

Geometry Restoration and Dewarping of Camera-Captured Document Images

Valery Istomin¹ [ORCID](#), Oleg Pereziabov² and Ilya Afanasyev^{3,4} [ORCID](#)

¹ Shuya Branch of the Ivanovo State University, 155908, Shuya, Russia (pc_valery@mail.ru)

² BIA Technologies, 196210, St. Petersburg, Russia (pereziabov.oa@gmail.com)

³ Saint Petersburg Electrotechnical University "LETI", 197022, St. Petersburg, Russia (imafanasev@etu.ru)

⁴ Innopolis University, 420500, Innopolis, Russia (i.afanasyev@innopolis.ru)

Corresponding author: Ilya Afanasyev, ilya.afanasyev@gmail.com

This research focuses on developing a method for restoring the topology of digital images of paper documents captured by a camera, using algorithms for detection, segmentation, geometry restoration, and dewarping. Our methodology employs deep learning (DL) for document outline detection, followed by computer vision (CV) to create a topological 2D grid using cubic polynomial interpolation and correct nonlinear distortions by remapping the image. Using classical CV methods makes the document topology restoration process more efficient and faster, as it requires significantly fewer computational resources and memory. We developed a new pipeline for automatic document dewarping and reconstruction, along with a framework and annotated dataset to demonstrate its efficiency. Our experiments confirm the promise of our methodology and its superiority over existing benchmarks (including mobile apps and popular DL solutions, such as *RectiNet*, *DocGeoNet*, and *DocTr++*) both visually and in terms of document readability via Optical Character Recognition (OCR) and geometry restoration metrics. This paves the way for creating high-quality digital copies of paper documents and enhancing the efficiency of OCR systems. Project page: <https://github.com/HorizonParadox/DRCCBI>

Keywords: Document Image Dewarping, Image Distortions, Geometry Restoration

1 Introduction

Digital document management is becoming increasingly important in the modern world, permeating all areas of public life, including government institutions, business, trade, healthcare, education, etc. As private companies and government agencies strive for efficient information management, the number of studies related to document digitization, electronic document management, image processing, and restoration increases. Among other things, the drivers of this process include: Implementation of Electronic Document Management Systems (*EDMS*) [1]; Growing popularity of cloud solutions for document storage and exchange; Digitization of documents and books [2]. Such a transition from paper-based document management to digital methods is becoming increasingly common due to the convenience of transferring, storing, searching, and processing documents [3]. However, creating a high-quality digital copy can require specialized equipment that may not always be available. Therefore, there is a need for effective, cheap and convenient methods to produce digital copies of paper media.

With the development of portable devices equipped with high-quality cameras, the process of converting documents into digital format has become much easier. Nevertheless, document images captured by a user camera, unlike the controlled operating environment of a scanner, often suffer from problems with illumination, shadows, unstable shooting conditions, and camera positioning, as well as distortions associated with both camera lens distortions and the physical deformation of the paper at the moment of photographing [4,5,6]. These factors affect the quality of document digitization and increase the complexity of information retrieval when using machine methods for digital document processing. To eliminate the influence of distortions in document image processing, a variety of approaches have been proposed in the literature (see Section 2). However, the automatic dewarping of digital copies captured by cameras remains a challenging research problem in computer vision and pattern recognition. The purpose of this work is to develop an automated method for restoring the image topology of a hard copy of a document obtained using a digital camera. The object of the study is an image of a document from camera shots, and the subject of the study is a pipeline of algorithms for detection, segmentation, geometry restoration, and text recognition from a document.

We propose an original approach that integrates a deep learning (DL) method for simultaneous segmentation and recognition of documents in an image with classical computer vision (CV) methods for subsequent operations, such as interpolating the document surface with curved lines, approximating each line with a cubic polynomial to create a grid of interpolated lines, and correcting nonlinear distortions by remapping the input image according to the obtained grid to a uniform rectangular grid to restore the original document geometry. In this approach, the *YOLOv8* DL model is used only at the beginning of the algorithmic pipeline to detect document boundaries and create a mask. The document's geometry is then restored using classical CV methods. This makes the document topology restoration process more efficient and faster, as it requires significantly fewer computational resources and memory, and does not need specialized training like DL networks. Furthermore, we observe better document reconstruction quality (both visually and in terms of OCR-based text readability and document geometry restoration metrics) compared to modern desktop DL models and mobile applications. This opens up prospects for creating high-quality digital copies of paper documents and increasing the efficiency of OCR-based scanning systems.

In summary, we make three-fold contributions as follows:

1. We propose a document geometry restoration and dewarping methodology that initially employs DL for document outline detection, followed by CV methods to create a topological 2D grid using cubic polynomial interpolation, correcting nonlinear distortions by remapping the image.
2. We designed a new pipeline to automatically dewarping and reconstruct digital copies of a document captured by a camera. Additionally, we are releasing the framework and annotated dataset in open access for the research community and invite verification of our solution's efficiency.
3. We validated the advantages of our document topology reconstruction and dewarping methodology through experiments, demonstrating its effectiveness and superiority over existing state-of-the-art benchmarks both visually and in terms of OCR-based document readability and geometry restoration. We evaluated and ranked mobile apps and popular desktop DL solutions (such as *RectiNet*, *DocGeoNet*, and *DocTr++*) based on their document topology restoration and text readability quality.

The rest of the paper is structured as follows: Section 2 reviews the literature, discusses the challenges of document image distortions and methods for document geometry restoration, and also analyzes mobile applications for document scanning. Section 3 is devoted to the analysis of both classical CV and DL-based algorithms for document outline search. In Section 4, we detail the methodology and implementation of our proposed image document geometry restoration and dewarping algorithm, with examples. Section 5 presents the evaluation metrics used and a comparative analysis of document readability and geometry restoration for our approach versus popular mobile apps for correcting distorted images. Section 6 evaluates the quality of document image recovery by comparing the outcomes of our method with popular DL solutions, such as *RectiNet*, *DocGeoNet*, and *DocTr++*. Finally, Section 7 discusses and concludes, highlighting the superior quality of our document restoration method compared to state-of-the-art (SOTA) solutions.

2 Background

In this section, we review papers and methods that analyze geometric distortions in document images and the solutions applied for restoring document geometry. Additionally, we examine popular mobile applications for document scanning, highlighting the main features and drawbacks of the digitization they provide.

2.1 Challenges and Methods in Document Geometry Restoration

Analyzing geometric distortions in document images is a key area of research in document processing, image recognition, restoration, and computer vision. Geometric distortions encompass changes in the shape, orientation, and size of document elements caused by factors such as perspective distortion at various camera angles and paper surface curvature. Understanding and quantifying these distortions are crucial for various applications, including optical character recognition (OCR), document classification, and content extraction. There exist various methods for analyzing document images to dewarp geometric distortions, including:

- **Feature-based methods** that rely on detecting and matching key features, such as corners, edges, or line intersections, to estimate geometric transformation parameters. For instance, methods like Scale-Invariant Feature Transform (*SIFT*) [7], Speeded-Up Robust Features (*SURF*) [8], and Random Sample Consensus (*RANSAC*) [9] are widely used for these purposes.

- **Pattern matching**, where a pattern of document image is compared to a target image to find the best match, accounting for geometric transformations [10].
- **Deep learning approaches** that utilize Convolutional Neural Network (CNN) and Recurrent Neural Networks (RNN) to analyze geometric deformations [11,12].

Among these methods, DL approaches are increasingly used for document geometry restoration, as they can learn complex data dependencies and efficiently approximate nonlinear transformations. They are capable of automatically extracting objects from images and accurately restoring document geometry. However, the primary challenge with document geometry restoration using popular desktop DL methods is their lack of precision in delineating document boundaries, which often exacerbates image distortions due to blurred boundary boundaries. Additionally, applying OCR technologies to restored documents and counting the correctly recognized characters frequently reveals a high number of recognition errors (see Section 6.2). This poses a potential barrier to integrating such neural network-based document scanning technologies into electronic document management systems.

2.2 Related Work

Numerous studies focus on enhancing the dewarping of document images captured by mobile devices and improving OCR accuracy. As noted in [4, 13], there are learning-free and deep learning-based approaches. In traditional learning-free methods, after implicit or explicit warp estimation, distortion correction is performed, which can be evaluated using 2D and 3D document models. In our classification, we consider: (1) learning-free methods based on classical computer vision (CV) techniques, (2) hybrid methods, and (3) deep learning (DL) methods.

Among learning-free methods using CV approaches, we would like to highlight the following. The multistage curvilinear coordinate transform method [14] employs an iterative approach with curvilinear homography and quality estimation without the need for ground truth data. Unlike regular homography that works with flat surfaces, this method handles curved surfaces using mathematical models to correct distortions due to document curvature. Instead of comparing with ground truth data, it assesses the dewarping quality based on metrics like parallelism, orthogonality, and linearity of text lines and objects, making it more flexible and applicable when such data is unavailable. If the quality is unsatisfactory, the dewarping process is repeated with finer approximations. Research [13] uses a math model to automatically assess deformation factors in book pages, considering comic panel boundaries. The method detects and identifies the boundaries of such panels as key structural elements of the page, then builds a distortion model, evaluates deformation factors (including curvature, bends, and perspective distortions), and aligns the comic panels. The Grid Regularization technique [15] minimizes image distortions by enforcing a regular grid structure and using mathematical optimization to find the best grid point configuration. This ensures that grid lines (both horizontal and vertical) remain as straight as possible, helping maintain the alignment and readability of text and other document elements. The Geometric Control Points method [6] uses geometric elements like document boundaries and text lines to correct distortions. The key idea is to identify control points on the image to align and correct the document's shape. These points create a grid reflecting the document's geometry, enabling more accurate correction. Thin plate spline interpolation is then applied for smooth and precise transformation based on these control points. This technique is akin to bending a thin metal plate at several fixed points. In document image dewarping, thin plate splines create a smooth transition between control points, allowing for precise alignment and distortion correction, ensuring the image's integrity and quality. The Probabilistic Discretization of Vanishing Points method [16] calculates vanishing points to reconstruct the 3D shape of pages, dewarping documents and improving their readability and text recognition accuracy. Vanishing points (where parallel lines appear to converge in the distance) help determine the angles and slopes of document planes. This allows for the reconstruction of the 3D page shape from distorted 2D images. The method is based on geometric properties such as lines and edges, not on text content, so it does not require text recognition or segmentation, making it less sensitive to preprocessing errors.

Many solutions use a hybrid of DL and classical computer vision. For instance, [17] employs a semi-CNN approach, evaluating pixel position changes using deformation and control parameters, assessed by a CNN on a synthetically created dataset. The *Inv3D* approach [18] utilizes a new Inv3D dataset and structural templates, extending the existing *GeoTr* method to enhance invoice image dewarping using an attention mechanism [19]. The Text-Lines and Line Segments strategy [20] applies document image dewarping by searching for regions of interest (text lines and line segments). First, it detects these regions: horizontal lines for text alignment, table and picture borders, underlines, and other elements for document alignment. By integrating this information, the strategy creates a comprehensive understanding of the document layout and existing distortions, then corrects them. The process may be iterative, where initial corrections are re-evaluated for further improvements, ensuring reliable and precise dewarping. Fourier Document Restoration (*FDRNet*) [21] for robust document dewarping and recognition restores

documents using Fourier transformation and thin-plate splines (*TPS*), focusing on high-frequency components in the frequency domain for accurate structural data recovery. The camera-captured input image trains FDRNet to predict control points for dewarping using a Coarse and Refinement Transformer. These control points serve as nodes for Coarse and Refined Meshes. The network computes rectification losses (*L1 loss*) based on high-frequency information from the input images via a Fourier Converter and corresponding scanned documents, without using annotations during training. During inference, the dewarped document is fed to a Fourier Converter for photometric restoration and recognition. *FDRNet* is robust to irregular deformations and depth variations, requiring fewer annotated training data.

