UDC 004.912.05 **doi: 10.32620/reks.2021.4.15**

K. DERGACHOV, L. KRASNOV, V. BILOZERSKYI, A. ZYMOVIN

National Aerospace University "Kharkiv Aviation Institute", Ukraine

DATA PRE-PROCESSING TO INCREASE THE QUALITY OF OPTICAL TEXT RECOGNITION SYSTEMS

The subject of study *in the article is the formulation of a modern concept of improving the quality of work of optical recognition systems by using a set of various algorithms for preprocessing document images at the user's discretion. The research synthesizes algorithms that compensate for external negative influences (unfavorable geometric factor, poor lighting conditions when photographing, the effect of noise, etc.). The* ***methods*** *used imply a certain sequence of data preprocessing stages: geometric transformation of the original images, their processing with a set of various filters, image equalization without increasing the noise level to increase the contrast of images, the binarization of images with adaptive conversion thresholds to eliminate the influence of uneven photo illumination. The following results were obtained. A package of algorithms for preliminary processing of photographs of documentation has been created, in which, to increase the functionality of data identification, a face detection algorithm is also built in, intended for their further recognition (face recognition). A number of service procedures are provided to ensure the convenience of data processing and their information protection. In particular, interactive procedures for text segmentation with the possibility of anonymizing its individual fragments are proposed. It helps provide the confidentiality of the processed documents. The structure of the listed algorithms is described and the stability of their operation under various conditions is investigated. Based on the results of the research, a text recognition software was developed using the Tesseract version 4.0 optical character recognition (OCR) program. The program "HQ Scanner" is written in Python using the OpenCV library. An technique for evaluating the effectiveness of the algorithms using the criterion of the maximum probability of correct text recognition has been implemented in software. A large number of examples of system operation and software testing results are provided.* ***Conclusions****. The results of the research conducted are a basis for developing software for creating cost-effective and easy-to-use OCR systems for commercial use.*

Keywords: *optical character recognition (OCR); image original geometry transformation; filter algorithm; picture equalization and binarization; face detection algorithm; probability of correct text recognition; segmentation of texts and anonymization of their individual fragments.*

Introduction

The rapid development of digital technologies based on the use of artificial intelligence algorithms has become possible due to incredible progress in such characteristics of computer technology as speed of operation and memory capacity, as well as the emergence of new operating systems and programming languages. This led to the broad introduction of innovative data processing techniques in computer vision routines goaled to image recognition [1 − 3].

Two most relevant tasks are solved using image recognition tools: face detection and recognition, and optical character recognition (OCR) needed for further translating a textual image into a digital computer format. The need to improve these technologies is constantly growing [4 − 6].

Numerous applications turn down usage of reading and editing information present in paper format. On the other hand, special systems for texts and digital data translation from paper to electronic format have been developed presently. Seeking for information and further utilization of that is much easier in digital documents than using of handwritten or printed paper data. Extracting of a digital textual pattern from the snapshot finds many useful uses. Among them are the digitization of paper archives, passports verification, automatic recognition of transport license plates, etc. Hence, the task of converting handwritten and printed texts into digital format is essentially topical.

Meaningful achievements in resolving problems of objects recognition and classification are based on the use of up-to-date methods and algorithms for constructing and learning deep neural networks (DNN). The most important role among them is played by convolutional neural networks (CNN), which make it possible to bring decision quality indicators up to 99 %. Although, serious limitations exist and they exert the image treatment outcome, namely: poor quality of photo/video shots, insufficient scene illumination, geometric distortions, a number of other factors.

To the moment, leading experts in OCR reviewed and analyzed thoroughly the works related to assessment of the negative impact of certain factors on the efficiency of text recognition [7]. They have brought a number of constructive proposals to compensate for these influences [8, 9]. The most attention in these works concerns algorithms for compensating the ill-conditioned angular position of the text documents picture relative to the shooting camera coordinate system. However, the imact issue of uneven and continuously changing illumination of the scene when photographing text documents is not adequately discussed, though, this may result in a sharp degradation of the text recognition quality, up to the loss of text fragments when they happen to be in shadowed picture areas. Unfortunately, there are no universal recommendations on the matter at present, let alone standards [10 – 13].

The analysis of the current state of the problem for enhancing of the OCR system quality, carried out by the authors, signifies the need for a comprehensive solution to the problem, which would cancel as many disadvantageous effects as present (images geometric distortions, uneven illumination, insufficient contrast, presence of noise, etc.). A preliminary treatment and certain correction of data is therefore required to supply their appropriate recognition and classification.

1. Formulation of the problem

Work objectives constitute elaboration of a set of algorithms for preprocessing pictures in the paper document recognition system, which are used to improve the OCR quality. Based on these results, it is necessary to develop software and provide high-quality optical text recognition considering disturbing influences. The ultimate goal is to create a system that transforms text documents from paper to electronic format.

The concept of the system establishment intends to provide effectual reduction of external factors influence on the quality of texts detection; inquired text recognition is being made as to the result of special processing photographs of the paper doc. The handling includes routines as follows:

- assessment of negative factors that could affect the operation of the text recognition system;
- specifying the criterion and related indicators concerning the quality of text recognition in the system;
- design of the set of algorithms for pre-processing images in the OCR system that will effectively compensate the influence of disturbing factors;
- developing and testing the computer program used for practical implementation of the approach.

2. Tools and methods of the study

Instrumentation and methods of study. The study is focused on the use of affordable and relatively cheap photography tools like phones, tablets, cameras, web cameras, and other non-volatile devices.

The factors of negatory impact on the system performance:

- low quality of original documents (poor text's printing, shots low contrast, poor paper quality);
- unfavorable lighting conditions during the photography provision;
- low quality of the photo equipment used, incorrect actions by the operator, hence shots defocusing and other flaws;
- geometrical distortions caused by the wrong shooting angle with respect to the frame of reference assigned for texts recognition.

Text recognition fidelity evaluation. To correctly assess the text recognizer efficiency under various conditions, the performance quantitative indicator was proposed. That allows us to get a primary estimation of recognition performance followed by carrying out a comparative analysis for various conditions of the experiment.

Dealing with texts recognizing, one needs first of all to assess the recognition fidelity, or **Accuracy**, quantitatively. In the study, the declared accuracy (percent) is evaluated as follows:

$$\mathbf{Accuracy} = \frac{\mathbf{N_{dst} - N_{extra} - N_{missed}}}{\mathbf{N_{src}}} \cdot \mathbf{100}\%\,,$$

where $\mathbf{N_{src}}$ is the number of characters in the source text; $\mathbf{N_{dst}}$ − number of characters correctly recognized; $\mathbf{N_{extra}}$ − number of excess characters appeared while recognizing; $\mathbf{N_{missed}}$ − number of unrecognized characters.

Usage of this indicator makes it possible to take into account not only the number of correctly recognized characters, but also recognizer errors that manifest the omission of individual characters or the addition of over characters that were not present in the text.

The $\mathbf{N_{src}}$ value is strictly positive and depends solely on the source text. The value $\mathbf{N_{dst}}$ is also positive and it may not be larger than $\mathbf{N_{src}}$:

$$\begin{cases} \mathbf{N_{src}} > 0; \\ \mathbf{N_{src}} \geq \mathbf{N_{dst}} \geq \mathbf{0}. \end{cases}$$

Resources used. While developing the project, the Python programming language and resources of the OpenCV library were used [20, 22, 23]. These were chosen through the open access to the software products and compatibility with Windows, Linux, and Android operating systems. As well, the Tesseract facility as one of the specialized program modules was used for text optical recognition [24 − 28]. This is the OCR open engine that employs neural networks to search and recognize textual information in images. The Tesseract engine uses the Long short-term memory architecture (LSTM) for recurrent neural networks [25, 26]. The chosen software puts an insignificant load on the processor and obtains preconditions to treat data in real-time. Under this approach, the operation of the optical character recognition system can be readily implemented on a PC or laptop, and a single-board computer Raspberry Pi [21], if a mobile version of the system is required.

3. Structure of the system and algorithms of text recognition

Construction of the high-performance text recognition system that was focused on protecting the system of the negative impact of disturbing factors assumes a strict sequence in treating the initial data, as well as the recognition routine structure optimizing.

Source data processing phases. Usually, the operation of the systems for textual-format documentation recognition does not require real-time data processing. Although, there are often needed interactive procedures (such as viewing, text segmentation, so forth), which slow down the program execution significantly.

In combination, while executing the program it is necessary to follow strictly the sequence of data processing phases to achieve the text recognition of high grade. In the proposed approach, these steps are as follows:

− uploading the beforehand shot photographs of the document (the jpg format, each page distinctly). In the system card index (DB), they are named according to the rule: `file_name_page_i`, where `i` is doc's page number (`i = 1 – N`);

− performing the geometric transformation that will adapt angular attitude of the document in the shot according to the reference system of the recognition facility. Even minor angular inconsistency (about 3°) drastically reduces fidelity of texts recognition;

− preview of the document photos made, and deleting, if necessary, failed ones;

− further, the action (very important occasionally) on running the interactive segmentation procedure is brought: using the mouse, user extracts a text fragment desired. This fragment will be a subject for future analysis and the rest text is ignored. Or conversely. Besides, it is equally important to use one more characteristic mode: the one, which deals with anonymizing individual text fractions. This will provide for confidentiality of the treated information;

− preliminary image processing. That implies a multivariate noise filtering followed by pictures equalizing in order to enhance their contrast; binarization of the gotten image is stimulated next to provide the more proper recognition;

− as finish, the text recognition and saving the results in a required format (docx, pdf, etc.) is being executed.

Each of the described procedure stages is written as a separate algorithm and takes the form of a special software module. The latter's structure and practical implementation will be discussed in detail below.

System's algorithms and software modules. A diagram of the text recognition system, which is designed on a modular principle and got the name "HQ Scanner", is shown in fig. 1. For brevity, we consider in detail the algorithms and instances for software implementation of the core ones only.

Resources used in the design of program interface. The project makes use of the Qt Designer software and PyQt5 library. The PyQt5 constitutes a set of graphical framework Qt extensions for the Python. Besides, the PyQt5 holds a special constructor of the user graphics interface − the Qt Designer (Qt Creator); the dedicated program `pyuic` generates a Python code from files created by the Qt Designer. The user interface windows and short description of their purposes are given later in the paper.

Geometric transformation of the source image. The objective of identification and placing of a text document to a certain coordinate system for subsequent recognition is the key issue while processing the shots taken with cameras, smartphones or tablets (especially with using neither tripod). The images gotten in this way have two fundamental disadvantages:

− the text of interest takes only a certain partition of the entire picture (we take that the background does not contain useful information);

− when shooting, the document vertical did not take the right angle to the camera optical axis. It is difficult to meet this requirement while photographing.

