

# Automatic Robotic Arm Calibration for the Integrity Test of Voting Machines in the Brazilian 2022's Election Context

Marcondes R. da Silva Júnior<sup>1</sup><sup>a</sup>, Jonas Ferreira Silva<sup>1</sup><sup>b</sup> and João Marcelo Teixeira<sup>2</sup><sup>c</sup>

<sup>1</sup>*Informatics Center, Universidade Federal de Pernambuco, Recife, Brazil*

<sup>2</sup>*Electronics and Systems Department, Universidade Federal de Pernambuco, Recife, Brazil*

**Keywords:** Robot Manipulation, Computer Vision, RGB-D Camera, Brazilian 2022's Election, Integrity Testing.

**Abstract:** The Brazilian electoral system uses the electronic ballot box to increase the security of the vote and the speed of counting the votes. It is subjected to several security tests, and the one that has the most human interaction and personnel involved is the integrity test. Our macro project proposed a solution to optimize the testing process and reduce the amount of human beings involved, using a robotic arm with the aid of computer vision to optimize the personal demand from 8 people to 2. However, in order to use the robot, technical knowledge was still required, and it could not be used by any user, as it was necessary to manually map the keys to the places where the robotic arm would press to perform the test. We present a solution for automatically mapping a workspace to a robotic arm. Using an RGB-D camera and computer vision techniques with deep learning, we can move the robotic arm with 6 Degrees of Freedom (DoF) through Cartesian actions within a workspace. For this, we use a YOLO network, mapping of a robot workspace, and a correlation of 3D points from the camera to the robot workspace coordinates. Based on the tests carried out, the results show that we were able to map the points of interest with high precision and trace a path plan for the robot to reach them. The solution was then applied in a real test scenario during the first round of Brazilian elections of 2022, and the obtained results were compatible to the conventional non-assisted approach.

## 1 INTRODUCTION

The Brazilian electoral system began its computerization process in 1995, when the TSE started the Electronic Ballot Box project. The old method of voting using paper ballots, which was slow and susceptible to fraud, gave way to electronic equipment that brought a greater degree of transparency and security to the process of voting and counting the votes. In 1996, in municipal elections, 57 cities had their first contact with the electronic voting machine, counting the votes of more than 32 million Brazilians. After that, in the 2000 elections, Brazil held its first completely computerized election (Court, 2021).

Currently, in order to serve the 147,918,438 people eligible to vote, the holding of elections has a high demand for resources, including logistics, personnel, costs and so on. It covers 5,568 municipalities around the country and 483,665 polling stations. Therefore,

the electronic voting system facilitates the electoral process by providing more efficient mechanisms for transporting and counting votes (Ferrão et al., 2019) (Pinto et al., 2004).

The electronic voting machine is an embedded system used in elections, consisting of two terminals: the polling station terminal and the voter terminal. The first consists of a numeric keypad, an LCD display and a biometric reader, where the voter is identified and the release of the vote at the voter's terminal is performed. The second is composed of a numeric keypad and an LED screen, on which the votes are recorded.

Despite its advantages, the technology of electronic voting machines is a recurrent target of criticism and attacks. Many people do not trust the electronic process and suggest the evolution of the system or the return of the traditional model, based exclusively on paper (VOGEL, 2006).

There are also arguments about the lack of printing of the vote and the difficulty of political parties and civil organizations to monitor and audit the stages of voting, counting and totaling. In this sense, the

<sup>a</sup> <https://orcid.org/0000-0003-0359-6113>

<sup>b</sup> <https://orcid.org/0000-0002-2277-132X>

<sup>c</sup> <https://orcid.org/0000-0001-7180-512X>

Brazilian electoral system has constant challenges related to the transparency, reliability and security of the process as a whole. Therefore, there are several ways to audit and test the polls.

