Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-6 (2018) 208–213

Recognition of CAPTCHA Characters by Supervised Machine Learning Algorithms

Ondrej Bostik * **Jan Klecka** *

* *Department of Control and Instrumentation, Brno University of Technology Brno, Czech Republic (e-mail: bostik,klecka@feec.vutbr.cz).*

Abstract: The focus of this paper is to compare several common machine learning classification algorithms for Optical Character Recognition of CAPTCHA codes. The main part of a research focuses on the comparative study of Neural Networks, k-Nearest Neighbour, Support Vector Machines and Decision Trees implemented in MATLAB Computing environment. Achieved success rates of all analyzed algorithms overcome 89%. The main difference in results of used algorithms is within the learning times. Based on the data found, it is possible to choose the right algorithm for the particular task.

Keywords: CAPTCHA, OCR, Supervised Learning, Template Matching, Decision Trees, k-NN, SVM, Neural Network

1. INTRODUCTION

Hard to find some other branch of the cybernetics increasing so rapidly and persistently as a computer vision. Computer vision techniques, methods, and algorithms solve an unbelievable range of traffic, industrial and other tasks intended for visual sensing. In an early age of the computer vision, only simple image processing tasks were implemented in industry. A well arranged and comprehensive survey can be found in Malamas et al. (2003). These pioneers usually realized an inspection of presence, dimensions or color of given product. This kind of industrial inspection smoothly continues in contemporary tasks of the food industry (e.g. for glass bottles inspection described in Horak et al. (2009b)), visual material analysis (e.g. visual classification of steel wires implemented in Horak (2015)) and many others. Traffic monitoring and offense systems are together with the so-called ADAS systems the second major class of the computer vision applications. A suitable example of traffic monitoring system for traffic queue estimation can be found in Siyal and Fathy (1995) and other examples of ADAS systems are described e.g. in Horak et al. (2009a) and in Horak and Kalova (2010). Besides industry and traffic, also several next domains use image processing tools. A 3D scene reconstruction for robotics in both outdoor and indoor environments are often studied. We have recently introduced, for example, a novel approach for an un-calibrated stereo reconstruction in Klecka and Horak (2015).

It is not a surprise, that not only engineering branches mentioned above use the computer or machine vision enormously as a solution tool. There is a lot of interesting applications of the image processing and especially image retrieval methods (Deb and Zhang (2004)) in an IT and the Internet security branches. One of the various applications of the image processing in the last mentioned domain of the Internet security is well-known Captcha concept as a type of the Turing test. Following chapters will be devoted just to the usage of the computer vision and machine learning techniques in an automated Captcha recognition.

1.1 CAPTCHA Concept

An anonymity of web services often leads to the situation when computer programs substitute humans in monotonous operations. Automated services can send a huge amount of unwanted e-mails, search in public databases for information or influence the online pools.

In the last 20 years, a lot of research has been published dealing with this problem. CAPTCHA (Completely Automated Public Turing Test to tell Computers and Humans Apart) is defined as a general task that must be very easy for the humans to solve, but it must be difficult to create an autonomous machine to solve the task both for the computing resources and for the algorithm complexity (von Ahn et al. (2003)).

1.2 Text-based CAPTCHA

Common most used approach to Captcha implementation for web services is based on OCR (Optical Character Recognition) problem. Current OCR algorithms can be very robust, but they have some weakness. This imperfection limits the usage of this algorithms but can be utilized for Captcha purposes with great advantage. Server sends image with a sequence of characters to the client side. This image is prepared in the way that uses known OCR issues against the computers. At the same time, people who try algorithmically solve this kind of Captcha challenges helps to improve OCR algorithms (Kaur and Behal (2015)).

This kind an iteration process help booth sides, but development advanced so far, that current Captcha schemes are very complex for both computers and humans. Many current Captcha challenges are so complicated, that humans cannot solve them, but machines can. Automated versatile systems for cracking Captcha can beat many schemes

10.1016/j.ifacol.2018.07.155

without any kind of human interaction. Some of these systems can be tweaked to learn new unknown Captcha challenge. As previous research shown us in Bursztein et al. (2011), this kind of system can overcome almost any possible Captcha scheme with high success rate.

