Methods of preprocessing screenshots of desktop applications for optical character recognition system

Abstract

This paper considers typical methods of image preprocessing before feeding them into machine vision systems for text recognition. The problem of data preprocessing for optical character recognition systems is solved using desktop applications as an example.

Using standard image preprocessing methods

The use of image preprocessing methods depends on the task area, image quality, and text orientation. Images obtained by screen capture of desktop applications are characterized by low resolution (100-150 dpi), an extensive color palette (both background and text color change), a variety of interfaces, fonts and the presence of characters that are not part of a given alphabet. The constant horizontal orientation of the text makes the task easier.

Some controls contain icons similar to the alphabet of the recognized language, which are difficult to remove. Interface components may be in different states, such as in focus, or unavailable for interaction. In the latter case, the hue of the text of the buttons is severely dimmed, the contrast deteriorates. There are also other cases where some of the text is beyond the scope of bringing in the desired appearance by thresholding alone. In some cases, partially solved by reducing the number of color channels, bringing the image into shades of gray. Therefore, it makes sense to create some sets of preprocessing methods applied to different kinds of components.

The use of preprocessing methods can improve the quality of both text detection and text recognition (without taking text detection errors into account). An example of poor localization and, as a consequence, poor text recognition is shown in Figure 1.

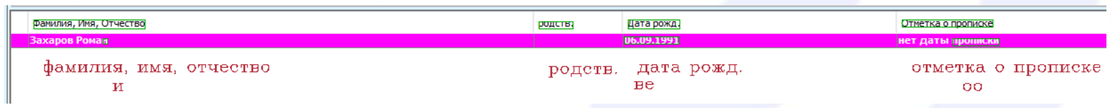

Figure 1. Localization and text recognition in the screenshot of the application table using Tesseract

Text localization

The initial step in most OCR technologies is text localization. Inaccurate localization consequently leads to inaccurate recognition and noisy results (Figure 2, Figure 3). In the context of solving the problem of text recognition on desktop images, a set containing 715 manually labeled screenshots, of existing desktop applications was collected. To augment the dataset, another 548 synthetic images were generated using three different UI creation libraries. As libraries of UI interfaces were used: WPF (C#), Pyside (Python), Java Swing. The use of different UI libraries allowed us to get the most diverse examples of interfaces at the lowest cost.

Figure 2. Localization of text by built-in Tesseract methods

Figure 3. Localization of text using YOLO

Results

To improve the quality of localization, a neural network of YOLO version 5 architecture was trained. The model was chosen because of the high values of metrics in solving the problem of finding objects on public datasets, high speed of operation. Training was done on the training part of the dataset. The results were compared using the IoU (Intersection over Union) metric. The numerical values of the performance of different methods on the test part of the sample are shown in Figure 4.

The best quality of text localization from OCR systems was demonstrated by Paddle. YOLO is successful in high-quality text localization, outperforming Paddle OCR by 5.8% in the IoU metric.

Figure 5 shows several examples of text recognition.

Method	IoU
Tesseract	0.31
Paddle (MobileNetv3)	0.50
CRAFT	0.86
YOLO	0.91

Figure 4. Text localization results

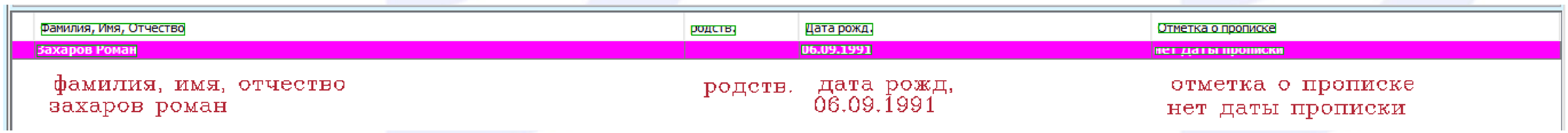

Figure 5. Example of localization and text recognition in the screenshot of the application table

Contact us: Rud' S. A. – jin_.96@mail.ru