Braille character detection using deep neural networks for an educational robot for visually impaired people

Diego Gonçalves*, Gabriel G. Santos*, Márcia B. Campos†, Alexandre M. Amory*, Isabel H. Manssour*
* School of Technology - PUCRS, Porto Alegre, Brazil
Email: dlngoncalves@gmail.com, gabriel.giordani@acad.pucrs.br, amamory@gmail.com, isabel.manssour@pucrs.br
†CESUCA University Center, Cachoeirinha, Brazil
Email: marcia.campos@cesuca.edu.br

***Abstract*—Teaching computer programming to the visually impaired is a difficult task that has sparked a great deal of interest, in part due to its specific demands. Robotics has been one of the strategies adopted to help in this task. One system that uses robotics to teach programming for the visually impaired, called Donnie, has as its key part the need to detect Braille characters in a scaled-down environment. In this paper, we investigate the current state-of-the-art in Braille letter detection based on deep neural networks. For such, we provide a novel public dataset with 2818 labeled images of Braille characters, classified in the letters of the alphabet, and we present a comparison among some recent detection methods. As a result, the proposed Braille letters detection method could be used to assist in teaching programming for blind students using a scaled-down physical environment. The proposal of EVA (Ethylene Vinyl Acetate) pieces with pins to represent Braille letters in this environment is also a contribution.**

***Index Terms*—Educational Robotics, Assistive Robotics,Deep Neural Networks, Braille.**

I. Introduction

There is a great concern of the community in developing new ways to teach programming to the visually impaired learners [1]–[3]. Several strategies have been proposed in the last years to make the programming learning process more accessible for these learners. Among several strategies that have been adopted, as the use of physical artifacts to provide tactile information [3], robotics has been used as a tool to assist in this task [4]–[7].

In previous work [8], we presented a new educational/assistive robot called Donnie and its programming environment, which allows both the practice of computational thinking and the training of orientation and mobility skills that is inclusive for visually impaired people[1].

It's well known that visually impaired people have orientation and mobility difficulties when facing a new environment. Let us assume a scenario where a blind student moves to a new school. One way this project can help the student is by modeling a scaled-down version of the school's floorplan. Then the student can practice some programming skills to perform mobility challenges in this scenario. For instance, the student can practice going from his classroom to the bathroom. The shorter distance he can move the robot, the more points are earned. In parallel, the student is also learning how to move independently in this new environment. However, there are some challenges to reach this goal. How can the blind student know he is driving the robot to the correct direction? How can the student know that he is getting closer or reached the goal? The approach chosen for this project is to place markers in the environment that will give hints to the students of the robot's location.

When executing the virtual scenario by software simulation, this task of object or place identification can be easily accomplished by using simulated fiducial markers or tags spread in the virtual environment. However, when using the real robot, the same task is much more complex. It is necessary to set up an environment with objects that must be recognized by the robot. For this reason, Donnie has a camera on its head with horizontal movement, to search for objects.

Using normal markers such as QR Code would be easy for the robot perception, but it is not useful for blind users that need to assemble and identify the objects in a dynamic physical environment. One of the goals of the project was accessibility for both users of the simulator and of the actual robot. Thus, the alternative is to build markers using the Braille alphabet, which can be both felt by the users and detected by the robot camera, as exemplified in Figure 1.

In this paper, we present a new approach for the detection of markers in the Braille alphabet using Convolutional Neural Networks to help in the assistive physical environment of the Donnie robot. Our main contributions include a novel public dataset [2] with 2818 labeled images of Braille characters, classi-

We gratefully acknowledge the support of NVIDIA Corporation with the donation of Titan Xp GPU used for this research. The result of this work was achieved in cooperation with HP Brasil Indústria e Comércio de Equipamentos Eletrônicos LTDA. using incentives of Brazilian Informatics Law (Law nº 8.2.48 of 1991). This work was supported by the CNPq/MCTIC/SECIS Nº 20/2016, National Council for Scientific and Technological Development – CNPq. It was also partially supported by PUCRS (Edital 01/2018 - Chamada Geral).

