Real-Time Braille Translation via FOTS and Feature-Based Decoding

Prithvi Pothupogu
Dept. of Electronics and Communication
NIT Warangal

Rohan Kiran Gunjal
Dept. of Electronics and Communication
NIT Warangal

Shivam Sethia
Dept. of Electronics and Communication
NIT Warangal

Levin Joji Mathews
Dept. of Electronics and Communication
NIT Warangal

***Abstract*—Braille translation from natural scenes is a complex task, primarily due to the lack of adequately labeled datasets. Most existing studies focus on specific aspects, such as segmenting Braille or detecting individual characters, rather than addressing the entire translation process. To address this gap, we propose a unified model capable of translating Braille from natural scenes, paired with a custom-built dataset that reflects real-world scenarios.**

Traditional methods employ a two-stage process involving bounding box-based detection followed by text decoding. However, these methods face difficulties with challenges like poor lighting, distortions, and obstructions. To overcome these limitations, our approach adapts FOTS for both segmentation and decoding, benefiting from its high accuracy and real-time detection capability to handle complex environments effectively. To further enhance the accuracy of Braille decoding, we will incorporate the Circular Hough Transform for identifying the circular dot patterns characteristic of Braille. A diverse dataset will be created to support this model, combining images from online sources, Google Street View, and manually crafted Braille samples, ensuring comprehensive coverage to improve its ability to generalize across various conditions.

***Index Terms*—Braille translation, Custom Dataset, Circular Hough Transform, text decoding, segmentation, FOTS**

I. Introduction

Braille is a crucial tactile writing system for the visually impaired, but translating it in natural scenes—like on signs and product labels—remains a challenge due to varying surfaces and environmental factors.

Traditional systems rely on tactile perception, limiting usability for those with partial vision loss. Automated Braille translation faces issues like noise, lighting fluctuations, and occlusions, and existing methods typically segment and decode Braille separately. However, they lack a unified framework for handling real-world variability effectively.

This paper proposes a model for Natural-Scene Braille Translation, using YOLO for accurate, real-time segmentation and the Circular Hough Transform to decode Braille dots. This approach aims to improve speed, accuracy, and accessibility for visually impaired users.

To support the model, a custom dataset has been curated, incorporating Braille from online repositories, custom samples, and Google Street View images. Unlike prior works, which have limitations in handling real-world variability and real-time performance, our model integrates YOLO and the Circular Hough Transform to address these challenges, enhancing Braille detection in diverse environments.

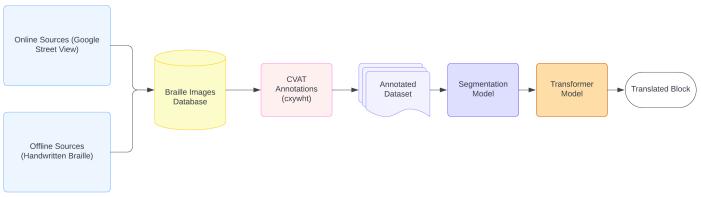

Fig. 1. Workflow

II. Implementation

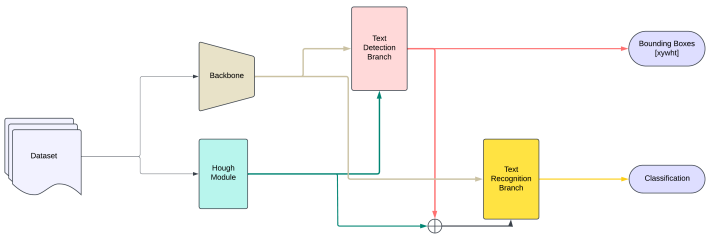

Fig. 2. Braille Spotting Model

A. Braille Image Collection

The process of data preparation involves transcribing Braille using a Braille slate from text extracts sourced from a variety of materials. Key data sources include:

- **Text Sources:** Images were collected from newspaper columns and Braille textbooks, providing structured examples of printed Braille.
- **Online Sources:** Live-scene images with Braille signs were sourced from Google Street View, showcasing real-world conditions like variable lighting and orientations.
- **Total Images:** A total of 120 images were gathered, with 60 images from each source.
- **Total Annotated Characters:** Approximately 7,000 Braille characters were annotated to create a robust dataset for training and evaluation.

