p(f_i|y_i) \cdot p(y_i)) \tag{3.50}$$

Hence, to calculate a unary potential for a pixel in a given image it is necessary to know the probability of occurrence of the chosen label and a probability of pixel features conditioned on this label. Both of them can be found from a probability distribution that needs to be modelled in an additional training process. With a set of correctly labelled training examples a label probability $p(y_i)$ is straightforward to calculate as it enough to find the ratio between pixels with a given label assigned and all pixels in a training set. Computation of $p(f_i|y_i)$ can be fairly complex, as a dimension of this distribution is dependent on the number of chosen features. However, assuming that chosen features are independent of each other, a problem of finding N-dimensional probability distribution can be transformed into N problems of finding one-dimensional distributions as in equation 3.51.

$$p(f_i|y_i) = \prod_{k=1}^{K} p(f_{i,k}|y_i) \tag{3.51}$$

Any number and kind of features can be chosen as long as they can be expressed in a numeric way. The method of assessing a conditional probability between a feature and a label is dependent on a feature type. In general, features can be either discrete or continuous. The first type also known as categorical data, represent such features that can have only one value from a set of predefined values. Then, to get a probability $p(f_{i,k}|y_i)$ for a pixel from a given test image, it is enough to have probabilities of every possible value of the given feature computed beforehand. Once computed during the training process, such probabilities are applicable for any new test image, as for discrete features the whole input domain is covered. For instance, if an image will be first subjected to the process of colour quantisation in order to limit the number of available colours in an image for example by using a 32 colour palette, then a feature representing a colour of a pixel would be a discrete feature taking one out of 32 possible values. Hence, in order to get the probability

distribution of such feature, it is enough to count how many times a given colour value appeared in the training set.

On the other hand, for continuous features the input domain is infinite, and therefore it is infeasible to compute a probability of occurrence of each value beforehand. A probability of continuous variables is represented by a probability density function. This function describes a relative likelihood that the variable of interest will have a given value. Then, the probability that the variable will fall within a certain range of values is equal to the integral of the variable density over this range [39]. However, usually no information is available on the type of features distribution, nor on parameters that define it. Fortunately, given a training set of data samples, it is possible to estimate the probability density function of a continuous variable even without knowing its type. It can be achieved by kernel density estimation, also known as Parzen–Rosenblatt window method. The idea of this method is based on superposing kernel functions that are placed over observations and later scaling them in order to create one smooth function that is an approximation of all data points. Hence, to obtain probability $p(f_{i,k}|y_i)$ of a chosen feature k of pixel i there is a need to estimate probability distribution $p(f_{i,k})$ based on all N pixels labelled with y_i according to formula 3.52.

$$p(f_{i,k}) = \frac{1}{Nh_k} \sum_{n=1}^{N} \mathcal{K}(\frac{f_{i,k} - f_{i,k}^n}{h_k}) \tag{3.52}$$

Kernel, denoted as $\mathcal{K}$, is a probability density function with known distribution that is symmetric around 0. The choice of the kernel determines the shape of those individual functions around data points which will be averaged. Gaussian function, which models normal distribution, is an example of a function that can be used as a

kernel. It is presented in formula 3.53.

$$\mathcal{K}(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2} \tag{3.53}$$

Other examples of kernel functions include uniform function, also known as rectangular window as in equation 3.54

$$\mathcal{K}(u) = \begin{cases} \frac{1}{2} & , \quad u \in [-1,1] \\ 0 & , \quad otherwise \end{cases} \tag{3.54}$$

triangular function, presented in equation 3.55,

$$\mathcal{K}(u) = \begin{cases} 1 - \|u\| & , \quad u \in [-1,1] \\ 0 & , \quad otherwise \end{cases} \tag{3.55}$$

or parabolic function, named Epanechnikov kernel, which is shown in 3.56.

$$\mathcal{K}(u) = \frac{3}{4}(1-u^2) \tag{3.56}$$

Bandwidth, denoted as h, is a hyperparameter of the kernel density estimation responsible for the level of data smoothing as it controls the width of individual superposed functions. Consequently, it reflects the range in which data points have a larger effect on an individual observation. The optimal choice of bandwidth is required for the good performance of the method. For large values of h the function will be underfitted, meaning it will be too simple to model known observations, nor it will be able to predict probabilities of new data points. On the other hand, for values of h that are too small, the model will exhibit overfitting. It will almost perfectly learn all training data, including noise, however, it will not be able to generalise for new data points. As any hyperparameter, bandwidth value should be subjected to the optimisation process before the estimation takes place. A commonly used method to find optimal values of hyperparameters is called cross-validation [40]. In

this method for every value of a hyperparameter that is under consideration, a set of observed data is divided into two partitions, a training set and a validation set. Then, the model is trained only with the first set and its performance is evaluated on the validation set. Typically, training and validation steps are performed multiple times using different partitions, so that most of the observed data will be validated against. Then, the results of each individual evaluation are aggregated to provide a measure of the model prediction performance. Optimal hyperparameter is the one for which this performance is the highest.

After finding the optimal value of the bandwidth h and choosing the kernel function $\mathcal{K}$ kernel density estimation can be used to predict probability $p(f_{i,k})$ of a feature k from pixel i from an unknown test image. This requires iterating through all the training data in order to compute the difference between the value of this feature and the observed value that is needed for a kernel function. For large training sets, this becomes computationally complex. Moreover, this needs to be repeated for every feature of all pixels in a test image, which makes it even more time-consuming. Hence, for large training sets, other methods of probability estimation should be used.

Parzen–Rosenblatt window method provides an accurate way to estimate the probability of an unknown sample based on the aggregated probability density function. However, this function is freshly generated each time estimation is needed. In order to limit the number of required calculations, there is a need to propose a method that will model the underlying probability density only once, during the training phase, and not every time probability of a new observation needs to be assessed. Unfortunately, without knowing parameters describing the generated probability distribution function there is no way to regenerate it without iterating through all

data points. However, the problem can be simplified if the training data would be firstly binned. Binning is the process aimed to discretise continuous data by grouping them into intervals of equal size that are called bins. The simplest way to model binned data is to use histograms, which give information on the number of data samples that fell into a particular bin. Then, to obtain the probability of a feature k of an unknown pixel i, it is enough to check to which bin its value falls and to calculate the ratio between the number of observations in a bin denoted as $b_{f_{i,k}}$ and all data points N as in equation 3.57.

$$p(f_{i,k}) = \frac{b_{f_{i,k}}}{N} \tag{3.57}$$

The number of observations in each bin can be computed only once, in a training phase, and it gives enough information to estimate the probability of a feature of any unknown pixel. The accuracy of this estimation is reflected by the number of bins used to create a histogram. If too few bins were used in a discretisation process, important data about observations, especially at the boundaries of bins, would be lost. On the other hand, the more bins are used, the more computationally complex the problem becomes. The number of bins is a hyperparameter of a histogram model, and as in the case of bandwidth in the Parzen–Rosenblatt window method, it should be found in an optimisation process.

After choosing an appropriate method of probability density estimation, it is possible to obtain a probability distribution of each feature $p(f_{i,k}|y_i)$ and consequently the whole distribution for a given image $p(f_i|y_i)$. This distribution is needed for Bayes' theorem, which was presented in formula 3.49. Then, for every pixel from an unknown test image, it will be possible to get a probability of each label $p(y_i|f_i)$, and therefore calculate the unary potential $E_1(y_i, x_i, w)$.

3.6.2 Pairwise potential

When it comes to the second component of the energy function - a pairwise potential, it is calculated for factors that connect outputs of neighbouring nodes and is aimed to handle noises in an image as it is responsible for smoothness of the predicted labels. It penalises situations in which nodes that are in close proximity to each other have different labels assigned. Thus, in the semantic image segmentation, a sample pixel that is surrounded by pixels with some label will be more likely to have the same label, even if the unary term is more prone to assign a different one because of the pixels noised features. In the simplest form, a feature function φ returns a two-dimensional vector with only two possible values $[1,0]$ for pairs with the same label, and $[0,1]$ for differently labelled pairs, as in equation 3.58.

$$\varphi_{y_i=y_j} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \varphi_{y_i \neq y_j} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3.58}$$

Then, a weight vector is a two-dimensional vector composed of weight $w_{2,0}$ which describes a situation in which neighbouring pixels have the same label, and weight $w_{2,1}$ for a pair differing in assigned labels.

$$w_2(y_i, y_j) = \begin{bmatrix} w_{2,0} \\ w_{2,1} \end{bmatrix} \tag{3.59}$$

Similarly to the unary component of the energy function, the pairwise component can be described as a linear relation between weight and feature vectors. This is, in fact, equivalent to expressing it in terms of two weights which directly reflect pairwise energy value. If two pixels that form neighbouring nodes in a factor graph have the same label assigned, then the energy of the factor between them will be equal to weight $w_{2,0}$, if they have a different label assigned, then it will be $w_{2,1}$ as

in equation 3.60. Again, as the state of smaller energy is the one that is going to be chosen for a given factor, if the situation of two neighbouring pixels having the same label is to be promoted, then $w_{2,0}$ should have a smaller value than $w_{2,1}$.

$$E_2(y_i, y_j) = \begin{Bmatrix} w_{2,0} & , y_i = y_j \\ w_{2,1} & , y_i \neq y_j \end{Bmatrix} \tag{3.60}$$

The pairwise term of energy function provides higher-order information about relations neighbouring pixels, however, the notion of neighbours is different depending on the model used. In general, a neighbour is any pixel that is adjacent to the given pixel, however, neighbourhoods of larger size can also be used. In such a situation factor graphs are highly beneficial as they allow to model relations between even larger groups of pixels in a clear and straightforward way. What is more, pairwise potential can be utilised to take into account even more complex relations between pixels by introducing a measure of how different their classes are from each other. Then, the feature vector, instead of having only two possible values 0 or 1, can take any value within the range of 0 and 1 which reflects a measure of similarity between pixel classes. Then, the formula for a feature vector used to calculate a pairwise potential can be expressed as in the formula 3.61. Assuming that the difference between labels $\left|(y_i - y_j)\right|$ is either 0 if labels are the same or 1 if labels are different, this equation is equivalent to formula 3.58.

$$\varphi(y_i, y_j) = \begin{bmatrix} 1 - \left|(y_i - y_j)\right| \\ \left|(y_i - y_j)\right| \end{bmatrix} \tag{3.61}$$

Penalising a situation in which adjacent pixels have different labels is well suited for regions in which there are some noised pixels inside. However, not every time when two neighbouring pixels have a different label assigned necessarily means that there is some noise in one of those pixels. On object boundaries assignment

of different labels should be promoted and not penalised. One of the most popular methods used to represent pairwise potential in Markov Random Fields is named Potts model [41]. This model represents spatial relations between neighbouring pixels taking into account their similarity. Neighbouring pixels affect each other just like in the previously described definition of a pairwise potential, however, in this model, only when pixels are alike the assignment of the same label is promoted. Potts model is applicable to problems in which labelling is locally constant and regions are separated by clear boundaries. Analogously to unary potential, a similarity between pixels is defined by their features. A pairwise feature function $\varphi(y_i, y_j)$ of a factor between output nodes i and j that is constructed with the use of Potts model is defined as in formula 3.62. Depending on the number of pairwise features chosen the size of this vector will be different. For one feature it will have two elements only - a term defining pixel similarity, and a unit element that allows incorporation of bias to the system.

$$\varphi(y_i, x_i, y_j, x_j) = \begin{bmatrix} \exp\left(-\beta_1 \left\|\phi_1(x_i) - \phi_1(x_j)\right\|^2\right) \\ \exp\left(-\beta_2 \left\|\phi_2(x_i) - \phi_2(x_j)\right\|^2\right) \\ \dots \\ \exp\left(-\beta_m \left\|\phi_m(x_i) - \phi_m(x_j)\right\|^2\right) \\ 1 \end{bmatrix} \tag{3.62}$$

Potts model requires a measure of similarity between pixels in order to check if there is an edge between them. This measure is represented as the difference between feature vectors of those pixels $\left\|\phi(x_i) - \phi(x_j)\right\|$ and a way of its calculation is dependent on a type of the chosen features. For example, a contrast sensitive Potts model that measures the difference between colour features of neighbouring pixels can be used [42], as differences in colour intensity are the simplest indicator of edges in an image. Then $\phi(x_i)$ and $\phi(x_j)$ would denote a three-dimensional vec-

tors representing colours in a chosen colour scheme, of a given pixel pair i and j. Depending on the problem a similarity of pixels can be calculated differently. If an image was pre-classified so that each region is represented by one discrete state, it is enough to propose a binary method of pixel similarity measure. Then the difference $\phi(x_i)$ and $\phi(x_j)$ would take value 0 if pixels are of the same class, and value 1 otherwise. For images that were not pre-classified, this measure can take any numeric value. For example, for colour features, it would the distance between colours in the chosen colour scheme. The CIELAB scheme would be a beneficial choice, as its way of modelling colour differences is the most similar to human colour vision. The described way of representing pairwise feature vectors requires also a parameter β. This is an image dependent coefficient that reflects the spatial correlation between adjacent image pixels [43]. For small values of β classification will remain noisy, while for too large β the result will be over-smoothed. Proper selection of this parameter is crucial for denoising abilities of the pairwise term, however, it is difficult to fix this value beforehand, as spatial organisation of images tend to differ. Therefore, a value of β should be adjusted separately for each image. One of the ways to make β image specific is to introduce a dependency between this parameter and an average value of a given feature difference $\left\langle \left\| \phi(x_i) - \phi(x_j) \right\| \right\rangle$ in the image, as in formula 3.63.

$$\beta(x_i, x_j) = 2 \cdot \left\langle \left\| \phi(x_i) - \phi(x_j) \right\| \right\rangle^{-1} \tag{3.63}$$

Knowing how to compute feature difference for each of the chosen pairwise features, and how to obtain the value of the parameter β it is possible to construct a feature vector that is needed to compute the pairwise potential. Together with a definition of the unary potential, the whole energy function was described and it can be used in the process of parameter learning, and inference.

Chapter 4

Semantic image segmentation system

The topic of this thesis concerned the task of semantic image segmentation with the use of Conditional Random Fields, which was described in details from the theoretical point of view in *chapter 3: Structured Prediction*. In order to state whether the explained algorithms are capable of performing the main task of this thesis, there was a need to implement them. Therefore, an image segmentation system was developed, which incorporates the previously described methods. In this chapter, basic information about the capabilities of the system, its components and the technological stack will be presented.

4.1 System goals

The primary goal of the developed system is to perform semantic segmentation on images, however, the way in which this final task is achieved is equally important to the results that are to be presented. The created system may be divided into two interconnected parts aimed to perform the segmentation in a slightly different manner. The components of both parts of the system are exactly the same and the only difference, and yet a substantial one, lies in a way a feature function needed for energy calculation is defined. The first part, which will be described in details in *chapter 5: Colour-based semantic image segmentation* is based on the simplest way of expressing features, which is by using a feature function that directly outputs a feature vector composed of low-level features image pixels. On the other hand, the feature function used in the second part of the system involves a higher level of abstraction as instead of directly operating on features, a probability estimator is introduced that outputs a probability of a given label conditioned on a set of pixel features. The details of this part of the system will be provided in *chapter 6: Shape-based semantic image segmentation*.

4.2 System components

The developed system incorporates all the standard components needed for supervised machine learning, which were described in details in the theoretical section. First of all, to perform any task concerning image processing, a dataset of images is needed. In the thesis, all required images were artificially generated. Thus, the first component of the system is an image generator that is aimed to create sets of

images that will be used as inputs for other system components. Having the dataset generated, the next step is preprocessing, which is done in a separate component. Its main goal is to perform a division of a single image into a set of superpixels, which largely limits the complexity of other processes. Having the dataset prepared there is a need to provide a method of transforming data from superpixels into meaningful features that are understandable by the algorithms used in the system. Hence, the next component performs feature selection and outputs a set of chosen feature that are then extracted from every superpixel. As it has been already presented in the theoretical chapter, Conditional Random Fields are best modelled with the use of factor graphs, thus the next part of the system performs a factorisation process, which models the underlying problem into a factor graph that is composed of input nodes, one per each superpixel, and output nodes representing superpixel labels. These are the main steps that are needed to provide input data for two main processes in the created semantic segmentation system which are parameter training and inference. However, for a part of the system in which a feature function is expressed in terms of the conditional probability of label occurrence given a set of features, also another component is required. This component is aimed to provide an estimation of the probability distribution of features and labels that is needed to obtain the requested probability. The next piece of the developed system provides a supervised learning algorithm with the use of Stochastic Gradient Descent. The aim of this component is to adjust the weights of the system so that the created model will generalize well for unknown data. Having a properly parametrised model it is possible to perform semantic image segmentation on an unknown image. This is done in the next component that conducts a process of maximum a posteriori inference. As it has been explained in the theoretical section, for Conditional Random Fields the most suitable inference algorithm is Loopy Belief propagation,

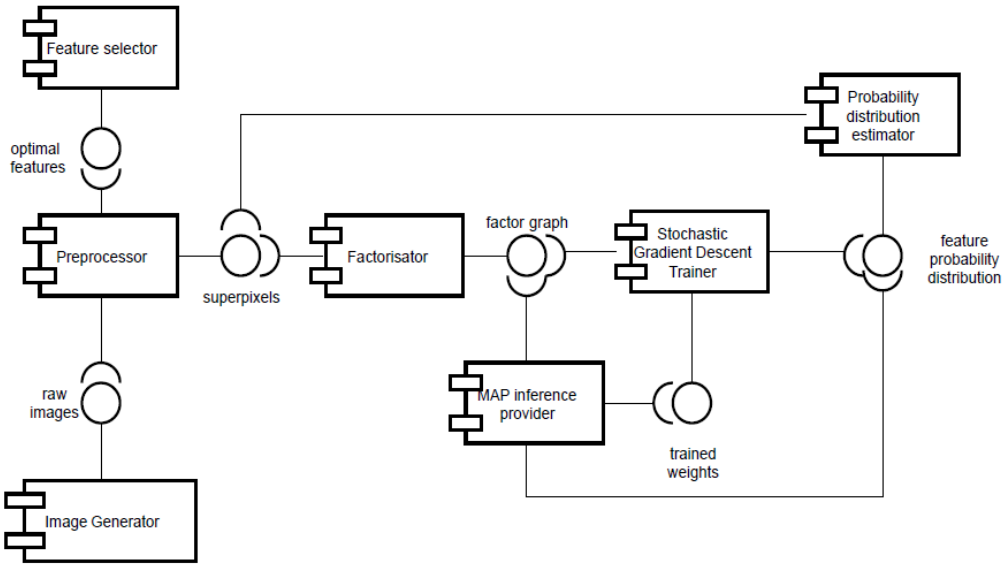


Figure 4.1: *Component diagram of the created semantic image segmentation system.*

which finds the most probable configuration of superpixel labels for a given image by means of the Min-Sum algorithm. Such configuration is exactly the result of semantic image segmentation. The way in which all components of the system are connected is presented in Figure 4.1.

4.3 Technological stack

From all the presented components only the preprocessing stage was implemented with the use of an external library, which was delivered by the supervisor of this thesis. Other components were written from scratch only with the use of some basic Java libraries that facilitate coding and testing processes such as Apache Commons, log4j or JUnit. As the preprocessing component was incorporated to the system

after some progress in the implementation of other components had already been made, during the initial planning phase there were no restrictions on the programming tools that are to be used. That is why the technological stack was mainly a matter of personal choice. A language chosen for implementation of this system was Java in version 1.8. It is a well-established language that is known from its platform independence. This is a huge advantage in situations in which the same program is to be developed on two different operating systems, or if the development and testing phases happen on different machines. Furthermore, according to TIOBE Programming Community index [44], which provides data on language popularity, Java is currently the most popular language and it has been in the top three for nearly 20 years. It also has a large availability of free tools, including Eclipse IDE, which was used for development of the system presented in this dissertation.

When it comes to the hardware on which all the phases of the system development took place, it was a personal computer with a processor Intel Core i5-5250U, 8GB of RAM and Windows 10 as an operating system.

