Real-Time Ukrainian Text Recognition and Voicing

Kateryna Tymoshenko[a], Victoria Vysotska[a], Oksana Kovtun[b], Roman Holoshchuk[a] and Svitlana Holoshchuk[a]

[a] *Lviv Polytechnic National University, S. Bandera Street, 12, Lviv, 79013, Ukraine*
[b] *Vasyl' Stus Donetsk National University, 600-richchia Street, 21, Vinnytsia, 21021, Ukraine*

Abstract
The main application task, solving which the project aims to, is to help people with visual impairments and teach the correct pronunciation of words to people learning a Ukrainian language as a foreign language. This problem is solved by developing software that will recognise text in video mode. As a result, when you tap the screen, you will hear how the selected text sounds.

Keywords 1
Text, real-time, recognition, image, text dubbing, text sounding, speech to text, text to speech, text recognition, text recognition algorithm, recognised text, video mode, android operating system, OCR algorithm, improved algorithm, optical character recognition, video recording mode, user-friendly interface, information technology, textual content, text analysis, intelligent system

1. Introduction

The main problems when recognising a text are the following [1-5]:

- Programs are designed to identify a text from only one image;
- Some programs are complicated for the user to understand;
- A large number of characters causes a slowdown in text recognition;
- The reader needs to be selected by the user to be recognised;
- Some Ukrainian symbols are incorrectly read;
- Programs are not adapted for text recognition in video mode;
- Programs are not designed for visually impaired people;
- No sound of the text from an image.

This study identifies the need for visually impaired people to read the text just by pointing the camera and hearing it as a result [1-5]. The software can be used by visually impaired and ordinary users who are learning the correct pronunciation of foreign words. The purpose of this work is to develop algorithms for text recognition and sound in video mode and their implementation for people with visual impairments and people who are learning a foreign language and correct pronunciation. The main idea of the project is to elaborate software for the Android operating system to help visually impaired people understand the text through its sound [5-7]:

- Text recognition technologies are promising but have some disadvantages for mobile phones (speed of text recognition and collection of all text within the phone screen);
- Ukrainian language in Android works incorrectly or does not exist at all.

The paper aims to improve the text recognition algorithm based on existing analogues, namely speed, by about 10-20%. To accomplish of the purpose and bring about the intended result, some specific tasks have to be solved:

COLINS-2021: 5th International Conference on Computational Linguistics and Intelligent Systems, April 22–23, 2021, Kharkiv, Ukraine
EMAIL: tymoshenko.k@outlook.com (K. Tymoshenko); victoria.a.vysotska@lpnu.ua (V. Vysotska); o.kovtun@donnu.edu.ua (O. Kovtun); roman.o.holoshchuk@lpnu.ua (R. Holoshchuk); svitlana.l.holoshchuk@lpnu.ua (S. Holoshchuk)
ORCID: 0000-0001-9218-2363 (K. Tymoshenko); 0000-0001-6417-3689 (V. Vysotska); 0000-0002-9139-8987 (O. Kovtun); 0000-0002-1811-3025 (R. Holoshchuk); 0000-0001-9621-9688 (S. Holoshchuk)

1. Implement text recognition in video mode;
2. Implement text recognition in Ukrainian and English;
3. Modify text recognition in Ukrainian and English for video recording mode;
4. Produce the sound of the text when you tap the screen;
5. The image is updated, and the found text is recognised at the number of frames (30-60 fps).

We expect to receive the following scientific results aimed at solving the problem specified above:

- Implement text recognition in video mode;
- Produce the sound of the text when you tap the screen;
- The image is updated, and the found text is recognised at the number of frames (30-60 fps).

The scientific novelty of our research is to develop an algorithm for recognising a text in video mode by overlaying the image so that the software can identify and process it faster.

2. Related works and applications

Consider the existing analogues to the created product, namely:

- ABBYYFineReader 10 HomeEdition [8-12];
- OcrCuneiform [13-16];
- CamScanner [17-21].

ABBYY FineReader 10 Home is a well-implemented utility for recognising and scanning the text [8-12]. The new version of the program copes well with transforming paper documents and various images into electronic form. The program possesses an excellent user-friendly interface in Russian. Once the processing of documents is completed, you can work with text data in any editor.

ABBYYFineReader 10 HomeEdition features [8-12]:

- The text recognition program works with 179 languages;
- After processing the text, there is an option to save it, share by e-mail or publish it on the Internet;
- Built-in tools responsible for increasing the quality of photos and images;
- The utility supports Windows 7, as well as Windows 8 and 10;
- The ability to convert finished text into PDF, DOC, RTF, XLS and HTML;
- The format is preserved, as well as the style of the document. The subsequent ability to run through Microsoft Word, Outlook and Excel;
- The program supports the recognition of documents received from MFPs, scanners, digital cameras and even from a mobile phone;
- There is a Russian version of the implementation of a simple interface.

Advantages [8-12]:

- ABBYY developer provides periodic updates to its scanning and text recognition utility;
- Processing of documents that were photographed on a mobile device;
- Availability of useful add-ons, such as ABBYYScreenshotReader and others;
- High accuracy of optical recognition of text information;
- Maintaining high image quality of documents;
- Any current Windows system is supported, excellent performance with Microsoft Office editors;
- Text data recognition program saves content in an edited format;
- The application processes paper documents with high quality.

Disadvantages are that a beta version of the program has limited validity and reduced functionality against the license background [8-12].

OcrCuneiform quickly recognises both scanned and photographed texts [13-16]. In the process of text recognition, a massive range of printed fonts is processed, while the original structure of electronic documents stays preserved. The finished result is obtained in the OCR program can be sent for further work in text editors. The utility's functionality is compared with the more famous program Abbyy Finereader, except that it is free to download Cuneiform. OCRCuneiForm features are [13-16]:

- Saving the original font structure;
- Possibility of optical recognition of various text documents;

- Ability to batch process multiple documents;
- Free Cuneiform program provides fast and efficient recognition of characters and text;
- Conversion of electronic graphic documents, as well as paper documents into an editable form;
- Cognitive Technologies developer periodically updates the recognising algorithms of the utility;
- The user-friendly interface of OcrCuneiform program has menu sections in Russian;
- Convert various text, graphic files and photocopies of faxes to editable file formats for Microsoft Office;
- The latest version of Cuneiform 12 includes adaptive content recognition;
- Very decent optimisation performance with operating systems such as Windows 7 as well as XP, Vista;
- Work is done with already recognised text, which provides document analysis and convenient search of tables, images, and text blocks.

We should also specify the advantages of the application [13-16]:

- The utility can be downloaded for texts scanning, recognition and analysis for free;
- High-quality recognition of text and graphic information;
- Program capabilities include processing documents obtained from laser and dot matrix printers;
- Convenient system of optical recognition of most existing printed fonts;
- Bringing any file structure into an editable format for well-known office programs and text editors;
- Complete analysis of scanned documents and a comfortable system for finding the desired table, any picture or text;
- Developers update their text recognition program by enlarging its options;
- Windows system including Vista and XP is supported;
- A high-level recognition system processes a poor photocopy of faxes.

Disadvantages are possible if to slow down the Suneiform program in the process of document recognition [13-16].

CamScanner is an intelligent document management solution for small businesses, organisations, government agencies, and schools [17-21]. It is ideal for those who want to sync, share and manage multiple files on all devices. Its functionality includes [17-21]:

- A quick scan of documents. The phone's camera is used to scan (photograph) any paper document: checks, notes, invoices, discussion boards, business cards, certificates. There is a batch scan mode which saves more time.
- Scanning quality optimisation. Trimming and auto-improvement make CamScanner unique. It ensures the purity and sharpness of texts and images.
- Easy search of documents. Search for any files in seconds. In searching for some papers, the powerful tool "OCR for the search" recognises the text in PDF documents.
- Intelligent document management is on mobile devices. You can divide documents into groups, sort messages by Date, tags, view as Sheet/Tile, etc. There is an option to set passwords for classified documents to prevent information leakage.
- Registration and synchronisation of documents. It is possible to register with CamScanner and save/synchronise documents immediately. With logging in, users can edit and synchronise documents on smartphones, tablets, PCs, and cloud services.

The analysis of programs and developments in the field helps us to define precise product requirements. The application, which is developed, aims to provide the user with a user-friendly interface that will automatically recognise text. After clicking on the frame on the screen, it voices an item. Both visually impaired and ordinary people can use this application. The following functions and algorithms are to be implemented in the program:

1. An algorithm for text recognition during a dynamic frame change;
2. Entering the recognised text into the frame;
3. Sound of the recognised text.

It is possible to define the basic requirements to the developed product having considered analogues.

The target audience can be divided into two groups. The first group is ordinary people who use the application for their personal everyday needs. The application might be used as an additional resource in learning a foreign language. It will help the user understand the correct pronunciation of foreign words and check the recognised text for grammatical errors. The second target group is people with visual impairments. The application will help them listen to the text recognised by the program. The software will be implemented for the Android operating system by using the Java programming language. All you have to do is tap on the phone screen icon to launch the software. Hardware requirements are 10 Mb of free disk space. System requirements include:

- Android 4.0 or higher operating system;
- Working language synthesis;
- A working camera on the phone.

The user documentation includes:

- Product use guide, which describes all the functionality and step-by-step algorithms for working with the product;
- Basic examples of work with software.

The user interface consists of a graphical interface and a text box. The graphical interface comprises the camera in video mode. The text box tells the user how to hear the text and how to zoom the camera. Because dynamic code processing takes many resources, a buffer is used to store the frames until the reader is fully recognised. The software is produced on any modern mobile phone.

The software must ensure the security of the user's input files and prevent access to the user's intellectual property or a third-party enterprise. Having considered all the specifications of the requirements and justification of the technology, we can draw the following conclusions:

1. Functions and algorithms that will be implemented in the program are [22-24]:
    a. Algorithm for text recognition when dynamically changing frames in Ukrainian and English and entering it in the structure;
    b. Algorithm for checking a recognised text for grammatical errors;
    c. Sound of the recognised text.
2. Hardware requirements are 10 Mb of free disk space.
3. System requirements:
    a. Android 4.0 or higher operating system;
    b. Speech synthesis applied;
    c. The working camera on the phone.
4. Quality requirements [25-35]:
    a. Functionality.
        i. Localisation – the software will support three languages (English, Russian and Ukrainian).
        ii. Reporting is used in case of errors, and the feedback will be sent to the developers.
        iii. Security of the software is a closed source, and the MD5 encryption algorithm encrypts the information.
    b. Usability.
        i. User-friendly interface.
        ii. The minimum number of items is on the screen.
        iii. The help system is built into the program.
    c. Reliability.
        i. Errors in text recognition are about 2%.
        ii. The software will be tested automatically and manually.
        iii. The software will be ready to work and will work 24/7.
        iv. In case of failure, the report will be sent to the developer's mail.
    d. Performance.
        i. It takes 0.10 seconds to run the program. After selecting the recognition language, the program will be entirely ready for use.
        ii. It takes up to10 Mb of free disk space.
        iii. The program requires 15 kWh.

e. Supportability.
   i. Autom atic tests are created to verify the program.
   ii. The program runs on Android 4.0 or higher.
   iii. Installation on a mobile phone is done by running .apk file.
   iv. There is a guide inside the system for its proper use, which can be called up through the menu.

General requirements are the following [36-45]:

- *Unification.* The software is developed according to the waterfall methodology of software development. Since most users use the Android operating system, the software runs on it. Java programming language in IDEA Android Studio is used for development.
- *Interoperability*. The program interacts with the phone's camera and speech synthesis to voice the text. It is tested for camera performance, correct text recognition, and speech synthesis to have the proper text voicing. The video mode plays while the camera is used.
- *Mobility*. The software is adjustable with Android phones.
- *Scalability.* The program is designed for visually impaired people and / or people who want to learn the correct pronunciation of foreign words.
- *Interaction with the user*. The software has an easy-to-understand interface that contains a menu and a screen for the user to interact.

3. Problem statement

Having studied modern approaches to the problem of mobile applications for visually impaired people, we have set the following tasks for the program to be developed:

- Implement text recognition in video mode;
- Implement text recognition in Ukrainian and English;
- Develop sound to the text when you tap the screen;
- Updated the image and text with the number of frames (30-60 fps).

3.1. Chart of a User Case

The main actor is the user who needs to recognise a text and how it sounds (Fig. 1).

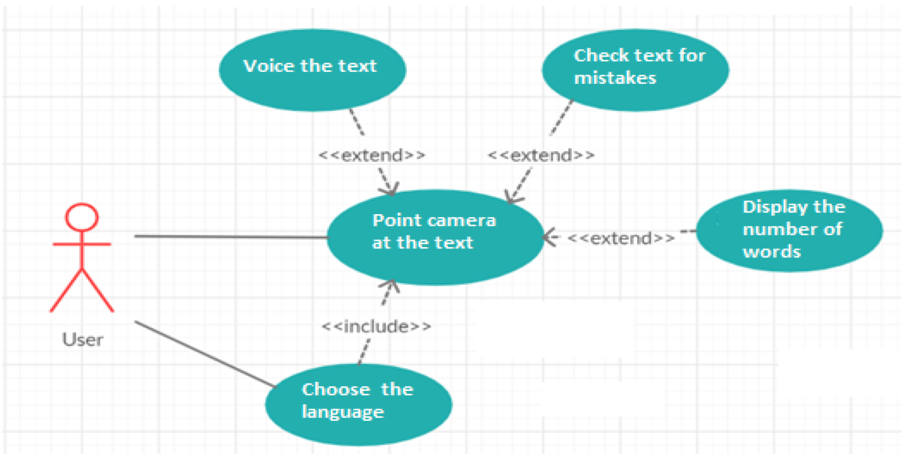


Figure 1: Chart of a user case

The main successful scenario:

1. The user opens the application and points the camera at the text;
2. The program begins to recognise a text;
3. After its full recognition, the text is highlighted by a frame;
4. The user clicks on the smartphone screen;
5. The system voices the recognised text.

Alternative streams:

1. Text in Ukrainian. The user selects the Ukrainian language in the application menu;
2. The user needs to know the number of words in the text or check it for errors:
    a. The user selects the appropriate menu item;
    b. The system analyses the recognised text;
    c. The system does not display the result on the screen.

