Automated Testing of Refreshable Braille Display

Shivam Kumar Singh, Sujit Kumar Chakrabarti, Dinesh Babu Jayagopi

Automated Testing of Refreshable Braille Display

Shivam Kumar Singh[1] [0000−0002−2363−2866], Sujit Kumar Chakrabarti[2], and Dinesh Babu Jayagopi[3]

[1] IIIT-B Bangalore, India
shivamkumar.singh@iiitb.org
[2]IIIT-B Bangalore, India
sujitkc@iiitb.ac.in
[3]IIIT-B Bangalore, India
jdinesh@iiitb.ac.in

Abstract. A majority of visually impaired population of India and other developing economies live in poverty. Accessibility without affordability has little meaning to this population. Assistive technology has great potential to make education accessible to this population, e.g. through refreshable Braille display devices. However, most existing solutions in this space remain out of reach for these users due to high cost. Innovation in data science and software engineering can play an important role in making assistive technological solutions affordable and accessible. In this paper, we present a machine-learning based automated testing approach that has played an important role in enabling us to design one of the most affordable refreshable Braille display devices of the world. The key component of our approach is a visual inspection module (VIM) created using Convolutional Neural Networks(CNNs). In our experiment, our model was able to detect malfunction of a Refreshable Braille display with 97.3% accuracy. Our model is small enough to be run on a battery-powered computer in real- time. Such accurate automatic testing methods have the potential to significantly reduce the cost of RBDs.

Keywords: Automated testing, RBD, CNN, OpenCv, ORB, DCT, KNN, VIM.

1 Introduction

Visual impairment is a global health issue. it is estimated that globally, there are 441 million visually impaired people encompassing range of impairment from mild levels to blindness. Over 90% of these live in developing countries like India [5]. It is estimated that India has more than 62 million people with some form of visual impairment out of which more than 8 million suffer from permanent blindness [25]. Education and integration of visually impaired people is a fundamental challenge that we need to solve. In this quest, we have taken help of many assistive technologies, one such technology is Refreshable braille display.

RBD is an electromechanical device that allows a visually impaired person to read contents of a text file using a refreshable tactile feedback reader. One of the most

important components of this device is the actuator which powers a retractable pin either up or down based on electrical voltage given by the control unit. Eight of these retractable pins combine to make a cell and multiple cells combine to make a complete display. As discussed, every cell having eight mechanical components means that they are prone to degradation and failure at the time of manufacturing and after heavy use. While manual testing is possible, it is very expensive and time-consuming, and also possible only much later in the production cycle – after the product integration. In the production stage, manual testing becomes a significant bottleneck in scaling up the production. Therefore, it is very important that the testing process be automated if we want to scale up the production process and make the product economically viable.

This paper proposes a new method to test an RBD using a digital image processing technique based on deep learning. In this method, we capture an image of the cell being tested. After processing this image, we feed this to our convolutional neural network [22] (CNN) which predicts a value. We compare this value to input we gave to the cell. If they both match with each other, we declare the cell to be error-free.

We have tested both traditional feature engineering and modern feature learning approach as explained in section 4.1 and we have also experimented with multiple Neural Network architecture and hyper-parameters as explained in section 4.2. After performing these experiment we have chosen feature learning approach with an architecture with excellent accuracy and acceptable inference time.

As a first step, we created a dataset by capturing images of different configurations of a cell. In all, 4096 photographs were taken for 256 different cell configurations with 16 different illumination conditions. The dataset was further augmented to over 12000 images using standard techniques. About two third of this dataset, labelled with the cell configurations they corresponded to, were used to train a CNN model to identify the label for any given image. One third of the set was used as validation set. We tested the model on 768 images. Our model identified the input correctly with over 97.3% accuracy. Each identification took close to 0.06 seconds, and a scheme with multiple photographs of the same RBD image input, say 50 such images will take 3 seconds, which is well within the acceptable limits for real-time deployment within a production line. We also benchmark our results against traditional feature engineering based approaches and show the efficacy of using a feature learning based approach (i.e. CNN in our case).

