Supporting automated grading of formative multiple choice exams by introducing student identifier matrices

Marko Čupić, Karla Brkić, Tomislav Hrkać, Zoran Kalafatić
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb
e-mail: {marko.cupic | karla.brkic | tomislav.hrkac | zoran.kalafatic}@fer.hr

***Abstract*—We propose a solution which enables automated attribution of formative exams to students. We define the student identifier matrix, which is a graphical representation of a student ID number. The matrix is annotated by the student during the exam. We introduce a robust image processing procedure which enables automated recovery of the student ID from the matrix. We show that the proposed procedure achieves a recognition rate of 100% on 296 samples in presence of skewing, rotation and offsets introduced by the printing and the scanning processes.**

I. Introduction

Multiple choice exams [1, 2, 3] are often used for a knowledge assesment in the university environment. Automated grading of multiple choice exams is a method of choice for university courses with a large number of students. Existing solutions [4, 5] support automated grading of summative exams, i.e. regularly announced exams which students typically take three times in a semester. In case of summative exams, attribution of an exam to a student is easy, as exam forms are printed with unique bar codes which represent individual students. However, there are scenarios where such an easy solution is not possible. In the context of formative exams – unannounced short exams which the teacher periodically gives to the attending students – it isn't practical to print individual student bar codes on every exam form. Formative exams need to be held in as short time as possible (typically 5-10 minutes), so that the teacher can have enough time to deliver the regular lecture. The process of handing out the exams to every student by name would add a significant overhead to the total time taken by the exam. Therefore, the formative exams are usually graded by hand, which is very cumbersome and prone to human error, especially for courses with hundreds of students. To speed up the grading process, usually just one or two variants (groups) of the exam are offered. On the other hand, exams graded automatically can be completely individualized. The individualization means that each student participating in the same exam can have their exam questions drawn from a set of possible questions and ordered in a specific way, so no two students have the same exam. The problem here is how to allow students to enter their student ID numbers by hand, while retaining automatic readability. An existing approach using the 7-segment digits is shown in [6].

We propose a novel approach which supports automated grading of formative exams. We introduce the student identifier matrix, which is a graphical rendition of the unique student number each student is given upon enrollment. The student identifier matrix is printed on the exam form. An example of such an exam is shown in Fig. 1. The student identifier matrix is printed in the upper right corner. The student then annotates the rows and columns of the matrix corresponding to her student ID number. We propose a robust image processing method which detects the student identifier matrix on the exam form and converts it into a numerical representation of the student ID number.

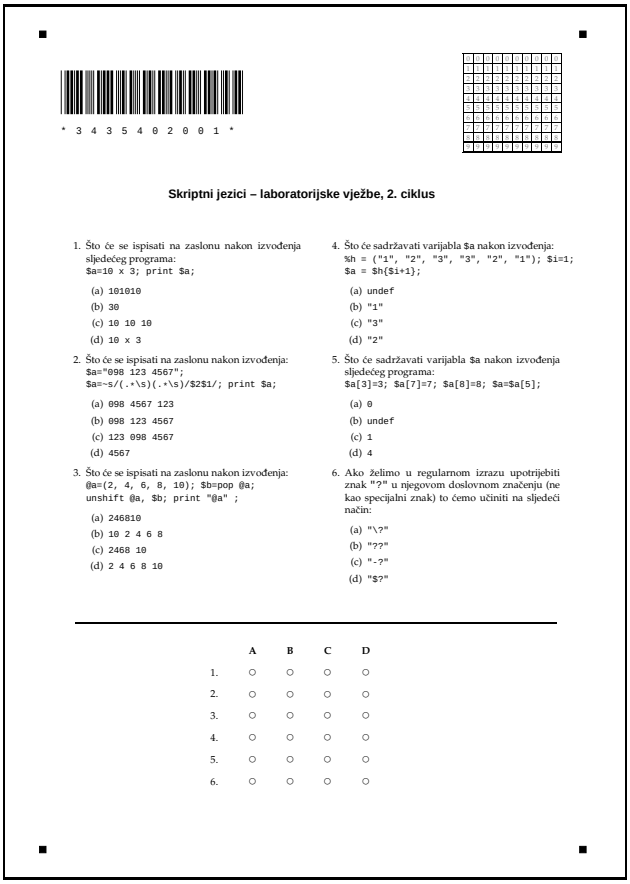
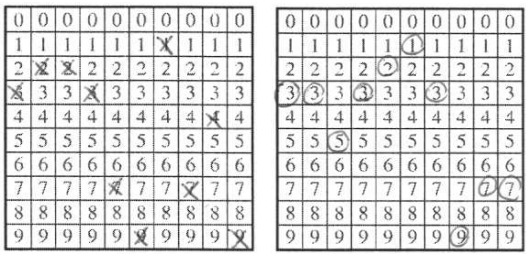