3.2. Activity Diagram

The activity diagram shows all possible alternatives that occur under certain sentinel conditions and parallel processes that run in the system (Fig. 2).

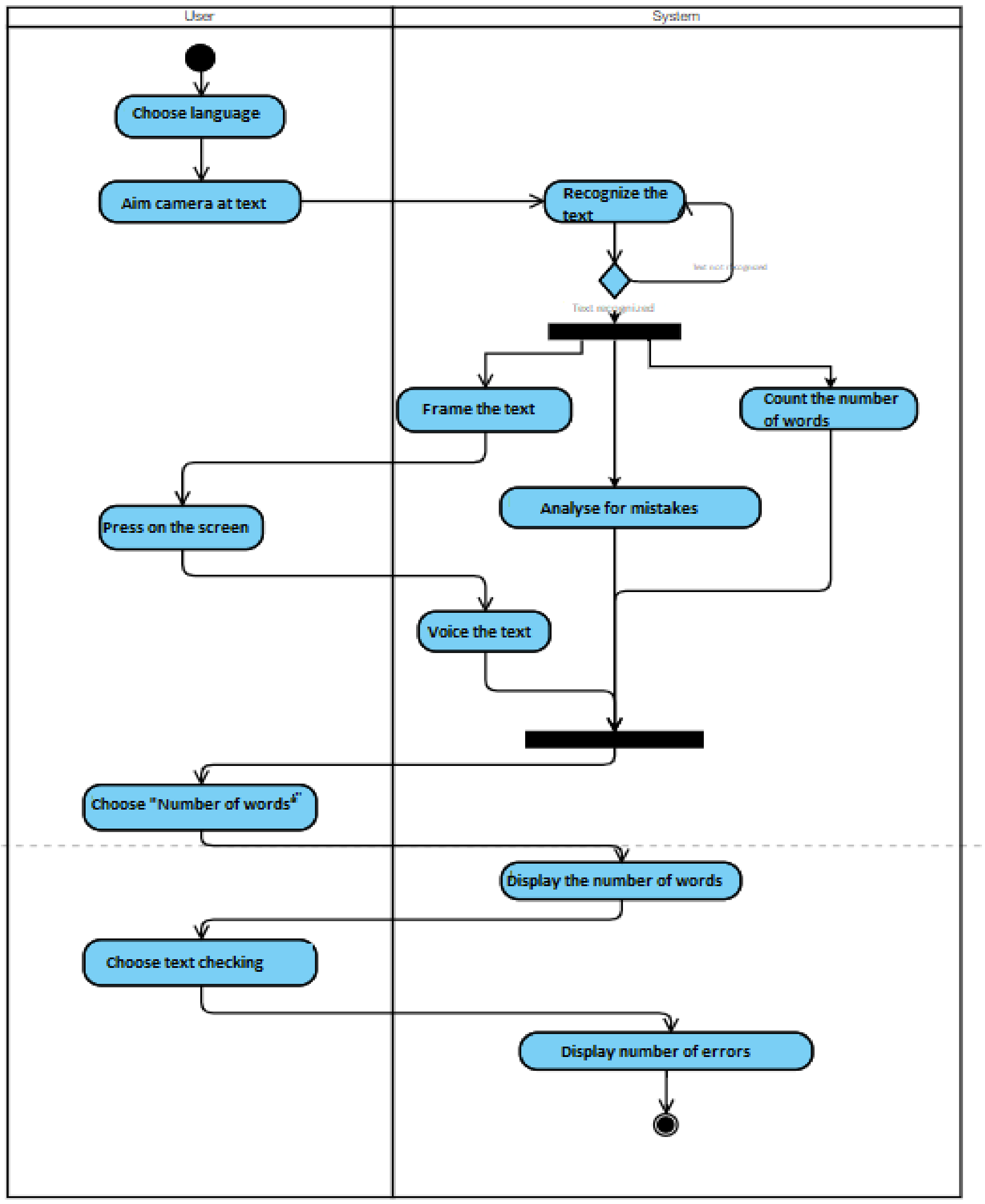
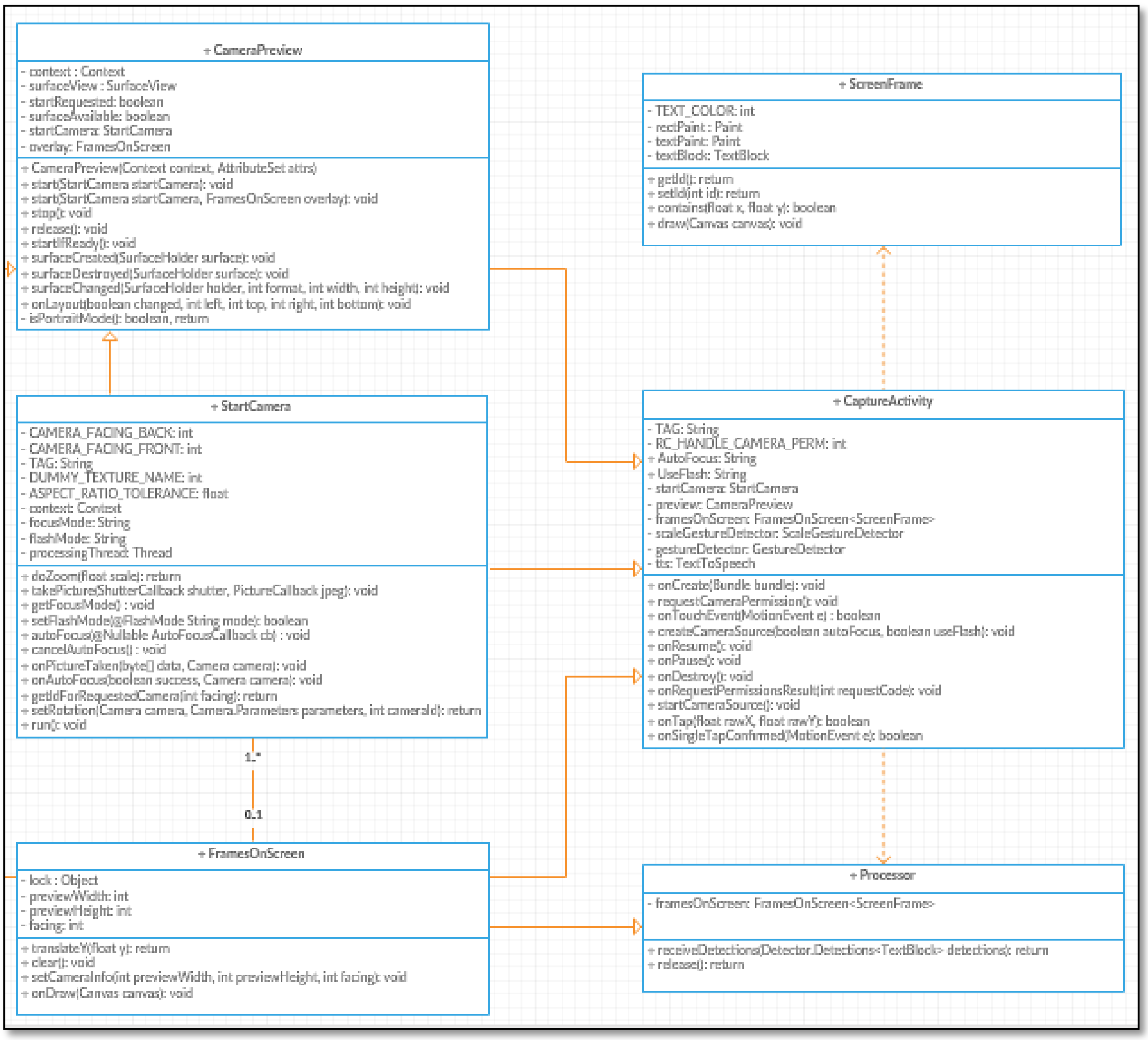


Figure 2: Activity Diagram

3.3. Class Diagram

The following classes are selected in the system: ScreenFrame, Processor, CaptureActivity, StartCamera, FramesOnScreen, CameraReview. Consider each separately (Fig. 3).

- CaptureActivity is the main class. All screen elements are initialised, all menu items are executed (English and Ukrainian languages, the number of words and their spelling are checked correctly). There are also checks for the camera and its ability to synthesise speech on a particular version of the Android operating system.



+ CameraPreview
- context : Context
- surfaceView : SurfaceView
- startRequested: boolean
- surfaceAvailable: boolean
- startCamera: StartCamera
- overlay: FramesOnScreen
+ CameraPreview(Context context, AttributeSet attrs)
+ start(StartCamera startCamera): void
+ start(StartCamera startCamera, FramesOnScreen overlay): void
+ stop(): void
+ release(): void
+ startIfReady(): void
+ surfaceCreated(SurfaceHolder surface): void
+ surfaceDestroyed(SurfaceHolder surface): void
+ surfaceChanged(SurfaceHolder holder, int format, int width, int height): void
+ onLayout(boolean changed, int left, int top, int right, int bottom): void
- isPortraitMode(): boolean, return
+ ScreenFrame
- TEXT_COLOR: int
- rectPaint : Paint
- textPaint: Paint
- textBlock: TextBlock
+ getId(): return
+ setId(int id): return
+ contains(float x, float y): boolean
+ draw(Canvas canvas): void
+ StartCamera
- CAMERA_FACING_BACK: int
- CAMERA_FACING_FRONT: int
- TAG: String
- DUMMY_TEXTURE_NAME: int
- ASPECT_RATIO_TOLERANCE: float
- context: Context
- focusMode: String
- flashMode: String
- processingThread: Thread
+ doZoom(float scale): return
+ takePicture(ShutterCallback shutter, PictureCallback jpeg): void
+ getFocusMode() : void
+ setFlashMode(@FlashMode String mode): boolean
+ autoFocus(@Nullable AutoFocusCallback cb) : void
+ cancelAutoFocus() : void
+ onPictureTaken(byte[] data, Camera camera): void
+ onAutoFocus(boolean success, Camera camera): void
+ getIdForRequestedCamera(int facing): return
+ setRotation(Camera camera, Camera.Parameters parameters, int cameraId): return
+ run(): void
+ CaptureActivity
- TAG: String
- RC_HANDLE_CAMERA_PERM: int
+ AutoFocus: String
+ UseFlash: String
- startCamera: StartCamera
- preview: CameraPreview
- framesOnScreen: FramesOnScreen<ScreenFrame>
- scaleGestureDetector: ScaleGestureDetector
- gestureDetector: GestureDetector
- tts: TextToSpeech
+ onCreate(Bundle bundle): void
+ requestCameraPermission(): void
+ onTouchEvent(MotionEvent e) : boolean
+ createCameraSource(boolean autoFocus, boolean useFlash): void
+ onResume(): void
+ onPause(): void
+ onDestroy(): void
+ onRequestPermissionsResult(int requestCode): void
+ startCameraSource(): void
+ onTap(float rawX, float rawY): boolean
+ onSingleTapConfirmed(MotionEvent e): boolean
1..*
0..1
+ FramesOnScreen
- lock : Object
- previewWidth: int
- previewHeight: int
- facing: int
+ translateY(float y): return
+ clear(): void
+ setCameraInfo(int previewWidth, int previewHeight, int facing): void
+ onDraw(Canvas canvas): void
+ Processor
- framesOnScreen: FramesOnScreen<ScreenFrame>
+ receiveDetections(Detector.Detections<TextBlock> detections): return
+ release(): return

Figure 3: Class Diagram

- ScreenFrame is a class that is responsible for drawing borders around text, colour and style. This class also considers the coordinates of the beginning and end of the rectangle (upper left edge and lower right). Due to this, the text is clearly in the frame.
- Processor – this class is responsible for the operation of the image overlay algorithm. That is, it works directly with the ScreenFrame and FramesOnScreen classes for the algorithm to function correctly.
- StartCamera is a class responsible for processing the image with the camera and the focus, which the user can change manually. A buffer that stores the image is created.

- FramesOnScreen – this class implements the image overlay algorithm. It uses methods and changes from the ScreenFrame level, but only for dynamic frame changes. Thus, the recognised text is stored in the clipboard along with the image and frame.
- CameraReview – a class in which the image overlay algorithm fully shows the result on the screen, and the user can tap the screen and hear the text in it.

3.4. Sequence Diagram

It is enough for the user to open the program and point the camera at the text to start working. The program will automatically begin to recognising it (Fig. 4). When the text is identified, it is surrounded by a frame on the screen, and you only need to tap the screen to hear it. There are also built-in functions for counting the number of words and grammatical errors in the text. To use these functions, you only need to select the appropriate menu item – the result will be displayed on the screen.

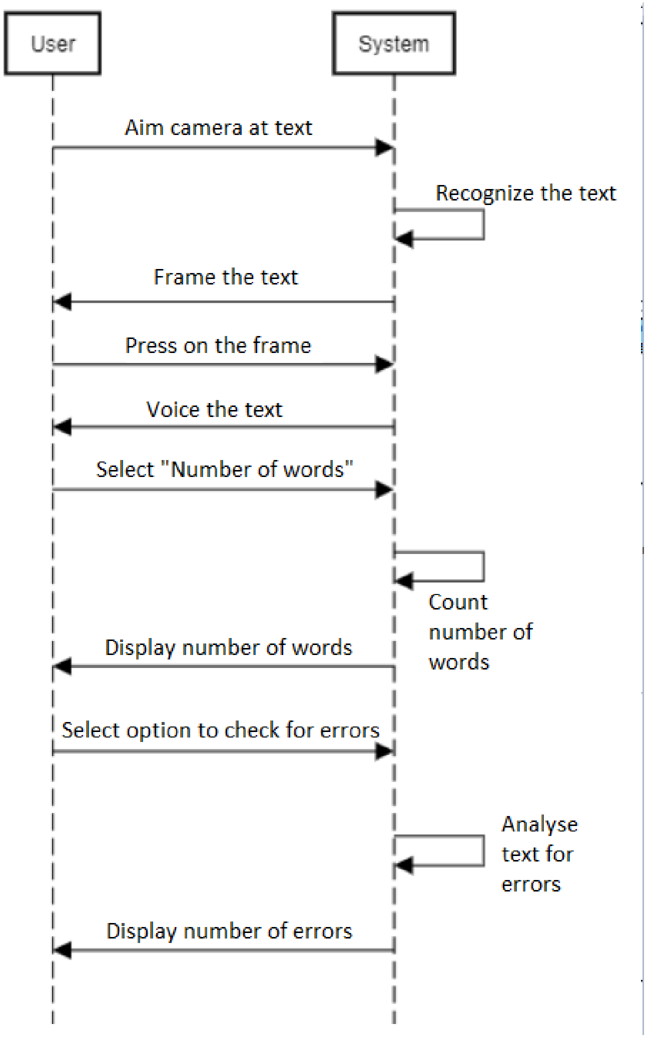


Figure 4: Sequence Diagram

3.5. Automaton Graph

After starting the system, the system waits for the text to appear in the smartphone's camera. As soon as the text seems, the program recognises it and circles it. The system then waits for the screen to click and voice the recognised text (Fig. 5).

