Automatic recognition of handwritten corrections for multiple-choice exam answer sheets

Marko Čupić, Karla Brkić, Tomislav Hrkać, Željka Mihajlović and Zoran Kalafatić
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10 000 Zagreb, Croatia
{marko.cupic, karla.brkic, tomislav.hrkac, zeljka.mihajlovic, zoran.kalafatic}@fer.hr

***Abstract*—Automated grading of multiple-choice exams is of great interest in university courses with a large number of students. We consider an existing system in which exams are automatically graded using simple answer sheets that are annotated by the student. A sheet consists of a series of circles representing possible answers. As annotation errors are possible, a student is permitted to alter the annotated answer by annotating the“error” circle and handwriting the letter of the correct answer next to the appropriate row. During the scanning process, if an annotated“error” circle is detected, the system raises an alarm and requires intervention from a human operator to determine which answer to consider valid. We propose rather simple and effecive computer vision algorithm which enables automated reading of a limited set of handwritten answers and minimizes the need for a human intervention in the scanning process. We test our algorithm on a large dataset of real scanned answer sheets, and report encouraging performance rates.**

I. Introduction

Automated grading of multiple choice exams is a necessity in modern university courses taken by a large number of students (e.g. a few hundreds). Although in recent years there has been a notable shift towards taking multiple choice exams on computers [1], [2], written exams are still inevitable when the number of students taking the exam surpasses the number of computers the university can provide at a given point in time. Manually grading hundreds of multiple choice exams is tedious work prone to errors, and a task more suitable for a machine than for a human. Using automated grading saves time, enables faster processing of exams and reduces the probability for grading errors. If exams are graded by a machine, the results of an exam can be known in as little as a few hours after the exam has finished, with very little human effort, hence offering a considerable benefit both for students and for course staff.

In this paper, we consider an existing system for automated exam grading [3], [4] currently in use at the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. The system operates by scanning and recognizing answer sheets of a predefined format on which the desired answers are annotated by the student. Although offering a fair amount of automation, the system is not yet fully automated, as it requires operator input in cases when the student makes an annotation error, i.e. decides to change an already annotated answer. In these cases, the student is expected to annotate a special error column (or annotate more than one answer where only a single answer is expected) and handwrite the letter corresponding to the appropriate answer in a dedicated rectangle. When the system detects that more than one answer or the error column had been annotated, a prompt is raised to the human operator, who is required to read the handwritten letter and input it back into the system. Hence, a certain amount of supervision is required during the automated grading process. We address this problem by proposing a computer vision technique for handwritten letter recognition suitable for use in the context of automated exam grading. The recognition task is constrained, as a limited number of letters need to be supported, given that multiple choice exams rarely offer more than six choices. Still, the relative simplicity of the task balances itself with high performance requirements: the recognition should be as fast as possible and as reliable as possible, and generalize well over a large number of different handwritings. Errors are highly undesirable, so the recognition module should be able to assess its own certainty in the recognition result. For uncertain recognitions, human input should be requested.

II. Related work

Character recognition has been a subject of intensive research for a long period of time. While recognition of printed text is today mostly considered a solved problem, recognition of handwritten text remains a challenging task, primarily due to high variability in handwriting among different people. A large number of methods have been proposed for handwritten text recognition. These methods can generally be categorized either as online or offline. Online recognition involves the automatic and real-time conversion of a text as it is written on a special digitizing device, where a sensor tracks the movements of the pen and an algorithm processes this information, while offline methods start from a static image usually obtained by scanning a document. An overview focused on offline handwritten character recognition methods can be found in [5].

The task of character recognition generally involves the following processing steps [5], [6]:

(i) Preprocessing (for instance including size normalization, image binarization, etc.),
(ii) Feature extraction,
(iii) Classification.

It is presumed that an adequate method for character localization has been previously applied, in order to find parts of the image containing each of the characters.