[1]Extensive documentation and manuals about Donnie are available at https://github.com/lsa-pucrs/donnie-assistive-robot-sw

[2]https://www.kaggle.com/dlngoncalves/donnie-braille

fied in the letters of the alphabet; a proposal of EVA (Ethylene Vinyl Acetate) pieces with pins to represent Braille characters in the physical environment that need to be recognized; and a comparison among the use of recent methods for detection of Braille alphabet.

The remainder of this paper is organized as follows. We briefly describe some related work in Section II. Section III describes our novel dataset. The detection methods we employ are presented in Section IV. Our results and main findings regarding the suitability of employing state-of-the-art deep neural approaches for Braille character recognition are described in Section V. Finally, we end this paper with our conclusions and future work directions in Section VI.

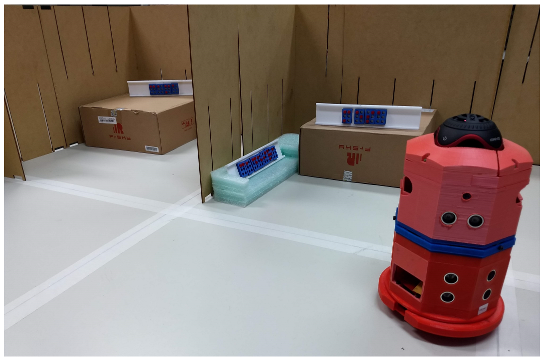


Fig. 1. Example of the robot in a real environment. The Braille markers are in blue with dots in red.

II. Background

Optical Braille Recognition (OBR) is the sequence of steps involved in converting the contents of images of Braille text documents into natural language. A description of the Braille script and a general methodology for OBR is presented by Isayed and Tahbou [9]. Their work compares different OBR techniques based on criteria such as image acquisition and Braille dots detection techniques. The authors notice that most of the techniques found in literature use scanner image acquisition, with few using mobile or standard cameras. They also observed a lack of artificial intelligence approaches as neural networks. Later, a review of some OBR methods that use a camera for image acquisition was presented by Nugroho et al. [10].

One early approach that used a multi-layer perceptron neural network for Braille character recognition was presented by Morgavi and Morando [11]. Another early work [12] used image processing techniques and probabilistic neural networks to recognize and transcribe Braille documents. Zhang and Yoshino [13] use images acquired by a mobile phone with image processing techniques for the recognition of Braille used in Japan. Li and Yan [14] used Support-Vector Machine to recognize Braille characters from an image. Artificial neural networks were also recently used by Waleed [15] to identify numbers in Braille from images.

Recently, deep learning methods have been used for OBR, such as the stacked denoising autoencoder proposed by Li et al. [16], which achieved good results compared to traditional methods. Shimomura et al. [17] also used deep learning techniques to convert Braille books into machine-readable electronic data.

Despite the different detection techniques used, almost all the works listed above use images acquired by traditional flat-bed scanners. Moreover, few of them use recent deep learning techniques. In this work, we use images acquired by a common camera (thus the Braille characters are viewed from different angles) and deep neural networks, which have shown promising results for OBR.

III. A Novel Braille Dataset

To the best of our knowledge, there is no public dataset of images of Braille characters, at least not one large enough to be used with machine learning techniques. In the review of OBR techniques presented by Isayed and Tahboub [9], the authors mention that there is no benchmark to test the algorithms and researchers use their database and Braille images to measure performance.

Thus, for this work, a dataset of images of Braille characters was produced, with the goal of using it to train a convolutional neural network for the detection and recognition of such characters. The images produced were of cropped individual letters of the Portuguese Braille alphabet. Some of such images used in this training are shown in Figure 2.