Previous study Bursztein et al. (2011) recommend several techniques to improve the security of Captcha schemes. The first to consider is to utilize some kind of anti-segmentation technique. Many systems use lines crossing the letters, but the common mistake is to use long lines (longer than the size of an letter), that can be filtered with Hough transformation. The most straightforward technique is to use variable keyword length, that makes more difficult to guess the position of individual letters.

The next stage of security is at the level of single characters. It is good practice to apply characters of different fonts types, sizes, and rotations. On the other hand, it is not recommended to use of random noise because the current algorithms are better suited to handle the noise than human brains. It is also not recommended to use alike characters like number 0, letter O and big D, which cannot differentiate either by computer either by a human.

A very interesting idea called reCaptcha was described in von Ahn et al. (2008). The further development were later held in Google. The rough estimations published in von Ahn et al. (2008) indicated about 100 million Captcha challenges solved every day with various time from 5 to 20 seconds. This leads us to a situation when humanity wasted dozens of years every day solving Captcha schemes.

Professor Ahn and his team ask a question, how to utilize this time.The solution is simple. Archives contain a great number of documents which are not digitalized.Original system reCaptcha (see fig. 1) consist of two parts. The preparation stage utilizes two OCR algorithms, that tries to transcribe submitted document independently. Outputs are then compared. Matched parts are then marked as correctly solved. Any disagreement in outputs is used to create Captcha challenge von Ahn et al. (2008).

Fig. 1. Sample of Google reCaptcha text scheme, (taken from Carnegie Mellon University (2010))

1.3 Image-based CAPTCHA

Another approach to Captcha challenge creation is to utilize images and prepare test in which subject must tell, what is on the picture, find similarities or point specific object. This kind of Captcha test is more user-friendly and easier to solve then conventional text-based Captchas. An excellent example is Google reCaptcha image scheme shown in the Fig. 2.

Recently an article Sivakorn et al. (2016) was published describing the way to overcome this kind of Captcha challenge. The essential part is to utilize Google Reverse Image Search to collect data about each image and use this information against the Google reCaptcha.

Fig. 2. Sample of Google reCaptcha image scheme, (taken from Sivakorn et al. (2016))

2. SUPERVISED MACHINE LEARNING

As it has been already mentioned in the Introduction chapter and further described in Mitchell (1997), the machine learning realizes a task of inferring a function f between X (input attributes) and Y (desired output) without explicit knowledge of the function f. Machine learning methods are divided into the three main classes: supervised, unsupervised and semi-supervised learning.

Supervised learning means we have an input vector of attributes X and the output label Y for each record (item) in the dataset. Then, we use an algorithm to learn the mapping function $Y = f(X)$. The goal is to approximate the previously mentioned mapping function, that an output prediction Y of new input data X of an unknown label is so accurately as feasible. It is called supervised learning because the labels (correct classifications) are known. A model of predictor iteratively computes predictions on the training data (part of a dataset) and is implicitly corrected by the learning algorithm based on a so-called loss function. Learning process stops when the overall error of the model falls under the specified limit.

Note that supervised learning (Bishop (2006)) tasks are further divided into regression and classification, where classification means the output label is a quality (or category, e.g. bright/dark, healthy/unhealthy, etc.). On the contrary, the regression means the output as a quantity (or number, e.g. 42 MPH, 77 kg, etc.). Almost all practical machine learning tasks lead to the supervised learning and also task of the CAPTCHA recognition in this paper employs the supervised learning.

The unsupervised learning represents a situation where we have input data X without a related label of the output attribute Y. Such tasks of unsupervised learning are generally grouped into clustering and association problems.

In the following sections, only methods selected for our experiments are shortly described. All of them come under the supervised learning group. Namely Decision Trees, K-Nearest Neighbours, Support Vector Machines and Artificial Neural Networks.