It is natural that many modern approaches use DL models exclusively for dewarping document images captured by mobile devices. Let's consider some notable publications. *DewarpNet* (Document Image Dewarping Network) [4] utilizes a combination of 3D and 2D regression networks for explicit modeling of the 3D shape of the document paper, and also introduces *Doc3D*, a large and comprehensive dataset for dewarping document images with various annotations. *DocUNet* (Document Unwarping Network) [22] uses a *U-Net* module with intermediate supervision to improve the accuracy of deformation grid predictions. It also presents a broad dataset of images with geometric distortions captured by mobile devices. *DocTr* (Document Image Transformer) [23] employs two transformers [19] for geometric dewarping (by capturing the global context of the document image) and illumination correction. The first transformer is used for geometric correction using a self-attention mechanism [19] and decodes the pixel shift solution to correct geometric distortions. The learned query embedding enables the transformer to capture the global context of the document image without distortions. The second transformer for illumination correction removes shadow artifacts post-geometric correction, enhancing visual quality and OCR accuracy. The approach of Adversarial Gated Unwarping Network [24] employs a pyramid encoder-decoder architecture with gated modules (to focus on significant visual features such as text lines, text blocks, and table lines) and adversarial training. These components work together to enhance geometric rectification of document images by reducing noise, ignoring insignificant details, and improving the accuracy of distorted image restoration. The Learning From Documents in the Wild [25] method employs two neural networks, Enet and Tnet, in its pipeline. Enet, a convolutional network, evaluates document edge information for coarse global dewarping and uses segmentation masks for weakly supervised training on real images. It then applies polyharmonic spline interpolation to approximate the inverse deformation field. Tnet refines the dewarping with local deformations learned from document texture. The *DocReal* [26] approach offers robust dewarping of real-life document images via an attention-enhanced control point (*AECP*) module, where Enet [25] performs edge-based dewarping to produce a coarse result by aligning the overall shape, and *AECP* then improves this result by predicting precise control points for local deformations. The final output is achieved through linear interpolation and remapping from the coarse result to the dewarped image, effectively handling various types of distortions. The Adaptive Dewarping on Document Map Generation method [27] fully automates the detection of control points (such as page corners, line intersections, text edges, and other key features) and the creation of document maps for adaptive dewarping. The document map provides a comprehensive understanding of the document's structure, simplifying the dewarping and alignment process, and helping restore the original document shape. Adaptive dewarping considers both global and local document features, correcting various distortions like bends, folds, and skewing, thereby improving readability and text recognition accuracy. The study [28] uses *CGAN* (Conditional Generative Adversarial Networks) to transform images from a distorted state to a corrected one. The network is trained on pairs of data: distorted image (input) and corrected image (target), featuring components like Generator, Condition, and Discriminator. Both Generator and Discriminator are trained simultaneously to enhance the quality of the generated corrected images. The Generator aims to create images from input distorted images that the Discriminator cannot distinguish from real ones, while the Discriminator compares the generated corrected images with real ones and improves its ability to differentiate between them. *CGAN* effectively eliminates distortions such as bends, folds, and skewing in document images, working with various types of documents without requiring image preprocessing, and preserving their original resolution. The Displacement Flow Estimation with *FCN* (Fully Convolutional Network) method described in [29] uses *FCN* to estimate pixel-wise displacements and correct distortions. *FCN* is effective for image segmentation and processing tasks as it preserves spatial information. It evaluates the displacements for each pixel in the distorted image, determining how and where each pixel should be moved. Using these displacement estimates, the image is corrected by repositioning the pixels appropriately. The *FCN* is trained on synthetically distorted images and their corrected versions, enabling the network to learn how to correct various distortions such as bends, folds, and skewing, effectively dewarping document images. Document Dewarping with Control Points [30] uses an encoder architecture to extract semantic information from the input deformed document image, predicting (a) control points and (b) reference points. Dewarping is achieved by matching control points with reference points and converting sparse mappings to dense ones through interpolation. This process fills gaps between sparse control points, turning them into a dense

representation (coordinates for each pixel) to accurately correct distortions. The method allows manual adjustment of suboptimal vertices for better results with limited data and can serve as a semi-automatic annotation tool for distorted document images. The Foreground and Text-lines approach [31] aims to rectify distorted document images by focusing on both the foreground (e.g., text and graphics) and text lines using CNNs Encoder, the Foreground and Text-line Attention Module, and the Transformer Decoder. This method combines global and local fusion with cross-attention to capture and correct features. Global fusion captures the overall context and structure, understanding the layout and relationships between elements. Local fusion focuses on detailed areas like individual text lines for precise corrections. Cross-attention allows the model to simultaneously focus on different parts, ensuring global corrections align with local details.

Finally, let's consider the popular DL models for document image dewarping: *RectiNet* [32], *DocGeoNet* [33], and *DocTr++* [34]. These models, selected as state-of-the-art benchmarks for our comparative performance analysis (Section 6), show promising results on the DocUNet dataset [22].

- *RectiNet* (A Gated and Bifurcated Stacked U-Net Module for Document Image Dewarping) [32]: Uses a CNN *U-Net* module with gating and bifurcation to predict deformation grids and correct perspective distortions and folds in document images. Trained on synthetic distorted images, it is evaluated on real-world images. Its novelty lies in bifurcating the *U-Net* to prevent grid coordinate mixing and using a gating network to add fine details to the model.
- *DocGeoNet* (Geometric Representation Learning for Document Image Rectification) [33] introduces explicit geometric representation by incorporating two key document image attributes: 3D shape and text lines. The 3D shape provides global cues for rectifying the distorted document image, while text lines offer local geometric features, enhancing correction quality.
- *DocTr++* (Deep Unrestricted Document Image Rectification) [34]: Employs a hierarchical encoder-decoder structure for multi-scale representation and pixel relationship reformulation for unrestricted distorted images. It handles documents with partially missing borders, improving rectification quality by enhancing image representation at various scales and redefining pixel relationships. This, along with new test datasets and metrics, allows for more effective training and rectification of diverse document images.

2.3 Analysis of Mobile Apps for Document Scanning

At the outset of our research, we explored existing mobile solutions available on the *Google Play Store* (in June 2023) and assessed the effectiveness of commercial applications for creating digital copies of documents using an Android smartphone camera. The aim of the analysis was to identify acceptable results for document geometry reconstruction. We used keywords such as "document scan" and selected the following four relevant applications from the list suggested: *DocScan* [35], *PDF scanner* [36], *TapScanner* [37], and *CamScanner* [38].

As part of the study, original images of randomly chosen documents (a crumpled medication package insert and a disease prevention leaflet) were uploaded to each of the selected applications. The documents were digitized automatically by the mobile apps without any manual intervention. The resulting images from the mobile applications are presented in Fig. 1 without additional processing.

Let's highlight the following features and drawbacks of digitization using these mobile applications (Fig. 2):

- Document search algorithms generally handle scenarios with foreign objects in the background successfully. However, false positives may occur if fragments of other documents are present in the image.
- Pages of an open book are often scanned as a single sheet and are not separated.
- For non-standard-sized documents (non-ISO format), detecting the edges of the page may be difficult even if there are no geometric distortions.
- Page edge detection may fail when the background color is similar to the document color or if the document has colored borders.
- When a document has geometric distortions, none of the evaluated mobile scanners can reliably restore the original document's geometry. This can lead to issues like missing parts of the digitized document or adding unwanted information (such as background) to the image. Additionally, in most cases, the image of the digitized document remains distorted.

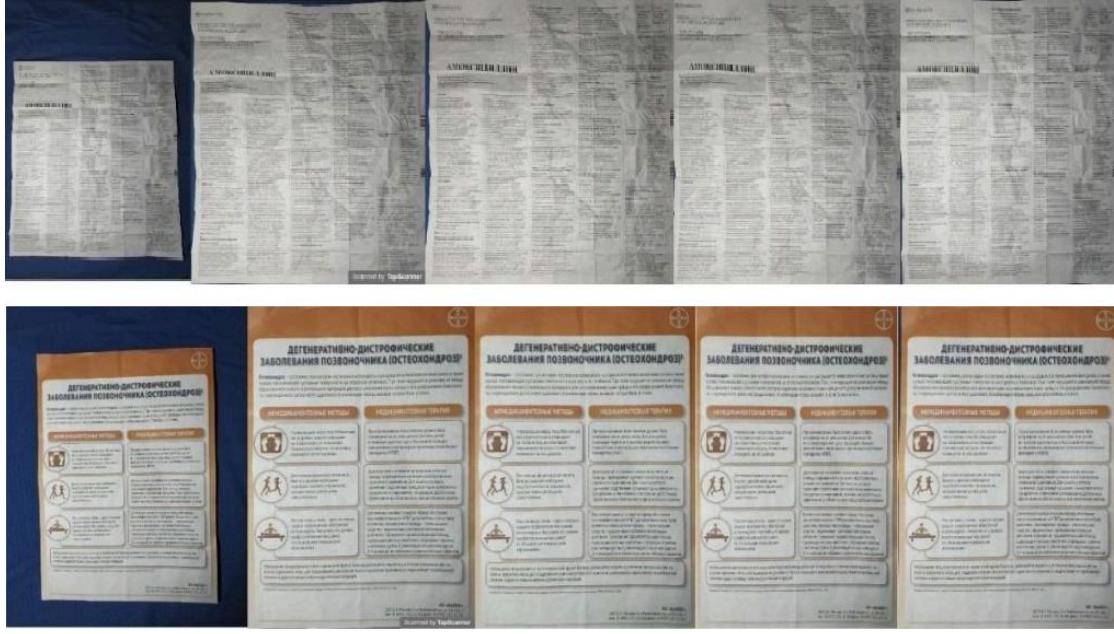


Fig. 1 The results of document digitization using mobile applications on an Android smartphone for a crumpled medication package insert (top images) and a disease prevention leaflet (bottom images). From left to right: the original image, images from *TapScanner*, *CamScanner*, *DocScan*, and *PDF Scanner*.

Final testing of the reconstructed documents, including OCR-based readability and geometry restoration evaluations, was conducted on the mobile applications *DocScan*, *TapScanner*, and *CamScanner*, and is described in Section 5.

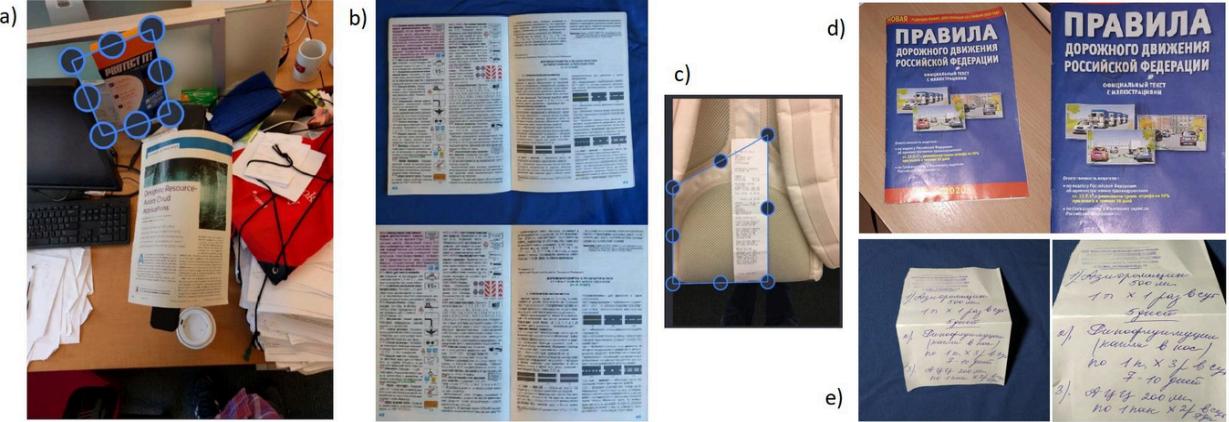


Fig. 2 Illustration of the features and limitations of current document geometry restoration methods in mobile app-based scanners: a) False document detection when other document fragments are present; b) Failure to separate individual pages in an open book spread; c) Incorrect boundary detection for non-standard sized documents; d) Cropping of the original document if it has colored borders; e) Inability to restore the original document geometry in the presence of significant paper deformations.

3 Analysis of Document Outline Detection Approaches

In this section, we explore document outline detection approaches using both classical computer vision algorithms and deep learning algorithms. We test document images and demonstrate the advantages of neural network approaches for document boundary detection with mask generation. After a comparative analysis of *YOLOv8* and *Mask R-CNN* for document detection, we choose the *YOLOv8* architecture for the dewarping and restoration pipeline of camera-captured document images.

3.1 Classical Computer Vision Approaches for Document Contour Detection

3.1.1 Edge Detection Algorithms for Document Outline Detection

We have explored edge detection algorithms to address the task of document boundary recognition. Edge detection algorithms in computer vision are represented by edge operators and detectors. These algorithms can detect object contours in an image by using intensity differences (gradients) between neighboring pixels. This allows them to highlight object boundaries and makes them applicable for outlining documents. The most widely used edge detection algorithms include the Roberts and Sobel operators, as well as the Canny detector [39].

The Roberts Cross operator is a gradient-based operator that calculates the sum of squared differences between diagonally adjacent pixels in an image through discrete differentiation, followed by gradient approximation using 2x2 diagonal kernels (also known as masks). By convolving the original image with these kernels, the Roberts operator quickly computes the two-dimensional spatial gradients on the image, effectively detecting edges, particularly diagonal ones. Similarly, the Sobel operator uses 3x3 kernels to compute brightness gradients in both horizontal and vertical directions. It divides the image into 3x3 blocks, computes the gradients and their directions, combines the results, and forms an image of contour gradients. The Canny operator, being more complex and precise, includes stages like blurring, gradient computation, Non-Maximum Suppression, and thresholding. The sequential steps involve blurring the image with a Gaussian filter, computing the gradient using the Sobel operator, applying thresholding to determine significant contour gradients, performing non-maximum suppression, and finally, using a two-threshold process to remove noisy contours and highlight the main contours.