Chapter 5

Colour-based semantic image segmentation

As it has already been presented, the created system is composed of two parts and this chapter will be devoted to the first one. It will involve an explanation of how the already described theoretical background can be adapted to perform the task specified in this dissertation. Implementation of methods chosen to address this task was based on a book titled "Structured Learning and Prediction in Computer Vision" by Sebastian Nowozin and Christoph H. Lampert [22], which provided a lot of essential information for the problem. The part of the created system that is described in this chapter was devoted to performing semantic image segmentation on a simple example of segmenting objects in an image based only on colours of those objects, without incorporating any contextual data. This part of the system was aimed to prove that with the use of Conditional Random Fields it is possible to perform semantic segmentation of simple images presenting objects that differ only

by colour. The theory behind all the algorithms that were used to accomplish this task has already been presented in *Chapter 3: Structured Prediction*.

The first section of this chapter will provide a detailed explanation of how the preprocessing stage was performed and why it is beneficial to incorporate it in the created system. The next section will describe the key difference between this chapter and *chapter 6: Shape-based semantic image segmentation*, which is the way in which features are represented. Then, a dataset used for training and testing of the solution will be presented. The next section will provide information on how the results of semantic segmentation will be evaluated. Last, but definitely not least, an overview of the conducted experiments and their results will be provided.

5.1 Preprocessing

The task of semantic image segmentation is aimed to assign one label to each pixel of an image. The system that is described in this dissertation required extensive computations both in training and in inference phases. Consequently, for large images and vast training sets, semantic image segmentation with the use of Conditional Random Fields becomes a time and resource consuming process. A lot of improvement can be achieved by incorporating all the simplifications that were described in chapter 3 Structured Prediction, however, still every computation needs to be performed for each pixel in an image. That is why, in the described system a preprocessing stage was introduced that is aimed to limit the number of required computations. In this stage, each image from training, validation and testing sets, is subjected to the process of superpixel segmentation. The aim of this process is to divide an image into a number of regions named superpixels that are composed of

multiple neighbouring pixels that share similar colour properties. This segmentation is performed in such a way that each pixel in an image is assigned to exactly one superpixel, therefore, superpixels are non-overlapping. Moreover, if the division into superpixels is done correctly, objects in the image should be divided into multiple superpixels, but no superpixel should be split by an object boundary [45].

The division into superpixels was the only part of the system that was accomplished with the use of an external library. This library takes three arguments on input, the first one being an image on which segmentation should be performed, second is an expected number of superpixels, and the third one is a compactness coefficient. The last parameter defines the importance of uniformity in superpixels shape. For a large value of compactness coefficient, resulting superpixels will have smooth boundaries and will be roughly in the same size, while for small values adherence to object boundaries is more promoted than the uniformity in superpixel size. The goal of the preprocessing stage is to limit the number of inputs in a constructed factor graph to as large extent as possible at the same time sustaining information about object boundaries. Hence, proper selection of the expected number of superpixels and the compactness constant is crucial for optimality of the image segmentation system described in this dissertation.

Input parameters for the process of superpixel division are image specific, therefore there is no exact way in which they should be defined. When it comes to an expected number of superpixels the choice is rather simple, as this parameter directly reflects the number of inputs in the image factor graph obtained in the modelling phase. Hence, it should be mainly dependent on the image size and on the extent to which a number of inputs should be reduced. On the other hand, the choice of the compactness coefficient is not that easy, as the relation between this parameter

value and a resulting superpixel shape is not so straightforward. The importance of the proper choice of this parameter can be visualised basing on an example of superpixel segmentation of one image. Figure 5.1 shows a sample image that will be subjected to the process of division into superpixels. The chosen image has the

Figure 5.1: *A sample input image for the task of initial segmentation into superpixels.*
Source: *Public-Domain Test Images for Homeworks and Projects [46].*

size of $512px \times 512px$. For this sample image, the parameter defining the expected number of superpixels was set to be 300. When it comes to compactness coefficient three values were chosen, 500, 50 and 2. First set of images shown in Figure 5.2 presents one sample image divided into superpixels per one compactness coefficient value. A colour of each pixel in those images is the same as the mean colour of a superpixel which contains the given pixel. As visible, for a large value of compactness coefficient being 500 as shown in Figure 5.2a, colours of the test image are blurred and it is hard to notice what this image presents without seeing it before the segmentation process. On the other hand, segmented images with compactness coefficient for values 50 and 2 are quite similar to the original image.

(a) Compactness 500 *(b) Compactness 50* *(c) Compactness 2*

Figure 5.2: *Division of a sample image into superpixels based on different values of compactness coefficient, with the mean colour of each superpixel marked.*

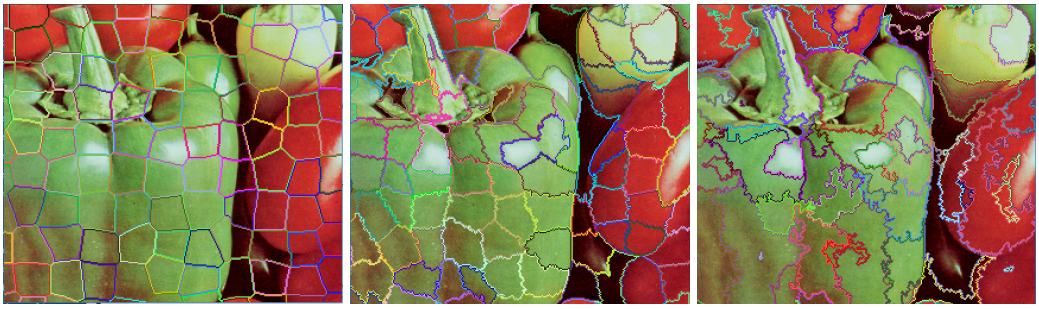
Next set of images shown in Figure 5.3 present the results of the same processes of segmentation, however, this time superpixel borders are marked on the original image. As depicted in Figure 5.3a for large values of compactness coefficient

(a) Compactness 500 *(b) Compactness 50* *(c) Compactness 2*

Figure 5.3: *Division of a sample image into superpixels based on different values of compactness coefficient, with borders of each superpixel marked.*

the resulting superpixels have smooth boundaries, are roughly in the same size and have a square-like shape. A similar division is presented in Figure 5.3b for compactness coefficient of 50. Tough boundaries of the created superpixels are less

smooth, still the dimension of them is pretty similar. Uniform size of created superpixels is more suitable for the case in which an image is to be modelled with a graph, as they produce a regular lattice [47], which is important for the proper definition of superpixel neighbours that is required for example to compute pairwise potentials in Conditional Random Fields. On the other hand, for small values of compactness coefficient, as in Figure 5.3c, the obtained superpixels adhere well to object boundaries, however, they differ largely in shape and size. Also the created lattice is highly irregular, as some objects are composed of only a few superpixels, while in others even a small detail or a change of brightness is marked as a separate superpixel, which would make it difficult to assign this superpixel into a proper class. The difference between the shape of created superpixels for different values of compactness coefficient is best visible on object boundaries. Figure 5.4 depicts a zoomed view of superpixels that were created on the boundary between green and red peppers.



(a) *Compactness 500* ***(b)*** *Compactness 50* ***(c)*** *Compactness 2*

Figure 5.4: *Division of a sample image into superpixels based on different values of compactness coefficient, with boundaries of each superpixel marked.*

It is clear that for compactness coefficient of 500 the created superpixels do not comply with the already presented rule that no superpixel should be split by an object boundary. With such superpixel division, semantic image segmentation would

not be successful, as the information about object boundaries would be lost during the preprocessing stage. Hence, such a large value of this parameter does not give acceptable results. For compactness of 2, as shown in Figure 5.4c, created superpixels have totally different shapes. Furthermore, some superpixels are composed of only a really small number of pixels and they usually present artefacts of an image, which do not carry any important information for semantic segmentation and would only increase the computational complexity of this task. Therefore, too small values of compactness coefficient should also be avoided. Figure 5.4b depicts superpixel segmentation that is the closest to optimal from all the presented divisions. The resulting superpixels properly define object boundaries at the same time creating a regular grid. What is more, small regions with changes of brightness that are irrelevant for semantic segmentation are not separated from surrounding pixels. Therefore, such division into superpixels would be applicable for further processing.

Thanks to the preprocessing stage, the number of inputs in the factor graph constructed upon a given image is largely lowered as instead of hundreds of thousands of pixels only a few hundreds of superpixels are needed. As a result, the complexity of calculations and required processing time are reduced. What is more, by division into superpixels, pieces of an image that are irrelevant for the task of semantic image segmentation are averaged and incorporated into one superpixel together with adjacent regions that carry more important information about the object they represent. This smoothing property is important for example for some slightly noised pixels or regions with the sudden change of brightness in few adjacent pixels.

5.2 Feature function definition

The part of the system that is described in this chapter was aimed to present a basic example of how Conditional Random Fields can be used to perform the task of semantic segmentation on images. Therefore, only simple features were chosen to represent images from the dataset. As it has already been explained in *section 3.6: Energy formulation* there are two main groups of features that represent individual nodes in a factor graph. Features from the first group are bound to input nodes and are used for computing the unary potential, while the others are for pairwise potential computation between output nodes. When it comes to the latter type, for the part of the system that is described in this chapter, a binary pairwise feature was introduced. It is in a form of a two-dimensional vector that can take only two values: $[1,0]$ if the labels of a given pair are the same, or $[0,1]$ if they are different as it has been already presented in formula 3.61.

When it comes to the unary potential, only colour features of a given superpixels were taken into consideration. Every input factor can be described by three continuous numbers, each representing one colour component from a chosen colour space. As it was already explained in the theoretical section, the unary potential is dependent not only on the chosen features but also on the label which is under consideration. As this part of the system performs classification into three distinct classes, there are three weight vectors that are used to calculate an energy of the system, one per each label. Then, the unary potential used in this part of the system

may be computed according to the formula 5.1.

$$E_1(y_i, x_i, w) = \begin{cases} \langle w_{1,0}, \varphi(x_i) \rangle, & \text{if } y_i = 0 \\ \langle w_{1,1}, \varphi(x_i) \rangle, & \text{if } y_i = 1 \\ \langle w_{1,2}, \varphi(x_i) \rangle, & \text{if } y_i = 2 \end{cases} \tag{5.1}$$

However, such representation is not very convenient from an implementation point of view, especially during the parameter training process in which every weight of the system needs to be updated iteratively. Therefore, instead of having three distinct weight vectors, a single, joined weight vector can be proposed. Then, it will contain nine elements, as there are three colour components and three available classes. However, a nine-dimensional weight vector requires the feature vector to contain nine elements as well. Though colour features are independent of the assigned label as they are specific to a given superpixel and unchangeable during the segmentation process, such a feature function can be introduced that returns a different feature vector depending on a label. Thus, this vector would contain three elements representing a superpixel colour, one per each colour component and six other elements with values equal to 0. Then, depending on the label that is under consideration, those three elements describing superpixel colour would occupy a different section of a feature resulting vector. For label 0, they will be placed on indices 0-2, for label 1 on 3-5, and for the last label on positions 6-8. Hence, the formulas for a weight vector and a feature vector that were used in this part of the system can be defined as in equation 5.2. Colour components are denoted as R, G, and B, though different colour space than RGB can also be used. A value of a single element in a feature vector is dependent on the difference between a label of a current vector position, denoted by y_0, y_1, or y_2, and a label of the superpixel that is under consideration. This difference can take a value of 0 if labels are the same,

or 1 if they are not.

$$w_1 = \begin{bmatrix} w_{R,0} \\ w_{G,0} \\ w_{B,0} \\ w_{R,1} \\ w_{G,1} \\ w_{B,1} \\ w_{R,2} \\ w_{G,2} \\ w_{B,2} \end{bmatrix} \qquad \varphi_1(x_i, y_i) = \begin{bmatrix} \phi_R \cdot (1 - |y_0 - y_i|) \\ \phi_G \cdot (1 - |y_0 - y_i|) \\ \phi_B \cdot (1 - |y_0 - y_i|) \\ \phi_R \cdot (1 - |y_1 - y_i|) \\ \phi_G \cdot (1 - |y_1 - y_i|) \\ \phi_B \cdot (1 - |y_1 - y_i|) \\ \phi_R \cdot (1 - |y_2 - y_i|) \\ \phi_G \cdot (1 - |y_2 - y_i|) \\ \phi_B \cdot (1 - |y_2 - y_i|) \end{bmatrix} \tag{5.2}$$

Having specified how the feature functions for unary and pairwise potential are defined, it is possible to perform parameter learning and then inference process. As the weight vector for unary potential contains nine elements, and two elements for pairwise potential, in overall, it is necessary to adjust eleven weights.

5.3 Dataset

In the thesis, all images were artificially generated in a separate component of the described system. Colouring of images was adjusted to the tasks of each experiment, however, the basics of the generation process stayed the same. for this part of the system, a single generated image can be divided into maximally four differently coloured sections, which have the role of a background. Then, in each section, there might be one foreground object placed and it can be a circle, a square, a pentagon or a hexagon. There are also situations in which a foreground object is placed

on the area of two sections. In such a way two sets of images were generated, namely a training set containing 200 images and a testing set containing 10 images. When it comes to the size of images, both width, and height was equal to 400px. This means, that during the training process, in which gradient should be computed based on data from every pixel in all images in a training set, it would be required to perform computations for 32 million cases in every single epoch. That is why all the images were subject to the process of superpixel segmentation into roughly 500 superpixels, which limited the number of required computations 320 times.

For the first experiment that was conducted, the generated images were composed of only three colours: red, green and blue. Figure 5.5 presents six sample images from a testing set that were used to evaluate whether semantic segmentation was successful. A black border around the images is not a part of the given image, it was added for the sake of better visibility.

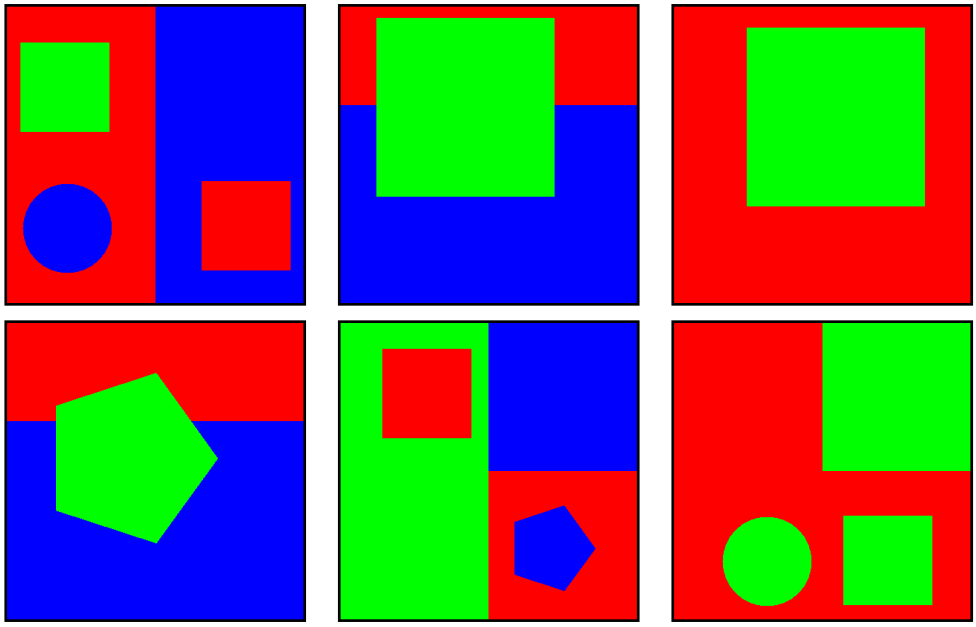


Figure 5.5: *Sample test images generated for the first experiment.*

In order to perform the semantic segmentation also a training set is needed, which is composed of images generated in the same way as testing images. However, in order to perform supervised learning not only training images are needed, but also the expected labellings for each of those images. The part of the system described in this chapter performs segmentation of objects into three classes, each specified by one label. All red objects should be assigned to label 0, all green to class 1 and all blue objects should have label 2. For each image from the training set, another image was generated that represents correct labelling of a given image. Pixels with label 0 assigned are painted in black, those with label 1 are white, and pixels with label 2 are grey. Figure 5.6 depicts six pairs of images representing sample training data and their expected labellings.

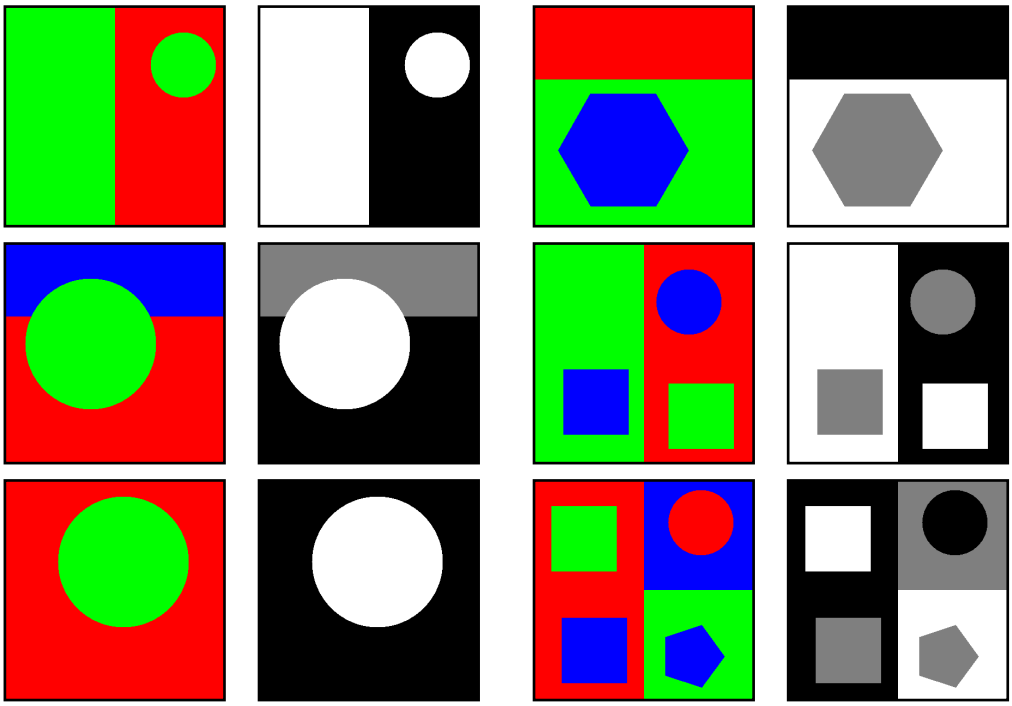


Figure 5.6: *Sample train images and their correct labellings generated for the first experiment.*

For the next experiment, different training and tests sets were required, which were to be composed of images containing not only primary colours but also different shades of them. Those sets were obtained by modifying the already generated images from the first experiment. Firstly, those images were divided into superpixels and then the colour of each superpixel was slightly changed in a random way. Figure 5.7 presents six pairs of training images with their expected labellings that were generated for the second experiment.