The paper is structured as follows; in Section 2, we relate our work with current literature. In Section 3, we explain our approach. In particular, we discuss the test architecture in section 3.1 and the data set creation process we followed in section 3.2. In Section 4, we discuss the experiments conducted to validate our approach and our CNN model and its architecture. In Section 5, we conclude the paper with a summary and a discussion on future work.

2 Related Works

Data science has come to the forefront in combating the challenges related to disability and inclusiveness through assistive technologies. A team at MIT in collaboration with

NUS developed self driving wheelchair, and design was improved by researchers from College of Engineering and Computer Science, California State University at Northridge. [1,17]. Navigation assistance is another area that has seen a significant boost in last 5 years because data from multiple sources like GPS, accelerometers, gyroscopes, and cameras is helping us solve this issue. Efforts have also gone in improving RBDs using Data Science like integrating optical character recognition (OCR) into the machine so that it can recognize and display any text [9].

Reliability of the cells is major concern in RBDs. Innovations in the direction of designing reliable RBD cells has a long history going back as far back as 1950s [7]. There have been made many more attempts in the line of design improvement [4,16,26,28,31–33,35].

In the quality assurance of any product a necessary compliment to design is testing. Better, faster, cheaper and effective testing is central to early discovery of faults thus preventing client site failures. This paper focuses on automated testing of RBD cells.

Automation of testing using visual inspection is done in various industries like automobile, lumber, bottling, textile etc. with great results since 1980s [10]. Visual based inspection of PCB is also prevalent [18]. In automobile sector, inspection of parts like brake cylinder, camshaft, cylinder bore etc. are being done using visual inspection [23]. Automated Visual Inspection(AVI) is also used for analysis of radiographic images like X-ray from as early as 70s and 80s [36]. Inspection for the glass and ceramic industry and the inspection for the food and packaging industry is also vision based [10]. With the advent of deep learning, testing based on visual inspections have become ubiquitous as neural networks have become particularly good at feature extraction and pattern recognition. Many industries have recently adopted this type of testing. One of the major areas that benefited from this is transportation sector. Visual inspections are used to find crack and anomaly in bridges, tunnels and railway tracks [15]. AVI has also found a fundamental place in healthcare. It is also used by doctors to complement them while verifying different medical reports. AVI makes it easier to detect diseases like Skin Cancer [13] and Parkinsons Syndrome [24]. AVI is used to automate the testing process of printed control boards. It has been found that use of this technology has proved to be highly efficient [34]. Classification of solder joints have also been done using visual inspection by the help of neural networks [20]. Visual-based automated testing is becoming prevalent now. Testing of Printed Circuits boards (PCB) shares a lot of similarity with testing of an RBD therefore, we have decided to take this approach over other approaches for our case.

In this paper, we propose a visual-based automatic inspection method for RBDs. We systematically compare, the traditional feature engineering versus modern feature learning based methods. After exploring the architecture and the hyperparameter space, we suggest an architecture with an excellent accuracy and acceptable inference time.

3 Our Approach

In this section, we describe the test architecture of our proposed solution for visual-based automatic inspection of RBDs. We describe the dataset curation process towards building the model to predict the bit pattern from the visual input.

3.1 Test Architecture

Figure 1 shows the overall architecture of our test setup. The test setup has three main components. The first component is the RBD that is to be tested. The second is the *visual inspection module* (VIM). The third is the *comparator(X)*.

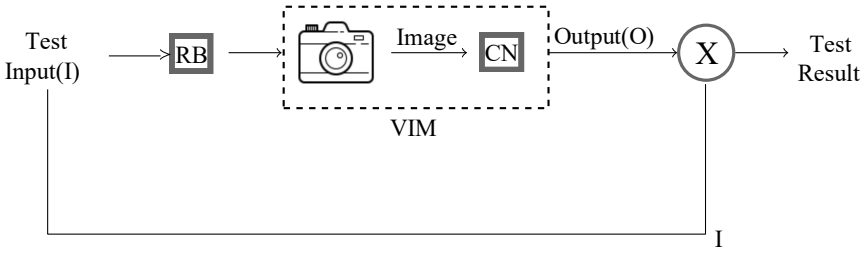


Fig. 1. Test Architecture

At the time of testing, an input (i.e. a byte corresponding to the character to be displayed) is given to the cell. The Braille character displayed by the cell as a result is the actual output of the system. Due to mechanical faults, there is a probability that this output may deviate from the input. It is the purpose of the testing system to find these deviations and notify when and where they happen. The approach is explained below:

