Automating the audit of the Brazilian electronic ballot operation: A new dataset for 6DoF pose estimation of the voter terminal based on domain randomization

Jefferson Medeiros Norberto
jmn@softex.cin.ufpe.br
Universidade Federal de Pernambuco
Recife, Pernambuco, Brazil

Kelvin Batista da Cunha
kbc@cin.ufpe.br
Universidade Federal de Pernambuco
Recife, Pernambuco, Brazil

Marcondes Ricarte da Silva Júnior
mrsj@softex.cin.ufpe.br
Universidade Federal de Pernambuco
Recife, Pernambuco, Brazil

Francisco Paulo Magalhães Simões
francisco.simoes@ufrpe.br
Universidade Federal Rural de Pernambuco
Recife, Pernambuco, Brazil

ABSTRACT

Performing detection and pose estimation of objects in six degrees of freedom (6-DoF) is a widely studied challenge in virtual and augmented reality, robotics and computer vision. For simulation and testing of the Brazilian voter terminal, its pose could allow automatic testing/auditing with robotics arms or virtual reality applications to simulate the voting process. For pose estimation using deep learning, it is necessary to generate large amounts of annotated real data, which is a costly task in time and resources. One way to avoid this issue is to create synthetic data through domain randomization, using 3D object modeling, to train the pose estimation technique with a reduced amount of annotated real data. In this work, domain randomization was utilized to generate a synthetic dataset, starting from a 3D model of the voter terminal, varying the lighting settings, camera position and distract insertion, to verify what impact this randomization has on training a single shot algorithm to perform the detection and pose estimation of this terminal in a different scenario. The new dataset with real and synthetic data from the voter terminal was built and will be publicly available.

CCS CONCEPTS

• **Computing methodologies** → *Virtual reality*; **Object recognition**; **Tracking**; Reconstruction.

KEYWORDS

Object pose estimation, Brazilian voter terminal, Domain randomization, Computer vision, Object tracking

ACM Reference Format:
Jefferson Medeiros Norberto, Kelvin Batista da Cunha, Marcondes Ricarte da Silva Júnior, and Francisco Paulo Magalhães Simões. 2023. Automating the audit of the Brazilian electronic ballot operation: A new dataset for 6DoF pose estimation of the voter terminal based on domain randomization. In *Symposium on Virtual and Augmented Reality (SVR '23), November 06–09, 2023, Rio Grande, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3625008.3625017

1 INTRODUCTION

The 6-DoF object pose estimation is essential in several applications, such as augmented reality, robotic manipulation, autonomous vehicles, and others. For example, it is necessary to provide visual information about the environment to increase the user's perception of the real world in AR applications, control the robot's movements and actions to avoid collisions when manipulating objects, or guide autonomous vehicles on the streets [4][22].

This type of task is divided into sub-tasks: detection and tracking. The first one finds the object of interest in a given scene without any previous information about the object's position and initial orientation. In contrast, given an initial pose, the tracking follows the object in a sequence of frames, using the previous information to predict a new one if the object is moving. Therefore, machine learning is a category of algorithms popularly used to estimate the object's pose. Machine learning enables the algorithms to learn features on the images to extract relevant information from the object, from the scenes' particularities, and to conduct correlations with previously acquired information [4].

To perform this task, it is necessary to generate large amounts of annotated data, which costs a lot of time and effort. A way out of this issue is to use synthetic data, through 3D CAD models of the object, to complement the algorithm's training by passing non-real scenes [23] [21] [4]. Thus, domain randomization is a type of data generation algorithm that helps create artificial datasets by creating different scenarios when generating each image, such as varying lighting settings, object position and distracts, among others.

This work focuses on the problem of automating the audit of the Brazilian electronic ballot operation. This type of audit takes place every two years in Brazil and there has been a recent increase in the number of Brazilian machine votes audited, increasing the cost and number of people necessary to carry out this audit task. With that, we tested a detection algorithm proposed by Cunha et al. [5] aided by a robotic arm to help in the automation process. Therefore, it is essential to use the detector to provide visual information to

SVR '23, November 06–09, 2023, Rio Grande, Brazil

ACM ISBN 979-8-4007-0943-2/23/11…$15.00
https://doi.org/10.1145/3625008.3625017

assist the robot in manipulating the voter terminal keys. Nevertheless, the extracted data can be useful for the development of new AR applications for analyzing the voting process or for training voters. By providing the spatial knowledge gathered from the pose estimator, we can also assist inspectors in checking the results of the automatic audition.

To help with network training, it was necessary to create datasets of the Brazilian voter terminal, both synthetic and real since this type of object is not publicly available. Virtual samples were created from 3D modeling, varying the position and orientation of the camera, including distracting objects to cause occlusion and shadows and changing the environment by adding lights with different sizes, colors, and intensities. Two datasets (train and test) were also generated with real images from this ballot to verify the impact of inserting the synthetic dataset when testing the algorithm in a different case not seen in the training. All datasets, weights, and models are available in a Google Drive folder of the university[1].

Several internal processes of the Brazilian Regional Electoral Courts (TREs) are undergoing a digital transformation, with different types of applications emerging in the most diverse areas, as shown at the Expojud event in Brasília [2]. Making these datasets available can help future applications, thinking of ways to do a virtual poll, for example.

The contributions of this work are:

- Construction and availability of the first Brazilian electronic ballot box dataset containing RGB information, depth, and point in the image related to the robot's base;
- Construction and availability of the first synthetic dataset of the electronic ballot box, as well as the models used to generate this dataset;
- Implementation and adaptation of the pose estimation technique, using domain randomization, for the automation scenario of the electronic ballot box auditing process;
- Quantitative evaluation of the pose estimation technique with the electronic ballot box.

2 AUDIT OF THE BRAZILIAN ELECTORAL SYSTEM

Previously known as parallel voting, in 2018, the "Audit of the Functioning of Brazilian Electronic Ballot under Normal Conditions of Use" aims to demonstrate the operation and security of the Brazilian electronic voting system being carried out by the Electoral Justice. This process takes place in each state, starting one month in advance of the official election date, and is initiated by the Regional Electoral Courts (TREs), which need to appoint an Audit Committee on the Operation of Electronic Voters. This commission is composed of a judge of the law, who is the president of the section, and at least six civil servants from the Electoral Justice, at least one from the Regional Electoral Internal Affairs, one from the Judiciary Secretariat, and one from the Information Technology Secretariat. [18].

