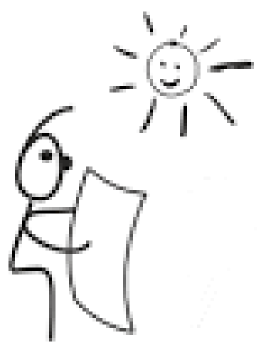
Checkbox Recognition

Matija Šuković, Montenegro
Mahdad Esmaili, Iran
Dávid Tóth, Hungary
Nikolina Dakić, Serbia



Summer School of Image Processing 2022
University of Szeged, Department of Image Processing and Computer Graphics
July 2022

Contents

1 Introduction

This is a report containing all the details on the SSIP 2022 project 'Checkbox recognition': Tasks of the project, the assigned team, and all the steps taken towards the final product.

1.1 Project description

When our team was assigned this project, we received a collection of binary images of scanned documents. All of them contain, among other elements, a number of checkboxes. Our task was to detect said checkboxes, and to compute and display whether they are checked or not. We were also told some additional instructions:

- The sides of the boxes could be anywhere between 25px and 75px long, and the width of a box can be up to 50% longer than its height.
- The boxes are parallel to the paper's edges, but a few degrees of skew is possible on the scanned images.
- The boxes can be unfilled or filled with either a checkmark (tick) or an X. It is not important to detect which type of filling is used.
- Some documents are poorly scanned, causing their checkboxes to have gaps up to 15px wide. Processing checkboxes with gaps larger than 5px is optional.
- Some of the samples have "shadowed" checkbox edges (simulating a 3D box). Processing them is optional.

1.2 Team members

Our team consists of four members:

- Matija Šuković, from Montenegro;
- Mahdad Esmaili, from Iran;
- Dávid Tóth, from Hungary;
- Nikolina Dakić, from Serbia.

2 Development of the project solution

Our team spent the first two days of the summer school on research. There are many ways to approach a problem such as ours, and we tried out a few different techniques for detecting objects in an image. We have discussed several methods with the summer school lecturers, most notably with Csaba Beleznai, who suggested an approach demonstrated in the article [1]. This method would use the symmetric nature of the checkbox's shape to find their centroid locations in the image. While it would be a very robust end solution, it would need a lot of work until it becomes efficient for our use case. Due to the short time that we had at hand, in the end we decided to turn to the simplest possible approach that would yield satisfying results.

2.1 First version

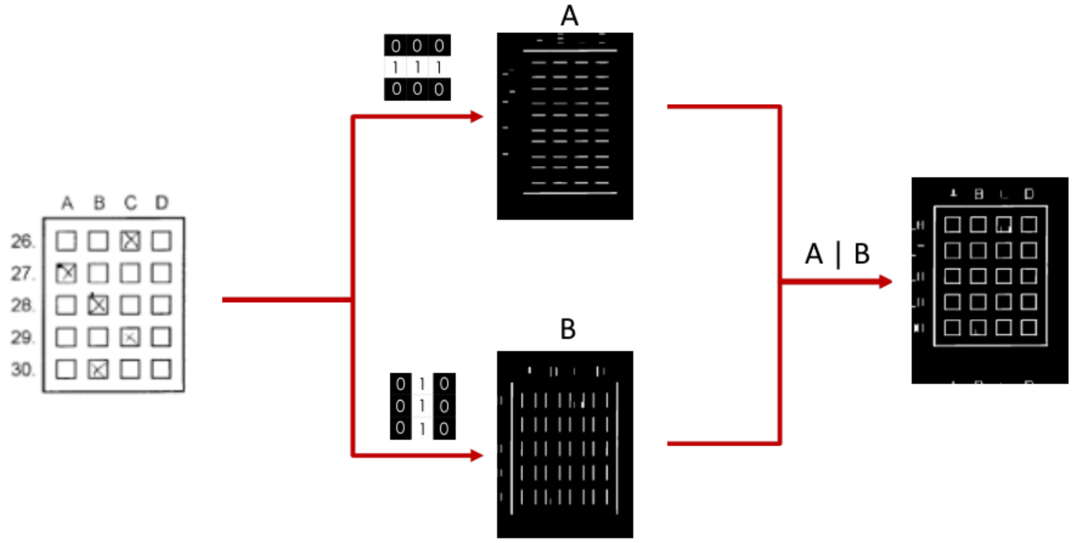
We chose to use edge detection to find all rectangular components in a given image, after which we would pick out the components that could represent a checkbox.