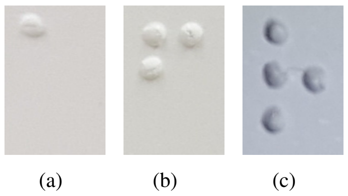
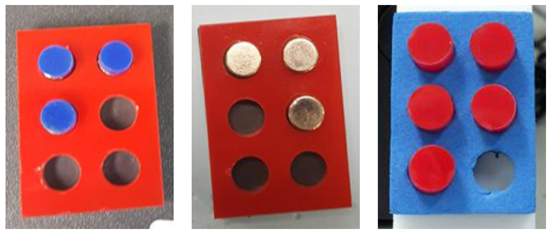


Fig. 2. A selection of Braille character images. (a) Braille character *a*. (b) Braille character *f*. (c) Braille character *r*.

For the Donnie project, it was necessary to be able to visually detect and classify Braille characters in a scaled-down real-world environment, with the robot's camera being used for image acquisition, as exemplified in Figure 1. The robot should be able to detect characters in distances up to a meter. Additionally, scenes needed to be dynamic and easily modifiable by users without additional difficulties for the visually impaired. However, this detection of Braille characters in a real-world environment proved to be challenging, because they are small, usually showing very little visual contrast with their backgrounds, making them hard to detect. Some characters are also very similar, a characteristic that makes them hard to classify from a distance when relying only on visual information.

Considering these problems and the needs of the Donnie project, we designed and produced small acrylic pieces that can be used to represent Braille characters, as shown in Figures 3(a) and 3(b). The pieces are larger and have significantly more contrast with the environment than traditional Braille characters printed on paper. Moreover, they facilitate the task of identifying objects by blind users.



(a) (b) (c)

Fig. 3. Types of acrylic Braille pieces. (a) Pins representing characters. (b) Magnets representing characters. (c) EVA Braille pieces with acrylic pins.

We used smartphones and the robot camera to collect images to compose the dataset. The images contained the pieces in different backgrounds and positions. Thus, we were able to capture several images of the Braille character from different angles and with lighting variations more quickly. Since the camera of the Donnie robot uses a resolution of 640×480, the images acquired with smartphones were resized to this size by a Python script before annotation. To annotate the images, we used the software labelImg[3]. In the end, we had a total of 26 classes, one for each letter.

Initially, the dataset was composed of images of two types of Braille pieces, both containing a magnetic layer overlaid with acrylic and with six holes. The first type showed in Figure 3(a), used acrylic colored pins attached to magnets to fit into the holes and form the characters. The second one showed in Figure 3(b), used only magnets to fit into the holes. This dataset had 3527 images but was incomplete since 6 letters did not have any instances. The testing phase showed that the neural network couldn't identify any instance of some letters.

Since the results weren't satisfactory, we decided to compose a dataset where all the pieces had similar pins. However, after some visually impaired people had manipulated the Braille pieces, they suffered changes in its design due to the difficulty encountered to use the magnets to compose the characters. The new pieces, shown in Figure 3(c), have the background part made of EVA, with acrylic pieces of a different color for the pins used to fit into the holes and represent the characters. This configuration was preferred by visually impaired users with regards to usability and accessibility after they performed some writing and reading activities with the pieces. The colors were chosen based on preliminary detection results.

To compose the new dataset, we removed all images that had pieces that only used magnets to represent characters. After the removal, the dataset remained with 991 images of pieces that used colored pins attached to magnets. Then, we added images of the new EVA pieces so that every letter had at least 100 different images. At the end of the process, the data set was with 2818 images separated in the 26 letters of the alphabet.

[3]https://github.com/tzutalin/labelImg

After several tests, we noticed that videos containing words composed by the Braille pieces presented worse results than the videos with individual Braille pieces. Thus, we decided to add annotated images of sequences of characters to the dataset. These images presented the Braille pieces side by side in a white acrylic base, as shown in Figure 4. A total of 217 images were annotated, with sequence length varying between 4 and 10 characters.