On the eve of the elections, the Electoral Justice draws, in a public ceremony, the polls audited according to the number of sections present in that federative unit of the draw. Even on this day, these randomly selected ballots are removed from their original sections and installed in the chosen spaces, where these spaces are equipped with recording cameras. The commission must provide ballots, having the number of such ballots between 75% and 82% of the number of registered voters in the respective polling station. These ballots must be filled out by representatives of political parties and coalitions and kept in sealed canvas boxes so that on election day, these ballots are typed into a computer with a parallel system to, at the end of this election, compare the Ballot box printed, with the auxiliary system bulletin [18].

As of 2022, there was an increase in the amount of necessary sampling of equipment to carry out the aforementioned process. This increase was due to changes proposed by the federal police, thus editing 2 articles of TSE resolution nº 23.673/2021, in order to increase transparency and scope throughout the electoral process. So, with this change, the ballot boxes were selected as follows: federative units with 15,000 sections or less, 23 sections would be drawn, the first 20 ballot boxes would be submitted to the Integrity Test, units between 15,001 and 30,000 sections would be drawn 35 ballot boxes, 27 of which for the Integrity Test, in the other units 43 sections would be randomly selected, 33 ballot boxes from that total to be submitted to the Integrity Test. All other ballot boxes that were drawn and that were not included in the integrity test were automatically submitted to the Electoral Systems Authenticity Test [19].

In the current circumstance, given this increase in samples, an opportunity arose to develop an automated audit process using robotic arms. The main objective of this automation was to reduce the number of people needed at the time of the audit, where the robot both enters the voter's registration number into the terminal and then types the votes into the ballot box, being able to utilize this work for the fine-tuning of the robot's position when pressing the keys on the voter's terminal.

3 BACKGROUND

This section discusses some basic concepts about object tracking and detection in six degrees of freedom (6-DoF).

3.1 Object detection and tracking in six degrees of freedom (6-DoF)

An object's pose, in computer vision, refers to its position and orientation relative to the camera's coordinates. The object's pose can change by estimating the relative movement between the object and the camera, moving the camera or the object. The object pose can be described by its location and orientation to the environment. The location, or translation, is composed of the 3D spatial coordinates (X, Y, and Z axes). The rotation describes the object orientation relative to the camera and can be represented, for example, by the Euler angles (roll, pitch, and yaw) [13].

When you know the camera's intrinsic parameters (e.g., focal length, optical center, distortion parameters) and the object's 3D points with their correspondent 2D projection in the image, it is possible to estimate the rotation vector (r = (r_a, r_b, r_y)) and translation vector (t = (t_x, t_y, t_z)), where they represent the relative motion between the camera and the object [13][4].

[1] https://drive.google.com/drive/folders/1UZ_1-SpYdwBGI9QNYR29ksYJ1M4UzR0d?usp=sharing

[2] Expojud event link: https://www.expojud.com/

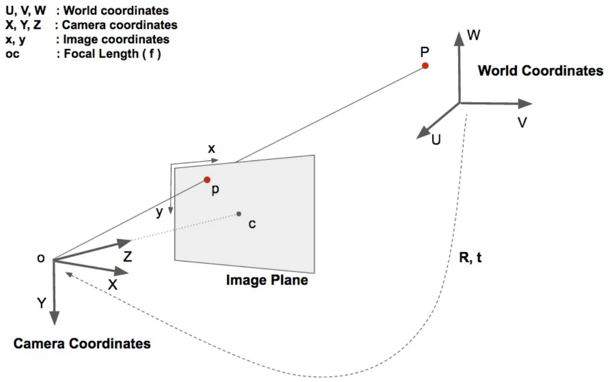


Figure 1: Projection of the 3D camera points and the actual coordinates relative to the 2D image [13]

According to Mallick [13], in Figure 1, the origin point is the center of the camera, the plane references the 2D plane of the image, and it is necessary to find the equations that project the point P of the 3D coordinates of the world relative to point p in the 2D image plane.

Assuming one knows the coordinates of point P in the world (U, V, and W) and also knows the rotation **R** (3x3 matrix) and translation **t** (3x1 vector) compared to the camera coordinates, one can calculate the location (X, Y, Z) of point P in the camera coordinate system using the following Equation 1 [13]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{R} \begin{bmatrix} U \\ V \\ W \end{bmatrix} + t \Rightarrow \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [\mathbf{R} \mid t] \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \tag{1}$$

The expansion of the equation is shown in Equation 2:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \tag{2}$$

If a sufficient number of corresponding points ((X, Y, Z) and (U, V, W)) are known, a linear system of equations can be computed, where r_{ij} and (t_x, t_y, t_z) are unknowns and can be solved trivially [13].

The 6-DoF pose estimation task consists in detecting and/or tracking a 3D object in an image by using the three degrees of freedom in both the rotation of the object and the three degrees of freedom in its translation, thus estimating the pose in six degrees of freedom or 6-DoF. Estimating the 3D object's pose by composing the rotation and translation in a 3x4 pose matrix [4] is possible.

Tekin et al. [15] divide pose estimation into 3 (three) groups, namely:

- Classical methods: using classical computer vision techniques.
- RGB-D based methods: Due to the emergence of cameras with depth sensors.
- CNN-based methods: that make use of Convolutional Neural Networks.

The classical methods use local key points and model feature correspondences. Some other such model-based methods are Hausdorff matching [8], edge-oriented chamfer matching [20], and 3D curve-based chamfer matching [10], for example.

With RGB-D methods came proposals for matching algorithms using models suitable for color images with the depth sensor. Extensions of this work used discriminative learning for cascade detection, increasing detection accuracy. Robots have also used these methods for 3D object recognition, pose estimation, and manipulation [15].

CNN methods use techniques such as ViewPoints, Keypoints, and Render for CNN, categorizing 3D objects and estimating their pose. Another example given by PoseNet [9], is to regress the RGB image to a 6D pose, even though estimating the camera pose is a slightly different task. Single shot architectures like YOLO and SSD also appear in this group by predicting the 2D projections of the corners of the 3D bounding boxes [15].