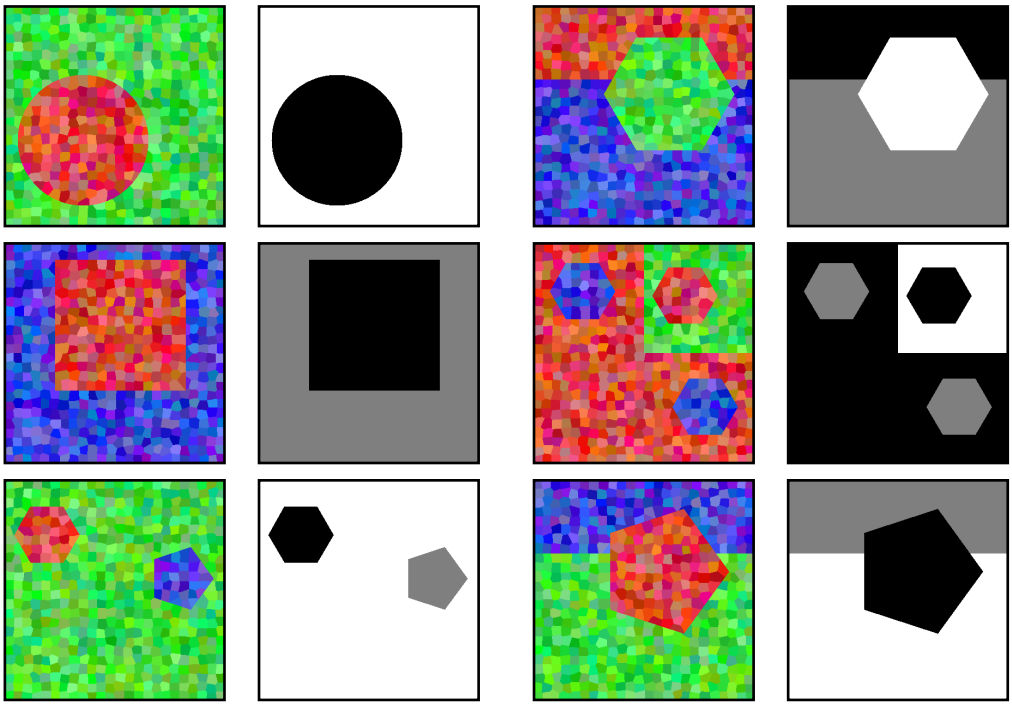


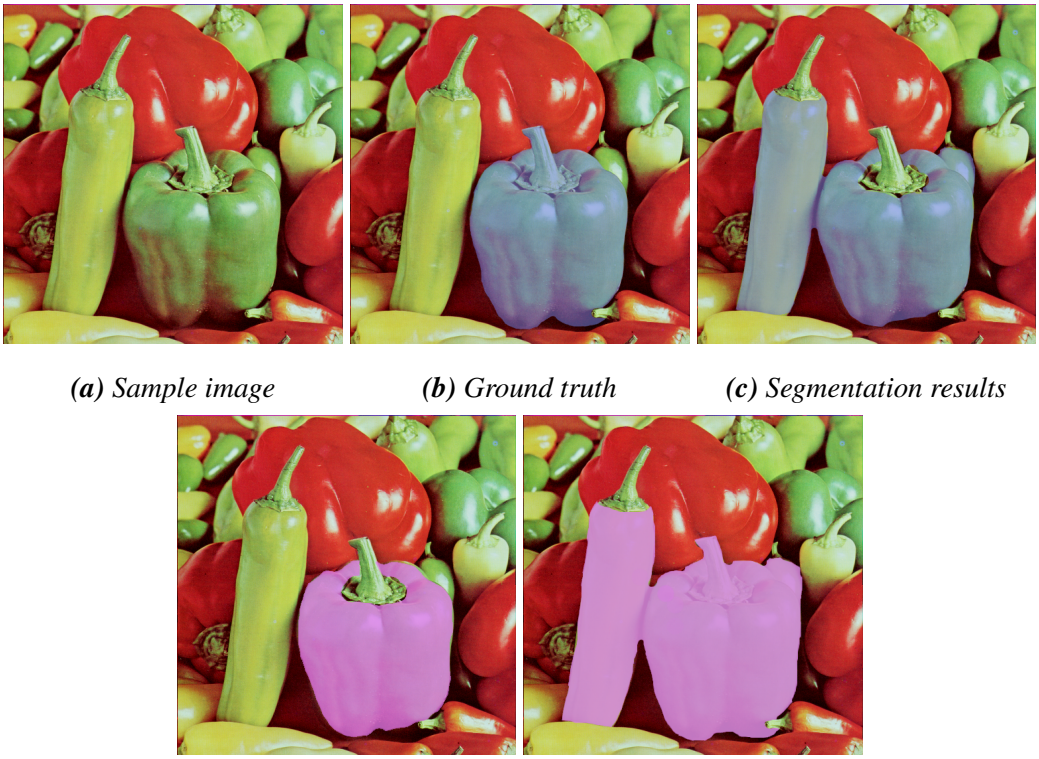
Figure 5.7: *Sample train images and their correct labellings generated for the second experiment.*

5.4 Evaluation method

In order to state whether the performed semantic segmentation was successful it is necessary to propose a method of assessing correctness of the resulting labelling. A trivial approach would be to present the precision of a method in terms of a percentage of correctly labelled superpixels. However, this is not a very informative method as it is enough to label each superpixel with the most popular class to obtain a high score. That is why, other assessment methods are required. One of the standard ways to evaluate a given segmentation method is to use the Jaccard Index[48], more known as Intersection over Union (IoU). In semantic image segmentation, IoU measures the similarity between ground truth and experimental results separately for each class. In order to visualise how this measure is obtained, Figure 5.8 was prepared. It shows a sample image on which semantic segmentation was performed. Ground truth is marked on a separate image as a semi-transparent blue layer. Similarly, Figure 5.8c presents the experimental results of a given segmentation method. Then, to calculate IoU two sets are needed, an intersection set, which is shown in Figure 5.8d that presents a mask of all pixels that were correctly labelled, and a union set from Figure 5.8e, which shows all pixels from the ground truth and from the experimental result marked together. Then, IoU can be calculated as a ratio between those two sets according to the formula 5.3.

$$IoU = \frac{\text{true positive}}{\text{true positive + false positive + false negative}} \tag{5.3}$$

The final evaluation of semantic segmentation results is based on an mean value of Intersection over Union (mIoU) for all test examples and all available classes, and this metric was used to assess the precision of results in each of the performed experiments.



(a) Sample image *(b) Ground truth* *(c) Segmentation results*

(d) Intersection *(e) Union*

Figure 5.8: *Intersection and union of ground truth and segmentation results for a sample image needed to compute the Jaccard index.*

5.5 Experimental results

Before conducting any kind of machine learning experiment there is a need to specify values of hyperparameters and coefficients that are required for proper functioning of the system. Table 5.1 presents chosen values of each parameter that was needed to perform all experiments described in this chapter.

Table 5.1: *Values of hyperparameters required for colour-based segmentation.*

Parameter name	**Parameter value**
Number of images (inputs)	200
Number of states (outputs)	3
Number of superpixels	500
Training step	0.0001
Regularisation factor	1000
Number of training epochs	5
Convergence tolerance	0.1

5.5.1 Semantic segmentation on tricoloured image

The first experiment of this part of the system was aimed to perform segmentation on a set of simple images composed of only red, green and blue regions as it has already been presented in *section 5.3: Dataset*. Every image was modelled with a factor graph containing two types of nodes: input, and output. Nodes of the first type contained information about properties of a given superpixel, which in this part of the system were three features representing three components of a superpixel colour. For this experiment the RGB colour space was chosen to model those features, hence, in a given image a feature vector of a superpixel can take only one of the three possible values $[255,0,0]$, $[0,255,0]$, and $[0,0,255]$. The goal of this experiment was to differentiate red, green, and blue regions, thus, the output nodes can also have one of the three possible values: label 0 for red superpixels, 1 for green, and 2 for blue ones. After the processes of feature extraction and factorisation are completed, the next step is parameter training by means of Stochastic Gradient Descent. In this process, eleven weights of the system were adjusted to match the training set

of images that was already described. The goal was to learn such weights that the total energy of a factor graph for a given image would be minimised. Hence, for label 0 red component of colour should be promoted, thus having a small weight, for label 1 a green one, and for label 2 a blue colour component. For this experiment weights for the pairwise potential are not so important as there is no noise in training images. Table 5.2 shows values of weights after the training process was completed. It can be seen in the table that weights for the local potential match the expectations, hence, the training process was successful.

Table 5.2: *Trained weights for segmentation of tricoloured images.*

local weights			
	red	green	blue
label 0	−0.04148	0.02089	0.01220
label 1	0.02054	−0.04250	0.01399
label 2	0.02095	0.02162	−0.02620
pairwise weights			
$y_i = y_j$	−0.21918		
$y_i \neq y_j$	0.21918		

Then, with a trained model an inference process for unknown images can be performed. Figure 5.9 depicts five sample test image together with their output labellings being a result of the inference process and a reference labelling. On all three types of images borders of superpixels were marked for the sake of better visualisation of the results. Based on the presented images it can be stated that the semantic segmentation of test images was successful. As expected, each red shape was marked with label 0, shown on the resulting image in black colour. Similarly,

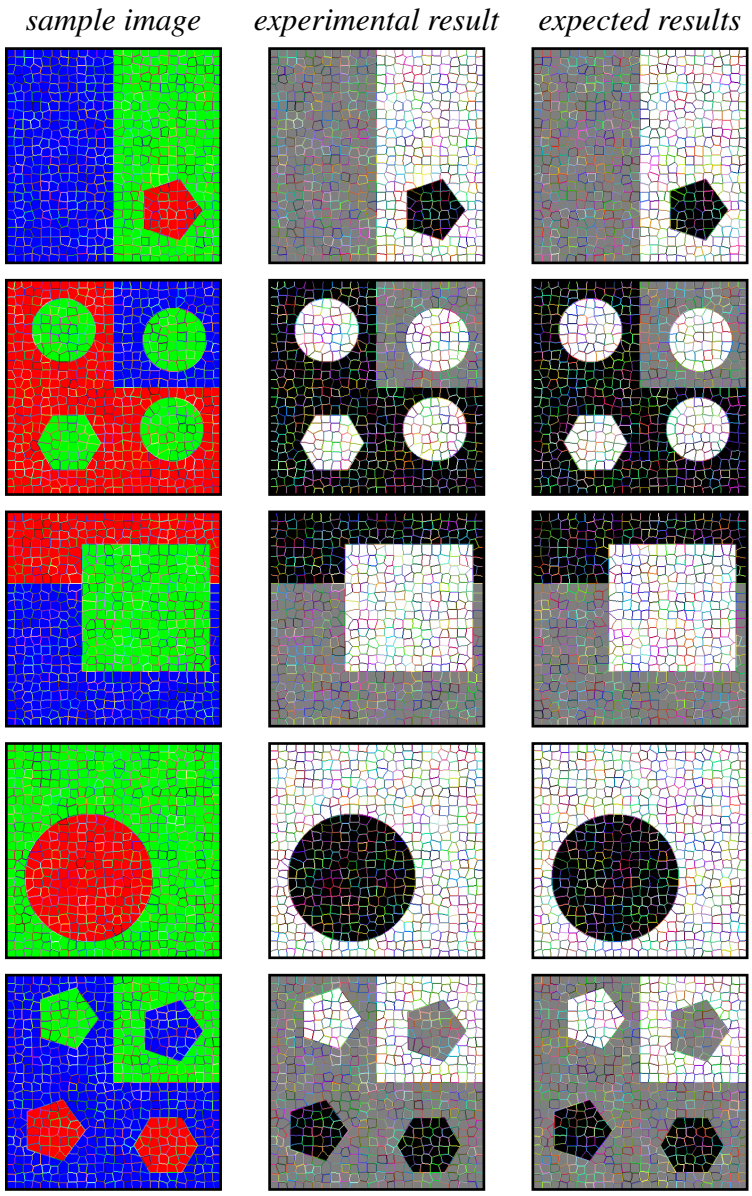


Figure 5.9: *Experimental results of semantic image segmentation on sample tricoloured images.*

all green and blue regions have a proper class assigned, which are label 1, painted in white and label 2 marked in grey, respectively.

The accuracy of the described segmentation method is presented in Table 5.3. For each label, Intersection over Union was computed based on the presented test set. The final accuracy of the method is expressed as mean Intersection over Union (mIoU), and for this experiment, the precision of results was equal to 100%.

Table 5.3: *Accuracy of segmentation for tricoloured images expressed in IoU for each class.*

class	label 0	label 1	label 2	mIoU [%]
IoU [%]	100.00	100.00	100.00	100.00

5.5.2 Semantic segmentation on multicoloured image

The goal of the second experiment was exactly the same as in the first one, which is to perform semantic segmentation of images into three classes. However, in this experiment a training and a testing set were composed of multicoloured images. Instead of having only red, green and blue superpixels also shades of those colours were introduced. Moreover, in this experiment importance of the pairwise potential will be tested as there are some superpixels in the test images which colour is less similar to the respective basic colour, meaning red, green, or blue, than the colour of its neighbours. For such regions, pairwise potential should promote a situation in which this superpixel will take the same label as the adjacent superpixels. The procedure of segmentation is exactly the same as previously, however, in this experiment feature vectors are composed of continuous values representing a shade of a given superpixel colour component and not one of the two available values 0 or 255, as it was before.

Table 5.4 shows the values of weights obtained by the process of parameter training. Once again, for label 0 the red component is smaller than the others, for label 1 the green component, and for label 3 the blue one, which is an expected behaviour. Pairwise weights are similar as in case of the first experiment.

Table 5.4: *Trained weights for segmentation of multicoloured images using RGB features.*

local weights			
	red	green	blue
label 0	−0.019 30	0.009 67	0.014 06
label 1	0.013 12	−0.013 69	0.015 13
label 2	0.006 18	0.004 02	−0.029 19
pairwise weights			
$y_i = y_j$	−0.235 32		
$y_i \neq y_j$	0.235 32		

After parameter training is completed, with the use of the established weights an inference process can take place. Similarly as in the first experiment, the results of this process are depicted in Figure 5.10. As the main goal of this experiment was to test whether the pairwise potential has a positive influence on the final segmentation, there was an extra column added, labelled as *experimental results (E_1 only)* that contains the resulting images which were obtained by setting both weights of pairwise potential to 0. Final results are presented in column titled *experimental results (E_1 & E_2)*.

Basing on the presented results of the inference process on test images, it can be stated that the presence of the pairwise potential improved the segmentation results. This is the most noticeable in the third test image for a red region. Without the pair-

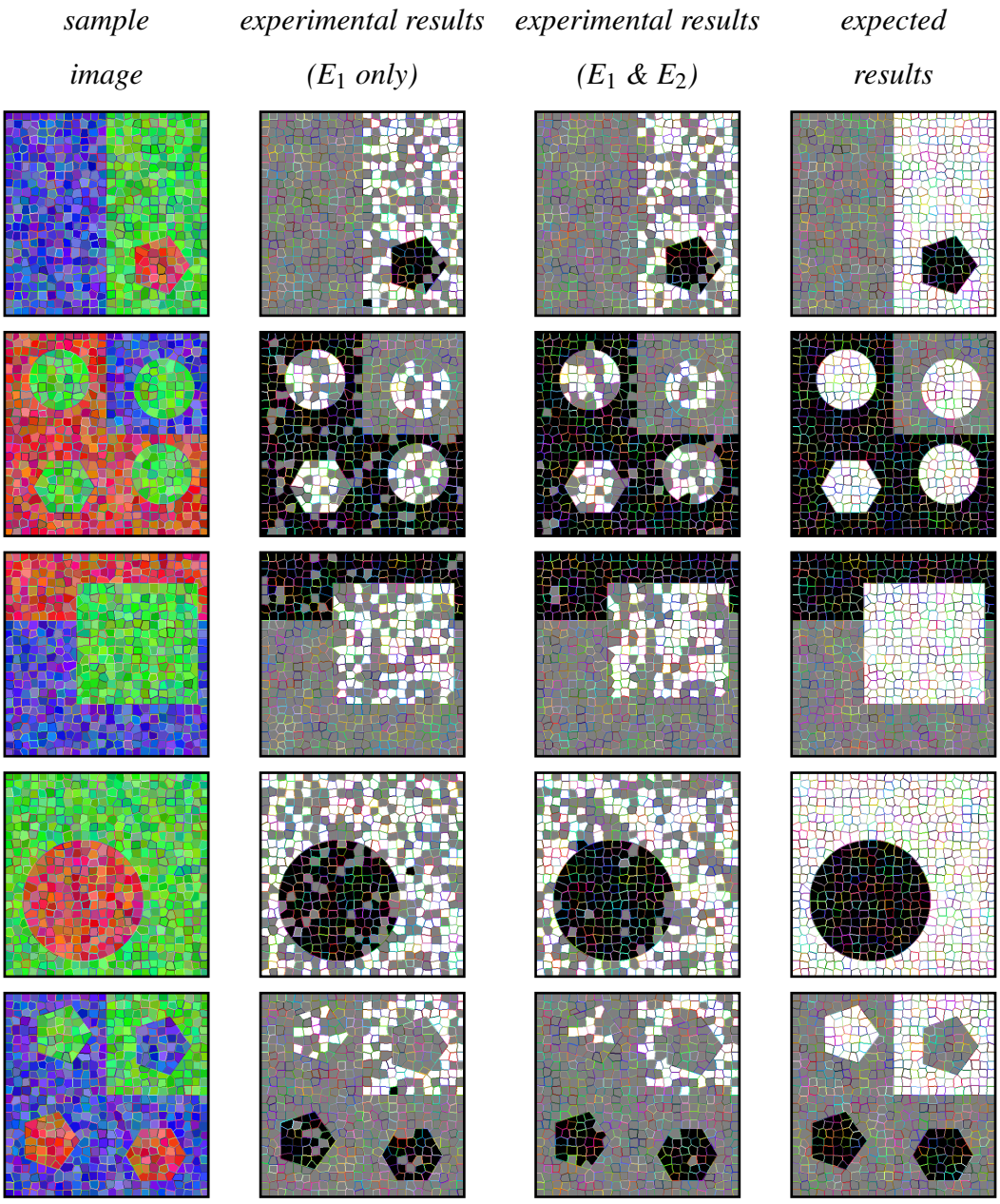


Figure 5.10: *Experimental results of semantic image segmentation on sample multicoloured images in RGB colour space.*

wise potential 10% of superpixels in this region were incorrectly classified, whereas after including pairwise potential this value dropped to 2%. Nevertheless, the results of the segmentation for multicoloured images are very poor regardless of the presence of the pairwise potential. Computed accuracy of the segmentation on the set of test images expressed in terms of mean Intersection over Union was equal to 69.41% for the case with local potential only, and 71.74% for the final result, as shown in Table 5.5. What is interesting is that incorporating the pairwise potential resulted in worsening of prediction abilities for green pixels, which is visible in the last presented test image.

Table 5.5: *Accuracy of segmentation based on RGB features for multicoloured images, expressed in IoU.*

class	label 0	label 1	label 2	mIoU [%]
IoU [%] (E_1 only)	84.24	54.72	69.27	69.41
IoU [%] (E_1 & E_2)	91.52	52.59	71.12	71.74

The obtained results are not satisfactory, therefore a modification to the described process was required. Instead of expressing colour features in RGB, the CIELAB colour space was introduced and the whole procedure was repeated. Table 5.6 presents trained weights of the described system after this modification.

Table 5.6: *Trained weights for segmentation of multicoloured images using CIELAB.*

local weights			
	luminance	a	b
label 0	−0.00044	−0.00841	−0.00680
label 1	−0.00107	0.01484	−0.01026
label 2	0.00151	−0.00643	−0.01706
pairwise weights			
$y_i = y_j$	−0.23760		
$y_i \neq y_j$	0.23760		

In RGB colour space it was possible to judge whether the weights were adjusted properly, simply by checking which of the three colour components for a given label had the smallest value, however, in CIELAB it is not so straightforward, hence it is hard to know whether the training was correct just by looking at the resulting weights. Therefore, an assessment of the training phase is done indirectly via the segmentation results.

With the presented values of system parameters that were obtained by the Stochastic Gradient Descent method, the inference process for test images was repeated. Figure 5.11 depicts the results of the process of semantic segmentation performed on test images based on colour features extracted in CIELAB colour space. Once again, a separate column shows the output labellings for the local potential only, and another one for results obtained with the use of a full energy function meaning for both local and pairwise potentials.