The key step in the process is the selection of an appropriate feature extraction method (or representation), since features must at the same time be sufficiently discriminative among the character classes, but also invariant to the expected variations inside the class. Furthermore, the size of feature vectors should be relatively small in order to reduce computational complexity and therefore the time needed to perform the classification. A variety of different feature representations have been proposed in the literature. A standard survey can be found in [7]. Commonly used feature representations can be classified in three broad classes [5], [7]:

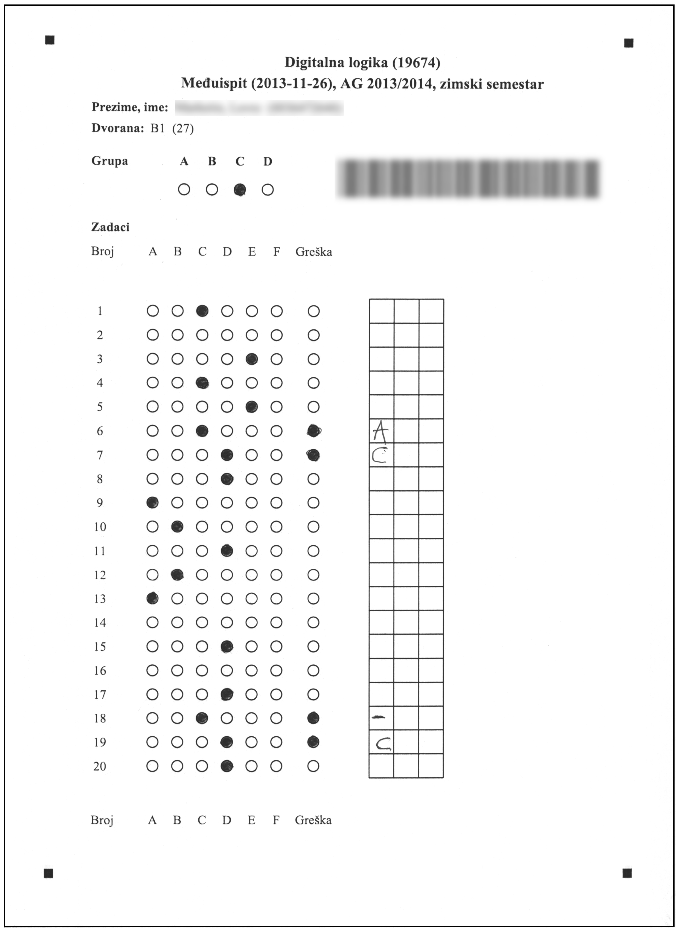
(i) *Statistical representations*, that represent a character image by a statistical distibution of points. This class includes methods such as analyzing densities of points or other features in different regions of an image (zoning), projection histograms, counting the number of crossings of an image contour with line segments in specified directions, and distances of line segments from given boundaries.

(ii) *Structural representations*, based on geometrical and topological properties, such as aspect ratio, distances between characteristic points or strokes, cusps, openings in different directions, chain codes, graph and tree representations, etc. These kinds of features encode some knowledge of the structure of the character, i.e. what sort of components it is made of.

(iii) *Representations obtained by global transformations*, that describe an image by a linear combination of a series of simpler well-defined functions. The features are then formed by coefficients of a such linear combination. Fourier transforms, Gabor transforms, wavelets, moments, Karhunen-Loeve expansion, etc., can be counted in this category.

As already stated, using distances of the image elements from the predefined boundaries has been used and described in the literature. Kim et al. [8] used distances of image pixels from the minimum bounding box of a whole word, along a certain number of horizontal and vertical scan-lines, in combination with several other feature types combined in 120-dimensional feature vector, applied to much harder problem of recognition of a limited set of handwritten cursive words. By using a heterogeneous classifier consisting of two multi-layer perceptrons and a hidden Markov model, they achieved recognition rate of 92.7% for 32 classes of legal words. Similarly, Mohamed [9] represented cursive handwritten word images by computing the location and number of transitions from background to foreground pixels along vertical lines (image columns). Hidden Markov model was used for classification. The obtained recognition rate was 83.9% (for the first ranked class). Surinta et al. [10] use the distance values between the closest black pixels and predefined hotspots in four or eight different directions to represent characters. They report the recognition rate of 90.1%.