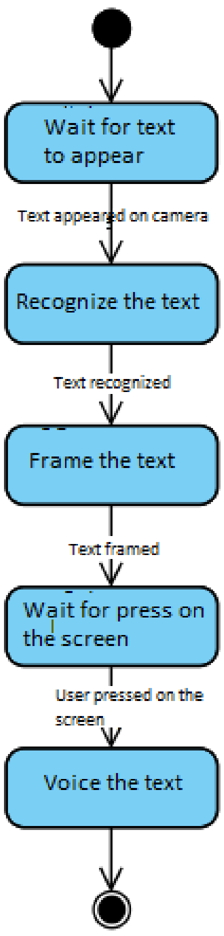


Figure 5: Automaton Graph

3.6. Batch diagram

The batch diagram shows the packets present in the system and their relations (Fig. 6):

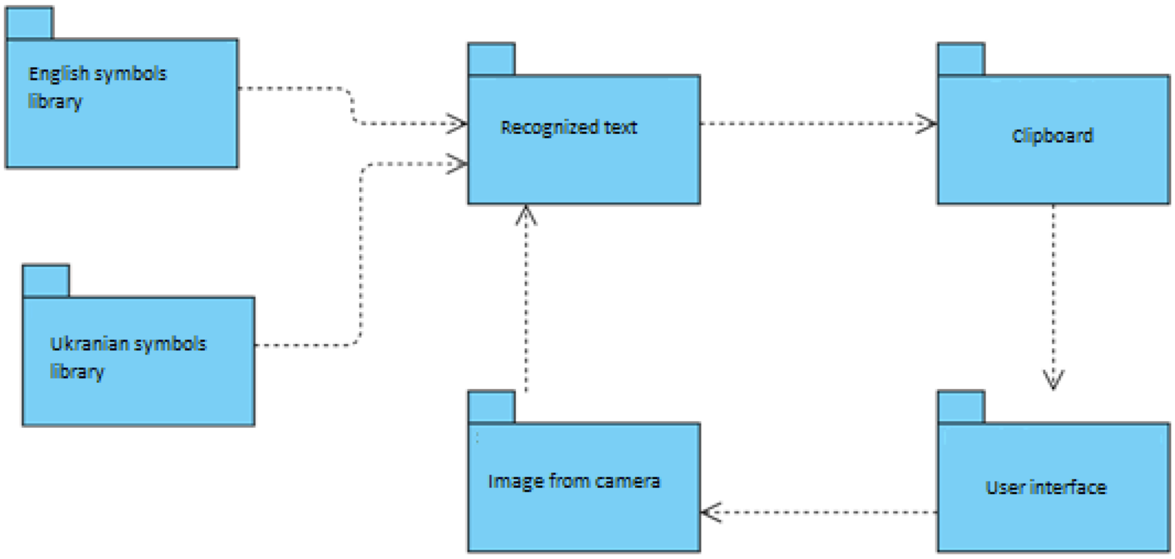


Figure 6: Batch diagram

4. Methods of analysis

4.1. Description of the text recognition algorithm

To understand how a text recognition algorithm works, we must first look at the types of these algorithms and find out which one works best in real life [46-67].

Of all the options, the ICDAR 2003 dataset has the highest character recognition performance. Unlike the more classic Optical Character Recognition (OCR) problems, where characters tend to be monotonous on fixed backgrounds, character recognition in an image scene is potentially more difficult due to a large number of possible variations with background, lighting, textures, and fonts. Indeed, significant efforts have made to create systems that integrate dozens of cleverly combined capabilities and stages of image processing.

Text recognition has aroused considerable interest in many areas of research. However, many methods are used to detect text and recognise characters based on artificial intelligence systems.

Functional teaching methods are currently the focus of research, especially for the visually impaired. As a result, a wide range of algorithms is now available for studying features with data. Many of the results obtained from the trait learning system have shown that higher performance in recognition tasks can be achieved due to the greater weight of the representation. Consider the architecture used to study the representation function and train the classifiers to detect and recognise the character system. The basic architecture is closely related to the neural network, but it can be used to build extensive sets of features with minimal configuration quickly, thanks to the learning method. Today, OCR technology uses all previous text recognition algorithms. Text recognition is usually reduced to OCR. It includes a computer system designed to transform images of printed text (usually obtained using a scanner) into machine-edited text or for - pictures of characters in a standard encoding scheme. Represents OCR is as a field of research in the field of artificial intelligence. People want to scan images and save them in a document and access the text of that document in .txt or .docx format.

Conversion is the first step in processing a scanned image. The scanned image is checked for shadow, skew, tilt [68-72]. There are options for capturing images with the left or the right orientation. The image is first converted to grayscale and then to binary. The result is an image suitable for further processing. After pre-processing, the meaningless image is transmitted to the segmentation stage, divided into individual characters. The binary image is checked for line spacing. The lines in the paragraphs are scanned to cross the horizontal space relative to the background. The image histogram is used to detect the width of the horizontal lines. The lines are then scanned vertically to cross the vertical space. Here, histograms are used to determine the width of words. The words are then broken down into characters that use character width calculations. Feature extraction occurs in the OCR segmentation phase, where the symbol of individual images is considered and extracted from the feature file. The classification is performed using the attributes extracted in the previous step, which corresponds to each character. These features are analysed using a set of rules and assigned to different classes. This classification is generalised for one type of font.

Similarly, classification rules are written for other characters. This method is standard because it extracts the shape of the characters and requires no training. Optical Character Recognition (OCR) aims to classify optical patterns (often contained in icons) that correspond to alphanumeric or other symbols. The OCR process involves several stages, including segmentation, feature use, and classification. Virtually any standard OCR software can now be used to recognise text in segmented frames. Most OCR software packages will have significant difficulty recognising text. Images of documents differ from raw images because they contain mainly text with several graphic images.

There are also many varieties of text recognition models.

The principle of operation of a neural network is that having received a new image on the input layer of neurons, the network responds with a pulse of a neuron. Because all neurons are named with letter values, the neuron that responded to the image corresponds to the recognised symbol. Going deeper into the terminology of networks, we can say that the neuron and the output also have many inputs. These inputs describe the pixel value of the image. That is, if there is a 16x16 image, the network inputs must be 256. Each information is perceived with a specific coefficient. As a result, a specific charge accumulates on each neuron at the end of recognition, where the order will be more significant. That neuron will emit a pulse. However, for the input coefficients to be set correctly, you must first train the network. A separate training module does this. This module takes another image from the training sample and transmits it to the network. This network analyses all positions of the black pixels. It aligns the coefficients minimising the error of matching the gradient method, after which a specific neuron is compared to this image. The algorithm of template methods is based on the comparison of the input graphic image. The first step in the template method is to convert the scanned image into a bitmap. In the recognition process, the patterns are sorted, and the difference between the image and the design is calculated. A class whose designs have minor difference is the result of recognition. These methods are divided into two categories: font-dependent and font-independent. Font-independent methods use specific templates that are universal for all types of fonts. However, this approach reduces the likelihood of correct recognition. Font-dependent algorithms are designed for only one type of font, which improves the quality of their work, but they are entirely inoperable when using other fonts. Methods have proposed recognising large volumes of text when some characters are guaranteed to be recognised by font-independent methods.

Then templates for font-dependent algorithms are built based on the recognised characters. With the existing variety of printed products in the learning process, it is impossible to cover all fonts and modifications. The advantages of this algorithm include ease of implementation, reliable operation in the absence of interference, high accuracy of recognition of flawed characters, the speed with a small alphabet. Disadvantages include a strong dependence on templates and the difficulty of selecting optimal templates, the inability to recognise a font that is different from the one embedded in the system, slow operation with a large number of obstacles, sensitivity to rotation, noise and distortion. Sign methods are because the image is placed by the N-dimensional vector of signs. The formation of the vector occurs during image analysis. This procedure is called symptom extraction. The standard for each class is obtained by similar processing of the symbols of the educational sample. The advantages of the method include ease of implementation, high final ability, resistance to changing the shape of the characters, high speed. The disadvantages of this method include instability to various image defects, loss of information about the symbol at the stage of obtaining features.

4.2. Methods of working with pixels in image recognition

The RAW format is used to work in the Android operating system with the camera, which is later converted to JPEG format. RAW stores more information, but there is no compression, while JPEG is a processed file that is stored in a container. Saving in JPEG can carry risks such as loss of clarity, general blur. When the picture is taken, the light passes through the lens by the shortest path and hits the sensor, filled with red, blue and green filters. It is called a Bayer filter, consisting of 25% red, 25% blue and 50% green photocells (Fig. 7).

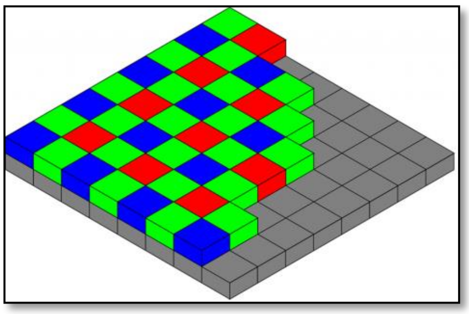


Figure 7: Bayer filter

It should be understood that a RAW file is information that contains a set of digital colour values that are assigned to each conditional pixel (Fig. 8). In the future, an array of values from the matrix enters the image processor, and it is the process of compression into a JPEG file. Inside the processor, algorithms that bring the image to the correct form (Fig. 9) function. During image processing and development, the data is erased, and only the picture with all the noise and artefacts remains. According to the JPEG scheme, image compression occurs in several stages. The first step is to convert the image from RGB space to YUV space, which has brightness and colour characteristics. Further work with the image will occur with this model, which allows obtaining a large degree of compression due to its factors.

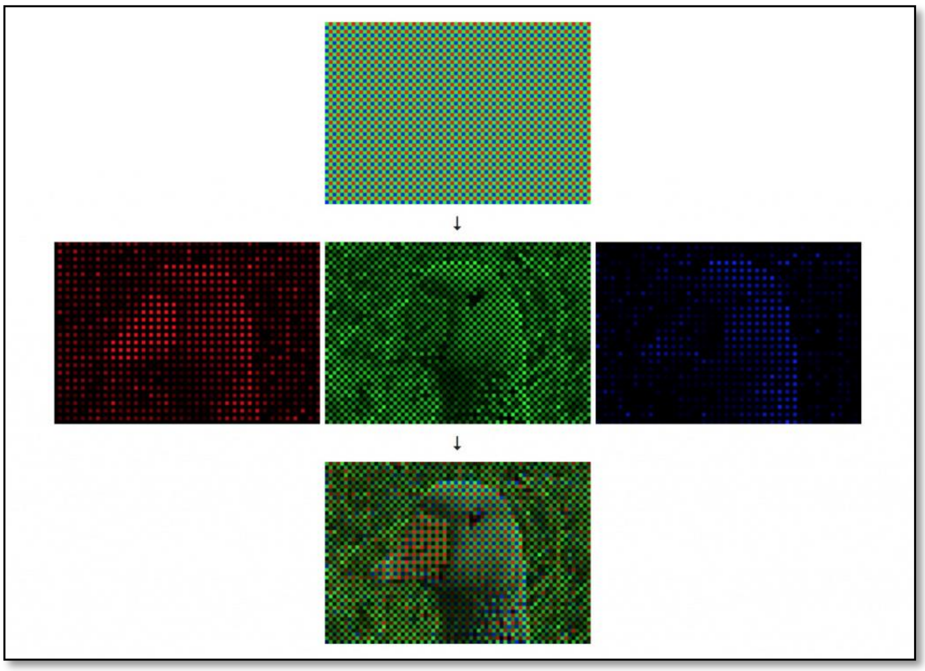


Figure 8: Images with different filters

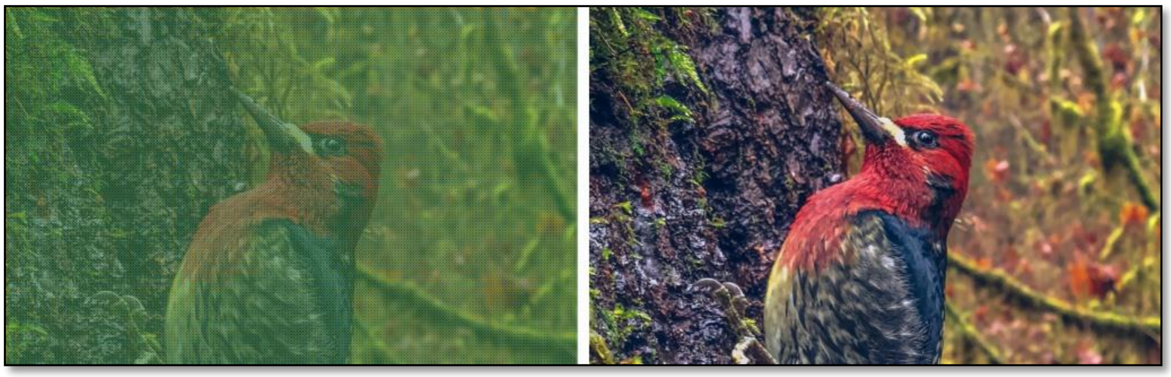


Figure 9: Image Handling

The Android operating system uses the YUV model when working with images. It consists of three components: brightness (*Y*) and two colour components (*U* and *V*) [25-30].