1. For each input, one photograph of RBD module(consisting of two cells as shown in figure 2) is taken by the digital camera mounted on the system.
2. The pre-trained CNN unit analyses each image and predicts the output that is displayed by the cell. This is the actual output O.
3. The comparator compares O with I . If they match, the cell is considered to be working as expected.

Our main objective is to design a reliable VIM that runs in real- time. To design this VIM, we created a image dataset first. We trained a CNN model using this dataset. After getting the desired accuracy, we use this model as our multiclass classifier to predict if an RBD is displaying the correct value or not. Our comparator compares this predicted value and original input (ground truth) to test if the RBD is working correctly or not. In sections 3.2 to 3.4 we will explain the complete process of creating the dataset and in section 4 we will explain the our experiments with different CNN architectures.

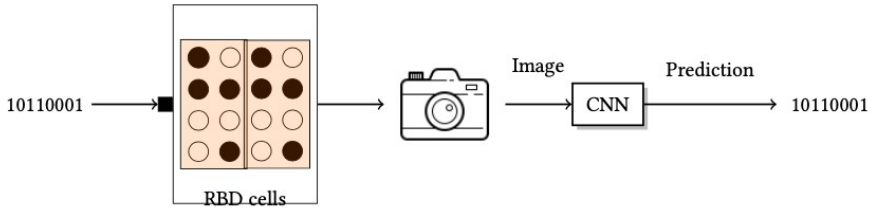


Fig. 2. Prediction by CNN

3.2 Creating The Dataset

A comprehensive dataset is needed in order to train a neural net- work. However, a dataset for RBDs does not exist, which meant that we had to create our own dataset. To create a dataset we need to capture pictures of all the combinations of a braille cell in different lighting conditions. One thing worth noting here is that in our design, two cells combined form one module i.e. each module has 16 pins. Even if one cell of the module malfunctions, we have to replace the whole module. Therefore, both cells are given the same input and tested at the same time. We came up with a comprehensive plan to efficiently capture all the images.

3.3 Capturing images

There are 8 pins on one cell of an RBD which means that total 256 i.e. 2^8 configurations are possible. We need to capture images in various lighting conditions hence we will use 4 different colored LEDs in four top corners of the box. With 4 different LEDs we will get 16 i.e. 2^4 lighting conditions. The design of the data creation enclosure can be seen in Figure 3. 2^4 images for every configuration means that we will get 2^{12} i.e. 4096 images. However, in practice it was found that 4096 training examples were not enough to train a CNN as the parameter space is huge, therefore, we needed to synthesize artificial data too.