2.1.1 Preprocessing

Our program will first load an input image and apply some preprocessing to it. The image needs to be grayscaled, so that we can apply thresholding and get an image containing only black and white pixels. We do this in order to make it easy to tell what is the background, and what is the text. The provided samples are already binarized, but this preprocessing will allow for any image to be examined for checkboxes. We will also invert the color of the image, because we will later use an OpenCV function to analyse connected components. Said function considers black pixels to be the background.

2.1.2 Edge detection

After applying preprocessing to the input image, we next analyse it using convolution kernels, one that will filter out only the horizontal lines of the image, and one that will filter out the vertical ones. We then merge the two images into one, resulting in an image containing all the rectangular components of the original. The process is demonstrated in the picture 1.

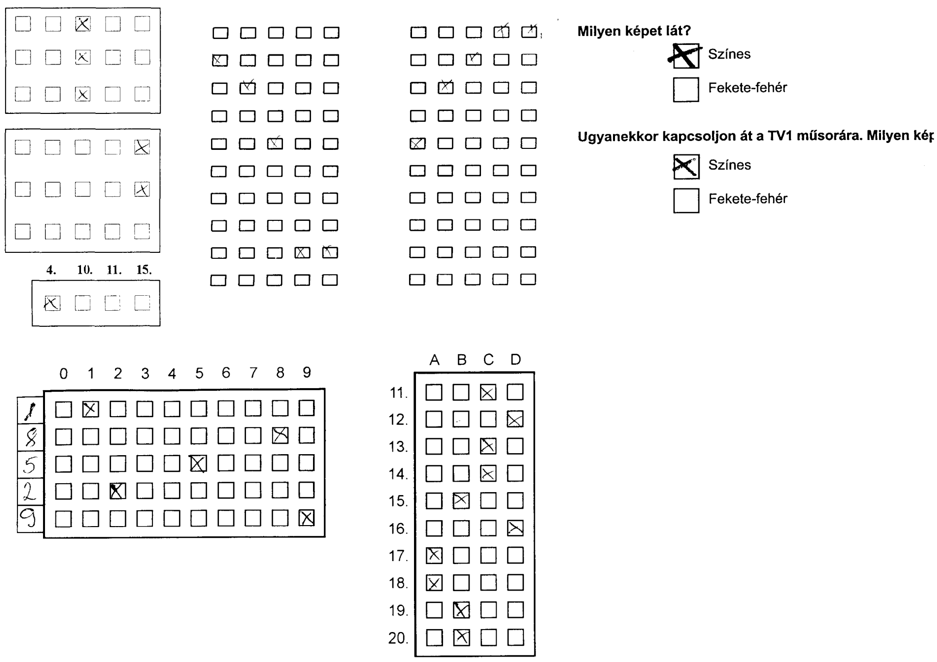


Picture 1: Isolating rectangular components from an image

2.1.3 Finding checkboxes

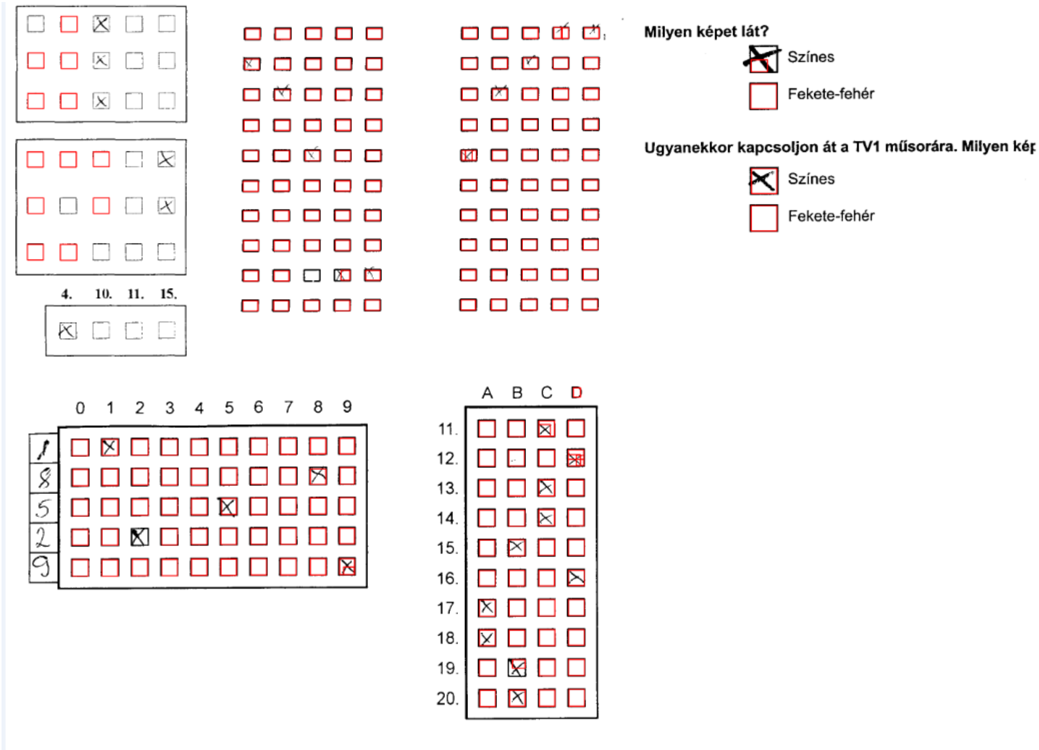
Now that we have a map of all the rectangular components, we can go through it and note all the components that form a closed rectangle. OpenCV has a great function that will do the hard work for us, and also provide us with additional information for each component it finds. The function is called connectedComponentsWithStats. It will return a list of connected components and a list of stats for each one, including the width and height of the component. We can use this information to isolate components which could be checkboxes, since we know their minimum and maximum width and height, as well as their possible aspect ratios.