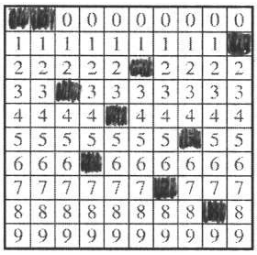
Figure 1: A sample of a formative multiple choice exam with a student identifier matrix.

II. Reading student ID numbers

In this paper, we assume a setup where exam forms are printed, the answers are marked by the student, and the forms are scanned and automatically graded. We use an existing system for automated grading described in [4]. In order for the automated grading to work, we need a way to establish a correspondence between an exam form and



(a) using crosses (b) using circles



(c) fully filled

Figure 2: Three typical ways to specify student ID

a student. We do this by using student identifier matrices.

A. The student identifier matrix

The student identifier matrix is a table of 10 rows and the number of columns equal to the length of the student ID number. Each column represents possible values of one digit of the student ID number. The number of rows is 10 as there are 10 possible digit values (0-9). The idea is that the student annotates a cell in each column corresponding to the appropriate digit in her student ID. To make the use of the student ID matrix easier, each cell is pre-printed with the value it corresponds to. We consider three possible ways of annotation: crossing, circling and filling. Three examples of annotated student identifier matrices are shown in Fig. 2. Corresponding student ID numbers are 3223791749, 3353213977, and 0036427581, respectively. The case in Fig. 2c is easiest to correctly recognize. However, the developed recognition procedure is robust enough to correctly handle cases illustrated in Fig. 2a and 2b.

B. Detecting the matrix

In order to detect the student identifier matrix, we make use of several image processing techniques. The student identifier matrix is printed on the exam form and the approximate location of the matrix is known in advance. However, the precise location needs to be verified because both the printing and the scanning process can introduce some rotations and offsets of the complete exam form. Considering that the matrices will be used on exams in courses with a large number of students, we require the detection algorithm to operate very fast, as in addition to scanning the student ID number, the student's solutions of the exam also need to be scanned and processed.

The process of detecting the student ID matrix begins by manual annotation of the approximate area of the matrix on a single exam form example. The user needs not annotate the area with much precision, as it is only necessary to exclude all surrounding text. For each subsequent form, we assume that the matrix is located in the same approximate area which was annotated on the first form.

To determine the student ID number, we need to obtain the precise location of the matrix. The exam form is first cropped to contain only the approximate area where the student ID matrix is expected. In order to determine the precise location of the matrix, the obtained image is divided into a regular grid of predefined size. To find the left and the right edge of the student ID matrix, we examine each horizontal line of the grid and find minimal and maximal value of x for which the image pixels are black. We thus obtain a number N of candidate points for the left and the right edge of the matrix. Analogously we obtain the N candidate points for the top and the bottom edge. Using linear regression, we find the line equations of the outer edges of the matrix.

To verify whether an individual cell was annotated, we also need to determine the equations of the horizontal bounding lines, which have the form given by Eq. (1):

$$y = ax + b \tag{1}$$

Having collected N candidate points for the top line and the bottom line, we define sum-of-squares error for each line:

$$E = \sum_{i=1}^{N} (ax_i + b - y_i)^2 \tag{2}$$

We try to find coefficients a and b which minimize the error defined by Eq. (2). By requiring Eq. (3) and Eq. (4) to hold:

$$\frac{\partial E}{\partial a} = 2 \sum_{i=1}^{N} (ax_i + b - y_i) x_i = 0 \tag{3}$$

