Automated Document Analysis for Quick Personal Health Record Creation

Nataliya Boyko [0000-0002-6962-9363], Olena Pylypiv [0000-0001-7970-8498],

Yulia Peleshchak [0000-0003-4284-9961], Yurko Kryvenchuk[0000-0002-2504-5833],

Jaime Campos[0000-0001-7048-8089]

Lviv Polytechnic National University, Lviv79013, Ukraine
Linnaeus University, Växjö, Sweden
nataliya.i.boyko@lpnu.ua
my-mylo@ukr.net
yulia.peleshchak@gmail.com
yurkokryvenchuk@gmail.com
jaime.campos@lnu.se

Abstract. We propose our own optimization methods to speed up the work and compare different approaches for high-quality implementation of the idea. The subproblem of the first stage in image processing is a filtration. As a result of the research, can distinguish the following main types of noises: additive and impulse. The impulse - is associated with losses in the transmissions through a communication channels. We can remove noise using the Gaussian blur, which can be used from the OpenCV library. The purpose of the first stage is to identify the edges, because with the development of technology, it simplifies the operation of data processing algorithms, and thus the storage and processing of information. The second step is to find out the contours in the image that represent the document, which we want to check. The last step lays in taking the four points, representing the outline of the medical document (found in the second step) and appling a perspective transform to get a top-down image with an angle of 90 degrees, because we need a view at a right angle, since it is more convenient to work with it.

Keywords: Methods, Technologies, OpenCV library, The Gaussian blur, Algorithms such as Canny, Algorithms such as Derich.

1 Introduction

Nowadays, technologies can accelerate and improve the field of healthcare delivery as medical information has to be received very quickly because it has an impact on human life, so this process should be optimized as fast as possible. In the case of a non-informative system, the patient is put at great risk because doctors do not know if there are any warnings about the treatment methods.

The most convenient way is to use the patient's documents to read information from him and make the right conclusions about the emergency or treatment scheme. To solve this problem, it was proposed to identify the patient's document - id card in real-time, then read and process information from it. Obviously, in order to solve this problem, firstly we must process our image and prepare it to make recognition and reading as accurate and fast as possible.

Development stages of the program:

- Step 1: edge detection, to highlight and focus on the research object.
- Step 2: usage of the image's edges to find the contour, representing the scanned document.
- Step 3: view perspective application to get the look of the document from the top-down.

Please note that the first paragraph of a section or subsection is not indented. The first paragraphs that follows a table, figure, equation etc. does not have an indent, either.

Subsequent paragraphs, however, are indented.

2 Review of the Literature

In the present, technologies of computer vision are actively developing, with their help, we can solve problems more effectively, one of which is recognition. As a result of active development, developers receive a large number of libraries to solve problems. Actually, in this article we face the task of determining the performance of these libraries. The best way to get detailed information about a particular system, library, or API is to get acquainted with the documentation that we will actually use for the research in this article [1, 2]. Works [6, 17] focus on the theoretical aspects of building a stable system for recognition.

The researchers [3, 5, 10-11] describe the actual methods and technologies for all stages of the development of the recognition system, since in the field of recognition, a huge number of unique solutions have been developed. In any system, there is a promising issue of its performance, and especially this applies to recognition systems, so the authors [4-9] investigate the speed of the operation of recognition methods.

3 Materials and Methods

Among the variety of methods and algorithms, that are already widely used in working with images, only highlighted and used are those, that allow us to maximally optimize the program's performance in relation to our needs. We propose our own optimization methods to speed up the work and compare different approaches for high-quality implementation of the idea [12-16].

Most images are exposed to various types of noise in the process of transmitting them through communication channels, as well as at the formation stage. Therefore, sub-

problem of the first stage in image processing is a filtration. The presence of noises in the image may lead to inaccuracies and distortions at the segmentation and recognition stage. For example, the system can perceive noise for individual objects, which may adversely affect further research. As a result of the research, can distinguish the following main types of noises: additive and impulse. The first one appears in digital image forming devices. While the impulse - is associated with losses in the transmissions through a communication channels. Since the student is bringing his document to the camera, we can remove noise using the Gaussian blur, which can be used from the OpenCV library. The purpose of the first stage is to identify the edges, because with the development of technology, it simplifies the operation of data processing algorithms, and thus the storage and processing of information. For this task, algorithms such as Canny or Deriche have already been implemented, but they are mostly complementary or improvement in certain cases. Usage of the Deriche makes sense in cases, where the processed image is noisy or requires a lot of smoothing. However, this does not apply to our document, since it does not contain complicated drawings. Therefore, we will use the Canny algorithm for optimal search [18].

