CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY

ASSIGNMENT OF BACHELOR’S THESIS

Title:	Color recognition of painted parts in automobile production
Student:	Dominik Dragoun
Supervisor:	Ing. Mikuláš Krupička
Study Programme:	Informatics
Study Branch:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	until the end of summer semester 2015/16

Instructions

Examine the color distinctiveness of painted parts from the automobile production photographed using a USB camera. Find appropriate features that can be extracted from camera images. Determine the optimal color of the illumination and determine which light components have influence on the measurement. Use a multilayer perceptron neural network as a classifier for the color recognition and find the optimal number of its layers, inputs and outputs. Design a software solving this task using the C++ programming language. Compare different numbers of layers of the neural network, the influence of neural network inputs (features obtained from the camera image), and colors of the illumination on the overall result.

References

Will be provided by the supervisor.

L.S.

doc.Ing. Jan Janoušek, Ph.D.
Head of the department

prof.Ing. Pavel Tvrdík, CSc.
Dean

Prague January 31, 2015

Czech Technical University in Prague

Faculty of Information Technology

Department of Theoretical Computer Science

Bachelor's thesis

Color recognition of painted parts in automobile production

Dominik Dragoun

Supervisor: Ing. Mikuláš Krupička

11th May 2015

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Ing. Mikuláš Krupička, for his continued support throughout my thesis.

I would also like to extend my sincere appreciation to the company Lithium a.s., especially it's director Mgr. Vladimír Řeháček and also to the company WITTE Nejdek s.r.o., especially to Dipl. Ing. Aleš Blažek, the head of the software development department, for the opportunity to work on this matter.

Special thanks to my friend Ondřej Ezr for his friendship throughout my studies and to my beloved one for being there for me.

Last but not least, my deepest gratitude is also extended to my beloved family for giving me the opportunity to follow my dreams and the love to make them a reality.

Declaration

Citation of this thesis

Dragoun, Dominik. *Color recognition of painted parts in automobile production.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2015.

Abstrakt

Tato práce se zabývá rozpoznatelností barev lakovaných částí z automobilového průmyslu. Vysvětluje některé metody zpracování obrazu a strojového učení. Zkoumá vliv barevného osvětlení a jednotlivých metod zpracování obrazu na schopnost naučení vícevrstvé perceptronové neuronové sítě k účelům rozpoznávání barev laků a jejich vzájemného rozlišení. Součástí této práce je také implementace aplikace v jazyce C++, která umožňuje uživateli extrakci dat z fotografií laků.

Klíčová slova rozpoznávání barev, automobilový průmysl, zpracování obrazu, strojové učení

Abstract

This thesis focuses on the color distinctiveness of painted parts in the automotive industry. It explains some methods of image processing and machine learning. It examines the influence of colored illumination and various image processing methods on the computer's ability to learn a multilayer perception neural network in order to detect the colors of the paint and their mutual distinction. Also, the implementation of an application in the C++ programming language, which enables the user to extract data from photographs of paints, is part of this thesis.

Keywords color recognition, automotive, image processing, machine learning

Contents

List of Figures

List of Tables

Introduction

Due to demand of high quality products, quality assurance evolved into strict, therefore expensive processes. Employers are forced to hire employees specifically for these positions and due to high expectations of customers those employees need to be trusted and highly qualified.

In industry with high volume production, a lot of quality checks need to be done. Therefore companies have to increase the number of these employees or scale down the number of these checks. Because the first method is usually very expensive, companies tend to go the other way, doing checks only on selected products, i.e. one product in a shipping box or few random checks in produced volume.

To make sure that the check was done thoroughly, employees doing them must be requalified very often or expensive quality testing machines have to be bought. Because hardware solutions are quite expensive and need frequent recalibrations, companies need to find compromises between those to maintain both their reputation and earnings.

Another way appeared with the evolution of computer science and recognition solutions. Precisely done software using specific algorithms can reduce the number of highly qualified employees and their frequent requalification as well as the need for expensive hardware.

Motivation

In the automotive industry, car manufacturers (customers) want the best quality products from the suppliers. WITTE Automotive is a globally operating company supplying all well-known vehicle brands with their products – latches, hinges, lock sets and door handles. Their products are used in tens of thousands of cars, thus they have to produce them in top quality. This involves the efficiency problems described above.

WITTE Automotive experiences such efficiency problems in their production of painted parts. Those parts have to be checked both for manufacturing and for paint defects. In this thesis I will be discussing an alternative solution to ones currently applied by WITTE Automotive to the problem of paint defects (see Chapter Solutions). This solution, if successful enough, will be extremely useful for this company and it will save it a significant amount of time and money.

Thesis objectives

The main objective of this thesis is to determine the extent to which painted parts can be recognized with the usage of image processing and machine learning algorithms (see Chapter Conclusion).

A further objective is to find useful image processing and other information gaining methods that can help improve the recognition rate of a model trained via machine learning (see Chapter Image processing and analysis).

Another objective is to find the best possible settings for the recognition – optimal number of layers of a neural network and their sizes, optimal color of the illumination for image capturing and which light components have influence on the measurement (see Chapter Experiments).

The last objective is to create a program in the C++ programming language, that is capable of capturing images with a USB camera, capable of processing those images to gain information and capable of creating a dataset from the information gained from these images with correct labels for the machine learning (see Subsection Hardware and software for image capturing and Subsection Algorithms for image processing).

Chapter 1

Color models

For a human, an image is one of the simplest representations of information. Therefore text written on a piece of paper, a photograph made through old photographic processing or a digital image displayed on a monitor of a personal computer is recognisable easily using photoreceptor cells in our eyes and then processed in our brain to classify this information. This is not that easy when it comes to computers. For doing anything with an image it first needs to save it to its memory. The way the image is represented in a computer is dependent on the intended use of that image. I will introduce image representation using color models – abstract mathematical models describing the colors as numbers.

1.1 RGB color model

Arguably the most common color model, RGB, is used for storing an image captured with digital camera and for displaying it on an electronic visual display such as a computer monitor or a phone display. This is so because of the way they work, in general, they are made of phosphor dots (CRT displays) or color segments (LCDs and plasma displays) emitting red, green and blue light depending on the color they want to display [1].

RGB is an additive color model, which combines three primary colors (red, green and blue) to produce other colors. It is based on how photoreceptor cells in the human eye work. Those cells are sensitive to certain wavelengths which are associated with colors [1]. Those three colors are chosen in a way, that each is stimulating one of the three types of the eye's color receptors while stimulating those other two as little as possible.

It's storage in computers is quite simple, each pixel (single unit of an image containing its color values) is in an RGB color space made of three numbers – intensity of red, intensity of green and intensity of blue light, called channels, which the computer should emit on the pixel position to display its color correctly. Thus an image can be simply stored as resolution of the image (its width and height) and a two dimensional array of these three color values.

The RGB color space can be described as a unit cube, where each axis is the intensity of each primary color (see Figure 1.1). For example in a 24-bit RGB image each channel is an 8-bit integer, therefore each channel contains discrete values in the range from 0 to 255.

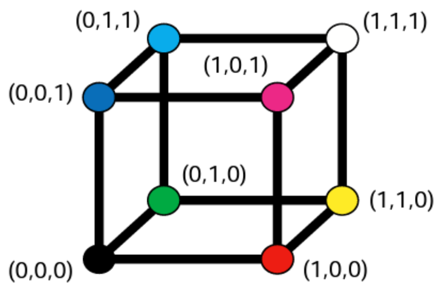


Figure 1.1: The RGB color cube, with each vertex labeled with its corresponding RGB triple [1].

1.2 HSV

HSV representation of color is more intuitive for humans. It is created through the transformation of the RGB cube into a hexagon (see Figure 1.2). The HSV has three components (channels) – hue (meaning color in the visible color spectrum), saturation (meaning intensity of the color) and value (meaning brightness of the color, therefore sometimes HSV is called HSB, which substitutes value for brightness in the name).

Then it could be extended into a cylinder (see Figure 1.3). A rectangle with one side equal to the height of this cylinder and another side equal to its radius is a sheet of one color. The hue channel in the HSV image spans from 0° to 360°. Saturation and value channels spans from 0 to 1.

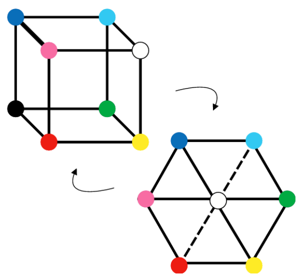


Figure 1.2: The projection of the RGB color cube is a hexagonal "hue circle" when black and white are aligned [1].