$$\frac{\partial E}{\partial b} = 2 \sum_{i=1}^{N} (ax_i + b - y_i) = 0 \tag{4}$$

one can simply obtain equations (5) and (6):

$$a = \frac{N \sum_{i=1}^{N} x_i y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{N \sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2} \tag{5}$$

$$b = \frac{1}{N} (\sum_{i=1}^{N} y_i - a \sum_{i=1}^{N} x_i) \tag{6}$$

The similar method can be employed to find the left and the right vertical bounding lines of the matrix. Equation (1) for obvious reasons can not be used for vertical lines, so the only needed modification is to start with equation (7):

$$x = ay + b \tag{7}$$

Then, we define the sum-of-squares error function (equation (8)):

$$E = \sum_{i=1}^{N} (ay_i + b - x_i)^2 \tag{8}$$

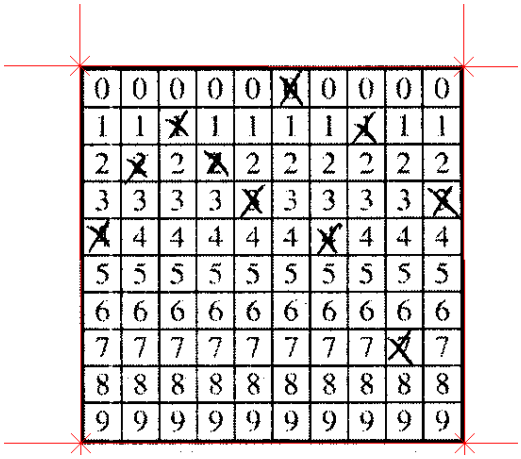


Figure 3: The detected bounding box of the student ID matrix. Corners are annotated with crosses.

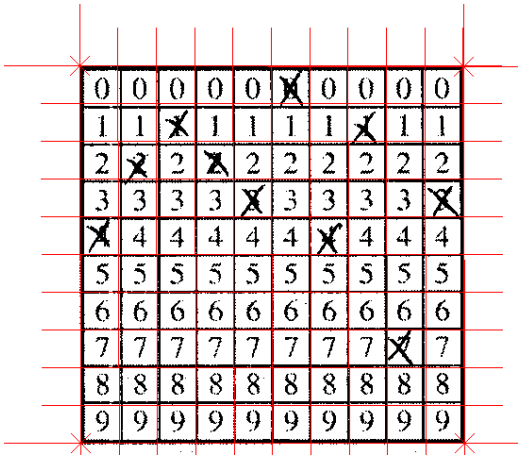


Figure 4: Rows and columns detected by using simple detection.

By requiring equations (9) and (10) to hold:

$$\frac{\partial E}{\partial a} = 2\sum_{i=1}^{N}(ay_i + b - x_i)y_i = 0 \tag{9}$$

$$\frac{\partial E}{\partial b} = 2\sum_{i=1}^{N}(ay_i + b - x_i) = 0 \tag{10}$$

one can simply obtain equations for coefficients a and b, which are given as equations (11) and (12):

$$a = \frac{N\sum_{i=1}^{N} x_i y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{N\sum_{i=1}^{N} y_i^2 - (\sum_{i=1}^{N} y_i)^2} \tag{11}$$

$$b = \frac{1}{N}(\sum_{i=1}^{N} x_i - a\sum_{i=1}^{N} y_i) \tag{12}$$

Corners of the matrix are then found as intersections of the edge lines. An example of the student ID matrix with annotated edges and corners found by the described algorithm is shown in Fig. 3.

Each outer edge of the matrix can be described by a vector. In order to find the corners of an individual cell, we need to move for some amount in a single horizontal and a single vertical direction. However, in a general case we allow the matrix to be slightly skewed, so the directions of the pairs of horizontal and vertical edges might differ, even though the pairs are in reality parallel. To overcome this, we calculate the average vector from each pair, thus obtaining the average horizontal and the average vertical direction. Using the obtained vectors, we can find the inner cells of the matrix by assuming that the matrix has 10 rows and the number of columns equal to the length of the student ID number.