In this paper, we adhere to the general idea of using image distances, and propose an offline character recognition method particulary suitable for recognizing a limited set of characters in the context of automated grading. In contrast to previously described methods, our method is specifically optimized to the set of supported characters, i.e. it can be said that it incorporates a degree of prior knowledge. It is designed so as to maximize separability between supported character classes by measuring specific properties characteristic to individual classes (e.g. it assumes that the existence or absence of the middle dash is an important classification cue).



Digitalna logika (19674)
Međuispit (2013-11-26), AG 2013/2014, zimski semestar

Prezime, ime: [redacted]
Dvorana: B1 (27)

Grupa A ○ B ○ C ● D ○

[redacted]

Zadaci

Broj	A	B	C	D	E	F	Greška			
1	○	○	●	○	○	○	○			
2	○	○	○	○	○	○	○			
3	○	○	○	○	●	○	○			
4	○	○	●	○	○	○	○			
5	○	○	○	○	●	○	○			
6	○	○	●	○	○	○	●	A		
7	○	○	○	●	○	○	●	C		
8	○	○	○	●	○	○	○			
9	●	○	○	○	○	○	○			
10	○	●	○	○	○	○	○			
11	○	○	○	●	○	○	○			
12	○	●	○	○	○	○	○			
13	●	○	○	○	○	○	○			
14	○	○	○	○	○	○	○			
15	○	○	○	●	○	○	○			
16	○	○	○	○	○	○	○			
17	○	○	○	●	○	○	○			
18	○	○	●	○	○	○	●	-		
19	○	○	○	●	○	○	●	C		
20	○	○	○	●	○	○	○			
Broj	A	B	C	D	E	F	Greška			

Fig. 1. An example of a standardized answer sheet.

III. The existing system

The first step in developing any automated grading system is to introduce a standardized answer sheet that can be machine-processed. In this paper, we consider an automated grading system called *FreeFormReader*, currently in use at the Faculty of Electical Enginnering and Computing at University of Zagreb. An example of a standardized answer sheet that can be processed by the system is shown in Fig. 1. The answers are annotated by the student in an answer matrix that consists of an appropriate number of rows of empty circles. Each row corresponds to a single question, and each column to a single answer. To annotate an answer, the student darkens the appropriate circle. It is often the case that a student annotates an answer and later changes his mind. If that happens, the student is expected to annotate the "Error" cell ("Greska" in Croatian, as shown in Fig. 1) in the appropriate row or darken the correct answer in the same row, and write the letter corresponding to the final answer in the rectangle near the appropriate row, also illustrated in Fig. 1. When the system detects that the "Error" cell is marked, or that two or more answers are marked, an alarm is raised and a human operator must inspect and manually enter the correct answer.

Given that many exams at our university are graded in a way that wrong choices add negative points to the exam score (usually -0.25 points), it sometimes happens that a student wishes to revoke his answer to a particular question, and leave the question unanswered. In that case, the same procedure is followed, except that instead of handwriting the letter of the appropriate answer, the student is expected to write the dash symbol (“-”). On the other hand, if the student is confident that for a certain question none of the offered answers are correct, or that there is more than one correct answer, the student is allowed to annotate the “Error” column in the corresponding row and write the letter X in the appropriate rectangle. This situation can occur when there has been a mistake in forming the exam. If an X is detected, the human operator should manually inspect the student’s solution.

Currently, the system supports exams with up to 23 answer choices. However, the most frequent case are questions with four to six answers and for these cases we are trying to further automate the recognition process. Our practical experiences suggest that errors in annotation happen in about 20% of exams. The process of manually correcting annotation errors is tedious and it is easy to make a mistake, given the repetitive nature of the task.

IV. Representing and Classifying the Characters

In devising an appropriate representation and classification method for handwritten characters that could be used within our exisiting system, we adhere to the following requirements.

(i) The required computations for representation and classification should be very fast, so that current processing speed of the system is retained.
(ii) It can be assumed that the set of supported characters is limited to six answer letters (A, B, C, D, E, F) and two characters for revoking the answer and signaling a mistake in the question (dash and X).
(iii) The classification framework should be able to output confidence level. Predictions with low confidence level should be presented to the human operator.

In the following section we describe the method that we propose for the character recognition.

