Pocket Compiler: A Versatile Benchmark for Compiling the Programming Language

Raj Vikram[1],Aman Arora[2],Pragyanshu Soni[3]

Under Guidance of Dr.Ranjit Singh
Lovely Professional University, Phagwara, Punjab

Abstract

The world is changing day by day and Technologies are advancing on machine learning. Creativity and problem solving are the most important role play in the present and future. Pocket Compiler takes one step forward to take the creativity of students. Introducing a new code scanner filter that will enable students to solve coding problems simply by clicking a picture using Pocket Compiler. The new filter is part of the search giant's 'back to school' initiative that aims to help students learn at home amid the coronavirus pandemics. To use the feature, students would have to take a snapshot of the code and highlight the code. Following this, they would get the solution of the code in real-time.

Keywords - *OCR, Python, OpenCV,Computer Vision, Image Processing, Android App.*

1. Introduction

Mobile Phones and Computers have seen a huge drop in size as screen technology improves, batteries enhance, and chips become smaller and more efficient. Pocket Compiler tool also looks forward to creating a smaller compiler for mobile and laptop. Pocket Compiler is based on text detection.
It based on segmentation-based approaches have become mainstream because such method is more capable of describing text instance of irregular shape
In summary, our contribution is that we propose a model which has good properties of text recognition to compiling different languages.

2. Related Work

I. Image Capture:

Image Capture on basic level gadgets, similar to those that run Android, is tricky since it must work on a wide variety of devices, many of which are more resource-constrained than flagship phones. Using cameras, we implemented two capture strategies to balance capture latency against performance impact.

II. Text Recognition

After Pocket Compiler catches an Image, it needs to sort out the shapes and letters that establish the words, sentences, and sections. To do this, the picture is downsized and moved to the server, where the preparation will be performed. Next, OCR is applied, which uses a region proposed network to identify character level bouncing boxes that can be converted into lines for text acknowledgment.

Combining these character confines to words is a two-venture, consecutive cycle. The initial step is to apply the Hough Transform, which accepts the content is circulated across equal lines. The subsequent advance uses text stream, which rather follows text that may follow a bend by tracking down the briefest way through a chart of identified content boxes. This guarantees that text with an assortment of appropriations, be they straight, banded, or blended, can be distinguished and handled.

Because the images captured by the pocket compiler may include sources such as signage, handwriting, or documents, a slew of additional challenges can arise.

All of these steps, from script detection and direction identification to text recognition, are performed by separable CNN with an additional quantized LSTM network.

3. METHODS TO DETECT CODES

I.Reading code using mobile camera:

In this method simply have to scan the code and the scanner will automatically detect the system, Extention has been used and it will give the output.

II.Recognition of highly distorted and resolution images:

The algorithm used in this approach gives fast and accurate results even when the quality of the image is poor. It is most suitable for small portable devices.

III.Code recognition system:

This API provides code recognition service, by identifying codes, sharing the information in them with the system, and providing service based on the information. The service can be integrated into a wide range of apps. This API can be used in multiple scenarios, including Wi-Fi and SMS messaging.

IV.Code reader health applications on android mobiles:

Code is scanned using the code scanner available in modern mobiles and the image is communicated to the server, which in turn communicates back the details of the product whose code was scanned.

4. SCHARR GRADIENT

One of the major steps in code Detection is detecting edges. They are being detected using a magnitude representation factor called Scharr Gradient. Scharr Gradient or Sobel Operator is prevalent in Computer Vision and Image Processing. This operator has its uses in those images which focus on Edge Detection [7]. The image intensities are being calculated through functions for an approximate value via this operator. We calculate Scharr Gradient in both x and y-direction. We do so using a parameter that we call ―size‖. We initialize it to negative of one for building the gradient factor for both Vertical and Horizontal Directions in Image.