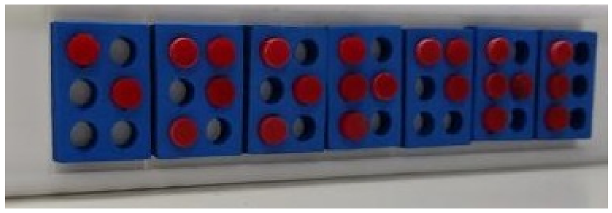
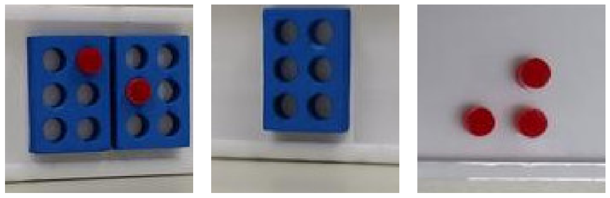


Fig. 4. Example of a sequence of pieces.

For the sake of testing false positives an additional 32 images were captured with the following scenarios:

- Pieces with configurations that do not represent a valid character;
- Two pieces forming one character using one column of each piece;
- Pieces with no configuration;
- Acrylic pins forming a character out of the base.

A sample of those scenarios is shown in Figure 5.



(a) (b) (c)

Fig. 5. Invalid configurations. (a) One character in two pieces. (b) Empty background. (c) Pins without a background.

IV. Detection Methods

In recent years Convolutional Neural Networks (CNN) have become very successful in image classification and object detection tasks, as shown in one pioneer work [18] that won the 2012 ImageNet image classification competition. They are one of the deep learning methods, which has many convolutional layers. In this work, we used state-of-the-art deep learning techniques for object detection and classification on our custom Braille dataset, and we evaluated their performances. The used techniques were Mask R-CNN and YOLOv2. In the next sections, we briefly describe and compare these techniques.

A. Mask R-CNN

Mask R-CNN [19] is a framework for object instance segmentation that extends the Faster R-CNN [20]. Both are based on the Region-based Convolutional Neural Network (R-CNN) [21] approach to bounding-box object detection.

R-CNN first generates regions of interest for potential bounding boxes and then runs a classifier on those regions. Faster R-CNN consists of two stages: in the first one occurs the proposition of candidate bounding boxes; the second stage use region of interest pool to extract features and perform classification and bounding box regression.

Mask R-CNN extends Faster R-CNN by adding the calculation of a binary mask to each region of interest. The calculation of the binary mask occurs in parallel with the second stage of Faster R-CNN, and the first stage continues the same.

B. *YOLO - You Only Look Once*

YOLO [22] presents a new approach to object detection. It treats object detection as a regression problem, using a single convolutional network to simultaneously predict bounding boxes and generate class probabilities for those boxes. This is done by analyzing the entire image during training, dividing it into a grid, and for each grid cell predicting a number of bounding boxes, the confidence score of those boxes, and a set of class probabilities.

Using this approach YOLO can achieve good results in terms of precision metrics, being the fastest object detector. One downside of the YOLO approach is that it has lower precision than other methods. Also, the spatial constraints imposed by the small number of bounding boxes predicted by each grid cell limit the number of nearby objects YOLO can predict. It also struggles with small objects [22].

V. Experimental Analysis

To validate the use of our dataset in the task of training models for detecting Braille characters and use on the Donnie project, we performed experiments with both the YOLO and Mask R-CNN approaches on it. The training was carried out on a dual Intel Xeon E5-2620 system, with 48GB of RAM, and an NVIDIA Titan Xp GPU with 12GB of VRAM.

For better initialization, we used pre-trained weights from ImageNet in the networks' backends. We also used data augmentation techniques to enlarge our dataset, such as blurring, sharpening, adding noise, and rotating the images. However, due to the similarities between characters, some techniques, as flipping the images, were avoided. To train the models, 80% of the dataset was used for training and the rest of the images were used for validation.