A. The edge distance representation

After scanning the document, an approximate location of the table containing the handwritten correction answers is manually annotated by the human operator. This has to be done only for the first exam sheet, since the approximate position of the table remains the same for all the other sheets. Then the precise location of the table and the locations of individual cells must be found. This can be done either by finding lines using Hough transform (and finding individual cells by finding their corners defined by intersections of found lines), or using a method described in [4], or any other. Currently, for experimentation purposes, we use (slow) Hough transform.

Once the individual cells have been found, a procedure for finding a minimal character bounding box is performed. Then the corresponding part of the image is further preprocessed by reducing its dimensions to 24×32 pixels, since we expect that in this resolution the distinguishing features of characters are still preserved. We then perform binarization by simple histogram-based adaptive thresholding. The result is a $24 \times$

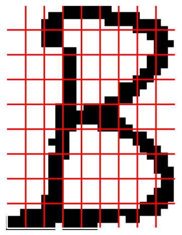


Fig. 2. The grid of $M \times N$ horizontal and vertical scan-lines

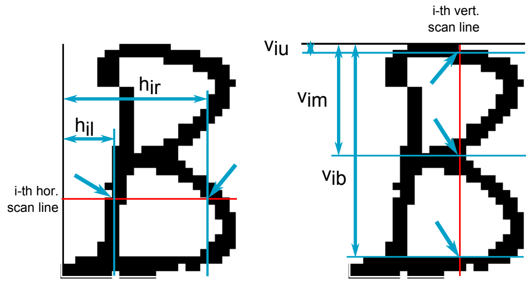


Fig. 3. The definition of features: (a) horizontal features; (b) vertical features

32 binary image in which the character extends from top to bottom and from left to right margin of the image. The size normalization is essential, since the proposed features will only have low variability in case of size normalized images, thus lending themselves to character recognition.

To facilitate the recognition, we propose a simple but effective feature extraction method resulting in an *edge distance* representation of characters (Figs. 2 and 3). In the image, M horizontal and N vertical equidistant scan-lines are defined (in our experiments, $M = N = 8$), as shown in Fig. 2. On each horizontal scan-line i, two features are defined (Fig. 3(a)): the relative distance of the leftmost black image pixel on that line from the left margin of the image (h_{il}) and the relative distance of the righmost black image pixel on the same line again from the left image margin (h_{ir}). On each vertical scan-line, however, three features are defined (Fig. 3(b)): the relative distance of the uppermost black pixel on the $i-th$ vertical scan-line from the upper image border (v_{iu}), the relative distance of the bottom-most black pixel on the same line from the top of the image (v_{ib}), and finally, the third feature (v_{im}) intended to reflect the position of the “middle stroke” that exists in some letters (such as B and E), i.e. the relative distance of the pixels of this “middle stroke” crossing the $i-th$ vertical line from the top of the image, if the “middle stroke” exists. More precisely, v_{im} is defined as the relative distance from the top of the image to the black pixel below the uppermost pixel that is at the same time disconnected both from the uppermost and from the bottommost pixels and is closest to the center of the letter. If it does not exist, i.e. if there are no pixels on the $i-th$ vertical scan-line disconnected from uppermost and bottommost black pixels, the feature v_{im} is set to 0. The described representation results in a feature vector of dimension $2M+3N$ (in our case the dimension was $2 \times 8 + 3 \times 8 = 40$).

Intuitively, the features defined in the described way should be sufficiently discriminative for characters belonging to distinct classes, while maintaining a relatively low variability inside a class, regardless of different handwriting styles. We emphasize here that our goal is not to define rotation invariant features, since we do not expect hanadwritten characters to have any larger degree of rotation. The features (h_{il}), (h_{il}), (v_{iu}) and (v_{ib}) describe in a rough way the outer shape of the letter, while (v_{im}) enables easier distinguishing between letters that could have similar outer shape in some handwritings (i.e. C and E; B and D).