5. BLURRING OF IMAGE

*After detecting the code portion of the image, we use a blurring effect to decrease the noise and therefore can focus on the code. For this, a 9*9 kernel requires a minimum blurring which reduces the high-frequency irregularities in the picture. A cutoff is then established in the fused picture, with white pixels of 255 pixels above 225, and black ones of less than 225. There are gaps between their vertical bars for your Unique code pic. The following steps can be introduced to close these gaps and simplify the detection by our algorithm of the bar code.*

6. TRANSFORMATION OF IMAGE MORPHOLOGICAL

This procedure depends on the composition of a code to determine the feature map. Only pictures that are com-positional can use it. We have two inputs to organize a methodology on the picture: the specified picture and the kernel. Two basic governors are photogenically active for the image analysis:

Erosion: Erosion:

This operator essentially removes the limits from the object, as does the idea of soil erosion. It affects every pixel on the border that is removed for noise to be reduced. This also decreases the image's width, hence reducing the image's size.

```
import cv2
import numpy as np

img = cv2.imread('input.png', 0

img_dilation = cv2.dilate(img, kernel,
```

```
iterations=1)

cv2.imshow('Input', img)
cv2.imshow('Erosion', img_erosion)
cv2.imshow('Dilation', img_dilation)

cv2.waitKey(0)
```

Fig. 1. Erosion Code Snippet

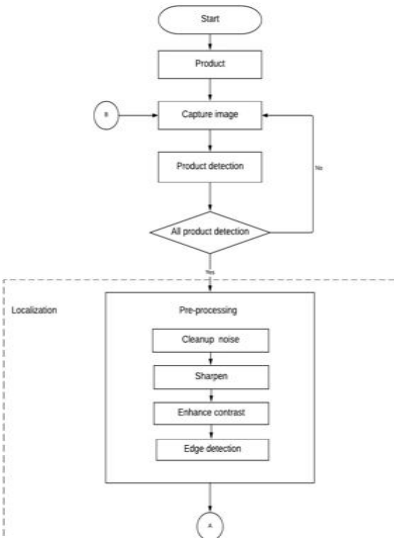
Dilation is the factor. This component has a specific structure than erosion. This factor enhances the oscillation of the oscillations area. Dilation increases the size of the object region to accommodate for the decreased noise from the excavation work.

```
I=imread('lenna.png');
I=im2bw(I);
se=ones(5, 5);
[P, Q]=size(se);
In=zeros(size(I, 1), size(I, 2));

for i=ceil(P/2):size(I, 1)-floor(P/2)
      for j=ceil(Q/2):size(I, 2)-floor(Q/2)

        on=I(i-floor(P/2):i+floor(P/2),
j-floor(Q/2):j+floor(Q/2));

        nh=on(logical(se));
        In(i, j)=max(nh(:));
     end
end

imshow(In);
```

Fig. 2. Dilation Code Snippet

Opening:opening*: We commonly adhere to the excavation work and then to the method of dilation. The word 'opening to' is used for that.*

Closing: Closing: *It's just the other way around the opening. It can be used to close down the giant holes in every instrument for which noise or other obstacles are caused. Detection of Unique code.*



7. ***Conclusion***

The mathematics and code scanning ought to do that, When we completed composing this paper, we had an extensive understanding of code scanning, their storage, the use of and the effective use of methods in them.

Reference -

[1] H. J. G. Kinjal H. Pandya, "A Survey on QR Codes: in context of Research and Application," International Journal of Emerging Technology and Advanced Engineering, vol. 4, no. 3, pp. 258-262, 2014.
[2] K.-T. W. M.-C. W. N.-Y. C. J.-H. W. Chun-Shun Tseng, "Retrospective Tracking for Barcode Reading," IEEE, pp. 114-119, 2010.
[3] N. M. Z. Hashim, N. A. Ibrahim, N. M. Saad, F. Sakaguchi, and Z. Zakaria, ―Unique code recognition system,‖ International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), vol. 2, issue 4, pp. 278-283, 2013.
[4] Gallo, O. and Manduchi, R. (2011). Reading 1D barcodes with mobile phones using deformable templates. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(9):1834–1843.
[5] Katona, M. and Nyul, L. G. (2012). A novel method for ´ accurate and efficient barcode detection with morphological operations. In Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on, pages 307–314.
[6] 1. Palmer, R.C.: The Bar Code Book: Reading, Printing, and Specification of Bar Code Symbols. Helmers Pub (1995)
[7] Bodn´ar, P., Ny´ul, L.G.: Improving barcode detection with combination of simple detectors. In: The 8th International Conference on Signal Image Technology (SITIS 2012). (2012) 300–306
[8] Wachenfeld, S., Terlunen, S., Jiang, X.: Robust recognition of 1-d barcodes using camera phones. In: 19th International Conference on Pattern Recognition (ICPR 2008). (2008) 1–4
[9] O. Parvu and A. G. Bˆ a˘lan, "A method for fast detection and decoding of specific 2D barcodes," in 17th Telecommunications Forum, TELFOR 2009, 2009, pp. 1137–1140.
[10] Y. Xue, G. Tian, R. Li, and H. Jiang, "A new object search and recognition method based on artificial object mark in complex indoor environment," in Intelligent Control and Automation (WCICA), 2010 8th World Congress on, july 2010, pp. 6648 –6653.
[11] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.
[12] Y. Ebrahim, W. Abdelsalam, M. Ahmed, and S.-C. Chau, "Proposing a hybrid tag-camera-based identification and navigation aid for the visually impaired," in Second IEEE Consumer Communications and Networking Conference (CCNC), 2005, pp. 172 – 177.
[13] J. Phaniteja and P. Derin ―Evolution of Unique code‖ International Journal for Development of Computer Science & Technology, vol. 1, no. 2, pp 18, 2013.