These components are defined based on RGB according to the formulas.

$$Y = K_R \times R + (1 - K_R - K_B) \times G + K_B \times B, \tag{1}$$

$$U = B - Y, \tag{2}$$

$$V = R - Y. \tag{3}$$

If a reverse transformation is required, it is performed according to the formulas:

$$R = V + Y, \tag{4}$$

$$G = Y - \frac{K_R \times V + K_B \times U}{1 - K_R - K_B}, \tag{5}$$

$$B = U + Y. \tag{6}$$

The inverse conversion preserves the range of change of the RGB component, but the degree of change of the components U and V more than in Y, so it is not convenient for encoding and data transmission. Therefore, rationing is introduced. By definition, a component varies in interval $[-(1 - K_B)A, (1 - K_B)A$, a V in the interval $[-(1 - K_B)A, (1 - K_B)A$, assuming that the RGB components change in the range $[0, A)$. To bring to the interval $[-A/2, A/2)$ components U and V normalise according to the formula [25-30]:

$$EU = \frac{1}{2} \times \frac{U - Y}{1 - K_B}, V = \frac{1}{2} \times \frac{R - Y}{1 - K_R}. \quad (7)$$

If a reverse transformation is required [25-30], it is performed according to the formula:

$$R = Y + 2 \times V \times (1 - K_R), \quad (8)$$

$$G = Y - \frac{2K_R \times V \times (1 - K_R) + 2K_B \times U \times (1 - K_B)}{1 - K_R - K_B}, \quad (9)$$

$$B = Y + 2 \times U \times (1 - K_B). \quad (10)$$

This method of component representation is used for analogue format [25-30].

The YUV digital representation is a YCbCr format. For the digital data format, integer powers of two are used. Most often 8-bit, 10-bit and so on. Because U and V may be harmful, they introduce an offset of the coding levels. Spatial coding, such as YUYV or YUV422, is also used to thin out less informative components. Consider the coefficients K_R and K_B (K_R= 0.299 and K_B = 0.114) [25-30].The same values are used to convert colour space to YPbPr and JPEG. The second stage when working with the image is the division into square sections of 8x8 pixels. After that, the transformation is performed over each plot. During the conversion, the analysis of each block is performed. Its decomposition into colour components and the analysis of the frequency of occurrence of each colour (Fig. 10).

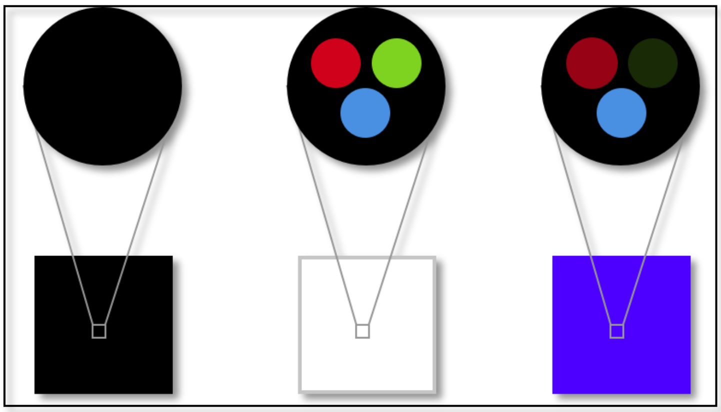


Figure 10: The principle of pixel formation

When compressed, the amount of information always depends on the image quality. The parts are entirely erased at high compression levels, and the image itself turns grey (Fig. 11).

Figure 11: Converting the image into black and white

At medium and low compression levels, the file stores approximate colour information. All these indicators depend on the degree of compression. JPG uses the Fourier series to conserve and rejects members of a higher order series at high compression ratios. The causes one problem – if the image is saved in JPEG format, it is impossible to restore it to the last pixel. The JPEG format is called the "format with loss", and therefore, the images from it become worse.

Colour and brightness information is encoded so that only the differences between adjacent blocks are preserved. As a result, strings of numbers represent the blocks. Since the blocks contain many zeros due to processing, the last stage of coding gives good results.

4.3. The principle of speech synthesis algorithm

To use text-sound, you need to analyse which method is best for software development.

The sound of the text is divided into types:

- Parametric synthesis;
- Compilation synthesis;
- Synthesis according to the rules (according to the printed text).

Parametric synthesis is used for any language, but it cannot be used immediately on the finished text. This method only works with a limited amount of text, so the recognition quality is relatively high. A tone is used for vowel sounds, and a noise generator is used for consonant sounds. However, it is used when recording music. Consider another method, namely compilation synthesis. This method is formed to recognise the text on the finished collection of characters. The recognition element itself must be at least a word. In general, the number of factors is limited to a few hundred words. This method is used in everyday life - in help desk or machines that are equipped with a voice response system. Another method is the complete synthesis that does not use human language but uses vocal linguistic algorithms. It is divided into two approaches. The first is a formant synthesis of language according to the rules, and the second is an articulatory approach. Formant synthesis is based on the frequency resonances of the speech speaker system. The algorithm simulates the operation of human speech, which works as a set of resonators. This universal and promising technology is only improving over the years. The articulatory method works as follows - adding to the model the phonetic features of individual sounds.

There is a rule of language recognition technology that uses recorded segments of natural language. They are divided into the following types of synthesis: allophonic and diphonic. For the diphone method, the essential elements are used in the combination of phonemes, and for the allophone method, the left and right phonemes are combined. In this case, different types of contexts are integrated into classes according to the degree of acoustic proximity. The main advantages of such methods are that they make it possible to synthesise text according to an already specified pattern from libraries. The disadvantage is only one, they do not reproduce authentic natural sounds. It is also challenging to control speech characteristics, namely the tone, speed, context of the phrase. Despite the high level in this area, the algorithm developers still have difficulty creating a perfect synthesis of language.

4.4. The structure of the program text recognition and sound

Since the image is not static but changeable, its speed is essential to us. The frame rate on the phone ranges from 30 to 60 frames per second. the program will circle the found text with a structure inside the text, to achieve maximum text recognition performance (Fig. 12).

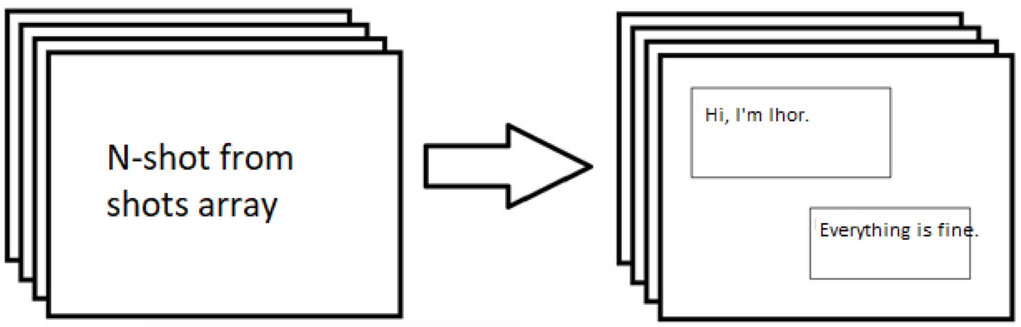


Figure 12: Text recognition

The frame and the text in it will be updated in about 15-30 frames to reduce the load on the phone. Thus, the user will not notice much difference. If the characters are not decrypted, the system will skip them, and they will not be visible on the screen. The system will automatically delete the old text frame in 15-30 frames and immediately create a new one with new data. Once the text is found, the user will click on it and hear the text within the frame (Fig. 13).

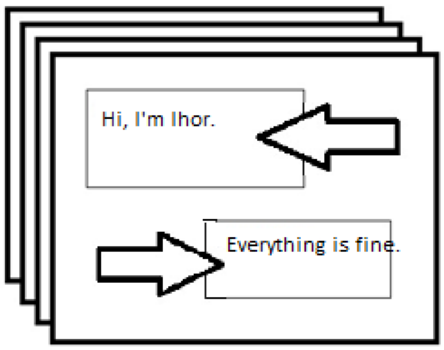


Figure 13: Tap the screen

The algorithm works so that it takes, for example, 15 frames. The text recognition algorithm is started, and each frame is recognised. These 15 frames are superimposed on each other, and when the text becomes recognised – the user can see the result on his phone. However, it should be noted that natural and technical factors affect its speed in text recognition. Therefore, the number of frames will change. The research will be performed when the brightness changes and the number of characters on the screen changes. The results should show how they affect the operation of the algorithm. Faster processing is achieved because the OCR algorithm checks each image, and at the slightest movement of the phone, the frame changes and the text that needs to be recognised is lost (Fig. 14).

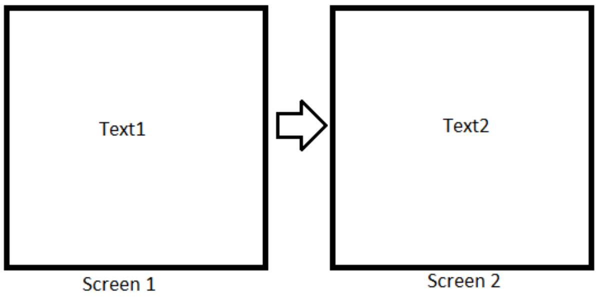


Figure 14: The principle of standard OCR when changing the frame

Text recognition requires a more explicit focus on the phone. The improved algorithm makes overlaying images, and thus there is no loss of frames. That is, the algorithm will more quickly recognise the text on the phone screen.

4.5. Methods of text recognition research

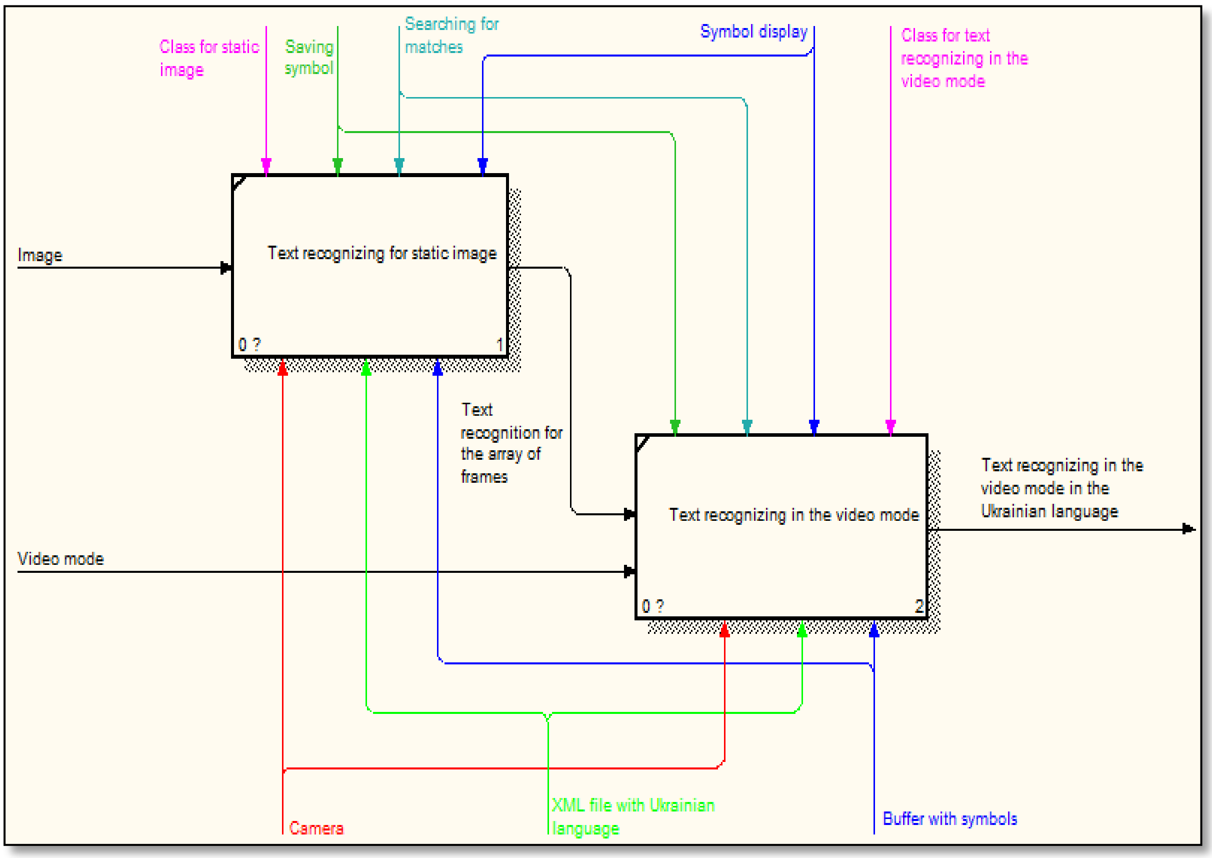
The first stage of scientific research is a complete analysis of text recognition in video mode. The study is carried out using various sources of information located in Ukraine and abroad. The primary source of data is the World Wide Web, which searches for articles on this topic. Based on the analysis, the task to the given theme is formed, the choice of research methods, the basic parameters for estimating the quality of work of the software and algorithm, and the expected economic effect.

The second stage of scientific research is the solution to the tasks set at the first stage. The software product is tested in actual conditions. The leading indicators for testing are the change in brightness and the number of characters on the phone screen. The study takes place in the form of 20 tests. Test results are shown in the form of tables and diagrams for a better understanding of the information.

The last stage is the analysis of the obtained results and their design. They are demonstrated as tables and charts. The results are compared with analogues, and the quality of the software is summarised.

The input for the research is the brightness and the number of characters on the screen. At the entrance to the software product, there is an image that is updated.

The source data of the software product is recognised text and graphically drawn frames of this text so that the user can click and the text itself is voiced as a result (Fig. 15).



Class for static image
Saving symbol
Searching for matches
Symbol display
Class for text recognizing in the video mode
Image
Text recognizing for static image
0 ?
1
Text recognition for the array of frames
Text recognizing in the video mode
0 ?
2
Text recognizing in the video mode in the Ukrainian language
Video mode
Camera
XML file with Ukrainian language
Buffer with symbols

Figure 15: Input and output data of the software

Having considered the methods of text recognition and sound, as well as the principle of the program, we can draw the following conclusions:

- Android Studio IDE and Java programming language are used for software development;
- Language synthesis, which is built into the Android operating system, is used to voice the selected text;
- Text recognition is performed by superimposing images on each other, which speeds up the algorithm;
- Text recognition requires several steps to ruffle the image, namely from POW to JPEG and YUV, for better text recognition.