B. *Selecting the classifier*

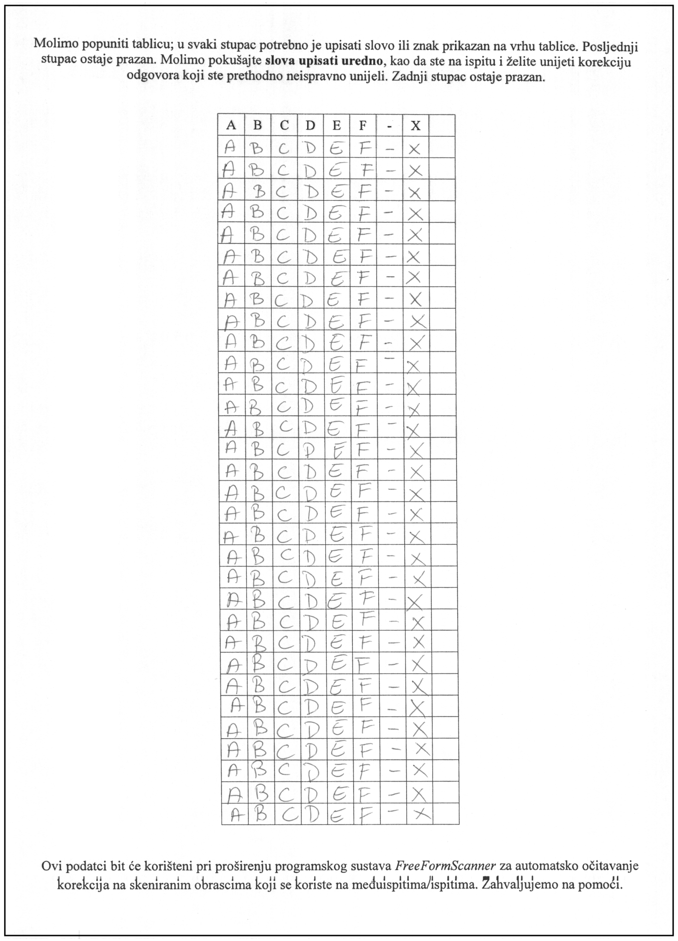
Given the desired performance requirements, we opted to use the random forest classifier proposed by Breiman [11]. The random forest classifier is reasonably fast to train, the classification results are fast to compute, and it offers performance comparable [12] to other popular classifiers such as support vector machines [13]. The main idea of the random forest approach is combining randomized decision trees into ensembles. When building a decision tree within a random forest, for each split at inner tree nodes, the algorithm randomly selects a subset of m attributes and then performs the split by the best attribute from the selected subset, based on the Gini impurity index. In contrast to SVM, which is posed as a binary classification algorithm, random forests inherently support multiple classes. Given a sample, each of the trees of the classifier casts a vote for one of the classes, and the final class of the sample is determined by the majority of votes. Tunable parameters of the random forest algorithm are the number of trees, the number of top candidates to consider (m), and the allowed depth of individual decision tree.

V. Experiments

For experimental validation of the proposed procedure, we collected two datasets. The training dataset contains handwriting samples from 289 individuals, and the test-dataset contains samples from 15 individuals. Each individual was asked to write all the considered characters (A, B, C, D, E, F, X, -) 32 times, using a specialized sheet as illustrated in Fig. 4. Hence, for each individual we collected a total of 264 samples plus the eight printed samples from the table header, resulting in the training dataset size of 76296 samples and the test set size of 3960 samples. There is no mixing of samples from the training and the test set, i.e. individuals contributing to the training set were not a part of the second group that contributed to the test set. Multiple samples per person were collected with the goal of improved learning, given the inevitable variations that occur within different instances of the same handwriting.

The edge distance representation was implemented in Java in a straightforward way, according to the described algorithm. For each scanned image, the table containing the handwritten letters was located using Hough transform; then the cells were located using intersections of appropriate border lines. The procedure for the minimum letter bounding box described in [4] was then applied, and the character image was extracted and further binarized. On the binarized image, the edge distance features were extracted.

For testing classification performance of the random forest classifier, we used the open source data mining tool Weka [14]. Weka offers a graphical user interface for experimenting



Molimo popuniti tablicu; u svaki stupac potrebno je upisati slovo ili znak prikazan na vrhu tablice. Posljednji stupac ostaje prazan. Molimo pokušajte **slova upisati uredno**, kao da ste na ispitu i želite unijeti korekciju odgovora koji ste prethodno neispravno unijeli. Zadnji stupac ostaje prazan.