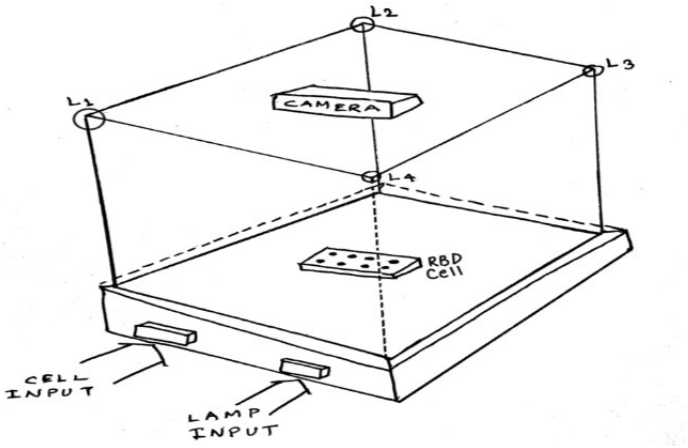


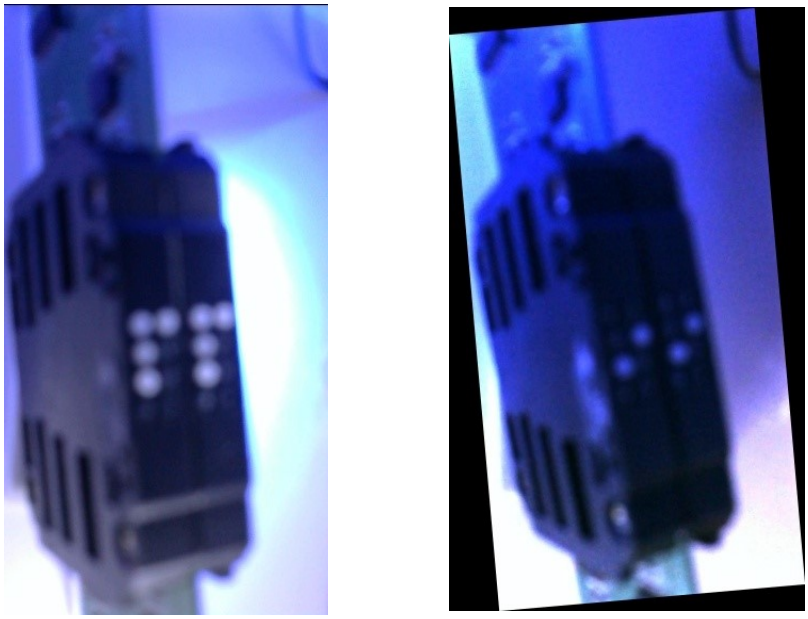
Fig. 3. Data Creation Enclosure

3.4 Data Augmentation

To create additional synthetic data we rotated and changed the colour patterns of all the images. Each image was rotated by 5 and -5 degrees and slight variations were introduced in the colour pattern. These variations ensures that our dataset has diverse range of illumination conditions. Rotation ensures that model will perform well even if images are captured from different angles because it will be an invariant model [14]. After doing this we got 48 images for each configuration bringing total no. of images to 12288. In Figure 4 and we can see a captured image and a synthesised image.

3.5 Data Pre-processing

In order to feed the data to a deep learning model, we need to make it as useful as possible as the real-world data is often incomplete, inconsistent, and lacking in certain behaviours or trends, and is likely to contain many errors [11]. However, in our case, the data was created in a controlled environment, therefore, it did not need extensive pre-processing. However, a series of steps were taken to make it more refined and useful. All the images were converted to gray-scale because when we tried the same experiment with RGB scheme, results were marginally better but significantly more computationally intensive. This also helped as it boosted the prediction time significantly. A Gaussian blur was also applied to make the images a little bit smoother [6]. Each image when captured was of size 640 × 480, it was reduced to size 100 × 100. 100 × 100 image contains enough features needed by a network and it is significantly faster to process than a 640 × 480 image.



(a) Captured image (**b)** Synthesised Image

Fig. 4. Data Creation Enclosure

4 Experiments and Results

Our task is a multiclass classification problem with 100×100 image as the input. The output classes correspond to the 256 possible pin configurations.

4.1 Feature Engineering Vs Feature Learning

We compared feature engineering approach versus feature learning based approach. Our engineered features were extracted as follows: we used Oriented FAST and Rotated BRIEF (ORB) [30] feature detector (which is an alternative to SIFT and SURF as they are patented). ORB uses FAST keypoint detector to determine the key points [29]. Then a Harris corner detector is applied to find top points. After finding top points, BRIEF descriptor is used [8]. ORB can detect both corners and blobs and it is rotation invariant and resistant to noise. Finally, we tried two classical ML multi-class classification models i.e. logistic regression(LR) and decision tree classifier(DT). As regards, feature learning, we used a CNN on the same data, the exact details of the architecture is explained in the subsequent section. Overall, results suggest feature learning is better. Our CNN model performs much better than engineered features along with shallow ML models. Details of the performance can be found in Table 1.