Since the student matrix can be poorly prepared, additional processing is required to better locate each cell. The possible problems are matrices in which not all rows have precisely equal height, or matrices in which not all columns have precisely equal width. Also, printing the matrix with the thick outer border results in easier matrix detection, but it can degrade the performance of cell detection. This is illustrated in Fig. 4, where we assumed that all columns are equally wide and that all rows have equal height. Notice that the first inner vertical line found by the algorithm does not correspond to the actual inner line which is offseted to the right by several pixels.

Beside the previously mentioned first inner vertical line, in Fig. 4 we can see several other misaligned horizontal and vertical lines. To fix the fact that the method can result with incorrect row and column lines, we developed a correction procedure which is applied for each row and column line.

The idea of the correction procedure for horizontal lines is illustrated in Fig. 5. The given matrix represents a 5-digit student ID. The position of the fourth horizontal line is distorted. The dashed line represents the position where we expect the fourth line to be. If we denote the top left corner of the matrix with $\vec{T}_0$, the average horizontal vector with $\vec{h}$ and the average vertical vector with $\vec{v}$, we can define the expected start of the fourth line by:

$$\vec{H}_4 = \vec{T}_0 + \frac{4}{10} \cdot \vec{v} \tag{13}$$

Then, each point vector on that horizontal line can be expressed as:

$$\vec{P} = \vec{H}_4 + \lambda \cdot \vec{h} \tag{14}$$

In order to perform the correction, we start by sampling points on the horizontal line. For example, let us assume that we selected a point $\vec{T}_S$ which lies on the expected horizontal line. Starting from this point, we search up and down (by changing the y-coordinate) for the closest black pixels. For the case presented in Fig. 5, the closest black point will be several pixels above the point $\vec{T}_S$. If, after the first black pixel there are more black pixels in continuum, we will set the components of the point vector $\vec{T}$ to $(\vec{T})_x = (\vec{T}_S)_x$ and $(\vec{T})_y$ equal to the average of the y-coordinates of first and last continous black pixel.

Let us observe that the equation (15) then holds:

$$\vec{T} = \vec{H}_4 + \lambda \cdot \vec{h} + \mu \cdot \vec{v} \tag{15}$$

where $\mu \cdot \vec{v}$ is the shift by which $\vec{H}_4$ should be corrected in order for T to lie on the line. We note the found

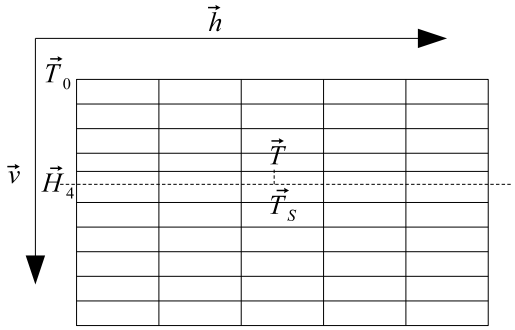


Figure 5: The correction procedure for horizontal lines.

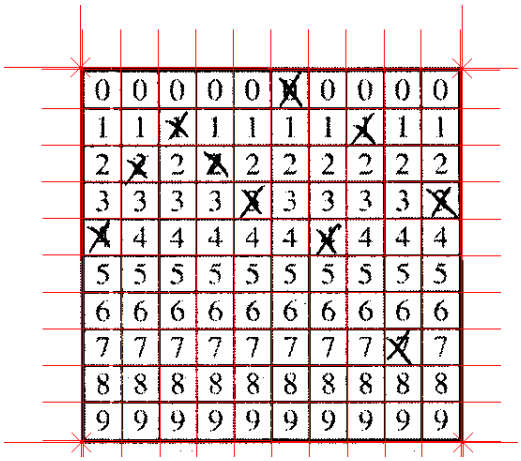


Figure 6: Rows and columns detected by using simple detection in conjunction with correction.

μ and repeat the procedure for several more times for new samples along the expected horizontal line, each time obtaining a new value for μ. Finally, for the correction, we take the median of all found values for μ, and using that value we correct $\vec{H}_4$. Vertical lines are corrected in similar way (by fixing y-coordinate and searching for the nearest left or right black pixels). Figure 6 shows the horizontal and the vertical lines after the correction. In comparison with the Fig. 4, we observe that there are no significant misalignments of the horizontal and the vertical lines.