4 Experiment

For our study, the authors chose a student ID, because the doctor's and patient's certificates were not available

To start, we upload the image (Student ticket, figure 1), thus translate it into the usual form for software development - the matrix of tuples with RGB values [R G B](fig. 2).

image = cv2.imread(args["image"])

Fig. 1. The image to work with

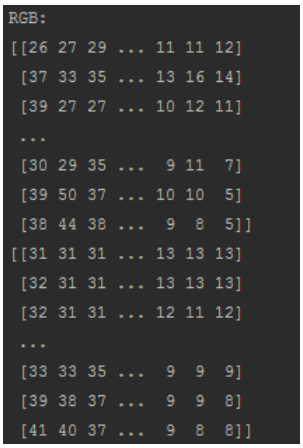


Fig. 2. RGB Student Ticket Matrix

To evaluate and analyze the structure of the objects in the image and their further processing, firstly we must find them, identify their edges. To detect edges we use the Canny algorithm to highlight the location of the main edges, ignoring any false edges caused by noise.

Because of that, the first thing the edge detector does - it uses a Gaussian blur to smooth the input image and remove the noise. Gauss filter averages the Gaussian pixel around the point by function:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)} \tag{1}$$

Then the first derivative operator is being applied to the smoothed image for highlight areas of the image, using the highest first spatial derivatives.

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

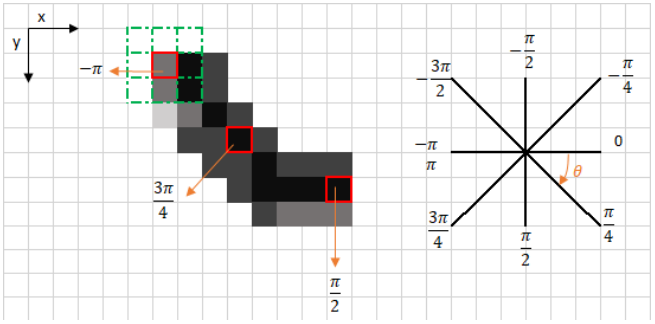
gray = cv2.GaussianBlur(gray, (5, 5), 0)

Fig. 3. Output images (left) - image size with Gaussian filter (sigma = 1 and kernel size 5x5)

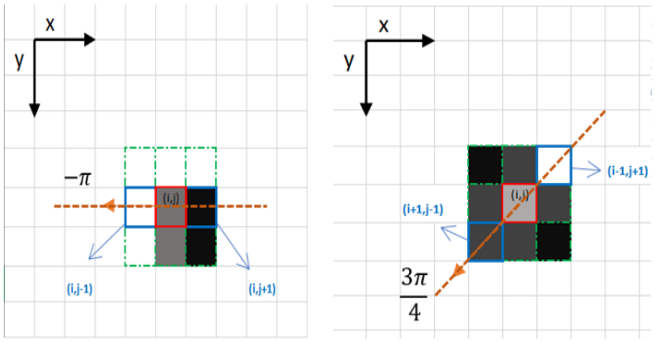
The algorithm treats the significant local changes occurring in the image intensity (e.g. pixel value), so the edge detector takes the image in grayscale as an input and creates an image that shows the location of the gap breakdown as output (namely edges). The second step is to calculate the gradient, which determines the intensity and direction of the edge, corresponding to the change in the intensity of the pixels.The easiest way to determine them is to apply filters that cover the intensity change in a horizontal and vertical direction, respectively x and y. When the image is smoothed, derivatives Ix, Iy and suitably x, y are calculated. Then the G value and the slope of the gradient θ are calculated as follows:

$$|G| = \sqrt{I_x^2 + I_y^2},$$
$$\theta(x, y) = \arctan(\frac{I_y}{I_x}) \quad (2)$$

To make all the edges of the same thickness it is necessary to soften the thick. This will be our third step - non-maximum suppression. The algorithm passes through all points of the gradient intensity matrix, that is found in the previous step, and finds pixels with the maximum value in the boundary directions. For an example, consider pixels of the image closere:



In the image, we can see the intensity pixels of the processed gradient matrix and the corresponding direction of the edge(that is represented by an orange arrow). For example, the red field of the upper left corner corresponds to the direction of the edge with the angle -pi radians (+/- 180 degrees).