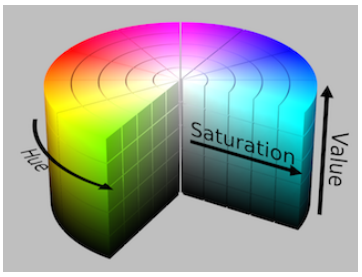


Figure 1.3: A HSV cylinder with the three components shown [2].

1.3 Grayscale

Pixels in a grayscale digital image do not contain color intensities, it is made of single channel pixels carrying only luminous intensity, which is the measure of the wavelength-weighted power emitted by a light source. When displayed, it only contains shades of gray color, and is sometimes called black-and-white image, which is in fact wrong.

As described above, receptors in a human eye are sensitive to specific wavelengths. In addition, their sensitivity differs for each wavelength, so some colors are more visible to us than others. This should be taken into account in a conversion from a colored image to the grayscale. There are many ways how to do that and in my work I used a ITU-R BT.709-5 recommendation (see Figure 1.4)[3].

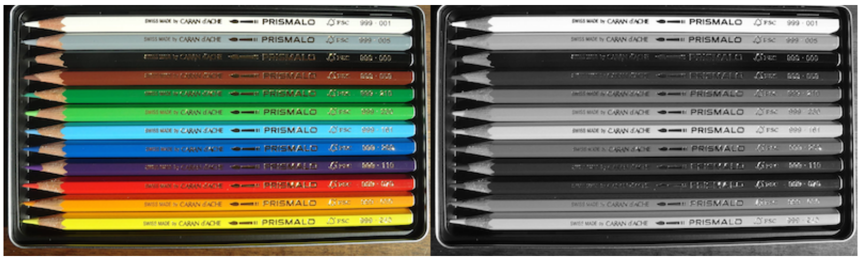


Figure 1.4: Image of colored pencils before (left) and after (right) grayscale conversion [4].

Chapter 2

Image processing and analysis

The image processing and the image analysis are studies of any algorithm that takes an image as input and returns either an image or a set of characteristics or parameters related to the image [5]. Those output parameters could be used as inputs for machine learning algorithms in order to teach a model (see Chapter Machine learning). In this chapter, I will introduce image processing techniques I was experimenting with.

2.1 Color histograms

Each pixel in an image contains values of each channel, depending on the image type. An image histogram of a certain channel is a graphical representation of the frequency of occurrence of each of the permitted values in the given image and the selected channel (see Figure 2.1).

If we normalize this histogram in a way that the sum of all frequencies is one, we get a discrete probability density function defining appearance probability of each channel value in the given image.

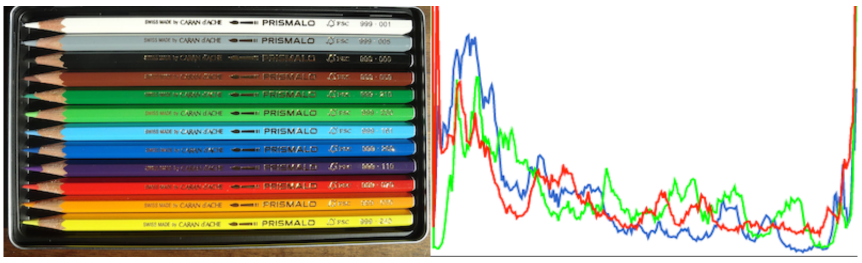


Figure 2.1: Image of colored pencils with its RGB histogram [4].

2.1.1 Data binning

Data binning is a technique used to group certain values into intervals. This technique can be used for both continuous and discrete data and there are several ways how to choose sizes of those bins [6, 7]. Binning in this work refers to the equal-width binning technique, which groups values into a certain number of bins with exactly same size (i.e. 0–9, 10–19 and 20–29).

Histograms of color values are useful to visualise the probability of each value in a picture. However for further analysis a high amount of value types are inconvenient (i.e. 256 values for channels of the RGB). Binning helps to reduce cardinality of data and can increase the predictive power of the variable [7].

2.2 Image segmentation

Image segmentation is a generic process by which an image is subdivided into its constituent regions or objects. After the segmentation is done, each region is then labeled for further analysis. The segmented objects are often termed the foreground and the rest of the image is the background. Each label corresponds to certain segments which have a certain degree of mutual similarity, i.e. color or shape [8].

2.2.1 Thresholding

Thresholding is a simple form of image segmentation, which transforms a color or grayscale image into a binary image (an image made of pixels containing only values 0 or 1), depending on whether its intensity is above or below the threshold value [8]. Because of its simplicity, no labels are required. White pixels of the resulting binary image could be the foreground and black pixels the background or vice versa.

Thresholding is usually used to separate certain regions within the image based upon its pixel values. This could be useful for detecting interesting regions in the image such as background or foreground objects (see Figure 2.2).

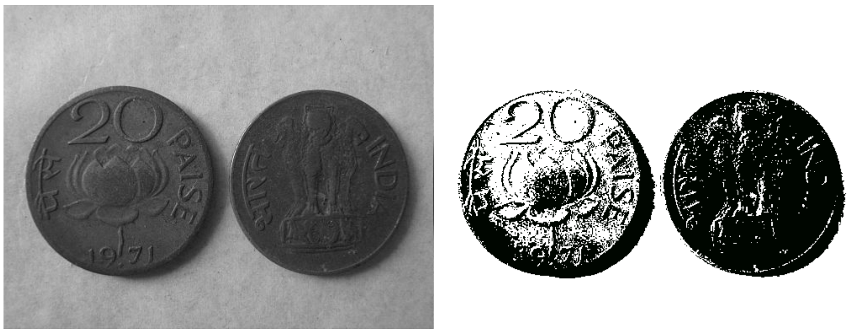


Figure 2.2: Image of coins before (left) and after (right) transformation into binary image [9]. Pixels in original image with intensity lower than 70 are black, others are white.

2.3 Dilation and mathematical morphology

The aim of mathematical morphology is to define the structure of an object by its geometrical structure, thus distinguishing itself from other image processing theories (their goal is to describe objects in the picture in accordance with the judgement of the human eye) [10, 11].

In mathematical morphology, dilation is a shape operator derived from Minkowski addition [10]. Its goal is to increase the volume of the given object by coloring adjacent pixels with its color. In binary images, this could be used to join regions which are close to each other but should be connected (see Figure 2.3). Thus if used cautiously it could remove errors made in previous transformations, otherwise it could connect unrelated regions and corrupt the image.

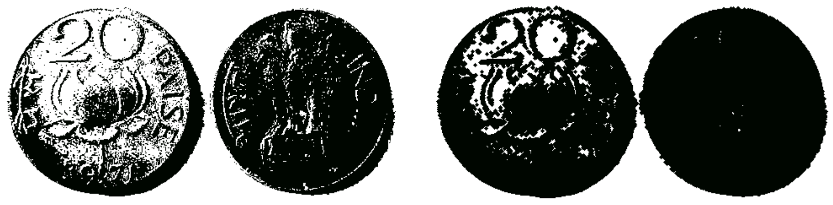


Figure 2.3: Binary image of coins before (left) and after (right) dilation of each black region by 3 pixels [9].

2.4 Local binary patterns

The local binary pattern (LBP) operator is defined as a grayscale invariant texture measure using the general definition of texture in a local neighbourhood. It is a unifying texture model that describes the structure of a texture micro-textons and their statistical distribution rules [12].

LPB operator labels a pixel in a grayscale image depending on its value and value of its neighbours'. The basic version of LBP takes a pixel and remembers its value, then it compares it to each of its 8 neighbours' values in a circle (see Figure 2.4). If the neighbour's value is brighter it writes "1", otherwise it writes "0". This creates an 8-bit number which is then assigned to the current pixels position in a LBP matrix (usually as a decimal number).

A histogram can be constructed using this LBP matrix. If this histogram is normalized and the sum of all its values is one, it describes a discrete probability density function defining probability of each pattern in the given image.

There are 256 different patterns (variations of 8 bits – $2^8 = 256$). Extension to the original operator is the so-called uniform patterns, used to reduce the length

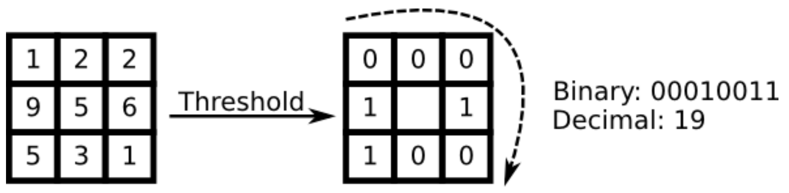


Figure 2.4: Transformation of one pixel and its neighbourhood into LBP [13].

of the feature vector and to implement a simple rotation invariant descriptor. The uniform local binary pattern (ULBP) is a pattern where the number of bitwise transitions from zero to one, or vice versa, when the bit pattern is considered circular, is two at most. This reduces the number of different patterns to 59 [12, 14].