Picture 2 will be our test image further ahead. It is a combination of several types of checkboxes. We will be applying our algorithm on it to demonstrate the progress we made as we kept working on our program.



Picture 2: Test image

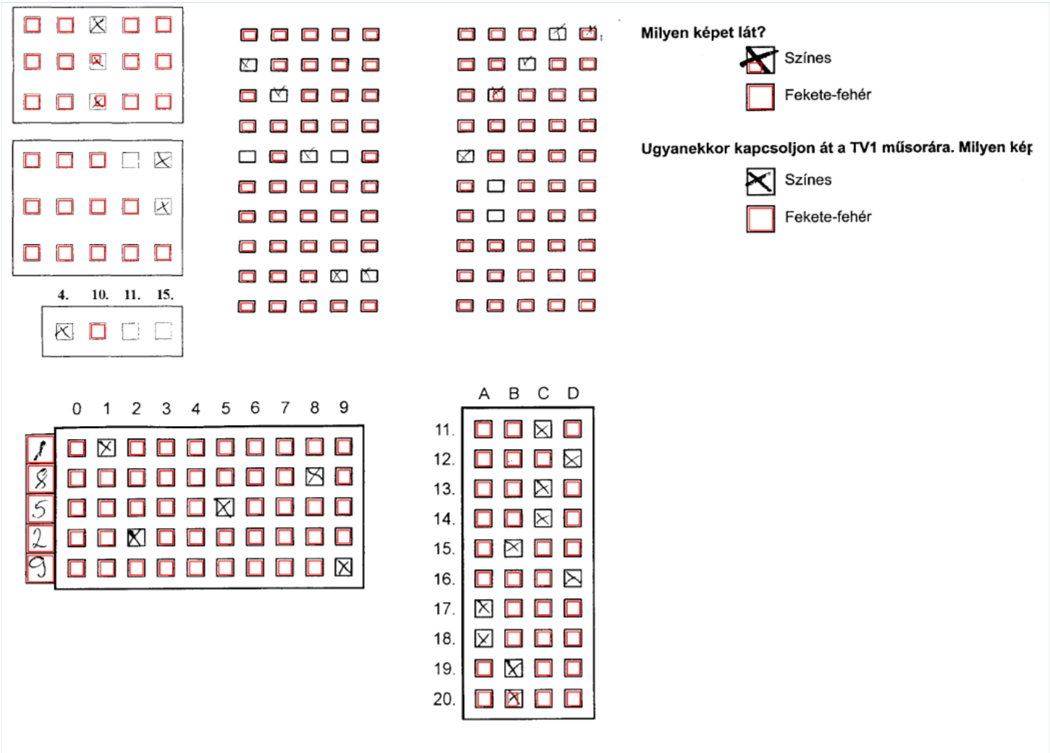
Picture 3 shows the result of the program's first version, whose inner workings we just described. It can already detect a good amount of checkboxes, but problems become obvious when we look at the top left corner. Those checkboxes are of poor quality and have a lot of gaps. Since they are not connected components they will be completely ignored by our program. The next step would be to find a way to fill small gaps, so that we could detect checkboxes in cases like this.