5. Results

The software is developed on Android Studio using the Java programming language. You must first upgrade your Google Repository to use the Mobile Vision Text API. To get started, you need to change the dependency block to work with Google services, as the main character library is located on Google's servers. Nevertheless, after connecting, the need for the Internet disappears, as everything you need will load [68-72].

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.google.android.gms:play-services-vision:15.0.0'
    implementation 'com.android.support:support-v4:27.1.1'
    implementation 'com.android.support:design:27.1.1'
}
```

Create an XML file that initialises the screen elements for the program's correct operation [70].

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/topLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true">

    <com.google.android.gms.samples.vision.ocrreader.ui.cameraFilming.CameraPreview
        android:id="@+id/preview"
        android:layout_width="match_parent"
        android:layout_height="match_parent">


<com.google.android.gms.samples.vision.ocrreader.ui.cameraFilming.FramesOnScreen
            android:id="@+id/graphicOverlay"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

    </com.google.android.gms.samples.vision.ocrreader.ui.cameraFilming.CameraPreview>

</LinearLayout>
```

Create a TextRecognizer. This detector object processes images and determines what text appears inside them. After initialisation, TextRecognizer is used to detect text in all types of images [68, 72].

```
@SuppressLint("InlinedApi")
private void createCameraSource(boolean autoFocus, boolean useFlash) {
    Context context = getApplicationContext();
    TextRecognizer textRecognizer = new TextRecognizer.Builder(context).build();
    textRecognizer.setProcessor(new Processor(framesOnScreen));
```

TextRecognizer is ready to go. If your device does not have enough memory or Google Play Services cannot load OCR dependencies, the TextRecognizer object will not work [68]. Before using it to recognise the text, you need to check if it is ready [68]. Add this check to create CameraSource after initialising TextRecognizer [68].

```
    if (!textRecognizer.isOperational()) {
        Log.w(TAG, " Detector dependencies are not yet available.");
        IntentFilter lowstorageFilter = new IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
        boolean hasLowStorage = registerReceiver(null, lowstorageFilter) != null;

        if (hasLowStorage) {
            Toast.makeText(this, R.string.low_storage_error, Toast.LENGTH_LONG).show();
            Log.w(TAG, getString(R.string.low_storage_error));
        }
    }
    startCamera =
            new StartCamera.Builder(getApplicationContext(), textRecognizer)
            .setFacing(StartCamera.CAMERA_FACING_BACK)
            .setRequestedPreviewSize(1280, 1024)
            .setRequestedFps(15.0f)
            .setFlashMode(useFlash ? Camera.Parameters.FLASH_MODE_TORCH : null)
            .setFocusMode(autoFocus ? Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO : null)
            .build();
}
```

Now that you have verified that the TextRecognizer is ready, you can use it to recognise individual frames [68]. Since the main functionality is text recognition in camera mode, we create CameraSource, which is pre-configured to control the camera. You need to set high-quality shooting and turn on autofocus to cope with the task of recognising small text. If users look at large blocks of text, such as signs, they can use lower quality, and then frame processing will be faster [68].

```
startCamera =
        new StartCamera.Builder(getApplicationContext(), textRecognizer)
        .setFacing(StartCamera.CAMERA_FACING_BACK)
        .setRequestedPreviewSize(1280, 1024)
        .setRequestedFps(15.0f)
        .setFlashMode(useFlash ? Camera.Parameters.FLASH_MODE_TORCH : null)
        .setFocusMode(autoFocus ?
Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO : null)
        .build();
```

The application can now detect text in individual frames using the detection method in TextRecognizer. Therefore, you can find the text, for example, in a photo. However, to read the text directly during video recording, you need to implement a Processor that will process the text as soon as it appears on the screen. Go to the Processor class to implement the Detector.Processor interface:

```
package com.google.android.gms.samples.vision.ocrreader;
import android.util.Log;
import android.util.SparseArray;
import com.google.android.gms.samples.vision.ocrreader.ui.cameraFilming.FramesOnScreen;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.text.TextBlock;

public class Processor implements Detector.Processor<TextBlock> {
    private FramesOnScreen<ScreenFrame> framesOnScreen;
    Processor(FramesOnScreen<ScreenFrame> ocrFramesOnScreen) {
        framesOnScreen = ocrFramesOnScreen;
    }

    @Override
    public void receiveDetections(Detector.Detections<TextBlock> detections) {
        framesOnScreen.clear();
        SparseArray<TextBlock> items = detections.getDetectedItems();
        for (int i = 0; i < items.size(); ++i) {
            TextBlock item = items.valueAt(i);
            if (item != null && item.getValue() != null) {
                Log.d("Processor", " Text found! " + item.getValue());
                ScreenFrame graphic = new ScreenFrame(framesOnScreen, item);
                framesOnScreen.add(graphic);
            }
        }
    }
    @Override
    public void release() {framesOnScreen.clear();}
}
```

To implement this interface, you need to redefine two methods. The first receive Detections and receive a TextBlocks input from TextRecognizer as detected [68]. The second, release, is used to free resources when destroying TextRecognizer. In this case, you just need to clear the graphics canvas, which will delete all OcrGraphic objects. Get TextBlocks and create objects ScreenFrame for each text block detected by the processor. Now that the processor is ready, you need to configure textRecognizer to use it [68].

```
TextRecognizer textRecognizer = new TextRecognizer.Builder(context).build();
textRecognizer.setProcessor(new Processor(framesOnScreen));
```

Implement the draw method in ScreenFrame. You need to check whether the image has text, convert the coordinates of its boundaries into a frame, and then draw both the structure and the text [68, 72].

```
@Override
public void draw(Canvas canvas) {
    if (textBlock == null) {
        return;
    }

    RectF rect = new RectF(textBlock.getBoundingBox());
    rect = translateRect(rect);
    canvas.drawRect(rect, rectPaint);

    List<? extends Text> textComponents = textBlock.getComponents();
    for(Text currentText : textComponents) {
        float left = translateX(currentText.getBoundingBox().left);
        float bottom = translateY(currentText.getBoundingBox().bottom);
        canvas.drawText(currentText.getValue(), left, bottom, textPaint);
    }
```

Now the text from the camera is converted into structured lines, and these lines are displayed on the screen. Using the TextToSpeech API, built into Android, and the method in ScreenFrame, you can teach the application to speak aloud when you click on the text [68]. First, the process is implemented in ScreenFrame. You need to check that the x and y coordinates are within the displayed text box [72].

```
private boolean onTap(float rawX, float rawY) {
    ScreenFrame graphic = framesOnScreen.getGraphicAtLocation(rawX, rawY);
    TextBlock text = null;
    if (graphic != null) {
        text = graphic.getTextBlock();
        if (text != null && text.getValue() != null) {
            Log.d(TAG, " The text is voiced! " + text.getValue());
            // Speak the string.
            tts.speak(text.getValue(), TextToSpeech.QUEUE_ADD, null, "DEFAULT");
        }
        else {
            Log.d(TAG, " No text");
        }
    }
    else {
        Log.d(TAG," text not detected");
    }
    return text != null;
}
```

The method of image overlay is used to increase the speed of text recognition. It calculates the coordinates of the text and the text itself [69-70].

```
@Override
protected void onLayout(boolean changed, int left, int top, int right, int bottom) {
    int Width = 320, Height = 240;
    if (startCamera != null) {
        Size size = startCamera.getPreviewSize();
        if (size != null) {
            Width = size.getWidth();
            Height = size.getHeight();
        }
    }
    if (isPortraitMode()) {
        int tmp = Width;
        Width = Height;
        Height = tmp;
    }
    final int viewWidth = right - left;
    final int viewHeight = bottom - top;
    int dynamicWidth, dynamicHeight;
    int dynamicX = 0, dynamicY = 0;
    float width = (float) viewWidth / (float) Width;
    float height = (float) viewHeight / (float) Height;
    if (width > height) {
        dynamicWidth = viewWidth;
        dynamicHeight = (int) ((float) Height * width);
        dynamicY = (dynamicHeight - viewHeight) / 2;
    } else {
        dynamicWidth = (int) ((float) Width * height);
        dynamicHeight = viewHeight;
        dynamicX = (dynamicWidth - viewWidth) / 2;
    }
    for (int i = 0; i < getChildCount(); ++i) {
        // One dimension will be cropped.  We shift child over or up by this offset and adjust
        // the size to maintain the proper aspect ratio.
        getChildAt(i).layout(
                -1 * dynamicX, -1 * dynamicY,
                dynamicWidth - dynamicX, dynamicHeight - dynamicY);
    }
    try {
        startIfReady();
    } catch (SecurityException se) {
        Log.e(TAG," No permission", se);
    } catch (IOException e) {
        Log.e(TAG, " Camera error.", e);
    }
}
```

TextToSpeech also depends on the recognition language. You can change the language based on the recognised text [68]. Language recognition is not built into the Mobile Vision Text API, so we use the Google Translate API. You can use the user's device language as the language for text recognition. Next, you need to check the correctness of the text and its quantity, which is in the frame. After that, a menu is created where the user selects the language and checks what he needs.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    switch (id) {
        case R.id.english:
            CameraPreview.getChildMeasureSpec(dispatchPopulateAccessibilityEvent(
                 onWindowStartingActionMode(FramesOnScreen)));
            return true;
        case R.id.ukranian:
            CameraPreview.getChildMeasureSpec(dispatchPopulateAccessibilityEvent(
                 onWindowStartingActionMode(FramesOnScreen)));
            return true;
        case R.id.textcheck:
            Toast toast1 = Toast.makeText(getApplicationContext(), rightText,
                 Toast.LENGTH_LONG);
            toast1.show();
            return true;
        case R.id.textcheck2:
            Toast toast2 = Toast.makeText(getApplicationContext(),
                 " Number of words:" + CountText, Toast.LENGTH_LONG);
            toast2.show();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Create a method of tapping the screen [69].

```
private void requestCameraPermission() {
    Log.w(TAG, "Camera permission is not granted. Requesting permission");
    final String[] permissions = new String[]{Manifest.permission.CAMERA};
    if (!ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.CAMERA)) {
        ActivityCompat.requestPermissions(this, permissions, RC_HANDLE_CAMERA_PERM);
        return;
    }
    final Activity thisActivity = this;
    View.OnClickListener listener = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            ActivityCompat.requestPermissions(thisActivity, permissions,
                    RC_HANDLE_CAMERA_PERM);
        }
    };
}
@Override
public boolean onTouchEvent(MotionEvent e) {
    boolean b = scaleGestureDetector.onTouchEvent(e);
    boolean c = gestureDetector.onTouchEvent(e);
    return b || c || super.onTouchEvent(e);
}
```

6. Experiments and discussion

6.1. The principle of operation of the developed algorithm and performance tests

Java software and tools for text recognition and sound in video mode are used to develop the software. The image is not static but changeable; an important aspect is its speed. The frame rate on the phone ranges from 30 to 60 frames per second. The program circles the found text with a frame to achieve maximum text recognition performance. The frame and the text in it are updated in about 15-30 frames to reduce the load on the phone. If the program finds characters that will not be decrypted, the system skips them, and they will not be visible on the screen. The system automatically removes the old text frame after 15-30 frames and immediately creates a new one with new data.

After the text is found, the user clicks on it and hears the text within the frame (Fig.16).

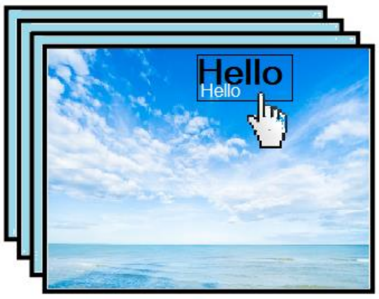


Figure 16: Tapping the screen

The algorithm works as follows: it takes, for example, 15 frames, the text recognition algorithm starts, and each frame recognises, then these 15 frames are superimposed on each other and when the text becomes recognised – the user can see the result on his phone. However, it should note that natural and technical factors affect its speed in text recognition. Therefore, the number of frames will change. A study was conducted when adjusting the brightness and changing the number of characters on the screen. The results showed how they affect the operation of the algorithm. Achieving faster processing is performed because the OCR algorithm checks each image, and at the slightest movement of the phone, the frame changes and therefore, the text that needs to be recognised is lost. That is, text recognition requires a more explicit focus on the phone.

The improved algorithm makes overlaying images, and thus there is no loss of frames. That is, the algorithm quickly recognises the text on the phone screen (Fig.17).

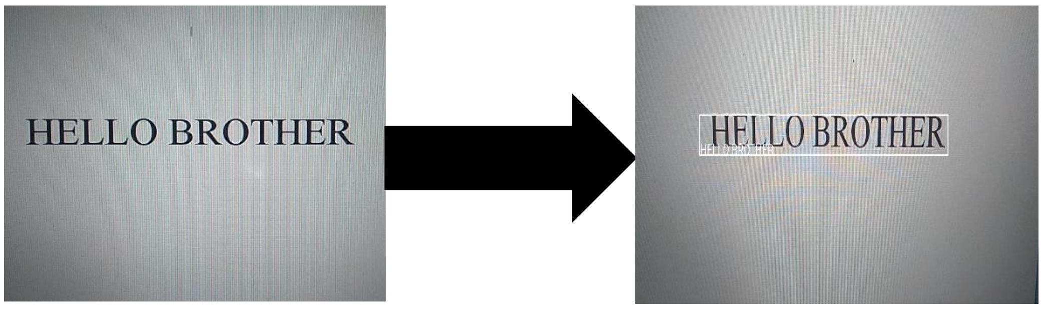


Figure 17: The principle of operation of the improved algorithm when overlaying images

A file with an .XML extension is created to recognise the text in English and Ukrainian. It contains the identification of Ukrainian symbols, namely uppercase and lowercase letters. Each character is given its name, depending on the surface. The next step is to connect to the class to recognise the text in the image. Text recognition takes place as follows. The symbol from the screen and compared with the sign from the XML file is taken (Fig. 15). The picture character is converted for a specific standard to resemble characters from an image and an XML file. Namely, the font and its size is change. In our case, it is Courier New 10pt (Fig.18).

