A Vote Tallying System Based on Computer Vision

Paulo Victor Fernandes Sousa[1], **Julio Cesar da Silva Rodrigues**[1], **Charles Figueredo de Barros**[2]

[1]Curso de Graduação em Ciência da Computação
Universidade Federal de São João del-Rei
São João del-Rei – MG – Brazil

[2]Departamento de Ciência da Computação
Universidade Federal de São João del-Rei
São João del-Rei – MG – Brazil

p.victorsousa2@aluno.ufsj.edu.br, julio.csr.271@aluno.ufsj.edu.br,

charlesbarros@ufsj.edu.br

Abstract. *In this paper, we describe an ongoing project on a vote tallying system based on computer vision. The core idea consists of recording the votes into paper ballots containing markings that can be read by a software using computer vision. The software reads the markings from a video recorded during the elections. The video can be made publicly available, so that any interested party can check that the tallying is correct.*

1. Introduction

Electronic voting has been in use around the world for about three decades. The use of computers during elections has brought many benefits, together with many challenges regarding security and reliability. While it is true that electronic voting systems are capable of mitigating some of the security problems underlying paper-based systems, they also bring other issues due to the conflicting nature of the security requirements of an election.

One of the main threats on electronic voting system is the fact that any error or malicious change in the software may affect the election result. This is an intrinsic feature of the so-called software dependent systems [Rivest and Wack 2008], based solely on the use of electronic voting machines (EVMs). Without a redundant medium for storing the votes, these systems are susceptible to malicious attacks or even accidental changes in the election result caused by software malfunctioning [Prasad et al. 2010, Calandrino et al. 2007, Aranha et al. 2019].

Due to the fact that software bugs may be hard to spot (even harder if we consider complex systems consisting of thousands of lines of code), it becomes infeasible to guarantee the integrity of an election solely by putting trust in the integrity of the system software. This is the main reason for the development of voting systems that either record the votes into redundant media (typically paper ballots, used for post-election audit and possibly recount) or offer some verifiability feature [Ryan et al. 2009, Chaum et al. 2009, Rivest 2006, Adida 2008] (some kind of mathematical proof that ensures the correctness of the election result and can be independently verified by any interested party).

The first type of voting system comprises those capable of generating physical evidence of each cast vote, in the form of a VVPAT (Voter Verifiable Paper Audit Trail), a redundant copy of the electronic record that can be checked by the voter before vote confirmation. The reason to use a paper trail is fairly simple: it can be directly verified that the vote was cast as intended, simply by reading what was printed in the paper. The same is not true for a digital record, where the vote is recorded in a human-unreadable format.

When the vote is stored solely on electronic medium, we must trust that the software responsible for encoding that vote and storing it is correctly doing its job. However, the requirement for putting blind faith on the software integrity violates the principle of transparency.

2. Tallying Votes With Computer Vision

Computer vision was created with the goal of analyzing how a machine sees the real world [HONORATO and MILANO 2010]. It makes possible to obtain information and patterns from images and videos, being widely used for face recognition, product quality control, detection of geographic areas, etc..

Computer vision can be especially suited for tasks that would be too costly for humans, like counting a huge amount of votes in an election. It is well known that counting paper ballots is a task prone to human error. With the aid of computer vision, it is possible to guarantee the correctness of the process.

In this paper, we propose a vote tallying system based on computer vision. The core idea is that voters cast their votes, using an electronic voting machine capable of printing a ballot containing markings that encode the vote and can be read by a software responsible for counting these votes.

2.1. Details of the Proposal

On the proposed tallying system, we assume the existence of an electronic voting machine equipped with an interface through which voters can choose their candidates (for instance, a physical keyboard or a touchscreen). However, instead of storing the votes in digital format (like a DRE machine), the EVM prints a paper ballot containing markings that encode the voter's choices. Note that the markings are not supposed to be hand-filled by the voter. Instead, they are filled by the voting software, so that the printed ballot is ready to be cast into some sort of appropriate ballot box.

Each paper ballot is prepared according to the layout illustrated on Figure 1. Regardless of the election format in action, every ballot presents various black rectangles that operates as delimiters. They are placed strategically to enable operations responsible for the recognition of the ballot's components, such as the voting field, header and the footer.