The purpose of the algorithm is to check if pixels in one direction are more or less intense than the processed pixels. In the above example, the pixel (i, j) is being processed, and the pixels in the same direction are highlighted in blue (i, j-1) and (i, j + 1). If one of these two pixels is more intense, then only more intense became stored. The pixel (i, j-1) seems to be more intense because it is white (value 255). Consequently, the intensity of the current pixel (i, j) is set to 0. If there are no pixels in the direction with more intense values, the current pixel value is saved. In the case of a diagonal line, the most intense pixel in this direction is the pixel (i-1, j + 1). So each pixel has 2 main criteria: 1) Destination edge in radians. 2) The intensity of the pixels (from 0 to 255).

Based on these input parameters, the algorithm of non-maximum suppression has the following steps: it creates a matrix filled with zeros, of the same size as the output matrix of the gradient intensity; determines the direction of the edge based on the angle matrix; checks whether the pixel in the same direction is more intense than the pixel that is being processed;

Fig. 4. Blur image (left) - gradient intensity (right)

As the result there is the same image with more subtle edges. However, some pixels seem brighter than others. We will solve this by using the last two steps - double threshold and track the edge with hysteresis. So the fourth task is aimed at identifying 3 types of pixels: strong, weak and irrelevant:

Strong pixels - pixels that have such high intensity that we are sure that they contribute to the final edge. Weak pixels - pixels that have an intensity value that is not enough to be considered powerful, but not small enough to be considered irrelevant to determine the edges. Other pixels are considered not relevant for the edge.

Fig. 5. The result of non-maximum suppression

The last step is to track the edge with hysteresis. Based on the results of the threshold, the hysteresis consists of converting the weak pixels into strong ones, if at least one of the pixels around the processed one is strong.

Fig. 6. The result of the hysteresis process

Fig. 7. The result of Canny algorithm

The second step is to find out the contours in the image that represent the document, which we want to check. We need to outline the image, so we assume it is the largest contour with exactly four points.

Fig. 8. Picture of contour points

This is also a rather safe assumption - the program assumes that a student ticket is the main focus of our image. And it can also be assumed that it has four edges. Let's start with finding contours in our frame. When choosing the largest contour we sort them by area and save only the largest ones. This allows to view only the largest contours, discarding the rest. Then in a loop we begin to cross the points of the potential contour. If the contour has four points then we assume that they have reached the goal - the contour is found. And again, this is a pretty safe assumption. The program will assume that a student card is the main focus of an image that has a rectangular shape and thus has four different edges.

```
# loop over the contours
for c in cnts:
    # approximate the contour
    peri = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, 0.02 * peri, True)

    # if our approximated contour has four points, then we
    # can assume that we have found our screen
    if len(approx) == 4:
        screenCnt = approx
        break
```

```
Countours:
[[[ 52  16]]

 [[ 37 474]]

 [[321 482]]

 [[335  29]]]
```

Fig. 9. The array of the largest found contour (already sorted)

Fig. 10. The result of the search for the contours of a student ticket in the image

5 Results

The last step lays in taking the four points, representing the outline of the document (found in the second step) and appling a perspective transform to get a top-down image with an angle of 90 degrees, because we need a view at a right angle, since it is more convenient to work with it.

It is very important for us to sequentially organize the contour points (sequentially we will consider clockwise sorting: the upper left (A), upper right (B), bottom right (C) and bottom left (D) point) obtained in the previous step. In fact, the order itself can be arbitrary if it is consistent throughout the implementation. So for this we will sort our points.

```
deforder_points(pts):
        rect = np.zeros((4, 2), dtype = "float32")

    s = pts.sum(axis = 1)
    rect[0] = pts[np.argmin(s)]
    rect[2] = pts[np.argmax(s)]
    diff = np.diff(pts, axis = 1)
    rect[1] = pts[np.argmin(diff)]
    rect[3] = pts[np.argmax(diff)]
    return rect
```

```
Ordered points:
[[104.  32.]
 [670.  58.]
 [642. 964.]
 [ 74. 948.]]
```

Fig. 11. The result of sorting points

This function takes a single argument pts, which is a list of four points (found in the previous step) that specifies the coordinates (x, y) of each rectangle point. Then we find the upper left point, which has the smallest x + y amount and lower right the point that will have the largest x + y sum. Of course, now we will have to find the upper right and lower left points. Here we will take the difference between the points (that is x - y). The coordinates associated with the smallest difference will be the upper right point, while the coordinates with the greatest difference will be lower-left points.