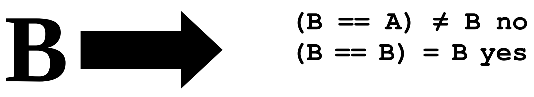


Figure 18: The character in the image to a specific standard

The characters in the image and the file are then compared. To do this, use a script to resemble characters from OCR technology. Once a character is located, it is stored in a temporary buffer, which consistently accumulates symbols. After recognising all the text is completed, the result is displayed on the screen as a frame, inside which is the recognised text. In the future, the temporary buffer is freed for the reader to recognise when the images are changed. And the new image is processed in a new way. To do this, the main class joins a class that works with an array of frames (Fig. 19). The result is the recognition of Ukrainian text when changing frames in video mode.

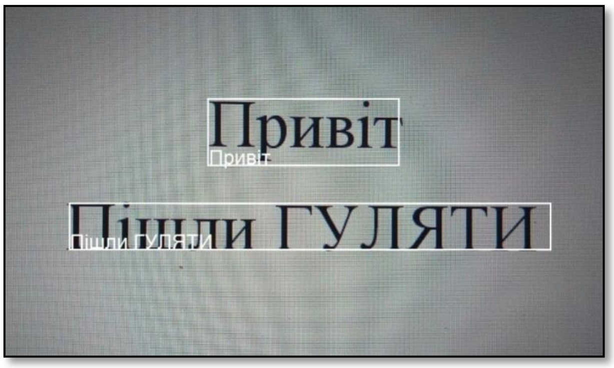


Figure 19: Text recognition

6.2. Software testing

The received software is tested to verify the quality and compliance with the requirements specified in the requirements specification. The quality of the system is checked during the testing process. Functional testing is chosen to test the system, as the software must function properly under certain conditions. The following tests and test data sets are created according to Table 1-2 to verify the quality of the software developed.

Table 1
Functional test cases

Options for use	Test cases	Test data
Text recognition in video mode	20	5
Working with the screen	20	5
Choice of language for text recognition and sound	20	5
Total	60	15

Table 2
General description: Audit TestCase id

TestCase id:	Steps:	ExpectedResults:
1 - Text recognition in video mode	Run program	The text should appear on the screen and change dynamically.
2 - Work with the screen	1. Run the program 2. Wait until the text is recognised; 3. Click on the frame in which the text is located.	The recognised text will be displayed on the screen and framed. The frame will become active for use.
3 - Choice of language for text recognition and sound	1. Run the program 2. Select Ukrainian or English from the menu.	Select the language from the menu and wait until the text is recognised.

Test result - **Passed successfully**.

Functional testing is successfully taken, and compliance with the requirements is confirmed. The following indicators are used to determine the success of the project:

- Developed tests cover requirement specifications;
- Various options of input data are tested.

So, all test patterns are successfully performed with a positive result.

6.3. Research on the accuracy of text recognition

Any text that is not perfect will cause difficulties even in the most advanced OCR system, which will significantly reduce the accuracy of insufficient image processing. For example, the recognition accuracy may be reduced by 20% when characters are separated due to poor image quality or if several characters merge due to a blurred or dark background.

First, a high-quality image is required for successful text recognition. A good example is a text that is printed on a laser printer, magazine, or book page. It is well recognisable in newspapers, printed on high-quality paper. If you are viewing an image from a camera or video, you need clear focus and the right conditions. Character boundaries should be clear, the font should be almost black, and the background should be close to white. As a rule, text recognition accuracy with favourable conditions exceeds 98%, i.e. one error per 50 characters or less. However, if one of the conditions is violated, the accuracy of recognition may fall by 15-20%.

Difficulties in recognising text occur when changing the image's contrast, namely a yellowed background, pale or faded paint on the reader. In addition, an important aspect is the recognition of the text when the light changes. As the brightness of the image decreases, the algorithm loses the accuracy of text recognition. According to the results of the study, the accuracy dropped by about 10-15%. The experiments are performed only with a class that recognised text from a single image. The next step was to study the accuracy of text recognition in video. The factors that influenced text recognition remained the same. Still, another factor in the deterioration inaccuracy was the dynamic change of the image, i.e. the array of frames that changed all the time. Recognition accuracy dropped to 40%.

The research is carried out in several stages. Because the image is updated dynamically and under different natural conditions, or rather the brightness and the different number of characters in the text that will be recognised, several graphs and tables will be built to understand the algorithm better.

The first stage conducted 20 tests that will show how often the text is recognised and the user sees it on the phone screen when the brightness changes, according to Table 3.

Table 3
Testing text recognition when the brightness changes

N of tests	Nb Frames:	Illumination, lux	N of tests	Nb Frames:	Illumination, lux
1	12	980	11	18	600
2	14	900	12	19	580
3	14	860	13	20	550
4	14	840.	14	19	530
5	15	800	15	20	500
6	17	780	16	19	490
7	16	720	17	18	470
8	19	700	18	19	430
9	17	650	19	17	400
10	18	630	20	19	350

The graph clearly shows that the darker the image on the phone, the longer it takes to recognise the text on it. On average, about 18 frames are obtained (Fig. 20). The examples show how the program recognises text when the brightness changes. The darkest image is at the top left, and the lightest from the sample is at the bottom right (Fig. 21).

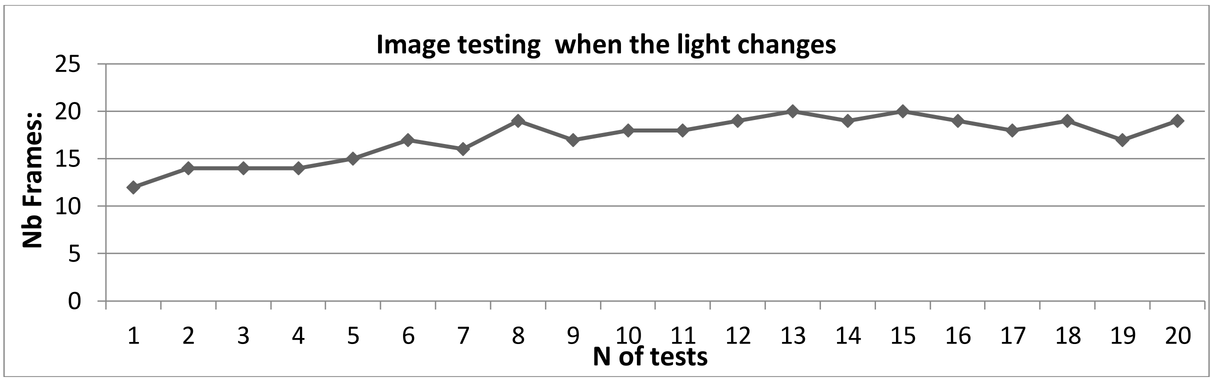


Figure 20: Testing the image when the brightness changes (Number of frames as the legend element)

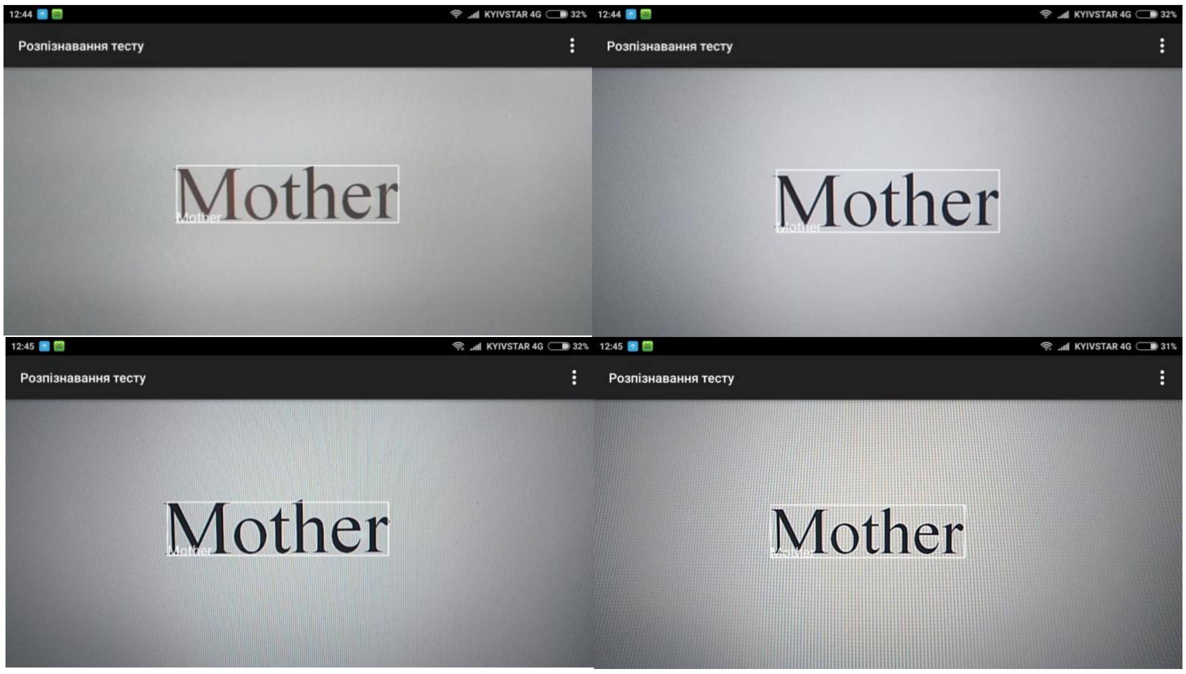


Figure 21: Examples of text recognition when the light changes

Next, a study of text recognition with different numbers of characters is carried out. For a more accurate analysis, the text is taken with a difference of one character by Table 4.

Table 4
Testing when changing the number of characters

N of tests	Nb Frames:	Number of characters	N of tests	Nb Frames:	Number of characters
1	18	3	11	23	13
2	20	4	12	25	14
3	21	5	13	28	15
4	21	6	14	30	16
5	22	7	15	31	17
6	22	8	16	32	18
7	22	9	17	30	19
8	23	10	18	34	20
9	26	11	19	33	21
10	24	12	20	32	22

Based on the results, we can say that the execution of the text recognition algorithm slows down with the increasing number of characters and the screen (Fig. 22). The examples show precisely how the program recognises the text and the boundaries for the reader (Fig. 23).

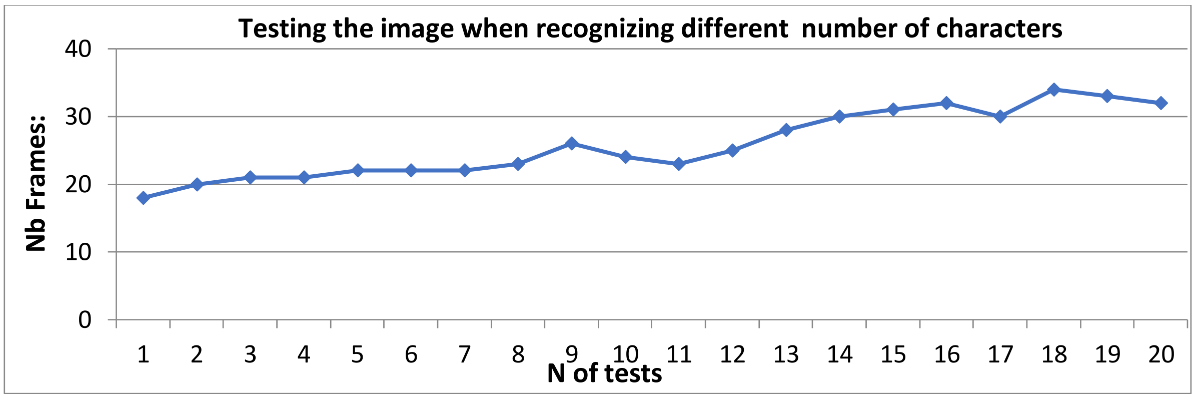


Figure 22: Test the image when the number of characters changes (Series "Number of frames" as Element of the legend)

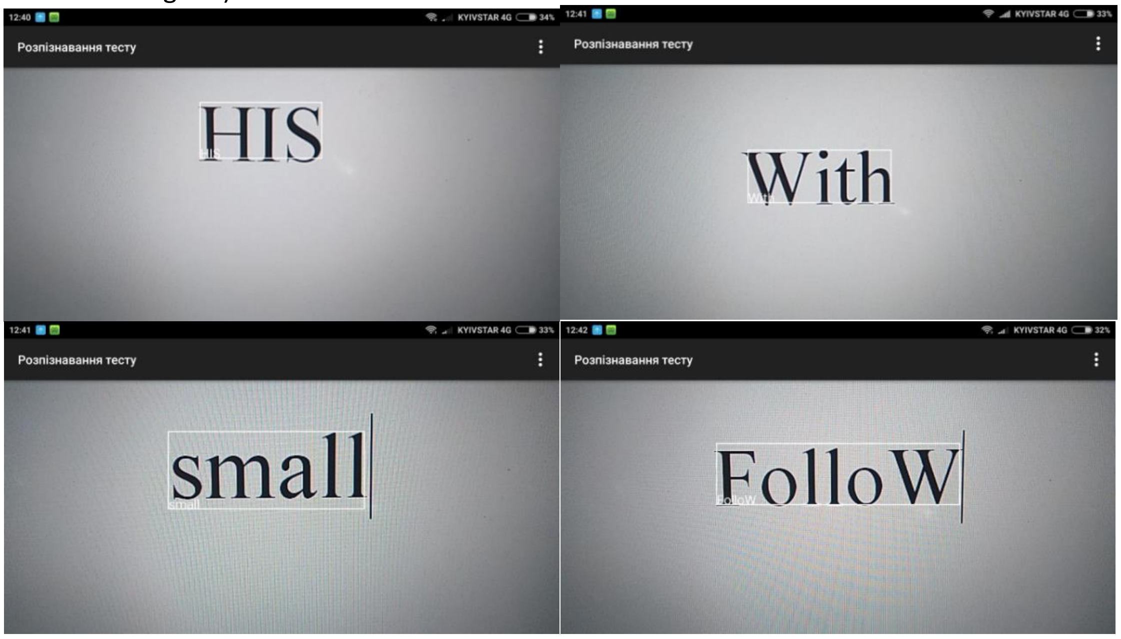


Figure 23: Examples of text recognition when changing the number of characters

The latest study is that the algorithm with changing the brightness and increasing the number of characters has been tested. It is 20 tests that are performed and which showed the following results according to Table 5. The graph clearly shows that the text recognition algorithm is strongly inhibited because many factors affect its operation (Fig. 24).