Table 1. Performance: Feature engineering vs Feature learning

Model	Logistic Regression	Decision Tree	CNN
Accuracy(%)	88	72	99.6

4.2 Experiments with CNN architecture

We tried multiple CNN architecture combinations before deciding on the final architecture. Hyper-parameters are the parameters that affect other parameters of the model. In our model, hyperparameters are: number of layers, size of each layer, kernel size of a convolution layer, number of filters in a convolution layer, size of Maxpooling, activation functions, batch size and number of epochs. We experimented with models with one and two convolution layers. Kernel sizes were picked from {3, 5}, number of filters was picked from {32, 64}. We also experimented with ReLu [12] and sigmoid [12] activation functions. However, sigmoid suffers from vanishing gradient problems. So we used ReLu. We found out that model with two convolution layers with Kernel size as 3, number of features as 64 and activation function as ReLu performs better than other models.

We trained our network on different combinations of hyperparameters and see what works best for us. Different hyper-parameters were tested and we monitored the loss function on both Validation set and Training set on Tensorboard [2]. Table 2 shows results from one such experiment. Model 1 is the chosen model, Model 2 had only 32 filters in convolution layer, Model 3 had only one convolution layer and Model 4 has 32 filters with kernel size of (5 × 5). Model 1 performed the best.

Table 2. Comparison of experimented models

Model	Train Accuracy	Validation Accuracy
Model 1	99.6%	99%
Model 2	98.1%	97.2%
Model 3	96.7%	95.4%
Model 4	98.8%	98.1%

Table 3. Summary of the model

Layer Type	Output Shape	Kernel Size
Convolution	98 × 98 × 64	3 × 3
ReLu	98 × 98 × 64	-
Max Pooling	49 × 49 × 64	3 × 3
Convolution	47 × 47 × 64	3 × 3
ReLu	47 × 47 × 64	-
Max Pooling	23 × 23 × 64	3 × 3

Flattened	33856	-
Dense	64	-
ReLu	64	-
Dense	256	-
Softmax	256	-

In our final model (Model 1) the first and second layers are convolution layers with ReLu activation function and Maxpooling [12]. Third and fourth layers are fully connected, after flattening. We have used Adam [21] as our optimiser with batch size of 32. Figure 5 shows architecture of the model. After trying out many combinations, we found the best balance between speed of prediction and accuracy with these hyper-parameters:

- No. of layers=4
- Kernel size of a convolution layer=(3,3)
- No. of filters in a convolution layer=64
- Size of Maxpool=(2,2)
- Activation functions= ReLu, ReLu, ReLu, Softmax
- Batch size= 32
- No. of epochs=10

After training the final model on training set of 8068, cross validation set of 3258 and test set of 768 images we got the training set accuracy: 99.6%, validation set accuracy: 99% and test set accuracy: 97.3%. This shows that our algorithm has both low bias and low variance.

There are two important factors that we need to consider while prediction i.e. confidence in the prediction and time elapsed while prediction. While we have over 97% accuracy, we would still like more confidence and for that we can take multiple samples and test all of them all. We have to make sure that process is as quick as possible because this will help us test a cell at the assembly line itself. Our model takes 2.8 seconds to predict 50 images and compare if it matches with the test input or not. This does not take into account the time taken by the camera to capture 50 images as image capturing task is easily parallelizable.

5 Conclusion and Future Work

Affordability is at the heart of accessibility. There are many examples reported in literature where data science and computing are used in the design of accessibility features [1,3,17]. To the best of our knowledge, there is not enough reported work reporting how data science and computing can significantly bring down the manufacturing cost, thus adding affordability to accessibility. In this paper, we have presented a method for automated testing of RBD cells based on deep learning that has played an important role in bringing down the manufacturing cost of an RBD. In our experiments, our method performed with an accuracy of 97.3%. We believe that our model can be used

on any type of Refreshable Braille Display. Also, note that in the production environment parameters like illumination and camera position/angle can be much more closely controlled. This makes it likely that the figures obtained in our experiments are more conservative in terms of accuracy and speed than those of a production environment.