Examples of these algorithms' performance in document edge detection are shown in Fig. 3. Initially, we used the Canny operator for edge detection in images. However, its universality was questioned due to the need for threshold adjustment based on lighting and color conditions. To address this drawback, we implemented algorithms based on the Sobel and Roberts operators, which do not require threshold tuning and are applicable to any image. The Roberts operator showed inferior results, while the Sobel operator demonstrated higher efficiency but faced challenges in contour detection on noisy images. Due to the low efficiency of these algorithms, we decided to discontinue the use of edge operators and detectors for our Document Outline Detection task.

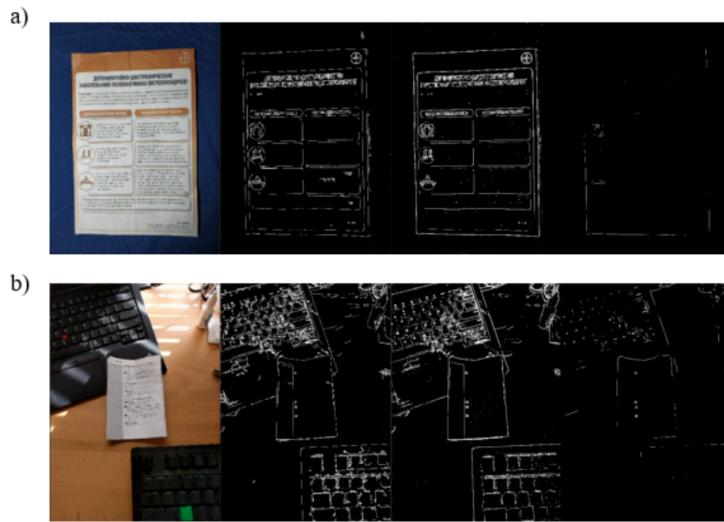


Fig.3 Examples of edge gradient algorithm performance (left to right: original image, Canny operator, Sobel operator, Roberts operator).

3.1.2 Superpixel Algorithms for Document Outline Detection

Superpixel algorithms are image segmentation methods that divide an image into compact and interconnected regions known as superpixels. Superpixels are clusters of pixels with similar colors and textures, combined into a single unit for image processing. At this stage of the research, we considered four superpixel algorithms: *SLIC* [40,41], *SEEDS* [42], *Felzenszwalb* [43], and *Quickshift* [44].

- The *SLIC* algorithm combines cluster analysis and dimensionality reduction methods. It divides the image into rectangular regions, each containing roughly the same number of pixels. The algorithm then performs clustering in color and spatial feature spaces to identify superpixels, resulting in a set of superpixels that correspond to connected areas in the image [40,41].

- The *SEEDS* algorithm is based on iterative assignment of pixels to superpixels. Initially, it divides the image into blocks and iteratively refines the boundaries of superpixels [42].
- The *Felzenszwalb* algorithm merges regions of similar pixels into superpixels. It uses a connectivity measure based on the difference in pixel intensity and image gradients [43].
- The *Quickshift* algorithm identifies superpixels based on local pixel density and their color properties. It employs dimensionality reduction and smoothing techniques to determine the density map of the image. The algorithm then locates local extrema on this map to define superpixel centers and expands these centers to form the final superpixels [44].

All four types of superpixel algorithms were tested to compare their performance and suitability for document segmentation tasks. Since the SLIC algorithm demonstrated the best results among the other superpixel algorithms, it was chosen for further testing to detect document boundaries. To implement this algorithm, we selected the open-source project *Fast-SLIC* [45], developed in C++, which operates 7-20 times faster than other existing implementations. Based on this implementation, a document mask search function was developed. This function takes an image as input, applies morphological closing to remove small details and noise, and then performs *SLIC* segmentation to create a segmented image. The segmented image and its mean value are passed to the function, which returns an image representing the per-pixel average of each segment. To enhance the contrast between the document and the background, the resulting image is converted to the *HSV* color model. A Gaussian filter and thresholding are then applied to create a binary mask. An example of the algorithm's operation is shown in Fig. 4.

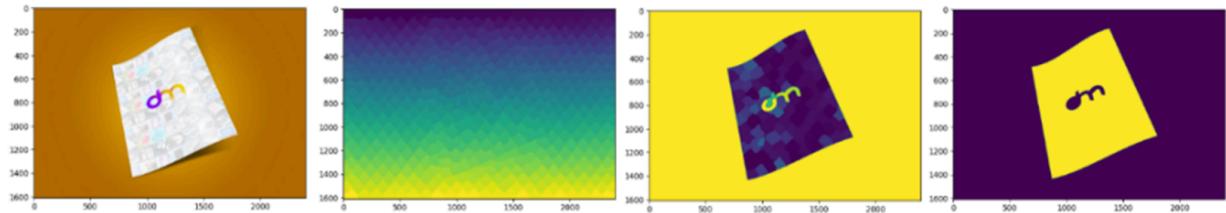


Fig. 4 Example of generating a binary document mask using the *Fast-SLIC* algorithm.

3.1.3 Conclusion on the Use of Edge Detection and Superpixel Algorithms for Document Outline Detection

During testing, we found that the reviewed edge detection algorithms are ineffective when processing images that contain extraneous objects along with the scanned document. The presence of such objects, e.g., stationery items, can lead to erroneous document image processing, complicating the unambiguous identification of its contours. Moreover, the algorithms are limited in their ability to handle documents with significant geometric deformations, leading to distortion of the document's quadrilateral shape. In contrast to edge detection algorithms, the *SLIC* superpixel algorithm demonstrates greater stability and predictability in document detection tasks. It effectively handles anomalous document shapes, noise, and varying lighting conditions, ensuring more accurate detection of document contours, even when the document is not fully visible. However, the *SLIC* algorithm has drawbacks, such as increased processing time compared to other methods and the need for careful tuning of optimal parameters. Despite its efficiency, it does not address the issue of extraneous objects in the image, which can lead to incorrect document mask construction.

As a result, we abandoned the use of edge detection and superpixel algorithms, as they are ineffective in the presence of extraneous objects in the frame and require manual parameter tuning. Instead, we decided to focus on exploring deep learning approaches.

3.2 Deep Learning-Based Document Outline Detection Algorithms

3.2.1 Dataset Preparation for Using DL Approaches

To tackle the task of document border detection using neural networks, it was necessary to prepare a training dataset. For this purpose, we compiled a dataset that includes several subsets with images of documents. The basis for the dataset was the *DocUNet* dataset [22,46], which includes 130 images of documents with various deformations and their corresponding scanned versions. Additionally, we utilized images from the widely-used *COCO* dataset [47], which contains over 330,000 images of diverse objects, scenes, and contexts. Although *COCO* does not have a specific "document" class, it includes a "book" class that features both book and document images. From this class, we initially obtained 5000 images, from which we manually selected 72 where the book or document is fully visible

in an open state. *SmartDoc-QA* [48] serves as an additional data source, containing 37 different documents captured at various angles, under different lighting conditions, and with varying image sharpness.

The remaining document photos were sourced from various internet resources and supplemented with personal smartphone images. A total of 392 document images were collected, with 10 used for testing, 23 for validation, and 359 for training. The dataset annotation was carried out using the *MakeSense* service [49], where images were manually annotated using polygons and exported in the *COCO* format. The annotation included only one class - "document." The dataset preparation was streamlined using the *Roboflow* service [50]. This service was used to convert all collected images to *JPEG* format and create annotations in two formats: *COCO* and *YOLOv8* [51], which were subsequently used for the *Mask R-CNN* [52] and *YOLO* models, respectively.

3.2.2 Mask R-CNN for Document Outline Detection and Masking

For tasks involving image classification or single-object detection, simple CNNs are typically used. However, these may be insufficient for more complex scenarios with multiple objects in an image. For such scenarios, an open-source library for object detection in images and videos based on *PyTorch*, *MMDetection* [53], can be employed. It includes various state-of-the-art object detection algorithms, such as *Faster R-CNN* [54], *Mask R-CNN* [52], *RetinaNet* [55], *Cascade R-CNN* [56], and others.

For our research, we selected the *Mask R-CNN* algorithm, which allows simultaneous segmentation and recognition of multiple objects in images. The *Mask R-CNN* model is developed based on *Faster R-CNN*. Unlike *Faster R-CNN*, which has two outputs for each object candidate - a class label and a bounding box offset - *Mask R-CNN* adds a third branch that generates an object mask. Generating an additional mask differs from generating a class and bounding box, as it requires a more accurate spatial representation of the object. The *Mask R-CNN* algorithm consists of two primary stages, similar to *Faster R-CNN* [54]. The first stage is the Region Proposal Network (*RPN*), which suggests candidates for object bounding boxes. The second stage extracts features using the *RoIPool* operation for each candidate box and performs classification and bounding box regression. The shared features used in both stages can be separated to improve computational efficiency. In addition to class and bounding box predictions, *Mask R-CNN* also generates a binary mask for each Region of Interest (*RoI*).

3.2.3 YOLOv8 for Document Outline Detection and Masking

It is known that *YOLO* is architecturally a convolutional neural network designed for real-time simultaneous object detection and classification in images. *YOLOv8* [51] supports a wide range of computer vision tasks, including detection, segmentation, pose estimation, tracking, and classification.

The *YOLO* algorithm operates as follows:

1. The input image is divided into a grid of fixed-sized cells, each responsible for detecting objects.
2. For each cell, the model generates predictions of objects, determining their position, size, and class.
3. Predictions are filtered using a threshold value.
4. The class of the object with the highest probability is determined for each prediction.
5. Predictions that overlap and belong to the same object are merged.
6. The final result of *YOLO* includes the coordinates of bounding boxes for the detected objects and their respective classes.

Segmentation in *YOLOv8* is implemented in the *YOLOX-Seg* module, which consists of several deconvolution layers and decoding blocks. To improve segmentation quality, *YOLOv8* applies augmentations such as changes in brightness, contrast, and color saturation. It also uses a pretrained object detector to generate preliminary predictions. For each prediction, features are extracted using the deconvolution method, which allows the expansion of the prediction size to the original image size. These features are fed into a network consisting of decoding blocks, sequentially increasing the original image size and generating a segmentation mask as the output.

3.2.4 Comparative analysis of Mask R-CNN and YOLOv8

We conducted a comparative analysis of *Mask R-CNN* and *YOLOv8* to address the task of document boundary detection followed by mask generation. Initially, we employed an object detection technique using the *Mask R-CNN* architecture. This approach enables the direct extraction of object masks during the detection phase. We opted to train the neural network on a specially curated dataset from various sources (see Section 3.2.1), using a pre-trained

model chosen for further refinement. The training was carried out on *Google Colab* for a duration of 100 epochs, and the outcomes of the trained model are illustrated in Figure 5.

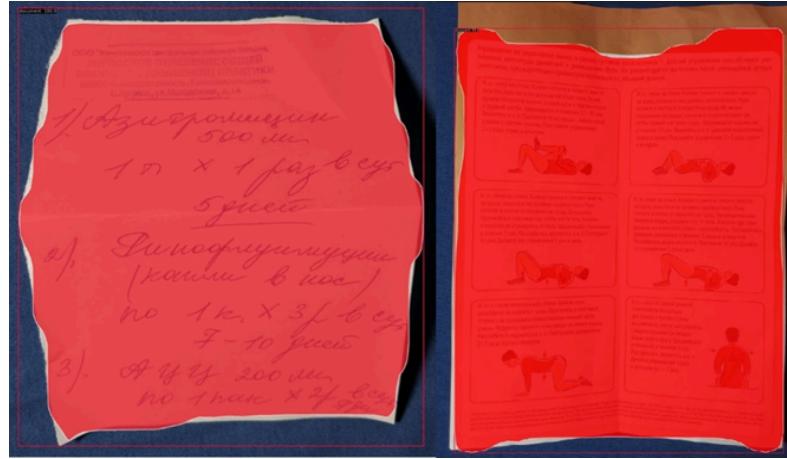


Fig. 5 Examples of *Mask R-CNN* performance on a fine-tuned model (100 epochs).

The analysis identified inaccuracies in the document boundary detection using the *Mask R-CNN* method. In many cases, the generated mask includes extraneous fragments or omits parts corresponding to the document. When a book is present in the image, the algorithm successfully identifies individual page masks. Additionally, the algorithm effectively processes maps and checks with atypical shapes.

Subsequently, we decided to train the *YOLOv8* algorithm, utilizing the pre-trained *YOLOv8x-seg* model for 100 epochs on *Google Colab*. Examples of the results are displayed in Figure 6.