The first issue leads to a slowdown of the recognizer practice, and the second may result in a sharp decrease of the recognition exactness. A special study was brought about the influence of the text doc angular attitude on the recognition fidelity. The study results are shown in fig 2. At relatively small deflection angles (up to 3°), the recognition accuracy decreases little – within 1 % only; however, along 4° mismatching, the index of right recognition drops to 54.66 %, and at 5° that becomes 25.46 % at all.

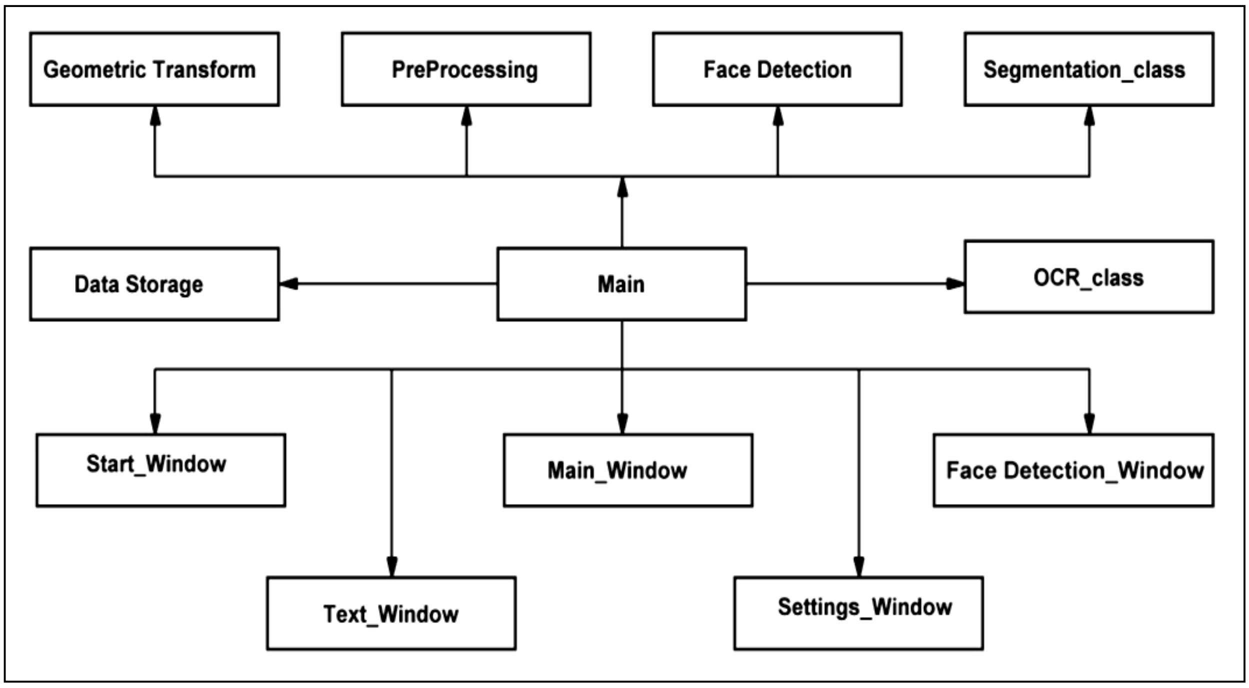


Fig. 1. Structure of the text recognition system «HQ Scanner»

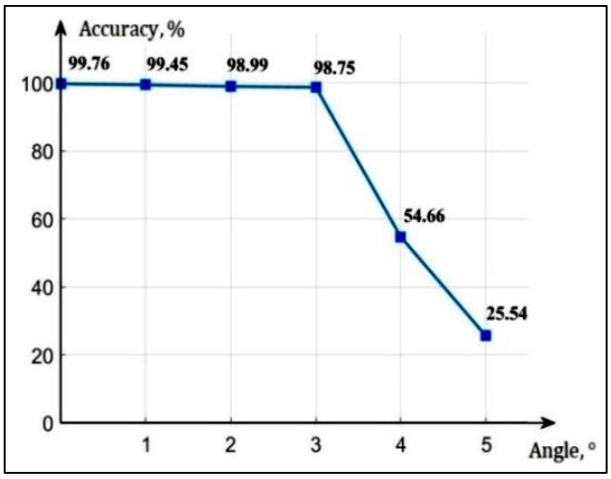


Fig. 2. Text recognition fidelity against document shooting angle deflection

The graph shows that shooting conditions for the text document exert a critical effect on the fidelity of the textual recognizer.

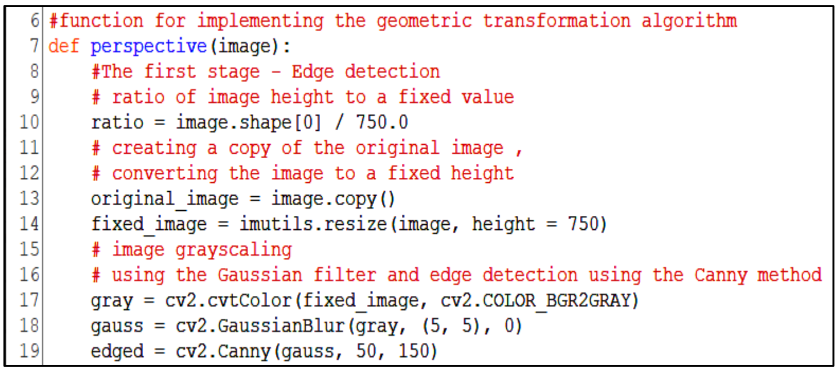
To solve the geometric transformation task, the algorithm is developed, a heart of which is outlining the routine of the textual content in the image, followed by transformation of the outline perspective with the use of the identified contour corner points. The routine implements the three basic stages:

1. Detection of the picture edges.
2. Specifying of the text outlines in the picture.
3. Transformation of the image perspective using corner points.

At the first stage, preprocessing of the source image is performed for edges appropriate detection by Canny method (fig. 3). With this routine, the ratio of the source image height over the reference value is calculated (line 10); then a copy of the original image is created, and conversion is terminated by reducing the original image to the control height value (lines 13, 14). The image is then recorded into a grayscale pattern and treated with the Gaussian filter (lines 17, 18). The procedure outcome is the textual picture with bounds edged by the Canny method (line 19). Conversion to a grayscale pattern speeds up the image post-processing routines, and the filter reduces the noise level.

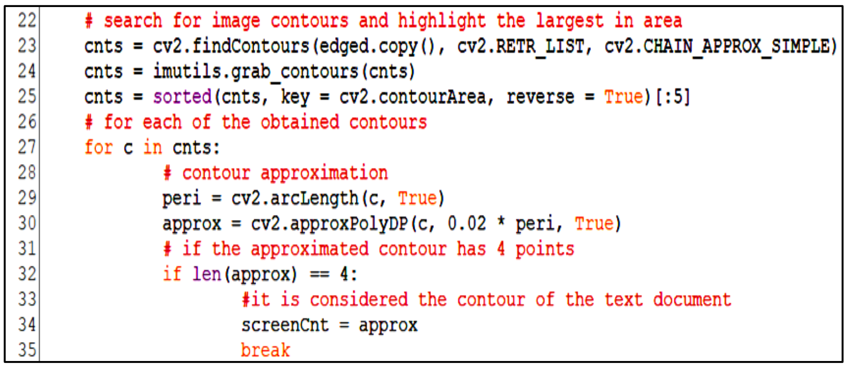
At the second stage, the search of the image particular pieces outline is carried; the action is terminated by outlining the biggest contour of the rectangular shape (fig. 4). In this excerpt, one of the two assigned tasks is being solved, namely the separation of the textual portion from the background with determining the text outlines.

At first, the script (appropriated for the Python version used) grabs all the contours on the modified image copy, i.e. the one with the edges specified (lines 23, 24). Next, the routine distinguishes and sorts the found contours by area. That reduces the number of contours to apply coming analysis to the patterns that have a large perimeter only (line 25). Further, travel is carried out along each of the accented contours and their approximation followed by identifying the corner points



```
 6 #function for implementing the geometric transformation algorithm
 7 def perspective(image):
 8     #The first stage - Edge detection
 9     # ratio of image height to a fixed value
10     ratio = image.shape[0] / 750.0
11     # creating a copy of the original image ,
12     # converting the image to a fixed height
13     original_image = image.copy()
14     fixed_image = imutils.resize(image, height = 750)
15     # image grayscaling
16     # using the Gaussian filter and edge detection using the Canny method
17     gray = cv2.cvtColor(fixed_image, cv2.COLOR_BGR2GRAY)
18     gauss = cv2.GaussianBlur(gray, (5, 5), 0)
19     edged = cv2.Canny(gauss, 50, 150)
```

Fig. 3. Code passage for identifying text document margins



```
22     # search for image contours and highlight the largest in area
23     cnts = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
24     cnts = imutils.grab_contours(cnts)
25     cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:5]
26     # for each of the obtained contours
27     for c in cnts:
28             # contour approximation
29             peri = cv2.arcLength(c, True)
30             approx = cv2.approxPolyDP(c, 0.02 * peri, True)
31             # if the approximated contour has 4 points
32             if len(approx) == 4:
33                     #it is considered the contour of the text document
34                     screenCnt = approx
35                     break
```

Fig. 4. Script for second pre-processing stage

from each of the approximated outline. If the number of points is four (rectangle), then we assume that the corresponding item is a textual content outline and we save this member as one of probable outcomes (lines 27 – 35). Note that it is necessary to ensure full visibility of the textual portion in the image to reveal its rectangular outline clearly and thus complete the second stage happily.

At the concluding stage, transformation of the perspective is performed using the corner points of the detected rectangular contour (fig. 5). This intends to correct angular mismatching of the doc picture due to unclear photographing.

In the beginning of the given script, the contour of the text box that is gained at the previous step takes the form of 4x2 format array (line 39). After defining the coordinates of the control points, the perspective transformation should be performed using the following procedure:

1. The control points are being normalized with accepting the previously defined ratio for the heights of the initial and rescaled image. The special function `'points_normalization'` together with writing the normalized point coordinates as distinct variables are used to implement this (lines 41, 42).

2. The width `maxWidth` and the height `maxHeight` for the new image are calculated as the maximum distance between the control points (lines 45 – 52).

3. The final coordinates `dst`, to which the control points will be bound in the perspective transformation routine, are formed (lines 54 – 58).