A. *Ablation Experiments*

We performed an ablation experiment on the networks by analyzing the use of different backbone architectures. Backbone architectures are used for feature extraction over an image before classification. We trained the YOLO network using the Full YOLO, Tiny YOLO, Inception v3 [23], SqueezeNet [24] and MobileNet [25] backbones. Mask R-CNN was trained using both the ResNet101 and the ResNet50 backbones. The backbone architectures differ mainly in relation to the number and organization of their convolutional layers. All models were trained for 200 epochs.

Another experiment was done to understand what features of the characters were learned by the networks. The obtained results are presented in the next sections.

B. *Comparative Performance*

The evaluation metric used to measure the accuracy of the trained models was mean Average Precision (mAP) as defined by the COCO dataset [26]. In this case, predictions are considered correct when their intersection over union (IOU) values regarding the ground truth bounding box are over 0.5. Scores for the models are shown in Table I.

TABLE I
mAP Scores for the different models trained.

Model	mAP Score
YOLO Full	0.9419
YOLO Tiny	0.9167
YOLO Inception V3	0.8872
YOLO SqueezeNet	0.8229
YOLO MobileNet	0.7905
Mask R-CNN (Resnet50)	0.9345
Mask R-CNN (Resnet101)	0.9248

Table II shows the average time in seconds that each of the models took to analyze a still image or a single frame of video, and generate predictions. Detection tests were run on the same platform where training was performed, using an NVIDIA Titan Xp GPU. Images used are always in the same resolution as those of the camera of the Donnie robot, i.e., 640x480 pixels.

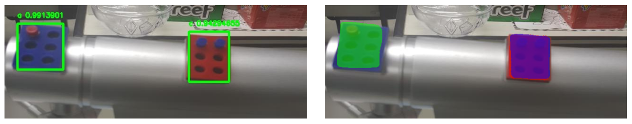
TABLE II
Detection times for the different models trained.

Model	Detection Speed (seconds)
YOLO Full	1.61
YOLO Tiny	1.21
YOLO Inception V3	4.15
YOLO SqueezeNet	0.95
YOLO MobileNet	1.89
Mask R-CNN (Resnet50)	3.76
Mask R-CNN (Resnet101)	4.40

C. *Performance Analysis*

What we can gather from the mAP scores is that all the trained models achieved good accuracy results. Of note, we see that the bigger number of layers in the resnet101 network actually decreases accuracy in the Mask R-CNN model, when compared to the smaller resnet50 network. The YOLO network benefits from the deeper architectures such as Full YOLO.

The greatest performance difference between the models is not their accuracy, but the detection time, as shown in Table II, with the YOLO models generally performing over three times faster than both Mask R-CNN configurations on the same hardware. This is expected, as one of the goals of the YOLO model is detection speed, with some implementations running detections at real-time speeds. Only the YOLO model using the Inception V3 backbone performed at similar speeds



(a) (b)

Fig. 6. Comparison between YOLO and Mask R-CNN prediction results. (a) YOLO bounding boxes. (b) Mask R-CNN detection masks.

of Mask R-CNN. There is not a clear correlation between detection speeds and mAP scores.

D. Qualitative Analysis

A benefit of the Mask R-CNN approach lies in the pixel mask it generates for predicted results. This mask is useful for increased precision in the detection of specific shapes of objects. A comparison of the bounding boxes predicted by the YOLO model and the masks of Mask R-CNN is shown in Figure 6. In our case, the shape of all objects of interest is the same, so this benefit is not particularly relevant. As such, given the good accuracy obtained and fast detection speeds, we decided to use the Full YOLO model for integration with the Donnie robot.

E. Feature Learning

As described in Section III, the characters in our dataset are formed from the same base components, a background piece, and a number of pins. This makes the classes to be detected and classified by the models very similar, with few distinctive visual characteristics.

It was of our interest to discover if those characteristics were enough to differentiate the characters, as the trained models were not originally proposed for this specific task. We decided to investigate which parts of the characters were more relevant to the learning. We did this by testing the trained models on images such as the ones shown in Figure 5.