2.1 Decision trees

The decision tree is one of the data mining techniques characterized by its clarity and easy interpretability. Obtained results can be quickly evaluated, key items easily identified and searched for segments of interest. The goal of decision

trees is to identify objects described by different attributes into classes. Decisions needed to obtain the results are ordered into a tree-like structure, which is then very fast. The decision tree must be created as a result of learning phase using training data.

Each tree node represents the decision by one (selected) property of the object, resulting in the final number of edges from that node. Node has to distinguish the objects as best as possible. For the root node, an attribute is selected to distinguish the objects from each other. Therefore, entropy (attribute information rate) is used. Commonly used algorithms for tree creation are ID3, C4.5, and CART (Rokach and Maimon (2014)).

2.2 K-Nearest Neighbors

The k-Nearest Neighbors algorithm (k-NN) designed in Altman (1992) is machine learning algorithm for pattern recognition used for classification and regression. The learning phase consists only of storing all the patterns. That is why we call it 'lazy learning'. During the deployment phase, the distances from all the saved patterns to an unknown pattern is calculated. The output is then set as the most common value from the k-nearest neighbor.

The rapid speed of learning phase and ability to enhance the output by dynamically adding new pattern with known outputs are the main benefits of this machine learning algorithm. The main disadvantage is the fact that for every unknown pattern to be classified all the distances from every saved pattern must be calculated every time which is time-consuming. The algorithm is also sensitive to noisy data as it makes no generalization.

2.3 Support Vector Machines

Support Vector Machines (SVM) designed in Cortes and Vapnik (1995) is machine learning classification algorithm which operates above feature space trying to find optimal hyperplane dividing given feature space. The optimal dividing hyperplane is defined as hyperplane with a maximal value of minimal distances between classification data from the plane. This means that hyperplane keeps maximal margin between two groups. Only a few closest points to hyperplane are required to define the borders between these two groups. These border points are called support vectors and from them, the name of this method is derived.

The dividing hyperplane is defined by equation 1:

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0 \tag{1}$$

where $\boldsymbol{w}$ is the normal vector to the hyperplane, $\boldsymbol{x}$ is vector of input points and $\frac{\boldsymbol{b}}{||\boldsymbol{w}||}$ is the offset of hyperplane from origin Cortes and Vapnik (1995).

Initial condition for the problem is stated in equation 2:

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 - \xi_i \geq 0 \quad \xi_i \geq 0 \quad \forall i \tag{2}$$

where $y_i = +1$ for all points from one class and $y_i = -1$ for the points from the other class. The optional parameter ξ_i can be used during classification of noisy data and it states the price for violation of the dividing hyperplane.

The learning is mathematically defined by equation 3:

$$||\boldsymbol{w}||^2/2 + C \cdot \sum_i \xi_i \tag{3}$$

where C is an optional tunable parameter representing the penalization of misclassified points.

The above equation was designed for linear separable problems. For the non-linear problem, SVM utilizes the kernel trick which translates the feature space into a higher dimension where the dividing hyperplane can be found as described in Boser et al. (1992).

2.4 Artificial Neural Networks

Artificial Neural Network (ANN) is structure simulating biological neurons and their interconnections (Bishop (2006)). Neurons are virtually connected to each other in a way that allows transferring information through the network from input to output. Neural networks commonly consist of multiple neurons divided into several layers.

During the learning phase, the sample data are presented to the network and known output is compared to the actual output and the difference is used to correct the nets behavior. A common approach is called Error Back Propagation which is the algorithm of spreading the error back from output to input in order to correct the value of weights and improve the network performance. Learning phase iterate through all sample data and lasted as long as the performance function reaches the target value.