Picture 3: Test image, after applying the first version of our program

2.2 Mind the gap

To tackle the problem of gaps, we applied a series of morphological transformations to our image map. With a combination of closing, dilation and erosion morphology, we managed to connect checkboxes that have gaps of up to 10px. Picture 4 shows the results of an early attempt at fixing broken checkboxes. It is included here to demonstrate problems that we encountered and later had to solve. We see great improvements in the top left checkbox detection. However, something wrong is going on with the rest. There are many missed boxes, and for a reason. Blurring the image too much will cause components to connect at unwanted places, thus being ignored by our filter for being too small or a wrong aspect ratio. Other parts of the image map will also connect with one another which is not something that we want, since it can cause components to appear where there are none in the original image, leading to 'false alarms'.

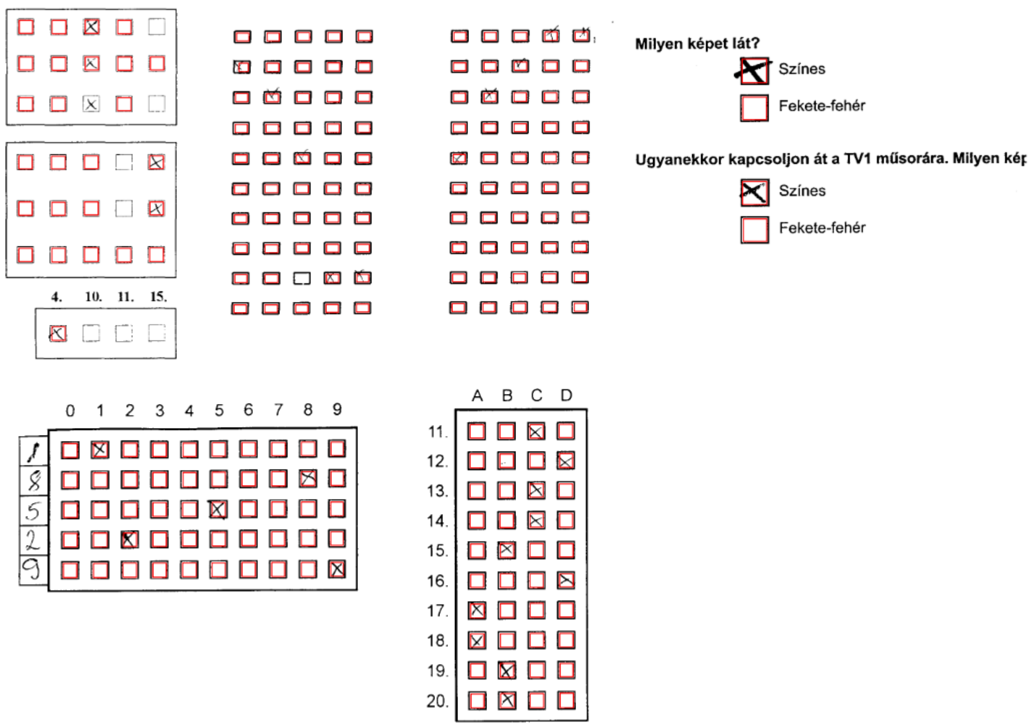


Picture 4: Early attempt to fix broken checkboxes using morphology

2.2.1 Refinements

We did a number of improvements to the line detection algorithm since our first attempt. Firstly, we updated the thresholding algorithm to an adaptive version, which will use different thresholding values for different regions of the image. Next, we tweaked our convolution kernels, so that they pick up less random noise and more of the straight lines. Then, we added a procedure to detect large rectangular components and remove them from the image map, which reduced the possibility of unassociated lines forming a closed component. Finally, we included an input query, so that the user can enter the rough size of the checkboxes for the given image. This will drastically improve the detection ability of our program, since we can limit the morphological transformations when working with smaller checkboxes, and thus create less unwanted connections.

Picture 5 shows the result the refined algorithm will give. Note that, in this case, the program was applied to each of the different checkbox batches separately, since for each batch we used a different rough checkbox size for the input. This is the final version of our checkbox detecting algorithm.