4 RELATED WORKS

The computer vision community widely studies the problem of adapting vision-based models from one domain to another. Generally, machine learning models need a set of samples to learn features about a specific task, and mapping these features to different domains is a challenge; each model needs to be retrained based on the target domain statistics available. Some approaches investigate how it is possible to create generalizable models, learning invariant features across domains without creating a new dataset for each one [16].

Obtaining labeled samples is one of the most significant problems when training machine learning models in a new task. It demands a lot of time and effort; the dataset must be manually annotated for each object and environment. One solution to this kind of problem is domain adaptation and transfer-learning, where usually the source domain has a different distribution than the target domain ($D_s! = D_t$), making it possible to reduce the impact of obtaining data and reusing available information. [4].

More recent approaches use synthetic data generation for training with domain adaptation, a type of transfer learning. Previously acquired knowledge is transferred to a new data set by mapping and using the information from the source domain. This kind of approach is most often used in DNN architectures. An alternative is to use domain randomization to generate more robust synthetic datasets to reduce the reality gap and be able to do proper training for the model, reducing the use of real datasets for a given problem [17].

Rozantsev et al. [12] used an algorithm capable of learning the parameters that define the environment using a set of real data. The main goal was to decrease the reality gap between the synthetic domain by developing several scenarios that simulated the variations of the target domain, such as illumination, occlusion, and reflection, where the object labeling was done automatically.

Kendall et. all [9] used a deep neural network architecture based on GoogLeNet, which is a pose regression network. This cited convolutional network uses 22 layers with six initial modules and two additional intermediate classifiers, which are discarded when testing the network. The modification made by them adds one more layer than the original, counting only the layers with training parameters. As a result, the three softmax classifiers were replaced by affine regressors, connecting the removed part to each final layer, producing a 7-dimensional pose vector representing 3 dimensions for position and 4 dimensions for orientation of an object. Another modification was in the input image, resizing it to 256 pixels, before performing the 224x224 cut in this image, when entering the GoogLenet network. Using parallel GPU processing, they achieved an increase in computational time from 5ms to 95ms per image.

Su et. all [14] presented an approach using a convolutional neural network, training it exclusively on synthetic images of a channel of objects, with the object of directly regressing the poses of these 6-DoF objects (SynPo-Net). The SynPo-Net proposal is to be a specific network architecture for object pose regression and a proposed domain, and adaptation scheme, transforming real and synthetic images into an intermediate domain, and adapting it to establish correspondences with each other. This system, they say, can be used to estimate the pose of an object in 6-DoF from a single frame or be integrated into a tracking system to provide an initial pose for that system.

By generating a larger number of random variations in the synthetic dataset, some authors [3] [16] have claimed that domain randomization can decrease the impact of training using these types of data when performing 2D object detection. Cunha [4], for example, proposed to adapt these procedures involving the synthetic dataset generation to the case of 6-DoF object pose estimation, training a CNN to estimate the object's pose. He proposed using domain randomization to improve object detection and tracking using the single-shot algorithm. In this case, it was printed 3D textureless objects placed in a scenario with several background objects and occlusions. It was possible to improve the training result using the synthetic dataset generated with the proposed randomization. The difference for this work is that it used a large textured object, where it was necessary to perform 3D modeling to achieve training improvements using domain randomization.

5 METHODOLOGY

With the lack of availability of datasets from the Brazilian voter terminal, it was necessary to build a new one containing different variations related to the voting use case. So, it was possible to proceed with the study. In the next subsections, it is explained how each dataset scenario was built, starting with the synthetic dataset, made from the 3D modeling, until the second real dataset, made from the equipment provided by TRE-PE to the Federal University of Pernambuco.

5.1 Synthetic dataset

5.1.1 3D modeling of the voter terminal. Initially, to create the annotated synthetic dataset, a 3D modeling of the voter terminal was performed using the CAD 3D Inventor software [3]. Inventor is a software that offers professional tools for mechanical design, documentation, and simulation of products [2]. All terminal measurements were verified to create a reliable 3D model.

By fixing the world coordinate system with the origin at the lower right corner behind the terminal, we could map the correspondent annotation in the image plane. From this, it was possible to keep the front view, with the buttons and the terminal screen, pointing toward the viewer. Taking the terminal's base as a reference, it has a value of 39.70 centimeters long (Y-axis), 13.65 centimeters high (Z-axis), and 26.94 centimeters wide (X-axis), as follows in the figure 2.

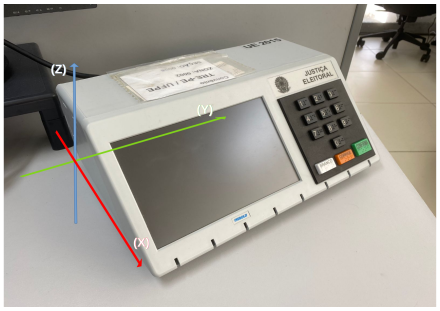


Figure 2: X, Y, Z axes to reference the size of the voter terminal.

On the front, an inclination angle of 45° was identified relative to the base, with a length of 19.30 centimeters, leaving at its top the value of 13.30 centimeters in length, as follows in the figure 3.

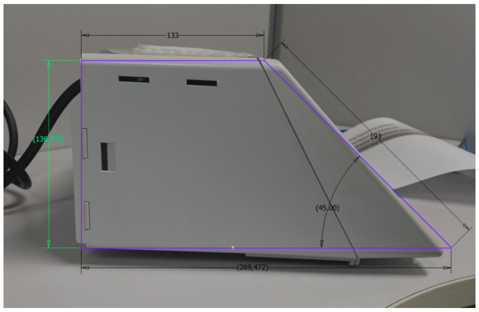


Figure 3: Degree angle sample on the front of the terminal.

The terminal has a screen width of 21.8 centimeters with a height of 13.5 centimeters, arranged at a distance of 1.6 centimeters from its leftmost edge. The keyboard is placed on a panel 13 centimeters wide and 13.7 centimeters high. Each key is 2 centimeters wide and 1.5 centimeters high, except for the keys confirm, correct, and blank, each 3 centimeters wide and 1.5 centimeters high, the confirm being the tallest at 2 centimeters high. The picture 4 shows the final model made from all these parameters.