As mentioned before, tests and inspections are carried out at least every 3 months in the Brazilian electronic voting machines, in order to ensure their integrity. One of the main tests carried out in this regard is the Electronic Ballot Box Integrity Test, which is a parallel vote to the official one, which aims to demonstrate the proper functioning of the equipment and the security of the Electronic Ballot Box. This process starts the day before the elections; in which 641 electronic voting machines, randomly selected from approximately 500,000 voting machines, are removed from their polling stations and replaced by new ones. The selected voting machines are then taken to the place where the Integrity Test will take place. In order to carry out this voting equivalent to the official vote, the process begins on the eve of the elections, when a public ceremony takes place for the drawing of the voting machines that will participate in the tests. Then, the randomly selected electronic voting machines are transported from their original sections to the place where the integrity test is carried out, and the entire audit process is monitored by cameras. In parallel with the draw, to simulate voters, voting ballots are filled out by students from the public education network, representatives of political parties and coalitions, which are deposited in sealed canvas urns.

On election day, the integrity test starts at the same time as the official voting and the first procedure is to print the zero, which is an extract that proves the absence of votes counted in the electronic voting machine. Then, all votes from the ballots that were filled in earlier are entered, one by one, in a parallel system. At the end of the day, a Ballot Box Report (BU) is printed and the auxiliary system also issues a bulletin. This way, both countings are compared in order to verify if the electronic voting machine worked normally, verifying that the votes received are in accordance with the votes counted.

Currently, the Integrity Test process has the participation of 6 (six) to 8 (eight) people per Electronic Ballot Box to be audited. To increase productivity and reduce the chance of human error involved in the testing process, a solution for test automation was devised: this alternative solution uses a robotic arm guided by computer vision and 1 (one) human being.

However, even with the automation of the process, it was still necessary to map the robot's position manually, so the main focus of this work is to present a process for the automatic mapping of the keys of the electronic voting machine as a whole. We use a robot

with Cartesian motion support, enabling the mapping of robot coordinates to the RGB-D camera coupled on it using computer vision techniques and deep learning, so that the machine understands what it is seeing.

The results obtained by this work were considered satisfactory, where our macro project, which is the "automation of the audit of the functioning of electronic voting machines in Brazilian elections" won a technological innovation award in the Brazilian judiciary and was presented to the minister of the federal supreme court (STF), who at the time was the president of the Superior Electoral Court (TSE). In addition, as a technical result, it was possible to detect, track and map all the keys on the Brazilian electronic voting machine.

The present work is structured as follows. Section 2 presents the background, which shows what was used to create our solution. Section 3 describes the methodology used to create our solution. Section 4 describes the tests and results, while section 5 concludes the work and points future work directions.

## 2 RELATED WORKS

Garcia et al. (Garcia et al., 2013) used a mapping approach very similar to the one of this work. They also used four target points to guide the robotic arm in tasks of interacting or manipulating objects within the created workspace. The depth information in the image improves the accuracy of the trajectory calculation.

Singh, Rakesh et al. (Singh and Kotecha, 2021) do a work focused on the optimization of the SLAM algorithm, the authors propose to use the deep learning technique to improve the SLAM algorithm so that it is able to differentiate between static and dynamic objects.

Viturino et al. (Viturino et al., 2021) used an instance segmentation method (Mask R-CNN), a point cloud collision verification system and a path planning technique with orientation control based on the Adaptive Artificial Potential Field algorithm to detect and grasp objects using a robotic arm with 6DoF.

## 3 BACKGROUND

In this section, we will present the main concepts that were underneath the proposed solution. We start by providing some background related to object detection and specifically YOLO's neural network, and end the section talking about the benefits of using an

RGB-D camera for robotic arm placement and manipulation activities.

### 3.1 Object Detection

Detection of objects in images is a computer vision task that aims to locate instances of objects of a certain class in a scanned image. An object detection system of a specific class can be defined as a function  $f$ , which maps a digital image  $I$  in the set of locations  $B$ , given the model  $W$ . Thus, we have:

$$B = f(I, W)$$

The location of an object can be defined in terms of a bounding box, a rectangle aligned to the  $x$  and  $y$  axes of the image that describes the object. A bounding box is defined by a vector  $b \in \mathbb{R}^4$ , which indicates the upper left and lower right coordinates of the rectangle. Each location detected by the system is associated with a value  $c \in [0, 1]$ , which indicates the reliability of the system that the bounding box correctly describes an object of the class it is desired to detect.

During training of an object detection system, a criterion must be defined to identify whether a location detected by the system corresponds to an object in the processed image. If the set of examples is annotated with the bounding boxes of the objects contained in their images, the detected locations are compared to each of the annotations, according to a similarity metric.