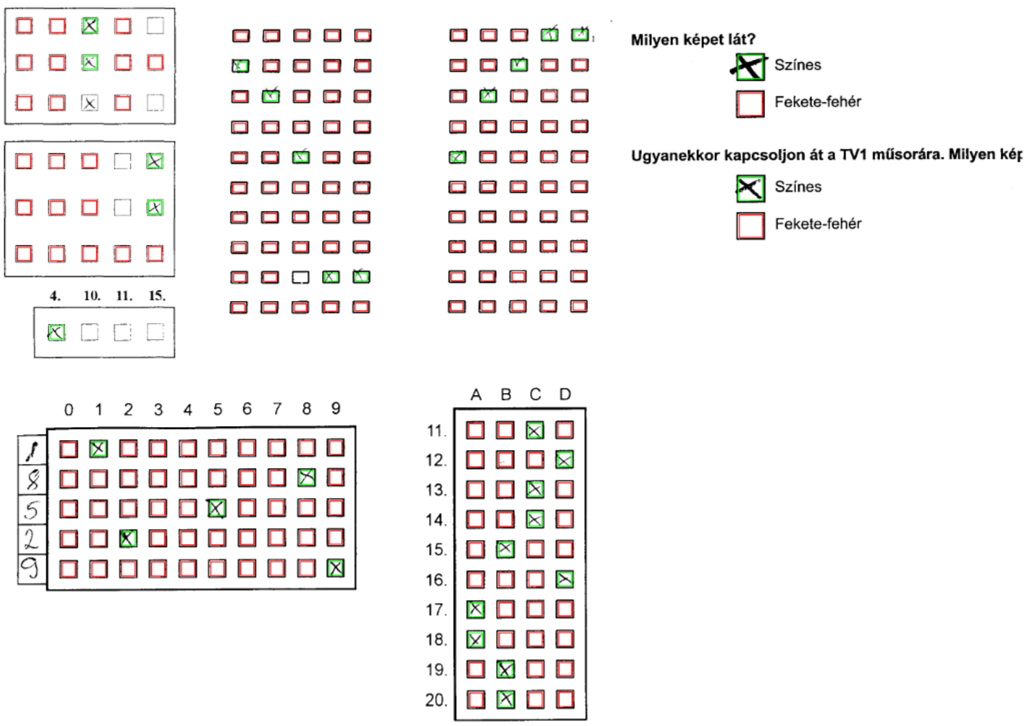
Picture 5: Final version of the checkbox detection algorithm

3 Checkmark recognition

With the checkbox detection covered, it was time to start analysing found boxes. For this we, once again, decided to go for the simplest approach, visualised in picture 6. For each detected checkbox, we take a look at the histogram of their insides. To put it simply, we compare the number of white pixels with the number of black ones. If the number of black pixels is not too small, nor too high, we say that the checkbox is checked. The upper threshold exists to address the case when the checkbox was scribbled over. We should assume that the person filling the form first checked the box, and then changed their mind. Thus, the box should be registered as not checked.

Picture 6: Determining the state of the checkbox depending on its histogram

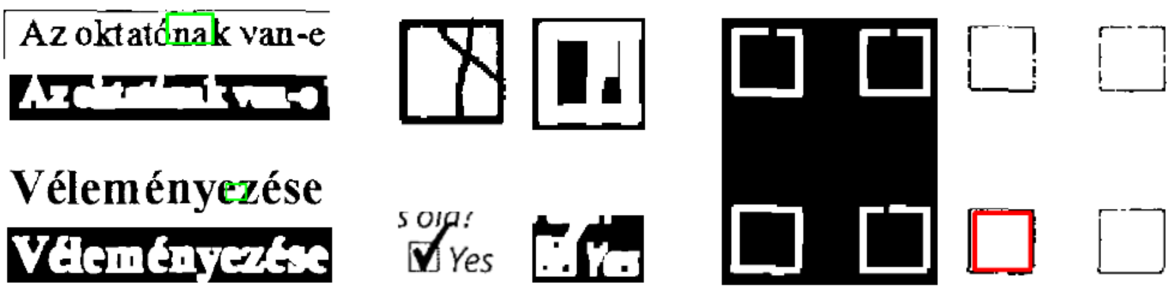
On picture 7 you can see the checkmark recognition in action.



Picture 7: Demonstrating the checkmark recognition algorithm

4 Room for improvement

Our solution seems to be working fairly well (see more examples in section 6). However, not all is perfect, and our algorithm still makes mistakes. Errors are mostly one of three types. You can see some examples in picture 8



Picture 8: Examples of errors our solution makes

1. In cases when content of the image is too crammed, accidental connections can happen. If these closed connections are of just the right size and aspect ratio, a false reading will occur. We did a lot of improvements to prevent these occurrences (mentioned in section 2), however, situations like these can theoretically still happen.

2. If a part of the checkmark is horizontal or vertical, stretches from one side of the box to another and is the right thickness, it will be picked up by the convolution kernel, effectively splitting the checkbox in half. This will cause the checkbox to be picked up as two separate components and either not be registered at all due to the components being too small, or being registered as multiple checkboxes.

3. If a checkmark is intersecting one of the checkbox corners and is thick enough, it will ruin the contour of the checkbox and the detector will ignore it.