5.1.2 Synthetic dataset generation. The creation of the annotated synthetic dataset was done using Blender [4]. Blender is an open-source 3D object creation software that can perform modeling, animation, texturing, compositing, rendering, and video editing [1]. Also, it can create scenarios and run Python scripts, managing to automate the domain randomization process. This execution of

[3] https://www.autodesk.com.br/products/inventor/

[4] https://www.blender.org/

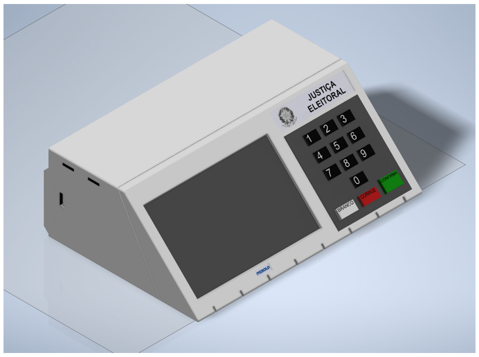


Figure 4: 3D final model of the voter terminal

scripts aligned with external libraries facilitates the loading and construction of new scenarios using 3D object modeling.

To avoid problems with the rotation and size of the object, when annotating the poses, the 3D model file from the elector terminal was imported into a blank project in Blender. Then, the object scale and orientation were adjusted to match the virtual camera coordinates since the Blender coordinate system considers inverted Z and Y axes compared to traditional computer vision software.

All the proportions of the voter terminal were maintained throughout the process; the only change made was in the scale of the object, putting, in Blender, a scale in the X, Y, and Z axes from 1 to 10, thus transforming measures that were, for example, 0.130 meters to 1.30 meters. This increase helps when capturing the object because it visually increases its size keeping the original proportion, taking better advantage of the textures placed during the randomization in the scene.

A total of 3001 images were generated for this dataset with a 680x680 resolution, splitting this total of images into 2101 images for training and 900 images for validation, with the following randomization configuration: the voter terminal is initially in the center of the scene origin. The camera always looks at its front. The scene takes place inside a skybox, where random textures are inserted in all internal faces of the box (top, floor, right side, left side, front, and back). A random number of point lights are inserted initially, ranging from 5 to 40 in the scene, alternating colors, intensity, and ranges. A simulation of the sun's illumination was also inserted, alternating the position and incidence of this illumination on the object.

Distractor objects with defined geometries (cubes, spheres, toroids, pyramid) may or may not be inserted in the first capture to cause some occlusions or shadows. At each new capture, the camera is in a different position from the previous one. It can capture at any angle at which the object is visible, varying the distance. Distractors may or may not cause occlusions or shadows on the object, varying the amount they appear. In this new capture, the lights also change in number, intensity, size, and color. Figure 5 briefly exemplifies this dataset generation process.

5.2 Real Dataset

With the intention of comparing the synthetic data and the real data, a dataset with real ballot box images was built for this paper, as there were no datasets available at the moment. Eighteen printed ARucos markers were placed around the voter terminal, in order to minimize the position variation with the visible markers. This number of markers was necessary to improve the accuracy of the 6-Dof pose annotation using the [7] program, because this is a large object and ends up making a lot of occlusions in the ARucos when changing capture positions. The Kinova Gen3 Lite robot shown in figure 6 with predefined movements was used to standardize the way each variation of the dataset was captured.

An Intel Realsense l515 camera was attached to the robot's claw using a 3D printed holder, responsible for capturing the RGB images in 640x480 resolution and the correspondent depth image, with the two images already aligned in each capture. To perform this capture and annotate the position of the voter terminal, the f2wang program [7] was used, being an open source program for automated annotation of objects in 6-DoF, which uses information from depth images to tell where the object is in the image, and it refines this pose by detecting the markers in the scene.

To use the mentioned program, the 3D model built in the previous section was used, which helps in defining the position to generate the object's mask. It was also necessary to update a small part of the code that used the ARuco detection since there was a change in the way this detection is used by the OpenCV library in newer versions.

To achieve variations of these captures, some changes were made to the environment, to have a greater amount of captured images. Initially, the voter terminal was captured at a distance of 67 centimeters from the base of the robot, using artificial lighting with a yellow tone and a window closed. In this same scenario, a second variation was built turning off the ambient lights and using only the natural light of the room itself, without having to open the previously mentioned window. Examples of these two setups are shown in Figures 7a and 7c.

A third variation was to place the hand, close to the buttons, to make a small occlusion, using the lighting with a yellowish tone again, this occlusion was thought of as a common use of the urn itself. Finally, a closer capture of the robot was made, reducing the distance from the base from 67 centimeters to 56 centimeters, using the yellowish lights again with the window closed, as shown in Figures 7d and 7b.

When performing each capture, a script created in Python, using the robot's own ROS libraries, saved the positions of the claw tip related to the robot base. This information was not used at the time but was made available along with the dataset. As an extra point, the translation and rotation of this gripper point to the camera sensor point concerning the robot base was also carried out. This translation was performed by measuring the distance of the grip point about the position where the camera sensor was fixed, as shown in Figure 8.

First, the translation was applied concerning the base of the robot, without worrying about the attachment angle of the gripper, reducing the X position in each capture by 0.065 meters, since the positive X is towards the front of where the gripper was positioned, and there was an increase of 0.083 meters related to the robot's base Z, since the higher the point, the greater the Z value for that robot. The idea is that when the robot's attachment angle is in the position θX at 90°, θY at 0° and θZ at 90°, only the translation affects the