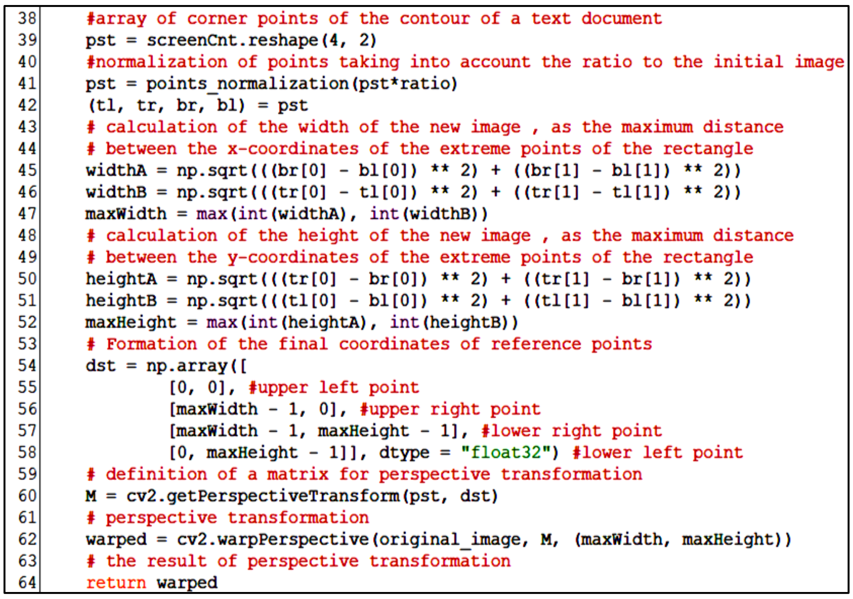
4. The matrix `M` to provide the control points perspective transformation to the final coordinates, is computed (line 62).

To complete the process, a special function `points_normalization()` is used. The operator normalizes the reference points under following the strict pass order: top-left, top-right, bottom-right, bottom-left. Related script of the code is shown in fig. 6.

The respective results at each stage of treating the given image are shown in fig. 7. The work effectiveness was proved by examining the same pictures turned by angles from 1° to 5°.

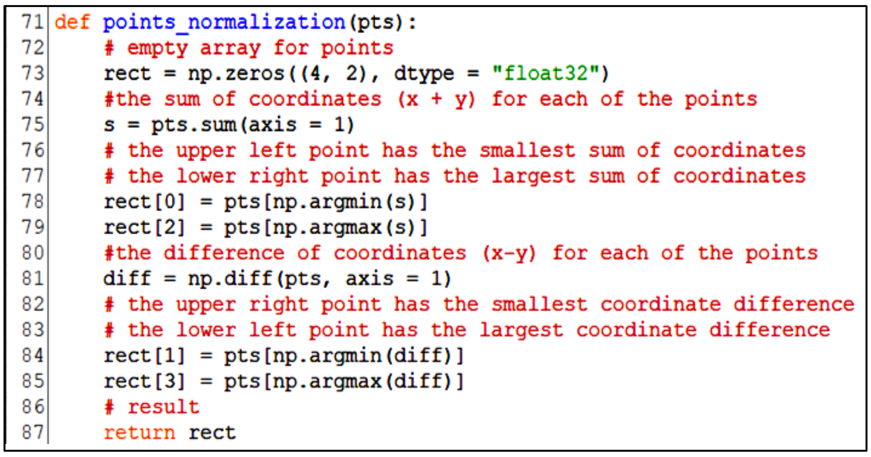
The algorithm of geometric transformations allows achieving the high grade recognition accuracy and its stability regardless of the initial shooting angle of the document when photographing. The routine maintains invariable value of recognition accuracy at different allowed angles of deviation and removes unnecessary background (fig. 7, d). With geometric transformations, the error does not exceed 2°. This is sufficient for high recognition accuracy.

Binarization in images processing. This handling allows us to bring the text characters to the united level of black, thereby increasing contrast of the treated text picture. The most important thing when providing a qualitative binarization is setting of a fitted threshold. If the image pixel index appears to be greater the threshold, then it is assigned with the first of two values (white), otherwise that accepts another value (black). To provide this, the function `cv2.threshold` is used. The function's first input introduces the source image in grayscale.



```
    #array of corner points of the contour of a text document
    pst = screenCnt.reshape(4, 2)
    #normalization of points taking into account the ratio to the initial image
    pst = points_normalization(pst*ratio)
    (tl, tr, br, bl) = pst
    # calculation of the width of the new image , as the maximum distance
    # between the x-coordinates of the extreme points of the rectangle
    widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
    widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
    maxWidth = max(int(widthA), int(widthB))
    # calculation of the height of the new image , as the maximum distance
    # between the y-coordinates of the extreme points of the rectangle
    heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
    heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))
    # Formation of the final coordinates of reference points
    dst = np.array([
            [0, 0], #upper left point
            [maxWidth - 1, 0], #upper right point
            [maxWidth - 1, maxHeight - 1], #lower right point
            [0, maxHeight - 1]], dtype = "float32") #lower left point
    # definition of a matrix for perspective transformation
    M = cv2.getPerspectiveTransform(pst, dst)
    # perspective transformation
    warped = cv2.warpPerspective(original_image, M, (maxWidth, maxHeight))
    # the result of perspective transformation
    return warped
```

Fig. 5. Script of the perspective transformation code



```
def points_normalization(pts):
    # empty array for points
    rect = np.zeros((4, 2), dtype = "float32")
    #the sum of coordinates (x + y) for each of the points
    s = pts.sum(axis = 1)
    # the upper left point has the smallest sum of coordinates
    # the lower right point has the largest sum of coordinates
    rect[0] = pts[np.argmin(s)]
    rect[2] = pts[np.argmax(s)]
    #the difference of coordinates (x-y) for each of the points
    diff = np.diff(pts, axis = 1)
    # the upper right point has the smallest coordinate difference
    # the lower left point has the largest coordinate difference
    rect[1] = pts[np.argmin(diff)]
    rect[3] = pts[np.argmax(diff)]
    # result
    return rect
```

Fig. 6. Script implementing normalization function

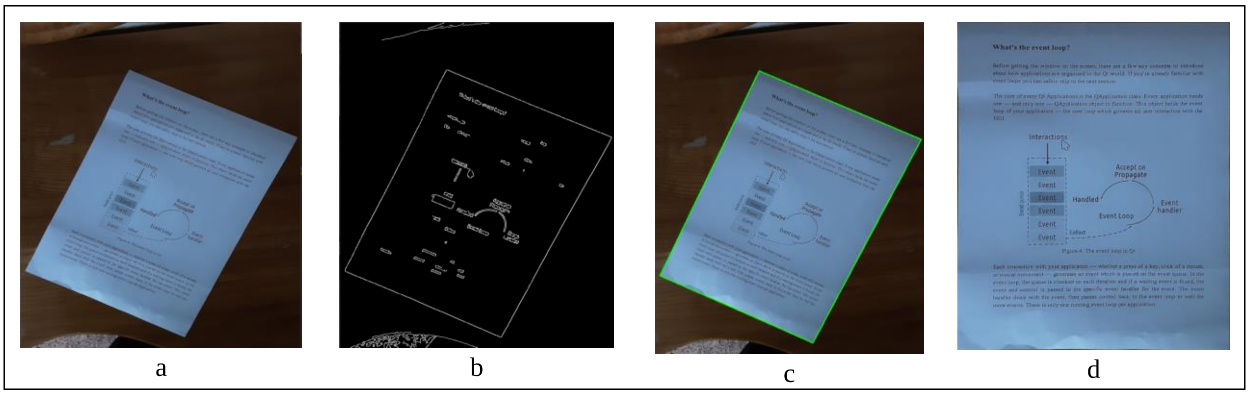


Fig. 7. Screenshots of algorithm operation step-by-step: a – the source image; b – edges detection; c – outline selection; d – perspective transformation

The second input is the threshold value required to arrange pixel indices. The third entry, `maxVal`, is the pixel return that would be assigned according to decision on the current value of the pixel output index – it was greater / less the threshold value. The OpenCV provides for several techniques to choose the threshold; the forth argument of the operator `'cv2.threshold()'` inputs a desired one. Here are these offers:

- `v2.THRESH_BINARY;`
- `cv2.THRESH_BINARY_INV;`
- `cv2.THRESH_TRUNC;`
- `cv2.THRESH_TOZERO;`
- `cv2.THRESH_TOZERO_INV.`

For bimodal images, which possess two distinct peaks in the histogram, a value between these peaks can be taken for the threshold. The same approach is used in binarization by Otsu. Herein, the threshold value for bimodal pictures is calculated automatically from the image histogram. If the image is non-bimodal, binarization will not be accurate.

The Otsu binarization employs also the `cv2.threshold()` operator, however that demands an additional flag – `cv2.THRESH_OTSU`. Initially, the threshold value is set to zero. After, the algorithm determines the optimal threshold and returns the `retVal` as the second parameter. If the Otsu threshold use is not needed, the `retVal` entry holds the same threshold value as previously used.

However, it is not always productive to use fixed thresholds. For example, if the picture scenes have essentially different illumination, this is such a situation. In this case adaptive thresholds are used. The algorithm calculates now a specific threshold value for the particular image area. The following OpenCV functions can be turned on for this:

– `cv2.ADAPTIVE_THRESH_MEAN_C`. The subroutine sets a threshold that is tuned for the average brightness of adjacent image areas;

– `cv2.ADAPTIVE_THRESH_GAUSSIAN_C`. The threshold is adjusted to a weighted sum of surrounding values, and the weights constitute the Gaussian window. Herein, the block dimensions determine the size of the area, and `C` is the constant computed from either the precalculated average or the weighted average.

Fig. 8 shows these routines operational code realizing the comparative study of every type of binarization; fig. 9 displays the efficiency results in each type. Light exposure of the scene was considered to be uniform everywhere.

As it can be seen from fig. 9, a threshold choice takes a key part in binarization. Thus, when using the standard threshold value 127, the recognition accuracy holds the 94.68 % value, and with the 160 threshold it drops to 58.36 %. In contrast, the accuracy grew up to 99.71 % when using a threshold of 100. With Otsu binarization, the threshold is determined automatically, and this gave rise the recognition accuracy increase to 99.71 %. The similar fidelity was obtained using adaptive binarization techniques.

It can be concluded from the experiment that using of binarization with conventional manual adjustment of thresholds is irrational as it requires individual tuning for every new image.

To conclude advantage of the binarization by Otsu and various adaptive methods, extra experiments were carried out. Several images with uneven scene areas illumination were chosen as the raw patterns.