3. PROPOSED RECOGNITION METHOD

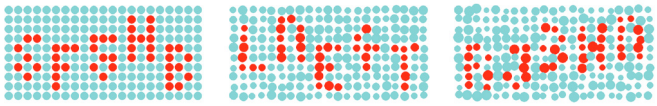
As we mentioned before, one of the key element to successful text-based Captcha scheme is the use of multiple fonts, see Bursztein et al. (2011). Our research takes this recommendation and to improve it, we start to think about the idea of generating the font for every character dynamically. The goal of Bubble Captcha is to utilize some kind of text-based Rorschach images, see Rorschach (1998).

Even that the initial purpose of Rorschach images was to evaluate humans psychical condition and even that some people can see them as a form of abstract art, this kind of Captcha images are going to be very useful to distinguish between humans and computers. This idea is based on the imagination insufficiency of machines. Of course, deploying of some kind of machine learning techniques can imitate imagination, but we hope that with the right amount of abstraction in Captcha generation, the Captcha is going to be a rough challenge for computers but easy for humans.

3.1 Bubble CAPTCHA Concept

In previous work Bostik et al. (2017), we developed elementary bi-color Captcha scheme with randomly positioned circles/bubbles forming the font. The system was designed as a web application in PHP. The main reason was to prepare a platform for rapid Bubble Captcha testing with a wide variety of people. The system can be parameterized and can be used to generate single Captcha challenge for security purposes on web pages or can be used to quickly generate the set for testing OCR algorithms.

A two-dimension array representing binary grid for every character used is one of the key elements entering the algorithm. Generative algorithm randomly selects characters from used character set and position them bubble



(a) (b) (c)

Fig. 3. Bubble Captcha variants overview with (a) no displacement - answer 6F5HB (b) medium displacement - answer LNE4T (c) big displacement - answer RJ3HN

by bubble to generate our Captcha scheme alongside with bubbles of different color. Every bubble is randomly positioned approximately to its correct position in the grid and randomly scaled in size within predefined limits. The three different variants (denoted as *a*, *b* and *c* hereinafter) of Bubble Captcha are generated in this way as an input dataset of the supervised machine learning algorithms. Implemented Captcha scheme is shown in the figure 3 displaying 3 main levels of randomness in the picture.

3.2 Feature Extraction

Bubble Captcha images showed in the previous figure are pre-processed in order to ensure proper input to the learning methods. Each Captcha word is divided into individual characters. This is done by firmly determined vertical lines because of the spacing between all characters is the same and is exactly known from training dataset creation stage. An example of the letter *R* separated from the Captcha word is depicted in the figure 4(a).

(a) (b)

Fig. 4. Color based feature extraction (a) original segmented letter (b) binary image

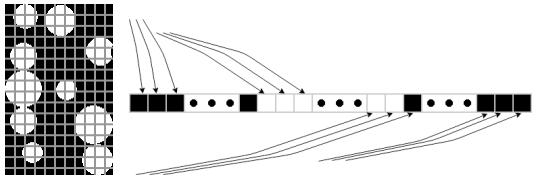
The next step is a color segmentation of the separated character. A simple color thresholding is employed during this step. All pixels with a red channel value above the specified threshold are classified as pixels of the letter and the others as pixels of background. Further, a segmented area of the character is normalized to dimensions of 102 by 172 pixels. The result of such color segmentation with normalization is shown in the figure 4(b) as a binary image.

The last step of the feature extraction stage is a linearization of the binary character image. Values of all pixels of the image are successively put down next to each other in a raw by a raw scheme. This image decomposition yields in a vector of the length of 17544 values and are schematically illustrated in the figure 5(b).

The feature extraction stage results into the vector of given and always the same length. This vector serves as an input to the machine learning models described below.

4. EXPERIMENTS

Following chapter describes experiments done in MATLAB computing environment with Statistics and Machine Learning Toolbox and Optimization Toolbox.



(a) (b)

Fig. 5. Reordering of 2D image matrix into input vector (a) binary 2D image 102x172px (b) input vector with lenght of 17544 for machine learning algorithms

4.1 Input gallery overview

Large gallery of synthetic character was generated with the PHP generator described above. For initial experiments, we use the same three levels of distortion as in previously tested Bubble Captcha scheme. Main parameters are shown in table 1 below.