A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	
A	B	C	D	E	F	-	X	

Ovi podatci bit će korišteni pri proširenju programskog sustava *FreeFormScanner* za automatsko očitavanje korekcija na skeniranim obrascima koji se koriste na međuispitima/ispitima. Zahvaljujemo na pomoći.

Fig. 4. An example of an annotation sheet used to collect handwritten character samples.

with various classifiers, but it is also available as a standalone Java library. Given that the existing automated grading system is written in Java, we expect that using Weka will ensure a seamless integration of the final classifier with the existing code.

A. *Classification results*

We experimented with several parameter settings, and found that the best performance is obtained by using a forest of 100 trees, considering $m = 6$ random attributes in each split, and not limiting the tree depth. Using this setup, the random forest classifier was able to correctly classify 96.5% of samples from the test set (3821 of 3960). The confusion matrix is shown in Table I. It can be seen that, as expected, most confusion occurs between visually similar letters. For example, 20 instances of the letter E were classified as C, although only four C's were classified as E's. It seems more common for the letter E to lack a middle dash, making it similar to C, than for the letter C to contain an additional dash that would make it similar to E. Other notable confusions include 18 instances of the letter B being classified as E, 16 instances of the letter E being classified as B, etc.

In order to better illustrate the amount of confusion among different classes, a visualization of the confusion matrix is

TABLE I. THE CONFUSION TABLE FOR THE TESTING SET. ROWS CORRESPOND TO CORRECT LABELS, AND COLUMNS TO THE NUMBER OF CLASSIFICATIONS AS A GIVEN SYMBOL OUTPUT BY THE RANDOM FOREST.

	A	B	C	D	E	F	-	X
A	483	2	1	5	0	1	3	0
B	5	454	0	14	18	3	0	1
C	0	1	487	0	4	1	2	0
D	1	9	3	479	1	0	2	0
E	1	16	20	0	454	3	0	1
F	2	2	0	0	2	489	0	0
-	0	0	0	0	0	1	493	1
X	11	0	1	0	0	0	1	482

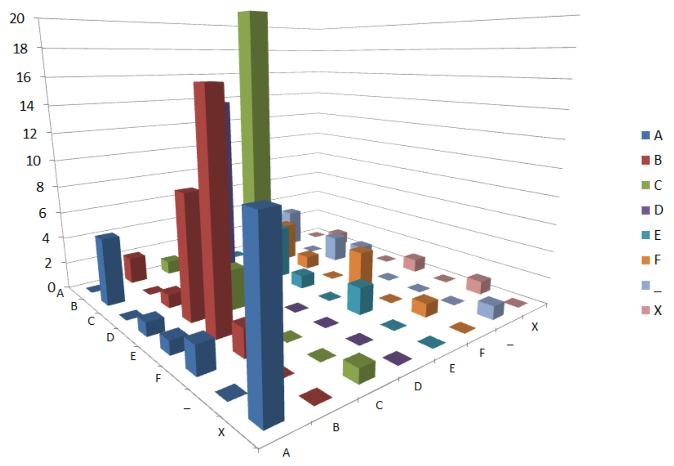


Fig. 5. Visualization of the confusion matrix without diagonal elements

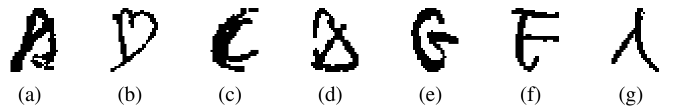


Fig. 6. Examples of misclassifications. From left to right: A classified as B, B classified as D, C classified as E, D classified as B, E classified as B, F classified as E and X classified as A.

shown in Fig. 5. In order to emphasize misclassifications, we removed the bars corresponding to correctly classified examples so that only misclassifications are left. The height of each bar corresponds to the number of misclassified examples at the corresponding location of the confusion matrix.

A few misclassified samples from the test dataset are shown in Fig. 6. The correct labels from left to right are A, B, C, D, E, F, X. Note that most of these examples are hard to classify even for a human. For example, the examplified letter B reminds of a heart symbol and in a certain way to letter D, the letter E looks like a G, the letter F could easily be confused for an E, and the character X reminds of the Greek letter λ.