If a detection is compared to reference locations and does not obtain a minimum similarity value, it is considered that the location does not correspond to an object of the detected class, that is, a false detection has occurred. If more than one detection obtains a similarity above the minimum value in relation to the same reference, repeated detections are considered to have occurred. If no detection obtains minimal similarity with respect to a specific reference, it is said that the detection has failed.

#### 3.1.1 YOLO

YOLO (You Only Look Once) is a computer vision technique used to detect objects in real time, according to Redmon et al. (2016) (Joseph Redmon and Farhadi, 2016) the YOLO method reformulates the detection of objects by transforming it into a regression problem, starting only from the pixels of an image, which results in the probabilities by class, in the coordinates and in the dimensions of the predictions that delimit the objects. In addition to being simple, since it uses an end-to-end approach with a single convolutional network, it also presents competi-

tive performance as a function of time. Unlike other object detection methods based on convolutional networks, YOLO only needs a propagation of the input through the network to generate predictions, presenting performance compatible with real-time object detection methods, maintaining predictive accuracy.

For the fourth version, the authors (BOCHKOVSKIY et al., 2020) proposed to perform an ideal balance between the input network resolution, the convolutional layer number, the parameter number and the number of outputs of the layer. As a result, the main features that can be highlighted in this version are the improvement in inference speed and accuracy. Another important feature is the fact that it is more efficient to run on GPUs, as it has been optimized to use less memory.

Chien-Yao Wan et al. (Wang et al., 2022) assert that YOLOv7 is the fastest and most accurate real-time object detection model for computer vision tasks.

YOLOv7, compared to its predecessor and its base, YOLOv4, provides much improved real-time object detection accuracy without increasing inference costs. YOLOv7 can effectively reduce about 40% of the parameters and 50% of the state-of-the-art real-time object detection computation and achieve faster inference speed and higher detection accuracy. Overall, YOLOv7 provides a faster and stronger network architecture that provides a more effective method of resource integration, more accurate object detection performance, a more robust loss function, and greater efficiency in labeling and training model.

In the work of Chien-Yao Wang et al. (Wang et al., 2022) YOLOv7's performance was evaluated based on previous versions of YOLO (YOLOv4 and YOLOv5) and YOLOR as baselines. The models were trained with the same settings. The new YOLOv7 shows the best balance between speed and accuracy compared to high-end object detectors.

### 3.2 Camera Calibration

When recording an image in a photograph, what an ordinary still camera basically does is sampling and geometrically mapping the 3D points of the world scene onto a 2D image plane. This mapping of the sampled points is called projection (Hartley and Zisserman, 2004). There are several models of cameras that can be used to project images of 3D scenes, among which there are those that use lenses and those that do not, with the models without lenses being the simplest ones.

In principle, to calculate the image projection of real objects from a 3D world onto a 2D plane, it is

necessary to define the position of the 3D points of the objects present in the scene in relation to a common coordinate system. This generic coordinate system is called the world coordinate system, its origin is at (0, 0, 0) and the location of each object in the scene, including the camera, can be expressed by three coordinates (X<sub>w</sub>,Y<sub>w</sub>,Z<sub>w</sub>), as illustrated in Figure 1.

As the image projection of an object does not depend only on its location in the world, but also on the position of the camera that projects it in relation to the object itself, it is common to establish an alternative coordinate system considering the coordinates of the camera center in relation to the world. This system is known as the camera coordinate system, whose values with respect to each axis are given by (X<sub>c</sub>,Y<sub>c</sub>,Z<sub>c</sub>). This new system corresponds to the world coordinate system after undergoing a certain transformation (rotation and translation) that leaves its axes coincident with the respective axes of the camera coordinate system, as shown in Figure 1.