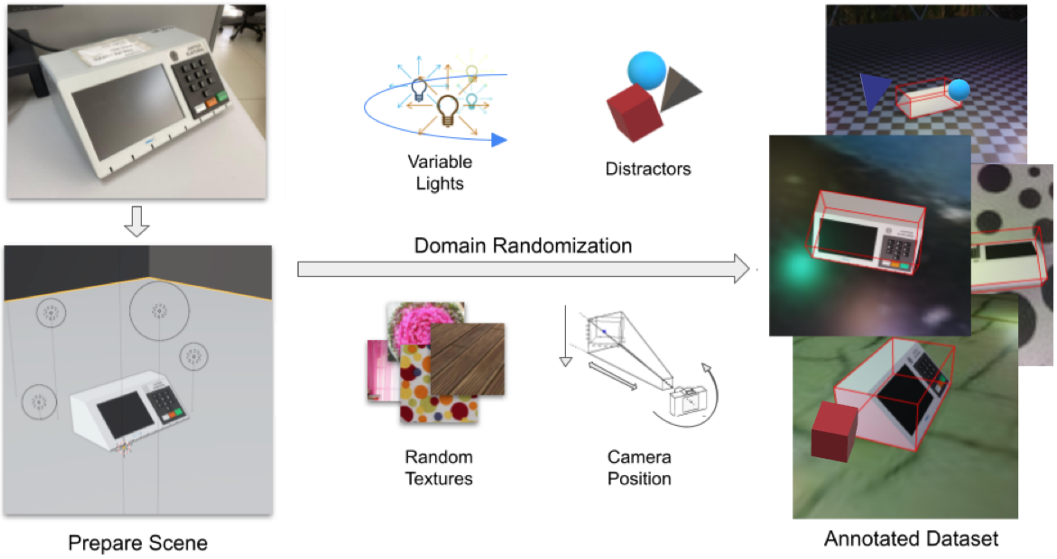


Figure 5: Randomized dataset creation flow

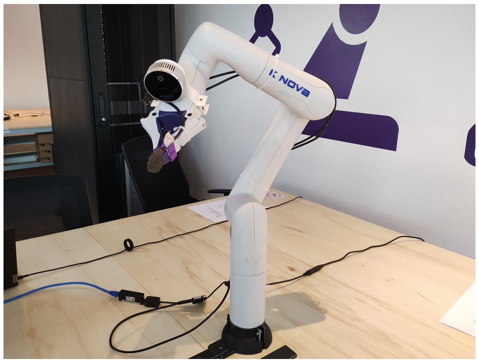
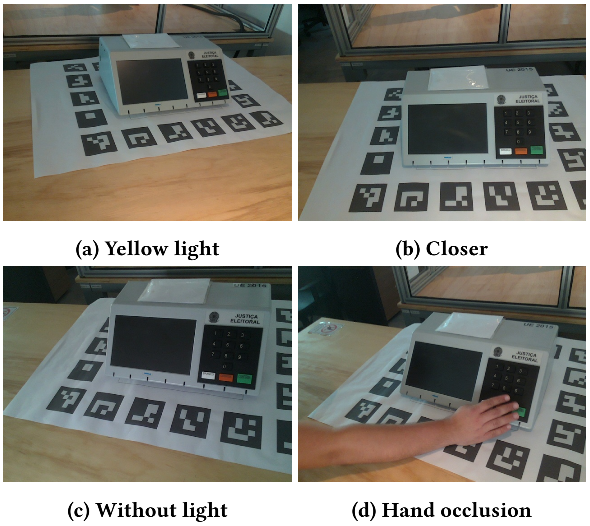


Figure 6: Robot Kinova gen3 lite used to capture the dataset with support and camera



(a) Yellow light

(b) Closer

(c) Without light

(d) Hand occlusion

Figure 7: Differences between captured datasets.

position of the sensor point because with these settings the gripper is aligned forward to the base of the robot.

After performing this translation, the current attachment angle of the gripper was verified for that position, applying the rotation of this new point following the rotation matrices for the X, Y, and Z axes respectively. As the reference angles are 90°, 0°, and 90° for the X, Y, and Z axes, before applying the rotation to the new angle, the difference between the current angle of the gripper and these reference angles is made. All these transformations were performed from a script created in Python using the Scipy and Numpy libraries, building the rotation matrix in each saved position and finding the new points in the captured frames.

The separation of the dataset was as follows: the images with variation in light and proximity were used to train the network (totaling 2083 labeled images), and the images generated with occlusion using a hand were used to validate the network during training (totaling 633 labeled images). So, a total of 2716 labeled images were generated.

5.3 Test Dataset

A fifth set of captures of the real voter terminal was performed during the execution of this work, as a way of having data to test the weights generated during training. The main objective is to verify how the training variation affects the pose estimation, in a different scenario never seen before.

In these captures, the same robot was used with the same movement, but the environment had less lighting, as it was getting to

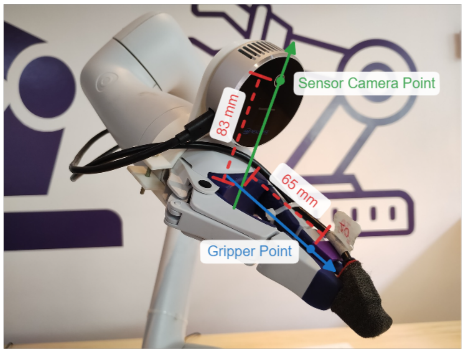


Figure 8: Distance of camera sensor to gripper point

the end of the day, and the lights in the room remained turned off. Another point of difference was in the captured resolution, starting from 640x480 to 960x540 both in the RGB image and in the depth image, having a total of 451 images annotated. Figure 9 shows an example of this dataset.

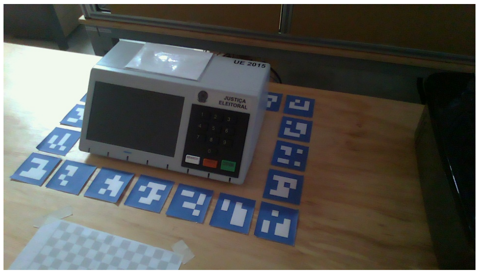


Figure 9: Voter terminal in a different scenario.

5.4 Pose Estimation and Domain Randomization

To perform the detection and pose estimation of the voter terminal, it was used the approach proposed by Tekin et al.[15] and updated by [5]. The approach is based on the single shot 2D object detector network named Yolo9000 [11]. The Tekin[15] work extends the Yolo architecture to predict the 2D projections of the corners of the 3D bounding box around the object. After obtaining these 2D coordinates and knowing its correspondent 3D points from the object's CAD model, it is possible to compute the 6D pose using an efficient PnP algorithm[11] (assuming that the camera calibration was performed and the camera's intrinsics values are known). From this extension, [5] updated the network inputs, adding the camera intrinsic parameters and the projection coefficient, enabling it to use multiple camera types.