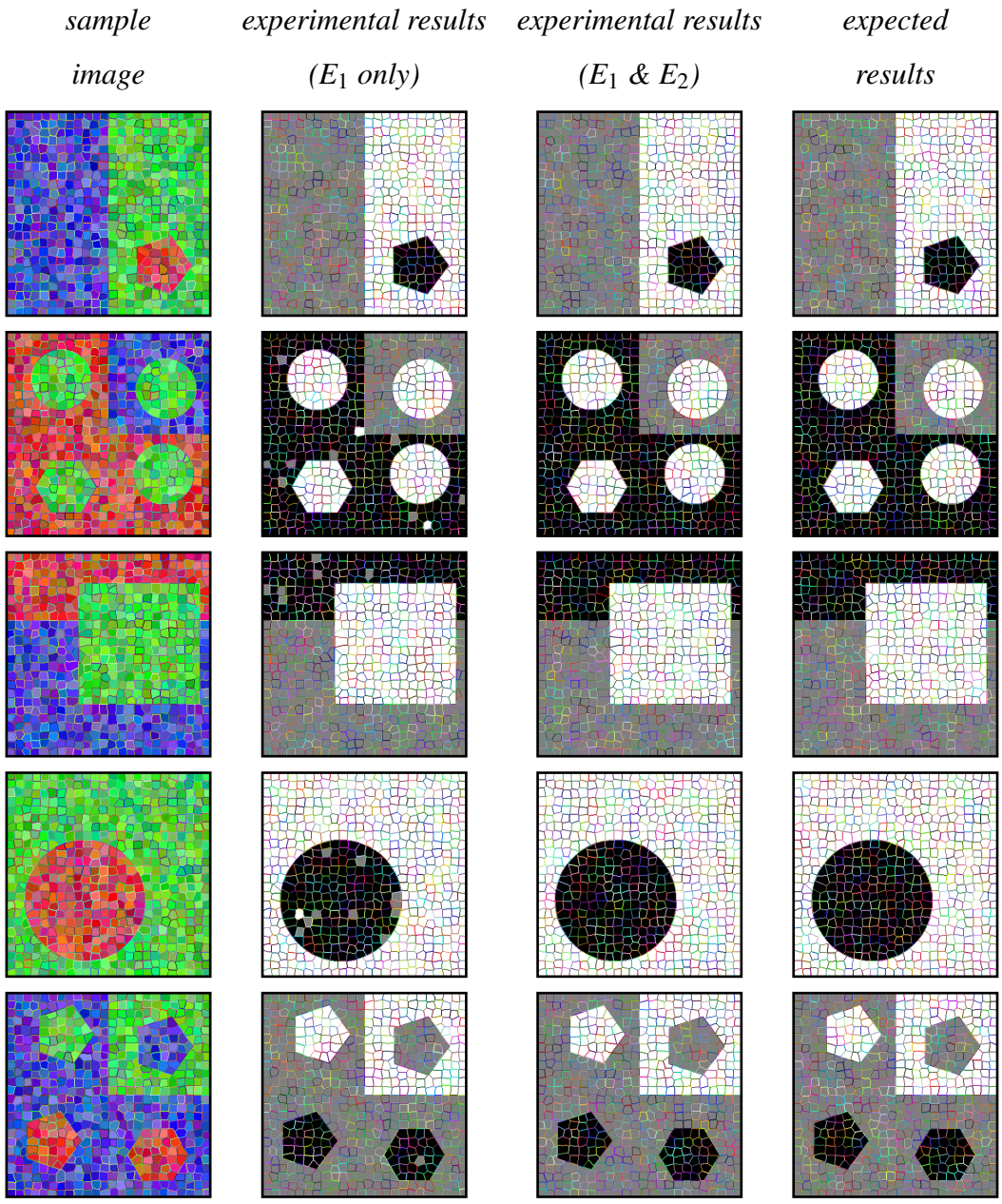


Figure 5.11: *Experimental results of semantic image segmentation on sample multicoloured images in CIELAB colour space.*

As visible from this column, the segmentation process was successful as all regions were correctly labelled. What is more, this experiment showed the importance of the pairwise potential. There are quite a few superpixels, which were not labelled correctly by means of the local potential only and after incorporation of the pairwise potential there were assigned to a proper class. This can be clearly seen for example in the second test image. On this image it is also visible that pairwise potential improved the results even on object boundaries, which is a more difficult task as then, not all neighbouring superpixels share the same label. Furthermore, the pairwise potential managed to fix incorrectly labelled superpixels even if one of their neighbours was assigned to a wrong class as well, which can be seen on the right boundary of a circle presented in the fourth test sample.

By representing colour features in CIELAB colour space a large improvement was achieved. Even with local potential only, the results are much better than final results of inference using the system based on RGB colour space. The accuracy of semantic segmentation of images represented in CIELAB colour space is equal to 100% if both local, and pairwise potentials are used and 90.90% for local potential only, as shown in Table 5.7.

Table 5.7: *Accuracy of segmentation based on CIELAB features for multicoloured images, expressed in IoU.*

class	label 0	label 1	label 2	mIoU [%]
IoU [%] (E_1 only)	97.15	87.22	88.34	90.90
IoU [%] (E_1 & E_2)	100.00	100.00	100.00	100.00

Chapter 6

Shape-based semantic image segmentation

This chapter will be devoted to the description of the second part of the created system. It will put an emphasis on differences between this part, and the part described in *chapter 5: Colour-based semantic image segmentation*. The main difference lies in a definition of the feature function, which in this chapter was inspired by an article "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context" [42] written by Jamie Shotton et al. A more complex feature function is needed as the goal of this part of the system involves detection of objects based on their shape, and not only on their colour.

The first section of this chapter will be solely devoted to an explanation of why there was a need of changing the feature function and in how it was redefined. Furthermore, this chapter will contain a detailed description of how the probability

distribution of labels and features available in the dataset was estimated. The next two sections will be devoted to an overview of the experimental part. There were two different types of experiments conducted - semantic segmentation on noise-free images, and on noised images. For each type a generated dataset generated will be presented. This part of the system involved an automatic process of feature selection and an explanation of how it works, and which features were chosen for a specific experiment will be also provided in this chapter.

6.1 Feature function definition

Conditional Random Fields are best known for their ability to predict optimal configurations of multiple interdependent variables, which in case of semantic image segmentation are pixels or superpixels. The importance of relations between adjacent superpixels was already presented in the previous chapter, though only in terms of the pairwise potential. In this chapter the usage of contextual data also in the local potential will be described as the goal of this part of the system is to differentiate objects by their shape. In the dataset for the experiments of this part of the system an object looking like a letter H was introduced. This object is always in the shades of red and is surrounded by greenish superpixels. In order to prove that indeed the shape of the object is recognised, also squares and circles were involved in the dataset, which base colour and colours of the surroundings are the same as in case of the letter H. Figure 6.1 depicts three sample objects that differ only by their shape together with images representing the results of the segmentation process that are to be obtained. Similarly as before, label 0 is the class for reddish superpixels, label 1 for greenish and label 2 for those in the shades of blue, however, this time another

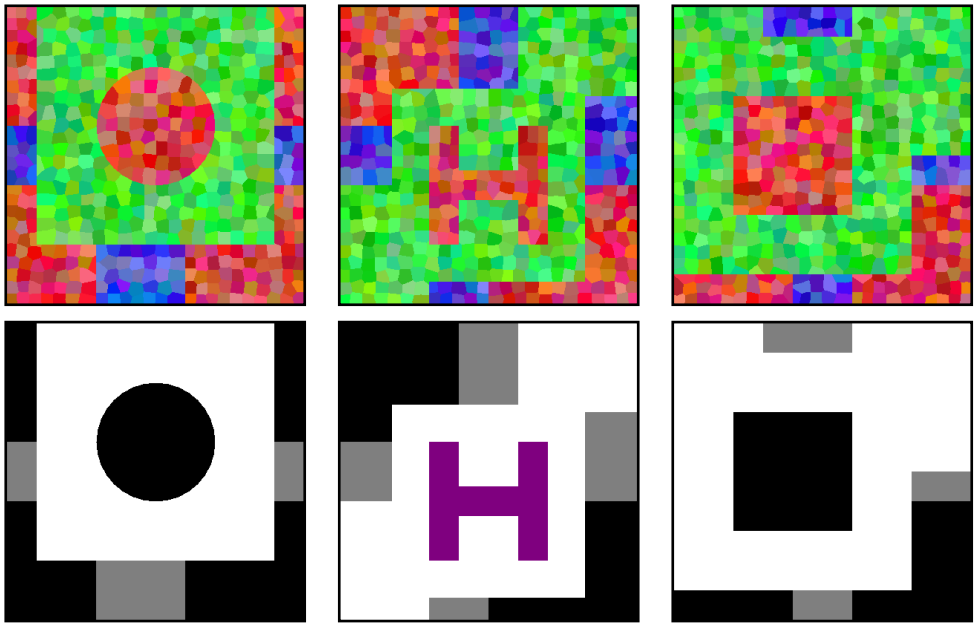


Figure 6.1: *Sample images that are to be differentiated basing on their shape and their expected labellings.*

class was introduced, denoted as label 3, which is devoted only to the objects with a shape of letter H.

6.1.1 Feature function for the unary potential

In this part of the system, a feature set used for unary potential is more diverse than previously, as it contains contextual data about image superpixels. In the already mentioned article by Jamie Shotton [42], the key component of the unary potential is a texture-layout potential, which operates on a texton map, and not on the original image. Therefore, before the segmentation process can take place a classification into textons needs to be conducted. Then, each pixel is assigned to exactly one texton, which makes it possible to model the relations between textures of neigh-

bouring pixels. However, the dataset generated for the experiments described in this chapter do not contain any texture information and the only difference between objects is their colours and shapes. Hence, instead of performing a process of textonisation, colour quantisation was introduced. Then, each assignment of the pixel to a given colour from the reduced colour palette simulates the assignment to a single texton. As images from the generated dataset are composed of regions in the shades of red, green, and blue, after the quantisation process tricoloured images are obtained, which are similar to the ones presented in the first experiment of the previous chapter. Those generated images are presented in Figure 6.2.

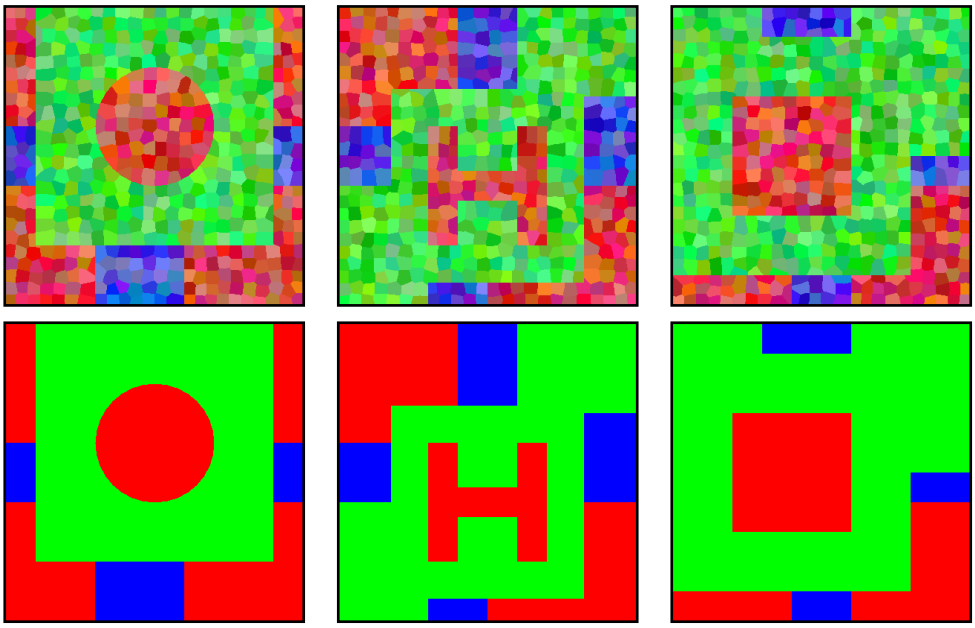
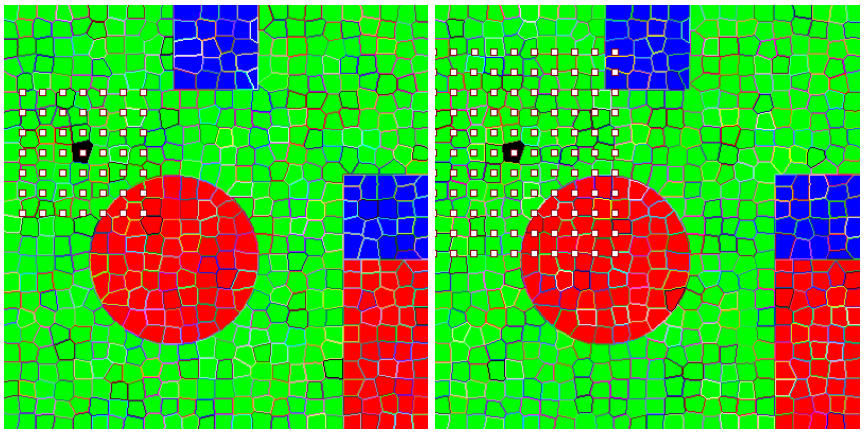


Figure 6.2: *Sample images after the process of colour quantisation.*

Having the dataset transformed into colour maps there is a need to extract features that are going to be used to compute the aimed unary potential. For the task of shape differentiation, two types of features were introduced and both of them include contextual data. The very first step that is needed obtain such features is to define a

boundary of influence for a given superpixel. It was achieved by creating a regular grid of points around the centre of mass of the chosen superpixel. The distance between grid points is defined as a mean distance between neighbouring superpixel centres in the whole image, and the size of a created grid is a hyperparameter that should be fixed before feature extraction takes place. Then, each point in a resulting grid represents exactly one neighbour which have an influence on a given superpixel during the energy computations. Figure 6.3 presents two grids, a 7×7 and 11×11



(a) *grid* 7×7 ***(b)*** *grid* 11×11

Figure 6.3: *Sample grid defining region of influence for a given superpixel.*

grid, that were created for the same sample superpixel in the same image which is marked with a black colour.

Having defined which superpixels have an influence on a superpixel under consideration, it is possible to define its feature vector. The first type of features that were used in the described system are colours of the superpixels from the grid. Those features are discrete, as after the process of colour quantisation only three values are available: red, green, and blue. Hence, taking for example a superpixel from a grid depicted in Figure 6.3a, the contextual colour feature can be expressed as a

vector of 49 values, each being a colour of one superpixel. Figure 6.4 depicts a visual representation of values from this vector with each feature being shown as a separate coloured box. The feature of a colour of the superpixel under consideration was marked with a pink border.

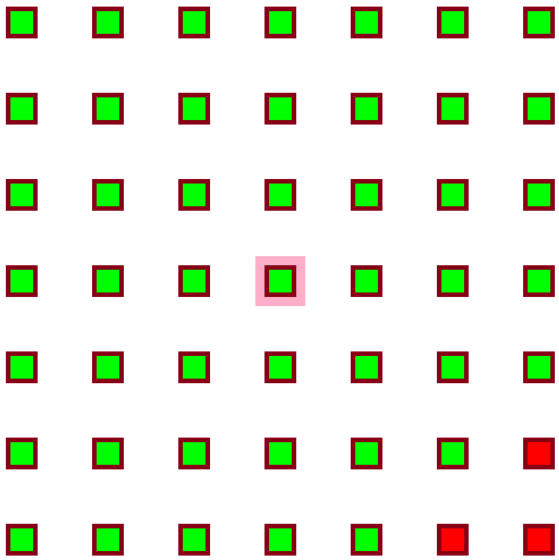


Figure 6.4: *Grid colour features for a sample superpixel.*

The second type of features used in the system was also based on the described grid. However, instead of taking the colour of a superpixel in which a grid point lies, the colours of this superpixel neighbours are taken into consideration. For every superpixel from the grid, the percentage of red, green and blue coloured neighbours is computed. Hence, for a grid 7×7 there are 49 triplets, which gives all together additional 147 features. Figure 6.5 serves as an example of how those features are calculated. In this image, values of 121^{st}, 122^{nd} and 123^{rd} features are to be computed, which are the percentage of red, green and blue neighbours for a superpixel that is marked in pink.

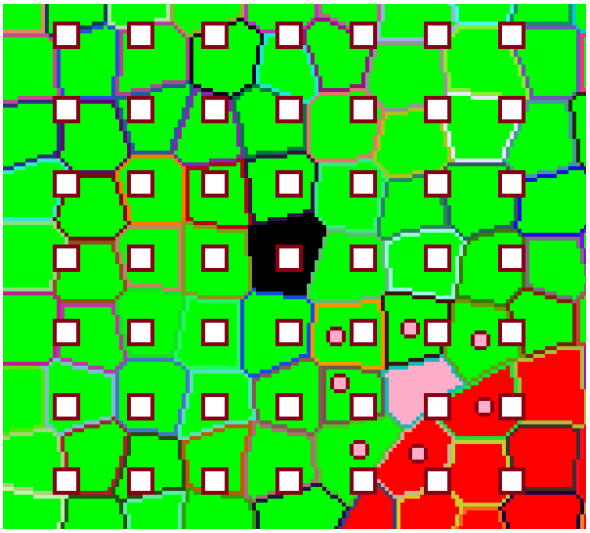


Figure 6.5: *A sample superpixel from the grid with its closest neighbours marked.*

The superpixel under consideration has seven closest neighbours, each marked with a pink circle, out of which five are green, two are red, and none are blue. Therefore, value of the 121st feature will be equal to around 0.29, 121st feature will have a value of 0.71, and for 123rd feature it will be 0. In such a way, every feature for every superpixel is computed. Similarly, as in the case of the previous type of features, also a visual representation of computed features might be useful especially when comparing two different superpixels. Figure 6.6 depicts such a representation.

For each superpixel, three boxes are shown, one per each of the available colours, with a border representing this colour. The percentage of red, green or blue neighbours is expressed as a number between 0 and 1, which is then is multiplied by 255. The inner colour of every box is representing exactly this value. It can be easily seen where there is a boundary between a green and a red region, as the features expressing percentage of green neighbours are gradually changing their colour from all white, meaning 100% of green neighbours to all black meaning 0% and con-

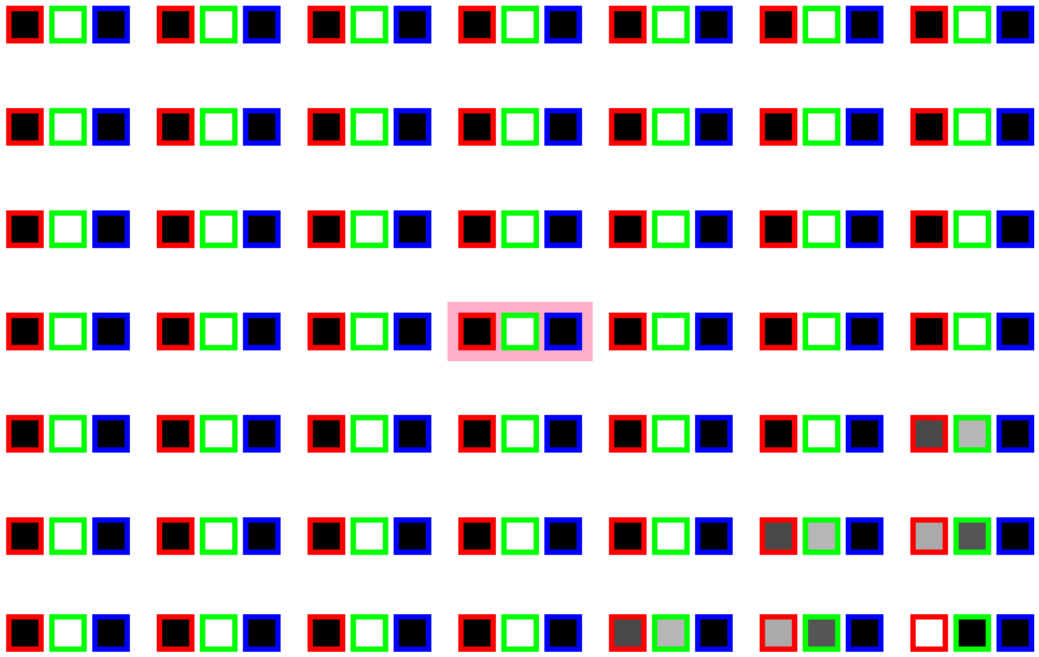


Figure 6.6: *Grid neighbour colour percentage features for a sample superpixel.*

versely a feature for red percentage goes through greyscale colours from black to white. Yet again, features of the superpixels that is under consideration are marked with a pink border.