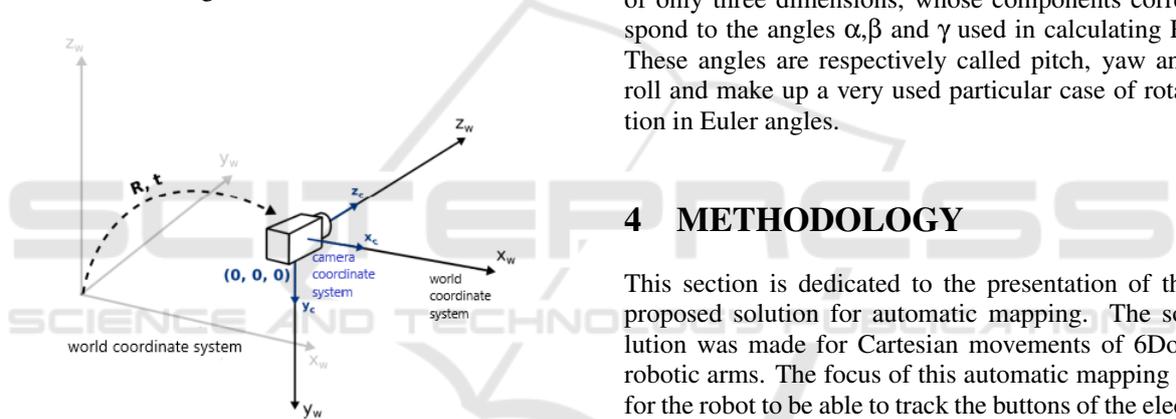


Figure 1: Camera coordinate system obtained from a world coordinate system transformation.

The rotation and translation that align the two coordinate systems are represented respectively by a rotation matrix R3x3 and a translation vector |R|T|3x3. In order to simplify the notation, it is common to use the concatenation of these matrices and represent them as a single matrix |R|T|3x4. The matrix |R|T| is known as the matrix of extrinsic camera parameters (pose matrix or simply pose), because with it it is possible to know the external parameters of the device, such as location and inclination. The camera pose matrix corresponds to the transformation needed to take points from one coordinate system to another. Mathematically, transforming any 3D point that is represented in the world coordinate system to the camera coordinate system using |R|T| means calculating the product of this matrix with the 4 x 1 matrix formed by the homogeneous coordinates of that point:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{21} & r_{31} & t_1 \\ r_{12} & r_{22} & r_{32} & t_2 \\ r_{13} & r_{23} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

In the aforementioned equation, [x<sub>w</sub>, y<sub>w</sub>, z<sub>w</sub>, 1]<sup>T</sup> is the matrix with the homogeneous coordinates of a point in the world and [x<sub>c</sub>, y<sub>c</sub>, z<sub>c</sub>]<sup>T</sup> is the matrix with the coordinates of the transformed point, that is, in the coordinate system from the camera.

The rotation matrix R3x3 is obtained through the matrix product of the rotations on the axes of the world coordinate system during the transformation, that is, if this rotation occurs with angles α,β and γ respectively on the axes X<sub>w</sub>, Y<sub>w</sub> and Z<sub>w</sub> that same order, the matrix R can be evaluated as follows:

$$R = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$

This rotation can still be represented by a vector of only three dimensions, whose components correspond to the angles α,β and γ used in calculating R. These angles are respectively called pitch, yaw and roll and make up a very used particular case of rotation in Euler angles.

## 4 METHODOLOGY

This section is dedicated to the presentation of the proposed solution for automatic mapping. The solution was made for Cartesian movements of 6DoF robotic arms. The focus of this automatic mapping is for the robot to be able to track the buttons of the electronic voting machine in a pre-defined workspace.

To implement the solution, some steps were necessary, namely: creation of the robot workspace, creation of the dataset, use of the YOLOv7 network, spatial transformation of Cartesian points of the camera to Cartesian points of the robot and path planning of robot movements.

### 4.1 Robotic Arm Workspace

The workspace of a robotic arm is the set of all positions it can reach. This depends on several factors, including arm dimensions.

For our work, a “template” was created, shown in Figure 2, where the electronic voting machine remains in a fixed direction in relation to the robot base, this was necessary so that the complexity of the solution was reduced, since with the template, it is possible to know in which axis of the robot the urn is located, thus, it is not necessary to track the urn, but only its buttons.

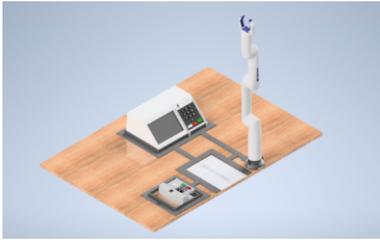


Figure 2: Robot template.