Fig. 10 shows the experimentation result with respect to illumination irregularity present in pictures. It is clearly seen that the right half of the original photo-document is substantially shaded.

```
#binarization
def binarization(image, method):
    #Checking the binarization method
    if method == 'threshold_127': #threshold binarization (threshold = 127)
        result = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)[1]
    elif method == 'threshold_100': #threshold binarization (threshold = 100)
        result = cv2.threshold(image, 95, 255, cv2.THRESH_BINARY)[1]
    elif method == 'threshold_160': #threshold binarization (threshold = 160)
        result = cv2.threshold(image, 155, 255, cv2.THRESH_BINARY)[1]
    elif method == 'otsu': #Otsu binarization
        result = cv2.threshold(image,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
    elif method == 'adaptive_mean': #adaptive mean binarization
        result = cv2.adaptiveThreshold(image,255,cv2.ADAPTIVE_THRESH_MEAN_C,\
            cv2.THRESH_BINARY,17,18)
    elif method == 'adaptive_gaussian': #adaptive gaussian binarization
        result = cv2.adaptiveThreshold(image,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
            cv2.THRESH_BINARY,17,18)
    return result
```

Fig. 8. Implementation of image binarization function

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

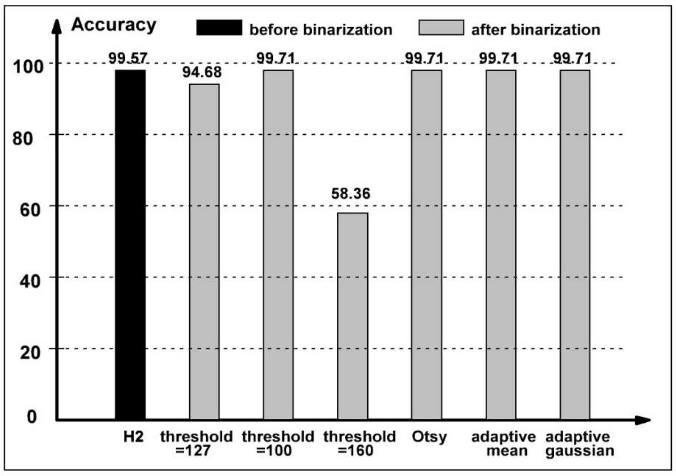


Fig. 9. Comparative characteristics in effectiveness of different binarization types

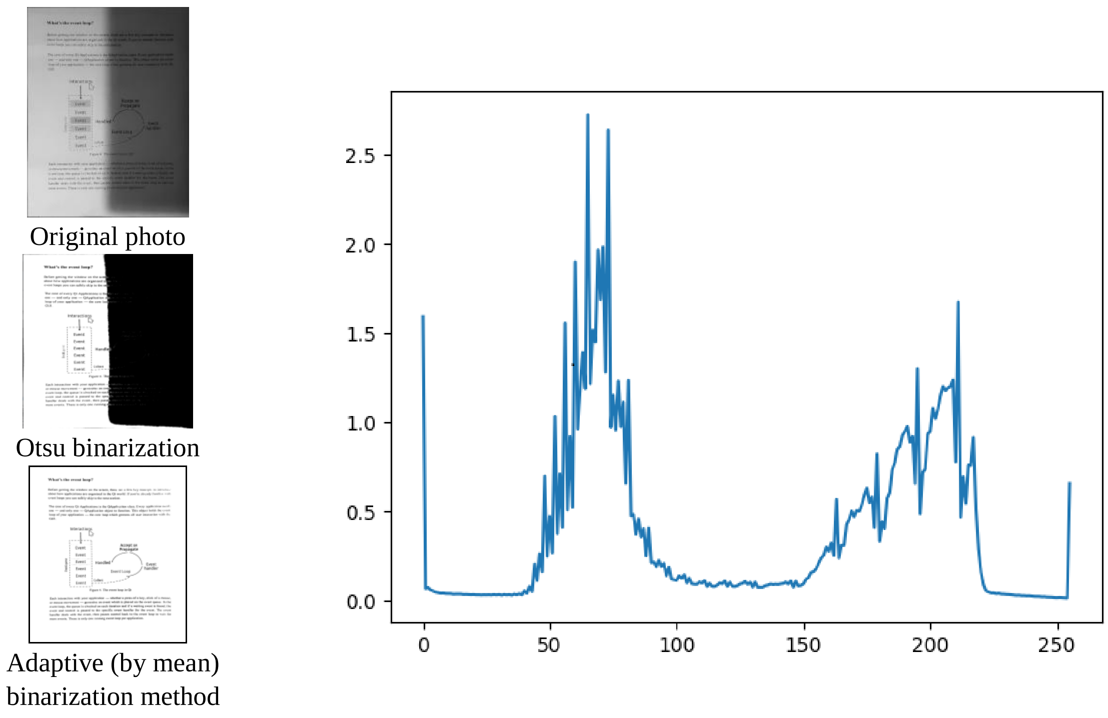


Fig. 10. Effectiveness of two binarization techniques

The normalized histogram for brightness distribution on this image contains a powerful component related to the dark pixels. At the same time, the use of image binarization by Otsu results in a complete loss of half of the information contained in the document, but the use of adaptive method allows you to fully recognize the contents of the document.

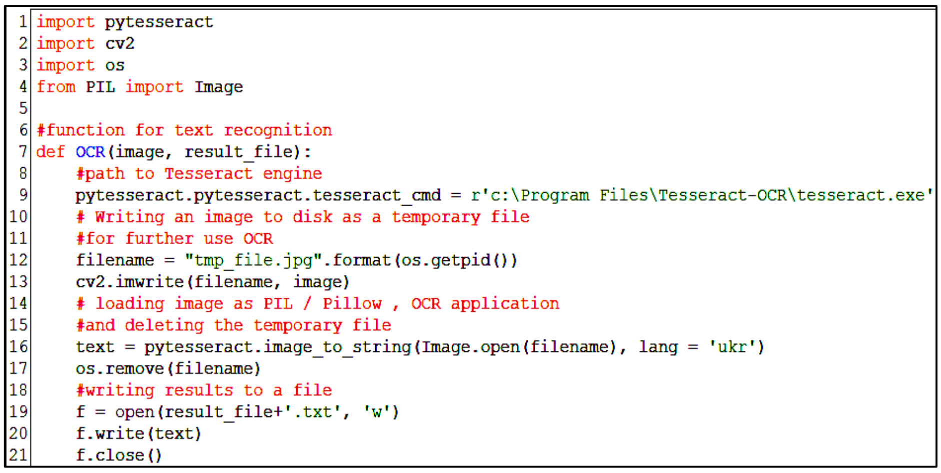
Text recognition with the use of Python and Tesseract. When using the Python and Tesseract programs, one can access a special `pytesseract` library [20]; on the basis of these, a special program `OCR_module` was developed for recognizing textual content in doc pictures followed by writing the results to a separate file (fig. 11).

This module generates the distinct `OCR` function, which possesses two entries, namely: the `image` file, which brings the template waiting for recognition, and the `result_file`, which has to return the operation result of the function.

The complete algorithm of the OCR function includes the following steps:

1. Captu re of the Tesseract engine to the device used (line 9).
2. Creation of the temporary (working) image file in jpg format (lines 12, 13).
3. Upload of the working file as a Pillow library file, then executing the image recognition and deleting the temporary file (lines 16, 17).

Download of the recognized text picture to a dedicated file (lines 19 – 21).



```
import pytesseract
import cv2
import os
from PIL import Image

#function for text recognition
def OCR(image, result_file):
    #path to Tesseract engine
    pytesseract.pytesseract.tesseract_cmd = r'c:\Program Files\Tesseract-OCR\tesseract.exe'
    # Writing an image to disk as a temporary file
    #for further use OCR
    filename = "tmp_file.jpg".format(os.getpid())
    cv2.imwrite(filename, image)
    # loading image as PIL / Pillow , OCR application
    #and deleting the temporary file
    text = pytesseract.image_to_string(Image.open(filename), lang = 'ukr')
    os.remove(filename)
    #writing results to a file
    f = open(result_file+'.txt', 'w')
    f.write(text)
    f.close()
```

Fig. 11. The module for image's textbox recognition

4. Case study. The interface and operation of the program «HQ Scanner»

In top, the resources needed for the program execution, algorithms and program codes of essential modules are described in detail. Let us study in more detail the structure of the interface and the features of users' handling the program.

Program user's windows. When you get started the «HQ Scanner» program, the initial window, shown in fig. 12, appears in the screen. One can run or terminate the program using conventional controls «Start» and «Exit» (fig. 13).

The start button calls the source data directory window. The catalog for storing the data can be located in either computer disk. Each thematic directory is arranged as a folder with the set of photos of the particular document. A multi-page document is composed of separate page files containing a corresponding picture (format `*.jpg` recommended). It is advisable to appoint to the file a name like «`filename_i.language.jpg`» where `i` is the number of the document page; for instance, '`image_1.eng.jpg`'.

In the «HQ Scanner» program, the treated files are being placed for convenience in the same folder with the photo files of the incoming documents. However, these should have a different format (*.docx or *.pdf). The program menu offers sorting files in the folder both by name and by format feature.

The master window button «Open» locates the file containing the picture of required doc page (fig. 14).