B. Varying the decision threshold

As discussed before, the proposed system should provide a confidence measure of the predicted label, so that labels with low confidence can be presented to the human operator. In order to find the optimal classification certainty threshold below which manual operator intervention will be required, we have considered two facts:

(i) The classifier will inevitably produce a certain number of misclassifications that will be misclassified with a certainty greater than the confidence threshold; these will initially pass undetected and will require later intervention based on student complaints.

(ii) There will be some correctly classified letters for which the system will ask the operator to manually confirm the classification, because the classification certainty will be smaller than the confidence threshold; this scenario costs operator time.

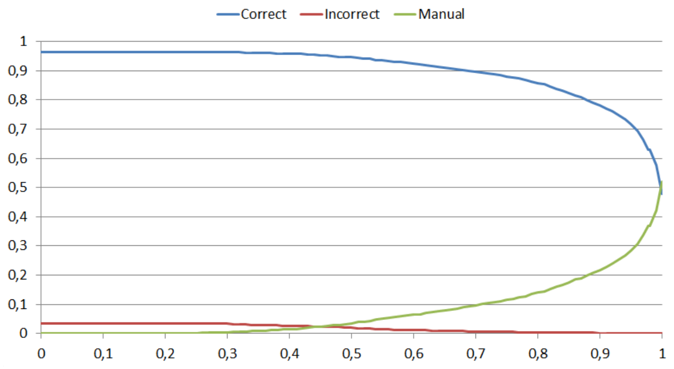


Fig. 7. The percentage of correct classifications, misclassifications and uncertain classifications requiring manual input, depending on confidence-level threshold (for the first method; for the second method the diagram is similar). X-axis corresponds to confidence-level thresholds, while y-axis represents the percentage of correct, incorrect and uncertain classifications.

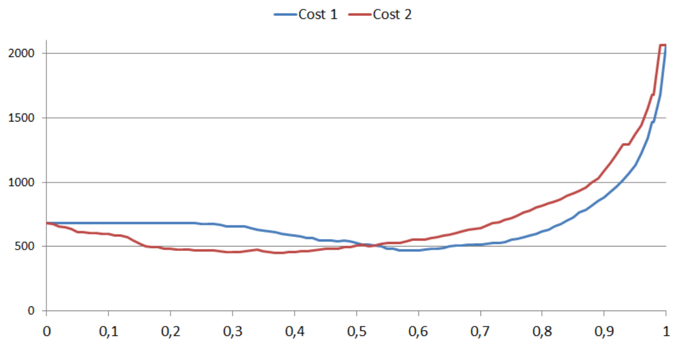


Fig. 8. Performance of penalty function. X-axis corresponds to confidence-level thresholds, while y-axis represents values of penalty functions for the two proposed approaches (less is better). *Cost 1* corresponds to method (i) and *Cost 2* to method (ii).

To tackle these two situations, we observe that it takes more time for course staff to manually inspect and correct missclasifications upon receiving a complaint by a student than at the time of sheet proccesing. Therefore, we define the following cost function: for each classification with insufficient confidence level we add a penalty of 1 point; for each undetected misclassification (misclassification with a confidence level greater than the threshold) we add a penalty of 5 points. Given this penalty function, we define two approaches to solving the problem of selecting the optimal threshold.

(i) In this approach we require that the absolute value of confidence for the predicted character class must be greater than θ. We evaluated the clasification performance for each distinct θ in $[0, 1]$ which occured as confidence of prediction for any character. With the change of threshold, the number of correct classifications, misclassifications and uncertain classifications changes, and is illustrated in Fig. 7. The performance of penalty function is presented in Fig. 8. As can be seen, for this case the optimal certainty-level threshold is 0.59 at which we have 92.8% correct classifications, 1.2% misclassifications and 6.1% manual interventions.