Knowing where the voter terminal and the polling station terminal would be, the robotic arm was moved to a position where it would be "seeing the polling station terminal", this way it was possible to save the cartesian coordinates(x,y,z) and angle position of attack (roll,pitch,yaw) of the robot as "see" position to start his workspace within our work.

From the "see" position, the robotic arm was taken manually, using its joystick, to the center of the points marked in figure 3, thus saving the Cartesian coordinate values (x,y,z) and angle of attack(roll,pitch,yaw) from each point, so that we could create our robotic arm workspace.



Figure 3: Robot workspace.

After that we would have 5 points saved, so we were able to extrapolate a "pivot" position, which would be an intermediate position for pressing the keys, where this position was calculated by the expressions:

$$X = \frac{(X_{min}+X_{max})}{2} \quad Y = \frac{(Y_{min}+Y_{max})}{2}$$

$$Z = (Z_{posSee} + Z_{min}) * 0.40$$

$$Roll = \frac{(\sum allRolls)}{5} \quad Pitch = \frac{(\sum allPitches)}{5}$$

$$Yaw = \frac{(\sum allYaws)}{5}$$

where Xmin and Xmax would be respectively the smallest and largest values of x among the 4 points that were marked in the workspace, Ymin and Ymax would be the smallest and largest values of Y and ZposSee and Zmin would be the Z values of the "see" position of the robot and the smallest z value among the 5 points, with that we have the robot workspace defined.

## 4.2 Dataset and Markup

For the construction of the dataset, images of the electronic voting machine were used, obtained through an Intel Realsense D435 camera, a total of 28,420 images of the Brazilian electronic voting machine were obtained, of which 15,300 were from the voter terminal and 13,120 from the polling station terminal.

This way, two sub-datasets were created, where one contained 984 images of the polling station terminal and the other with 1,147 images of the voter's terminal. Derived from the two sub-datasets, images were separated for training and validation of the network, where 752 were for training and 196 for validation of the polling station terminal and 917 for training and 230 for validation of the voter's terminal.

After splitting the datasets we had an interesting variety of images for training, as shown in Figure 4. It was manually checked if it was possible to generate enough images of the necessary base cases.



Figure 4: Example of dataset images.

With the datasets separated, the images were marked and the labels were generated for the training of YOLOv7. This marking was done using the LabelImg application(Tzutalin, 2015). The buttons that were found in the image were marked.

## 4.3 YOLOv7

For the training of YOLOv7, the Google Colab execution environment was used. The version used for YOLOv7 was YOLOv7x, which contains 71.3M of description parameters, with the size of the images for training and validation being 640 and with a batch-size of 16, in addition to 200 training periods for the polling station and 300 training periods for the voter terminal.

After training, we obtained the confusion matrices, where it was possible to observe that the training and validation were successful, with 0.99 as the lowest accuracy value.

Concluding the training, the models were tested with images that were reserved. 578 images were used for the polling station terminal and a minimum confidence level of 0.7 was used. This way, in only 2 images the number 7 was not classified. As for the voter terminal, 913 images were used for testing using

a minimum confidence value of 0.7. In this case, every button was correctly classified, showing the confidence of the classifier.

#### 4.4 RGB-D Image

At this point, an image is obtained using the Intel Realsense D435 camera. After storage, we followed two paths: the first path being the detection of the points that form the robot's workspace, as shown in figure 5. Having acquired the Cartesian positions(x,y,z), the robot's workspace is built, being: (sx, sy, sz), (where sx is the smallest x value of the points, sy the smallest y value, and sz being 0, since it is the value at the camera origin) and (bx,by ,bz) (where bx is the largest x-value of the points, by the largest y-value, and bz is the largest z-value among the 4 points in the image).



Figure 5: RGB-D workspace.

The second way is to pass the raw image to the YOLOv7 detection, where each button is classified and the value(x,y) of the centroids of each class is returned, along with the class to which it belongs.

Having the return of the centroids of each button, the depth value is calculated, along with the specific library of the realsense camera, and the Z value is saved, along with the x and y values of each centroid of each button.