Chapter 3

Machine learning

Machine learning is a field in computer science which addresses the construction and study of algorithms that uses data to learn from them and make predictions on new data provided to them. This is useful for a broad set of fields from search engines, stock market analysis and fraud detection to agriculture, bioinformatics and medical diagnoses [15].

There are three main types of machine learning – unsupervised learning, reinforcement learning and supervised learning which are distinguished depending on the type of the problem to solve [16].

3.1 Unsupervised learning

In unsupervised learning, the program is given set of input data without any prior knowledge about them. Its goal is to find hidden patterns in the data and use those patterns to decide or predict on new data. One of unsupervised learning tasks is clustering problems. The program is given a set of input data in which it tries to find potentially useful clusters [16].

For example if given data of traffic, it could develop a concept of fast paths and slow paths for future direction of cars without ever being given labeled examples of each by a teacher.

3.2 Reinforcement learning

In reinforcement learning, the program is getting experience from past actions. This is done by getting a series of reinforcements – rewards and punishments for its previous actions. Actual labels for data are still not given to the program, but it is rewarded for successful actions and punished for unsuccessful ones [16].

An example could be playing a chess game. If the program wins, it is rewarded and when it looses, it is punished. It is then another task for the program to decide, which actions were the ones that made its game successful or unsuccessful in future games, to increase its probability to win.

3.3 Supervised learning

Supervised learning is a method where a teacher gives the program labeled data on which the program is trained. This is done in several ways depending on the learning algorithm chosen. After the training is done, the program is then able to label new unlabeled data [16].

For example, the program is given images (or processed information from them) of characters from the alphabet with each labeled with the depicted character. After training is done, the program is then given an image of a character without a label and is able to classify it and give it the correct label.

3.4 Approaches and algorithms

There are a lot of approaches to machine learning [17]. I will be mainly talking about artificial neural networks which is the approach I used in my research.

3.4.1 Artificial neural network

An artificial neural network (ANN) is a statistical learning algorithm based on the function of a biological neural network. ANNs are made of artificial neurons, which are interconnected via a set of weights (analogous to synaptic connections in the nervous system), allowing signals to travel through the network both in parallel and serially. In the brain, neurons work like a switch, when enough neurotransmitter has accumulated in the cell body, an action potential is generated. This is modelled as a weighted sum of all incoming signals to the artificial neuron, which is compared with a threshold [18].

Computational power in an ANN is derived from the density and complexity of the interconnections, rather than complexity of each processing unit as in the conventional computer. Another difference between ANN and conventional computer is that ANNs have their memory distributed throughout its structure, in the weights and is modified by experience [19].

3.4.2 Multilayer perceptron network

One of the most used and studied networks is the multilayer perceptron [20]. The structure of the multilayer perceptron network (MPN) consists of layers of nodes, where each node is connected with a weight to every node in the following layer (see Figure The structure of a MPN with one hidden layer [21]). There is an input layer, one or more hidden layers and an output layer. Each node in the input layer is an input feature provided for learning and the number of inputs determine the number of nodes in this layer. Nodes in the hidden and output layer are artificial neurons (AN). The count of the nodes in the hidden layer is chosen by the architect of the network and the count of nodes in the output layer is defined by the number of output classes.

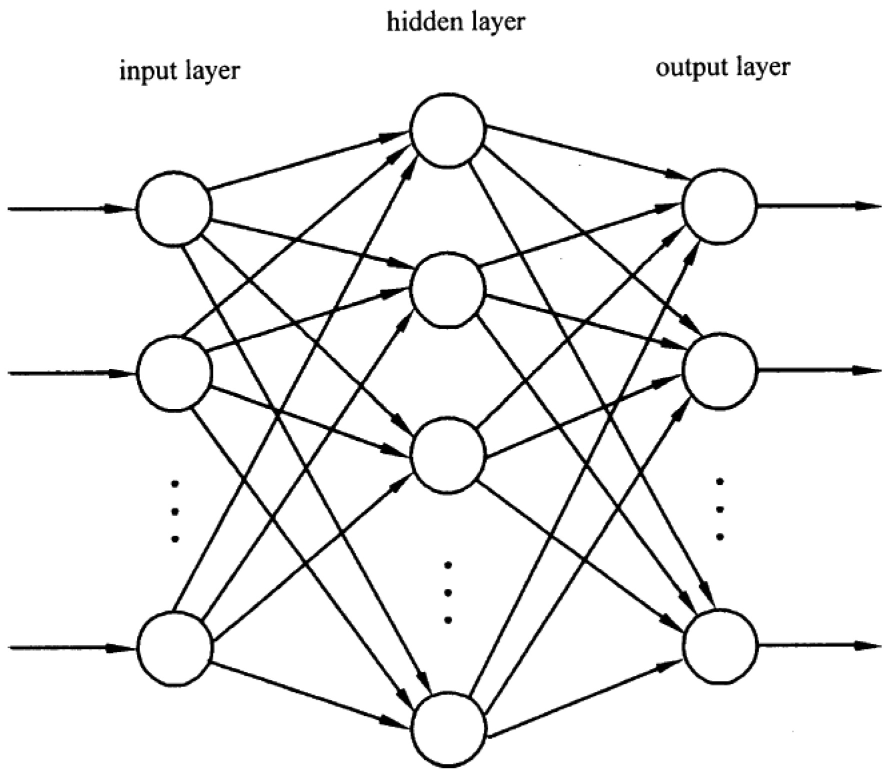


Figure 3.1: The structure of a MPN with one hidden layer [21].

Learning in the MPN consists of two phases, a forward and a backward phase. In the feedforward phase, an AN takes the input weights, uses them in its activation function and sends its output to each node in the following layer, layer by layer. If outputs are not as desired, errors are propagated backward from the output neurons. This is done recursively until the selected error criterion is satisfied [21].

Chapter 4

Solutions

There are more ways to assure paint quality on the painted parts. I will describe two solutions that are currently applied in WITTE Automotive and the solution I am currently working on.

4.1 Manual recognition

One of the two solutions to this problem is manual recognition of the painted parts with qualified employees. This solution is probably the most accurate one, because only humans are able to check a product for all defects with very little error. This is due to the fact, that the human brain is very powerful in the recognition and learning processes. It takes mere days for a human to be able to successfully recognize paint defects with guidance of another qualified employee, using magnifying glasses or similar tools and a non-defective exemplar.

This solution also brings problems closely associated with medical and trust issues with humans. The human eye is a very complex optical system capable of detecting the slightest defects on an object. As much as it is complex, it is sensitive to pathogenic influence. Thus these employees need to be examined often and thoroughly by the ophthalmologist (specialist in medical and surgical eye problems). This is both expensive and time-consuming and that is the main reason for employers to look for alternative solutions. Another problem is that the employee must be trustworthy enough because rejection of a whole batch of products because of paint defects would be very expensive.

4.2 Color sensors

An alternative solution is the use of special color sensors capable of distinguishing slight changes in the colors. This solution brings some advantages and some disadvantages to manual recognition. The biggest advantage, in my opinion, is that the recognition could be integrated into the production line. This can eliminate the time and money costs for frequent medical examinations. It can also check many times more products than a human is capable of and it is also able to do those checks continuously which also increases the amount of checked products.

This would be a great improvement over manual recognition, if it was not for its disadvantages – those sensors are very expensive, the price of each sensor is on the order of thousands of euros. Another problem with them is that they need recalibration with the slightest change in the paint batch. Also, they are quite sensitive to changes in illumination, resulting in incorrect results.

4.3 Machine learning

Paint recognition can also be achieved by using an inexpensive usb camera, a computer and few sophisticated algorithms combined together. When enough information is gained from the images of the painted parts using image processing algorithms, then a model can be trained with machine learning algorithms using this information. That creates a solution which can be used in the production line, just like the RGB sensors. Because it is set up specially for this task, it can overcome problems that the RGB sensors have with the paint batch changes and it is not as expensive as the two previous solutions.

4.3.1 Hardware and software for image capturing

For the image capturing, I am using a Conrad USB Digital Microscope, which has a 10x to 200x magnification range, that allows capturing images of the painted parts with high detail even though it is only able to capture them in a maximal resolution of 1600x1200 pixels (2.0 Megapixels) [22]. In fact, for the purposes of the image processing, I am using much lower resolution images – 640x480 pixels, which is still detailed enough and is processed more than six times faster than the images with the resolution of 1600x1200 pixels. This camera costs only about 70 euros, which is significantly less than the mentioned RGB sensors. In the application, I am using an open source computer vision library OpenCV, specifically its C++ interface [23].