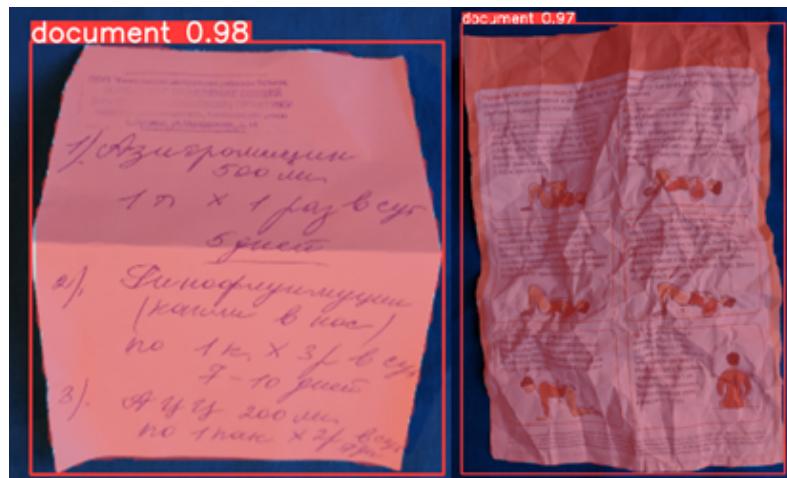


Fig. 6 Examples of *YOLOv8* performance, using the pre-trained *YOLOv8x-seg* model (100 epochs).

When comparing these two DL approaches, a shortcoming of *Mask R-CNN* was identified, manifesting in the imprecise shape of mask boundaries (Fig. 7). Even with minor geometric distortions or tilting of the document, the edges of the mask can appear wavy. In contrast, *YOLOv8* is less susceptible to such distortions, providing more accurate mask boundary shapes regardless of the document's geometry.

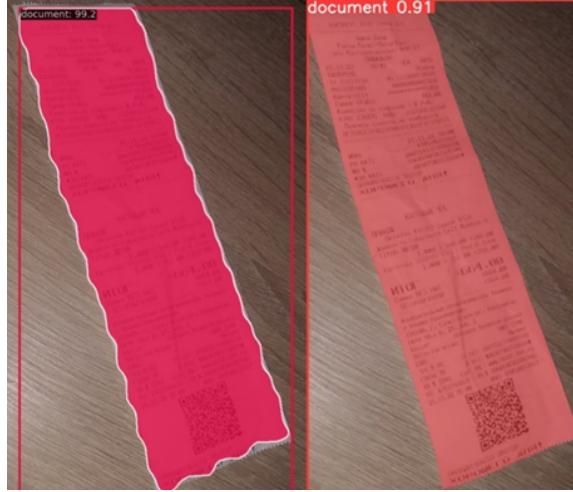


Fig. 7 Comparison of document boundary and shape detection results for *Mask R-CNN* (left) and *YOLOv8* (right).

An additional benefit of *YOLOv8* is the availability of the *ultralytics Python* library for convenient result processing, whereas with *Mask R-CNN*, it is necessary to download the project from *GitHub* and use extra libraries, complicating integration and data processing. Based on the conducted comparison, we chose the *YOLOv8* architecture for solving the document detection task in images. We also experimented with training the *YOLOv8* model for 150 epochs and compared the results with training for 100 epochs (shown in Fig. 8), which demonstrated better achievement of optimal document boundary segmentation for the 150 epochs.

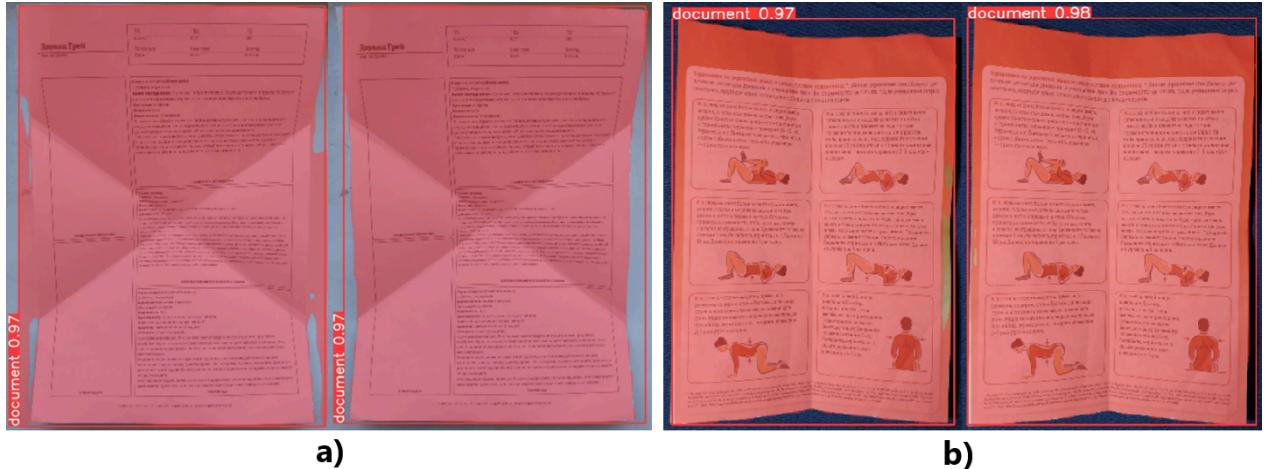


Fig. 8 Comparison of *YOLOv8* segmentation on two documents (a) and (b), trained for 100 epochs (left) and 150 epochs (right). The figures indicate that undertraining results in defects: visible on both documents at 100 epochs, with excessive space in document image (a), and necessary space missing in document (b).

4. Methodology for the Document Geometry Restoration and Dewarping Algorithm

In this section, we thoroughly describe the methodology of our algorithmic pipeline for correcting distortions and restoring the topology of camera-captured document images. We demonstrate the functionality of our algorithmic pipeline with specific examples of document images.

4.1 Algorithm Overview

To address the task of obtaining a digital copy of a document using a smartphone camera, we developed a dewarping algorithm for document images based on reconstructing the document's topology using information about its shape

and edges. This method leverages a combination of traditional computer vision techniques and neural networks, as well as binary search, interpolation, and approximation.

The algorithm comprises the following sequential steps (see Fig. 9):

1. Defining a document mask using the *YOLOv8* model.
2. Detecting the document contour edges, followed by edge approximation and calculating the coordinates of the document corners within the image.
3. Segmenting the document contour into fragments corresponding to the individual sides of the document.
4. Constructing a topological 2D grid of the document by interpolating its opposite sides with evenly spaced curved lines.
5. Approximating and extrapolating each curved line with a cubic polynomial.
6. Locating the intersection points of the detected curved lines within the 2D grid.
7. Building the resulting grid of points to which the document image needs to be transformed.
8. Creating an image transformation map based on two sets of points in 2D space.
9. Remapping the original image from the interpolation grid to a uniform grid using the obtained transformation map.

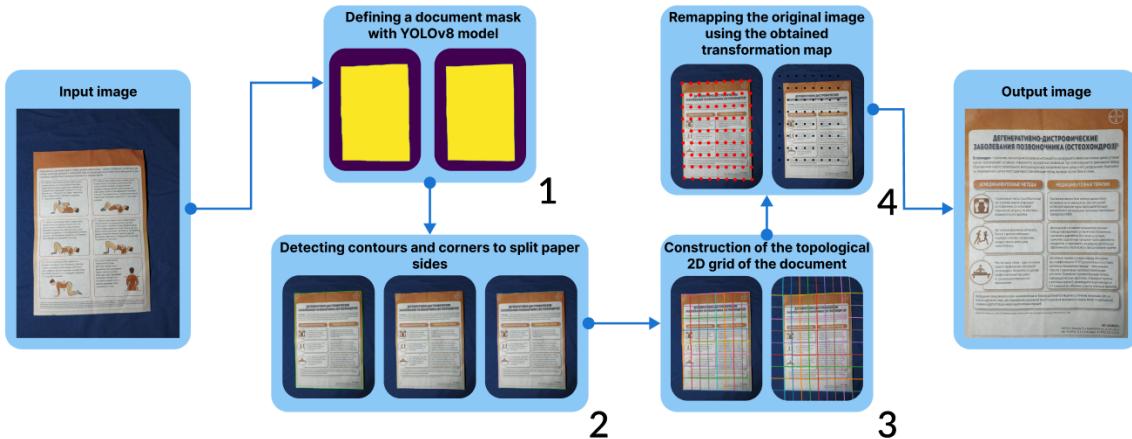


Fig.9 The flowchart of the document geometry restoration and dewarping algorithm: 1) Identifying the document mask using the *YOLOv8* model; 2) Detecting the contour edges of the document, approximating the corners, and segmenting the contour into fragments corresponding to each side of the document; 3) Creating a 2D grid of the document by interpolating its opposite sides with evenly spaced curved lines, approximating each line with a cubic polynomial; 4) Detecting the intersection points of the curved lines, constructing the resulting grid for image transformation, and creating a transformation map based on the 2D points, followed by remapping the original image using this map.

Next, we will delve deeper into the primary steps of the algorithm and explore the implementation details more thoroughly.

4.2 Document Mask Detection Using the *YOLOv8* Model

In the initial stage of the document restoration and dewarping algorithm, the document mask is identified utilizing the *YOLOv8* model. Detection results encompass bounding boxes, segments, and confidence scores. In instances where multiple documents are detected within an image, the algorithm selects the two documents with the highest scores and checks for intersections between their segments. If an intersection is detected, both documents are included in further analysis, assuming they represent two pages of an open book. If no intersection is found, the document with the highest confidence score is considered.

Next, each document undergoes processing. Initially, based on segments obtained via *YOLO*, the document contours are constructed, and the detected document mask is visualized. Subsequently, a guided filter [57] is applied to the mask. This filter operates on the principle of a hybrid filter, which merges information from the input image (also referred to as "guidance") and the target image (Eq. 1):

$$Q(i) = a(i) \times I(i) + b(i), \quad (1)$$

where $I(i)$ represents the input image; $a(i)$ denotes the filter strength coefficient; and $b(i)$ signifies the radius of the filter window. The primary concept behind guided filtering is to transfer structural information and details from the guiding image to the target image. This approach allows for the control of blur levels while preserving edges and textures, leading to detail retention and a more natural visual perception. The window radius $b(i)$ determines the size of the window for pixel value averaging and is set at 1% of the minimum side of the mask. The second parameter, the strength coefficient $a(i)$, is calculated using a formula [57] that includes the smoothing coefficient ε (in the denominator), which controls the level of pixel blur while preserving edge sharpness, and is defined as half of the radius. These values were chosen based on our observations.

The subsequent step of the algorithm involves threshold binarization of the filtered mask using the Otsu method. In its simplest form, the method returns a single intensity threshold that segregates pixels into two classes: foreground and background. This threshold is determined by minimizing the intra-class intensity variance or equivalently, maximizing the inter-class variance (Eq. 2):

$$\sigma_{\omega}^2 = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t), \quad (2)$$

where $\omega_0(t)$ and $\omega_1(t)$ represent the probabilities of the two classes divided by threshold t , and σ_0^2 and σ_1^2 signify the variances of these two classes. Thus, a list of all the filtered document masks is generated, and their visualization is performed (Fig. 10).

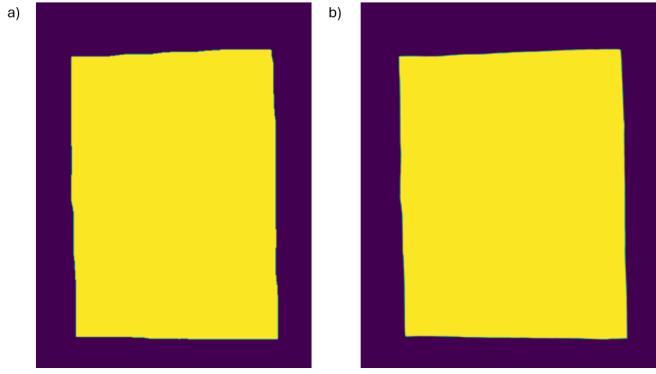


Fig. 10 a) Example of a mask obtained using *YOLO*; b) mask after the application of the guided filter. Enlarged fragments are displayed in the corners of the images.

4.3. Polynomial Estimation for Document Mask

The list of detected masks is utilized in the polynomial search function. The contour of the largest mask by area is identified using an algorithm based on the edge-following method for topological structural analysis of digital binary images [58]. Subsequently, the value of the ε smoothing coefficient for polynomial curve approximation is calculated as the product of the threshold coefficient and the contour length (the perimeter of the closed contour). The contour is then approximated by a polynomial curve with fewer vertices so that the distance between them is less than or equal to the specified precision. The Douglas-Peucker algorithm [59] is employed for this purpose, serving to simplify geometric shapes, such as lines or polygons, by removing some points.

Finally, the convex hull of the contour is computed using the Sklansky algorithm [60]. This algorithm is used to calculate the convex hull of a point set and has a complexity of $O(N \log N)$, where N is the number of points in the input set. As a result, the function returns a list containing the outcomes of the polynomial curve approximation, convex hulls, and contours for each document mask. Fig. 11 (left) displays the visualization of the contour for the obtained document mask.