When using the synthetic dataset and the real dataset, it was necessary to update the structure of the saved pose annotations, inserting correctly the detected object class at the beginning file, organizing the pose estimation data of each image following the needs of the network and separating the training images from the valid images. For the partitions, approximately 70% of the samples were used for training images and 30% for validation in both cases. A .ply file was also generated from the 3D CAD model of the object, used to capture the projection metrics for the voter terminal during training.

For the generation of synthetic images, this work followed the domain randomization method proposed by [6]. This method follows the following structure for this randomization: one object on a flat surface positioned at the origin of the scene, one camera always pointing to the origin, varying its generation location in each scene, up to 5 floating distractors that appear at random locations to make occlusions in the object, up to 15 point lights varying position and color at each scene generation. All these objects are contained in a skybox (50-meter skybox cube), causing the camera to change its radial distance from the object from 4 meters to 10 meters, azimuth angle from 0° to 180°, and polar angle from 0° to 360°. There was a variation of textures both in the plane in which the object was, and in the internal faces of the box, using the same textures proposed by the work mentioned. The intrinsic parameters of the cameras and the output size of the images were configured to simulate the parameters of the real cameras used in the mentioned work, as well as the exclusion of images where the object was with occlusion greater than 35% of its total, but these two last configurations were not made to generate the images of this paper.

6 RESULTS

The following metrics were used to evaluate the performance of the detection algorithm proposed by [5], following what was done in [15], using the dataset generated in the methodology of this work. The mean corner error is taken by computing the 2D distance of the ground truth values with the predicted points in the image. The 2D Projection accuracy is based on the re-projection accuracy (Rep.), which checks the ground truth with the mean 2D Euclidean distance predicted by the model and considers the value correct if this 2D projection of all mesh vertices is smaller than a threshold defined in pixels (i.e., 10 pixels). As shown in [5] the definition of this metric is $X_{2D} = \pi(K \cdot E \cdot \tilde{v})$, being K the 3x3 intrinsic camera matrix, E is the 3x4 extrinsic parameter matrix (3D objects rotation and translation), $\tilde{v}$ is a vector with CAD model vertex homogeneous coordinates.

3D transformation accuracy (ADD) is related to 3D Pose Accuracy being detailed as $P_e = \frac{1}{N} \sum_{i=0}^{N} \left\|(E_{pr} \cdot V_i) - (E_{gt} \cdot V_i)\right\|$ by [5], where the set V of length N being average 3D Euclidean distance between mesh vertices, E_{gt} as 6DoF ground truth, E_{pr} is the predicted pose, having the accuracy of the pose if $P_e < 0.15d$, where d is the maximum distance between the vertices of the two meshes, showing the accurately the algorithm is estimating the object's pose in the scene. The translation error has to do with the average of errors of the predicted translation vectors and the angle error is related to the predicted rotation vectors, this value also being the average after the execution of the algorithm in the test images. For evaluation purposes, smaller values shown in the metrics related to errors are better (i.e., mean corner, translation, and rotation errors), while larger values related to the mentioned accuracy (i.e., 2D projection and 3D transformations) are better.

All experiments mentioned in this work used a desktop with an Intel Core I7-12700 processor with 12 physical cores at @3.80 GHz frequency, 64 GB of DDR5 RAM at 4000MHz, 2TB SSD HD with 7000 MB/s W/R and 8GB Nvída Geforce RTX3070 Ti graphics card.

In the first generations of training inspired by [6], 1000 images were created from the synthetic dataset, performing a rotation movement through the voter terminal, with approach and distance from the camera, varying the amount and colors of lights, but no distracts were placed. Besides this rotation movement, a movement from top to bottom was also performed to have more variations in the poses of these images. This amount of images was not enough to train the network correctly, because the results of the train stay under 50% when the validation is the same synthetic scenario.

To improve the results, first, there was an increase in the amount of annotated images artificially generated, going from 1000 images to 3000 images, aiming to avoid a possible underfit in the training. Another point that changed was the way of capturing this dataset, now with random poses around the object, so that there were more distinct poses, both for training and testing. Distractors were placed during the process, to cause some occlusions and shadows on the object, thus increasing the variance of the generated dataset. All configurations can be seen in Table 1.

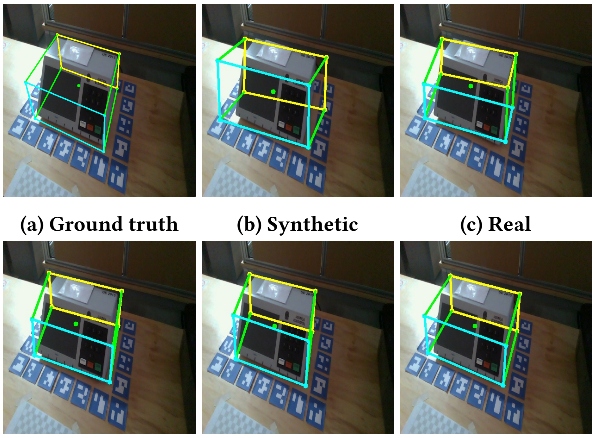
Table 1: Configuration for generating the second synthetic dataset

INFORMATION	VALUE
Number of images	3001
Camera movement	Random around the terminal
Angle camera	Random with terminal always visible
Range number of lights	Between 5 and 40
Light intensive range	Between 50 and 2500
Changing position lights	Yes
Sun angle range	Between 2° and 10° in relation to the terminal
Sun incidence on the scene	Between 1 and 3
Variation sun's positions	Yes
Distractors	Yes
Number distractors	Between 1 and 10

There was also the control on the rotation of the 3D model exported from the voter terminal, adjusting the X, Y, and Z axes, according to the dataset generation program and also the pose detection and estimation program, as mentioned in the methodology section. The batch settings, image size, channels, and the number of key points were kept the same as the first dataset generation, only augmentation was used during this new training, increasing and decreasing the amount of image noise during the run. With this fine-tuning, it was possible to reach a result of 95.33% in the 2D projection and 90.44% in the 3D transformation, as shown in Table 2 for the dataset in the synthetic generation.