In its final part, the values of the camera workspace and the values of the centroids of each button are observed, at this point it was necessary to convert the values of x and y of the workspace of the camera and buttons to the millimeter system, thus the values of x and y of them were divided by 1000 and sent to our cartesian transformation module.

#### 4.5 Cartesian Transformation

In this module we have access to the robot and camera workspace that were obtained previously, from the two workspaces, the following mathematical formula is used to calculate the x, y and z points of each specific button:

$$X_w = X_{w_{max}} - \frac{(Y_{cam_{min}} - Y_{cam}) * (X_{w_{max}} - X_{w_{min}})}{(Y_{cam_{min}} - Y_{cam_{max}})}$$

$$Y_w = \frac{(X_{cam} - X_{cam_{max}}) * (Y_{w_{max}} - Y_{w_{min}})}{(X_{cam_{min}} - X_{cam_{max}})} + Y_{w_{min}}$$

$$Z_w = \frac{(Z_{cam} - Z_{cam_{max}}) * (Z_{w_{max}} - Z_{w_{min}})}{(Z_{cam_{min}} - Z_{cam_{max}})} + Z_{cam_{min}}$$

The robot workspace is defined by  $(X_{w_{max}}, Y_{w_{max}}, Z_{w_{max}})$ , which is the largest (x,y,z) of the workspace and  $(X_{w_{min}}, Y_{w_{min}}, Z_{w_{min}})$ , the smallest (x,y,z) of the workspace of the robot. With the camera workspace, we have  $(X_{cam_{max}}, Y_{cam_{max}}, Z_{cam_{max}})$  which is the largest (x,y,z) of the camera workspace and  $(X_{cam_{min}}, Y_{cam_{min}}, Z_{cam_{min}})$  which is the smallest (x,y,z), in addition to the workspaces, we have the values of  $(X_{cam}, Y_{cam}, Z_{cam})$ , which are the values of (x,y,z) of the button that we are currently calculating. Finally, we have the values of  $(X_w, Y_w, Z_w)$ , which are the Cartesian values of (x,y,z) that are provided to the robot as the position of the button. Regarding the values of the angle of attack (pitch, yaw, roll), which are necessary for the Cartesian movement of the robot, the same values of the "pivot" point (shown in the robotic arm workspace section) were used, which is calculated as an extrapolation based on the robot workspace's Cartesian points. This way, we have the points (x,y,z,pitch, yaw, roll) of the robot that are passed to the robot's movement function to generate the path planning.

#### 4.6 Path Planning

Initially, attempts were made to directly use the robot's Cartesian movement functions to move to the "see" position, considering it as "0" position. However, when directly using the robot's own Cartesian movement functions, the movement could not be completed successfully because the robot's internal system could not calculate the path planning to arrive at the Cartesian point that was passed to it, as shown in Figure 14. To get around this problem, the robot was sent from the "0" position to the "Home" position, which is also shown in Figure 6, and then to the "see" position, to start the mapping.



Figure 6: From right to left and from top to bottom, the positions: "0", "Home", movement attempt to "see" from the polling station's terminal, "see" from the polling station's terminal, see from the voter's terminal.

After reaching the "see" position, the image that

will be used for the automatic mapping is captured. The process is started using the YOLOv7 detection, the camera workspace generation, depth inference, and the module for converting Cartesian points from the camera to the robot and with that we have our mapping of pressed keys that is shown in the figure 7, along with the intermediate positions such as the “pivot”.

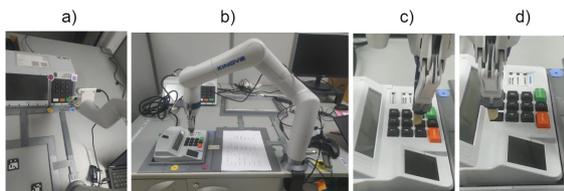


Figure 7: a) “pivot” of the voter terminal, b) “pivot” position of the polling station terminal, c) key “9” pressed, d) key “1” pressed.

## 5 TEST AND RESULTS

For the test for our system, two different light configurations were used, with artificial light and without artificial light, in addition to decoupling the template from the polling station terminal and the voter terminal from the robot’s general template and moving it around the table in 10 different positions. for the polling station’s terminal and 5 for the voter’s terminal, taking into account locations that the robotic arm can reach, as shown in figure 8. This way, a robot workspace was generated for each position that the poll’s terminal template and the voter terminal is positioned, besides, for each position we placed the terminals, it was necessary to manually define a position to “see” this terminal, so that it was possible to map.