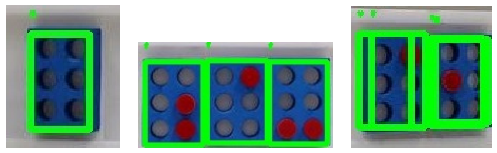
Figure 5(a) with one character in two background pieces was chosen to simulate a situation where two pieces are placed next to each other and a potentially valid character is accidentally formed between them. This test was intended to discover if the shape of the individual pieces was relevant compared to the contrast of the pins against the background color.

The empty background piece (Figure 5(b)) and the loose pins (Figure 5(c)) images were used to test the recognition of the models on the specific parts of the characters.

Another test was performed with images such as Figure 7(b), displaying invalid character configurations. This was to allow us to determine how characteristics such as color density and distribution, and the orientation of the characters were being used in the classification process.

None of the trained models detected the loose pins as a valid character. By contrast, some of the models detected the empty background piece as one or more characters, as shown in Figure 7(a). This suggests that the background pieces have a stronger influence on the detection than the pins.

Some models also detected false positives in images with invalid characters, like in Figure 7(b). However, there was no correlation between the characters the models reported finding and the configurations in the images. The reported characters did not have a similar amount of pins, or pin positions that were reverse of the ones in the images. In addition, the confidence of those detections was lower than the one of the valid characters.



(a) (b) (c)

Fig. 7. Detections performed on a number of test configurations. (a) Detection in empty background piece. (b) Detections in invalid configurations. (c) Detections in two pieces with one combined character.

False positives also appeared in detections performed in the situation exemplified in Figure 5(a). This is shown in Figure 7(c). But the false positives were of wrong characters in the individual background pieces, not of the character between them.

The false positives on empty pieces and invalid characters suggest that the color of the pins and their positions don't provide enough differentiation for the classes.

F. Word Detection

Some of the trained models have problems in accurately detecting and classifying images with a large number of characters. This is more frequent in the models trained with smaller backbone architectures such as Tiny YOLO. An example is shown in Figure 8(a).

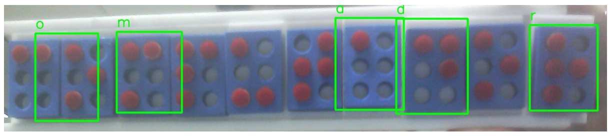
Another problem arises with the detection of more than one character in a single piece. An example is presented in Figure 8(b). However, we solve this problem by choosing the detected character with the highest confidence value.

As our goal was to detect complete words in the images, characters detected are sorted from left to right and grouped by how close their position is. A single wrong character prediction can cause an entire word to be wrong, despite the confidence of the detection of the other characters. To fix this problem, the words are passed through the spellchecking script pyspellchecker [4]. This also helps users who might have made a mistake placing pins.

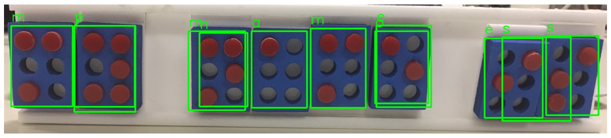
VI. Conclusions and Future Work

In this paper, we presented a system for detecting Braille-like characters in scaled-down real-world environments. Our contributions are mainly regarding the novel public dataset of labeled images of those Braille characters classified in the letters of the alphabet, and the controlled comparison among the deep learning state-of-the-art methods for detecting and classifying the characters. Another novelty is the proposal of the EVA pieces to represent Braille characters to allow the generation of the dataset and to allow the visually impaired

[4]https://pypi.org/project/pyspellchecker/



(a)



(b)

Fig. 8. Errors in word detection. (a) Partial Detection of Words. (b) Detection of overlapping characters in the same pieces.

to set up the real environment that must be recognized by the Donnie robot.

We are now working to embed the models in the Donnie robot, and further developing them with the aim of helping the visually impaired. Thus, for future work, we intend to perform tests with users that present visual impairment. We also intend to further explore possibilities of detecting regular embossed Braille characters in real-world locations, not only in the context of the Donnie system but possibly as a general tool to assist Braille literacy.