But the found amounts or differences may coincide, so to solve this problem, we can improve the function for sequential ordering of the points.

```
deforder_points(pts):
xSorted = pts[np.argsort(pts[:, 0]), :]
 leftMost = xSorted[:2, :]
rightMost = xSorted[2:, :]
leftMost = leftMost[np.argsort(leftMost[:, 1]), :]
(tl, bl) = leftMost
```

```
D = dist.cdist(tl[np.newaxis], rightMost, "euclidean")[0]
(br, tr) = rightMost[np.argsort(D)[::-1], :]
return np.array([tl, tr, br, bl], dtype="float32")
```

This function also accepts a single argument pts, which is a list of four points that indicate the coordinates (x, y) of each point of the rectangle. But here we first sort our points. After that you need to trim the array to two parts, which will include the corresponding two left (upper and lower) and two right points. By sorting an array of left coordinates, we can find the left upper and lower points. We use the upper left point as the basis, apply the Pythagorean theorem to it, since we can divide our rectangular contour into rectangular triangles, and find Euclidean distances between it and right points.

Fig. 12. Image of Euclidean distances. B is the distance between the left and right upper points, C is the distance between the left upper and lower right points (C > D)

So we can find the right lower point, since the distance to it will be the largest (as the hypotenuse of a rectangular triangle). So we can find the upper and lower right coordinates.

```
Ordered points:
[[104.  32.]
 [670.  58.]
 [642. 964.]
 [ 74. 948.]]
```

Fig. 13. The result of sorting points

Now we need to determine the size of our new deformed image. Define the width - the greatest distance between the right lower and left lower x-coordinates or the upper right and upper left x coordinates.

```
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))
```

Similarly, we determine the height - the maximum distance between the upper right and bottom right y coordinates or the upper left and lower left y coordinates.

```
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))
```

```
dst = np.array([
    [0, 0],
    [maxWidth - 1, 0],
    [maxWidth - 1, maxHeight - 1],
    [0, maxHeight - 1]], dtype = "float32")
```

Now we define 4 points representing our top-down view. Then create a list in which the first entry (0, 0) corresponds to the upper left corner. The second record (maxWidth - 1, 0) accordingly corresponds to the upper right corner. Then we have (maxWidth - 1, maxHeight - 1), which is the bottom right corner. Finally, we have (0, maxHeight - 1) that is below the left corner. These points are determined in a sequential order of representation - and will allow us to get the image of the image from the top down.

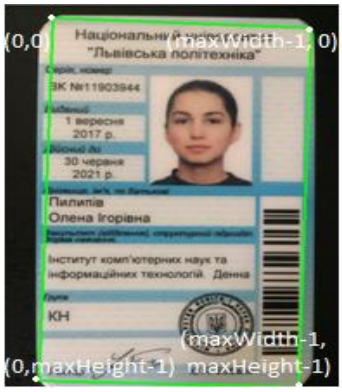


Fig. 14. The image from the top down

Let’s find the transformation matrix using cv2.warpPerspective.

```
[[ 9.87371065e-01  3.23374803e-02 -1.03721390e+02]
 [-4.58648365e-02  9.98442210e-01 -2.71802077e+01]
 [-1.94612654e-05  2.61318873e-06  1.00000000e+00]]
```

Fig. 15. Transformation Matrix

After that we pass the image to the transformation matrix M along with the width and height of our source image into the function cv2. warpPerspective, which returns a view of the image from the top to the bottom. The warpPerspective function converts the original image using the specified matrix:

$$dst(x, y) = src(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}) \quad (3)$$

We could use the function cv2.warpAffine, which takes the transformation matrix 2x3, but the warpPerspective is best for us, because it translates the transformation of straight lines into straight lines.

To get a black-white image, we convert it to gray and apply an adaptive threshold value.

```
[[ 45  41  55 ... 135  85  41]
 [ 40  79 125 ... 155 120  70]
 [ 93 138 173 ... 157 136  90]
 ...
 [ 57 136 248 ... 122 120 120]
 [ 62 101 190 ... 115 125 114]
 [ 60  55 106 ... 117 101  62]]
```

Fig. 16. An array of tuples RGB without the usage of the adaptive threshold value

```
[[  0   0   0 ... 255   0   0]
 [  0   0 255 ... 255 255   0]
 [  0 255 255 ... 255 255   0]
 ...
 [  0   0 255 ... 255 255 255]
 [  0   0 255 ... 255 255 255]
 [  0   0   0 ... 255 255   0]]
```

Fig. 17. An array of tuples RGB with the usage of the adaptive threshold value

Fig. 18. The result of image processing after the above three steps

6 Conclusions

In this work, a program implementation for processing a patient's document was developed. The methods and algorithms that are best suited and optimized for this task are covered. We propose our own sorting methods to exclude program bugs and show examples of implementation. Algorithms and methods for processing medical images can be used in the same way.