The presented example was based on the grid 7×7 in which the percentage of neighbours of the given colour was computed on the closest superpixels only. However, sometimes it might be beneficial to include a larger neighbourhood. Then, instead of processing only adjacent superpixels, also neighbours of those superpixels could be taken into account. The size of this neighbourhood may have a large influence on the results, especially on the borders between two regions. With larger neighbourhood, even if a superpixel has only one neighbour of a different class, this neighbour may be mostly surrounded by superpixels with the same class, and therefore the total percentage of this class may rise. The neighbourhood size

is yet another hyperparameter that should be fixed before feature extraction takes place.

With such a definition of features, each superpixel can be defined in terms of a feature vector containing 196 elements. Out of all these features, only such a set should be chosen which carry meaningful information that will allow differentiation of objects by shape. Performing segmentation with too many features is not only more time-consuming, but also less accurate. One of the most popular and simplest algorithms for finding an optimal set of features is a stepwise regression, which was implemented in the thesis. This is a greedy algorithm that creates a resulting set of features step by step. This method can involve only forward steps, only backward steps, or a combination of those two. Forward selection starts by an empty set of features and the first step is to choose such a feature from the whole feature set that will give the highest accuracy of segmentation for a validation set. Then, in each iteration a new best feature is added to the set of already chosen features. Figure 6.7 presents an example of how this method is applied for feature selection from a set of five available features, each depicted as a separate coloured block. Numbers in those blocks represent the accuracy of segmentation for a set of features under consideration. As presented, in the initial iteration a green feature gives the highest accuracy of 62%, therefore it would be chosen as the first element of the resulting set. Next, the violet feature would be added to a green one as the accuracy of segmentation based on this pair of features is the highest. Similarly, the red and blue features would be added in the following step. In the 5^{th} iteration the segmentation accuracy would drop by adding another feature to a chosen set, therefore in this step the algorithm would stop and the final feature set would be the one chosen in the previous iteration.

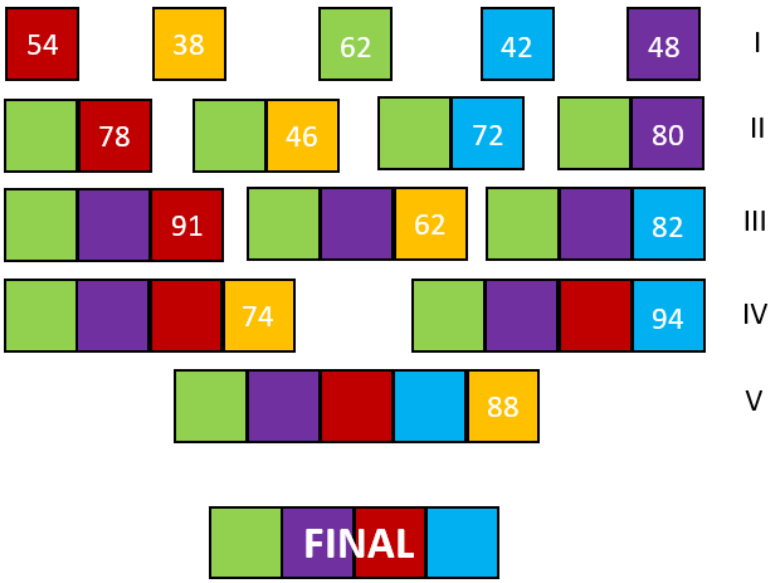


Figure 6.7: *Greedy forward feature selection.*

In every step, the best feature is added to the chosen set of features, therefore there needs to be a method of assessment based on which feature set the segmentation will give the highest accuracy. As the process of feature selection is applied only to choose features for the unary component of energy, there is no need to perform a full inference process to check the segmentation accuracy. Instead, it is enough to choose the most probable label for each superpixel basing on the computation of unary potential and compare the final labelling to the ground truth, and this is the measure that was implemented to choose the best feature in the current step. Hence, in the first iteration for all sets containing only one element this procedure is performed and the feature with the highest accuracy is added to the final set. Then, to this feature every other feature is added forming sets of two elements and again, the accuracy of segmentation for each set is measured. This procedure continues iteratively until no more features are left, or until a situation in which adding any other feature results in worsening of the model performance. Backward feature

selection is a very similar process, however, instead of adding new features to the set under consideration, feature elimination is performed. The procedure starts with a full set of available features and then iteratively, the worst feature is removed from this set until further elimination results in poorer accuracy of segmentation. Another possibility is to join those two methods, and make some steps forward and then some steps backwards.

Next part of the segmentation process that happens after feature selection concerns a feature function definition. Initially, the same feature function was used as in the case of colour-based segmentation, however, the obtained results were really poor. Not only did the method fail to distinguish between red objects of different shapes, but it was also not capable of assigning the proper class to any blue region. Figure 6.8 presents the results of such segmentation on four sample images.

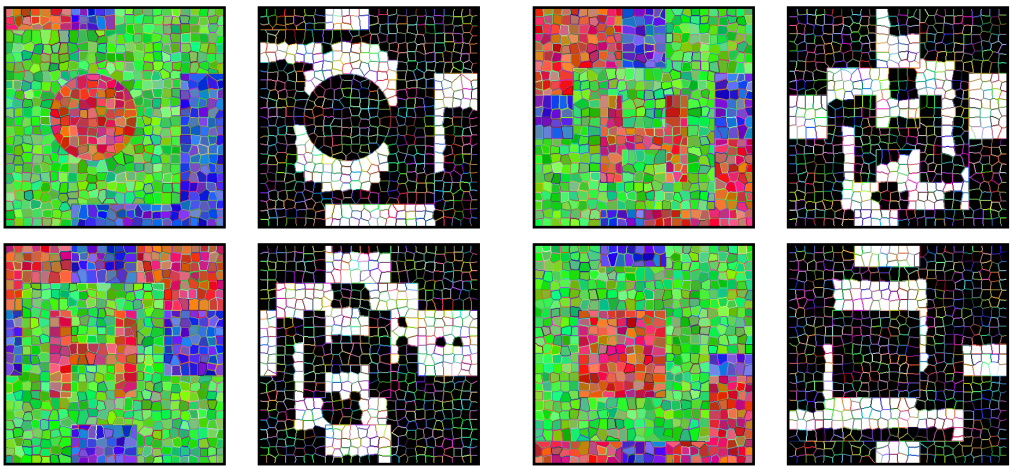


Figure 6.8: *Results of shape-based semantic image segmentation using a standard feature function.*

As the already implemented feature function did not perform well on the task assigned for this part of the system, a different way of representing features had to be

chosen. For the experiments described in this chapter, a feature function for unary potential was expressed as a conditional probability of assigning a label to the current pixel given its features. Then, having an optimal feature vector constructed, it is possible to express unary potential as a negative log-likelihood of a superpixel label given its features as in equation 6.1.

$$\varphi(x_i, y_i) = -\log p(y_i|f_i) \tag{6.1}$$

In order to compute a probability of a label conditioned on superpixel features, it was necessary to know what is the probability distribution of all labels and all features that were chosen during the process of stepwise regression. Estimation of this distribution is made only once, just after the stage of preprocessing, which divides images into superpixels, and is then used during the training and inference phases to compute the system energy.

6.1.2 Probability distribution estimation

The methods of probability distribution were already explained in *chapter 3: Structured Prediction*. This estimation is conducted separately for every feature and it is based on a training set of images. For the first type of features, which are dependent on a colour of neighbouring superpixel and its position in a grid, estimation is pretty simple. Because of the fact that each such feature is discrete, it is enough to go through the whole training set and count how many times a given colour appears in a given position. Then, by dividing this number by the total number of superpixels in all training images the probability of a given feature is obtained. For every feature, probabilities of each label are cached, and then in a training and inference processes conditional probability $p(y_i|f_i)$ can be computed straight away.

For features representing a percentage of colours in the neighbourhood of superpixels from the already described grid, the probability distribution is more difficult to obtain. Those features have continuous values and therefore it is not possible to count how many times a given value occurs in the training set. Initially in the thesis, this distribution was computed using Parzen-Rosenblatt Kernel estimation, which allowed to estimate a probability of an unknown sample for which the distribution was modelled. However, this method had one but huge drawback, as in order to obtain this estimation, it was necessary to iterate through every training sample. During the training process, in each epoch and for every superpixel in every image there was a need to compute this probability. Furthermore, while constructing a Gibbs sample the procedure had to be repeated for every label in every position in a created Markov chain. Because of this, the resources needed to for the training process were enormous, even for small datasets. Therefore, another method of probability distribution estimation had to be implemented. Similarly, as in the case of colour-position features it was necessary to choose such a method that will model this distribution only once, and give a probability of an unknown sample without a need of excessive computation. This can be obtained by means of probability histograms, in which the whole distribution is divided into a limited set of bins, and each bin is characterised by its probability of occurrence. The number of bins is the next hyperparameter of the described system. Tough the colour percentage features have continuous values, they are quite repetitive as the number of neighbouring superpixels is roughly the same and therefore, the number of histogram bins does not need to be large.

Having constructed histograms for each feature and each label, a conditional probability of a label on a given feature can be obtained. A description of histograms that are used to calculate this probability will be explained on an example of features

number 73,74 and 75. These features represent a percentage of red, green and blue neighbours of a middle superpixel, being a superpixel for which the label is to be predicted. Figure 6.9 shows histograms for 73rd feature divided into 17 bins that was constructed based on a grid of 7×7 elements and the neighbourhood size of two. The upper border value of each bin is marked on the x axis of the presented histogram. From this figure, the dependence between the colours of neighbouring

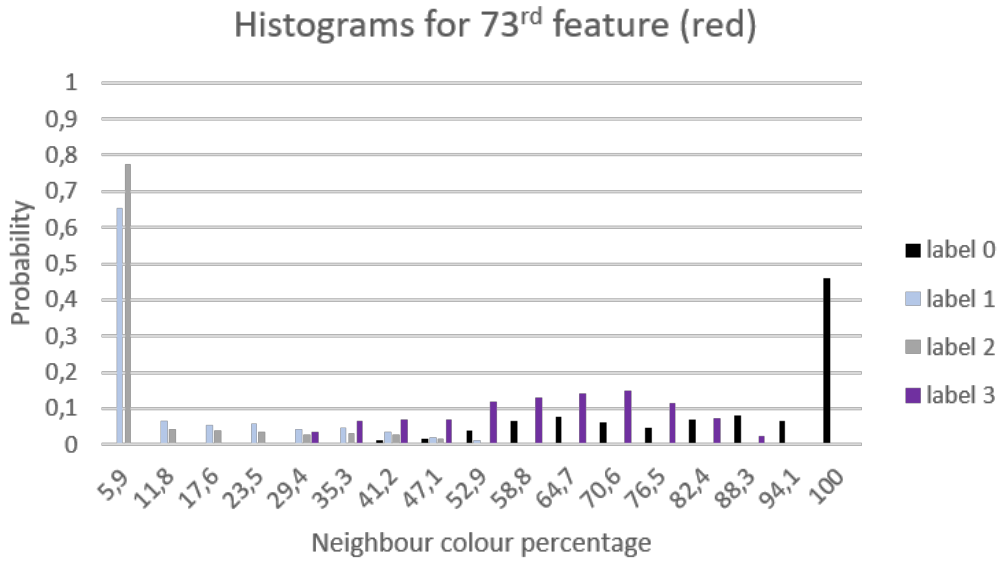


Figure 6.9: *Histograms for 73rd feature - percentage of red neighbours.*

superpixels and a label is clearly visible. If all neighbours are red then the most probable label for a given superpixel is 0. If not all, but the majority of superpixels are red, then either label 0 or label 3 are winning. On the other hand, if no neighbouring superpixels are red, then there is no chance of this superpixel to be classified with label 0, nor 3. The next histogram, for 75th feature, is presented in Figure 6.10. This histogram shows how green neighbours affect the label of the superpixel under consideration. If there is around 80% or more of green neighbours than the highest

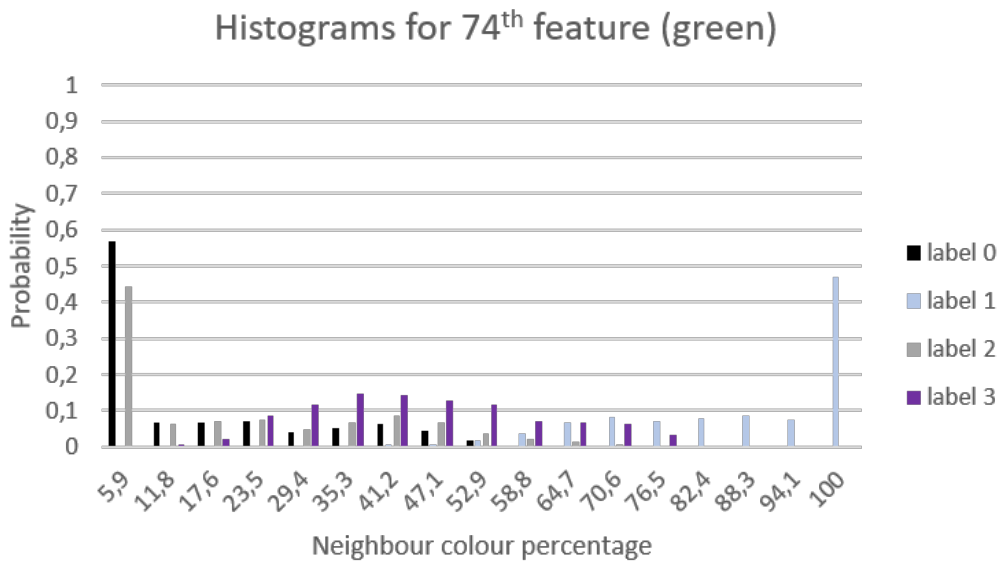


Figure 6.10: *Histograms for 74th feature - percentage of green neighbours.*

probability of occurrence has the label 1. For a smaller number of green neighbours, label 3 is the most probable, however, if there are only a few green superpixels in the neighbourhood or none then the given superpixel is not a part of the object of class 3. Last histogram, presented in Figure 6.11 was created for 75th feature. From this histogram it can be read, that label 3 may be assigned to the given superpixel only if there are no blue superpixels in the neighbourhood. Even if there is only one blue neighbourhood this probability drops to 0%, which is true because letter H is always surrounded by a green region. The bluer the neighbourhood is, the more probable is the label 2.

With the use of such histograms probability distribution for every feature and label was modelled. In order to get the probability of a given label for an unknown sample conditioned on a single feature from the chosen feature set, it is enough to check in which bin value of this feature falls and read its probability from the respective

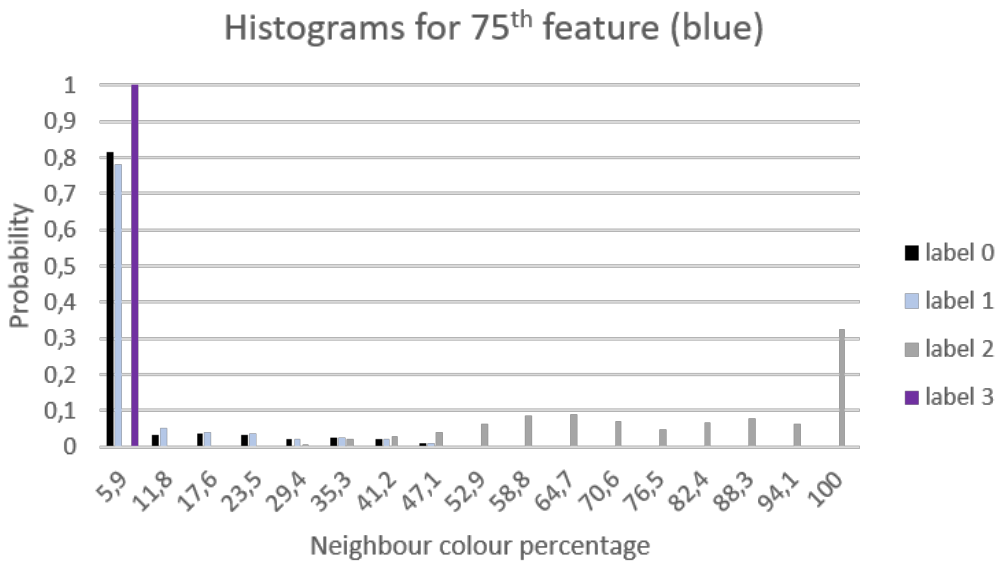


Figure 6.11: *Histograms for 74th feature - percentage of blue neighbours.*

histogram. By analysing such histograms for a combination of different features it is possible for Conditional Random Fields to learn a shape of objects that are present in an image. The resulting conditional probability is an output of the feature function for the unary potential that was chosen for this part of the system.

6.1.3 Feature function for the pairwise potential

The next component that is needed to compute the energy of the whole system is the pairwise potential and its definition needed to be changed as well in order to perform shape-based semantic segmentation. While the unary potential was responsible for detecting shapes, the role of the pairwise potential is to remove noises from the images. For this part of the system, feature function based on the Potts model was used. The general definition of this method was already presented in *section*

3.6.2: Pairwise potential. Computation of the pairwise potential is based on differences between features of neighbouring superpixels from the original image, and not from the image after the process of colour quantisation as it was in case of the unary potential. There are four possible configurations of relations between a pair of superpixels. Two superpixels can be similar in terms of features and have the same label, which should be promoted, or a different label, which should be penalised. Similarly, two not similar superpixels can be assigned to the same class, which is a segmentation mistake, or to different classes, which is expected behaviour. Hence, such a feature function had to be proposed to model those relations between superpixel similarity and their labels at the same time. A chosen feature function returns a vector of two elements, which is different depending on whether the labels of a superpixel pair are the same or different, as in equation 6.2.

$$\varphi(x_i, y_i, x_j, y_j) = \begin{cases} \begin{bmatrix} \exp\left(-\beta \left\|\phi(x_i) - \phi(x_j)\right\|^2\right) \\ 1 \end{bmatrix} & , \quad y_i = y_j \\ \\ \begin{bmatrix} 1 - \exp\left(-\beta \left\|\phi(x_i) - \phi(x_j)\right\|^2\right) \\ 1 \end{bmatrix} & , \quad y_i \neq y_j \end{cases} \tag{6.2}$$

The second element of both vectors is a unit term needed for bias, and the first one defines similarity between neighbouring superpixels, which is dependent on the difference between their features $\left\|\phi(x_i) - \phi(x_j)\right\|$. For experiments in this part of the system only one feature was chosen to compute the pairwise potential, and this feature is modelled as a three-dimensional vector of colours in CIELAB colour space. The difference between two colours is computed in terms of the Euclidean distance as in equation 6.3.

$$\left\|\phi(x_i) - \phi(x_j)\right\| = \sqrt{(L_i - L_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2} \tag{6.3}$$

To compute the pairwise potential also an image dependent parameter β had to be established. It is calculated basing on on the average value of colour differences between superpixels for a given image.

Hence, knowing the definitions of the feature function for unary and pairwise potentials it was possible to perform the process of parameter learning. As the unary potential is defined just by one number, representing the probability of occurrence of a given label conditioned on superpixels features, and the pairwise potential is in a form of a two-dimensional vector, there are three weights to be learned by Stochastic Gradient Descent method. Those weights model the importance of both types of potentials for a given problem. Then, with a trained system, it was possible to perform the experiments by means of inference.
The values of all hyperparameters that were required to perform segmentation in experiments presented in this chapter are shown in Table 6.1

Table 6.1: *Values of hyperparameters required for the experiment on noise-free images.*

Parameter name	**Parameter value**
Number of images (inputs)	800
Number of states (outputs)	4
Number of superpixels	500
Training step	0.0000001
Regularisation factor	5000
Number of training epochs	10
Convergence tolerance	0.000005
Number of histogram bins	17
Grid	7×7
Neighbourhood size	3

6.2 Semantic segmentation on noise free images

The first experiment that was conducted for this part of the system concerned semantic segmentation of noise-free images. The main goal was to distinguish red objects in a green neighbourhood, which differ from each other only by shape.