Table 2: Generation of training controlling export rotation, increasing dataset, and performing augmentation

INFORMATION	VALUE
Mean corner error	5.37
Acc using 10 px 2D Projection	95.33%
Acc using 1.19 vx 3D Transformation	90.44 %
Translation error	1.37 m
Angle error	3.35 degrees



(a) Ground truth (b) Synthetic (c) Real

(d) 50%/50% (e) 30%/70% (f) 70%/30%

Figure 10: Pose estimation of ballot using the different scenario.

The next step was to train with real images of the voter terminal, so with the real dataset using 2083 images, already with its respective generated mask, it was possible to train the network and obtain the following results shown in Table 3 looking at the same scenario to which it was trained to validate.

Table 3: Generation training using real annotated terminal images

INFORMATION	VALUE
Mean corner error	6.21
Acc using 10 px 2D Projection	89.57%
Acc using 1.19 vx 3D Transformation	97.63%
Translation error	0.65 m
Angle error	2.10 degrees

For this specific training scenario, the algorithm proved to be quite robust, being able to identify the terminal well and estimate its pose correctly, reaching a little over 89% in the 2D projection and 97% in the 3D transformation using only real images in the training.

The first experiment was to test the result of the training performed only with real images of the voter terminal itself. Then, using the test dataset, being the one built with a resolution and lighting very different from the training one, it was possible to estimate the voter terminal pose as shown in Sub-figure 10c, with some error about the fundamental truth.

Table 4: Result of testing domain randomization in 6-DoF pose detection and estimation in a different training scenario.

INFORMATION	S50-R50	S70-R30	S30-R70	Real	Synthetic
Mean corner error	11.14	10.88	11.10	9.75	43.59
2D Projection	79.16%	73.84%	62.53%	51.00%	0.22%
3D Transformation	71.62%	61.20%	95.34%	94.90%	38.36%
Translation error	1.54	1.87	1.03	1.13	9.31
Angle error (degree)	7.73	7.29	7.11	5.45	33.36

Table 5: Result of testing domain randomization in 6-DoF pose detection and estimation in the same training scenario.

INFORMATION	S50-R50	S70-R30	S30-R70	Real	Synthetic
Mean corner error	10.65	11.59	8.45	6.21	27.09
2D Projection	50.39%	50.39%	77.88%	89.57%	0.47%
3D Transformation	82.15%	76.78%	98.10%	97.63%	75.04%
Translation error	1.86	1.75	0.80	0.65	27.09
Angle error (degree)	3.31	3.36	2.56	2.10	15.77

A second experiment performed was to use only synthetic data for training, in the test with the already mentioned set of images. In this scenario, it was possible to notice that the network comes very close to having good detection, but with high error numbers, compared to training only with real images. Figure 10b shows an example of this result.

The next experiment was to use 50% of the total images of the real voter terminal and 50% of the total images of the modeled voter terminal. This was a more balanced scenario, which visibly had a similar result compared to training using only the real images, as shown in Figure 10d. Still, when we see the results expressed in numbers, it managed to do better in the 2D detection of the voter terminal, reducing the estimation of the 3D corners a little.

Another verified point was to decrease the percentage of real images to 30% and increase the rate of synthetic images to 70%. This new scenario had a slightly lower result when compared to the previous result (50% and 50%). There was not much change in the display of the pose estimation, as shown in Figure 10e.

Finally, the inverse mixture of the previous experiment was performed, that is, 70% of the real images of the urn in training and 30% of the synthetic images were used. In this scenario, it was possible to have a better result compared to the previous ones, since it obtained 62.53% in the 2D detection, surpassing the training with only real images, and 95.34% in the estimation of the 3D points of the bounding box, surpassing all the others. Visibly, it is also what shows the best pose estimation as in Figure 10f.

To facilitate the comparison and visualization of the obtained results, Table 4 shows the results of each training using the following terminologies:

- 50% of the synthetic dataset and 50% of the real dataset is S50-R50;
- 70% of the synthetic dataset and 30% of the real dataset is S50-R50;
- 30% of the synthetic dataset and 70% of the real dataset is S50-R50;
- Only using synthetic images of voter terminal is called synthetic;
- Only using real images of the voter terminal is called real.

As a last experiment, a comparison of the results obtained between the training sessions was carried out to verify their impact in relation to the validation dataset, that is, the dataset with the same scenario and the same resolution used in the training. Table 5 shows the values obtained for each of the weights, following the same logic as Table 4 presented above.

It was possible to verify in the table 4 that there was satisfactory performance in all of the models, even when inserting the synthetic dataset in the training. Results show that by adding a controlled amount of synthetic images, it was possible to improve the results, rising from 51% to 79.16% in the 2D Projection, and from 94.9% to 95.34% in 3D Transformation compared with train using only real images of terminal voter for a different scenario of training. For a similar scenario (training and testing in the same scene) the insertion of synthetic data ended up reducing the 2D projection values, but in the 3D transformation metric, training with 30% synthetic data and 70% real data ended up increasing the value to 98.10%.

6.1 Discussion

Based on the results obtained in the use of domain randomization to generate synthetic data from a 3D model of the Brazilian electronic ballot, it was possible to perceive that using only the synthetic dataset became unfeasible in the scenarios proposed by this work. This is probably related to the type of object used in the generation of these annotated images, as a 3D model of the object was used and not its reconstruction, possibly widening the reality gap between them.

In the first experiment, with the testing scenario different from the training scenario, it was possible to verify that the generated models could perform the pose estimation satisfactorily and that using the synthetic data in training improved the task. Analyzing the results of the model generated only from the real dataset, it was verified that when adding controlled amounts of synthetic images (30% of the total value of synthetic images), the accuracy improved, going from 51.00% in the 2D projection to 62.53% and 3D transform from 94.90% to 95.34%.

When the synthetic dataset is inserted together with the real dataset, for algorithm training, in the experiment where the test images are related to the training images, it is noticed that there were losses related to the 2D projection compared to training done only with real data. This reduction could be directly related to the reality gap problem. It is noteworthy that in this scenario the training using 70% of real images and 30% of synthetic images still obtained a better result related to the 3D projection when compared to training only with real images, going from 97.63% to 98.10%, showing a possible future improvement, making the necessary adjustments in the models presented in this work.