Having found all the horizontal and the vertical lines, we can find the individual cells. Each cell is decribed by four points denoting its corners. The corners of each cell are determined as the intersections of apropriate horizontal and vertical lines.

C. Determining the student ID number

Having found the image area corresponding to the individual student ID matrix cell, we can determine whether the cell was annotated by the student. We first consider the following simple procedure: For each cell, we count the total number of black pixels. Then we calculate the ratio between the number of black pixels and the total number of the pixels in the cell. If the ratio is greater than some chosen threshold (let us denote it by *selection threshold*), we consider the cell annotated. To obtain the student ID number, we find the annotated cell in every column. Here we assume an error-free case, where only a single cell in a column can be annotated. As each cell

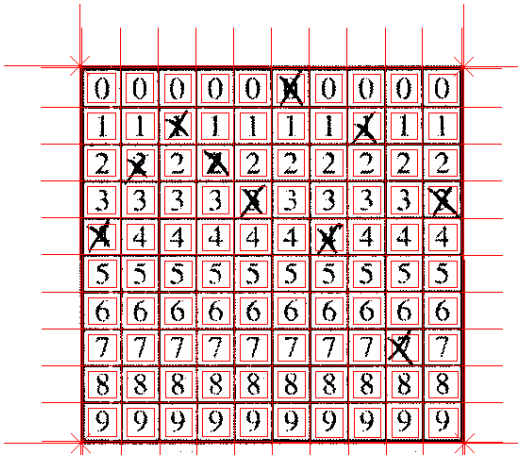


Figure 7: Adding margins to cells for better detection.

corresponds to exactly one digit of the student ID number, by repeating the procedure for every column, we get the complete number. In case more than one cell per row is found, the system will raise an alert to the human operator.

The described procedure is quite efficient for the case when the cells are annotated by filling, as illustrated in Fig. 2c. However, for cases presented in Fig. 2a and 2b, problems may occur due to possible matrix rotation and the fact that the student did not sufficiently fill the cell. To tackle this, we modify the procedure as follows. If, in some column, there is no cell having the ratio of the black pixels larger than the selection treshold, we find the cell with the largest black pixel ratio (the *candidate* cell). Then, we look at the ratios for cells in the same row but in other columns. We only analyze the cells for which we are certain that the student did not select them. If our candidate cell has the black-pixel ratio that is significantly larger than the other unselected cells (regulated by another theshold), we assume our candidate cell to be selected.

To make the recognition process more robust, we must observe that often when recognition is unsucessful, the reason is the influence of the unstable thickness of cell borders and of skewing and rotation. To tackle this, we consider only a part of the detected cell. The area is defined as a maximum rectangle that can be fitted to the cell. This rectangle is then trimmed for a percentage of its width and height. If we have a case illustrated in Fig. 2a, we can allow for somewhat larger margins (e.g. 15%), as illustrated in Fig. 7. On the other hand, if the student filled the matrix as in Fig. 2b, using such margins will elliminate a significant amount of the student generated circles, so the recognition will be unsuccessful. This raises the question of determining the appropriate margins.

Taking all of the described cases into account, our final recognition procedure is as follows.

1) Find horizontal and vertical bounding lines.
2) Find row and column lines.
3) Perform correction of row and column lines.
4) Find cell edges as intersections of appropriate row and column lines.
5) Start the recognition procedure using the whole detected cell. If successful, return the result.
6) Start the recognition procedure using the detected

cells with 20% margins. If successful, return the result.

7) Start the recognition procedure using the detected cells with 10% margins. If successful, return the result.
8) Otherwise show the matrix to the human operator and ask for help.

D. More robust matrix border detection

In order to detect the matrix border lines, we make use of several scan-lines from left, right, top and bottom and follow these lines until first black pixel is detected. Using this technique, we obtain four sets of points, each of which contains points sampled from one border line. However, due to scanner introduced distorsions or student generated artefacts, some of the points can be outliers, and in presence of outliers the lines generated by the sum-of-squares minimization can have signifficant offset from actual border lines.