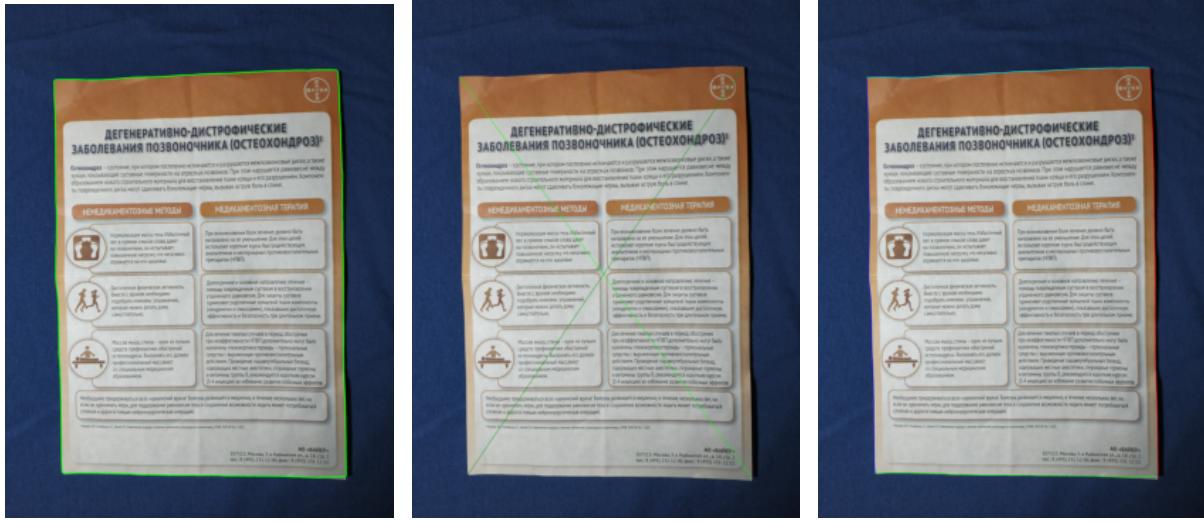


Fig.11 Digital document processing: (left) Visualization of the document contour utilizing polynomial curve approximation; (middle) Constructing diagonals; (right) Edge detection.

4.4. Detection and Interpolation of Document Edges

Once the document contour is obtained, it is segmented into parts corresponding to each side of the document. Initially, the coordinates of the edges and corners are extracted from the contour and convex hull arrays. Then, the corners are sorted in a specific order. This involves calculating the center of the figure by averaging the coordinates along the X and Y axes. The list of corner coordinates is then ordered based on the increasing angle formed between the direction from the center of the figure to each point and the positive Y-axis. This is achieved by computing the arctangent of the ratio of the difference between the X and Y coordinates for each point to the difference between the coordinates of the figure's center. Diagonal lines connecting the corners are then formed, as visualized in Fig. 11 (middle).

Next, the edges lying to the left of the diagonal lines are highlighted. This is achieved by calculating the cross product between each diagonal line and each point on the polygon's edge. For instance, if the cross product result is negative for one diagonal line and positive for another, then the edge point is to the left of the diagonal lines. Similarly, the cross product is calculated for each edge of the polygon. Then, the arrays of points forming the top edges are sorted in ascending order. The order of points for the right edges is reversed as shown in Fig. 11 (right).

The dimensions of the polygon in the image are then calculated. The length of each polygon side is determined using the Euclidean distance function (3):

$$\sqrt{\sum_{k=1}^n (p_k - q_k)^2}, \quad (3)$$

where p_k and q_k are the coordinates of vectors p and q ; k is the index of the vector coordinate; n is the dimensionality of the vectors.

Next, the average length and width of the polygon are computed by finding the arithmetic mean of the corresponding side lengths. Subsequently, new coordinates for the opposite sides of the polygon are calculated using linear interpolation:

$$f(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0} (x - x_0), \quad (4)$$

where x_0 and x_1 are the coordinates of adjacent points; x is the coordinate of the intermediate point for which interpolation is performed; $f(x_0)$ and $f(x_1)$ are the values of the function at points x_0 and x_1 , respectively.

The obtained coordinates are then smoothed using the Savitzky-Golay filter [61], which is a linear filter used for smoothing time series or one-dimensional signals. This is achieved, in a process known as convolution, by fitting successive sub-sets of adjacent data points with a low-degree polynomial by the method of linear least squares:

$$Y_j = \sum_{i=\frac{1-m}{2}}^{\frac{m-1}{2}} C_i y_{j+i}, \quad \frac{m+1}{2} \leq j \leq n - \frac{m-1}{2}, \quad (5)$$

where Y_j is the value of the element after applying the filter; y_{j+i} are the original values of the array elements; C_i are the filter (convolution) coefficients dependent on the chosen filter window width m .

4.5. Construction of Approximation Grid and Its Interpolation

The x and y coordinates obtained earlier (see section 4.4) are used to generate approximation lines on the image. These lines form the basis for creating a document grid, defining its topology. The number of lines is a parameter of the algorithm and can be set manually. Intermediate grid lines are constructed by linearly interpolating the coordinates of the lines representing the sides of the document image. Approximations of the coordinates of lines passing through the left and right sides (vertical) as well as through the bottom and top sides (horizontal) are calculated. An image of the document with the constructed grid lines is shown in Fig. 12 (left).



Fig. 12 Building interpolated lines on the document image: (left) Constructing the grid from interpolated lines; (right) Grid of lines extrapolated with polynomial approximation.

Subsequently, for each line constituting the grid, approximation and interpolation are performed individually. Each line is approximated using the method of nonlinear least squares with variable constraints (Eq. 6) and a cubic polynomial (Eq. 7):

$$F(x) = \frac{1}{2} \sum_{i=0}^{m-1} \rho f_i^2(x), \quad (6)$$

where ρ is the loss function; $f_i(x)$ represents the difference between the observed values and the values predicted by the approximating function; m is the dimensionality of differences.

$$P(x) = ax^3 + bx^2 + cx + d, \quad (7)$$

here, a, b, c, d are the coefficients of the polynomial.

The results of the approximation are stored as arrays of coordinate points. The function then performs extrapolation of the approximated lines to obtain additional points on each line. For horizontal lines, extrapolation is performed along the Y-axis, while for vertical lines, it is performed along the X-axis. This process increases the number of points on each line and ensures their intersection. Fig. 12 (right) illustrates the approximated and extrapolated grid lines overlaid on the original image.

4.6. Searching for Intersections of Interpolation Lines

Based on the arrays of coordinate points that form the grid lines, the process of finding intersection coordinates between the approximated lines in the image is carried out. This involves identifying the nearest points among all pairs of vertical and horizontal grid lines. To implement the intersection search for each pair of grid lines, a method

utilizing the k-d tree algorithm [62] is employed. A k-d tree is a data structure used for organizing multidimensional points in space to accelerate nearest neighbor searches [63]. The efficiency of this algorithm was the primary reason for its selection. A k-d tree is a binary tree where each node represents a hyperplane that divides the space along one of the axes. Each node of the k-d tree stores a point, and the left and right subtrees contain points located on opposite sides of the dividing hyperplane. This partitioning is based on comparing the coordinates of points in multidimensional space. A query is then made to find the nearest neighbors between the trees using a threshold value set at 1% of either the length or width of the image, whichever is greater. The indices of the identified neighbors are then combined into a single list, with duplicate indices removed. Subsequently, the average values of the coordinates along the X and Y axes are calculated to determine the intersection point of the two lines. The result of this part of the method is an array of coordinate points representing the intersections of the grid lines found earlier, as illustrated in Figure 13 (left).

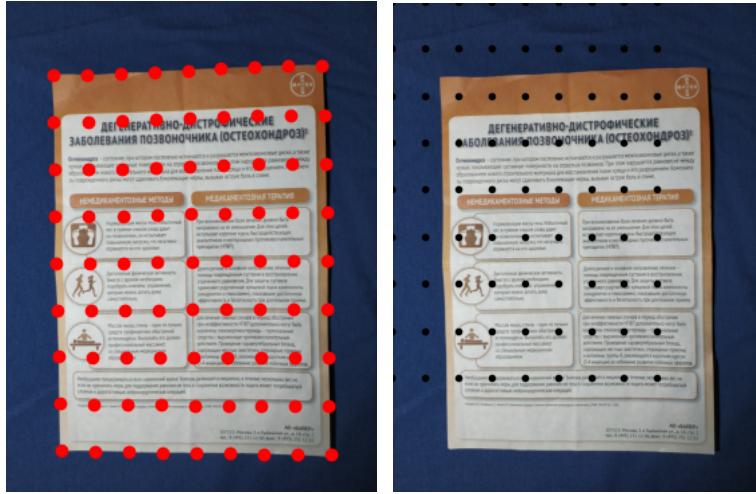


Fig.13 Visualization of coordinate point arrays on the document image: (left) Intersection points approximated by lines derived using the k-d tree algorithm; (right) A uniform rectangular grid overlaying the document image.

4.7. Construction of Aligned Grid and Intersection Points

For further image processing, it is necessary to create an array of evenly spaced coordinates on a rectangular grid that covers the specified document. Arrays of point coordinates along the X and Y axes are created in quantities corresponding to the width and length of the original image. The points of the resulting grid are displayed in Fig. 13 (right).

4.8. Remapping from the Interpolation Grid to a Uniform Grid

The final stage of the geometry restoration algorithm involves mapping the original image onto a specified uniform grid of points, followed by transformation using the cubic interpolation method. Initially, the grid points are transformed into a complex form and serve as the foundation for forming a grid with a specific step size, depending on the height and width of the image. This grid is regular and allows the image space to be represented as uniformly distributed nodes. Subsequently, the points within the original curvilinear grid are interpolated onto a rectilinear one using cubic interpolation to create a displacement map. The interpolation is based on a cubic function that approximates the function values at intermediate grid points based on known data. The displacement values of the image coordinates are then separately extracted from the resulting interpolated grid along the x and y directions. The pixel values in the original image are redefined using the obtained displacement map, thus assigning each pixel in the original image a new coordinate value based on the corresponding grid coordinates obtained in the previous step:

$$dst(x, y) = src\left(map_x(x, y), map_y(x, y)\right), \quad (8)$$

where $map_x(x, y)$ and $map_y(x, y)$ represent the displacement values of coordinates x and y , respectively.

The original and transformed images are displayed in Figure 14.

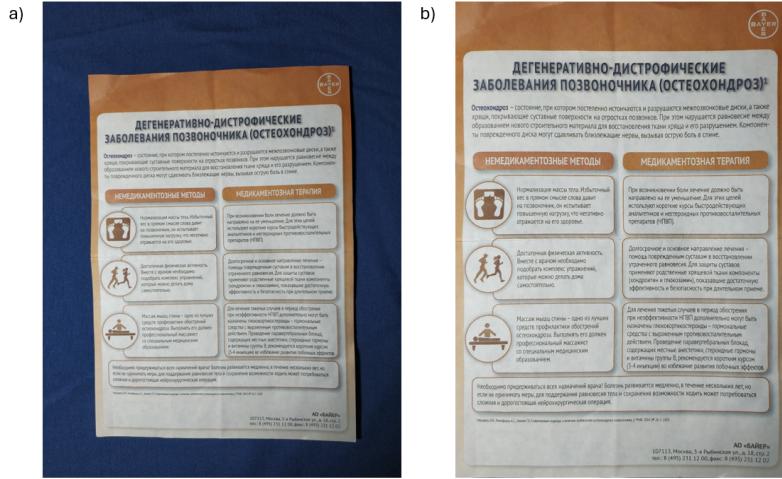


Fig. 14 a) Original document image; b) Corrected document image based on remapping using cubic interpolation.

5. Evaluation Metrics and Test Results on Document Readability and Topology Recovery

In this section, we examine the evaluation metrics for document readability and geometry restoration that we use in comparative testing with the results provided by popular mobile applications for correcting distorted images. The tests demonstrate the effectiveness of our proposed document restoration and dewarping algorithm.

5.1. Document Readability and Geometry Restoration Evaluation Metrics

To assess the accuracy of the developed algorithm, we conducted an analysis of the results using metrics that can be categorized into two main types: (1) Metrics for Evaluating Text Readability by Optical Character Recognition (OCR) models; and (2) Document Geometry Restoration Metrics.

5.1.1 Metrics for Evaluating Text Readability by OCR models

Let's consider the Text Readability Metrics: (1) Levenshtein Distance; (2) Jaro-Winkler Similarity; (3) Character Error Rate (*CER*); and (4) Comparison of recognized characters with the true value.

5.1.1.1 Levenshtein Distance

The Levenshtein Distance [64], also known as the edit distance, quantifies the difference between two strings by determining the minimum number of edit operations, such as insertions, deletions, and substitutions, needed to transform one string into the other.

5.1.1.2 Jaro-Winkler Similarity

Jaro-Winkler Similarity (Eq. 9-10) is a measure of string similarity used to determine the distance between two sequences of characters. It builds on the Jaro distance [65] and incorporates a modification [66] known as the Winkler coefficient to account for the higher similarity of strings starting with the same characters:

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3} \left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m} \right) & \text{if } m \neq 0 \end{cases} \quad (9)$$

where s_i denotes the length of string s_i ; m represents the number of matching characters; t is half the number of transpositions.

$$d_\omega = d_j + (l_p(1 - d_j)) \quad (10)$$

where d_j represents the Jaro distance for strings s_1 and s_2 ; l denotes the length of the common prefix from the beginning of the string up to a maximum of 4 characters; p is a constant scaling factor.