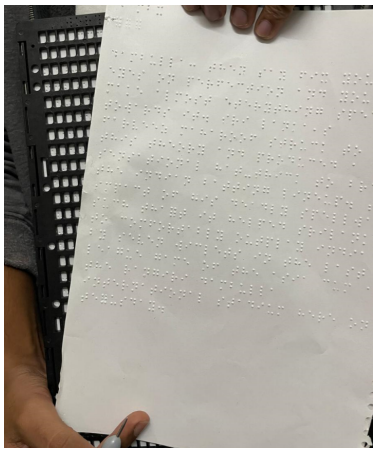

Fig. 3. Custom Dataset Creation

B. Annotations

- Images were annotated using CVAT (Computer Vision Annotation Tool) to ensure accurate and consistent labeling of Braille characters.
- Annotations followed a standardized Braille convention, as illustrated in the attached image.

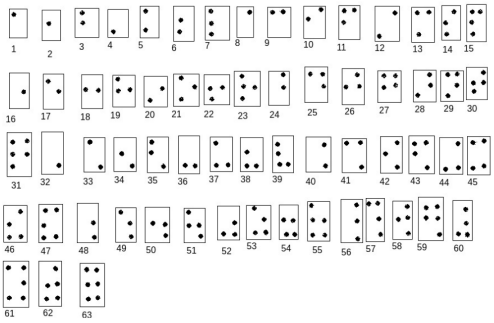

Fig. 4. Braille Conversion

C. Dataset Generation

- **Anchor Boxes:** Anchor boxes for Braille characters were generated using k-means clustering, optimizing their parameters for [aspect-ratio, angle]. This ensures that the anchor boxes align well with the shapes and orientations of Braille characters in the images.
- **Image Splitting:** To standardize input for the model, images were split into fixed sizes of 256 pixels x 256 pixels, making them compatible with the model architecture while retaining sufficient detail for character detection.
- **Bounding Box Format:** Each bounding box is defined in the format [class_name, x_center, y_center, width, height, theta]

D. Segmentation Model

- The model uses Darknet-53 (YOLOv3) as its backbone. The backbone provides a good number of convolution layers for the model to extract features.
- All convolutional blocks consist of convolutional layers, followed by batch normalization and finally SiLU activation layer.
- The Hough Transform Module creates a mask to learn the regions with circular silhouette which is particularly useful for the model to learn and extract features from. The Transformed image is then followed by convolutional layers to extract features.

Fig. 5. Hough Transform

- The output of the Hough Module and the backbone is summed and passed to the Braille Detection Module. The Braille Detection Module further uses sequential convolutions to predict the bounding boxes.
- The output of the bounding boxes form a mask which is summed with the hough module and then passed to the Braille Recognition Model. It uses FCNNs to then predict the classes of all bboxes present.
- Braille detection module uses Dice Loss and the classification / recognition module uses multi-cross entropy loss. The overall loss is defined as:

$$L_{\text{dice}} = 1 - \frac{2 \cdot p_{\text{pred}} \cdot y + \delta}{p_{\text{pred}}^2 + y^2 + \delta}$$

$$L_{\text{cls}} = -\sum_{i=1}^{N} \log(p_i)$$

$$L_{\text{overall}} = L_{\text{cls}} + \lambda \cdot L_{\text{dice}}$$

here, we used λ as 1 as suggested by the FOTS algorithm.

III. Implemented Work

Significant strides have been made across various components of the project. To begin with, the architecture for the Braille spotting model has been finalized.
The model is designed with a YOLO backbone for feature extraction and includes advanced techniques such as the

Hough Transform and RoI rotation to enhance the accuracy of Braille detection and recognition. On the data collection side, we have successfully transcribed nine pages of Braille text using a Braille slate. Additionally, around 40 pages of Braille text have been annotated with the help of the Computer Vision Annotation Tool (CVAT), resulting in the annotation of approximately 7,000 Braille characters. This process has been completed, ensuring the availability of a high-quality dataset for training and evaluating the model. Moreover, a set of 60 Braille signage images sourced from Google Street View, representing real-world environments, has been incorporated into the dataset to further strengthen its diversity.

The dataset has been tested with several leading object detection models, such as Faster R-CNN, YOLOv5, and YOLOv11n, to evaluate their performance in detecting Braille.