At the header, we can note the presence of three distinct components, and each of them has their relevancy on the proposed layout. The barcode represents a unique serial number given to each ballot on the printing process. The QR code corresponds to a digital signature applied on the ballot's contents, using an authentic public key provided by the electoral authority. The logo slot can accommodate any kind of information about the election (e.g., electoral body logo).

Generally, each voting field for each position in dispute will have the role's name at the top, followed by an empty grid destined to the voting choice by inserting the markers (black rectangles). As already mentioned, the main idea is that this process would be made via software, instead of allowing the voter to hand-fill the markings, but the first version of the implementation does not have a module responsible for that marking yet. The various markers surrounding the voting field only have the purpose of offering assistance in the recognition process, providing references for identifying each digit, and the slots located at the right side of the ballot (aligned with the QR code) are destined to portraying pictures of the candidates.

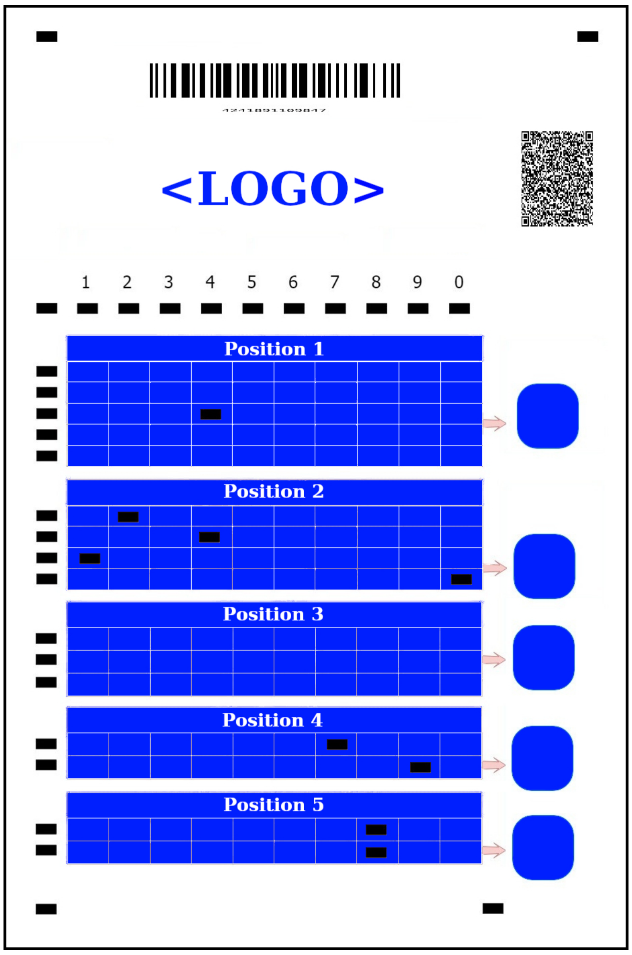

Figure 1. Ballot layout including markers, QR Code and Barcode.

The number of positions in dispute varies from election to election. In our experiments, we tested ballots suited for five positions, simulating a Brazilian presidential election scenario.

2.2. Implementation

Our proposal was based upon a previously existing implementation, developed by a team of undergraduate students from the Computer Science course at the Federal University of São João del-Rei.

The system was implemented using Python language and it is publicly available at `https://github.com/Stanryu/VJPC-VoteCounter/tree/`

`gerarBoletas`. We also used several Python and cross-platform libraries, such as a main base set, composed by PyCryptodome, OpenCV, Barcode, qrcode, numpy, amongst others. Finally, it is important to mention that a cross-platform image editor (GIMP) was used to perform modifications on the ballot layouts that were necessary during the developing stages of this system.

Briefly describing the original project, the system was already capable of executing the vote reading process for a unique ballot layout, in addition to performing the tally processes given a set of ballots or a video containing the images of the ballots, divided into one frame per ballot included in the media. Finally, the last relevant system functionality consisted of printing the results, since the candidates and the specific ballot layout were properly specified before the execution of the election operations. Our goals then were to generalize the application possibilities, expand the use cases, allow the use of any ballot layouts, pre-configure the election, besides adding some cryptography resources in order to guarantee the system security and integrity.