Figure 8: Polling station and voter terminal for testing.

In addition to the previous test, another test was also carried out where we kept the light settings from the previous test and removed the Polling sta-

tion terminal from the template and placed it inside a workspace defined on the table itself and moved it 15 times, 5 of which rotated in relation to the robot, as shown in Figure 9. Unlike the previous test, for this one it was only necessary to calculate one robot workspace, one camera workspace and only one “see” position.



Figure 9: Polling station terminal in a workspace on the desk itself.

as a result, with what we propose in our work, we had a 100% success in pressing the keys on the polling station and on the voter terminal, and it should be noted that at no time in the tests carried out with the automatic mapping did the robot pressing two keys at the same time, or missed the key you should press, both in the test using the template, and in the test using a workspace on the table.

With the larger workspace, the desktop, the press accuracy was higher at the center of the key than when the workspace is on the template. In addition, with the workspace on the table, it is only necessary to create a workspace for the robot and a single “see” position, reducing even more the work of mapping the keys and allowing the movement of the desk polling station terminal within the workspace already defined.

also as a result it was possible to use automatic mapping in the macro project, where we ended up auditing two electronic voting machines on the day of the country’s general election and receiving an innovation award in the Brazilian judiciary.

## 6 CONCLUSION AND FUTURE WORK

In this article we propose a method of detecting the buttons of the Brazilian electronic voting machine, using a YOLOv7 network and the automatic mapping of a robotic arm for pressing the buttons of this electronic voting machine.

During the project development process, some problems were encountered, it was noticed by the team that there were no datasets of the Brazilian ballot box available for use in the work and also that the robot's path plan could not calculate some necessary positions.

To solve the problems, we built the necessary datasets of the urn for the work in question and created intermediate positions with angular movements so that the robot could move to the Cartesian positions that were automatically mapped by our solution. The project is available online.

As future work, the team intends to improve the automatic mapping method, making new tests, trying to remove the robot's general template and replacing the robot's path plan with a path plan made by the Theseus framework (Pineda et al., 2022).

## ACKNOWLEDGEMENTS

This work has been supported by the research cooperation project between Softex (with funding from the Ministry of Science, Technology and Innovation—Law 8.248) and CIn-UFPE.

## REFERENCES

- BOCHKOVSKIY, A., WANG, C.-Y., and LIAO, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Court, S. E. (2021). Urna eletrônica 25 anos: lançado em 1996, equipamento é o protagonista da maior eleição informatizada do mundo.
- Ferrão, I. G., Chervinski, J. O., da Silva, S. A., Kreutz, D., Immich, R., Kepler, F., and da Rosa Righi, R. (2019). Urnas eletrônicas no brasil: linha do tempo, evolução e falhas e desafios de segurança. *Revista Brasileira de Computação Aplicada*, 11(2):1–12.
- Garcia, G. J., Gil, P., Llácer, D., and Torres, F. (2013). Guidance of robot arms using depth data from rgb-d camera.
- Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.
- Joseph Redmon, Santosh Divvala, R. G. and Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Pineda, L., Fan, T., Monge, M., Venkataraman, S., Sodhi, P., Chen, R., Ortiz, J., DeTone, D., Wang, A., Anderson, S., Dong, J., Amos, B., and Mukadam, M. (2022). Theseus: A library for differentiable nonlinear optimization.
- Pinto, R. R., Simões, F., and Antunes, P. (2004). Estudo dos requisitos para um sistema de votação eletrônica.
- Singh, R. and Kotecha, R. (2021). Improving mapping of static and dynamic objects in slam using deep learning.
- Tzutalin (2015). Labeling. Free Software: MIT License.
- Viturino, C. C. B., de Oliveira, D. M., Conceição, A. G. S., and Junior, U. (2021). 6d robotic grasping system using convolutional neural networks and adaptive artificial potential fields with orientation control. In *2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE)*, pages 144–149.
- VOGEL, L. H. (2006). A segurança do voto eletrônico e as propostas de fiscalização da apuração pela sociedade.
- Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.