To cope with this, the number of scan-lines should be adequately large, and a robust procedure for model parameter estimation should be employed. With the increase of the required number of scan-lines come performance penalties, so the actual number should be kept as small as possible, but large enough to provide robustness. In our approach, we use 20 scan-lines for both the horizontal border lines and the vertical border lines. Among the point samples, we determine the median point and then remove all points on a sufficiently large distance (the median for the x and y coordinates are calculated separately). This procedure serves as a simple outlier removal tool. Having removed obvious outliers, sum-of-squares minimization can be employed to determine the correct parameters.

Further improvement can be obtained by using a parameter estimation procedure that is more robust than sum-of-squares minimization. One simple solution can be the usage of *RANSAC* algorithm proposed by Fischler and Bolles [7].

III. Experiments

To test the described method, we collected a set of 148 annotated student ID matrices. The matrices were annotated by five different people, using filling, crossing and circling as annotation methods. No special care was given to produce tidy annotations, as we tried to simulate how a student might annotate the matrix. We experimented with scanning resolutions of 200 and 300 DPI, giving us a total of 296 samples. Using the described procedure and the selection threshold of 0.5, we were able to achieve 100% recognition rate.

To further explore the impact of the selection threshold on the recognition performance, we repeated all of the experiments for the selection threshold values from 0 to 1 with the step 0.04. The results are presented on Figures 8 and 9.

As can be seen, for too small values of selection threshold, recognition is unsuccessful, since in each column more than one cell becomes selected, which leads to a situation in which there are multiple candidates for a digit. Since we do not have a recovery procedure, these situations are fatal for recognition.

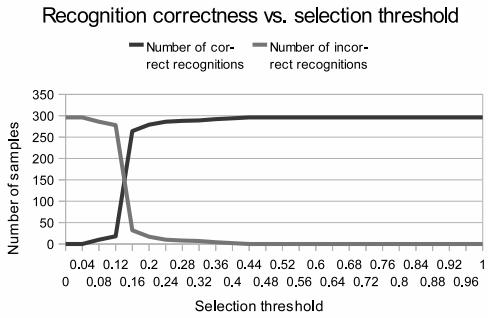


Figure 8: Performance of the proposed method.

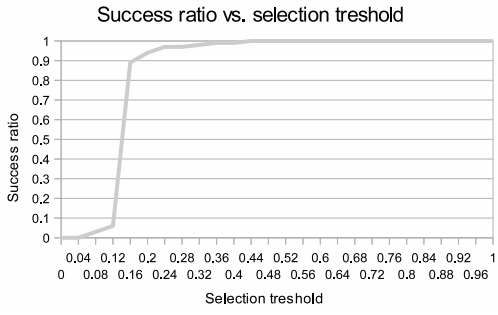


Figure 9: The success ratio of the proposed method (the number of correctly recognized samples divided by the total number of samples).

By increasing the selection treshold, the success ratio also increases. For the selection threshold greater or equal to 0.44, success ratio is 100%. If the recognition procedure were guided only by the selection threshold, after some additional threshold increase, the success ratio would start to decrease, since no cell candidates would be detected. However, as we previously described, situations when no cell candidates are detected are handled by computationally more expensive procedure of selecting the cell with the highest black-pixel ratio and comparing that ratio with the other unselected cells in the same row to see if enough evidence can be collected to declare the candidate selected. It is this fallback mechanism which is responsible for the success ratio retaining its constant value of 100% with the further increase of selection threshold.

Based on these observations, we can take the selection threshold of 0.5 to be near-optimal choice by which many cells are correctly recognized based only on the selection threshold, and only a small number of cases requires activation of the fallback procedure.

To determine the influence of auto-margin adjustment (*AM-procedure*) and evidence-seeking for selecting candidate cells (*ES-procedure*) on the recognition success ratio, we repeated the experiments with the procedures turned on or off. Results when both of the procedures are used are already presented on Fig. 9. The situation for both procedures turned off, so that the recognition is only steered by the selection threshold is presented on Fig. 10. For the total range of selection threshold from 0 to 1, the success ratio is never 100%. It reaches its maximal value of 72% for the selection threshold of about 0.36 and then

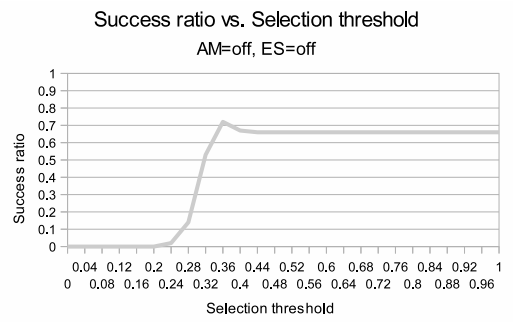


Figure 10: The success ratio of the proposed method for AM = off, ES = off.