6.2.1 Dataset

In this part of the system three datasets were generated, which are a training set for Stochastic Gradient Descent algorithm, a validation set for feature selection, and a testing set. Each set consisted of three different image types: the original images, those after the process of colour quantisation and the ground truth representing the expected labelling. Figure 6.12 presents five sample images of each type that were generated for this experiment. The first column depicts original images that are to be segmented, which contain regions coloured with various shades of red, green and blue. On every image there is one red object on a green neighbourhood. The main goal of this experiment is to detect if a given superpixel is a part of an object with a shape of a letter H or a part of a different object. Next column presents images with a reduced colour palette, which are to simulate the result of division into textons. The last type of images represents the expected result. In this part of the system classification is done into four classes. Label 0 should be assigned to red regions which are not a part of a letter H. Label 1 and 2 are for green and blue segments respectively. The last class, marked with label 3, is devoted to objects in a shape of a letter H. On images representing the ground truth, regions with label 0 ale black, with label 1 are white, and grey colour is reserved for label 2. The last label is marked with magenta colour.

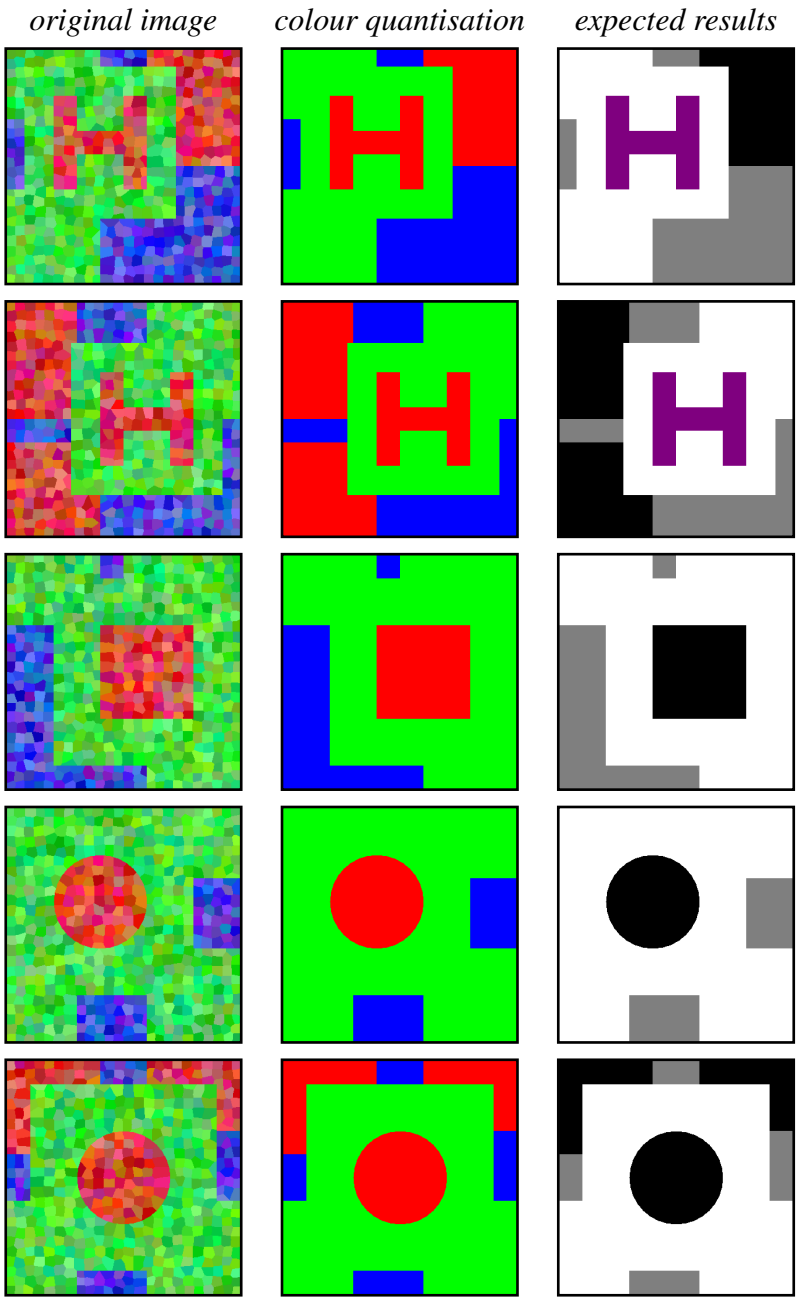


Figure 6.12: *Sample images generated for the problem of segmentation by shape on a noise-free case.*

6.2.2 Feature selection results

Before the training or inference processes can take place a feature selection should be performed and for this experiment, a stepwise regression algorithm with five steps forward and three steps backwards was used. From 196 available features, a set of 16 features was chosen. The optimal set is a combination of features representing colours of superpixels from the already defined grid as well as the percentage of red, green and blue superpixels in the neighbourhood of a given grid point. Figures 6.13 and 6.14 present visualisations of which features were chosen by the selection algorithm. The first figure depicts neighbour colour features for a sample superpixel, and the second shows neighbour colour percentage features. The way of feature presentation is the same as before, however, this time features which were not chosen are covered by an extra semitransparent layer, while those which form the final feature set are left unchanged.

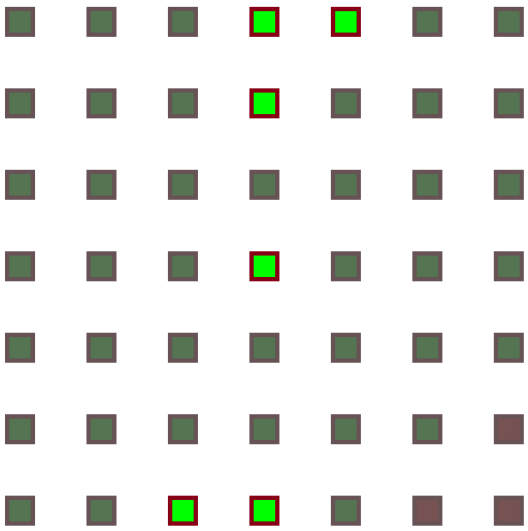


Figure 6.13: *Chosen grid colour features for experiments on noise-free data.*

Figure 6.14: *Chosen grid neighbour colour percentage features for experiments on noise-free data.*

Optimal features were selected based on a validation set of generated images, hence, to check whether the selection was indeed optimal an extra test was performed. For images from the testing set, which were not used during the feature selection process, the conditional probability $p(y|x)$ was computed for every label and for the selected feature set for every test image. The resulting probabilities for three sample images, each with a main object of a different shape, are presented in a visual way in Figure 6.15. In the first row an original image is depicted and the next four rows show images for calculated conditional probabilities, one per each label. Colour of the superpixel represents a probability that this superpixel will be assigned to the given class. The higher is this probability the lighter is the colour. Hence, when a superpixel has 100% chances of being assigned to some class, it will be white, and conversely, for 0% of probability, such pixel will be black.

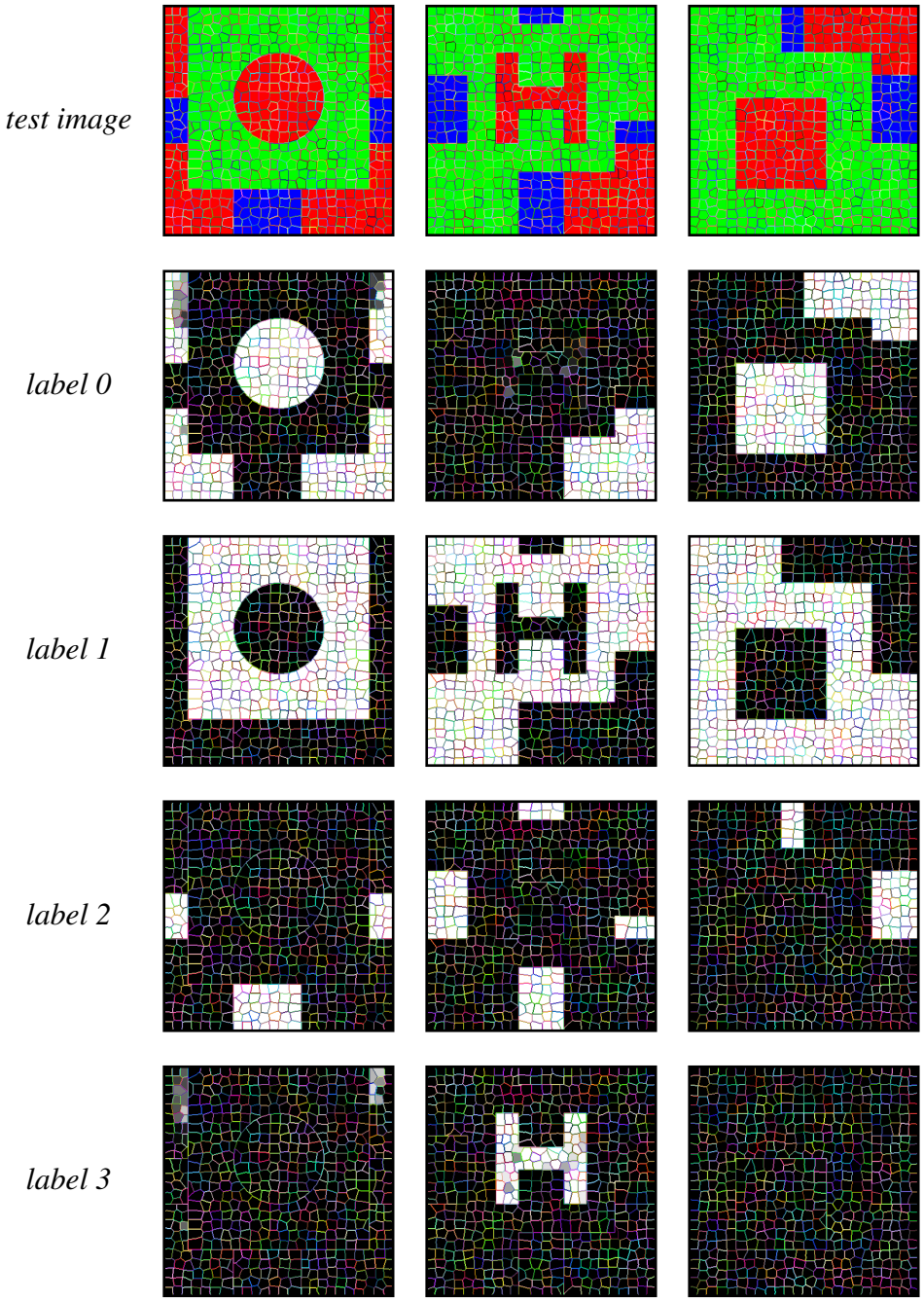


Figure 6.15: *Visual representation of conditional probabilities $p(y|x)$ for three sample noise-free images.*

In presented images, it is visible that the algorithm had no problem in assigning proper classes to green and blue regions. The difficulty of the task lies in labelling objects into classes 0 and 3, as the only difference between objects of those classes is their shape. For these classes, the resulting probability is smaller than 100%, therefore, there are some superpixels with different shades of grey.

The presented probabilities form a base of computation of the unary potential. If for some regions, the resulting probabilities for label 0 and 3 are similar, some noised classifications may occur. In such situations, a pairwise potential is needed to ensure that neighbouring pixels from the same object will have the same label.

6.2.3 Experimental results

After the process of feature selection was completed, parameter training by means of Stochastic Gradient Descent could take place. Values of the trained weights are presented in Table 6.2.

Table 6.2: *Trained weights for segmentation of noise-free images.*

w_1(unary potential)	$w_{2,1}$ (pairwise potential)	$w_{2,2}$ (bias)
0.11973	1.00649	0.38711

Having a model parametrised with those weights it was possible to perform inference on the set of unknown test samples. Figure 6.16 presents the results of the semantic segmentation process on sample test images. In the first column an original image that is to be segmented is shown, and in the second this image after the process of colour quantisation. The next two columns depict segmentation results,

one obtained with local potential only, and the second one with both potentials involved. The last column presents the expected labellings.

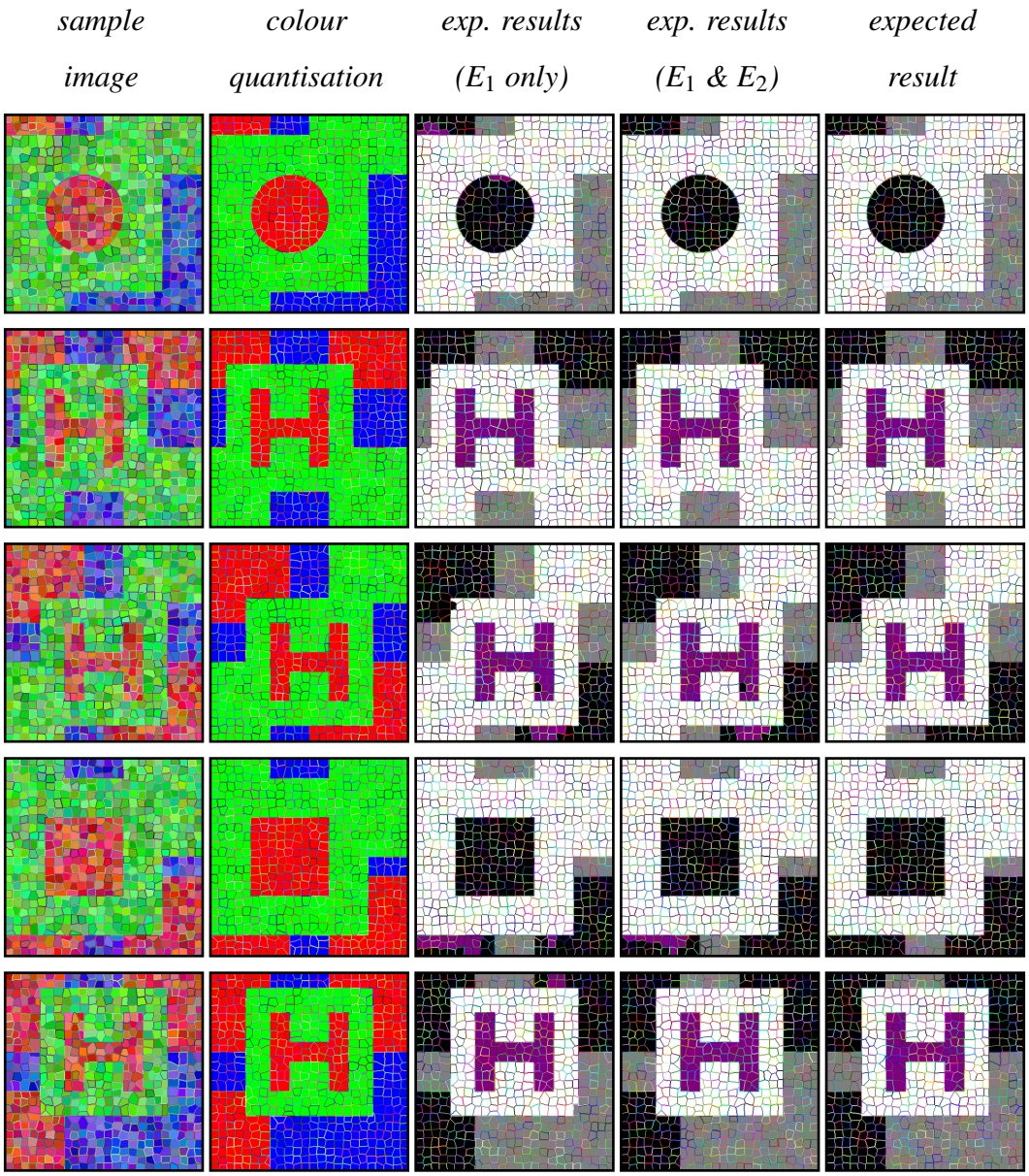


Figure 6.16: *Experimental results of shape-based semantic image segmentation on noise-free images.*

As presented, the semantic segmentation on the test set of images that was based on the new definition of the feature function was much more successful. Objects with a shape of the letter H were properly assigned to the class of label 3 and distinguished from other red objects, which belong to class 0. Green regions have label 1 and blue label 2, just as expected. When it comes to the comparison between the results obtained with local potential only, and final results, by including a pairwise potential the accuracy of the performed segmentation was improved. It is especially visible on individual pixels on object boundaries, which were assigned to an incorrect class by local potential but are correctly labelled in a final result. The last row of Figure 6.16 shows a large improvement after addition of the pairwise potential as a green superpixel at the bottom right corner of surroundings of the letter H had his labelling improved. Furthermore, also a classification of a group of four red superpixels that were incorrectly assigned to class 3, was corrected by using a full energy function. However, not in all cases did the pairwise potential manage to improve wrong labellings. It is especially visible in narrow, vertical, red regions surrounded only by green and red superpixels. In such cases, the system confuses them with a vertical stripe of the letter H and incorrectly assigns them to class 3. Tough, the final results show some improvement comparing to the results of a local potential only, not every superpixel has a correct labelling. The final accuracy of the described method on a set of test images expressed in terms of Intersection over Union is presented in Table 6.3.

Table 6.3: *Accuracy of segmentation based on shape detection for noise-free data.*

class	label 0	label 1	label 2	label 3	mIoU [%]
IoU [%] (E_1 only)	97.89	99.95	99.91	95.51	98.32
IoU [%] (E_1 & E_2)	99.32	100.00	100.00	98.84	99.54

As shown, after including the pairwise potential the accuracy of labellings for class 1 and 2 is equal to 100%. Classification of objects with the shape of the letter H has the lowest accuracy, though it is still on a level of 98.84%. Mean Intersection over Union over all labels for this experiment was equal to 99.54%. In general, the addition of a pairwise potential improved the final precision of the segmentation process by 1.22 percent points.

6.3 Semantic segmentation on noised images

The aim of the second experiment of this part of the system was exactly the same as in the first one, which was to perform semantic segmentation on images containing objects differing only by shape. However, this time some noise was introduced to the process of colour quantisation of original images. A reason behind the addition of noise was to simulate the wrong assignment of regions to textons during the process of texton map creation from the original work by Jamie Shotton [42]. The whole procedure of semantic image segmentation is the same as in case of noise-free images and the only difference lies in a generated dataset.

6.3.1 Dataset

The dataset for this experiment was created on the bases of the dataset of noise-free images. Just like before, it contained various images with regions in different shades of red, green and blue. In each image, there was one red object in a green neighbourhood that could have a shape of square, circle or letter H. Original images and those presenting expected results were exactly the same as in case of noise-free

images. The difference between those two datasets lies in images with a reduced colour palette as for the described experiment these images were subjected to the process of random addition of noise. For each image from training, validation and testing sets, which was obtained by the colour quantisation process, there was a 40% chance of wrong colour assignment. The number of noised superpixels was randomly chosen and varied from 1 to 20. Figure 6.17 presents five sets of sample images that were generated for this experiment.

Noise on the images has an effect on the whole probability distribution of features that is needed to compute the unary potential. Even if there are only a few noised images versus few hundreds of noise-free images the segmentation method may perform much worse. The best way to explain why the accuracy of the method is so dependent on the presence of noise is to take as an example classification of green and blue regions. For noise-free images, green regions are always labelled as 1^{st}, and blue as 2^{nd} class. Given the fact that in the set of all available features there is a feature k that represents the colour of a given superpixel, the probability $p(f_{i,k}|y_i)$ for example for a blue superpixel will be equal to 100% for label 2 and 0% otherwise. Then, no matter what are the probabilities of other features, probability $p(f_i|y_i)$, and consequently the final probability $p(y_i|x_i)$ of a given label conditioned on the current superpixel will also be equal to 100% for label 2, and 0% for other labels. Hence, the labelling chosen by the unary potential for those two classes will always be perfect. On the other hand, for noised images, even if there are only a few blue superpixels which do not have label 2, the probability $p(f_{i,k}|y_i)$ will be close to 0, but not equal to 0%. Hence, probabilities of other features would have a large influence on the final result, and the resulting labelling may be incorrect. Therefore, proper selection of features is crucial for the experiments on noised images.