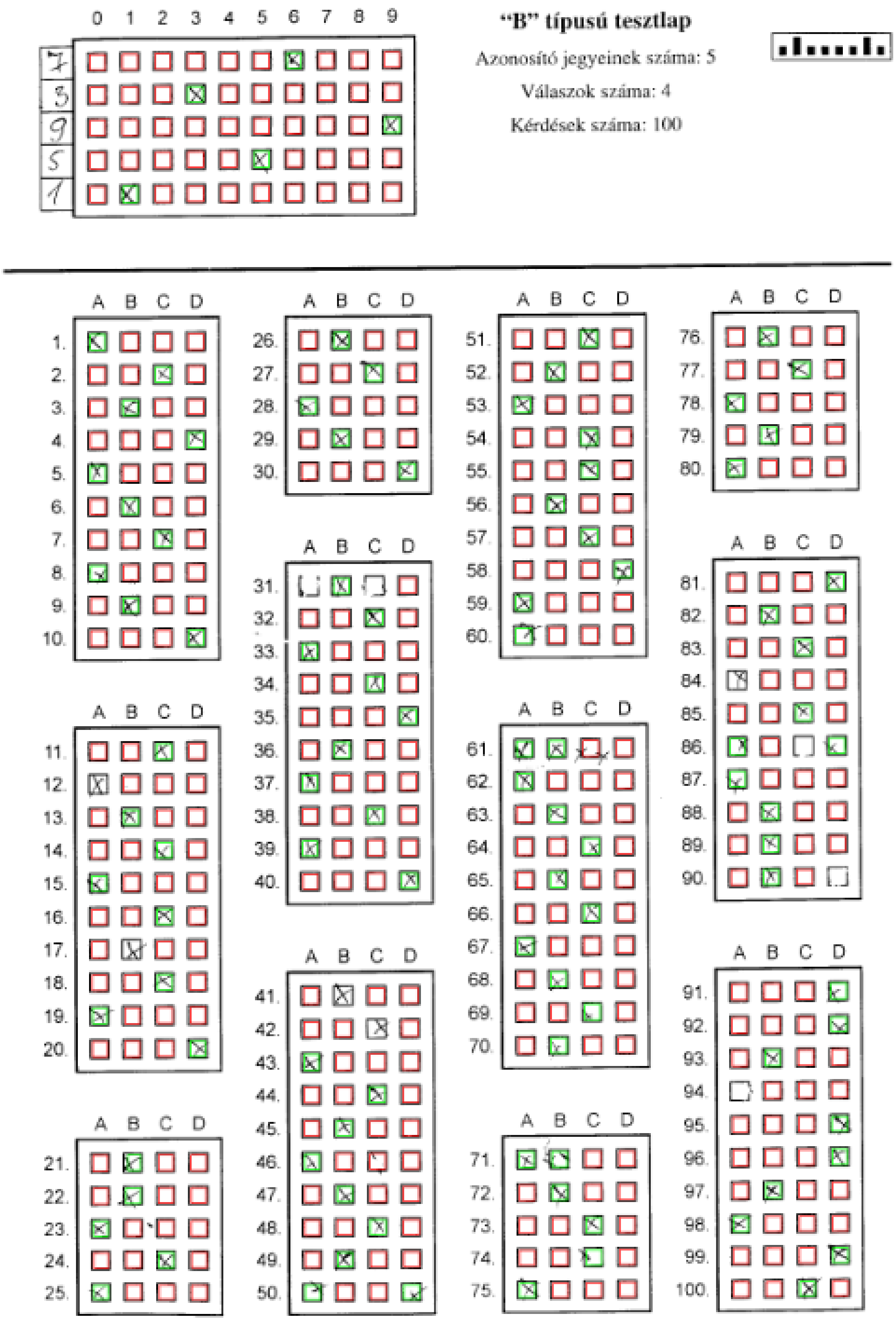
4. If the checkbox has a gap too large, it will not be picked up by the detector.

A method to fix most of these problems was proposed by one of the lecturers, Dr. István Marosi. He suggested to take note of the most common shape that is considered a checkbox, and then go through the image map one more time in an attempt to template-match 'bad' checkboxes.