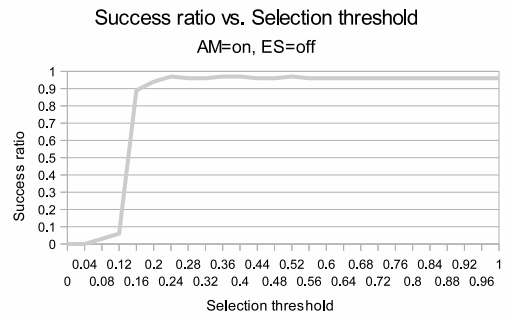


Figure 11: The success ratio of the proposed method for AM = on, ES = off.

starts to decline.

If we use only the AM-procedure, the success ratio starts to rise rather early, and it reaches a maximal value of 97% for the selection threshold of 0.24 after which it oscillates for some time and then drops to 96%. For this situation the success ratio never reaches 100%.

If we use only the ES-procedure, the success ratio starts to rise from the selection threshold of about 0.30, and it reaches maximal value of 96% for the selection threshold 0.44 after which it remains stable. For this situation the success ratio never reaches 100%.

As these experiments show, only by using both procedures we can obtain recognition success ratio of 100%.

The proposed method was implemented in Java within an existing system for automated grading [5].

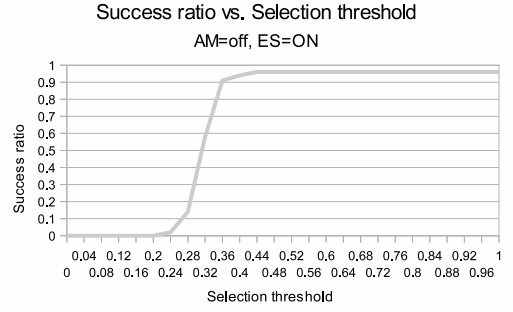


Figure 12: The success ratio of the proposed method for AM = off, ES = on.

IV. Conclusion and Future Work

In this paper we proposed the usage, the design and the automatic recognition method of student ID matrices. Using this approach, formative assessments during lectures can be completed in relatively short amount of time, while assessments remain automatically processable.

We tested the procedure on images obtained by two scanning resolutions and several annotation methods. Using a simple selection threshold method in conjuction with two fallback procedures we were able to obtain the recognition ratio of 100% .

It still remains to see how this matrix will be accepted by students, and how intuitive it will be. This will be tested in class during the second semester of the academic year 2010/2011.

References

[1] T. Haladyna, *Developing and Validating Multiple-choice Test Items*. Mahwah, New Jersey: Lawrence Erlbaum Associates, 3rd ed., 2004.

[2] K. Woodford and P. Bancroft, "Multiple choice questions not considered harmful," in *Proceedings of ACE 2005, Australian Computer Society*, pp. 109–116, 2005.

[3] A. Rhodes, K. Bower, and P. Bancroft, "Managing large class assessment," in *Proceedings of the sixth conference on Australian computing education*, pp. 30:285–289, 2004.

[4] 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, pp. 90–100, 2010.

[5] M. Čupić, J. Šnajder, and B. Dalbelo Bašić, "Post-test analysis of automatically generated multiple choice exams: a case study," in *Proceedings of ICL 2009*, pp. 1(10)–10(10), 2009.

[6] I. Bonačić, T. Herman, K. T., E. Mangić, G. Molnar, and M. Čupić, "Optical character recognition of seven–segment display digits using neural networks," in *Proceedings of MIPRO 2009 - 32st International Convention on Information and Communication, Technology, Electronics and Microelectronics*, pp. 323–328, 2009.

[7] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography.," *Communications of the ACM*, vol. 24, pp. 381–395, 1981.