Table 5
Testing of both parameters

N of tests	Nb Frames:	Number of characters	Illumination, lux
1	18	3	980
2	20	4	900
3	21	5	860
4	22	6	840.
5	23	7	800
6	22	8	780
7	21	9	720

8	23	10	700
9	25	11	650
10	24	12	630
11	23	13	600
12	24	14	580
13	29	15	550
14	30	16	530
15	32	17	500
16	32	18	490
17	31	19	470
18	34	20	430
19	34	21	400
20	35	22	350

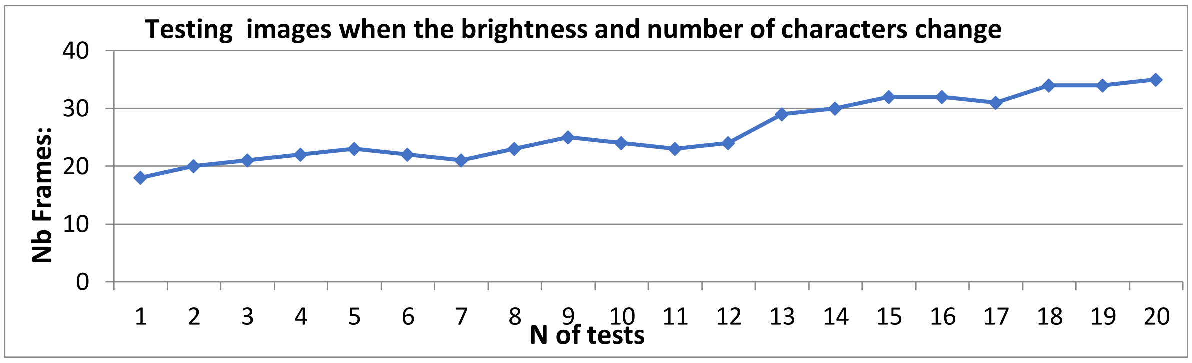


Figure 24: Test the image when changing the number of characters and changing the brightness (Series "Number of frames" as Element of the legend)

The research revealed that natural and technical conditions affect the algorithm's operation and its speed in text recognition. The darker the image or more characters, the more frames are needed to process it. To compare the quality and speed of the algorithm, we take the well-known algorithm OCR. A study of the operation of this algorithm is conducted on the same terms as when testing your product. Graphs are formed for a clearer vision of the research results. Twenty tests were also performed. The chart clearly shows that the OCR algorithm needs more time and frames to recognise the text. Based on the results, on average, you need one frame more than your development (Fig. 25).

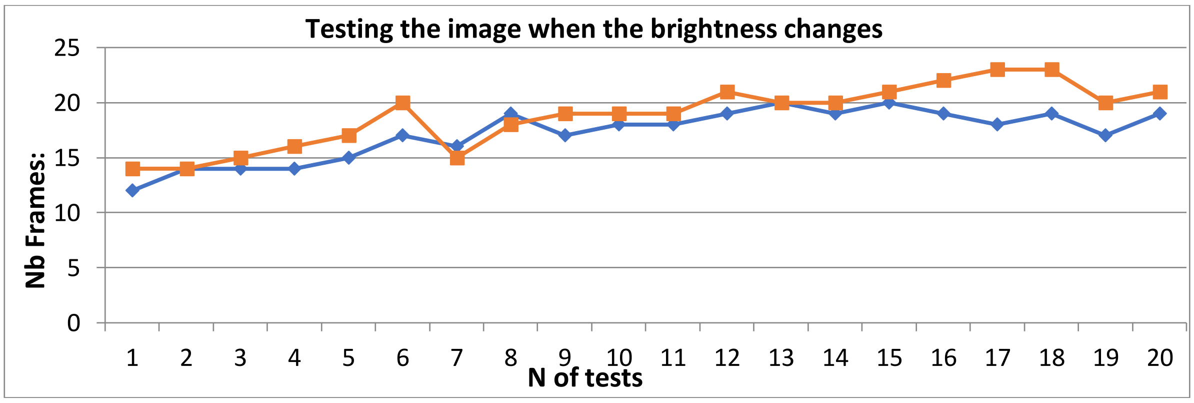


Figure 25: Testing the image when the brightness changes (Blue colour is improved algorithm, the orange colour is OCR algorithm)

Testing is also performed when changing the number of characters for both algorithms. The graph shows that text recognition when changing the number of characters is significantly different from tests when adjusting the brightness. The OCR algorithm takes more time and frames to recognise the text compared to its development. Here, the deviation is 1.5-2 frames (Fig. 26). The last test was testing algorithms based on changes in the number of frames and changes in brightness.

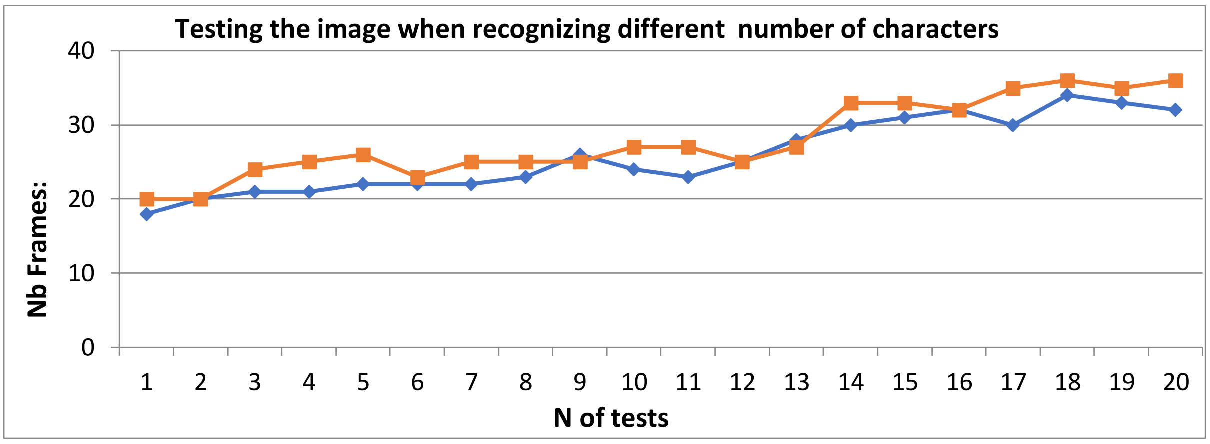


Figure 26: Testing the image when changing the number of characters (Blue colour is improved algorithm, the orange colour is OCR algorithm)

The graph shows that no notable differences are visible, but the difference between the graphs is still observed (Fig. 27). We can conclude that the improved algorithm increases productivity by 4-5% and if, more precisely, the text recognition is 1-2 frames less than the OCR algorithm.

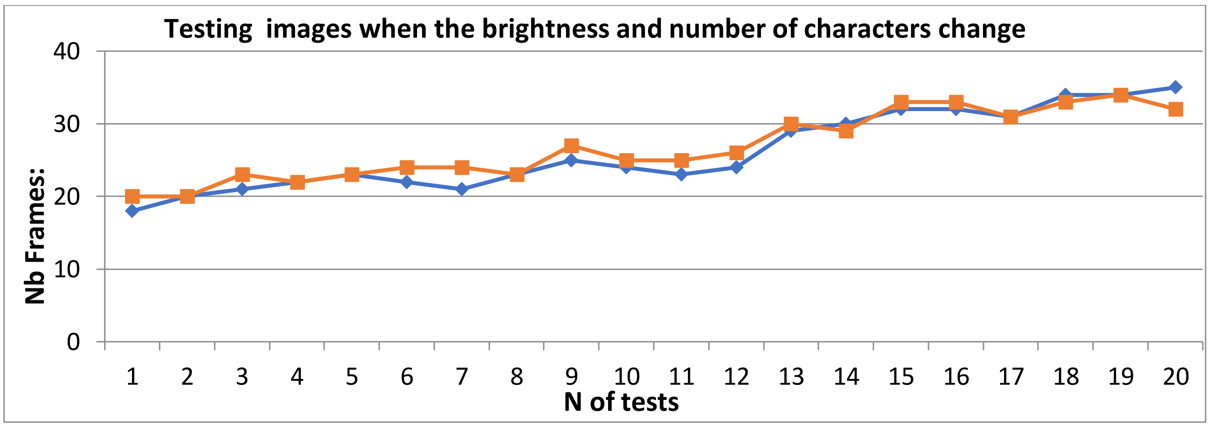
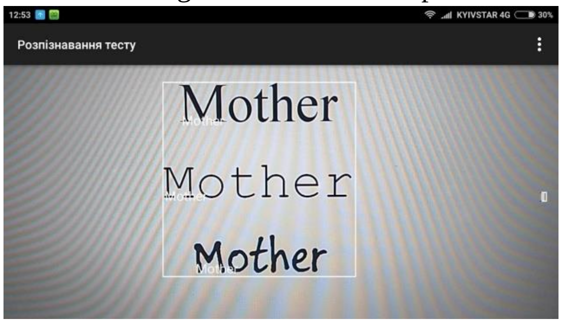


Figure 27: Testing the image when changing the number of characters and changing the brightness (Blue colour is improved algorithm; the orange colour is OCR algorithm)

Faster processing is achieved because the OCR algorithm checks each image. The phone, the frame changes at the slightest movement. And therefore, the text is lost that needs to be recognised. It also looks at how the program responds to different sizes and fonts. Fig. 28 shows how the program handles different fonts. Fig. 29 shows an example of text recognition at different sizes.



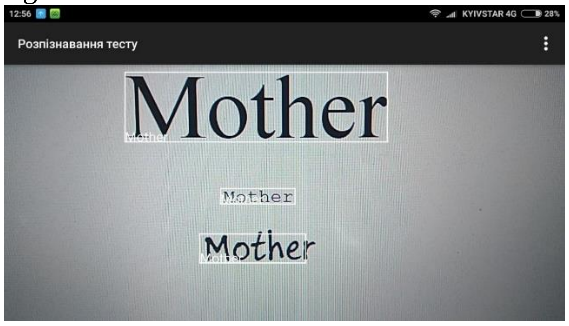


Figure 28: Text recognition in different fonts **Figure 29**: Text recognition at different sizes

We can conclude that the program works well with different types of text – different fonts and sizes.

7. Conclusion

The developed software has shown an improved high-speed algorithm for recognising printed texts and their sounding to implement them in mobile devices to help people with visual impairments and / or people who are learning the correct pronunciation of foreign words. The result of the research project:

- Implementation of a text recognition algorithm in video recording mode for people with visual impairments;
- Algorithm for recognition in Ukrainian and English in video recording mode;
- Implementation of the algorithm for sounding the text when you click on the screen;
- Image update and recognition of found text occurs at the number of frames (30-60 fps);
- Support for Android 4.0+;
- Implementation of highlighting the boundaries of the found text;
- Software for the Android operating system for the visually impaired and / or people learning the correct pronunciation of foreign words.

Studies have shown that the improved algorithm speeds up text recognition by 4-5%, which helps to recognise text better, but the effectiveness depends on external factors. A study is conducted on the accuracy of text recognition. The results showed that recognition accuracy depends on external factors, which impairs the recognition by about 40%. New functionality and features of the software product will be added for further research of the software product:

- Updating the user interface for easier use of the software;
- Further optimisation of the text recognition algorithm for video recording mode;
- Researching for the algorithm to assess the behaviour of the program when changing the screen tilt (from 0 to 90 degrees);
- Analysing with handwritten texts to analyse how the program will interact with them;
- Adding new languages for text recognition and sound.

8. References

[1] S., Deshpande, R. Shriram, Real time text detection and recognition on hand held objects to assist blind people, in: IEEE 2016 International Conference on Automatic Control and Dynamic Optimisation Techniques (ICACDOT), 2016, pp. 1020-1024.

[2] Z. Liu, Y. Li, F. Ren, W. L. Goh, H. Yu, Squeezedtext: A real-time scene text recognition by binary convolutional encoder-decoder network, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 32(1), 2018.

[3] Z. Cheng, Y. Xu, F. Bai, Y. Niu, S. Pu, S. Zhou, Aon: Towards arbitrarily-oriented text recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5571-5579.

[4] H. Li, P. Wang, C. Shen, G. Zhang, Show, attend and read: A simple and strong baseline for irregular text recognition, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33(01), 2019, pp. 8610-8617.

[5] F. Bai, Z. Cheng, Y. Niu, S. Pu, S. Zhou, Edit probability for scene text recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1508-1516.

[6] N. Shakhovska, O. Basystiuk, K. Shakhovska, Development of the Speech-to-Text Chatbot Interface Based on Google API, Vol-2386 of CEUR Workshop Proceedings, 2019, pp. 212-221.

[7] N. Grabar, T. Hamon, Automatic Detection of Temporal Information in Ukrainian General-language Texts, volume Vol-2136 of CEUR Workshop Proceedings, 2018, pp. 1-10.

[8] ABBYYFineReader 10 HomeEdition, 2020. URL: https://www.abbyy.com.

[9] A. P. Tafti, A. Baghaie, M. Assefi, H. R. Arabnia, Z. Yu, P. Peissig, OCR as a service: an experimental evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym, in: International Symposium on Visual Computing, Springer, Cham, 2016, pp. 735-746.

[10] M. Koistinen, K. Kettunen, J. Kervinen, How to Improve Optical Character Recognition of Historical Finnish Newspapers Using Open Source Tesseract OCR Engine–Final Notes on Development and Evaluation, in: Language and Technology Conference, 2017, pp. 17-30.