4.3.2 Algorithms for image processing

Most of the algorithms, used in my image processing application, are also available in the OpenCV library. The reason I chose to implement them on my own is that to gain all the information from the image, they have to go through the matrix of its pixels several times. Thus usage of the OpenCV methods would need to loop though this matrix several times, one algorithm after another. In my implementation I can do most of the information gaining in one or two loops through the image – form of loop fusion technique, and it can also speed up the program because it will be much more prepared for pipelining. This reduces the branching of compiled code and makes it easier for the compiler to optimize the program upon compilation. When combined with other compiler optimizations like loop unrolling, caching and others, it could speed up the application up to several times (see Chapter Conclusion).

Chapter 5

Experiments

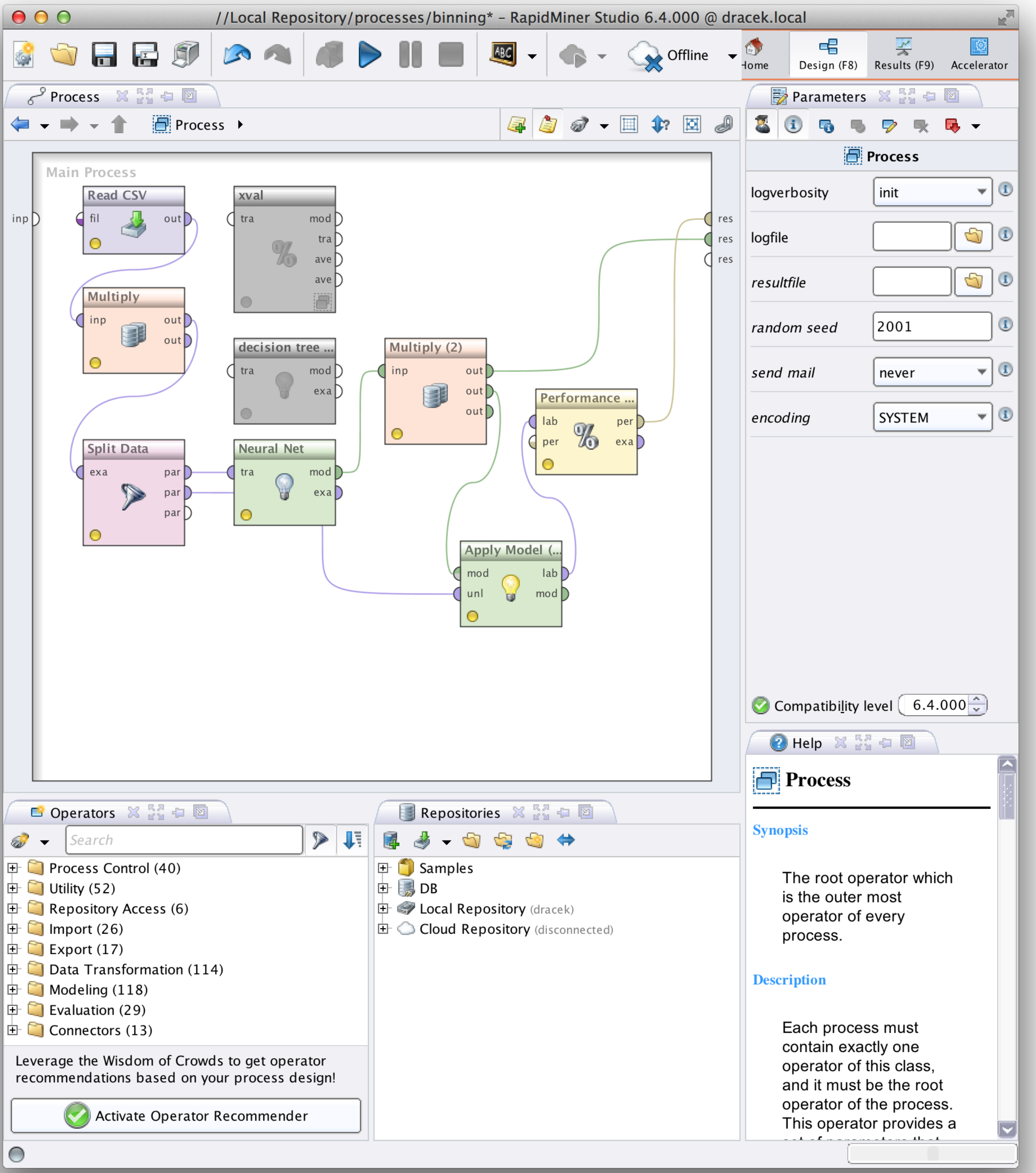
For the purpose of experiments, I am using software platform RapidMiner Studio, which, among other advantages, offers various implementations of machine learning algorithms and is able to display results nicely and clearly [24]. Therefore screenshots in this chapter are obtained from this software. RapidMiner allows the user to create so called 'processes' in which he connects desired algorithms in the graphical user interface (GUI) using them as a black box – an object which can be viewed in terms of input and outputs, without any knowledge of its internal workings. This allows fast prototyping with no programming required (see Figure 5.1).

5.1 Format of experiments

The first paragraph of each experiment section in this thesis defines the filenames of datasets used and the filename of a RapidMiner process contained of the DVD enclosed with this thesis. It also defines which image features and RapidMiner operators are used for the training and settings of the MPN.

5.2 Testing machine

All tests in this chapter were done on the MacBook Pro (Retina, Mid 2012) with 2,3 GHz Intel Core i7 processor with Turbo Boost up to 3.3 GHz. Although it has 8 GB of 1600 MHz DDR3L memory, the free version of RapidMiner Studio can use only 1 GB of memory.



//Local Repository/processes/binning* – RapidMiner Studio 6.4.000 @ dracek.local

Process

logverbosity	init
logfile	
resultfile	
random seed	2001
send mail	never
encoding	SYSTEM

Compatibility level 6.4.000

Process Control (40)
Utility (52)
Repository Access (6)
Import (26)
Export (17)
Data Transformation (114)
Modeling (118)
Evaluation (29)
Connectors (13)

Leverage the Wisdom of Crowds to get operator recommendations based on your process design!

Samples
DB
Local Repository (dracek)
Cloud Repository (disconnected)

Process

Synopsis

The root operator which is the outer most operator of every process.

Description

Each process must contain exactly one operator of this class, and it must be the root operator of the process. This operator provides a

Figure 5.1: Screenshot of the RapidMiner GUI.

5.3 Experiments with color features

In Experiment 1, Experiment 2 and Experiment 3 I tried to solve recognition with only the color of the given image. These were first features I gathered from the images in a thought, that it could be enough information for the MPN to distinguish different paint colors. As the results in respective experiments show, I was not far from the truth. Nevertheless, for production purposes recognition performance have to be as high as possible.

Features used in these experiments are the values of the color histograms and their modifications. The RGB histogram was obtained from the image and the HSV histogram was obtained from the transformed pixels into the HSV color space. I experimented with the histograms themselves and with their modifications using data binning.

5.3.1 Experiment 1

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_1.1.dat
- dataset_1.2.dat

Features used:

- histogram of the hue channel of the HSV, no binning (181 bins)
- histogram of the red channel of the RGB, no binning (256 bins)
- histogram of the green channel of the RGB, no binning (256 bins)
- histogram of the blue channel of the RGB, no binning (256 bins)

RapidMiner operators used:

- see Figure 5.2

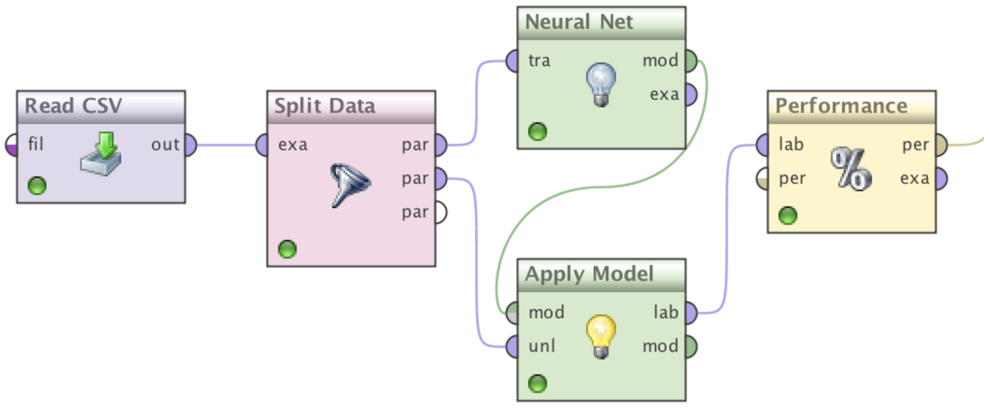


Figure 5.2: Screenshot of how the operators in process 1 are connected.