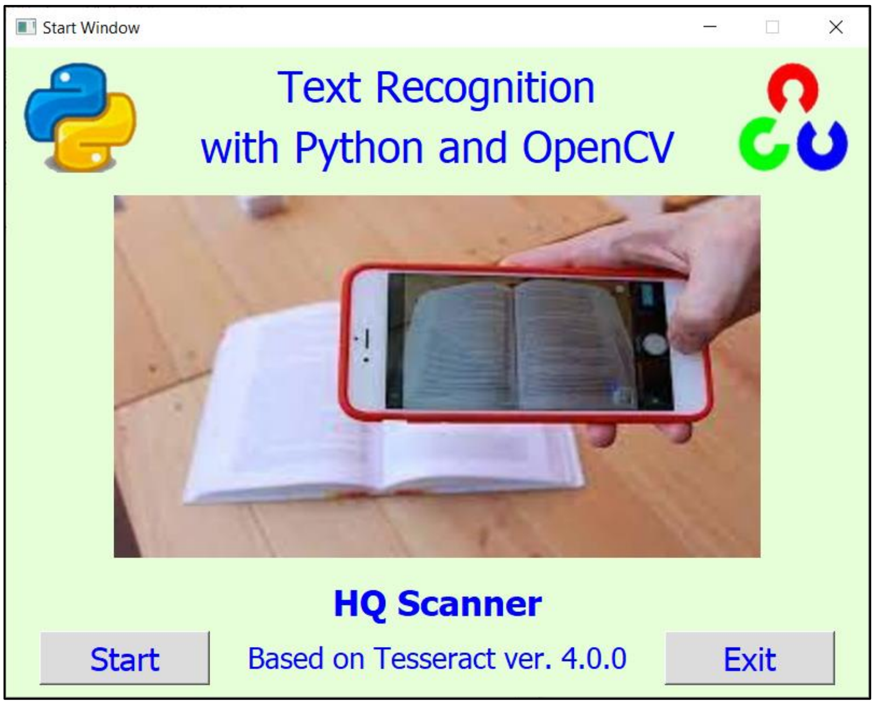


Fig. 12. Start window of the program

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

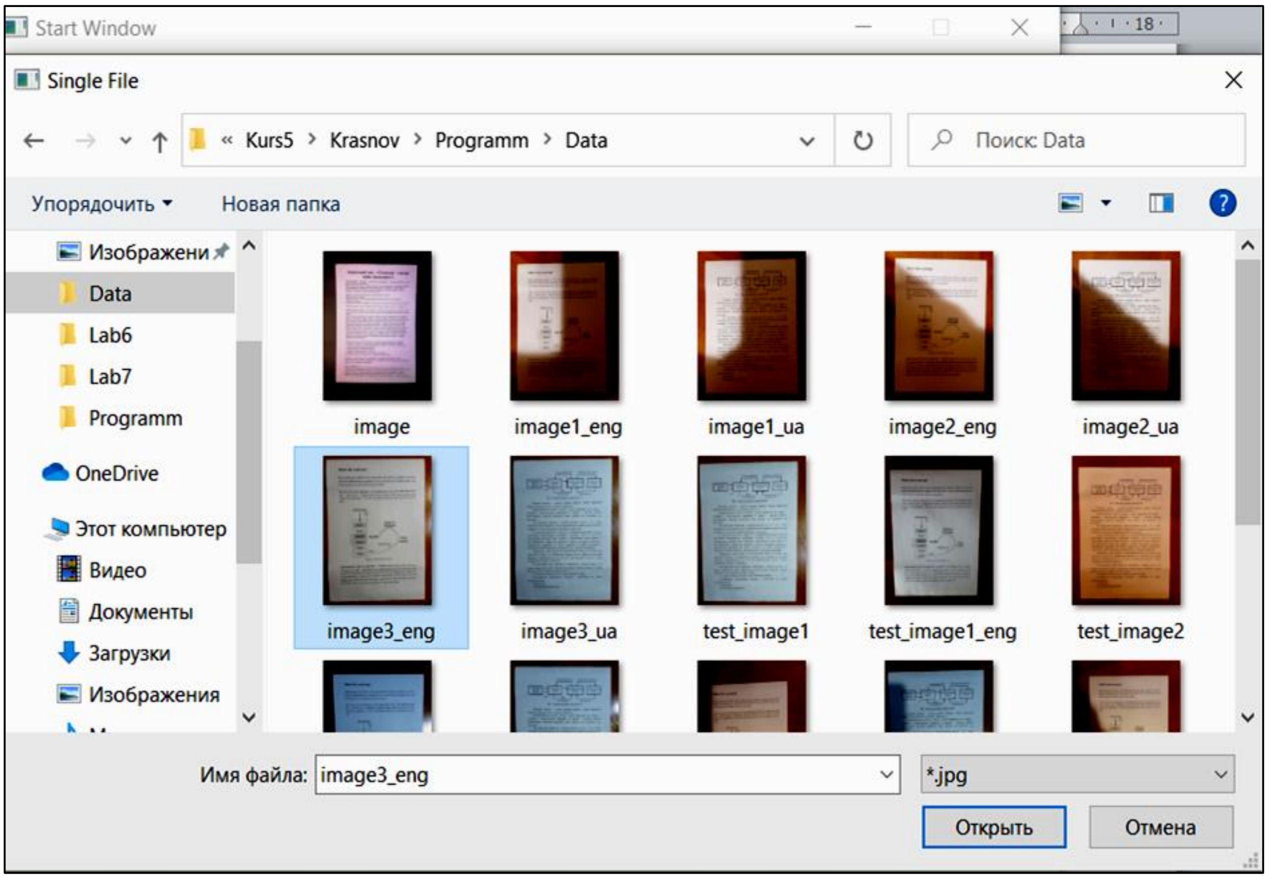
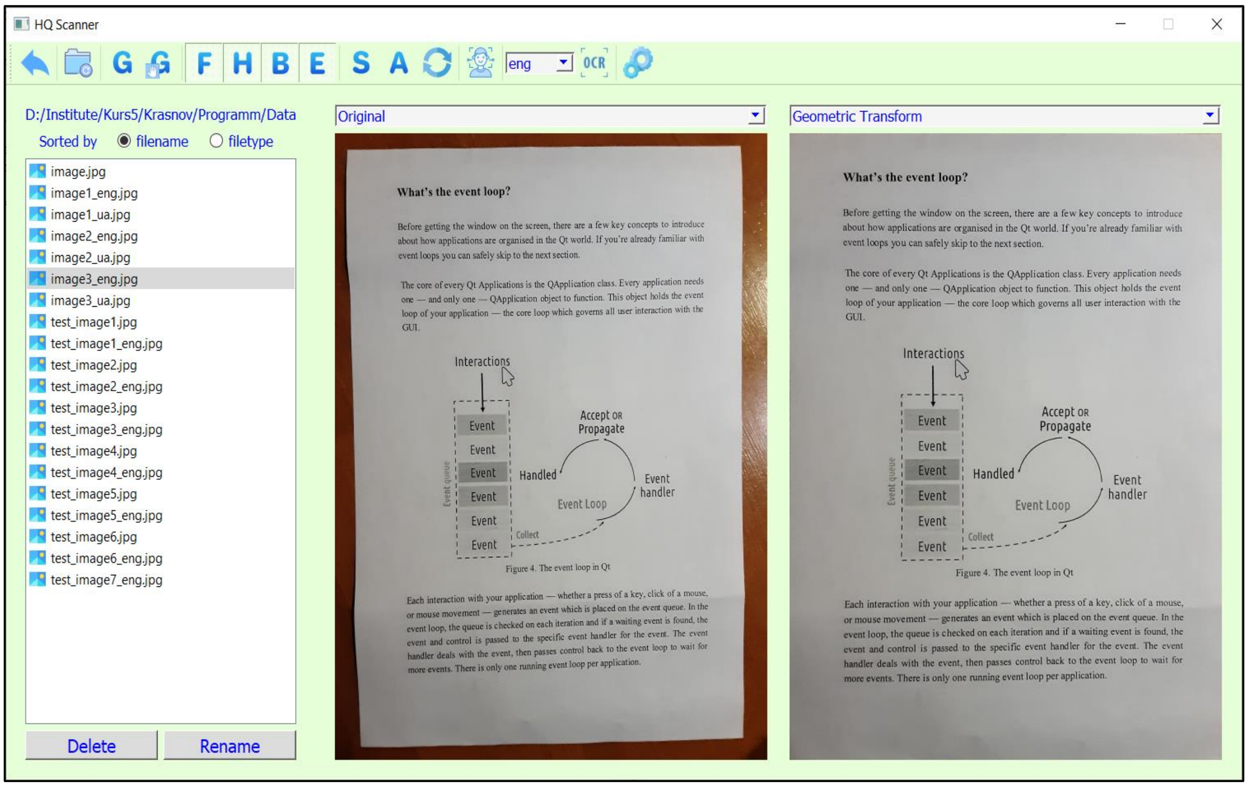


Fig. 13. Source data tab



Sorted by ● filename ○ filetype

image.jpg
image1_eng.jpg
image1_ua.jpg
image2_eng.jpg
image2_ua.jpg
image3_eng.jpg
image3_ua.jpg
test_image1.jpg
test_image1_eng.jpg
test_image2.jpg
test_image2_eng.jpg
test_image3.jpg
test_image3_eng.jpg
test_image4.jpg
test_image4_eng.jpg
test_image5.jpg
test_image5_eng.jpg
test_image6.jpg
test_image6_eng.jpg
test_image7_eng.jpg

What's the event loop?

Before getting the window on the screen, there are a few key concepts to introduce about how applications are organised in the Qt world. If you're already familiar with event loops you can safely skip to the next section.

The core of every Qt Applications is the QApplication class. Every application needs one — and only one — QApplication object to function. This object holds the event loop of your application — the core loop which governs all user interaction with the GUI.

Figure 4. The event loop in Qt

Each interaction with your application — whether a press of a key, click of a mouse, or mouse movement — generates an event which is placed on the event queue. In the event loop, the queue is checked on each iteration and if a waiting event is found, the event and control is passed to the specific event handler for the event. The event handler deals with the event, then passes control back to the event loop to wait for more events. There is only one running event loop per application.

What's the event loop?

Before getting the window on the screen, there are a few key concepts to introduce about how applications are organised in the Qt world. If you're already familiar with event loops you can safely skip to the next section.

The core of every Qt Applications is the QApplication class. Every application needs one — and only one — QApplication object to function. This object holds the event loop of your application — the core loop which governs all user interaction with the GUI.

Figure 4. The event loop in Qt

Each interaction with your application — whether a press of a key, click of a mouse, or mouse movement — generates an event which is placed on the event queue. In the event loop, the queue is checked on each iteration and if a waiting event is found, the event and control is passed to the specific event handler for the event. The event handler deals with the event, then passes control back to the event loop to wait for more events. There is only one running event loop per application.

Fig. 14. The program master window

At the top, program master window locates a toolbar. The workspace of the main window of the program can be conditionally divided into three parts. On the left-hand side there is a panel for viewing files and sorting them between patterns "filename" or "file type". The panel is used to brows working files in the program and displays the path to the source of input data at the top of the box. Lower, the files identifier controls are placed – «`Filename`» and «`Filetype`» buttons. At the box bottom there are buttons «`Delete`» and «`Rename`» for manipulating files under the treatment.

The toolbar includes a set of buttons for operational control by the program modes (function tools are labeled by the first letter of the required action; for example, S means «`segmentation`»). When one aims the mouse pointer at a button, a hint will pop up.

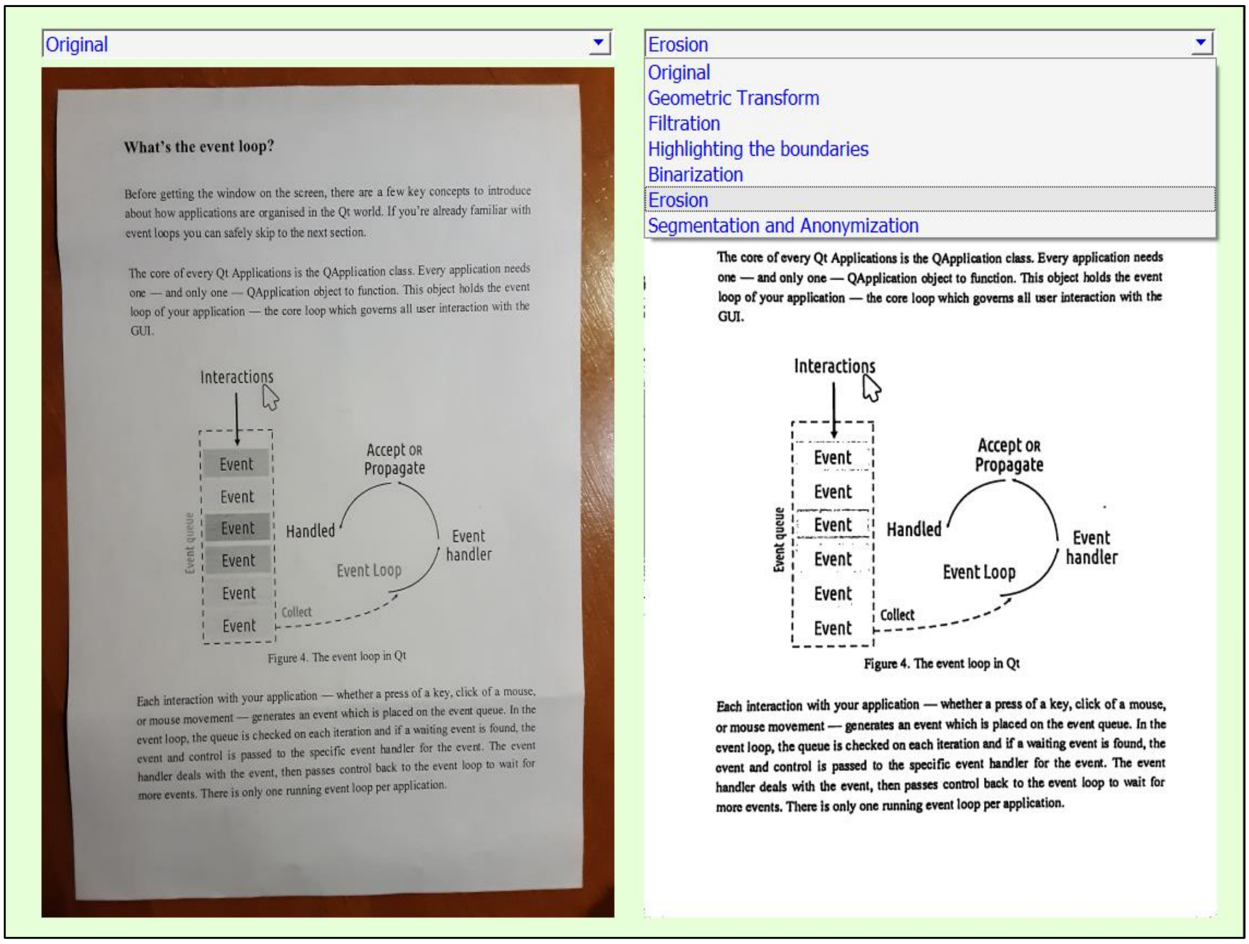
The master window locates two boxes (in center and towards right) for viewing and comparing pictures of the original and treated documents. The each provides a drop up menu to select a picture; at that, the second box assumes user's selection of the processing stage needed (see fig. 15).