7 CONCLUSION

Therefore, it is possible to conclude that the use of domain randomization, to generate synthetic datasets of the voter terminal, using a 3D model of the object, aids in estimating the 6-Dof pose of the terminal when inserted in a controlled way in the scenario. This research opens possible adjustments both in the modeling of the 3D object to generate the synthetic dataset, as well as in the generation of a more distinct real dataset, to improve the impact of the technique. With the results already obtained, it was possible to have an improvement concerning the usage of just real images.

In future work, it is suggested that other annotation techniques can be performed for the real dataset, to validate each pose annotated in this work. More recent techniques regarding pose estimation can also be verified from the available dataset. Based on the quality of the results presented, another work can test the use of the proposed method of pose estimation with domain randomization in the automation process of auditing the voter terminal using the robotic arm.

ACKNOWLEDGMENTS

This work was supported by the research and innovation cooperation project between Softex (funded by the Ministry of Science, Technology, and Innovation through Law 8.248/91 in the scope of the National Priority Program) and CIn-UFPE.

REFERENCES

[1] Blender Foundation (2002). 2022. About Blender. https://www.blender.org/about/.

[2] Autodesk. 2022. Inventor: software avançado de projeto mecânico para suas ideias mais ambiciosas. https://www.autodesk.com.br/products/inventor/overview.

[3] João Borrego, Atabak Dehban, Rui Figueiredo, Plinio Moreno, Alexandre Bernardino, and José Santos-Victor. 2018. Applying Domain Randomization to Synthetic Data for Object Category Detection. arXiv:1807.09834 [cs.CV]

[4] Kelvin Batista Da Cunha. 2019. *Detecção de objetos em 6-DoF em tempo real utilizando técnicas de aprendizagem profunda.* Master's thesis. Universidade Federal de Pernambuco.

[5] Kelvin B. Da Cunha, Caio Brito, Lucas Valença, Lucas Figueiredo, Francisco Simões, and Veronica Teichrieb. 2022. The impact of domain randomization on cross-device monocular deep 6DoF detection. *Pattern Recognition Letters* 159 (2022), 224–231. https://doi.org/10.1016/j.patrec.2022.04.008

[6] K. B. da Cunha, C. Brito, L. Valenca, F. Simoes, and V. Teichrieb. 2020. A Study on the Impact of Domain Randomization for Monocular Deep 6DoF Pose Estimation. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI).* IEEE Computer Society, Los Alamitos, CA, USA, 332–339. https://doi.org/10.1109/SIBGRAPI51738.2020.00052

[7] F2Wang. 2021. Object Dataset Tools. https://github.com/F2Wang/ObjectDatasetTools.

[8] Daniel P Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. 1993. Comparing images using the Hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence* 15, 9 (1993), 850–863.

[9] A. Kendall, M. Grimes, and R. Cipolla. 2015. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *2015 IEEE International Conference on Computer Vision (ICCV).* IEEE Computer Society, Los Alamitos, CA, USA, 2938–2946. https://doi.org/10.1109/ICCV.2015.336

[10] K. Ramnath, S. N. Sinha, R. Szeliski, and E. Hsiao. 2014. Car make and model recognition using 3D curve alignment. In *2014 IEEE Winter Conference on Applications of Computer Vision (WACV).* IEEE Computer Society, Los Alamitos, CA, USA, 285–292. https://doi.org/10.1109/WACV.2014.6836087

[11] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 6517–6525. https://doi.org/10.1109/CVPR.2017.690

[12] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. 2015. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding* 137 (2015), 24–37.

[13] Mallick Satya. 2016. Head Pose Estimation using OpenCV and Dlib. https://learnopencv.com/head-pose-estimation-using-opencv-and-dlib/

[14] Yongzhi Su, Jason Rambach, Alain Pagani, and Didier Stricker. 2021. Synponet—Accurate and fast CNN-based 6DoF object pose estimation using synthetic training. *Sensors* 21, 1 (2021), 300.

[15] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. 2018. Real-Time Seamless Single Shot 6D Object Pose Prediction. arXiv:1711.08848 [cs.CV]

[16] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE Computer Society, Los Alamitos, CA, USA, 23–30. https://doi.org/10.1109/IROS.2017.8202133

[17] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. 2018. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).* IEEE Computer Society, Los Alamitos, CA, USA, 1082–10828. https://doi.org/10.1109/CVPRW.2018.00143

[18] TSE. 2020. Veja como é feita a auditoria de funcionamento das urnas eletrônicas. https://www.tse.jus.br/comunicacao/noticias/2020/Dezembro/veja-como-funciona-a-auditoria-de-funcionamento-das-urnas-eletronicas. (Accessed on 10/10/2022).

[19] TSE. 2022. Plenário do TSE triplica número de urnas eletrônicas auditadas no dia da eleição. https://www.tse.jus.br/comunicacao/noticias/2022/Marco/plenario-do-tse-triplica-base-amostral-de-urnas-eletronicas-auditadas-no-dia-da-eleicao. (Accessed on 2023/01/16).

[20] A. Veeraraghavan, R. Chellappa, O. Tuzel, and M. Liu. 2010. Fast directional chamfer matching. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE Computer Society, Los Alamitos, CA, USA, 1696–1703. https://doi.org/10.1109/CVPR.2010.5539837

[21] Mei Wang and Weihong Deng. 2018. Deep visual domain adaptation: A survey. *Neurocomputing* 312 (2018), 135–153. https://doi.org/10.1016/j.neucom.2018.05.083

[22] Y. Xu, K. Lin, G. Zhang, X. Wang, and H. Li. 2022. RNNPose: Recurrent 6-DoF Object Pose Refinement with Robust Correspondence Field Estimation and Pose Optimization. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE Computer Society, Los Alamitos, CA, USA, 14860–14870. https://doi.org/10.1109/CVPR52688.2022.01446

[23] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. 2023. Domain Generalization: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 4 (2023), 4396–4415. https://doi.org/10.1109/TPAMI.2022.3195549