Table 1. Parameters of the Bubble Captcha generator

Variant [–]	Grid step [*px*]	Bubble's size degradation [%]	Bubble's position degradation [%]
a	35	0 %	0 %
b	35	10 %	20 %
c	35	15 %	30 %

Common Captcha schemes do not utilize alike characters, that can be easily confused with each other (for example number zero and letter O). For that reason, input gallery contains 33 different sets of characters. Each character set that contains 50 instances for each variant.

In conclusion, the gallery contains 4950 character samples.

4.2 K-Nearest Neighbors

First classification algorithm used was K-Nearest Neighbors classification algorithm. Standard setting was used, algorithm classification was based on exactly one closest element. When more than one stored element has the same distance from the classified item, the algorithm chooses the one with the lowest index (numbers and letters are sorted alphabetically with the numbers before letters).

Euclidean distance with no weighting was used as metric for algorithm evaluation. Computation of distances uses an unoptimized exhaustive algorithm, that means all distances are evaluated. No score transformation is utilized.

As expected, the k-NN algorithm is the fastest to learn - it stores only of all learning pattern. In contradictory, the algorithm is also the slowest during the classification phase because of the exhaustive distance computing.

4.3 Pattern Recognition Artificial Neural Networks

Pattern Recognition Artificial Neural Networks is a feed-forward neural network, that can be used for pattern recognition and classification to target classes.

This neural network consists of an input layer, two hidden layers, and output layer. Input layer with 17544 neurons

only propagates signal into the network. Two hidden layers are almost identical - both utilizes Nguyen-Widrow initialization method, the both has a transfer function of hyperbolic tangent sigmoid, neurons in both layers use the summation of weight and biases as an input function. Fist hidden layer consist of 200 neurons and second hidden layer consist of 50 neurons. Output layer has 33 neurons with transfer function Soft Maximum. Remaining parameters of output layer are the same as with the hidden layers.

Performance of the network is evaluated based on cross-entropy calculated with equation 4:

$$ce_i = -t_i \log(y_i) \tag{4}$$

where y_i are the output data and t_i are the target data for i-th pair of output-target values. Aggregate cross entropy is evaluated as average of individual cross entropy values (Bishop (2006)). For learning the Scaled Conjugate Gradient Backpropagation described in Møller (1993) was used. The maximum number of the epoch was set to 1000 and minimal gradient to 1e-6.).

4.4 Feed-forward Artificial Neural Networks

To compare results with a universal neural network the standard implementation of a neural network within MATLAB was used. The parameters and topology are almost identical. The first of two main differences were using a linear identical function in the output layer. The second difference was a function to compute performance of the network - mean squared normalized error.

4.5 Decision Trees

In this method, the binary regression decision tree was fit to data. To create the decision tree, the standard CART algorithm was utilized. The split criterion used in this task was Gini's Diversity Index.

The decision tree uses no score transformation. Final decision tree created with described input gallery has 65 nodes in total. No pruning was applied to the final tree, but all the child nodes with the risk greater or equal to the risk of parent nodes are merged.

4.6 Support Vector Machines

The used learning algorithm to create classifier operates based on Error-Correcting Output Codes (ECOC) algorithm Dietterich and Bakiri (1994). This approach extends the SVM binary classifier to be used for multi-class classification. This particular implementation uses one-to-one coding design, which means that for every combination of two classes within the learning set, one is declared as positive, the second one is declared as negative and the rest of the classes are omitted. The number of resulting binary learners for K classes is $K(K-1)/2$.

For evaluation performance of each learner, the binary loss is calculated with Hinge function described by equation 5:

$$Hinge = max(0, (1 - y_j s_j))/2 \tag{5}$$

where y_j is the class label for binary learner and s_j is the score for observation j.