[14] M. Canadi, W. Hoepken, and M. Fuchs, ―Application of QR codes in online travel distribution,‖ Information and Communication Technologies in Tourism, vol. 4, pp. 137–148, Springer, New York 2010.
[15] D. Kuo, D. Wong, J. Gao, and L. Chang, ―A 2D Unique code Validation System for Mobile Commerce,‖ International Conference on Grid and Pervasive Computing, Advances in Grid and Pervasive Computing, vol. 6104, pp. 150–161. Springer, Heidelberg, 2010.
[16] Karan Aggarwal, Mohammad Salameh, and Abram Hindle. Using machine translation forconverting python 2 to python 3 code. Technical report, PeerJ PrePrints, 2015.
[17] Miltiadis Allamanis, Earl T Barr, Christian Bird, and Charles Sutton. Learning natural coding conventions. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, pages 281–293, 2014.
[18] Uri Alon, Shaked Brody, Omer Levy, and Eran Yahav. code2seq: Generating sequences from structured representations of code. ICLR, 2019.
[19] Uri Alon, Roy Sadaka, Omer Levy, and Eran Yahav. Structural language models for any-code generation. arXiv preprint arXiv:1910.00577, 2019.
[20] Matthew Amodio, Swarat Chaudhuri, and Thomas Reps. Neural attribute machines for program generation. arXiv preprint arXiv:1705.09231, 2017.
[21] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Learning bilingual word embeddings with (almost) no bilingual data. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 451–462, 2017.
[22] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Unsupervised statistical machine translation. arXiv preprint arXiv:1809.01272, 2018.
[23] Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. In International Conference on Learning Representations (ICLR), 2018.
[24] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.
[25] Antonio Valerio Miceli Barone and Rico Sennrich. A parallel corpus of python functions and documentation strings for automated code documentation and code generation. arXiv preprint arXiv:1707.02275, 2017.
[26] Avishkar Bhoopchand, Tim Rocktäschel, Earl Barr, and Sebastian Riedel. Learning python code suggestion with a sparse pointer network. arXiv preprint arXiv:1611.08307, 2016.
[27] Xinyun Chen, Chang Liu, and Dawn Song. Tree-to-tree neural networks for program translation. In Advances in neural information processing systems, pages 2547–2557, 2018.
[28] Zimin Chen, Steve James Kommrusch, Michele Tufano, Louis-Noël Pouchet, Denys Poshyvanyk, and Martin Monperrus. Sequencer: Sequence-to-sequence learning for end-to-end program repair. IEEE Transactions on Software Engineering, 2019.
[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805, 2018.
[30] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. Codebert: A pre-trained model for programming and natural languages. arXiv preprint arXiv:2002.08155, 2020.
[31] Cheng Fu, Huili Chen, Haolan Liu, Xinyun Chen, Yuandong Tian, Farinaz Koushanfar, and Jishen Zhao. Coda: An end-to-end neural program decompiler. In Advances in Neural information Processing Systems, pages 3703–3714, 2019.