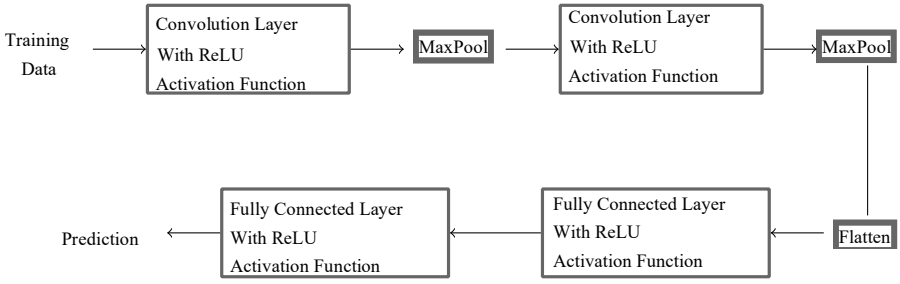


Fig. 5. Model Architecture

Currently, our method tests one module at a time. We can extend the method to test a complete RBD consisting of 7 such modules at one time, which will significantly enhance the testing speed. There are 2 different approaches to solve this problem: Semantic Segmentation and Sliding window Method. State of the art semantic segmentation models like Mask-RNN, Faster R-CNN [27] are too slow (5-8 FPS) for our use case as we need to test the modules on a moving conveyor belt. Sliding window method does not work well without borders around each module. Therefore, we need to slightly change the design of the module itself by putting contrasting color borders. To test the entire RBD display at a time, we also have to create a dataset to train our neural network for testing multiple cells at a time. However, this means that the number of input combinations will go to 2112 which well beyond feasible range. Test generation techniques like T-wise coverage [19] can be used to bring down the input space within feasible range.

A. Dataset. The entire dataset has been released in public domain and can be found at https://www.kaggle.com/shivam3376/refreshable-braille- display- cell

B. Code. We are also releasing the code to train and delpoy the model. It can be found at https://github.com/shivamkumarsingh114/Automated- testing-of-RBDs.git

Acknowledgement. This work was supported by Karnataka Innovation Technology Society, Dept. of IT, BT and ST, Govt. of Karnataka

References

1. Smart fm trials self-driving wheelchair (May 2017), https://smart.mit.edu/newsevents/smart-fm-trials-self-driving-wheelchair

2. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). pp. 265–283 (2016)
3. Barbosa, J., Tavares, J., Cardoso, I., Alves, B., Martini, B.: Trailcare: An indoor and outdoor context-aware system to assist wheelchair users. International Journal of Human-Computer Studies **116**, 1–14 (2018)
4. Blazie, D.: Refreshable braille now and in the years ahead. Braille Monitor **43**(1), 1–6 (2000)
5. Bourne, R.R., Flaxman, S.R., Braithwaite, T., Cicinelli, M.V., Das, A., Jonas, J.B., Keeffe, J., Kempen, J.H., Leasher, J., Limburg, H., et al.: Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis. The Lancet Global Health **5**(9), e888–e897 (2017)
6. Bradski, G.: The opencv library. Dr Dobb's J. Software Tools **25**, 120–125 (2000)
7. Bryce, J.W., Wheeler, J.N.: Reading apparatus (Sep 5 1950), uS Patent 2,521,338
8. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: Binary robust independent elementary features. In: European conference on computer vision. pp. 778–792. Springer (2010)
9. Chakraborty, P., Mallik, A.: An open source tesseract based tool for extracting text from images with application in braille translation for the visually impaired. International Journal of Computer Applications **68**(16) (2013)
10. Chin, R.T., Harlow, C.A.: Automated visual inspection: A survey. IEEE transactions on pattern analysis and machine intelligence (6), 557–573 (1982)
11. Dharmarajan, R., Vijayasanthi, R.: An overview on data preprocess- ing methods in data mining. nternational Journal For Scientific Research and Development **3**(3), 3544–3546 (Jun 2015)
12. Ertam, F., Aydın, G.: Data classification with deep learning using tensorflow. In: 2017 International Conference on Computer Science and Engineering (UBMK). pp. 755–758. IEEE (2017)
13. Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature **542**(7639), 115–118 (2017)
14. Gandhi, A.: Data augmentation: How to use deep learning when you have limited data (Aug 2019), https://nanonets.com/blog/data-augmentation-how-to-usedeep-learning-when-you-have-limited-data-part-2/
15. Gibert, X., Patel, V.M., Chellappa, R.: Deep multitask learning for railway track inspection. IEEE transactions on intelligent transportation systems **18**(1), 153–164 (2016)
16. Grunwald, A.P.: Reading and writing machine using raised patterns (Nov 30 1971), uS Patent 3,624,772
17. Hartman, A., Nandikolla, V.K.: Human-machine interface for a smart wheelchair. Journal of Robotics **2019** (2019)
18. Jarvis, J.F.: A method for automating the visual inspection of printed wiring boards. IEEE Transactions on pattern analysis and machine intelligence (1), 77–82 (1980)
19. Jorgensen, P.C.: Software testing: a craftsman's approach. Auerbach Publications (2013)
20. Kim, T.H., Cho, T.H., Moon, Y.S., Park, S.H.: Visual inspection system for the classification of solder joints. Pattern Recognition **32**(4), 565–575 (1999)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
22. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)