Settings of the MPN:

- one hidden layer, default size (see Formula 5.1)
- training cycles: 500
- learning rate: 0.3
- error epsilon: 1.0E-5

$$(number\ of\ attributes + number\ of\ classes)/2 + 1 \tag{5.1}$$

5.3.1.1 Results

Datasets in this experiment are created only from the histogram values of the hue channel and all three RGB channels. That results in 949 attributes for each image. Even though the results of this model are sufficient with larger training set, training from this dataset is very time consuming, even if the training is done only on 5 images per class (see Table 5.1 and 5.2).

Table 5.1: Experiment 1 results with dataset 1.1

Training data count	**Training time**	**Training performance**
5	7 m 20 s	92.11%
10	8 m 35 s	98.07%
15	21 m 05 s	97.96%
20	34 m 45 s	98.23%

Table 5.2: Experiment 1 results with dataset 1.2

Training data count	**Training time**	**Training performance**
5	5 m 39 s	95.56%
10	6 m 22 s	95.75%
15	7 m 04 s	96.57%
20	7 m 53 s	98.83%

5.3.2 Experiment 2

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_2.1.dat
- dataset_2.2.dat

Features used:

- histogram of the hue channel of the HSV, 8 bins
- histogram of the red channel of the RGB, 8 bins
- histogram of the green channel of the RGB, 8 bins
- histogram of the blue channel of the RGB, 8 bins

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.3.2.1 Results

This experiment shows, that when binning is used for the histogram values in datasets, training time decreases greatly (see Table 5.3 and 5.4). However the performance of the model with a small training set is not very high.

Table 5.3: Experiment 2 results with dataset 2.1

Training data count	Training time	Training performance
5	1 s	96.58%
10	3 s	98.02%
15	4 s	98.60%
20	5 s	97.77%

Table 5.4: Experiment 2 results with dataset 2.2

Training data count	Training time	Training performance
5	1 s	94.44%
10	2 s	95.88%
15	3 s	97.86%
20	5 s	98.17%

5.3.3 Experiment 3

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_3.1.dat
- dataset_3.2.dat

Features used:

- histogram of the hue channel of the HSV, 16 bins
- histogram of the red channel of the RGB, 16 bins
- histogram of the green channel of the RGB, 16 bins
- histogram of the blue channel of the RGB, 16 bins

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.3.3.1 Results

The dataset of histogram values with 16 bins shows better performance for the second dataset, but does not change much on the first dataset (see Table 5.5 and 5.6). This points to the problem with binning and its information loss – only a small number of bins are used (as in Experiment 2), results from that model are worse than with double the number, because of this information loss.

Table 5.5: Experiment 3 results with dataset 3.1

Training data count	Training time	Training performance
5	3 s	97.06%
10	6 s	97.25%
15	9 s	98.38%
20	13 s	98.17%

Table 5.6: Experiment 3 results with dataset 3.2

Training data count	Training time	Training performance
5	2 s	96.22%
10	6 s	97.88%
15	8 s	98.43%
20	11 s	98.17%

5.4 Experiments with thresholding and segmentation

In Experiment 4 and Experiment 5 I am using images converted to grayscale. The attributes gained from them are the number of white and black pixels with high and low threshold value, respectively, and how many white segments are present in the picture. White segments are also dilated before counting, to reduce the segmentation error.

5.4.1 Experiment 4

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_4.1.dat
- dataset_4.2.dat

Features used:

- number of black pixels, threshold value 30
- number of white pixels, threshold value 230
- number of white segments, threshold value 220

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.4.1.1 Results

Performance of the model trained in this experiment is quite low (see Table 5.7 and 5.8). However, theoretical random choice performance is only about 4.5%, so with combination with other features, it could improve the final model performance.

Table 5.7: Experiment 4 results with dataset 4.1

Training data count	Training time	Training performance
5	0 s	70.39%
10	1 s	79.97%
15	1 s	78.08%
20	1 s	81.63%

Table 5.8: Experiment 4 results with dataset 4.2

Training data count	Training time	Training performance
5	0 s	68.33%
10	0 s	70.75%
15	1 s	76.14%
20	1 s	79.17%

5.4.2 Experiment 5

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_5.1.dat
- dataset_5.2.dat

Features used:

- number of black pixels, threshold value 45
- number of white pixels, threshold value 215
- number of white segments, threshold value 205

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.4.2.1 Results

When the threshold value is increased and decreased for the black pixels and white pixels, respectively, to increase the amount of pixels counted, it improves the performance slightly (see Table 5.9 and 5.10). Therefore usage of these features in the final model with these threshold values is wiser.

Table 5.9: Experiment 5 results with dataset 5.1

Training data count	Training time	Training performance
5	0 s	71.69%
10	1 s	79.92%
15	1 s	79.75%
20	1 s	81.97%

Table 5.10: Experiment 5 results with dataset 5.2

Training data count	Training time	Training performance
5	0 s	66.22%
10	0 s	72.00%
15	1 s	75.57%
20	1 s	83.50%

5.5 Experiments with local binary patterns

In Experiment 6, Experiment 7 and Experiment 8 I am using the LBP and four statistical measures from the LPB matrix – average (see Formula 5.2), deviation (see Formula 5.3), energy (see Formula 5.4) and entropy (see Formula 5.5), where the LBP matrix dimensions are H – height and W – width and $f(i,j)$ is the LBP of the pixel on the position (i,j) in the LBP matrix.

$$AVE = \frac{1}{HW}\sum_{i=0}^{H}\sum_{j=0}^{W} f(i,j) \tag{5.2}$$

$$DEV = \sqrt{\frac{1}{HW}\sum_{i=0}^{H}\sum_{j=0}^{W}(f(i,j) - AVE)^2} \tag{5.3}$$

$$ENE = \sqrt{\frac{1}{HW}\sum_{i=0}^{H}\sum_{j=0}^{W} f^2(i,j)} \tag{5.4}$$

$$ENT = -\sum_{i=0}^{H}\sum_{j=0}^{W}\left(\frac{f(i,j)}{ENE}\ln\left(\frac{f(i,j)}{ENE}\right)\right) \tag{5.5}$$

5.5.1 Experiment 6

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_6.1.dat
- dataset_6.2.dat

Features used:

- histogram of local binary patterns

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.5.1.1 Results

The images of the painted parts do not have any visual patterns in them that could distinguish one from another. In spite of the that fact the model, trained with the usage of LBP histogram, has a useful recognition rate (see Table 5.11 and 5.12). This makes this feature potentially good for usage in the final model, even if its training times are not so fast.

Table 5.11: Experiment 6 results with dataset 6.1

Training data count	Training time	Training performance
5	30 s	83.53%
10	1 m 2 s	88.31%
15	1 m 35 s	90.52%
20	1 m 57 s	90.56%

Table 5.12: Experiment 6 results with dataset 6.2

Training data count	Training time	Training performance
5	28 s	88.44%
10	54 s	93.88%
15	1 m 19 s	95.00%
20	1 m 52 s	96.00%

5.5.2 Experiment 7

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_7.1.dat
- dataset_7.2.dat

Features used:

- histogram of uniform local binary patterns

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.5.2.1 Results

The ULBP histogram used to train the model in this experiment does not achieve as good results as it's non-uniform version, but is much faster, making it a better candidate for the final model (see Table 5.13 and 5.14).

Table 5.13: Experiment 7 results with dataset 7.1

Training data count	Training time	Training performance
5	3 s	78.24%
10	6 s	84.24%
15	9 s	83.31%
20	12 s	86.49%

Table 5.14: Experiment 7 results with dataset 7.2

Training data count	Training time	Training performance
5	2 s	82.67%
10	5 s	89.12%
15	7 s	92.14%
20	10 s	94.17%

5.5.3 Experiment 8

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_8.1.dat
- dataset_8.2.dat

Features used:

- average of local binary patterns
- deviation of local binary patterns
- energy of local binary patterns
- entropy of local binary patterns

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.5.3.1 Results

This experiment is using four statistical measures described in Section Experiments with local binary patterns. Performance of the model trained in this experiment is very low with the usage of only five pictures to train it, but reaches moderate performance with the usage of a larger training dataset (see Table 5.15 and 5.16).