3. Main Changes to the Original Implementation

Given that we already had a pretty solid implemented base, capable of analyzing and computing votes by the means of computer vision resources, we started to formalize and build new modules, which were associated directly with the existing project. This brand new modules were built to amplify the use cases of the proposed system.

The project was preceded by a theoretical study on the subject, so that we could get sufficiently familiarized with the utilization of the computer vision library, and to select carefully the first developing steps according to the most critical problems that the initial project had at the start. Amongst that problems, we had to deal with some malfunctions on the contours identifications, in addition to some frail logic that were spoiling the votes printing process, inducing a variety of errors that were most noticed with ballots filled ”incorrectly” (e.g., identifying a null or void field of votes or more than a marker per line).

3.1. Election Configuration Module

The first new module was developed to enable the creation of distinct election configurations, which at first, would allow the system utilization on a wide variety of contexts and environments (e.g., Collegiate members election). In details, this module allows the election administrators to determine the number of election runs, their designations, order, and finally the quantity of digits assigned to each run. We put this information set in a *.TXT* extension file that will be utilized later to configure several aspects of the determined election, including a ballot layout generator, which will be presented and discussed later in this article.

In order to adapt the vote reading functionality to the inclusion of a configuration module, which allows to define multiple settings of election, the reading function was modified. In the original design, it was set to read exactly five runs for a given election, each run with a predefined number of digits identifying each candidate. In the modified version of the reading function, it reads the number of runs from the configuration file and chooses the correct parameters in order to be able to read the ballots.

3.2. New Ballot Layouts

The next developing step was not focused on the implementation, but on the ballot layout. We introduced a threshold type (black rectangles) that delimiters the portion of the ballots that corresponds to their header (where the logo is situated). Our goal with that modification was to enable the detach of the voting fields to prevent any inconsistencies that could be caused by the header portion presence on the interpretation process of the votes (e.g., misleading identification of the header contours, processed as vote contours), possibly leading to improper vote tally. The markers positioning are shown on Figure 2, before and after they've been added.

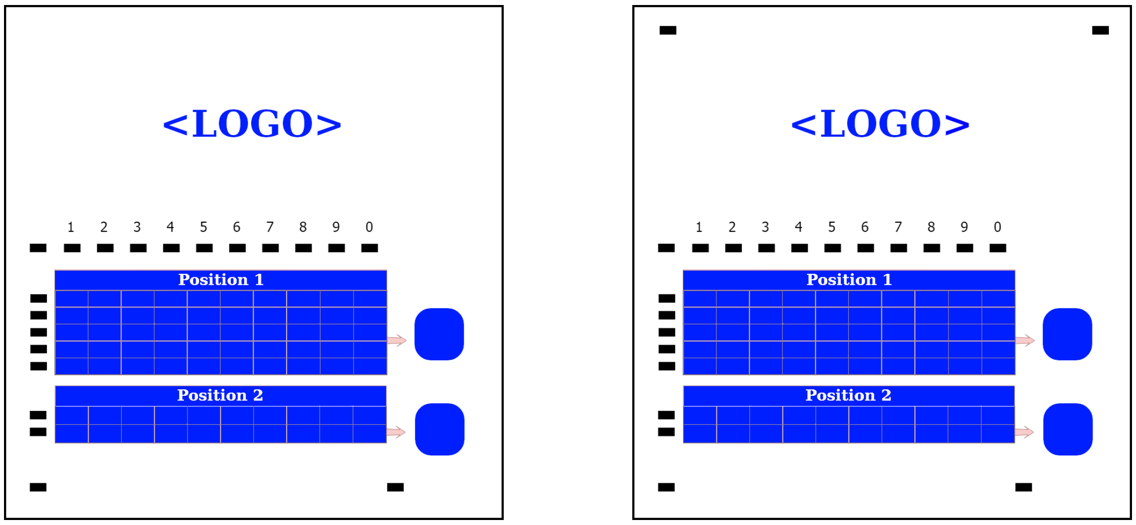

Figure 2. Before and after header and foot markers insertion

Another important change was the application of a clipping process. The voting area ranges from the black rectangles below the header where the numbers are located to the rectangles at the bottom, as shown in Figure 3. Hence, everything that is not relevant for the counting process in the ballot is discarded, leaving only the voting area. After having the area cut out, the counting process begins. From this, using the openCV Python library, it is possible to identify all the black rectangles. With openCV it is detected where there are filled rectangles and from that they are stored in an array. After that, the matrix is transformed into a matrix within python itself, and from an identification logic it is possible to identify who the user voted for.