Segmentation and anonymization. The program provides the ability of segmenting the treated text doc page to reduce redundancy; through segmentation, only the selected picture fragments can be treated and therefore be recognized. The segmentation routine is executed by the user in the pop-up «Segmentation Window» interactively using the mouse as it is shown in fig. 16, a.

Similarly, to maintain information confidentiality, anonymization of the necessary sections of the text is brought using the pop-up «Window» (fig. 16, b).

You can activate the windows `Segmentation` and `Anonymization` in the main program window (see fig. 14) by clicking the S or A buttons in the toolbar.

Saving results of the text recognition. The click on the toolbar OCR button, the pop-up window appears in the screen that will display the optical text recognition results with possibility to provide their saving. There is a small toolbar in that outcomes window (it can be seen in fig. 17) that allows you to increase / decrease the font size and, hence, the scale of text displaying). Two buttons more are present there used to save in the format `docx` or `pdf` the outcomes of text recognition.



What's the event loop?

Before getting the window on the screen, there are a few key concepts to introduce about how applications are organised in the Qt world. If you're already familiar with event loops you can safely skip to the next section.

The core of every Qt Applications is the QApplication class. Every application needs one — and only one — QApplication object to function. This object holds the event loop of your application — the core loop which governs all user interaction with the GUI.

Figure 4. The event loop in Qt

Each interaction with your application — whether a press of a key, click of a mouse, or mouse movement — generates an event which is placed on the event queue. In the event loop, the queue is checked on each iteration and if a waiting event is found, the event and control is passed to the specific event handler for the event. The event handler deals with the event, then passes control back to the event loop to wait for more events. There is only one running event loop per application.

The core of every Qt Applications is the QApplication class. Every application needs one — and only one — QApplication object to function. This object holds the event loop of your application — the core loop which governs all user interaction with the GUI.

Figure 4. The event loop in Qt

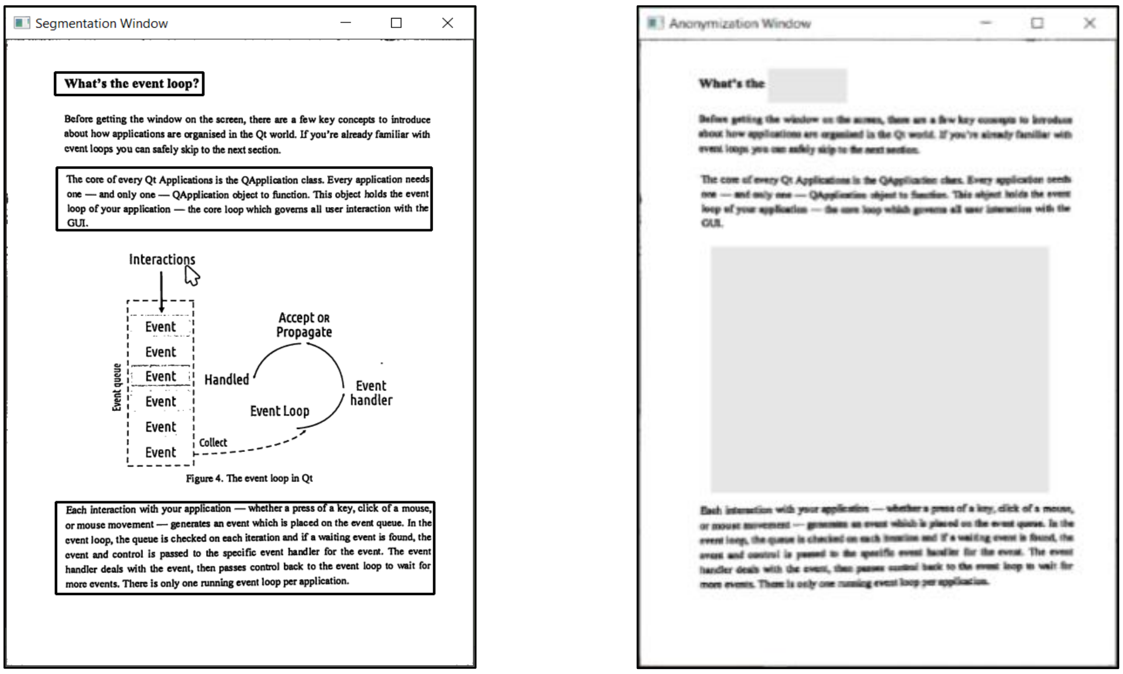
Each interaction with your application — whether a press of a key, click of a mouse, or mouse movement — generates an event which is placed on the event queue. In the event loop, the queue is checked on each iteration and if a waiting event is found, the event and control is passed to the specific event handler for the event. The event handler deals with the event, then passes control back to the event loop to wait for more events. There is only one running event loop per application.

a b

Fig. 15. Windows for viewing documents at various stages of processing: a − displaying source text (after «Original» mode selected); b – window of drop up menu to run process needed («Erosion» selected)

When one clicks the button wanted, the file of text recognition outcomes appears in the main window (see fig. 14) with the name of origin image, though with the chosen extension. For instance, for the input source file `image1_eng.jpg`, the treated text file is named `image1_eng.pdf` or `image1_eng.docx` if the icon pdf was pressed.

Face detection. If there is a photo of a human on the document page (e.g. a photo for person identification in a demographic application), the program «HQ Scanner» can apply the option of face detection. The «Face Detection» software module employs the classic Viola-Jones algorithm based on the Haar primitives. By default, the module is running continuously. Therefore, when you click on the button in the toolbar of the main window, the pop-up box with the recognition outcome «Face Detection Results» comes at once (Fig. 18).



Segmentation Window

What's the event loop?

Before getting the window on the screen, there are a few key concepts to introduce about how applications are organised in the Qt world. If you're already familiar with event loops you can safely skip to the next section.

The core of every Qt Applications is the QApplication class. Every application needs one — and only one — QApplication object to function. This object holds the event loop of your application — the core loop which governs all user interaction with the GUI.

Figure 4. The event loop in Qt

Each interaction with your application — whether a press of a key, click of a mouse, or mouse movement — generates an event which is placed on the event queue. In the event loop, the queue is checked on each iteration and if a waiting event is found, the event and control is passed to the specific event handler for the event. The event handler deals with the event, then passes control back to the event loop to wait for more events. There is only one running event loop per application.

Anonymization Window

What's the

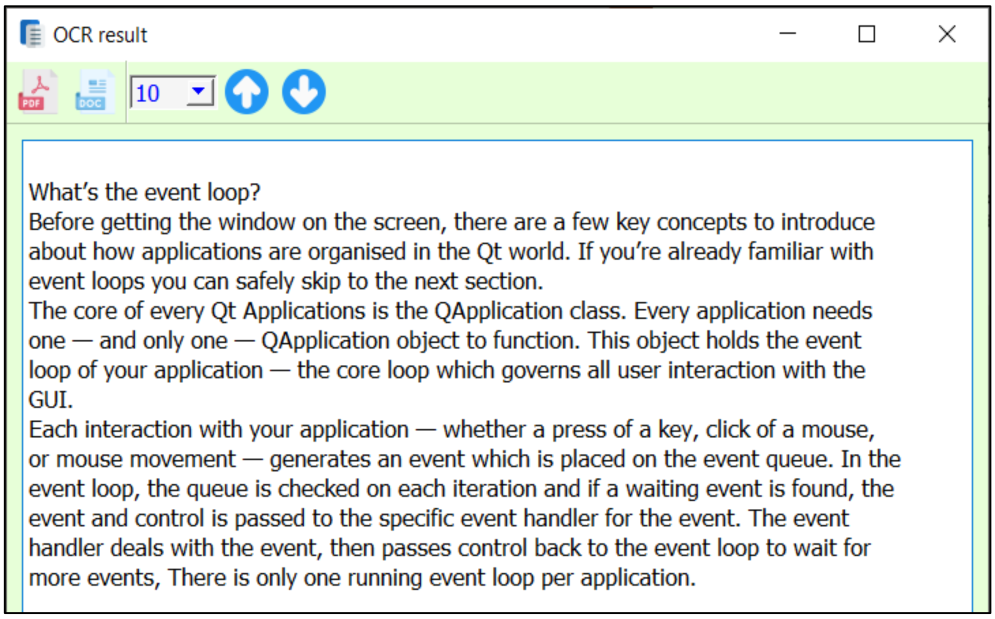
Before getting the window on the screen, there are a few key concepts to introduce about how applications are organised in the Qt world. If you're already familiar with event loops you can safely skip to the next section.

The core of every Qt Applications is the QApplication class. Every application needs one — and only one — QApplication object to function. This object holds the event loop of your application — the core loop which governs all user interaction with the GUI.

Each interaction with your application — whether a press of a key, click of a mouse, or mouse movement — generates an event which is placed on the event queue. In the event loop, the queue is checked on each iteration and if a waiting event is found, the event and control is passed to the specific event handler for the event. The event handler deals with the event, then passes control back to the event loop to wait for more events. There is only one running event loop per application.