Table 5.15: Experiment 8 results with dataset 8.1

Training data count	Training time	Training performance
5	0 s	57.39%
10	1 s	64.67%
15	1 s	72.27%
20	2 s	72.24%

Table 5.16: Experiment 8 results with dataset 8.2

Training data count	Training time	Training performance
5	0 s	53.78%
10	0 s	70.12%
15	1 s	71.57%
20	1 s	72.33%

5.6 Experiments with combinations of previously described features

In Experiment 9, Experiment 10, Experiment 11 and Experiment 12 models are trained using different combinations of the features from previous experiments. The experiment with the best performance is used in the Section Experiments with colored illumination and in the Section Experiments with MPN settings.

5.6.1 Experiment 9

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_9.1.dat
- dataset_9.2.dat

Features used:

- histogram of the hue channel of the HSV, 16 bins
- histogram of the red channel of the RGB, 16 bins
- histogram of the green channel of the RGB, 16 bins
- histogram of the blue channel of the RGB, 16 bins
- number of black pixels, threshold value 45
- number of white pixels, threshold value 215
- number of white segments, threshold value 205

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.6.1.1 Results

Combination of the color histogram features binned to 16 bins and the thresholding features reaches great performance (see Table 5.17 and 5.18). This makes these features the best candidate for the final model and is used for the experiments with MPN settings.

Table 5.17: Experiment 9 results with dataset 9.1

Training data count	Training time	Training performance
5	4 s	97.16%
10	7 s	97.46%
15	10 s	98.60%
20	14 s	98.05%

Table 5.18: Experiment 9 results with dataset 9.2

Training data count	Training time	Training performance
5	3 s	97.11%
10	6 s	97.62%
15	9 s	98.71%
20	12 s	98.50%

5.6.2 Experiment 10

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_10.1.dat
- dataset_10.2.dat

Features used:

- histogram of the hue channel of the HSV, 16 bins
- histogram of the red channel of the RGB, 16 bins
- histogram of the green channel of the RGB, 16 bins
- histogram of the blue channel of the RGB, 16 bins

- histogram of uniform local binary patterns

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.6.2.1 Results

Results of Experiment 5 and Experiment 7 indicate that the model from the thresholding features performs worse than the model from the ULBP histogram. This is not true when both of those models are also trained with the color histogram features, as shown in the results of this experiment and Experiment 9. The average performance of this model is lower than in the previous experiment, therefore this model will not be used as the final one (see Table 5.19 and 5.20).

Table 5.19: Experiment 10 results with dataset 10.1

Training data count	**Training time**	**Training performance**
5	8 s	95.57%
10	17 s	97.56%
15	25 s	97.95%
20	34 s	97.82%

Table 5.20: Experiment 10 results with dataset 10.2

Training data count	**Training time**	**Training performance**
5	7 s	96.00%
10	15 s	96.25%
15	22 s	98.29%
20	29 s	99.00%

5.6.3 Experiment 11

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_11.1.dat
- dataset_11.2.dat

Features used:

- histogram of the hue channel of the HSV, 16 bins
- histogram of the red channel of the RGB, 16 bins
- histogram of the green channel of the RGB, 16 bins
- histogram of the blue channel of the RGB, 16 bins
- average of local binary patterns
- deviation of local binary patterns
- energy of local binary patterns
- entropy of local binary patterns

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.6.3.1 Results

The model trained with combination of the color histogram features and the statistical measures of the LBP has better performance for most of the data than the Experiment 10, despite the fact that it seemed that statistical measures were useless (see Table 5.21 and 5.22).

Table 5.21: Experiment 11 results with dataset 11.1

Training data count	**Training time**	**Training performance**
5	3 s	96.97%
10	7 s	97.20%
15	10 s	98.01%
20	14 s	98.17%

Table 5.22: Experiment 11 results with dataset 11.2

Training data count	Training time	Training performance
5	3 s	96.78%
10	6 s	98.25%
15	9 s	98.43%
20	13 s	98.17%

5.6.4 Experiment 12

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_12.1.dat
- dataset_12.2.dat

Features used:

- histogram of the hue channel of the HSV, 16 bins
- histogram of the red channel of the RGB, 16 bins
- histogram of the green channel of the RGB, 16 bins
- histogram of the blue channel of the RGB, 16 bins
- number of black pixels, threshold value 45
- number of white pixels, threshold value 215
- number of white segments, threshold value 205
- average of local binary patterns
- deviation of local binary patterns
- energy of local binary patterns
- entropy of local binary patterns

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.6.4.1 Results

When the color histogram features, thresholding features and statistical measures of the LBP matrix are combined to train the model, performance is lower than in the model without the statistical measures (see Table 5.23 and 5.24).

Table 5.23: Experiment 12 results with dataset 12.1

Training data count	Training time	Training performance
5	3 s	96.63%
10	7 s	97.66%
15	11 s	98.87%
20	14 s	98.23%

Table 5.24: Experiment 12 results with dataset 12.2

Training data count	Training time	Training performance
5	3 s	96.67%
10	6 s	97.75%
15	9 s	98.71%
20	12 s	98.67%

5.7 Experiments with colored illumination

Experiments in this sections are focused to find out the ideal color of the illumination for image capturing. Since Experiment 9 performed the best on the two datasets used in previous experiments, its features will be used in the following experiments.

Those two datasets are images captured under the white illumination of the camera. There are four new datasets in this section. All of them are of the same paints as in dataset number 2, but under red, green, blue and yellow illumination. This colored illumination is made by covering the LED of the USB microscope with colored translucent plastic sheets.

5.7.1 Experiment 13

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_9.2.dat
- dataset_13.blue.dat
- dataset_13.green.dat
- dataset_13.red.dat
- dataset_13.yellow.dat

Features used:

- histogram of the hue channel of the HSV, 16 bins
- histogram of the red channel of the RGB, 16 bins
- histogram of the green channel of the RGB, 16 bins
- histogram of the blue channel of the RGB, 16 bins
- number of black pixels, threshold value 45
- number of white pixels, threshold value 215
- number of white segments, threshold value 205

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- Same as Experiment 1

5.7.1.1 Results

This experiment shows that images captured under yellow illumination are the most recognisable. The blue illumination is the second best color, the white and the red illuminations perform approximately the same. The green illumination has mixed performance when trained with different dataset sizes, which makes it the worst color for illumination (see Table 5.25, Table 5.26, Table 5.27, Table 5.28 and 5.29).

These results, however, should be considered cautiously, because colored translucent plastic sheets were used instead of colored LEDs, due to limited resources. Despite all this, this experiment shows that different color of illumination could change the performance of recognition significantly. Because of these results, the dataset of images captured under yellow illumination is included in the following experiments.

Table 5.25: Experiment 13 results with dataset 9.2

Training data count	Training time	Training performance
5	3 s	97.11%
10	6 s	97.62%
15	9 s	98.71%
20	12 s	98.50%

Table 5.26: Experiment 13 results with dataset 13.blue

Training data count	**Training time**	**Training performance**
5	3 s	96.44%
10	6 s	98.25%
15	9 s	98.86%
20	12 s	98.67%

Table 5.27: Experiment 13 results with dataset 13.green

Training data count	**Training time**	**Training performance**
5	3 s	96.78%
10	6 s	98.12%
15	9 s	96.86%
20	12 s	95.67%

Table 5.28: Experiment 13 results with dataset 13.red

Training data count	**Training time**	**Training performance**
5	3 s	95.33%
10	6 s	96.75%
15	9 s	98.43%
20	12 s	98.50%

Table 5.29: Experiment 13 results with dataset 13.yellow

Training data count	**Training time**	**Training performance**
5	3 s	98.67%
10	6 s	99.25%
15	9 s	99.00%
20	12 s	99.33%

5.8 Experiments with MPN settings

Previous experiments helped with the selection of the best features for model training and also considered other colors of illumination for the image capturing itself. Experiments in this section are focused on finding the best settings for the MPN with the usage of the RapidMiner's Loop Parameters operator.

5.8.1 Experiment 14

RapidMiner process:

- process_2.rmp

Datasets:

- dataset_9.1.dat
- dataset_9.2.dat
- dataset_13.yellow.dat

Features used:

- histogram of the hue channel of the HSV, 16 bins
- histogram of the red channel of the RGB, 16 bins
- histogram of the green channel of the RGB, 16 bins
- histogram of the blue channel of the RGB, 16 bins
- number of black pixels, threshold value 45
- number of white pixels, threshold value 215
- number of white segments, threshold value 205

RapidMiner operators used:

- see Figure 5.3 and 5.4

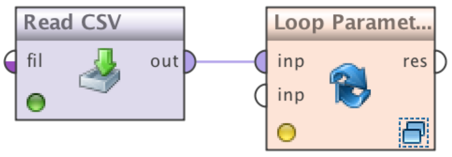


Figure 5.3: Screenshot of outside operators in process 2 and their connection.