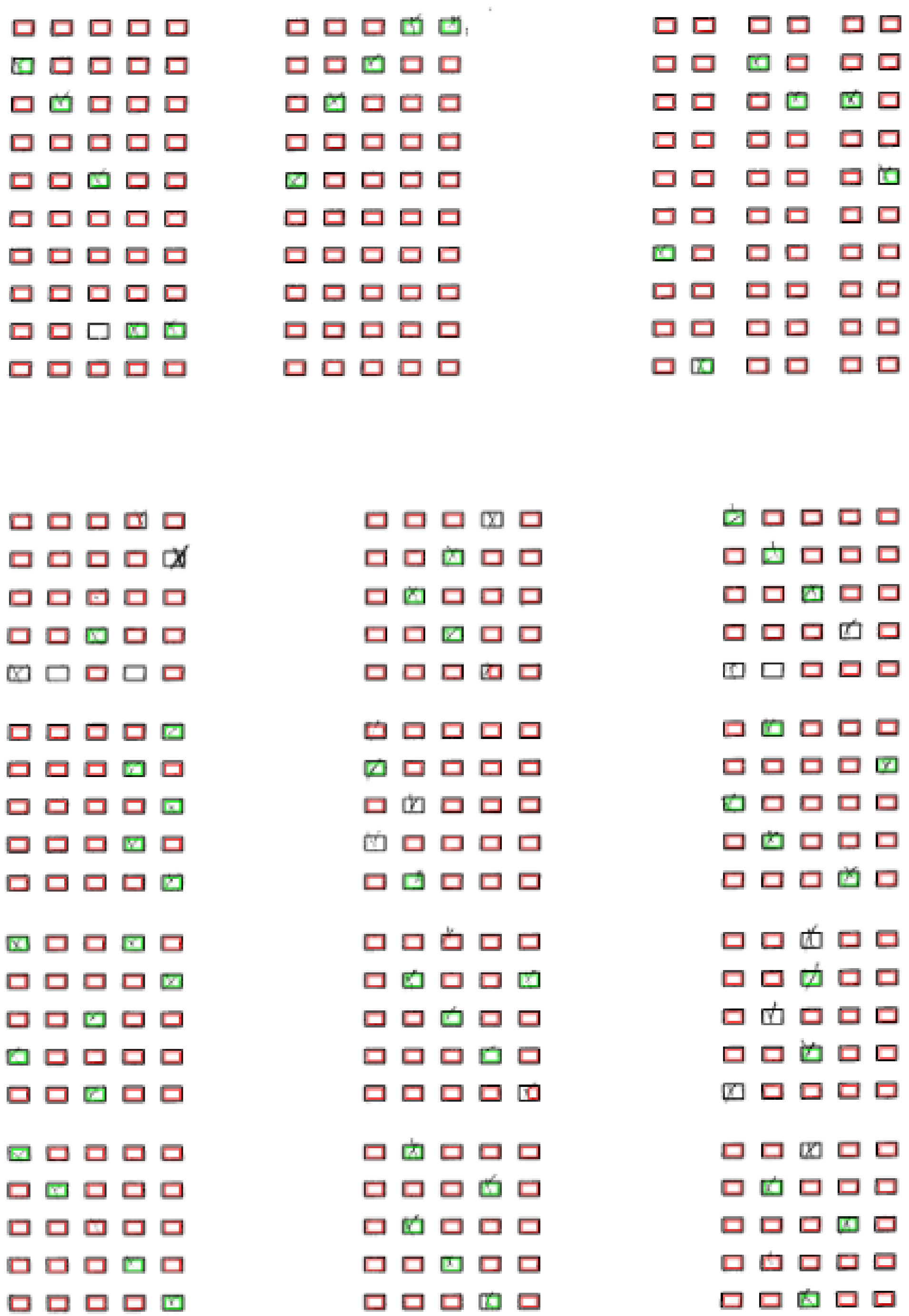
5 Statistics

6 More examples

In this section you can take a look at a few more examples and see the results our program produces.



Picture 9: First example



Picture 10: Second example

BME .... **Kar**

Oktatók Hallgatói Véleményezése

Az oktató neve: ....

A tanszék: ....

A tárgy neve: ....

A tárgy kódja: ....

előadó: gyakorlatvezető:

Kitöltés előtt olvasd el a lap alján található útmutatót!