References

[1] L. Luque, L. O. Brandão, E. Kira, and A. A. F. Brandão, "On the inclusion of learners with visual impairment in computing education programs in brazil: practices of educators and perceptions of visually impaired learners," *Journal of the Brazilian Computer Society*, vol. 24, no. 1, p. 4, Feb 2018. [Online]. Available: https://doi.org/10.1186/s13173-018-0068-0

[2] M. Konecki, N. Ivković, and M. Kaniški, "Making programming education more accessible for visually impaired," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2016, pp. 887–890.

[3] A. Hadwen-Bennett, S. Sentance, and C. Morrison, "Making programming accessible to learners with visual impairments: A literature review," *International Journal of Computer Science Education in Schools*, vol. 2, no. 2, 2018.

[4] S. Ludi and T. Reichlmayr, "The use of robotics to promote computing to pre-college students with visual impairments," *Trans. Comput. Educ.*, vol. 11, no. 3, pp. 20:1–20:20, Oct. 2011. [Online]. Available: http://doi.acm.org/10.1145/2037276.2037284

[5] R. Dorsey, C. H. Park, and A. Howard, "Developing the capabilities of blind and visually impaired youth to build and program robots," 2014.

[6] S. L. Ludi, L. Ellis, and S. Jordan, "An accessible robotics programming environment for visually impaired users," in *Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility*. ACM, 2014, pp. 237–238.

[7] R. P. Barros, A. M. F. Burlamaqui, S. O. de Azevedo, S. T. de Lima Sa, L. M. G. Goncalves, and A. A. R. S. da Silva Burlamaqui, "Cardbot - assistive technology for visually impaired in educational robotics: Experiments and results," *IEEE Latin America Transactions*, vol. 15, no. 3, pp. 517–527, March 2017.

[8] G. H. M. Marques, D. C. Einloft, A. C. P. Bergamin, J. A. Marek, R. G. Maidana, M. B. Campos, I. H. Manssour, and A. M. Amory, "Donnie robot: Towards an accessible and educational robot for visually impaired people," in *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, Nov 2017, pp. 1–6.

[9] S. Isayed and R. Tahboub, "A review of optical Braille recognition," *2015 2nd World Symposium on Web Applications and Networking, WSWAN 2015*, pp. 1–6, 2015.

[10] B. Nugroho, I. Ardiyanto, and H. A. Nugroho, "Review of optical braille recognition using camera for image acquisition," in *2018 2nd International Conference on Biomedical Engineering (IBIOMED)*. IEEE, 2018, pp. 106–110.

[11] G. Morgavi and M. Morando, "A neural network hybrid model for an optical braille recognitor," in *International Conference on Signal, Speech and Image Processing*, vol. 2002, 2002.

[12] L. Wong, W. Abdulla, and S. Hussmann, "A software algorithm prototype for optical recognition of embossed braille," in *Proceedings - International Conference on Pattern Recognition*, vol. 2, 2004, pp. 586–589.

[13] S. Zhang and K. Yoshino, "A braille recognition system by the mobile phone with embedded camera," in *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*, Sep. 2007, pp. 223–223.

[14] J. Li and X. Yan, "Optical braille character recognition with support-vector machine classifier," in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, vol. 12. IEEE, 2010, pp. V12–219.

[15] M. Waleed, "Braille identification system using artificial neural networks," *Tikrit Journal of Pure Science*, vol. 22, no. 2, 2017.

[16] T. Li, X. Zeng, and S. Xu, "A deep learning method for Braille recognition," *Proceedings - 2014 6th International Conference on Computational Intelligence and Communication Networks, CICN 2014*, pp. 1092–1095, 2014.

[17] Y. Shimomura, H. Kawabe, H. Nambo, and S. Seto, "Construction of restoration system for old books written in braille," in *International conference on management science and engineering management*. Springer, 2017, pp. 469–477.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2980–2988.

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2014. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2014.81

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[24] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[25] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[26] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312