IV. Results

A. Braille Spotting Model

Braille Spotting Model outperforms Faster R-CNN, YOLOv5, and YOLOv11 by combining YOLO for fast, real-time segmentation with the Circular Hough Transform for accurate Braille dot recognition. This unified approach ensures high accuracy even in challenging conditions like occlusions and lighting variations. While Faster R-CNN is slower and YOLO models trade accuracy for speed, our model strikes a better balance, providing robust performance in real-time applications. Additionally, our custom dataset improves generalization across diverse Braille formats, making it more adaptable to real-world scenarios.

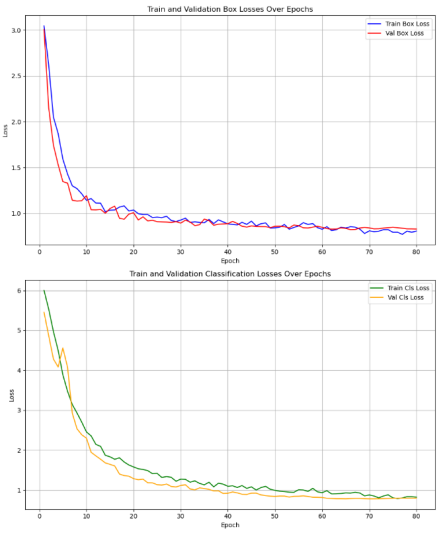

Fig. 6. Loss Graph for Braille Spotting Model

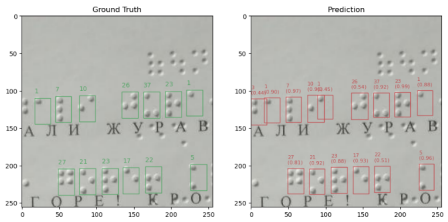

Fig. 7. Prediction Results for Braille Spotting Model

B. Faster R-CNN

Faster R-CNN is a two-stage object detection model that utilizes a Region Proposal Network (RPN) to generate region proposals, followed by a convolutional neural network for classification and bounding box regression. This model is known for its high accuracy, especially when detecting small or complex objects in cluttered or noisy environments. However, it is computationally intensive and slower than other models like YOLO. In your project, Faster R-CNN was applied to detect Braille text within complex, crowded scenes, where precise identification of small or overlapping Braille characters was essential.

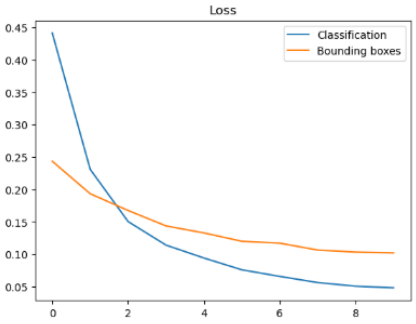

Fig. 8. Classification and Bounding Box Loss for Faster R-CNN

Fig. 9. Additional Results for Faster R-CNN

C. YOLOv5

YOLOv5 is a real-time object detection model designed to offer both speed and efficiency. It processes an entire image in a single pass, predicting both the bounding boxes and class labels for detected objects. The model provides a great balance between speed and accuracy, making it ideal for dynamic, real-world environments where rapid detection is crucial. YOLOv5 was used in your dataset to quickly identify Braille text in challenging conditions such as varying lighting and partial obstructions, ensuring real-time performance without sacrificing too much accuracy.

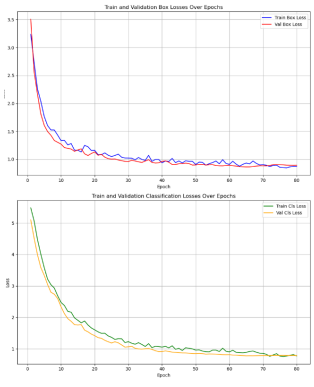

Fig. 10. Loss Graph for YOLOv5

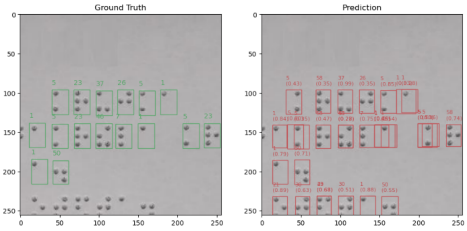

Fig. 11. Prediction Results for YOLOv5

D. YOLOv11

YOLOv11n is a lightweight variant of the popular YOLO model, optimized for environments with limited computational resources. While it offers faster inference times, it may sacrifice some accuracy compared to YOLOv5. This model is particularly beneficial for use in applications running on devices with limited processing power, such as mobile phones or edge devices. YOLOv11n was applied in your project to perform rapid Braille detection in situations where resources were constrained, making it suitable for deployment in real-time applications on smaller or less powerful devices.