The initial version of the software did the counting without making this cut of the matrix area. With the changes made in this project, with the user being able to edit the ticket by changing the header, adding QR Code, barcode and being able to choose how many voting fields it would have, this ended up causing problems when reading it with openCV, since it could identify more rectangles in places other than the voting area depending on the ballot disposal. Applying this clipping process, the problem was solved.

The ballot layout had to be altered by means of repositioning of the header, so that it could contain the QR code and a barcode. The QR code encodes a digital signature of the ballot, while the barcode contains the unique serial code that identifies the ballot.

Lastly, we also introduced a mechanism capable of generating the ballots layout,

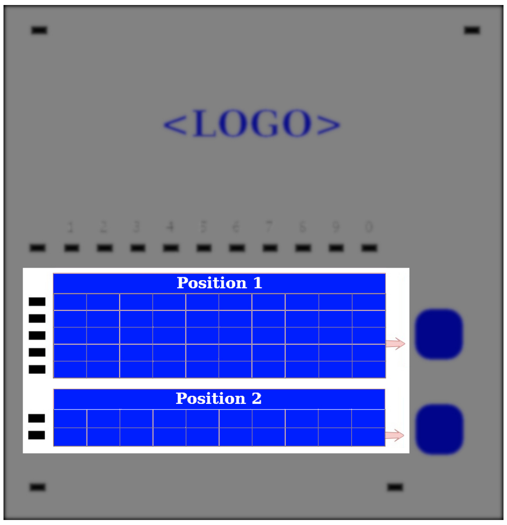

Figure 3. Highlighted clipping area

a ballot generator. The basic behavior of this module is to produce a blank white image, whose height dimension can vary according to the number of runs given by the election configuration file mentioned earlier.

Thenceforth, we begin to position the elements that compose the ballot. In other words, we set the basic components already produced, the header, voting field, footer, and customize them, adding the logo, and the position names in the ballot. As a main limitation on this generator, we can mention its inability to produce a ballot whose number of digits for each run could vary. In other words, the implemented module can only generate ballots whose number of digits for each run is immutable (we choose five digits for each position by default).

3.3. Adding a Digital Signature Mechanism

One of the defense systems against fraud attempts is the use of digital signatures to ensure that the ballot will not be exchanged or modifies in the middle of the process. A digital signature is a technology that serves to authenticate documents, from encrypted keys, to guarantee the integrity, authentication and non-repudiation of data, preventing malicious attackers from inserting fake ballots.

For testing purposes, the authentication of the ballots was done with the El Gamal [El Gamal 1985] digital signature scheme. Other schemes, such as ECDSA, should be used for real applications. Prior to the election, the election authority builds a key pair and distributes the public key.

The serial number encoded into the barcode is used to generate a digital signature, which is encoded into a QR code and added to the top margin of the ballot. Prior to counting the votes encoded in the ballot, the software reads the QR code and proceeds to signature verification. If verification is successful, the votes are counted. Otherwise, the ballot is rejected and any votes encoded into it will not be included in the final tally.

4. Future Works

There were some functionalities this system could benefit from by applying a new phase of development to produce a future version of this implementation.

Amongst them, we could mention enabling of the vote tallying process by a live video recording device. The basic idea of that functionality would be to create some sort of auditable mechanism during the election process. With that feature in hands, we could perform the counting in parallel with *DRE* machines for example, in addition to provide a re-tallying process that would assess the result correctness and possibly eliminate a few fraud schemes. It is important to reassure that this whole process would be assisted by the cryptographic mechanisms described earlier, in order to guarantee the ballots authenticity and integrity.