		1	2	3	4	5
1.	Az előadások/gyakorlatok kb. hány %-án voltál jelen? 1. Kevesebb, mint 20%, 2. 20-40%, 3. 40-60%, 4. 60-80%, 5. 80-100%					
2.	Előző félévi tanulmányi ösztöndíj-átlagod? 1. 2.5 alatt, 2. 2.5 és 2.99, 3. 3.0 és 3.49, 4. 3.5 és 3.99, 5. 4.0 fölött					
3.	Mennyire fedi le az írásos segédanyag vagy jegyzet a tanított témaköröket? 1. 0-20%, 2. 20-40%, 3. 40-60%, 4. 60-80%, 5. 80-100%					
4.	A gyakorlatok segítik-e az előadáson hallottak gyakorlását? * 1. Csak zavart keltenek, 2. Semmi közük az előadásokhoz, 3. Alig segítik, 4. Többé-kevésbé segítik, 5. Nagy mértékben segítik					
5.	Az oktatónak van-e biztos tárgyi tudása? 1. Gyakran téved, 2. Időnként téved, 3. Időnként téved, de korrigálja, 4. Legfeljebb apró elbizonytalanodásai vannak, 5. Teljesen biztos a dolgában					
6.	Az oktató logikusan építi-e fel a tananyagot? 1. Áttekinthetetlennek látszott, 2. Csak néha látszott összefüggés, 3. Többé-kevésbé logikus volt, 4. Összefüggő, általában rendszerezett volt, 5. Világos struktúrájú, könnyen áttekinthető volt					
7.	Fordulhatok-e az oktatóhoz kérdésekkel az órán kívül is? Segítőkész-e az oktató? 1. Teljesen elzárkózik, 2. Általában elzárkózik, 3. Korrekt, de nem örül a kérdéseknek, 4. Általában szívesen fogad, 5. Mindig szívesen fogad és magyaráz					
8.	Segítik-e a táblára (írásvetítőre, stb.) felírtak az anyag megtanulását? 1. Nem használ táblát, írásvetítőt, stb, vagy ez nem hasznos, 2. Kicsit segít, 3. Korrekt, de nem különösebben jó, 4. Logikus és áttekinthető, 5. Szinte művészien tökéletes					
9.	Mennyire lehet az oktató óráin jegyzetelni? 1. Gyakorlatilag nem jegyzetelhető, 2. Rosszul jegyzetelhető, 3. Nagyrészt jegyzetelhető, 4. Jegyzetelhető, 5. Könnyű a lényeget leírni					
10.	Korrekt-e a számonkérések értékelése? * 1. Teljesen igazságtalan, 2. Sokszor igazságtalan, 3. Hol igazságos, hol nem, 4. Sokszor igazságos, 5. Teljesen korrekt					
11.	Az előadás/gyakorlat tanít-e mérnöki gondolkodásra? * 1. Nem tanít erre 2. Csak kis mértékben tanít erre, 3. Többé-kevésbé erre tanít, 4. Nagy mértékben erre tanít, 5. Teljes mértékben erre tanít					
12.	Mennyire mutat rá az oktató a tárgyon belüli, ill. a saját tantárgya és a többi tárgy közötti összefüggésekre? 1. Nem foglalkozik velük, 2. Ritkán mutat rá, 3. Néha rámutat, néha nem, 4. Gyakran rámutat, 5. Sok érdekes összefüggésre rámutat					
13.	Mennyire érzed nehéznek a tananyagot a többi tárgyhoz képest? 1. Nagyon könnyű 2. Könnyű, 3. Közepes, 4. Nehéz, 5. Nagyon nehéz					
14.	A tempó általában megfelelő-e? 1. Nagyon lassú, 2. Lassú, 3. Megfelelő, 4. Gyors, 5. Nagyon gyors					
15.	Mennyire tartod szigorúnak a számonkérést és osztályozást a félév folyamán? * 1. Nagyon enyhe, 2. Enyhe, 3. Közepes, 4. Szigorú, 5. Nagyon szigorú					
16.	Ajánlanád-e a tárgyat másoknak? 1. Feltétlenül lebeszélném, 2. Inkább lebeszélném róla, 3. Esetleg ajánlanám, 4. Nyugodtan ajánlanám, 5. Mindenkit rábeszélnék					
17.	Ajánlanád-e az oktató óráit másoknak? 1. Feltétlenül lebeszélném, 2. Inkább lebeszélném róla, 3. Esetleg ajánlanám, 4. Nyugodtan ajánlanám, 5. Mindenkit rábeszélnék					
18.	*Az oktató által feltett kérdés:*					

* Ha nem volt gyakorlat a tárgyból, akkor a 4. mezőbe, ha nem volt számonkérés, akkor a 10., 15. mezőbe, illetve nem mérnöki tárgynál a 11. mezőbe tégy X-et:

4. 10. 11. 15.

Kitöltési útmutató:

1. Az adatlapot fekete vagy sötét színű (de a papíron nem átütő) tollal, esetleg puha fekete ceruzával töltsd ki!
2. A négyzetekbe tett X-ek ne érjenek ki a több négyzetet összefogó keretig.

Olvasd el a túloldalt is!

Picture 11: Third example

Literature

[1] Loy, Gareth & Barnes, Nick. (2004). *Fast shape-based road sign detection for a driver assistance system*. 1. 70 - 75 vol.1. 10.1109/IROS.2004.1389331.