References

[1] Zhang, C., Murayama, Y.: Testing local spatial autocorrelation using, vol. 14, pp. 681–692, Intern. J. of Geogr. Inform. Science (2000).

[2] Estivill-Castro, V., Lee, I.: Amoeba: Hierarchical clustering based on spatial proximity using Delaunay diagram, 9th Intern. Symp. on spatial data handling, pp. 26–41 Beijing, China (2000).

[3] Kang, H.-Y., Lim, B.-J., Li, K.-J.: P2P Spatial query processing by Delaunay triangulation, Lecture notes in computer science, vol. 3428, pp. 136–150, Springer/Heidelberg (2005).

[4] Boehm, C., Kailing, K., Kriegel, H., Kroeger , P.: Density connected clus-tering with local subspace preferences, IEEE Computer Society, Proc. of the 4th IEEE Intern. conf. on data mining, pp. 27–34, Los Alamitos (2004).

[5] Boehm, C., Kailing, K., Kriegel, H., Kroeger, P. : Density connected clus-tering with local subspace preferences" IEEE Computer Society, Proc. of the 4th IEEE Intern. conf. on data mining, pp. 27–34, Los Alamitos (2004).

[6] Harel, D., Koren, Y. :Clustering spatial data using random walks, Proc. of the 7th ACM SIGKDD Intern. conf. on knowledge discovery and data mining, pp. 281–286, San Francisco, California (2000).

[7] Tung, A.K., Hou, J., Han, J. :Spatial clustering in the presence of obstacles, The 17th Intern. conf. on data engineering (ICDE'01), pp. 359–367, Heidelberg (2001).

[8] Veres O., Shakhovska N.: Elements of the formal model big date, The 11th Intern. conf. Perspective Technologies and Methods in MEMS Design (MEMSTEH), pp. 81-83, Polyana (2015).

[9] Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic sub-space clustering of high dimensional data, vol. 11(1), pp. 5–33, Data mining knowledge discovery (2005).

[10] Ankerst, M., Ester, M., Kriegel, H.-P.: Towards an effective cooperation of the user and the computer for classification, Proc. of the 6th ACM SIGKDD Intern. conf. on knowledge discovery and data mining, pp. 179–188, Boston, Massachusetts, USA (2000).

[11] Peuquet, D.J. : Representations of space and time, N. Y.: Guilford Press (2002).

[12] Guo, D., Peuquet, D.J., Gahegan, M.: ICEAGE: Interactive clustering and exploration of large and high-dimensional geodata, vol. 3, N. 7, pp. 229–253, Geoinfor-matica (2003).

[13] Mochurad, L., Solomiia , A.: Optimizing the Computational Modeling of Modern Electronic Optical Systems.Advances in Intelligent Systems and Computing, vol 1020. Springer, pp 597-608 (2019)

[14] Tung, A.K, Hou, J., Han, J.: Spatial clustering in the presence of obstacles, The 17th Intern. conf. on data engineering (ICDE'01), pp. 359–367, Heidelberg (2001).

[15] Boyko, N. A look trough methods of intellectual data analysis and their applying in informational systems, In Scientific and Technical Conference "Computer Sciences and Information Technologies (CSIT), pp. 183-185, IEEE, XIth International (2016).

[16] Boyko, N., et. al. Use of machine learning in the forecast of clinical consequences of cancer diseases, In 7th Mediterranean Conference on Embedded Computing, pp. 531-536, IEEE MECO'2018 (2018).

[17] Boyko, N.: Advanced technologies of big data research in distributed information systems, Radio Electronics, Computer Science, Control. № 4, pp. 66-77, Zaporizhzhya: Zaporizhzhya National Technical University (2016).

[18] Mochurad, L.I.: Extending the sphere of the application method of model order reduction during calculating the electrostatic field. Mathematical modeling and computing, Vol. X, № X, 10 p. (2019)