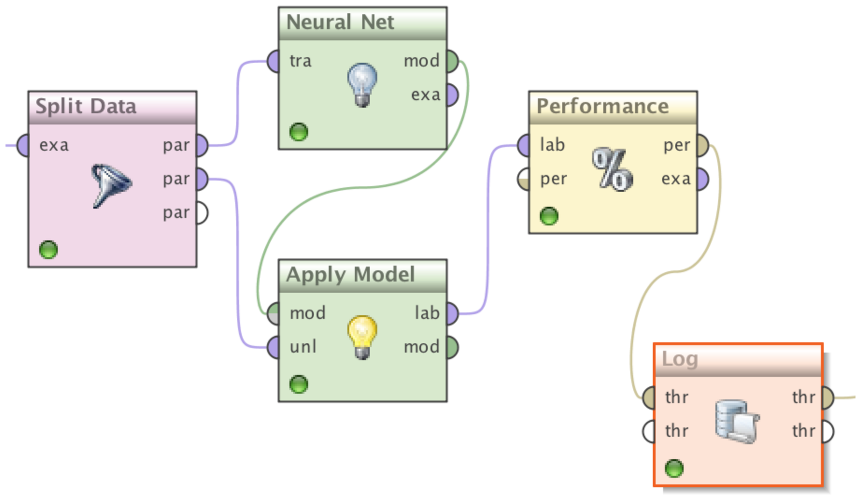


Figure 5.4: Screenshot of inside operators in process 2 and their connection.

5.8.1.1 Results

An experiment, which is not included in the tables below, showed that the best results are achieved with training rates between 900 and 1000, therefore this experiment uses cycle counts of 900, 950 and 1000. The most promising number of training cycles for the MPN and its learning rate is 1000 and 0.2, respectively (see Table 5.30, 5.31 and 5.32).

Table 5.30: Experiment 14 results with dataset 9.1

Training data count	**Learning rate**	**Training cycles count**	**Training time**	**Training perform-ance**
5	0.1	900	6 s	95.61%
5	0.1	950	6 s	95.13%
5	0.1	1000	7 s	96.24%
5	0.2	900	6 s	97.01%
5	0.2	950	6 s	97.15%
5	0.2	1000	7 s	96.67%
5	0.3	900	6 s	95.18%
5	0.3	950	6 s	95.04%
5	0.3	1000	7 s	97.11%
10	0.1	900	12 s	96.89%
10	0.1	950	13 s	97.30%
10	0.1	1000	14 s	96.74%
10	0.2	900	12 s	98.52%
10	0.2	950	13 s	97.81%
10	0.2	1000	14 s	98.01%
10	0.3	900	13 s	97.66%
10	0.3	950	13 s	97.55%
10	0.3	1000	14 s	98.32%
15	0.1	900	19 s	95.69%
15	0.1	950	20 s	98.06%
15	0.1	1000	21 s	98.16%
15	0.2	900	19 s	97.84%
15	0.2	950	20 s	97.73%
15	0.2	1000	21 s	98.54%
15	0.3	900	19 s	98.76%
15	0.3	950	20 s	98.27%
15	0.3	1000	21 s	98.49%

Table 5.31: Experiment 14 results with dataset 9.2

Training data count	**Learning rate**	**Training cycles count**	**Training time**	**Training perform-ance**
5	0.1	900	5 s	95.44%
5	0.1	950	6 s	95.88%
5	0.1	1000	6 s	96.88%
5	0.2	900	5 s	97.55%
5	0.2	950	5 s	96.88%
5	0.2	1000	6 s	96.55%
5	0.3	900	5 s	96.11%
5	0.3	950	5 s	96.55%
5	0.3	1000	6 s	95.00%
10	0.1	900	11 s	98.25%
10	0.1	950	11 s	96.62%
10	0.1	1000	12 s	97.75%
10	0.2	900	11 s	97.62%
10	0.2	950	11 s	98.37%
10	0.2	1000	12 s	98.62%
10	0.3	900	11 s	98.12%
10	0.3	950	11 s	98.75%
10	0.3	1000	12 s	98.12%
15	0.1	900	16 s	98.85%
15	0.1	950	17 s	98.85%
15	0.1	1000	18 s	97.71%
15	0.2	900	16 s	98.57%
15	0.2	950	17 s	98.42%
15	0.2	1000	18 s	99.00%
15	0.3	900	16 s	98.14%
15	0.3	950	17 s	99.14%
15	0.3	1000	18 s	98.71%

Table 5.32: Experiment 14 results with dataset 13.yellow

Training data count	**Learning rate**	**Training cycles count**	**Training time**	**Training perform-ance**
5	0.1	900	5 s	99.00%
5	0.1	950	5 s	98.55%
5	0.1	1000	6 s	96.33%
5	0.2	900	5 s	96.88%
5	0.2	950	6 s	99.66%
5	0.2	1000	6 s	99.33%
5	0.3	900	5 s	96.33%
5	0.3	950	6 s	98.88%
5	0.3	1000	6 s	97.88%
10	0.1	900	11 s	99.62%
10	0.1	950	12 s	98.75%
10	0.1	1000	12 s	99.12%
10	0.2	900	11 s	98.50%
10	0.2	950	11 s	99.50%
10	0.2	1000	12 s	99.12%
10	0.3	900	11 s	99.37%
10	0.3	950	12 s	98.75%
10	0.3	1000	12 s	98.87%
15	0.1	900	16 s	99.42%
15	0.1	950	17 s	99.14%
15	0.1	1000	18 s	98.85%
15	0.2	900	16 s	99.57%
15	0.2	950	17 s	99.42%
15	0.2	1000	18 s	99.71%
15	0.3	900	16 s	99.00%
15	0.3	950	17 s	99.14%
15	0.3	1000	18 s	99.42%

5.8.2 Experiment 15

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_9.1.dat
- dataset_9.2.dat
- dataset_13.yellow.dat

Features used:

- histogram of the hue channel of the HSV, 16 bins
- histogram of the red channel of the RGB, 16 bins
- histogram of the green channel of the RGB, 16 bins
- histogram of the blue channel of the RGB, 16 bins
- number of black pixels, threshold value 45
- number of white pixels, threshold value 215
- number of white segments, threshold value 205

RapidMiner operators used:

- see Figure 5.2

Settings of the MPN:

- layers are described in the tables below
- training cycles: 1000
- learning rate: 0.2
- error epsilon: 1.0E-5

5.8.2.1 Results

The most promising number of hidden layers of the MPN used for training the model is 2 and their sizes are $numberofinputfeatures$ for the first hidden layer and $numberofoutputfeatures$ for the second one (see Table 5.33 and 5.34).

Table 5.33: Experiment 15 results with dataset 9.1

Training data count	MPN layers count	MPN hidden layers sizes	Training performance
5	1	default	97.16%
5	1	22	96.29%
5	1	67	96.92%
5	2	default, default	94.90%
5	2	67, 22	97.16%
10	1	default	97.46%
10	1	22	97.97%
10	1	67	97.20%
10	2	default, default	97.00%
10	2	67, 22	97.00%
15	1	default	98.60%
15	1	22	98.22%
15	1	67	98.55%
15	2	default, default	95.82%
15	2	67, 22	96.17%

Table 5.34: Experiment 15 results with dataset 9.2

Training data count	MPN layers count	MPN hidden layers sizes	Training performance
5	1	default	96.22%
5	1	20	97.44%
5	1	67	96.67%
5	2	default, default	97.33%
5	2	67, 20	97.33%
10	1	default	97.50%
10	1	20	97.12%
10	1	67	97.50%
10	2	default, default	98.12%
10	2	67, 20	97.38%
15	1	default	98.86%
15	1	20	98.86%
15	1	67	98.71%
15	2	default, default	98.29%
15	2	67, 20	97.91%

5.9 The final model

When all of the results from the experiments are gathered and evaluated, features and MPN settings can be chosen for the final model that will have the best performance. The final experiment is a showcase of this model and its performance on the dataset 9.1 and 9.2 as well as on the dataset 13.yellow.