23. McKeown, P., Cooke, P., Bailey, W.: The application of optics to the quality control of automotive components. In: Solving Quality Control and Reliability Problems with Optics. vol. 60, pp. 77–84. International Society for Optics and Photonics (1975)
24. Ortiz, A., Mart´ınez-Murcia, F.J., Garc´ıa-Tarifa, M.J., Lozano, F., Go´rriz, J.M., Ram´ırez, J.: Automated diagnosis of parkinsonian syndromes by deep sparse filtering-based features. In: International Conference on Innovation in Medicine and Healthcare. pp. 249–258. Springer (2016)
25. Pascolini, D., Mariotti, S.P.: Global estimates of visual impairment: 2010. British Journal of Ophthalmology **96**(5), 614–618 (2012)
26. Ren, K., Liu, S., Lin, M., Wang, Y., Zhang, Q.: A compact electroactive polymer actuator suitable for refreshable braille display. Sensors and Actuators A: Physical **143**(2), 335–342 (2008)
27. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
28. Roberts, J., Slattery, O., Kardos, D.: 49.2: Rotating-wheel braille display for continuous refreshable braille. In: SID Symposium Digest of Technical Papers. vol. 31, pp. 1130–1133. Wiley Online Library (2000)
29. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: European conference on computer vision. pp. 430–443. Springer (2006)
30. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: 2011 International conference on computer vision. pp. 2564–2571. Ieee (2011)
31. Runyan, N., Blazie, D.: Eap actuators aid the quest for the'holy braille'of tactile displays. In: Electroactive Polymer Actuators and Devices (EAPAD) 2010. vol. 7642, p. 764207. International Society for Optics and Photonics (2010)
32. Runyan, N., Nassimbene, E.: Alphanumeric'displays' for the blind: A technology search. Tech. rep., IBM Technical Report, IBM Corp., Los Gatos, CA (1974)
33. Tretiakoff, O., Tretiakoff, A.: Electromechanical transducer for relief display panel (Aug 23 1977), uS Patent 4,044,350
34. Vitoriano, P.M., Amaral, T.G., Dias, O.P.: Automatic optical inspection for surface mounting devices with ipc-a-610d compliance. In: 2011 International Conference on Power Engineering, Energy and Electrical Drives. pp. 1–7. IEEE (2011)
35. Wallace, G.G., Teasdale, P.R., Spinks, G.M., Kane-Maguire, L.A.: Conductive electroactive polymers: intelligent materials systems. CRC press (2002)
36. Wright, L.: Results of x-ray television inspection of electronic parts. (1967).