5.1.1.3 Character Error Rate (CER)

The Character Error Rate (CER) (Eq. 11), also known as symbol error rate, is used to assess the quality of character or text recognition in optical character recognition tasks, automatic speech recognition, and other natural language processing tasks. CER measures the percentage of errors that occur when comparing the recognized text with the original source text.

$$CER = \frac{S+D+I}{N} \quad (11)$$

where S is the number of substituted characters; D is the number of deleted characters; I is the number of inserted characters; N is the total number of characters.

5.1.1.4 Comparison of recognized characters with the true value

Comparison of recognized characters with the true value measures the proportion of correctly recognized characters from the total number of characters in the reference text. Unlike *CER*, which focuses on errors, this metric focuses on correct recognitions and is expressed as a percentage of accuracy.

5.1.2 Document Geometry Restoration Metrics

Now let's review the Document Geometry Restoration Metrics: (1) Structural Similarity Index (*SSIM*); (2) Mean Squared Error (*MSE*); (3) Normalized Root Mean Squared Error (*NRMSE*).

5.1.2.1 Structural Similarity Index (SSIM)

SSIM (Eq. 12-13) is a metric used to evaluate the quality of compressed or processed images. It allows for comparing and measuring the structural similarity between the original and processed images. Introduced in 2004 [67], *SSIM* measures similarity based on three main aspects of the image: brightness, contrast, and structure.

$$\begin{cases} c_1 = (k_1 L)^2 \\ c_2 = (k_2 L)^2 \end{cases} \quad (12)$$

where L refers to the dynamic range of pixels; k_1 is a constant valued at 0.01; k_2 is a constant valued at 0.03.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (13)$$

where μ_x is the mean of x ; μ_y is the mean of y ; σ_x^2 is the variance of x ; σ_y^2 is the variance of y ; σ_{xy} denotes the covariance of x and y ; c_1 and c_2 are two variables.

5.1.2.2 Mean Squared Error (MSE) and Normalized Root Mean Squared Error (NRMSE)

MSE and *NRMSE* [39] are metrics used to gauge differences between two datasets, frequently applied in signal processing, computer vision, and statistics. *MSE* (Eq. 14) assesses the mean squared deviation between the values of original and predicted data, while *NRMSE* is a normalized version of *MSE*, representing the ratio of *MSE* to the range of the original data values. This normalization allows for comparing results across different datasets, considering their amplitudes:

$$MSE = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f(x, y) - g(x, y)]^2 \quad (14)$$

where M is the width of the image; N is the height of the image; $f(x, y)$ and $g(x, y)$ are pixel values in the two images. *MSE* is a primary metric for evaluating model accuracy as it shows how much the predicted values deviate from the true values. *NRMSE* allows comparing results between different datasets as it considers the amplitudes and scales of the data. *NRMSE* is easier to interpret since it is expressed as a percentage or fraction relative to the data range, making it easy to assess the degree of error relative to the original data. In the context of document geometry restoration, using *NRMSE* may be preferable as it provides a clearer and more comparable evaluation of restoration quality.

5.2. OCR-based Text Readability for Documents Reconstructed by Our Algorithm and Popular Mobile Apps

5.2.1. Testing with *EasyOCR* and *Tesseract* Libraries for OCR-based Text Readability Evaluation

As is known, Optical Character Recognition (OCR) technology allows for the automatic recognition and interpretation of text contained in images or scanned documents. One of the widely-used OCR systems is *EasyOCR* [68], which, being an open-source *Python* library, is based on deep learning and utilizes CNN for character recognition. *EasyOCR* supports over 80 languages. Key features of *EasyOCR* include text recognition in images with various orientations and sizes, adaptive recognition for processing low-quality or noisy images, and the capability to handle multi-page PDF documents. Additionally, the library provides a user-friendly API for working with text, allowing to obtain recognition results as strings as well as the coordinates of detected text blocks.

Tesseract [69] is a library and tool for OCR developed and maintained by the *Google* team. *Tesseract* employs machine learning techniques and statistical models for character recognition. It is based on Hidden Markov Models (HMM), which enable the modeling of character sequences and transition probabilities between them. *Tesseract* also utilizes algorithms for character segmentation and classification and supports a wide range of languages and alphabets. It offers a collection of trained models for text recognition in various languages, including Latin, Cyrillic, Asian characters, and others. *Tesseract* provides an API for integration into applications and systems, making it versatile for various scenarios and projects. In *Python*, the *pytesseract* library provides a simple interface for using the *Tesseract* OCR engine.

5.2.2. Comparative Analysis of Text Readability for Documents Reconstructed by Our Algorithm and Popular Mobile Apps

We conducted a comparative study of the developed algorithm with commercial mobile applications mentioned in Section 2.3. Specifically, the analysis was performed on the *DocScan*, *TapScanner*, and *CamScanner* applications. For each of the selected applications, 15 test images were uploaded without any prior manual adjustments or parameter changes (see example in Fig. 15). As a result, we obtained transformed images without additional processing. Additionally, two extra images were included in the analysis: a true digital copy of the document obtained using an electronic scanner (referred to as "Scanned image" in Tables 1 and 2) and the original unprocessed document image captured by a smartphone camera (referred to as "Original image" in Tables 1 and 2). This allowed comparison of the results with both the ideal scanning scenario and the worst-case scenario.



Fig. 15 An example of four original images that were used in the comparison.

To ensure the integrity of the experiment, the original text on the document was manually transcribed. We then developed several scripts for text recognition using the *EasyOCR* and *Tesseract* libraries. We began by testing *EasyOCR*; after text recognition, the output contained recognized words, phrases, and sentences. Some words were incorrectly recognized, leading to word errors, some words were missing, and occasionally extra words were added. Since this complicates word-by-word comparison, we decided to concatenate all recognized elements into a single string without spaces and calculate all metrics based on this string. Additionally, we counted the number of recognized characters to evaluate the presence of extra or missing words based on the true number of characters. Metric values were obtained for all images, and the median value was calculated. These values were grouped and presented in Table 1.

Table 1 Text Readability Comparison Using *EasyOCR* library on the Restored Document Image

Image source	Levenshtein distance ↓	Jaro-Winkler similarity ↑	CER ↓	Number of characters (true value 684)
<i>Original image (camera-based)</i>	524	0.80	0.79	726
<i>Scanned image</i>	50	0.91	0.05	683
Our method	268	0.88	0.39	686
<i>DocScan</i>	364	0.82	0.47	692
<i>CamScanner</i>	432	0.87	0.64	708
<i>TapScanner</i>	479	0.83	0.67	706

The first conclusion drawn from analyzing the results in Table 1 is that, as expected, the true digital copy of the document (scanned) shows the best values across all metrics, whereas the original image captured by the camera shows the worst results. Next, the digital copy of the document based on our algorithm ranks first among document image dewarping solutions. *DocScan* ranks second, demonstrating some lag in all metrics compared to the developed algorithm. The performance indicators for *CamScanner* and *TapScanner* are roughly equal, showing a significant increase in Levenshtein distance, indicating incorrect character recognition. The Jaro-Winkler similarity for our algorithm differs slightly from *CamScanner*. As for the CER metric, the values significantly differ from the next two positions, indicating a considerable number of character errors in mobile application solutions. The actual number of characters is 684, and the scanned image of the actual document is closest to this value, containing 683 characters. The result obtained using our algorithm showed three more characters than the original text, while other solutions exhibited an excessive number of recognized characters, also indicating recognition errors.

Next, we utilized the *Tesseract* OCR library, applying the same metrics for analysis. *Tesseract* also evaluates the confidence value for each recognized word, reflecting the algorithm's certainty in recognition accuracy. Consequently, we decided to include the median confidence value in the assessment of each image (see Table 2).

Table 2 Text Readability Comparison Using *Tesseract* library on the Restored Document Image

Image source	Levenshtein distance ↓	Jaro-Winkler similarity ↑	CER ↓	Confidence of the OCR algorithm ↑	Number of characters (true value 684)
<i>Original image (camera-based)</i>	879	0.67	1.32	31	1150
<i>Scanned image</i>	22	0.9	0.03	96	688
Our method	145	0.82	0.21	96	679
<i>DocScan</i>	160	0.8	0.22	88	641
<i>CamScanner</i>	298	0.75	0.43	85	820
<i>TapScanner</i>	175	0.79	0.26	89	673

Although the obtained metric values differ from the results in Table 1 (where *EasyOCR* library were used), the overall trend and ranking distribution remain similar (with the exception that *TapScanner* showed significantly better results than *CamScanner*). The document image created using our algorithm maintains a leading position among the methods considered. The Levenshtein distance between the true scan and the scan produced by our algorithm differs by 123, whereas for mobile applications, this value is even higher. The Jaro-Winkler similarity is approximately the same for all methods; however, our method shows a slight advantage of about 0.02 compared to *DocScan*. The highest median confidence metric values are held by the true digital copy of the document and the algorithm we developed. The CER metric values for all solutions became smaller compared to the *EasyOCR* results in Table 1, but this did not affect the ranking of the algorithms. As for the number of recognized characters, the digital copy of the document (scanned) naturally demonstrates the closest correspondence to the original text. The copy of the document restored by our algorithm showed the best result compared to mobile applications.

5.3. Document Topology Recovery Evaluation

Next, we compared the same images as in Section 5.2 using metrics that describe document topology reconstruction, such as SSIM, MSE, and NRMSE (discussed in Section 5.1.2). Each image was converted to grayscale for applying the SSIM metric. Each of these metrics compares two images against each other, so the actual digital copy of the document (*Scanned image*) was chosen as the reference value. The results of comparing the median value for all documents are presented in Table 3.

Table 3 Comparison of document geometry reconstruction relative to the Scanned document image

Image source	SSIM ↑	MSE ↓	NRMSE ↓
<i>Original image (camera-based)</i>	0.27	72542	0.63
Our method	0.66	15476	0.29
<i>DocScan</i>	0.53	16528	0.3
<i>CamScanner</i>	0.49	20779	0.4
<i>TapScanner</i>	0.61	18828	0.32

Analyzing the results, it can be concluded that the digital copy of the document obtained using our developed algorithm outperforms other solutions across all selected metrics. It exhibits a higher SSIM value and lower MSE and NRMSE values, indicating a more precise reconstruction of the document's geometry captured by a smartphone camera. This demonstrates the efficiency of our developed algorithm compared to other mobile app solutions and opens up prospects for creating high-quality digital copies of paper documents.

6. Comparative Analysis of Documents Reconstructed by Our Algorithm and Popular Desktop DL Models

This section evaluates the quality of document image recovery by comparing the outcomes of our method with well-known DL solutions such as *RectiNet*, *DocGeoNet*, and *DocTr++*. The quality of document geometry restoration is evaluated both visually and based on the Document Geometry Restoration Metrics (described in Section 5.1.2), while the document readability is compared based on the OCR-based Text Readability Metrics (discussed in Section 5.1.1). The conclusion highlights the superiority of our method over SOTA DL approaches, which opens up prospects for the improvement of high-quality digital copies of documents.

6.1. Popular DL Models for Distorted Document Recovery and Visual Comparison with Our Method

To evaluate the quality of document geometry restoration, we applied SOTA desktop DL models such as *RectiNet* [32], *DocGeoNet* [33], and *DocTr++* [34], and prepared the results for visual comparison with our algorithm (see Figure 16). Six images of distorted documents were used for the comparative analysis. These images vary in the degree of physical deformations, document formats, and color palettes. The presented results confirm the superior quality of geometry restoration and dewarping for our algorithm over its counterparts. In all examined examples, our proposed algorithm demonstrated high clarity in document boundary detection, unlike the compared methods, which primarily exacerbate image distortions due to blurry boundary delineation. In the first example (the topmost image in Figure 16), all algorithms exhibit a defect in geometry restoration along the left boundary with a vertical line that

appears curved (convex), caused by the physical deformations of the document. However, despite this, our proposed algorithm shows more presentable results compared to alternative methods, where this distortion is stronger.