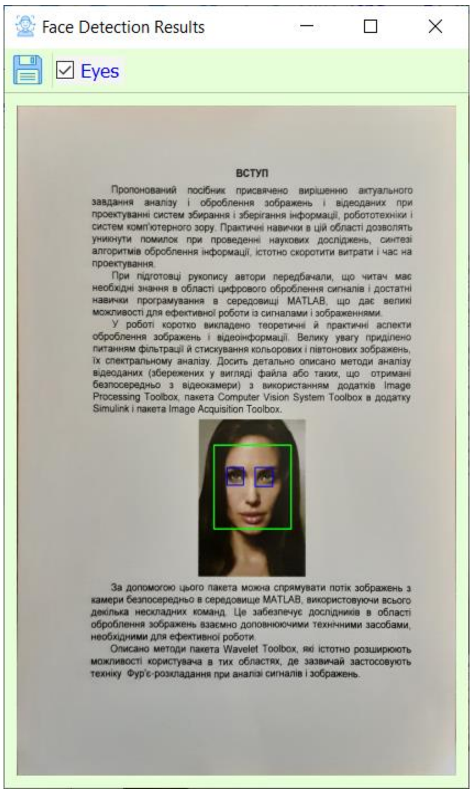
a b

Fig. 16. Pop-up windows to provide segmentation (a) and anonymization (b)



What's the event loop?
Before getting the window on the screen, there are a few key concepts to introduce about how applications are organised in the Qt world. If you're already familiar with event loops you can safely skip to the next section.
The core of every Qt Applications is the QApplication class. Every application needs one — and only one — QApplication object to function. This object holds the event loop of your application — the core loop which governs all user interaction with the GUI.
Each interaction with your application — whether a press of a key, click of a mouse, or mouse movement — generates an event which is placed on the event queue. In the event loop, the queue is checked on each iteration and if a waiting event is found, the event and control is passed to the specific event handler for the event. The event handler deals with the event, then passes control back to the event loop to wait for more events, There is only one running event loop per application.

Fig. 17. Pop-up window to display and save text recognition result



ВСТУП

Пропонований посібник присвячено вирішенню актуального завдання аналізу і оброблення зображень і відеоданих при проектуванні систем збирання і зберігання інформації, робототехніки і систем комп'ютерного зору. Практичні навички в цій області дозволять уникнути помилок при проведенні наукових досліджень, синтезі алгоритмів оброблення інформації, істотно скоротити витрати і час на проектування.

При підготовці рукопису автори передбачали, що читач має необхідні знання в області цифрового оброблення сигналів і достатні навички програмування в середовищі MATLAB, що дає великі можливості для ефективної роботи із сигналами і зображеннями.

У роботі коротко викладено теоретичні й практичні аспекти оброблення зображень і відеоінформації. Велику увагу приділено питанням фільтрації й стискування кольорових і півтонових зображень, їх спектральному аналізу. Досить детально описано методи аналізу відеоданих (збережених у вигляді файла або таких, що отримані безпосередньо з відеокамери) з використанням додатків Image Processing Toolbox, пакета Computer Vision System Toolbox в додатку Simulink і пакета Image Acquisition Toolbox.

За допомогою цього пакета можна спрямувати потік зображень з камери безпосередньо в середовище MATLAB, використовуючи всього декілька нескладних команд. Це забезпечує дослідників в області оброблення зображень взаємно доповнюючими технічними засобами, необхідними для ефективної роботи.

Описано методи пакета Wavelet Toolbox, які істотно розширюють можливості користувача в тих областях, де зазвичай застосовують техніку Фур'є-розкладання при аналізі сигналів і зображень.

Fig. 18. Results of face detection
(face and eyes detection)

The program can serve either face detection solely or face detection and, in addition, eye recognition. To terminate, click the icon 💾 and store the file.

Conclusion

A set of new algorithms aimed at increasing the performance quality of the optical text recognition systems is proposed. The study shows the high efficiency of the technical solutions proposed. Source data preprocessing allows you to achieve near 100 % fidelity over text recognition.

Based on the proposed algorithms, dedicated Python software modules were developed and tested using the OpenCV library. The originality of the developed "HQ Scanner" program consists in the extension of the ordinary functionality of the processing facility on account of added face detection module. The latter is needed for many demographic projects. According to the authors, the most original in this project are procedures for geometric image correction and adaptive image binarization algorithms.

The ultimate goal of the project is to create a cost-effective and easy-to-use OCR system for commercial use. The HQ Scanner program can be considered a perfectly acceptable prototype for solving this problem.

References (GOST 7.1:2006)

1. Gradient-Based Learning Applied to Document Recognition [Text] / Y. Lecun, et al. //Proceedings of the IEEE. −1998. − Vol. 86, no. 11. − P. 2278-2324. DOI: 10.1109/5.726791.

2. ImageNet Classification with Deep Convolutional Neural Networks [Text] / A. Krizhevsky, et al. // Communications of the ACM. – 2017. – Vol. 60, no. 6. – P. 84–90. DOI: 10.1145/3065386.

3. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification [Text] / Kaiming He, et al. // IEEE International Conference on Computer Vision (ICCV). – 2015. − P. 1026-1034. DOI: 10.1109/iccv.2015.123.

4. The ImageNet database is a project for the creation and maintenance of a massive database of annotated images, intended for the development and testing of image recognition and machine vision methods Available at: https://en.wikipedia.org/wiki/ImageNet. – 3.06.2021.

5. Krizhevsky, A. ImageNet classification with deep convolutional neural networks [Text] / A. Krizhevsky, I. Sutskever, E. Hinton // Advances in neural information processing systems. – 2015. − P. 1097–1105.

6. Steps Involved in Text Recognition and Recent Research in OCR; A Study [Text] / K. Karthick, K. B. Ravindrakumar, R. Francis, S. Ilankannan // International Journal of Recent Technology and Engineering (IJRTE). – 2019. – Vol. 8, iss.1. – P. 3095-3100. Retrieval Number A2670058119/19©BEIESP.

7. A Study on Optical Character Recognition Techniques [Text] / N. Sahu, M. Sonkusare // The International Journal of Computational Science, Information Technology and Control Engineering (IJCSITCE). – 2017. – Vol. 4, No. 1. – 14 p. DOI: 10.5121/ijcsitce.2017.4101.

8. Offline optical character recognition (OCR) method: An effective method for scanned documents [Text] / Mujibur Rahman Majumder et al. // 22nd International Conference on Computer and Information Technology (ICCIT) – 2019. – P. 1-5. DOI: 10.1109/ICCIT48885.2019.9038593.

9. Improved OCR quality for smart scanned document management system [Text] / Anh Phan Viet et al. / Journal of Science and Technique − Le Quy Don Technical University. – 2020. − No. 210. – P. 51-67.

10. Image to Text Conversion Using Tesseract [Text] / N. Pawar, Z. Shaikh, P. Shinde, Y. Warke // International Research Journal of Engineering and Technology (IRJET). – 2019 – Vol. 6 iss 02. – P. 516-519.

11. Scan.it − on Advances in Computing, Communication and Control (Text Recognition, Translation and Conversion [Text] / M. Acharya, P. Chouhan, A. Deshmukh // International Conference (ICAC3). – 2019. – P. 1-5. DOI: 10.1109/ICAC347590.2019. 9036849.

12. Jaume, G. A Dataset for Form Understanding in Noisy Scanned Documents [Text] / G. Jaume, H. Ke-

mal Ekenel, J. Thiran // International Conference on Document Analysis and Recognition Workshops (ICDARW). – 2019. – P. 1-6. DOI: 10.1109/ICDARW.2019.10029.

13. Chernyshova, Y. S. Two-Step CNN Framework for Text Line Recognition in Camera-Captured Images [Text] / Y. S. Chernyshova, A. V. Sheshkus and V. V. Arlazarov // IEEE Access. – 2020. – Vol. 8. – P. 32587-32600. DOI: 10.1109/ACCESS.2020.2974051.

14. An Effective Background Evaluation Method for Removing Shadows from Document Images [Text] / B. Wang, et al. // IEEE International Conference on Image Processing (ICIP). – 2019. – P. 3611-3615. DOI: 10.1109/ICIP.2019.8803486.

15. Caffe, a comprehensive open source machine learning platform [Electronic resource]. – Available at: http:// caffe.berkeleyvision.org/. – 12.04.2021.

16. Tensorflow, a comprehensive open source machine learning platform [Electronic resource]. – Available at: https://www. tensorflow.org/. – 12.04.2021.

17. Torch.ch, a comprehensive open source machine learning platform [Electronic resource]. – Available at: http://torch.ch/. – 12.04.2021.

18. Open source neural networks in C [Electronic resource]. – Available at: https://pjreddie.com/darknet/. – 12.04.2021.

19. Keras, a comprehensive open source machine learning platform [Electronic resource]. – Available at: https://keras.io/. – 18.05.2021.

20. OpenCV / dnn modules [Electronic resource]. –. Available at: https://github.com/opencv/opencv/ tree/master/modules/dnn. – 18.05.2021.

21. Hardware - high-performance Raspberry Pi computers and accessories [Electronic resource]. – Available at: https://www. raspberrypi.org. – 18.05.2021.

22. Python Developer's Guide [Electronic resource]. – Available at: http://python.org. – 18.05.2021.

23. OpenCV Tutorials − Image Processing (imgproc module) [Electronic resource]. – Available at: https://opencv.org/ – 18.05.2021.

24. Abbyy Finereader (Scanner with artificial intelligence for digitizing to PDF and OCR) [Electronic resource]. – Available at: https://www.abbyy.com/ en/finereader/ –18.05.2021.

25. OCRopus − tesseract based OCR system for text recognition [Electronic resource]. – Available at: https://en.wikipedia.org/wiki/Cognitive_Technologies. – 18.05.2021.

26. Optical character recognition (OCR) [Electronic resource]. – Available at: https://en.wikipedia. org/wiki/OCRopus. – 18.05.2021.

27. Tesseract − ocr / Tesseract [Electronic resource]. – Available at: https://github.com/tesseract-ocr/tesseract.– 18.05.2021.

28. Python-tesseract − Optical character recognition (OCR) tool for Python [Electronic resource]. – Available at: https://pypi.org/project/pytesseract/. – 18.05.2021.

References (BSI)

1. Lecun, Y., et al. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998, vol. 86, no. 11, pp. 2278–2324. DOI: 10.1109/5.726791.

2. Krizhevsky, A., et al. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 2017, vol. 60, no. 6, pp. 84–90. DOI: 10.1145/3065386.

3. He, Kaiming., et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026-1034. DOI: 10.1109/iccv.2015.123.

4. The ImageNet database is a project for the creation and maintenance of a massive database of annotated images, intended for the development and testing of image recognition and machine vision methods. Available at: https://ru.wikipedia.org/ wiki/ImageNet. (accessed 3.06.2021).

5. Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 2012, pp. 1097-1105.

6. Karthick, K., Ravindrakumar, K. B., Francis, R., Ilankannan, S. Steps Involved in Text Recognition and Recent Research in OCR; A Study. *International Journal of Recent Technology and Engineering (IJRTE)*, 2019, vol. 8, iss.1, pp. 3095-3100. Retrieval Number A2670058119/19©BEIESP.