4.7 Experiments Evaluation

Two main aspects are compared during this experiment. The first one is the success rate of each algorithm. As can be seen in table 2, the best success rate was achieved by Pattern Recognition Artificial Neural Network which on provided sample data has the success rate of 100%. The Support Vector Machine classifier ended on second place with only 2 misclassification which leads to the success rate of 98.80%.

The worst obtained classifier was Feed-Forward Artificial Neural Network. This algorithm was not designed to pattern classification task and ended with the success rate of 98.79%. Despite this, the results are very good and they reflect the strength of Artificial Neural Network.

Table 2. Comparison of Algorithms' Precisions

	True positive	False positive	Precision
k-NN	980	10	98.99 %
Pattern net	990	0	100.00 %
Feed-forward net	978	12	98.79 %
Decision tree	980	10	98.99 %
SVN	988	2	99.80 %

The second aspect to compare is the time consumption and in that manner computational complexity of each individual algorithm. The results are shown in table 3.

Table 3. Comparison of Algorithms' Computational Costs

	T_{learn} $[s]$	T_{class} $[s]$	T_{comb} $[s]$	Comparation $[-]$
k-NN	1.37	716.12	717.50	7.95 %
Pattern net	565.33	192.61	757.94	8.40 %
Feed-forward net	8827.57	193.49	9021.06	100.00 %
Decision tree	21.52	1.67	23.19	0.26 %
SVN	314.97	322.17	637.14	7.06 %

The second aspect to compare is the time consumption and in that manner computational complexity of each individual algorithm. The results are shown in table 3.

One of the two-time parameters is learning time T_{learn}. The obvious winner is k-Nearest Neighbor algorithm because of the simplicity of learning phase - storing all patterns are very time efficient. Decision trees are a bit slower, but the resulting time of $20s$ is good too. The slowest predictor to learn is the inappropriately used Feed-Forward Artificial Neural Network with learning phase that lasts almost $2,5$ hours.

The best classification time was achieved by Decision tree algorithm. The worst time was recorded with k-Nearest Neighbor algorithm because of the lazy learner nature of the classifier. Two examples of neural networks have almost identical classification time because of their identical sizes and topology.

We must mention some notes at the end. Table 2 shows only columns True positive and False positive. Other values are not shows as the values are zero, as all classifier returns some character in all cases. Also, column T_{comb} in table 3 has only informational character.

5. CONCLUSION

This work compares commonly used supervised machine learning algorithms for Optical Character Recognition of Captcha codes. Our experiments reveal that all of the used algorithms can classify the objects into right class with success rate around 99%. The main difference between algorithm is in computational costs. The overall winner in both categories combined is Patter recognition neural network as it has both a good precision and low computational cost.

There is also a great difference between two almost identical neural network algorithm. The inappropriately used neural network called Feed-forward net which was not optimized for pattern recognition had great difficulties during learning phase to achieve a right performance.

Our following work will be focused on improving Bubble Captcha scheme. We want to utilize this research to test success rate of this improved version of Bubble Captcha against automated solvers.

ACKNOWLEDGEMENTS

The completion of this paper was made possible by the grant No. FEKT-S-17-4234 - 'Industry 4.0 in automation and cybernetics' financially supported by the Internal science fund of Brno University of Technology and Competence Center realized by TACR (reg. number TE01020197).

REFERENCES

Naomi S. Altman. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *Am. Stat.*, 46(3): 175–185, aug 1992. ISSN 0003-1305. doi: 10.1080/00031305.1992.10475879.

Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006. ISBN 0387310738.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. fifth Annu. Work. Comput. Learn. theory - COLT '92*, pages 144–152, New York, New York, USA, 1992. ACM Press. ISBN 089791497X. doi: 10.1145/130385.130401.

Ondrej Bostik, Karel Horak, Jan Klecka, and Daniel Davidek. Bubble Captcha - A Start of the New Direction of Text Captcha Scheme Development. In *Mendel 2017, 23rd Int. Conf. Soft Comput.*, volume 23 of *23*, pages 57–64. Brno University of Technology, 2017.