5.9.1 Final experiment

RapidMiner process:

- process_1.rmp

Datasets:

- dataset_9.1.dat
- dataset_9.2.dat
- dataset_13.yellow.dat

Features used:

- histogram of the hue channel of the HSV, 16 bins
- histogram of the red channel of the RGB, 16 bins
- histogram of the green channel of the RGB, 16 bins
- histogram of the blue channel of the RGB, 16 bins
- number of black pixels, threshold value 45
- number of white pixels, threshold value 215
- number of white segments, threshold value 205

RapidMiner operators used:

- Same as Experiment 1

Settings of the MPN:

- one hidden layer, default size (see Formula 5.1)
- training cycles: 1000
- learning rate: 0.2
- error epsilon: 1.0E-5

5.9.1.1 Results

The performance of the final model is almost perfect, on average it is at 99.08% for the small dataset of 10 images for each class (see Table 5.35, 5.36 and 5.37). This outcome is higher than expected.

Table 5.35: Final experiment results with dataset 9.1

Training data count	Training time	Training performance
5	7 s	96.73%
10	14 s	99.12%
15	21 s	99.12%
20	28 s	98.12%

Table 5.36: Final experiment results with dataset 9.2

Training data count	Training time	Training performance
5	6 s	97.82%
10	12 s	99.00%
15	18 s	98.86%
20	24 s	98.87%

Table 5.37: Final experiment results with dataset 13.yellow

Training data count	Training time	Training performance
5	6 s	98.63%
10	12 s	99.12%
15	18 s	98.71%
20	24 s	99.50%

Conclusion

The main goal of this thesis was to explore the recognizability and distinctiveness of colors of painted parts from the automotive industry and what influences its performance. In the theoretical part of this thesis, methods of image processing and their usage with machine learning, to solve this task, were explained. The practical part contains a series of experiments with the features extracted from the images of the painted parts and their influence on recognition performance. Some of the experiments also focus on the influence of colored illumination upon the image capturing itself on the recognition rate. The experiments also focused on the settings of the multilayer perceptron network and its influence on performance. A subset of methods were selected from the image processing and mathematical morphology methods, which achieved the highest recognition performance in the experiments. With these methods and the best settings of the neural network, a very successful model was created for each dataset. The average recognition rate of these models, when tested with images which were not involved in the training, is beyond 99%. All goals of this thesis were fulfilled and based on these facts and on the results of the experiments, I assess this thesis as successful.

Contribution

The results of this thesis will be used in the company WITTE Automotive for reducing costs and facilitating the recognition of colors of painted parts during paint quality assurance. For this purpose, an application in the programming language C++ was implemented, which enables image capturing and the creation of datasets from them. With the usage in another application, which implements some machine learning algorithm, these datasets can be used directly on the production line for paint recognition. This will provide continuous recognition in real time, which allows more painted parts to be checked than a human being is able to check in a given time.

Future work

In the future, further testing of the influence of colored illumination during the image capture would be useful. Also more image processing methods could be tested to find those that could increase the robustness of the model used for recognition. Another useful approach could be an examination of different machine learning algorithms, that could provide some advantages, such as the ability to further learn already learned models when new colors will be added to the dataset. A useful expansion to the implemented application would be the implementation of a machine learning algorithm. That would unify the whole process of recognition into a single application and would simplify its usage on the production line. Also, a combination of implemented functions of image processing methods in the application into one function could increase the processing speed. This would increase the maximal number of checked parts which could lead to the increased production of these parts in WITTE Automotive.

Bibliography

[1] Stone, M. C. *A Field Guide to Digital Color.* A. K. Peters, 2003, ISBN 978-1568811611.

[2] SharkD. HSV color solid cylinder alpha lowgamma. 2010, [Online; accessed 24-04-2015]. Available from: http://commons.wikimedia.org/wiki/File%3AHSV_color_solid_cylinder_alpha_lowgamma.png

[3] ITU-R. Parameter values for the HDTV standards for production and international programme exchange. Recommendation BT.709-5, International Telecommunication Union, Geneva, February 2002.

[4] Michael, A. Caran d'Ache Farbstifte. 2012, [Online; accessed 25-04-2015]. Available from: http://commons.wikimedia.org/wiki/File:Caran_d%27Ache_Farbstifte.JPG

[5] Fletcher, T. Introduction: What is Image Processing. University lecture, 2012. Available from: http://www.coe.utah.edu/~cs4640/slides/Lecture0.pdf

[6] Kordík, P.; Borkovec, J. Přednáška 3: Předzpracování dat. University lecture, 2012. Available from: https://edux.fit.cvut.cz/courses/BI-VZD/_media/lectures/03/p3-preprocessing.pdf

[7] Refaat, M. *Data Preparation for Data Mining Using SAS.* Morgan Kaufmann, 2006, ISBN 978-0123735775.

[8] Solomon, C.; Gibson, S.; Breckon, T. *Fundamentals of Digital Image Processing : A Practical Approach Using Matlab.* John Wiley & Sons, 2010, ISBN 978-0470844724.

[9] Arivumathi. Indian Old Coins. 2013, [Online; accessed 26-04-2015]. Available from: https://commons.wikimedia.org/wiki/File%3A'INDIAN_OLD_COINS_20_PAISA'.JPG

[10] Deguchi, K.; Ghosh, P. K. *Mathematics of Shape Description : A Morphological Approach to Image Processing and Computer Graphics*. Wiley, 2009, ISBN 978-0470823071.

[11] Goutsias, J.; Vincent, L. M.; Bloomberg, D. S. *Mathematical Morphology and Its Applications to Image and Signal Processing*. Kluwer Academic Publishers, 2000, ISBN 978-0792378624.

[12] Ojala, T.; Pietikainen, M.; Maenpaa, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, volume 24, no. 7, July 2000: pp. 971–987, ISSN 0162-8828, doi:10.1109/TPAMI.2002.1017623.

[13] Bradski, G. LBP. 2000, [Online; accessed 27-04-2015]. Available from: `http://docs.opencv.org/_images/lbp.png`

[14] Liu, Z.-G.; Yang, Y.; Ji, X.-H. Flame detection algorithm based on a saliency detection technique and the uniform local binary pattern in the YCbCr color space. *Signal, Image and Video Processing*, 2015: pp. 1–8, ISSN 1863-1703, doi:10.1007/s11760-014-0738-0.

[15] Witten, I. H.; Frank, E. *Data Mining : Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005, ISBN 978-0120884070.

[16] Russell, S. J.; Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003, ISBN 978-0137903955.

[17] Wikipedia. List of machine learning concepts — Wikipedia, The Free Encyclopedia. 2015, [Online; accessed 01-05-2015]. Available from: `http://en.wikipedia.org/wiki/List_of_machine_learning_concepts`

[18] McCulloch, W. S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, volume 5, no. 4, 1943: pp. 115–133, ISSN 0007-4985, doi:10.1007/BF02478259.

[19] Lukić, M.; Ćojbašić, Z.; Rabasocić, M. D.; et al. Neural Networks-Based Real-Time Determination of the Laser Beam Spatial Profile and Vibrational-to-Translational Relaxation Time Within Pulsed Photoacoustics. *International Journal of Thermophysics*, volume 34, no. 8-9, 2013: pp. 1795–1802, ISSN 0195-928X, doi:10.1007/s10765-013-1507-y.

[20] Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning representations by back-propagating errors. *Cognitive modeling*, volume 5, 1986: pp. 533–536.

[21] Feng, L.; Hong, W. Classification error of multilayer perceptron neural networks. *Neural Computing and Applications*, volume 18, no. 4, 2009: pp. 377–380, ISSN 0941-0643, doi:10.1007/s00521-008-0188-0.

[22] Conrad USB Digital Microscope 10x to 200x, 2.0 Megapixel from Conrad Electronic UK. [Online; accessed 01-05-2015]. Available from: http://www.conrad-electronic.co.uk/ce/en/product/191341/Conrad-USB-Digital-Microscope-10x-to-200x-20-Megapixel?ref=searchDetail

[23] Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[24] RapidMiner. RapidMiner. 2006. Available from: https://rapidminer.com/

Appendix A

Acronyms

RGB	red-green-blue
CRT	cathode ray tube
LCD	liquid-crystal display
HSV	hue-saturation-value
HSB	hue-saturation-brightness
LBP	local binary pattern
ULBP	uniform local binary pattern
ANN	artificial neural network
MPN	multilayer perceptron network
AN	artificial neuron
GUI	graphical user interface
LED	light-emitting diode

Appendix **B**

Contents of enclosed DVD

```
readme.txt ................ the file with DVD contents description
data ............................................ the data files directory
  processes ................ the directory of processes for RapidMiner
  images .................... the directory of images used for datasets
  datasets ................................... the directory of datasets
src ...................................... the directory of source codes
  implementation ......... the directory of application's source codes
  thesis ................ the directory of LaTeX source codes of the thesis
text ............................................ the thesis text directory
  BP_Dragoun_Dominik_2015.pdf ................. the thesis in PDF format
```