The second main improvement still to be made, is the flexibilization of the ballot generator. As mentioned earlier in this article, the generator only produces ballots with an immutable number of digits for each position. The upgraded version of that module would be capable of creating a ballot which roles could have an unrestricted number of digits, expanding the system's utilization contexts.

The third approach to be made on the current state of the system would be to provide a voting module, which would be responsible for receiving the voting choices by the voters, and marking it properly on the electronic ballot.

Finally, one of the most important aspects that needs attention on a future step of development of this implementation is the proper insertion of a digital signature at the ballot. At this stage, only the barcode (serial number) is signed, but ideally, we must sign not just the barcode, but also the ballot's content, which includes the voter's choices.

5. Final Remarks and Conclusions

We presented a vote tallying system based on computer vision that could increase the reliability of the voting process by encoding the votes into paper ballots and allowing software-assisted tallying. Encoding the votes into paper ballots brings more transparency, because voters are able to verify that their votes were correctly cast. Using a software to tally the votes, by means of computer vision techniques, eliminates human errors typically associated with the manual process of counting paper ballots.

Applying some initial ordinary tests with personal and portable computing devices (e.g., laptops and desktops), we measured the average time observed on the ballot detection process. It is pretty clear that the ballot layout can impact on this metrics. In other words, the greater the number of roles and digits featured in the ballots, the longer the vote tallying process will take. Simulating a Brazilian election scenario, the average time spent on the reading process of the votes for each ballot was around 0.03 seconds.

Preliminary results suggest that this system could offer good levels of performance on the tallying process for different scales of elections. Considering a reasonable average country population worldwide as forty million people, this tallying process could take from two to three weeks to be completed. That demanded time obviously would be reduced drastically increasing the computational power of the machine and distributing the data to computing conglomerates. That could be an interesting approach in contrast with the USA electoral tally system, providing results at a much shorter period of time utilizing computer vision.

Although it is pretty clear this is a project whose development is still in progress, we can notice that it is a possibility to benefit from the usage of a system like that. We intend to deepen our studies on developing the features this system could provide for the sake of democratic elections. The possibility of a implantation of a similar system could bring several benefits to whoever utilizes voting systems that only perform vote storage in one mean (e.g., Brazilian DREs), eliminating the blind trust of the voters on result correctness required by the electoral body in some cases.

Acknowledgements

We would like to thank Vinicius Marques do Nascimento, from Accenture Applied Intelligence, for providing us with valuable insights on the original implementation and guiding us throughout the process of improving the initial project of which he is the main author. We also thank the anonymous referees for their helpful comments and suggestions.

References

Adida, B. (2008). Helios: Web-based open-audit voting.

Aranha, D. F., Barbosa, P. Y., Cardoso, T. N., Araújo, C. L., and Matias, P. (2019). The return of software vulnerabilities in the brazilian voting machine. *Comput. Secur.*, 86(C):335–349.

Calandrino, J. A., Feldman, A. J., Halderman, J. A., Wagner, D., Yu, H., and Zeller, W. P. (2007). Source code review of the Diebold voting system.

Chaum, D., Carback, R. T., Clark, J., Essex, E., Popoveniuc, S., Rivest, R. L., Ryan, P. Y. A., Shen, E., Sherman, A. T., and Vora, P. L. (2009). Scantegrity ii: end-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE Transactions on Information Forensics and Security*, page 13.

El Gamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 10 – 18, Berlin, Heidelberg. Springer-Verlag.

HONORATO, L. B. and MILANO, D. d. (2010). Visão computacional. Universidade Estadual de Campinas (UNICAMP).

Prasad, H. K., Alex, J., Gonggrijp, H. R., Wolchok, S., Wustrow, E., Kankipati, A., Krishna, S., and Yagati, S. V. (2010). Security analysis of india's electronic voting machines.

Rivest, R. (2006). The threeballot voting system.

Rivest, R. L. and Wack, J. P. (2008). On the notion of software independence in voting systems. *PHIL. TRANS. R. SOC. A*, pages 3759–3767.

Ryan, P. Y. A., Bismark, D., Heather, J., Schneider, S., and Xia, Z. (2009). Prêt à voter: A voter-verifiable voting system. *Trans. Info. For. Sec.*, 4(4):662 – 673.