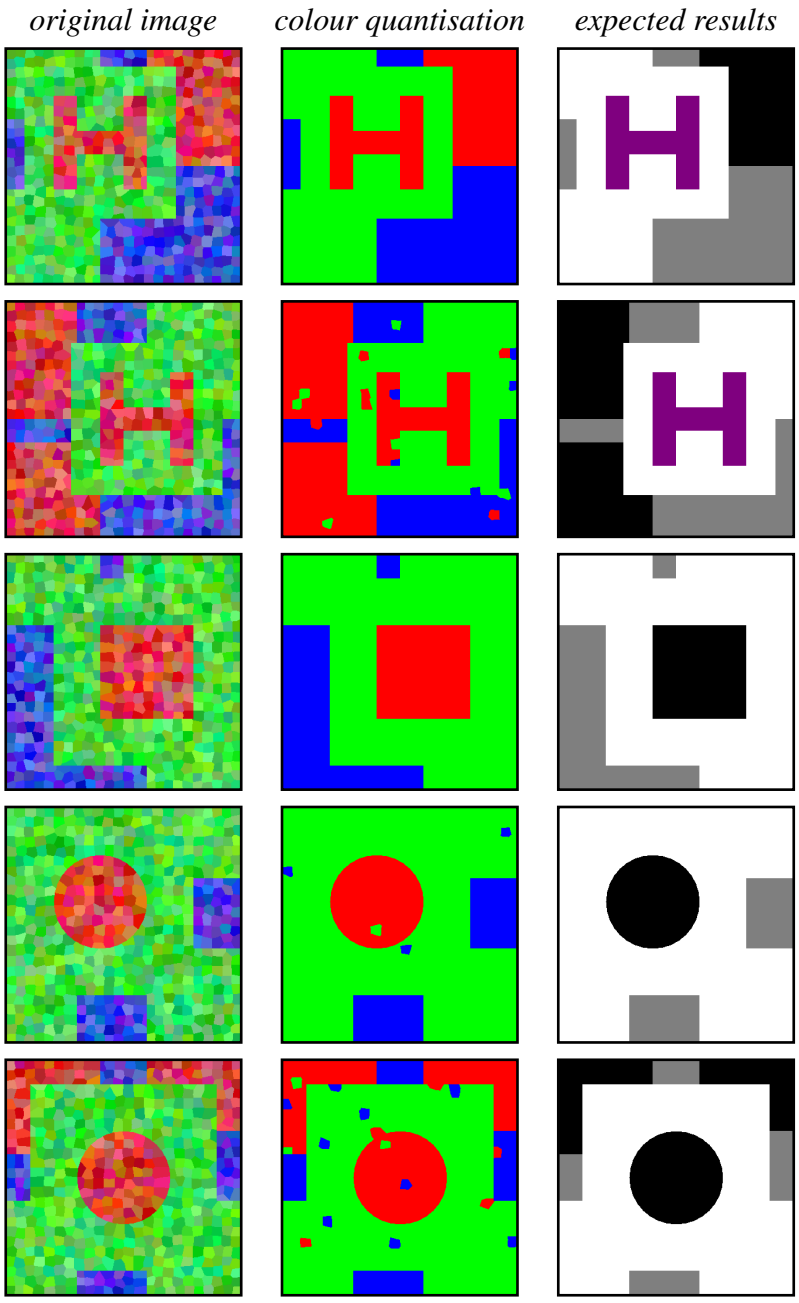


Figure 6.17: *Sample images generated for the problem of segmentation by shape on a noised images.*

6.3.2 Feature selection results

Feature selection was performed in the exact same way as in case of noise-free images, meaning by a stepwise regression algorithm with five steps forward and three steps backwards. As a result of this process, from a set of 196 available features, 10 features were chosen. Figures 6.18 and 6.19 depict which features were chosen for this experiment just like in case of the previous experiment.

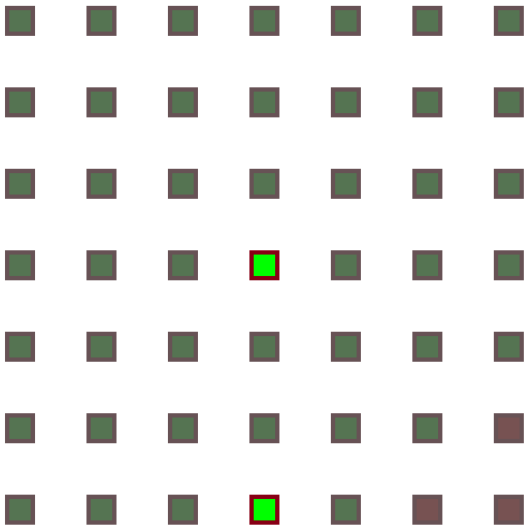


Figure 6.18: *Chosen grid colour features for experiments on noised data.*

When it comes to grid colour features only two were chosen, which is four less than in case of the optimal feature set for segmentation of noise-free images. However, the chosen two features, which are the colour of the superpixel under consideration, and the colour of a middle superpixel in the bottom row of the grid were also a part of a feature set used in the previous experiment. When it comes to features of neighbour colour percentages, two fewer features were chosen comparing to a noise-free case. Once again, a percentage of green neighbours of the superpixel

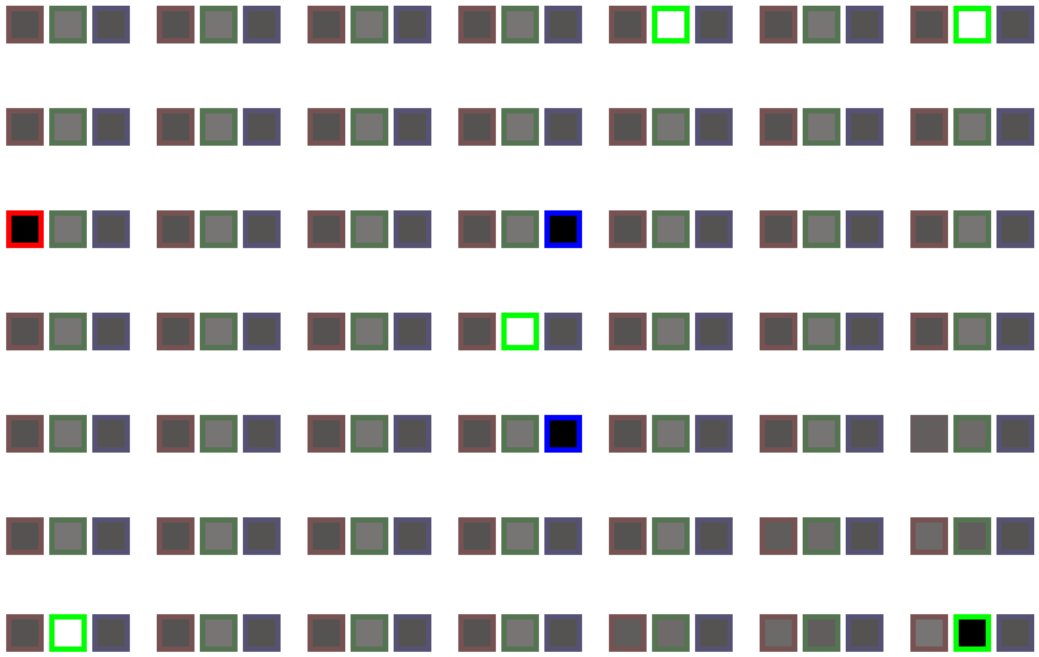


Figure 6.19: *Chosen grid neighbour colour percentage features for experiments on noised data.*

under consideration is taken into account. There are also two other features of green neighbours percentage that were chosen for both, noise-free and noised data.

In order to prove the importance of proper feature selection for semantic segmentation of noised data, figures 6.20 and 6.21 were prepared. Both figures show probabilities $p(y|x)$ computed for each label in the same form as it was in the previous section about semantic segmentation of noise-free images. On these figures the same 3 test samples are shown, however, the first one presents the results if all 196 features were used, and the second one if a set of 10 selected features only.

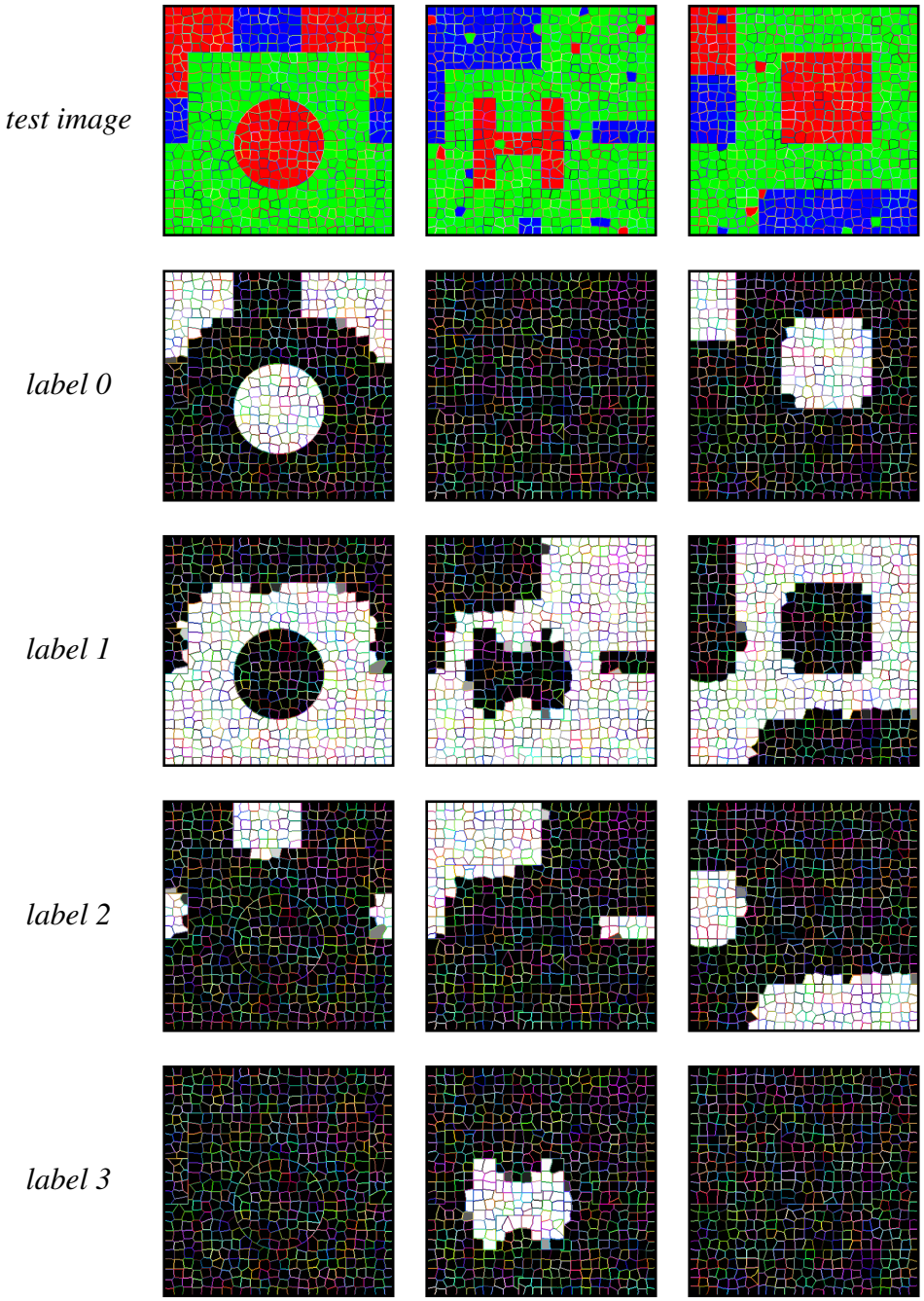


Figure 6.20: *Visual representation of conditional probabilities $p(y|x)$ for noised images without feature selection.*

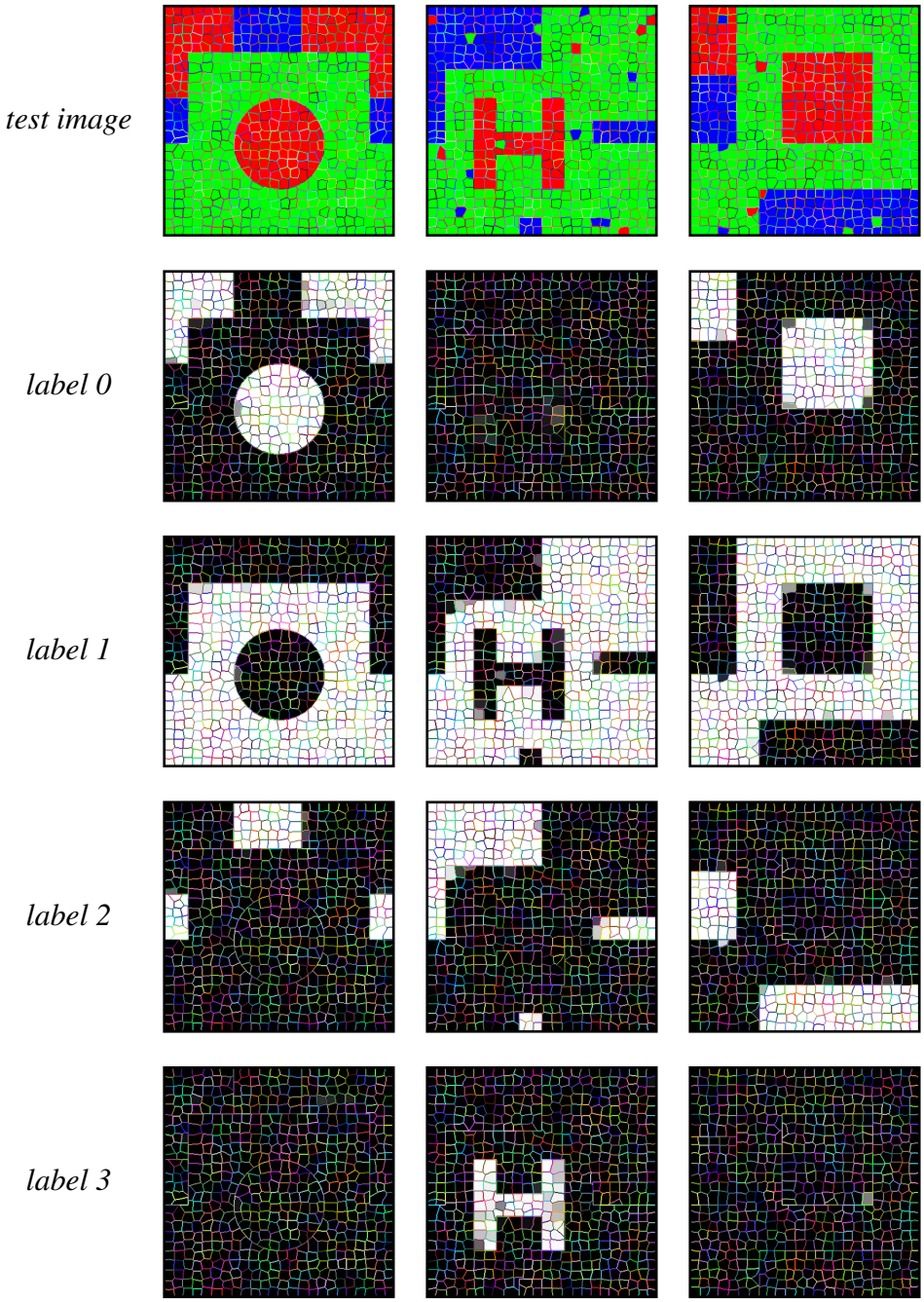


Figure 6.21: *Visual representation of conditional probabilities $p(y|x)$ for noised images after feature selection.*

As visible, for the set of all features, labellings of individual superpixels based on the computed probabilities $p(y|x)$ are much worse. Though classes of the objects are generally recognised correctly, the classification of superpixels on the boundaries between differently coloured objects is quite poor. This is especially visible on regions with label 3, as the shape of a letter H is more complex than that of a circle or a square and as a result, more superpixels are incorrectly labelled. This happens even on test images that do not contain any noise. If an original image was not shown, then it would be hard to say in what shape those objects are. On the other hand, using only a limited set of 10 selected features the number of incorrectly labelled superpixels is much smaller. Individual objects are clearly distinguishable and even the shape of a letter H for regions with 3$^{\text{rd}}$ class is sustained.

What is more, thanks to the presence of contextual features in a feature function definition, the unary potential has denoising abilities. If there is only one noised superpixel surrounded by neighbours of the same colour, the effect of the noise on the final labelling is diminished. This is especially visible in the second test sample presented in Figure 6.21. There are around 15 noised superpixels and for the majority of them the probability of the correct label is still close to 100%. Usually, the difficulty lies in a proper labelling of noised superpixels which are close to the boundary between different objects. However, even if the most probable label chosen by the unary potential is not the expected one, there is also a pairwise potential which aim is to cope with noises in an image basing on the colour similarities between superpixels of the original image.

6.3.3 Experimental results

With the use of the selected set of features parameter learning by means of Stochastic Gradient Descent took place. The resulting weights obtained by this process are presented in Table 6.4.

Table 6.4: *Trained weights for segmentation of noised images.*

w_1(unary potential)	$w_{2,1}$ (pairwise potential)	$w_{2,2}$ (bias)
0.14058	0.44002	0.16923

With the trained parameters of the model, the inference process was conducted on the testing set of unknown images and its results are presented in Figure 6.22. Similarly as in case of the experiment on noise-free data, there are five columns in this figure containing the original image, noised image after colour quantisation, results of local potential only and for both potentials, and finally the expected labelling.

In this experiment images after colour quantisation have some noised superpixels that are randomly placed over all image regions. It can be seen on images obtained from inference based on local potential only, that most of the noised areas are labelled correctly. This is an advantage of using the contextual data, as the final labelling is dependent on the number of different features and not only on the colour of a given superpixel. However, some noised regions, especially on the object boundaries would be incorrectly labelled if only local potential was used. This can be noticed for example in the 4^{th} row of Figure 6.22. Furthermore, due to the presence of noise in the training data, some superpixels from test images are incorrectly labelled even though they are not noised, which can be seen in 3^{rd} row on an

object with the shape of letter H. In both cases, by adding the pairwise potential the results of semantic segmentation improved.

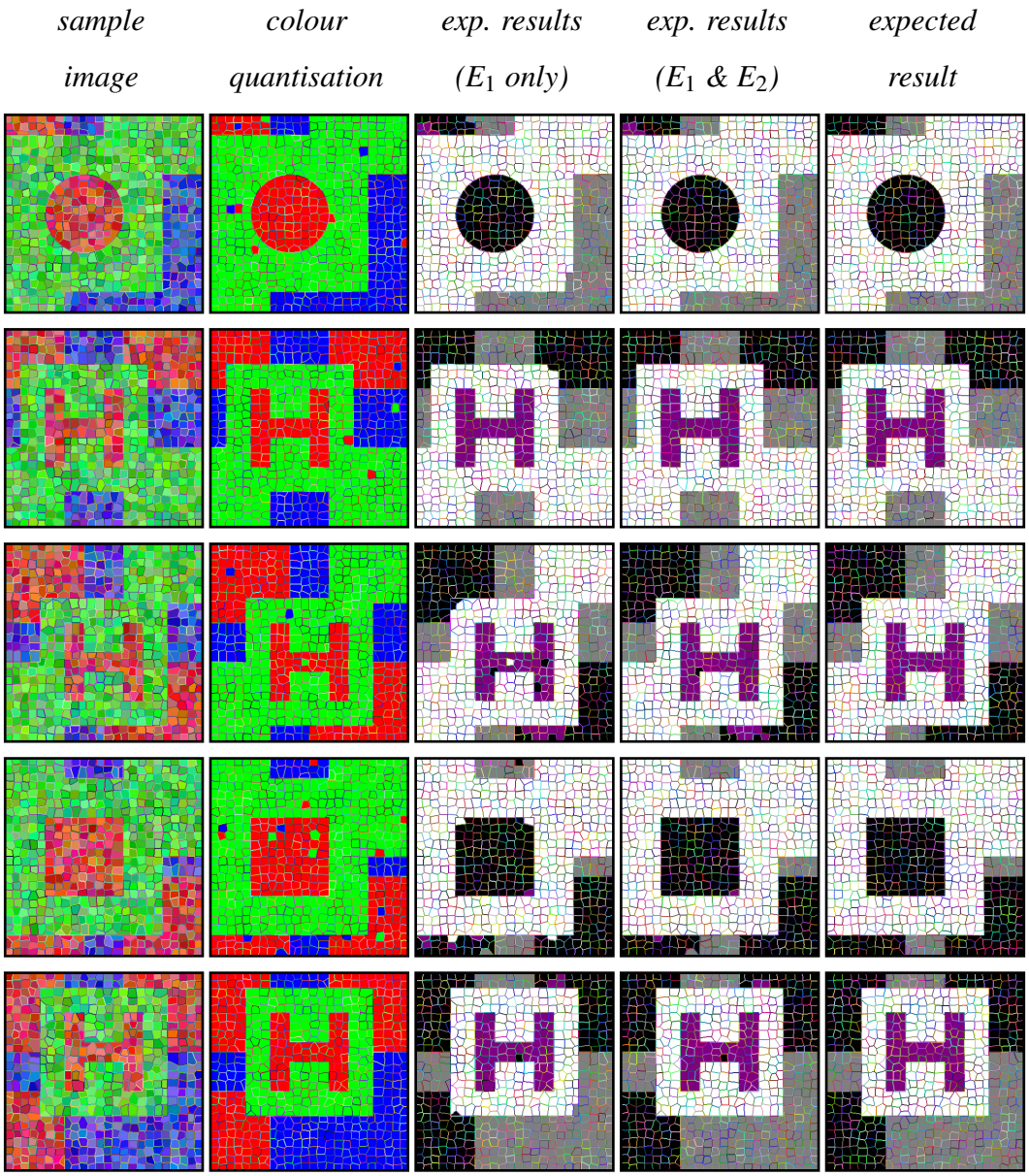


Figure 6.22: *Experimental results of shape-based semantic image segmentation on noised images.*

Due to this large effect of pairwise potential in this experiment, it is visible how the inference process gradually corrects the final labelling until the belief propagation reaches the state of convergence. Figure 6.23 depicts first five and, and the last five iterations of the inference process performed on a sample image. As shown, after the initial labelling there were five incorrectly classified superpixels, after the second iteration only two, and after the third, just one. It took 28 steps until the algorithm converged resulting in a correct labelling of all superpixels.