7. Sahu, N., Sonkusare, M. A Study on Optical Character Recognition-Techniques. *The International Journal of Computational Science, Information Technology and Control Engineering* (IJCSITCE), 2017, vol. 4, no. 1. 14 p. DOI: 10.5121/ijcsitce.2017.4101.

8. Mujibur Rahman Majumder et al. Offline optical character recognition (OCR) method: An effective method for scanned documents. *22nd International Conference on Computer and Information Technology (ICCIT)*, 2019, pp. 1-15. DOI: 10.1109/ICCIT48885.2019. 9038593.

9. Anh Phan Viet et al. Improved OCR quality for smart scanned document management system. *Journal of Science and Technique − Le Quy Don Technical University*, 2020, no. 210, pp. 51-67.

10. Pawar, N., Shaikh, Z., Shinde, P., Warke, Y., Image to Text Conversion Using Tesseract, *International Research Journal of Engineering and Technology (IRJET)*, 2019, vol. 6, iss. 2, pp. 516-519.

11. Acharya, M., Chouhan, P., Deshmukh, A. Scan.it - on Advances in Computing, Communication and Control (Text Recognition, Translation and Conversion, *International Conference (ICAC3),* 2019, pp. 1-5. DOI: 10.1109/ICAC347590.2019. 9036849.

12. Jaume, G., Kemal Ekenel, H, Thiran, J. A Dataset for Form Understanding in Noisy Scanned Documents. *International Conference on Document Analysis and Recognition Workshops (ICDARW),* 2019, pp. 1-6, DOI: 10.1109/ICDARW.2019.10029.

13. Chernyshova, Y. S., Sheshkus, A. V., Arlazarov, V. V. Two-Step CNN Framework for Text Line Recognition in Camera-Captured Images. *IEEE Access,* 2020, vol. 8, pp. 32587-32600. DOI: 10.1109/ACCESS.2020.2974051.

14. Wang, B., et al. An Effective Background Evaluation Method for Removing Shadows from Document Images. *IEEE International Conference on Image Processing (ICIP),* 2019, pp. 3611-3615. DOI: 10.1109/ICIP.2019.8803486.

15. *Caffe, a comprehensive open source machine learning platform.* Available at: http:// caffe.berkeleyvision.org/ (accessed 12.04.2021).

16. *Tensorflow, a comprehensive open source machine learning platform*. Available at: https://www. tensorflow.org/ (accessed 12.04.2021).

17. *Torch.ch, a comprehensive open source machine learning platform*. Available at: http://torch.ch/ (accessed 12.04.2021).

18. *Open source neural networks in C.* Available at: https://pjreddie.com/darknet/ (accessed 12.04.2021).

19. *Keras, a comprehensive open source machine learning platform*. Available at: https://keras.io/ (accessed 18.05.2021).

20. *OpenCV / dnn modules.* Available at: https://github.com/opencv/opencv/tree/master/modules/ dnn (accessed 18.05.2021).

21. *Hardware - high-performance Raspberry Pi computers and accessories.* Available at: https://www. raspberrypi.org (accessed 18.05.2021).

22. *Python Developer's Guide.* Available at: http://python.org (accessed 18.05.2021).

23. *OpenCV Tutorials − Image Processing (imgproc module).* Available at: https://opencv.org/ (accessed 18.05.2021).

24. *Abbyy Finereader (Scanner with artificial intelligence for digitizing to PDF and OCR)*. Available at: https://www.abbyy.com/ru/finereader/ (accessed 18.05.2021).

25. *OCRopus − tesseract based OCR system for text recognition*. Available at: https://ru.wikipedia.org/ wiki/Cognitive_Technologies (accessed 18.05.2021).

26. *Optical Character Recognition (OCR).* Available at: https://en.wikipedia.org/wiki/OCRopus (accessed 18.05.2021).

27. *Tesseract − ocr / Tesseract.* Available at: https://github.com/tesseract-ocr/tesseract (accessed 18.05.2021).

28. *Python-tesseract − Optical character recognition (OCR) tool for Python*. Available at: https: //pypi.org/project/ pytesseract/(accessed 18.05.2021).

Надійшла до редакції 25.05.2021 розглянута на редколегії 26.11.2021

ПОПЕРЕДНЯ ОБРОБКА ДАНИХ ДЛЯ ПІДВИЩЕННЯ ЯКОСТІ СИСТЕМ ОПТИЧНОГО РОЗПІЗНАВАННЯ ТЕКСТУ

К. Ю. Дергачов, Л. О. Краснов, В. О Білозерський., А. Я. Зимовин

Предметом вивчення в статті є формулювання сучасної концепції підвищення якості ра-боти систем оптичного розпізнавання текстів шляхом використання набору різноманітних алгоритмів для попередньої обробки зображень документів на розсуд користувача. **Мета роботи** − синтезувати алгоритми, що компенсують зовнішні негативні впливи (несприятливий геометричний фактор, погані умови освітлення при фотографуванні, вплив шумів і ін.). Використовувані методи мають на увазі певну послідовність етапів попередньої обробки даних: геометричне перетворення вихідних зображень, їх обробку набором різних фільтрів, еквалізацію зображень без збільшення рівня шуму для підвищення контрастності знімків, бінаризація зображень з адаптивними порогами перетворення для усунення впливу нерівномірного засвічення фото. Отримані наступні **результати**. Створено пакет алгоритмів попередньої обробки фотознімків документації, в який для підвищення функціональних можливостей при ідентифікації даних також вбудований алгоритм детектування осіб (face detection), призначений для подальшого їх розпізнавання (face recognition). Передбачено ряд сервісних процедур, що забезпечують зручність обробки даних і їх інформаційний захист. Зокрема, запропонована інтерактивна процедура сегментації тексту з можливістю анонімізації окремих його фрагментів. Це сприяє збереженню конфіденційності оброблюваних документів. Детально описано структуру перерахованих алгоритмів і досліджена стійкість їх роботи в різних умовах. За результатами проведених досліджень розроблено програмне забезпечення для розпізнавання текстів, що базується на програмі оптичного розпізнавання символів (OCR) Tesseract версії 4.0. Програма отримала назву «HQ Scanner», вона написана на мові Python з використанням сучасних ресурсів бібліотеки OpenCV. Програмно реалізована оригінальна методика оцінки ефективності роботи алгоритмів за критерієм максимуму ймовірності правильного розпізнавання текстів. Наводиться велика кількість прикладів роботи системи і результати тестування програмного забезпечення. **Висновки**. Результати проведених досліджень можуть бути взяті за основу при розробці програмного забезпечення для створення економічних і зручних у використанні систем оптичного розпізнавання текстів, призначених для комерційного використання.

Ключові слова: оптичне розпізнавання символів (OCR); геометричні перетворення вихідних зображень; алгоритми фільтрації; еквалізація і бінаризація фотознімків; алгоритм виявлення осіб (face detection); ймовірність правильного розпізнавання тексту; сегментація текстів і анонімізація їх окремих фрагментів.

Radioelectronic and Computer Systems, 2021, no. 4(100)
ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ ДЛЯ ПОВЫШЕНИЯ КАЧЕСТВА СИСТЕМ ОПТИЧЕСКОГО РАСПОЗНАВАНИЯ ТЕКСТА

К. Ю. Дергачёв, Л. А. Краснов, В. А. Билозерский, А. Я. Зимовин

Предметом изучения в статье является формулировка современной концепции повышения качества работы систем оптического распознавания текстов путем использования набора разнообразных алгоритмов для предварительной обработки изображений документов по усмотрению пользователя. **Цель работы** – синтезировать алгоритмы, компенсирующие внешние негативные воздействия (неблагоприятный геометрический фактор, плохие условия освещения при фотографировании, влияние шумов и пр.). Используемые методы подразумевают определенную последовательность этапов предварительной обработки данных: геометрическое преобразование исходных изображений, их обработку набором различных фильтров, эквализацию изображений без увеличения уровня шума для повышения контрастности снимков, бинаризацию изображений с адаптивными порогами преобразования для устранения влияния неравномерной засветки фото. Получены следующие **результаты.** Создан пакет алгоритмов предварительной обработки фотоснимков документации, в который для повышения функциональных возможностей при идентификации данных также встроен алгоритм детектирования лиц (face detection), предназначенный для дальнейшего их распознавания (face recognition). Предусмотрен ряд сервисных процедур, обеспечивающих удобство обработки данных и их информационную защиту. В частности, предложена интерактивная процедура сегментации текста с возможностью анонимизации отдельных его фрагментов. Это способствует сохранению конфиденциальности обрабатываемых документов. Подробно описана структура перечисленных алгоритмов и исследована устойчивость их работы в различных условиях. По результатам проведенных исследований разработано программное обеспечение для распознавания текстов, базирующееся на программе оптического распознавания символов (OCR) Tesseract версии 4.0. Программа получила название «HQ Scanner», она написана на языке Python с использованием современных ресурсов библиотеки OpenCV. Программно реализована оригинальная методика оценки эффективности работы алгоритмов по критерию максимума вероятности правильного распознавания текстов. Приводится большое количество примеров работы системы и результаты тестирования программного обеспечения. **Выводы**. Результаты проведенных исследований могут быть взяты за основу при разработке программного обеспечения для создания экономичных и удобных в использовании систем оптического распознавания текстов, предназначенных для коммерческого использования.

Ключевые слова: оптическое распознавание символов (OCR); геометрические преобразования исходных изображений; алгоритмы фильтрации; эквализация и бинаризация фотоснимков; алгоритм выявления лиц (face detection); вероятность правильного распознавания текста; сегментация текстов и анонимизация их отдельных фрагментов.

Дергачов Костянтин Юрійович − канд. техн. наук, старш. наук. співроб., зав. каф. систем управління літальних апаратів, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Краснов Леонід Олександрович – канд. техн. наук, старш. наук. співроб., доцент каф. управління літальних апаратів, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Білозерський Владислав Олександрович – магістр каф. систем управління літальних апаратів, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Зимовин Анатолій Якович − канд. техн. наук, проф., проф. каф. управління літальних апаратів, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Konstantin Dergachov − Candidate of Technical Science, Senior Researcher, Head of the Department «Aircraft Control Systems», National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine, e-mail: k.dergachov@khai.edu, ORCID: 0000-0002-6939-3100.

Leonid Krasnov – Candidate of Technical Science, Senior Researcher, Assistant Professor of Department «Aircraft Control Systems», National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine, e-mail: leonid.krasnov.1947@gmail.com, ORCID: 0000-0003-2607-8423.

Vladislav Bilozerskyi – master student of Department «Aircraft Control Systems», National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine,
e mail: b1llays123@gmail.com, ORCID: 0000-0002-5503-3163.

Anatoly Zymovin − Candidate of Technical Science, Professor of Department «Aircraft Control Systems», National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine,
e-mail: zim301g@gmail.com, ORCID: 0000-0001-8580-2317.