(ii) In this approach we require that the relative value of confidence for the predicted character class must be greater than θ. This means that the difference between the probability of the most probable class and the second most probable class must be greater than θ. We evaluated the clasification for each distinct θ in $[0, 1]$ which occured as confidence of prediction for any character. With the change of threshold, the number of correct classifications, misclassifications and uncertain classifications changes, and is similar to the one illustrated in Fig. 7. The performance of the penalty function is presented in Fig. 8. As can be seen, for this case the optimal certainty-level threshold is 0.37 at which we have 92.6% correct classifications, 1.0% misclassifications and 6.4% manual interventions.

VI. Conclusion and outlook

We have presented a technique based on edge distances and random forest classifier that can be used to improve an existing automated grading system by machine-based recognition of handwritten characters. In the existing setup, these characters represent answer alterations in case a student annotates an answer in a predefined answer matrix and then changes his mind. Extensive experiments were performed on a large dataset containing samples of eight supported characters. Experiments indicate that by introducing the proposed technique into the existing automated grading system, more than 92% of such answers could be machine-read, hence saving valuable operator time and further improving the automated grading process.

Acknowledgements

This work has been financially supported by the University of Zagreb, Croatia, under grant Research support 2, contract 2013-ZUID-19, and by the Research Centre for Advanced Co-operative Systems ACROSS (EU FP7 #285939). This support is gratefully acknowledged.

References

[1] M. Čupić and Ž. Mihajlović, "Computer-based knowledge, self-assessment and training," *International Journal of Engineering Education*, vol. 26, no. 1, pp. 111–125, 2010.

[2] Ž. Mihajlović and M. Čupić, "Software environment for learning and knowledge assessment based on graphical gadgets," *International Journal of Engineering Education*, vol. 28, no. 5, pp. 1127–1140, 2012.

[3] M. Čupić, "A case study: using multiple-choice tests at university level courses – preparation and supporting infrastructure," *International Journal of Intelligent Defence Support Systems (IJIDSS)*, vol. 3, no. 1/2, pp. 90–100, 2010.

[4] M. Čupić, K. Brkić, T. Hrkać, and Z. Kalafatić, "Supporting automated grading of formative multiple choice exams by introducing student identifier matrices," in *MIPRO, 2011 Proceedings of the 34th International Convention*, May 2011, pp. 993–998.

[5] N. Arica and F. T. Yarman-Vural, "An overview of character recognition focused on off-line handwriting," *Trans. Sys. Man Cyber Part C*, vol. 31, no. 2, pp. 216–233, May 2001. [Online]. Available: http://dx.doi.org/10.1109/5326.941845

[6] G. Vamvakas, B. Gatos, and S. J. Perantonis, "Handwritten character recognition through two-stage foreground sub-sampling." *Pattern Recognition*, vol. 43, no. 8, pp. 2807–2816, 2010.

[7] Ø. D. Trier, A. K. Jain, and T. Taxt, "Feature extraction methods for character recognition - a survey," *Pattern Recognition (PR)*, vol. 29, no. 4, pp. 641–662, 1996.

[8] J. H. Kim, K. Kyung, Kim, and C. Y. Suen, "Hybrid schemes of homogeneous and heterogeneous classifiers for cursive word recognition," in *In Proc. 7th International Workshop on Frontiers in Handwriting Recognition*, 2000, pp. 433–442.

[9] M. Mohamed and P. Gader, "Handwritten word recognition using segmentation-free hidden markov modeling and segmentation-based dynamic programming techniques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 5, pp. 548–554, May 1996. [Online]. Available: http://dx.doi.org/10.1109/34.494644

[10] O. Surinta, L. Schomaker, and M. Wiering, "Handwritten character classification using the hotspot feature extraction technique." in *ICPRAM (1)*, P. L. Carmona, J. S. Snchez, and A. L. N. Fred, Eds. SciTePress, 2012, pp. 261–264.

[11] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1010933404324

[12] A. Jović, K. Brkić, and N. Bogunović, "Decision tree ensembles in biomedical time-series classification." in *DAGM/OAGM Symposium*, ser. Lecture Notes in Computer Science, A. Pinz, T. Pock, H. Bischof, and F. Leberl, Eds., vol. 7476. Springer, 2012, pp. 408–417.

[13] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: http://dx.doi.org/10.1023/A:1022627411411

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278