Elie Bursztein, Matthieu Martin, and John C. Mitchell. Text-based CAPTCHA strengths and weaknesses. *Proc. 18th ACM Conf. Comput. Commun. Secur.*, 2011:125–138, 2011. ISSN 15437221. doi: 10.1145/2046707.2046724.

Carnegie Mellon University. The Official CAPTCHA Site, 2010. URL `http://www.captcha.net/`.

Corinna Cortes and Vladimir N. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, sep 1995. doi: 10.1007/BF00994018.

Sagarmay Deb and Yanchun Zhang. An Overview of Content-based Image Retrieval Techniques. *18th Int. Conf. Adv. Inf. Netw. Appl. 2004 AINA 2004*, 1:59–64, 2004. doi: 10.1109/AINA.2004.1283888.

Thomas G. Dietterich and Ghulum Bakiri. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *J. Artif. Intell. Res.*, dec 1994.

Karel Horak. Classification of segregation level in steel wires by image processing. *2015 38th Int. Conf. Telecommun. Signal Process. TSP 2015*, 2015. doi: 10.1109/TSP.2015.7296413.

Karel Horak and Ilona Kalova. Eyes Detection and Tracking for Monitoring Driver Vigilance. In *33rd Int. Conf. Telecommun. Signal Process.*, pages 204–208, 2010.

Karel Horak, Miloslav Richter, and Ilona Kalova. Human eyes localization for driver inattention monitoring system. *Mendel*, 15(chapter 2):283–288, 2009a. ISSN 18033814.

Karel Horak, Miroslav Richter, and Ilona Kalova. Automated Flaws Detection on Bottles in Food Industry. *Proc. 20th Int. DAAAM Symp.*, (May 2014):4–6, 2009b.

Kiranjot Kaur and Sunny Behal. Designing a Secure Text-based CAPTCHA. In *Procedia Comput. Sci.*, volume 57, pages 122–125. Elsevier, 2015. doi: 10.1016/j.procs.2015.07.381.

Jan Klecka and Karel Horak. Fusion of 3D model and uncalibrated stereo reconstruction. *Adv. Intell. Syst. Comput.*, 378:343–351, 2015. ISSN 21945357. doi: 10.1007/978-3-319-19824-8_28.

Elias N. Malamas, Euripides G M Petrakis, Michalis Zervakis, Laurent Petit, and Jean Didier Legat. A survey on industrial vision systems, applications and tools, 2003.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997. ISBN 0070428077.

Martin F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *NEURAL NETWORKS*, 6 (4):525—-533, 1993.

Lior Rokach and Oded Maimon. *Data Mining with Decision Trees*. Series in Machine Perception and Artificial Intelligence. World Scientific, 2 edition, oct 2014. ISBN 978-981-4590-07-5. doi: 10.1142/9097.

Hermann Rorschach. *Psychodiagnostics: A Diagnostic Test Based on Perception*. Hogrefe Huber Pub, 10th edition, 1998. ISBN 3456830246.

Suphannee Sivakorn, Iasonas Polakis, and Angelos D. Keromytis. I am Robot: (Deep) Learning to Break Semantic Image CAPTCHAs. In *2016 IEEE Eur. Symp. Secur. Priv. (EuroS P)*, pages 388–403, 2016. doi: 10.1109/EuroSP.2016.37.

Mohammed Y. Siyal and Mahmood Fathy. Real-Time Measurement of Traffic Queue Parameters by Using Image Processing Techniques. In *Fifth Int. Conf. Image Process. its Appl.*, pages 450–454, 1995. doi: 10.1049/cp:19950699.

Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using Hard AI Problems for Security. In *Lect. Notes Comput. Sci.*, pages 294–311. Springer, Berlin, Heidelberg, 2003. doi: 10.1007/3-540-39200-9_18.

Luis von Ahn, Benjamin Maurer, Colin Mcmillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science (80-.).*, 321(12 September 2008):1465–1468, sep 2008. ISSN 0036-8075. doi: 10.1126/science.1160379.