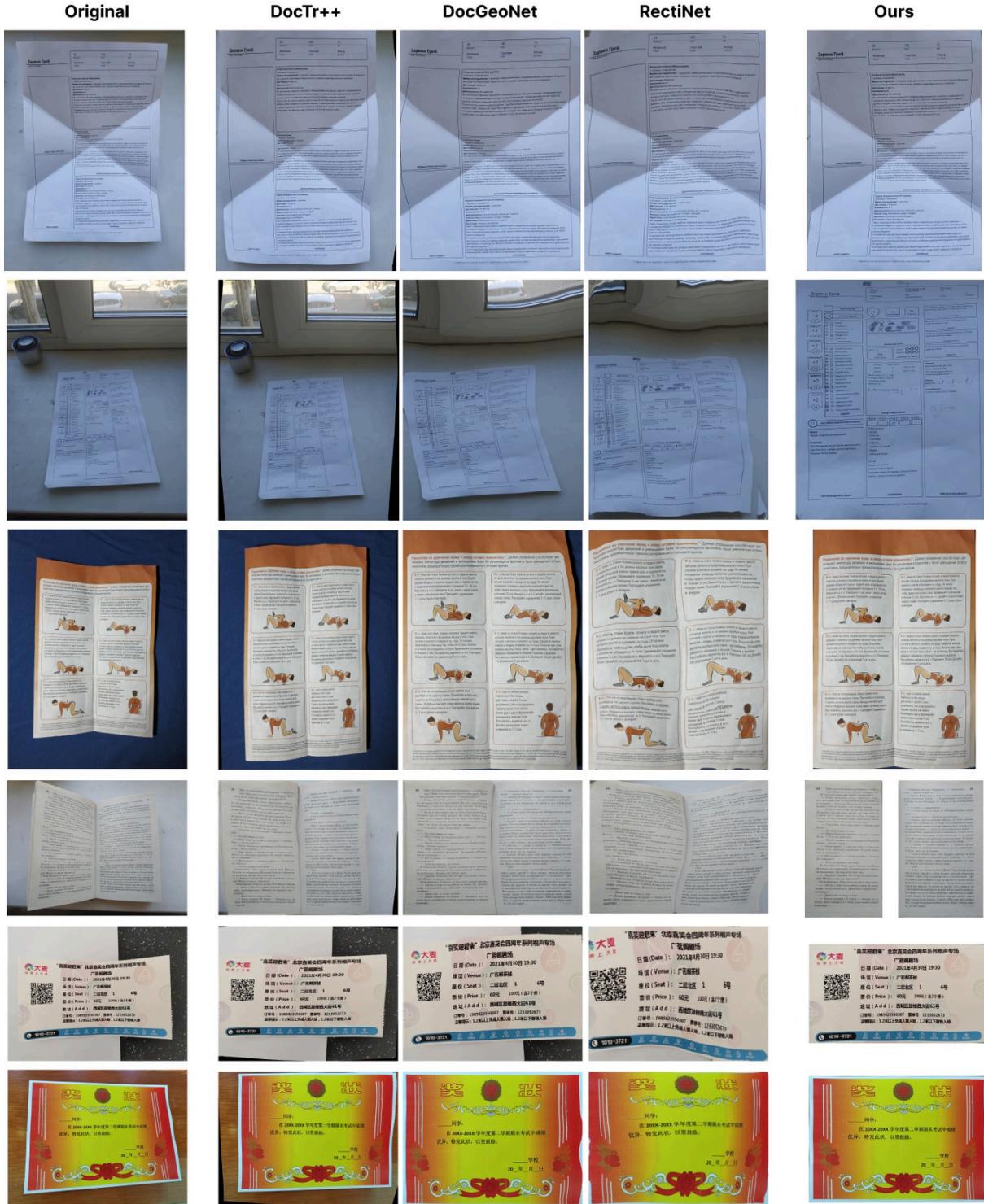


Figure 16. The visual comparative analysis of documents reconstructed by our algorithm and popular desktop DL models - DocTr++, DocGeoNet and RectiNet

6.2. Comparative Analysis of Text Readability Evaluation for Documents Restored by Our Algorithm and Popular Desktop DL Models

We conducted a text readability assessment using the *Tesseract* and *EasyOCR* libraries, leveraging the metrics described in Section 5.1.1, for documents reconstructed by our algorithm and popular desktop DL models. For this evaluation, the same 15 images were selected (shown in Fig. 15, containing English and Russian texts). For each image, a reference text was manually transcribed to validate the OCR results.

The images were processed using our algorithm, as well as *RectiNet*, *DocGeoNet*, and *DocTr++*, followed by OCR application on the resulting images. OCR was then applied to the resulting images to assess the quality of the extracted text. The final median values were calculated and presented in Tables 4-5. Additionally, these images were evaluated using metrics for assessing document geometry restoration quality: SSIM, MSE, and NRMSE, with the median values presented in Table 6.

Table 4 Text Readability Comparison Using *EasyOCR* library

Image source	Levenshtein distance ↓	Jaro-Winkler similarity ↑	CER ↓	Number of characters (true value 684)
<i>Original image (camera-based)</i>	361	0.80	0.79	673
<i>Scanned image</i>	20	0.89	0.05	683
Our method	126	0.87	0.33	674
<i>RectiNet</i>	465	0.78	0.80	680
<i>DocGeoNet</i>	287	0.83	0.70	685
<i>DocTr++</i>	165	0.81	0.51	668

Table 5 Text Readability Comparison Using *Tesseract* library

Image source	Levenshtein distance ↓	Jaro-Winkler similarity ↑	CER ↓	Confidence of the OCR algorithm ↑	Number of characters (true value 684)
<i>Original image (camera-based)</i>	974	0.60	1.32	31	1042
<i>Scanned image</i>	25	0.90	0.04	96	688
Our method	194	0.72	0.43	84	554
<i>RectiNet</i>	904	0.57	1.60	33	2545
<i>DocGeoNet</i>	620	0.60	0.94	36	1123
<i>DocTr++</i>	488	0.68	0.79	37	939

Table 6 Comparison of document geometry reconstruction relative to the Scanned document image

Image source	SSIM ↑	MSE ↓	NRMSE ↓
<i>Original image (camera-based)</i>	0.18	24180	0.63
Our method	0.56	4527	0.27
<i>RectiNet</i>	0.29	15499	0.50
<i>DocGeoNet</i>	0.19	24470	0.61
<i>DocTr++</i>	0.16	25840	0.67

The document readability metrics for *EasyOCR* and *Tesseract* have demonstrated that our algorithm surpasses other document restoration technologies in almost all criteria. Special attention should be given to the recognition confidence metric, embedded within *Tesseract* itself. In this regard, our algorithm shows clear superiority over all

other models considered. Key quality indicators also include similarity metrics, in which our algorithm significantly outperforms alternative technologies, further validating its effectiveness. The only parameter for *EasyOCR* metrics where our method lagged behind competing solutions such as *DocGeoNet* and *RectiNet* is the "Number of characters" (Table 4). However, the number of recognized characters does not guarantee recognition accuracy. To verify the quality of recognition by all methods considered, we conducted an additional test on text recognition within a segment of the instructions, shown at the lower part of the first image in Figure 15 (top left image).

The qualitative analysis results of text recognition using *EasyOCR* and *Tesseract* on document images (instructions): the camera-based photo (Original image), the scanner-based image (Scanned image), and images reconstructed by Our method, as well as *RectiNet*, *DocGeoNet*, and *DocTr++* on a specific text document (instructions). Here is the text fragment that we manually transcribed for comparative analysis:

"White mode brushing instructions 1 Brush the first 2 minutes as explained in section 'Brushing instructions'. 2 After the 2 minutes of Clean mode, the White mode starts with a change in brushing sound and motion. This is your signal to start brushing the upper front teeth for 15 seconds. 3 At the next beep and pause, move to the bottom front teeth for the final 15 seconds of brushing".

Table 7 Comparison of text recognition results using *EasyOCR* and *Tesseract* on document images (instructions): *Original*, *Scanned*, and those reconstructed by Our method, *RectiNet*, *DocGeoNet*, and *DocTr++*

Source	Recognized text with <i>EasyOCR</i>	Recognized text with <i>Tesseract</i> OCR
<i>Original image</i> (camera-based)	White mode brushing in as Brush the first 2 section 'Brushing the White After the 2 minutes of Cleain brushing starts with & change in brushing mode 'signal to and motion. This is 15 the upper front teeth for to the At the next beep and final 15 bottom front teeth for the of brushing:	White mode brushing ins' Fedo Battie \ cas eae = i a Pee \ ion eee Brush the first 2 minutes is © P Re Gi sue ing instructions - : Ko ee 3 ; section 'Brushing instr' sie the White) . : - After the 2 minutes of oe brushing sound ; mode starts with a change 1 rare brushing \ ; and motion. This is your Se on (the upper front teeth for 1 ff z ve to At the next beep and Pe final 15 seconds Marte ; bottom front teeth for t ss 2 t of brushing.
<i>Scanned image</i>	White mode brushing instructions Brush the first 2 minutes as explained in ; ; section 'Brushing instructions' 2 After the 2 minutes of Clean mode; the White mode starts with a change in brushing sound and motion This is your signal to start brushing the upper front teeth for 15 seconds. 3 At the next and pause, move to the bottom front teeth for the final 15 seconds of brushing:	White mode brushing instructions Brush the first 2 minutes as explained in section 'Brushing instructions'. After the 2 minutes of Clean mode, the White mode starts with a change in brushing sound and motion. This is your signal to start brushing the upper front teeth for 15 seconds. — At the next beep and pause, move to the bottom front teeth for the final 15 seconds of brushing.
Our method	White mode brushing instructions Brush the 2 minutes as explained in section 'Brushing instructions' After the 2 minutes of Clean mode; the White mode starts with a change in brushing sound and motion. This is your signal to start brushing the upper front teeth for 15 seconds At the next beep move to the bottom front teeth for the final 15 seconds of brushing:	White mode brushing instructions Brush the first 2 minutes as explained in section 'Brushing instructions'. After the 2 minutes of Clean mode, the ae mode starts with a change in brushing soun and motion. This is your signal to start brushing the upper front teeth for 15 seconds. } IEW At the next beep and pause, move to the bottom front teeth for the final 15 seconds of brushing.
<i>RectiNet</i>	White mode brushing instructions in Brush the first 2 minutes a5 section instructions' the White After the 2 minutes of Clean sound mode starts with & change in brushing H brushing and motion This is to the upper front for 15 At the move to the next beep = bottom teeth for the final 15 of mode Mode for gentle gum stimulation: PgDn PgUp Ctrl Prtsc 88 Alt ENGLISH Personalizing experience Philips ' Sonicare Clean mode_ personalze Prior through The Brushing Clean Standard superior White mode front explained _ 'Brushing mode; start _ signal your seconds: teeth pause; and seconds front brushing:	White mode brushing instructions Brush the first 2 minutes as explained section 'Brushing instructions' . ite After the 2 minutes of Clean mode, the ae! mode starts with a change in br ushing can and motion. This is your signal to start brushing — the upper front teeth for 15 seconds. if At the next beep and pause, move to the bottom front teeth for the final 15 seconds of brushing.
<i>DocGeoNet</i>	White mode brushing instructions Brush the first 2 minutes as explained in section 'Brushing instructions' After the 2 minutes of Clean mode; the White mode starts with a in brushing and motion This is your signal to start brushing the upper front teeth for 15 At the next and pause, to the bottom front teeth for the final 15 seconds of brushing:	White mode brushing instructions ; Brush the first 2 minutes as explained in - section 'Brushing instructions'. After the 2 minutes of Clean mode, the White \ mode starts with a change in brushing sound \ and motion. This is your signal to start brushing the upper front teeth for 15 seconds: (BEB At the next beep and pause, move to the bottom front teeth for the final 15 seconds of brushing.
<i>DocTr++</i>	SWhite mode brushing in Brush the first 2 minutes as section 'Brushing the White After the 2 minutes of Clean mode starts with change in brushing bushing to and motion. This is your zhe upper front teech for 15 to the At the beep and final 15 bottom front teeth for the of brushing:	Wyte ircae brushinelimseauccious Brush the first 2 minutes as explained Mn section 'Brushing instructions - wi ; ite : a After the 2 minutes of Clean ciara i : . <—ee = mode starts with a change !" Brune brushing : : aS eee . = and motion. This is your signal to eae \ an ee the upper front teeth for 15 seconcs: if Ries Spi Te : se eS 2 : ; At the next beep and pause, Move to eer ss ; bottom front teeth for the final 15 seconds - <n of brushing.

The qualitative analysis indicates that for this example of an image with a text fragment, text recognition using the EasyOCR library for the document, whose geometry was restored by our method, was performed with the highest quality, surpassing even the scanned document. The only drawback of our method revealed by the analysis of this table is that the OCR overlooked the item numbering, which leaves room for future improvement. Analysis of Table 6, which includes metrics for evaluating document geometry restoration *SSIM*, *MSE*, and *NRMSE*, also supports the conclusions drawn from OCR text readability metrics, confirming that the digital copy of the document obtained using our method outperforms other solutions across all selected metrics. This also opens up prospects for creating high-quality digital copies of paper documents, making the document topology restoration process more efficient and faster, as it requires significantly fewer computational resources and memory.