[11] F. Alkhateeb, I. A. Doush, A. Albsoul, Arabic optical character recognition software: A review, volume 27(4) of Pattern Recognition and Image Analysis, 2017, pp. 763-776.
[12] R. Giritharan, A. G. Ramakrishnan, Color Components Clustering and Power Law Transform for the Effective Implementation of Character Recognition in Color Images, in: Progress in Intelligent Computing Techniques: Theory, Practice, and Applications, 2018, pp. 131-139.
[13] OcrCuneiform, 2020. URL: https://softcatalog.info/ru/programmy/ocr-cuneiform.
[14] K. Yamauchi, H. Yamamoto, W. Mori, Building a handwritten cuneiform character imageset, in: Proceedings of the Eleventh Int. Conference on Language Resources and Evaluation, 2018.
[15] L. Rothacker, D. Fisseler, G. G. Müller, F. Weichert, G. A. Fink, Retrieving cuneiform structures in a segmentation-free word spotting framework, in: Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing, 2015, pp. 129-136.
[16] S. M. H. Mousavi, V. Lyashenko, Extracting old persian cuneiform font out of noisy images (handwritten or inscription), in: IEEE 10th Iranian Conference on Machine Vision and Image Processing (MVIP), 2017, pp. 241-246.
[17] CamScanner, 2020. URL: https://www.camscanner.com.
[18] C. P. Chandrika, J. S. Kallimani, Polarity Identification for Handwritten Text in Multilingual Documents Using Open Source Optical Character Recognition Tools, volume 17(9-10) of Journal of Computational and Theoretical Nanoscience, 2020, pp. 4045-4049.
[19] O. Bamasag, M. Tayeb, M. Alsaggaf, F. Shams, Nateq Reading Arabic Text for Visually Impaired People, in: Int. Conf. on Universal Access in Human-Computer Interaction, 2018, pp. 311-326.
[20] A. Maulidiyah, ,Translation Strategies of Noun Phrases with Derived Noun as Head in Academic Text, Doctoral dissertation, UNIVERSITAS 17 AGUSTUS 1945, 2018.
[21] N. Mannov, C. M. Lüders, A. Kaznin, ReqVision: Digitising Your Analog Notes into Readable and Editable Data, in: IEEE 4th International Workshop on Requirements Engineering for Self-Adaptive, Collaborative, and Cyber Physical Systems (RESACS), 2018, pp. 20-23.
[22] V. Lytvynenko, N. Savina, J. Krejci, M. Voronenko, M. Yakobchuk, O. Kryvoruchko. Bayesian Networks' Development Based on Noisy-MAX Nodes for Modeling Investment Processes in Transport, volume Vol-2386 of CEUR Workshop Proceedings, 2019, pp. 1-10.
[23] V. Lytvyn, V. Vysotska, P. Pukach, I. Bobyk, D. Uhryn, Development of a method for the recognition of author's style in the Ukrainian language texts based on linguometry, stylemetry and glottochronology, volume 4(2-88) of Eastern-European Journal of Enterprise Technologies, 2017, pp. 10-19.
[24] C. Shu, D. Dosyn, V. Lytvyn, V. Vysotska, A. Sachenko, S. Jun, Building of the Predicate Recognition System for the NLP Ontology Learning Module, in: International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS, 2, 2019, pp, 802-808.
[25] B. Rusyn, O. Lutsyk, R. Kosarevych, Y. Varetsky, Automated Recognition of Numeric Display Based on Deep Learning, in: Proceedings of 3rd International Conference on Advanced Information and Communications Technologies, AICT, 2019, pp. 244-247.
[26] Yu.M. Furgala, B.P. Rusyn, Peculiarities of melin transform application to symbol recognition, in: the 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET, 2018, pp. 251-254.
[27] B.E. Kapustiy, B.P. Rusyn, V.A. Tayanov, Peculiarities of application of statistical detection criteria for problems of pattern recognition, volume 37(2) of Journal of Automatioin and Inrormation Science, 2005, pp. 30-36.
[28] B.O. Kapustiy, B.P. Rusyn, V.A. Tayanov, A new approach to determination of correct recognition probability of set objects, volume 2, of Upravlyaushchie Sistemy i Mashiny, 2005, pp. 8-12.
[29] O. Veres, I. Rishnyak, H. Rishniak, Application of Methods of Machine Learning for the Recognition of Mathematical Expressions, volume Vol-2362 of CEUR Workshop Proceedings, 2019, pp. 378-389.
[30] V. Lytvyn, I. Peleshchak, R. Peleshchak, Increase the speed of detection and recognition of computer attacks in combined diagonalised neural networks, in: International Scientific-Practical Conference Problems of Infocommunications Science and Technology, 2018, pp. 152-155.

[31] P. Zhezhnych, O. Markiv, Recognition of tourism documentation fragments from web-page posts, in: 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET, 2018, pp. 948-951.
[32] P. Zhezhnych, O. Markiv, Linguistic comparison quality evaluation of web-site content with tourism documentation objects, volume 689 of Advances in Intelligent Systems and Computing, 2018, pp. 656-667.
[33] P. Zhezhnych, O. Markiv, A linguistic method of web-site content comparison with tourism documentation objects, in: International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT, 2017, pp. 340-343.
[34] N. Melnykova, O. Markiv, Semantic approach to personalisation of medical data, in: Computer Sciences and Information Technologies, CSIT, 2016, pp. 59-61.
[35] R. Martsyshyn, M. Medykovskyy, L. Sikora, (...), N. Lysa, B. Yakymchuk, Technology of speaker recognition of multimodal interfaces automated systems under stress, in: The Experience of Designing and Application of CAD Systems in Microelectronics, CADSM, 2013, pp. 447-448.
[36] V. Lytvyn, V. Vysotska, Y. Burov, O. Veres, I. Rishnyak, The Contextual Search Method Based on Domain Thesaurus, volume 689 of Advances in Intelligent Systems and Computing, 2018, pp. 310-319.
[37] J. Su, V. Vysotska, A. Sachenko, V. Lytvyn, Y. Burov, Information resources processing using linguistic analysis of textual content, in: Intelligent Data Acquisition and Advanced Computing Systems Technology and Applications, Romania, 2017, pp. 573-578.
[38] V. Vysotska, Linguistic Analysis of Textual Commercial Content for Information Resources Processing, in: Modern Problems of Radio Engineering, Telecommunications and Computer Science, TCSET, 2016, pp. 709-713.
[39] Lytvyn Vasyl, Vysotska Victoria, Dosyn Dmytro, Holoschuk Roman, Rybchak Zoriana, Application of Sentence Parsing for Determining Keywords in Ukrainian Texts, in: International Conference on Computer Sciences and Information Technologies, CSIT, 2017, pp. 326-331.
[40] V. Vysotska, V. Lytvyn, Y. Burov, P. Berezin, M. Emmerich, V. B. Fernandes, Development of Information System for Textual Content Categorizing Based on Ontology, volume Vol-2362 of CEUR Workshop Proceedings, 2019, pp. 53-70.
[41] V. Lytvyn, V. Vysotska, O. Veres, I. Rishnyak, H. Rishnyak, Classification methods of text documents using ontology based approach, volume 512 of Advances in Intelligent Systems and Computing, 2017, pp. 229-240.
[42] V. Lytvyn, V. Vysotska, I. Peleshchak, T. Basyuk, V. Kovalchuk, S. Kubinska, L. Chyrun, B. Rusyn, L. Pohreliuk, T. Salo, Identifying Textual Content Based on Thematic Analysis of Similar Texts in Big Data, in: Proceedings of the International Conference on Computer Sciences and Information Technologies, CSIT, 2019, pp. 84-91.
[43] O. Bisikalo, V. Vysotska, Linguistic analysis method of Ukrainian commercial textual content for data mining, volume Vol-2608 of CEUR Workshop Proceedings, 2020, pp. 224-244.
[44] V. Lytvyn, V. Vysotska, I. Budz, Y. Pelekh, N. Sokulska, R. Kovalchuk, L. Dzyubyk, O. Tereshchuk, M. Komar, Development of the quantitative method for automated text content authorship attribution based on the statistical analysis of N-grams distribution, volume 6(2-102) of Eastern-European Journal of Enterprise Technologies, 2019, pp. 28-51.
[45] V.-A. Oliinyk, V. Vysotska, Y. Burov, K. Mykich, V. Basto-Fernandes, Propaganda Detection in Text Data Based on NLP and Machine Learning, volume Vol-2631 of CEUR workshop proceedings, 2020, pp. 132-144.
[46] V. Vasyliuk, Y. Shyika, T. Shestakevych, Information System of Psycholinguistic Text Analysis, volume Vol-2604 of CEUR workshop proceedings, 2020, pp. 178-188.
[47] O. Artemenko, V. Pasichnyk, N. Kunanets, K. Shunevych, Using sentiment text analysis of user reviews in social media for e-tourism mobile recommender systems, volume Vol-2604 of CEUR workshop proceedings, 2020, pp. 259-271.
[48] I. Gruzdo, I. Kyrychenko, G. Tereshchenko, O. Cherednichenko, Applıcatıon of Paragraphs Vectors Model for Semantıc Text Analysıs, volume Vol-2604 of CEUR workshop proceedings, 2020, pp. 283-293.
[49] O. Kuropiatnyk, V. Shynkarenko, Text Borrowings Detection System for Natural Language Structured Digital Documents, Vol-2604 of CEUR workshop proceedings, 2020, pp. 294-305.

[50] M. Sazhok, V. Robeiko, R. Seliukh, D. Fedoryn, O. Yukhymenko, Written Form Extraction of Spoken Numeric Sequences in Speech-to-Text Conversion for Ukrainian, volume Vol-2604 of CEUR workshop proceedings, 2020, pp. 442-451.
[51] V. Lytvyn, S. Kubinska, A. Berko, T. Shestakevych, L. Demkiv, Y. Shcherbyna, Peculiarities of Generation of Semantics of Natural Language Speech by Helping Unlimited and Context-Dependent Grammar, volume Vol-2604 of CEUR workshop proceedings, 2020, pp. 536-551.
[52] R. Bekesh, L. Chyrun, P. Kravets, A. Demchuk, Y. Matseliukh, T. Batiuk, I. Peleshchak, R. Bigun, I. Maiba, Structural Modeling of Technical Text Analysis and Synthesis Processes, volume Vol-2604 of CEUR workshop proceedings, 2020, pp. 562-589.
[53] A. Taran, Information-retrieval System "Base of the World Slavic Linguistics (iSybislaw)" in Language Education, volume Vol-2604 of CEUR workshop proceedings, 2020, pp. 590-599.
[54] L. Chyrun, Model of Adaptive Language Synthesis Based on Cosine Conversion Furies with the Use of Continuous Fractions, Vol-2604 of CEUR workshop proceedings, 2020, pp. 600-611.
[55] T. Kovaliuk, N. Kobets, G. Shekhet, T. Tielysheva, Analysis of Streaming Video Content and Generation Relevant Contextual Advertising, volume Vol-2604 of CEUR workshop proceedings, 2020, pp. 829-843.
[56] I. Khomytska, V. Teslyuk, A. Holovatyy, O. Morushko, Development of methods, models, and means for the author attribution of a text, volume 3(2-93) of Eastern-European Journal of Enterprise Technologies, 2018, pp. 41-46.
[57] I. Khomytska, V. Teslyuk, Authorship and Style Attribution by Statistical Methods of Style Differentiation on the Phonological Level, volume 871 of Advances in Intelligent Systems and Computing III, AISC, Springer, 2019, pp. 105-118.
[58] V. Vysotska, V. Lytvyn, V. Kovalchuk, S. Kubinska, M. Dilai, B. Rusyn, L. Pohreliuk, L. Chyrun, S. Chyrun, O. Brodyak, Method of Similar Textual Content Selection Based on Thematic Information Retrieval, in: Proceedings of the International Conference on Computer Sciences and Information Technologies, CSIT, 2019, pp. 1-6.
[59] V. Vysotska, Ukrainian Participles Formation by the Generative Grammars Use, volume Vol-2604 of CEUR workshop proceedings, 2020, pp. 407-427.
[60] O. Bisikalo, V. Vysotska, Y. Burov, P. Kravets, Conceptual Model of Process Formation for the Semantics of Sentence in Natural Language, volume Vol-2604 of CEUR workshop proceedings, 2020, pp. 151-177.
[61] O. Cherednichenko, N. Babkova, O. Kanishcheva, Complex Term Identification for Ukrainian Medical Texts, volume Vol-2255 of CEUR Workshop Proceedings, 2018, pp. 146-154.
[62] N. Sharonova, A. Doroshenko, O. Cherednichenko, Issues of fact-based information analysis, volume Vol-2136 of CEUR Workshop Proceedings, 2018, pp. 11-19.
[63] U. Shandruk, Quantitative Characteristics of Key Words in Texts of Scientific Genre (on the Material of the Ukrainian Scientific Journal), volume Vol-2362 of CEUR Workshop Proceedings, 2019, pp. 163-172.
[64] Y. Bobalo, P. Stakhiv, B. Mandziy, N. Shakhovska, R. Holoschuk, The concept of electronic textbook "Fundamentals of theory of electronic circuits", volume 88(3 A) of Przeglad Elektrotechniczny, 2012, pp. 16-18.
[65] NB. Shakhovska, R.Yu. Noha, Methods and tools for text analysis of publications to study the functioning of scientific schools, volume 47(12) of Journal of Automation and Information Sciences, 2015, pp. 29-43.
[66] J. Pach, P. Bilski, A Robust Binarization and Text Line Detection in Historical Handwritten Documents Analysis, volume 15(3) of International Journal of Computing, 2016, pp. 154-161.
[67] T. Batura, A. Bakiyeva, M. Charintseva, A method for automatic text summarisation based on rhetorical analysis and topic modeling, volume 19(1) of International Journal of Computing, 2020, pp. 118-127.
[68] Google Codelabs, 2020. URL: codelabs.developers.google.com.
[69] Android, 2020. URL: www.andrious.com.
[70] Insights on Kotlin, 2020. URL: 101droid.wordpress.com.
[71] R. Seethalakshmi, T. R. Sreeranjani, T. Balachandar, A. Singh, M. Singh, R. Ratan, S. Kumar, Optical character recognition for printed Tamil text using Unicode, volume 6(11) of Journal of Zhejiang University-SCIENCE A, 2005, pp. 1297-1305.
[72] Get AI Code Completions for your IDE, 2020. URL: www.codota.com.