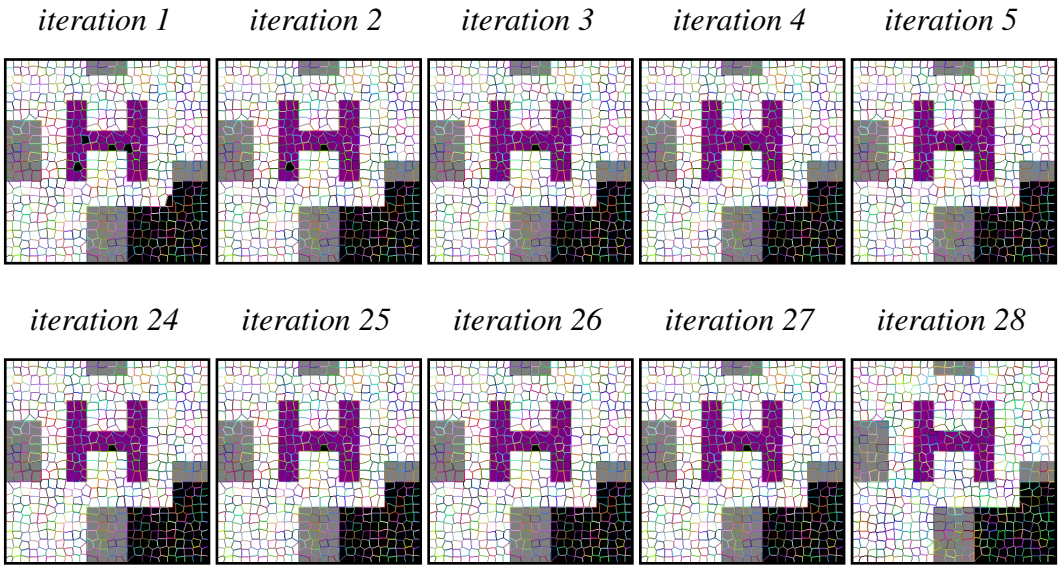


Figure 6.23: *Inference steps for a sample noised image.*

As in the case of the noise-free images, in this experiment there was also a slight problem with a distinction between label 0 and 3 for narrow and vertical, red stripes that are in the green neighbourhood. Furthermore, for some noised superpixels that were on the object or image boundaries even the pairwise potential did not manage to correct the wrong labelling obtained by the unary potential. However, the final accuracy of semantic segmentation on images with objects of different shapes

was still good. This accuracy expressed as Intersection over Union is presented in Table 6.5. Due to the addition of noise, it is slightly smaller than in the previous experiment. The method obtained a mean IoU of 98.39% for noised images, which is 1.15 percent points lower than in a noise-free case. Segmentation of green and blue regions was correct in 99.93% and 99.42% cases respectively. When it comes to red regions, it was 97.92% for label 0, and 96.27% for label 3, which is the lowest precision from all classes, just like in case of noise-free images. In this experiment, the addition of the pairwise potential improved the final segmentation results to a larger extent than in the previous one. The largest improvement was for objects with a shape of letter H as with the local potential only the accuracy of segmentation into class 3 had the IoU of 90.55%, which was improved by 5.72 percent points after addition of the pairwise potential. In overall, by using both potentials to define the energy function segmentation results were more precise by 2.41 percent point comparing to the classification based on local potential only.

Table 6.5: *Accuracy of segmentation based on shape detection for noised data.*

class	label 0	label 1	label 2	label 3	mIoU [%]
IoU [%] (E_1 only)	95.95	99.21	98.21	90.55	95.98
IoU [%] (E_1 & E_2)	97.92	99.93	99.42	96.27	98.39

Chapter 7

Discussion and conclusions

This chapter will include a summary of what was achieved during the thesis and the description on which steps and algorithms needed to be studied and implemented in order to achieve the task specified in the topic of this dissertation. Furthermore, a few possibilities of future expansion of the developed system and possible improvements will be described.

7.1 Thesis summary

The purpose of this thesis was to develop a system with which it will be possible to perform semantic image segmentation with the use of Conditional Random Fields. Before this system was created there was an excessive theoretical research conducted on what are the necessary steps and algorithms that may be used for this task. The created system was developed mostly from scratch and included an

implementation of the described algorithms. The dissertation was mostly based on a book *"Structured Learning and Prediction in Computer Vision"* by Sebastian Nowozin and Christoph H. Lampert and an article *"TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context"* written by Jamie Shotton et al.

In this dissertation two sets of experiments were described. The first one was aimed to perform semantic segmentation on images into three classes, one for objects in the shades of red, second one for bluish regions and the third one for greenish. In the second set of experiments, objects were to be detected also by their shape. There was one extra class introduced that represented reddish objects with a shape of letter H surrounded by greenish superpixels, which should be distinguished from other red regions in the same surroundings.

There were three main procedures required to achieve the task described in this dissertation. First of all, images had to be modelled in a way that makes further processing possible. A chosen method for this step was a factorisation process, which represented an image in terms of an undirected graph composed of two types of nodes, input nodes which decode image features, and output nodes, which are used to model the labelling of the system. Factor graphs are a perfect representation of Conditional Random Fields as they allow to model various types of features and contextual relations between image pixels, or superpixels. Features chosen to perform the experiments described in this dissertation were based solely on colour, though they included also contextual information about colours in the neighbourhood of the processed region. For the second type of experiments an automatic feature selection based on stepwise regression was implemented that allowed choosing an optimal set of features with which data from images was modelled. The next core

part of the developed system was aimed to train parameters of the established model using supervised machine learning methods based on the training set of images with known labellings. For this part Stochastic Gradient Descent algorithm was chosen. The last step concerned finding the optimal labelling for a set of previously unused test data in the process of inference. This was achieved by Loopy Belief Propagation, which is a message-passing algorithm used on graphical models to find such a sequence of states that has the largest probability of occurrence conditioned on known inputs. For semantic image segmentation this sequence is represented by a labelling of each superpixel of a given test image.

The first set of experiments described in this dissertation was aimed to present how the implemented algorithms work on a very basic example of image segmentation based on superpixel colour only. For images with only three colours available: red, green and blue, segmentation was correct in 100% cases for all labels. In the next experiment, in which shades of those colours were introduced, the results were much worse, however, they were largely improved by expressing the features in CIELAB colour space. In this experiment, it was clear that because of the contextual nature of Conditional Random Fields segmentation results are more precise, as only after incorporating the pairwise potential, which models relations between neighbouring superpixel, did the precision expressed in Intersection over Union reach 100%. Hence, for the first set of experiments, semantic image segmentation with use of Conditional Random Fields was successful.

Second part of the system was devoted to distinguishing objects in an image based not only by their colour but also by their shape. For this task, a different feature function had to be defined which included also contextual data of the given superpixel surroundings. In order to conduct the experiments proposed for this part of the

system, initial images were subjected to the process of colour quantisation, which limited available colours to three basic ones only. In the first experiment segmentation precision was equal to 99.54%, with most mislabelled superpixels being in the last class that should represent objects with a shape of a letter H. Chosen features turned out to be insufficient to determine the proper class of narrow, red regions at the image boundaries, which were too similar to vertical stripes in a letter H and therefore wrongly assigned to class 3 instead of 1. With few exceptions the algorithm had no problems with correct labelling of other regions. In the last experiment noise was introduced to the process of colour quantisation, and the aim of Conditional Random Fields was to provide an expected configuration of labels regardless of the noise. A resulting mean Intersection over Union for all labels was equal to 98.39%, and again class of objects shaped as a letter H had the lowest precision. However, in this experiment it was proved that proper definition of the feature function used for unary potential has a large influence on the final result. What is more, even without using the pairwise potential most of the noised superpixels were correctly labelled. The only problematic regions were on the boundaries of different objects, however, they effect of the noise in such regions was diminished when the pairwise potential was incorporated to the system.

In general, during this thesis different algorithms and methods were tested in each step of the system implementation. A few simplifications had to be introduced, especially during the training phase and estimation of features probability density, without which even a simple semantic segmentation task would require a lot of time and resources. However, the chosen algorithms even after simplifications managed fulfil tasks specified by experiments conducted in the thesis. Even with not complex features used, which were based only on superpixel colours, the possibilities of Conditional Random Fields were visible. By incorporating contextual data to

the model, which were based only on colours as well, it was possible to distinguish red objects on green neighbourhood that were differing only by their shape, which shows how powerful Conditional Random Fields may be once based on more complex features.

7.2 Improvements perspective

As in every machine learning task that was more focused on checking whether the method may be applied to a specific problem rather than its results, a lot of improvement could be obtained by increasing the number of training and validation data and making it more diverse. This could for example cope with the problem of distinguishing the vertical stripe of the letter H with other narrow, red regions. Moreover, as in the system there is a lot of hyperparameters that influence different steps of the segmentation task only a few configurations of those parameters were tested, simply because it would take a lot of time to check them all. Hence, another possible improvement would be to perform an extensive search of optimal values of the hyperparamter set used in the developed system.

The thesis was mostly focused on choosing the right algorithms that will allow to perform a semantic image segmentation without a need of extensive resources. As feature engineering is on itself a demanding and time-consuming task, in the thesis it was highly limited. Therefore, a lot of improvement can be achieved if proper features would be incorporated to the system. Basing on colour information only, it is not possible to perform semantic segmentation on natural images, as natural objects are usually not colouristically homogeneous as well as colours of objects of the same class tend to vary. Hence, the first possible extension to the system

would be to increase a feature space by incorporating more complex features than just pixel colours.

Another possible extension of the thesis would be to use the developed system only as the postpocessing stage, after classification with some other algorithm takes place. Then, the initially segmented image obtain with this classification would become an input of Conditional Random Fields just like an image after colour quantisation, which was used in the thesis. State-of-the art systems are using Convolutional Neural Networks to define which regions should belong to which class, and Conditional Random Fields only to make those results better at object boundaries and to reduce noise being a side effect of initial classification. Hence, in the future work it may be tested whether the developed system would improve the segmentation results obtained by Convolutional Neural Networks or any other classifier.

Bibliography

[1] A. C. Müller and S. Guido. “Introduction to Machine Learning with Python”. In: O’Reilly Media, 2017. Chap. Introduction. ISBN: 978-1-449-36941-5.

[2] R. C Gonzalez and R. E Woods. “Digital Image Processing (2nd Edition)”. In: New Jersey. Prentice Hall, Jan. 2002. Chap. Image Segmentation. ISBN: 0201180758.

[3] C. A. Glasbey and G. W. Horgan. *Image Analysis for the Biological Sciences*. 1995. ISBN: 0-471-93726-6.

[4] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. "http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html". [Online; accessed 19.06.2018].

[5] B. Fan, Z. Wang, and F. Wu. “Local Image Descriptor: Modern Approaches”. In: Jan. 2015. Chap. Classical Local Descriptors.

[6] T. K. Ho. “Random Decision Forests”. In: *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*. ICDAR ’95. IEEE Computer Society, 1995, pp. 278–. ISBN: 0-8186-7128-9.

[7] C. Cortes and V. Vapnik. “Support-Vector Networks”. In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 1573-0565.

[8] D. Ciresan et al. “Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images”. In: *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 2843–2851.

[9] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, pp. 3431–3440.

[10] L. Chen et al. “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs”. In: *CoRR* abs/1412.7062 (2014).

[11] Y. Altun, I. Tsochantaridis, and T. Hofmann. “Hidden Markov Support Vector Machines”. In: *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. ICML’03. AAAI Press, 2003, pp. 3–10. ISBN: 1-57735-189-4.

[12] Y. Liu, E. P. Xing, and J. Carbonell. “Predicting Protein Folds with Structural Repeats Using a Chain Graph Model”. In: *Proceedings of the 22Nd International Conference on Machine Learning*. ICML ’05. ACM, 2005, pp. 513–520. ISBN: 1-59593-180-5.

[13] A. Torralba, K. Murphy, and W. T. Freeman. “Contextual models for object detection using boosted random fields”. In: *NIPS* (Oct. 2004).

[14] J. Pearl. “Causality: Models, reasoning, and inference, second edition”. In: *Causality* 29 (Jan. 2000).

[15] P. Pardalos and J. Xue. “The Maximum Clique Problem”. In: *Journal of Global Optimization* 4 (Apr. 1994), pp. 301–328.

[16] S. P. Chopra. “Factor Graphs for Relational Regression”. PhD thesis. 2008. ISBN: 978-1-109-90255-6.

[17] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML ’01. Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. ISBN: 1-55860-778-1.

[18] C. Sutton and A. McCallum. “An Introduction to Conditional Random Fields”. In: *Found. Trends Mach. Learn.* 4.4 (Apr. 2012), pp. 267–373. ISSN: 1935-8237.

[19] P. Krähenbühl and V. Koltun. “Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials”. In: *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2011, pp. 109–117.

[20] A. Blake, P. Kohli, and C. Rother. *Markov Random Fields for Vision and Image Processing*. The MIT Press, 2011. ISBN: 9780262015776.

[21] J. Pearl. “Reverend Bayes on inference engines: a distributed hierarchical approach”. In: *in Proceedings of the National Conference on Artificial Intelligence*. 1982, pp. 133–136.

[22] S. Nowozin and C. H. Lampert. “Structured Learning and Prediction in Computer Vision”. In: *Found. Trends. Comput. Graph. Vis.* 6 (Mar. 2011), pp. 185–365. ISSN: 1572-2740.

[23] J. S. Yedidia, W. T. Freeman, and Y. Weiss. “Exploring Artificial Intelligence in the New Millennium”. In: Morgan Kaufmann Publishers Inc., 2003. Chap. Understanding Belief Propagation and Its Generalizations, pp. 239–269. ISBN: 1-55860-811-7.

[24] B. Kim, A. Yedla, and H. D. Pfister. "Message-Passing Inference on a Factor Graph for Collaborative Filtering". In: *CoRR* abs/1004.1003 (2010).

[25] C. Wang, N. Komodakis, and N. Paragios. "Markov Random Field Modeling, Inference & Learning in Computer Vision & Image Understanding: A Survey". In: *Computer Vision and Image Understanding (CVIU)* 117.11 (Nov. 2013), pp. 1610–1627.

[26] Q. V. Le et al. "On Optimization Methods for Deep Learning". In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML'11. Omnipress, 2011, pp. 265–272. ISBN: 978-1-4503-0619-5.

[27] F. E. Curtis and K. Scheinberg. "Optimization Methods for Supervised Machine Learning: From Linear Models to Deep Learning". In: *The Operations Research Revolution*. 2017, pp. 89–113.

[28] P. M. Lee. "The Gibbs sampler and other numerical methods". In: *Bayesian Statistics: An Introduction*. John Wiley & Sons, 2012, p. 281. ISBN: 9781118359778.

[29] A. Zheng and A. Casari. "Feature Engineering for Machine Learning. Principles and Techniques for Data Scientists". In: O'Reilly Media, Apr. 2018. Chap. The Machine Learning Pipeline, pp. 1–5. ISBN: 978-1-491-95324-2.

[30] *AutoML: Automatic Machine Learning*. `http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html/`. [Online; accessed 16.01.2019].

[31] *TPOT: Data Science Assistant*. `https://epistasislab.github.io/tpot/`. [Online; accessed 16.01.2019].

[32] *auto-sklearn*. `https://automl.github.io/auto-sklearn/stable/`. [Online; accessed 16.01.2019].

[33] U. Khurana et al. "Cognito: Automated Feature Engineering for Supervised Learning". In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. Dec. 2016, pp. 1304–1307.

[34] M. Nixon and A. S. Aguado. *Feature Extraction & Image Processing for Computer Vision, Third Edition*. 3rd. Academic Press, Inc., 2012. ISBN: 9780123965493.

[35] A. Saglam and N. A. Baykan. "Effects of color spaces and distance norms on graph-based image segmentation". In: *2017 3rd International Conference on Frontiers of Signal Processing (ICFSP)*. Sept. 2017, pp. 130–135.

[36] S. Nowozin et al. "Decision tree fields". In: *2011 International Conference on Computer Vision*. Nov. 2011, pp. 1668–1675.

[37] G. Lin et al. "Efficient piecewise training of deep structured models for semantic segmentation". In: *CoRR* abs/1504.01013 (2015).

[38] A. Kirillov et al. "A Generic CNN-CRF Model for Semantic Segmentation". In: (Nov. 2015).

[39] D. R. Anderson. "Continuous Probability Distribution". In: *Modern Business Statistics*. Oct. 2017. ISBN: 9781538803660.

[40] P. Refaeilzadeh, L. Tang, and H. Liu. "Cross-Validation". In: *Encyclopedia of Database Systems* 532–538 (Jan. 2009), pp. 532–538.

[41] M. Hao et al. "A contrast-sensitive Potts model custom-designed for change detection". In: *European Journal of Remote Sensing* 47.1 (2014), pp. 643–654.

[42] J. Shotton et al. “TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context”. In: *International Journal of Computer Vision* 81.1 (Jan. 2009), pp. 2–23. ISSN: 1573-1405.

[43] M. Pereyra et al. “Estimating the Granularity Coefficient of a Potts-Markov Random Field Within a Markov Chain Monte Carlo Algorithm”. In: *IEEE Transactions on Image Processing* 22.6 (June 2013), pp. 2385–2397. ISSN: 1057-7149.

[44] *TIOBE Index for January 2019*. `https://www.tiobe.com/tiobe-index/`. [Online; accessed 31.01.2019].

[45] P. Neubert. “Superpixels and their Application for Visual Place Recognition in Changing Environments”. PhD thesis. 2015.

[46] F. a. p. petitcolas. *Public-Domain Test Images for Homeworks and Projects"*. `https://homepages.cae.wisc.edu/~ece533/images/`. [Online; accessed 29.01.2019].

[47] R. Achanta et al. “SLIC Superpixels Compared to State-of-the-Art Superpixel Methods”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (Nov. 2012), pp. 2274–2282. ISSN: 0162-8828.

[48] G. Csurka, D. Larlus, and F. Perronnin. “What is a good evaluation measure for semantic segmentation?” In: *BMVC*. 2013.

List of Figures

List of Tables