BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2023
http://excel.fit.vutbr.cz

Keyboard and Keys Image Recognition

Jan Lorenc*

Abstract

The goal of this thesis is to create a solution for keyboard keys recognition to automate robotic writing on keyboards. Datasets for keyboard detection in an image, character detection in an image and post-processing correction of the character detection based on various keyboard layouts were created as prerequisits for this work. This research presents several approaches towards keyboard keys detection problem and selects the most suitable one. The chosen strategy is to split the problem into 3 phases which correspond to the prepared datasets. Firstly, a separate keyboard detection is run. Secondly, characters are recognized in the detected keyboard region. These tasks are accomplished using neural networks and Canny edge detection technique. The last phase is the post-processing of the detection results (character correction, autocompletion of undetected characters, special keys distinction etc.). The results of each phase are evaluated. The contribution of the thesis lies in the creation of the datasets for keyboard and keys detection, and novel modular and extensible solution for the recognition process that yields very promising results.

*xloren15@vut.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

The idea for this thesis comes from the company Y Soft which develops a robotic automation solution AIVA. The ultimate goal is to teach the robot to autonomously write on keyboards. To accomplish such a task, tools for keyboard and keys detection are required. As no other open-source project seems to exist, a custom solution has to be developed.

There are two tasks to be solved. The first one is keyboard recognition in an image. This enables the robot to decide if it can write and where. The second task is single-character detection where characters represent the keys. The objective is to recognize alphanumerical keys and some special keys. In addition, special character detection is attempted. For both tasks, training datasets had to be created.

On this topic, only one other solution could be found. Researchers from Amazon also tried to solve automated keyboard typing for their framework [1]. Their article [2] has been an inspiration, especially in the solution design. Unfortunately, neither the code nor the dataset seems to be publicly available.

The proposed solution is to split the image recognition process into 3 phases. Firstly, a keyboard region is detected using YOLOv7 [3] neural network, the current SOTA in object detection. Secondly, characters are recognized in the detected keyboard region using YOLOv7 and Canny edge detection technique. Lastly, the character detection results are processed to make corrections.

The main contribution of the work is the improvement of the production solution Y Soft AIVA. It greatly simplifies the current keyboard typing process and reduces the time for automation script definitions. Furthermore, it provides datasets for keyboard and character object detection to the public. Moreover, it is to the best of my knowledge the only open-source solution of its kind.

2. Datasets creation

No available keyboard object detection dataset could be found. Therefore, custom datasets needed to be prepared. In total, 3 datasets were created. These are for training keyboard and single-character detectors and validating post-processing correction algorithms.

Concerning the keyboards dataset, 615 keyboards of different types from various devices were collected. These were data augmented and generated to random backgrounds. The augmentation methods used were

scaling, blurring, brightness changing and the addition of various noises, transparency and moiré effects.

When it comes to the characters, the dataset is in grayscale. Randomly gray-colored characters and backgrounds to put them on were generated with a constraint of at least a 24-pixel intensity difference between them. This can be done because keys on keyboards are very contrastive and moving to grayscale removes any color design effects. The augmentation methods used were the same as for keyboards.

Concerning the post-processing validation dataset, keys on 120 selected keyboards with different layouts were annotated. The goal is to check if correct characters were recognized, redundant characters removed or missing characters computed.

3. Solution design

For the keyboard detection model was used *tiny* version of YOLOv7 neural network architecture. It achieved the same accuracy as the *standard* version while using fewer resources and it is significantly faster. It was trained for 30 epochs with batch size 16 and it takes 640x640 input images. It can accept any size but is scaled or padded to this resolution.

The input for the character detection is the output of the keyboard detection model. On the detected keyboard region is simultaneously run YOLOv7 model trained for characters and Canny edge detection. The reason for this is that the character detector cannot recognize e.g. space key when it is blank. The Canny detector provides supplementary results for the subsequent post-processing. The neural network was in comparison to the keyboard detector trained for 50 epochs due to having 99 classes instead of 1, so it takes longer. Both *tiny* and *standard* models were trained and this time the difference was significant. On the other hand, the post-processing results demonstrate why *tiny* version might be sufficient.

The goal of the post-processing algorithm is to recognize the layout and fix any incorrections. Undetected characters can be computed, letter case for characters such as xX, oO etc. corrected or special key keywords found. This further improves the recognition results.

4. Recognition results

The keyboard detection was a huge success. Even the *tiny* model achieves 100 % recall and precision and over 97 % mAP@.95 for both validation and testing data. Concerning the character detection, the numbers on the single-character test dataset are lower as table 1 depicts. Nevertheless, the results of the *standard* model are still nice considering the inclusion of special characters such as dots and commas. The *tiny* version fares much worse and while it is quite sure with its results with relatively high precision, the lower recall says it cannot find a lot of characters.

	Precision	Recall	mAP@.95
Keyboards (tiny)	1	1	0.97
Characters (tiny)	0.948	0.852	0.748
Characters (yolov7)	0.979	0.951	0.848

Table 1. Trained model results on test datasets

The post-processing algorithm, however, improves the character detection results and also removes the gap between *tiny* and *standard* models. On the post-processing validation dataset, the character recognition with applied post-processing achieves similar though slightly worse results shown in table 2 than the single-character detection on the character dataset. Notwithstanding, the results are still great considering this is a real-world and not a generated dataset. Moreover, these are averaged numbers impaired with special character detection on which not much post-processing is done. If focused only on the target alphanumeric characters, both precision and recall are incredibly good. What is more, the worse recall of the *tiny* model actually helps the post-processing as it does not handle many false positives and it leads to even better results. This demonstrates the power of the post-processing algorithm to correct detections and compute missing characters based on predefined layouts such as qwerty.

	Precision	Recall
All keys (yolov7)	0.942	0.949
All keys (tiny)	0.964	0.942
Alphanumeric (yolov7)	0.999	0.997
Alphanumeric (tiny)	1	0.998

Table 2. Post-processing algorithm results on post-processing validation dataset

5. Conclusions

The thesis offers a working modular solution for keyboard and keys image recognition problems. Any model can be easily retrained and switched to a different one. Similarly, additional post-processing techniques to cover other special cases can be included. The achieved results exceed expectations and in the future, support for more character sets can be added.

Acknowledgements

I would like to thank my supervisor Jan Pluskal for his help and Y Soft for making the thesis possible.

References

[1] Zongyi Liu, R.K. Gupta, Kun Chen, Bruce Ferry, and Simon Lacasse. A scalable computer vision framework for mobile device auto-typing. 01 2020.

[2] Zongyi Liu, Bruce Ferry, and Simon Lacasse. A deep neural network to detect keyboard regions and recognize isolated characters. 09 2019.

[3] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-yuan Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. 07 2022.