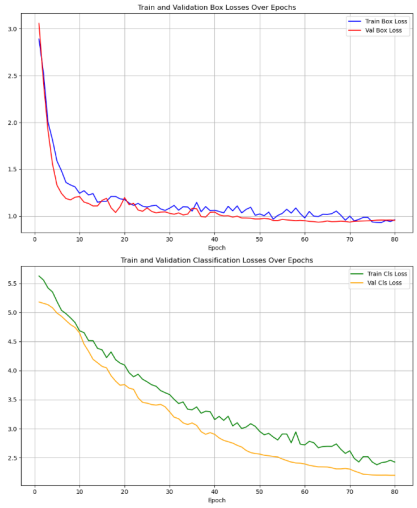

Fig. 12. Loss Graph for RCNN

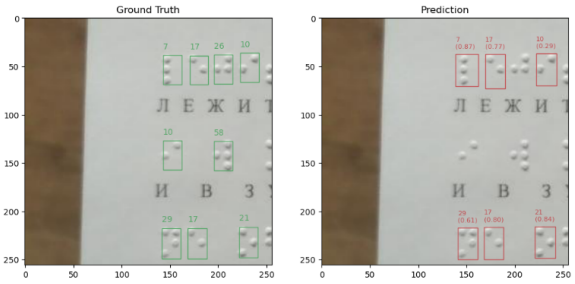

Fig. 13. Prediction Results for YOLOv11

V. Comparative Analysis

This section provides a detailed comparison of performance metrics for various Braille detection models, including Faster R-CNN, YOLOv5, YOLOv11, and our proposed Braille Spotting model. The evaluation encompasses key metrics such as classification loss (Cls Loss), bounding box loss (BBox Loss), and accuracy (mAP@0.5). The results demonstrate the strengths and weaknesses of each model in terms of efficiency and accuracy in detecting Braille characters. All experiments were conducted under consistent conditions to ensure fairness and reproducibility.

Model	Accuracy	Cs Loss	Bbox Loss
Faster R-CNN	79.56%	0.0481	0.1020
YOLOv11n	84.74%	0.804	0.92
YOLOv5	85.51%	0.432	0.632
Braille Spotting Model	92.82%	0.699	0.830

TABLE I
Performation Matrix of Braille Detection Models

VI. Conclusion

This work presents a model for automatic Braille translation from natural scenes, overcoming challenges like lighting

variations, skewed orientations, and partial occlusions. By using YOLO for text segmentation and Circular Hough Transform for dot recognition, the model delivers accurate and fast real-time translation. A custom dataset with diverse Braille samples improves adaptability.

Integrating YOLO with the FOTS algorithm enables seamless Braille detection, offering a practical solution for visually impaired individuals in dynamic environments. Future work will focus on refining the model to handle complex formats, enhance robustness, and optimize for mobile devices, advancing accessibility for those with vision impairments.

REFERENCES

[1] A. Al-Salman and A. AlSalman, "Fly-LeNet: A deep learning-based framework for converting multilingual braille images," *Heliyon*, vol. 10, no. 4, pp. e26155, 2024. doi: https://doi.org/10.1016/j.heliyon.2024.e26155.

[2] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan, "FOTS: Fast Oriented Text Spotting with a Unified Network," *arXiv preprint arXiv:1801.01671*, 2018. doi: https://doi.org/10.48550/arXiv.1801.01671.

[3] Z. Khanam and A. Usmani, "Optical Braille Recognition using Circular Hough Transform," *arXiv preprint arXiv:2107.00993*, 2021. doi: https://doi.org/10.48550/arXiv.2107.00993.

[4] M. Buta, L. Neumann, and J. Matas, "FASText: Efficient Unconstrained Scene Text Detector," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1206-1214. doi: 10.1109/ICCV.2015.143.

[5] A. Antonacopoulos and D. Bridson, "A Robust Braille Recognition System," in *Document Analysis Systems VI, 6th International Workshop, DAS 2004, Florence, Italy, September 8-10, 2004, Proceedings*, 2004. doi: 10.1007/978-3-540-28640-0_50.

[6] T. Li, X. Zeng, and S. Xu, "A deep learning method for braille recognition," in *Computational Intelligence and Communication Networks (CICN), 2014 International Conference on*, IEEE, pp. 1092–1097, 2014.