7. Conclusions and Discussion

7.1. Conclusions

This study developed a method for geometry restoration and dewarping of digital documents from camera images by integrating DL and computer vision methods. It reviewed modern applications (including Mobile Apps) and desktop DL algorithms, identifying and discussing their shortcomings. Various approaches to document detection and segmentation were explored to select the optimal algorithm, ultimately choosing *YOLOv8*. The methodology we propose creates high-quality digital copies of paper documents, using DL for outline detection and CV to create a topological 2D grid with cubic polynomial interpolation, correcting nonlinear distortions by remapping the image. We developed a new pipeline for automatic document dewarping and reconstruction and provided a framework and dataset to demonstrate its efficiency. The methodology was evaluated against state-of-the-art DL solutions, such as *RectiNet*, *DocGeoNet*, and *DocTr++*, and validated through experiments, demonstrating its effectiveness and superiority in OCR-based document readability and geometry restoration using *SSIM*, *MSE*, and *NRMSE* metrics. The comparative analysis confirmed our method's advantage in accurately restoring document topology and dewarping for creation of precise digital copies.

7.2. Discussion

While our proposed methodology opens new avenues for creating high-quality digital copies of paper documents and enhancing OCR systems, there are further prospects for development and improvement of the algorithm. These can include the following research directions:

- Expanding the training dataset for *YOLO* to achieve more accurate document segmentation in images;
- Exploring the potential of using built-in smartphone sensors or deep learning methods to obtain depth maps for use in document geometry restoration tasks.
- Utilizing newer versions of *YOLO* with Programmable Gradient Information (*PGI*) to preserve important features during training, and Generalized Efficient Layer Aggregation Network (*GELAN*) to enhance accuracy and speed in document outline detection and segmentation.

We would also like to discuss the challenges in performing a correct comparative analysis of different approaches to document geometry restoration and dewarping. For comparing our method with desktop DL applications, we added an option in the DL to output the recognized document image at a specific resolution to evaluate OCR-based document readability and geometry restoration metrics on images of the same size. However, in mobile apps, the output resolution of the detected document in the image is an internal process dependent on the detector used in the apps, and it can vary, which might affect the evaluated metrics. Nevertheless, we are convinced that the overall evaluation and ranking of mobile apps in terms of document geometry restoration quality and OCR-based text readability was correctly determined, as confirmed by the visual comparison of application results.

Data Availability

The datasets considered during the current study are publically available in the repository: <https://github.com/HorizonParadox/DRCCBI>

Conflict of Interest

The authors declare no conflict of interest.

References

1. Electronic document management system. Wikipedia. Available on April 26, 2024. https://en.wikipedia.org/wiki/Document_management_system
2. Yang, P., Antonacopoulos, A., Clausner, C., Pletschacher, S., & Qi, J. (2017). Effective geometric restoration of distorted historical document for large-scale digitisation. *IET Image Processing*, 11(10), 841-853.
3. Alaei, A., Bui, V., Doermann, D., & Pal, U. (2023). Document Image Quality Assessment: A Survey. *ACM computing surveys*, 56(2), 1-36.
4. Das, S., Ma, K., Shu, Z., Samaras, D., & Shilkrot, R. (2019). Dewarpnet: Single-image document unwarping with stacked 3d and 2d regression networks. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 131-140).
5. Wagdy, M., Amin, K., & Ibrahim, M. (2021). Dewarping document image techniques: survey and comparative study. *International Journal of Image and Graphics*, 21(03), 2150031.
6. Li, R. X., Yin, F., & Huang, L. L. (2023, November). Dewarping Document Image in Complex Scene by Geometric Control Points. In *Asian Conference on Pattern Recognition* (pp. 265-278). Cham: Springer Nature Switzerland.
7. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60, 91-110.
8. Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7–13, 2006. Proceedings, Part I* 9 (pp. 404-417). Springer Berlin Heidelberg.
9. Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381-395.
10. Brunelli, R. (2009). *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons.
11. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, part III* 18 (pp. 234-241). Springer International Publishing.
12. Feng, H., Zhou, W., Deng, J., Tian, Q., & Li, H. (2021). DocScanner: Robust document image rectification with progressive learning. *arXiv preprint arXiv:2110.14968*.
13. Garai, A., Dutta, A., & Biswas, S. (2023). Automatic dewarping of camera-captured comic document images. *Multimedia Tools and Applications*, 82(1), 1537-1552.
14. Dasgupta, T., Das, N., & Nasipuri, M. (2020). Multistage curvilinear coordinate transform based document image dewarping using a novel quality estimator. *arXiv preprint arXiv:2003.06872*.
15. Jiang, X., Long, R., Xue, N., Yang, Z., Yao, C., & Xia, G. S. (2022). Revisiting document image dewarping by grid regularization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4543-4552).
16. Simon, G., & Tabbone, S. (2021, January). Generic document image dewarping by probabilistic discretization of vanishing points. In *2020 25th International Conference on Pattern Recognition (ICPR)* (pp. 2344-2351). IEEE.
17. Garai, A., Biswas, S., Mandal, S., & Chaudhuri, B. B. (2021). Dewarping of document images: A semi-CNN based approach. *Multimedia Tools and Applications*, 80(28), 36009-36032.
18. Hertlein, F., Naumann, A., & Philipp, P. (2023). Inv3D: a high-resolution 3D invoice dataset for template-guided single-image document unwarping. *International Journal on Document Analysis and Recognition (IJDAR)*, 26(3), 175-186.
19. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. Attention is all you need in advances in neural information processing systems, 2017. Search PubMed, 5998-6008.
20. Kil, T., Seo, W., Koo, H. I., & Cho, N. I. (2017, November). Robust document image dewarping method using text-lines and line segments. In *2017 14Th IAPR international conference on document analysis and recognition (ICDAR)* (Vol. 1, pp. 865-870). IEEE.
21. Xue, C., Tian, Z., Zhan, F., Lu, S., & Bai, S. (2022). Fourier document restoration for robust document dewarping and recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4573-4582).
22. Ma, K., Shu, Z., Bai, X., Wang, J., & Samaras, D. (2018). Docunet: Document image unwarping via a stacked u-net. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4709).

23. Feng, H., Wang, Y., Zhou, W., Deng, J., & Li, H. (2021). Doctr: Document image transformer for geometric unwarping and illumination correction. arXiv preprint arXiv:2110.12942.
24. Liu, X., Meng, G., Fan, B., Xiang, S., & Pan, C. (2020). Geometric rectification of document images using adversarial gated unwarping network. Pattern Recognition, 108, 107576.
25. Ma, K., Das, S., Shu, Z., & Samaras, D. (2022, July). Learning from documents in the wild to improve document unwarping. In ACM SIGGRAPH 2022 Conference Proceedings (pp. 1-9).
26. Yu, F., Xie, Y., Wu, L., Wen, Y., Wang, G., Ren, S., ... & Li, W. (2024). DocReal: Robust Document Dewarping of Real-Life Images via Attention-Enhanced Control Point Prediction. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 665-674).
27. Nachappa, C. H., Rani, N. S., Pati, P. B., & Gokulnath, M. (2023). Adaptive dewarping of severely warped camera-captured document images based on document map generation. International Journal on Document Analysis and Recognition (IJDAR), 26(2), 149-169.
28. Ramanna, V. K. B., Bukhari, S. S., & Dengel, A. (2019, February). Document Image Dewarping using Deep Learning. In ICPRAM (pp. 524-531).
29. Xie, G. W., Yin, F., Zhang, X. Y., & Liu, C. L. (2020). Dewarping document image by displacement flow estimation with fully convolutional network. In Document Analysis Systems: 14th IAPR International Workshop, DAS 2020, Wuhan, China, 2020, Proceedings 14 (pp. 131-144). Springer International Publishing.
30. Xie, G. W., Yin, F., Zhang, X. Y., & Liu, C. L. (2021). Document dewarping with control points. In Document Analysis and Recognition-ICDAR 2021: 16th International Conference, Lausanne, Switzerland, 2021, Proceedings, Part I 16 (pp. 466-480). Springer International Publishing.
31. Li, H., Wu, X., Chen, Q., & Xiang, Q. (2023). Foreground and Text-lines Aware Document Image Rectification. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 19574-19583).
32. Bandyopadhyay, H., Dasgupta, T., Das, N., & Nasipuri, M. (2021, January). A gated and bifurcated stacked u-net module for document image dewarping. In 2020 25th International Conference on Pattern Recognition (ICPR) (pp. 10548-10554). IEEE.
33. Feng, H., Zhou, W., Deng, J., Wang, Y., & Li, H. (2022, October). Geometric representation learning for document image rectification. In European Conference on Computer Vision (pp. 475-492). Cham: Springer Nature Switzerland.
34. Feng, H., Liu, S., Deng, J., Zhou, W., & Li, H. (2023). Deep unrestricted document image rectification. IEEE Transactions on Multimedia.
35. PDF scanner - DocScan. Google Play. Available on May 04, 2024. <https://play.google.com/store/apps/details?id=pdf.scanner.scannerapp.free.pdfscanner>
36. Adobe Scan: PDF Scanner, OCR. Google Play. Available on May 04, 2024. <https://play.google.com/store/apps/details?id=com.adobe.scan.android&hl=de>
37. PDF Scanner app - TapScanner. Google Play. Available on May 04, 2024. https://play.google.com/store/apps/details?id=pdf.tap.scanner&hl=en_US
38. CamScanner- scanner, PDF maker. Google Play. Available on May 04, 2024. https://play.google.com/store/apps/details?id=com.intsig.camscanner&hl=en_US
39. Gonzalez, R. C., & Woods, R. (2009). Digital image processing: Pearson education india. Digital image processing: Pearson education india.
40. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. IEEE transactions on pattern analysis and machine intelligence, 34(11), 2274-2282.
41. Savvin, S. V., & Sirota, A. A. (2016). Superpixel segmentation methods and their application to analyze images with heterogeneous textures. Bulletin of Voronezh State University. Series: System Analysis and Information Technologies, (4), 165-173.
42. Van den Bergh, M., Boix, X., Roig, G., De Capitani, B., & Van Gool, L. (2012). Seeds: Superpixels extracted via energy-driven sampling. In Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VII 12 (pp. 13-26). Springer Berlin Heidelberg.
43. Felzenswalb, P. F., & Huttenlocher, D. P. (2004). Efficient Graph-Based Image Segmentation International Journal of Computer Vision.
44. Vedaldi, A., & Soatto, S. (2008). Quick shift and kernel methods for mode seeking. In Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part IV 10 (pp. 705-718). Springer Berlin Heidelberg.
45. Fast Slic. GitHub. Available on May 19, 2024. <https://github.com/Algy/fast-slic>

46. DocUNet: Document Image Unwarping via A Stacked U-Net. Available on May 19, 2024. <https://www3.cs.stonybrook.edu/~cvl/docunet.html>
47. COCO - Common Objects in Context. Available on May 19, 2024. <https://cocodataset.org>
48. Smartphone document capture competition (SmartDoc QA). Available on May 19, 2024. <http://smartdoc.univ-lr.fr/smartdoc-qa/>
49. Make Sense. Available on May 19, 2024. <https://www.makesense.ai/>
50. Roboflow. Available on May 19, 2024. <https://roboflow.com/>
51. Ultralytics YOLOv8. GitHub. Available on May 19, 2024. <https://github.com/ultralytics/ultralytics>
52. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).
53. MMDetection. GitHub. Available on May 19, 2024. <https://github.com/open-mmlab/mmdetection>
54. Panina, V. S., & Amelichev, G. E. (2022). Application of convolutional neural networks Mask R-CNN in intelligent parking systems. *E-Scio*, (6(69)), 425-432.
55. Ross, T. Y., & Dollár, G. K. H. P. (2017, July). Focal loss for dense object detection. In proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2980-2988).
56. Cai, Z., & Vasconcelos, N. (2018). Cascade r-cnn: Delving into high quality object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6154-6162).
57. He, K., Sun, J., & Tang, X. (2012). Guided image filtering. *IEEE transactions on pattern analysis and machine intelligence*, 35(6), 1397-1409.
58. Suzuki, S. (1985). Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1), 32-46.
59. Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2), 112-122.
60. Sklansky, J. (1982). Finding the convex hull of a simple polygon. *Pattern Recognition Letters*, 1(2), 79-83.
61. Savitzky, A., & Golay, M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8), 1627-1639.
62. Maneewongvatana, S., & Mount, D. M. (1999, December). It's okay to be skinny, if your friends are fat. In *Center for geometric computing 4th annual workshop on computational geometry* (Vol. 2, pp. 1-8).
63. Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9), 509-517.
64. Levenshtein, V. I. (1966, February). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklad* (Vol. 10, No. 8, pp. 707-710).
65. Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406), 414-420.
66. Winkler, W. E. (1990). String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage.
67. Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4), 600-612.
68. EasyOCR. GitHub. Available on May 19, 2024. <https://github.com/JaidedAI/EasyOCR>
69. Tesseract OCR. GitHub. Available on May 19, 2024. <https://github.com/tesseract-ocr/tesseract>

Ethics Approval

Not Applicable

Funding

This study was not funded by any of the grants.

Author Contributions

Conceptualization, V.I. and O.P.; Methodology, V.I. and O.P.; Data collection and analysis, V.I. and O.P.; Coding, V.I.; Formal analysis, V.I., O.P. and I.A.; Investigation, V.I.; Background review, V.I. and I.A.; Writing - original draft preparation, V.I.; Writing - review & editing, V.I. and I.A.; Visualization, V.I.