Journal of Information Systems and Informatics
Vol. 6, No. 3, September 2024 e-ISSN: **2656-4882** p-ISSN: **2656-5935**
DOI: 10.51519/journalisi.v6i2.759 Published By DRPM-UBD

Image Segmentation On Form Documents Using Bounding Box

Alhafiz[1], Susan Dian Purnamasari[2]*, Yesi Novaria Kunang[3], Ilman Zuhri Yadi[4], Irman Effendy[5]

[1,2]Information System Departement, Bina Darma University, Palembang, Indonesia
Email: [1]al.hafizulquranul007@gmail.com, [2]susandian@binadarma.ac.id, [3]yesinovariakunang@binadarma.ac.id, [4]ilmanzuhriyadi@binadarma.ac.id, [5]irman.effendy@binadarma.ac.id

Abstract

The OKU Timur script is a unique and complex writing system that has been used for centuries. This script has a distinctive script form and has significant historical and aesthetic value for the local community. Although the use of the OKU Timur script is mainly limited to everyday communication, this script is also often used in traditional ceremonies, historical documents, and other cultural contexts. This study aims to process the script by segmenting images into several parts, using techniques such as Bounding Box and image cropping. The results of this study indicate that the process of segmenting images or documents containing the OKU Timur script is effective in separating the script into new images, each of which only contains the desired area within the specified boundaries.

Keywords: East OKU script, Segmentation, Cropping, Bounding box.

1. INTRODUCTION

The OKU Timur Regency is the birthplace of the Komering tribe's civilization. OKU Timur is a regency located in the province of South Sumatra. There are many historical heritage sites in the OKU Timur area, one of which is the Komering script. It is in this land that we can find a variety of artistic heritages of the Komering tribe. In these regions, we find many forms of artistic heritage from the Komering tribe. One of the forms of artistic heritage that still exists for us to enjoy is the Komering script. In ancient times, the Komering script was the most effective and efficient means of communication. In this 4.0 era of advancement, the Komering script has not been eroded by modern times, but many young generations have forgotten and are afraid to learn how to write Komering. This is due to differences in the letter shapes and the way alphabet letters are used in everyday life. Therefore, considering the importance of the role and function of the Komering script, it is only appropriate that the Komering script be preserved

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

as a cultural heritage of the Komering tribe in OKU Timur Regency, South Sumatra province [1].

A script is a system of visual symbols that appears on paper or other media to express the expressive elements of a language. Another term for a script is a writing system [2]. Today, advances in information technology have been rapidly developing and encompassing many aspects of all areas of life. These advancements have resulted in the availability of vast and abundant data, ranging from industry, economy, science, and technology to various other aspects of life [3]. One of the technologies that facilitate human activities is segmentation.

Image segmentation is an important process in digital image processing before the image is further processed in the next stage. Therefore, accurate and precise segmentation is needed to separate objects from the background [4]. Character segmentation needs to be done before the character recognition process. Usually, the image to be segmented is already a binary image (black and white) to facilitate character separation, commonly used in certain languages or cultures.

This study uses image segmentation algorithms with Bounding Box and Crop Image techniques, which are the most common types of data labeling annotations. They are used in Computer Vision to determine the location of image objects, allowing the separation and identification of each character. Image segmentation divides an image into objects based on certain characteristics, and then each object can be processed individually. This, of course, facilitates and speeds up the image processing process. Therefore, the author presents this research with the title "Image Segmentation On Form Documents Using Bounding Box."

2. METHODS

This study uses the Bounding Box method, which was chosen for its ability to detect script characters with a high level of accuracy, especially in cases where characters are in clearly segmented positions. This method marks and separates each character in the image document using a bounding box, allowing for the specific identification of the desired script characters. This technique is very helpful in preparing images for processes such as pattern recognition or data extraction.

In addition, the Crop Image technique is highly suitable for this segmentation process, where the image is cropped based on the predetermined bounding box to produce image fragments focused on the area of the OKU Timur script characters. The advantages of each step include process control but also have drawbacks that require adjustments to optimize the quality of the results. This

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

image processing technique ensures good image processing but requires special attention to detail and handling of issues that may arise during the process.

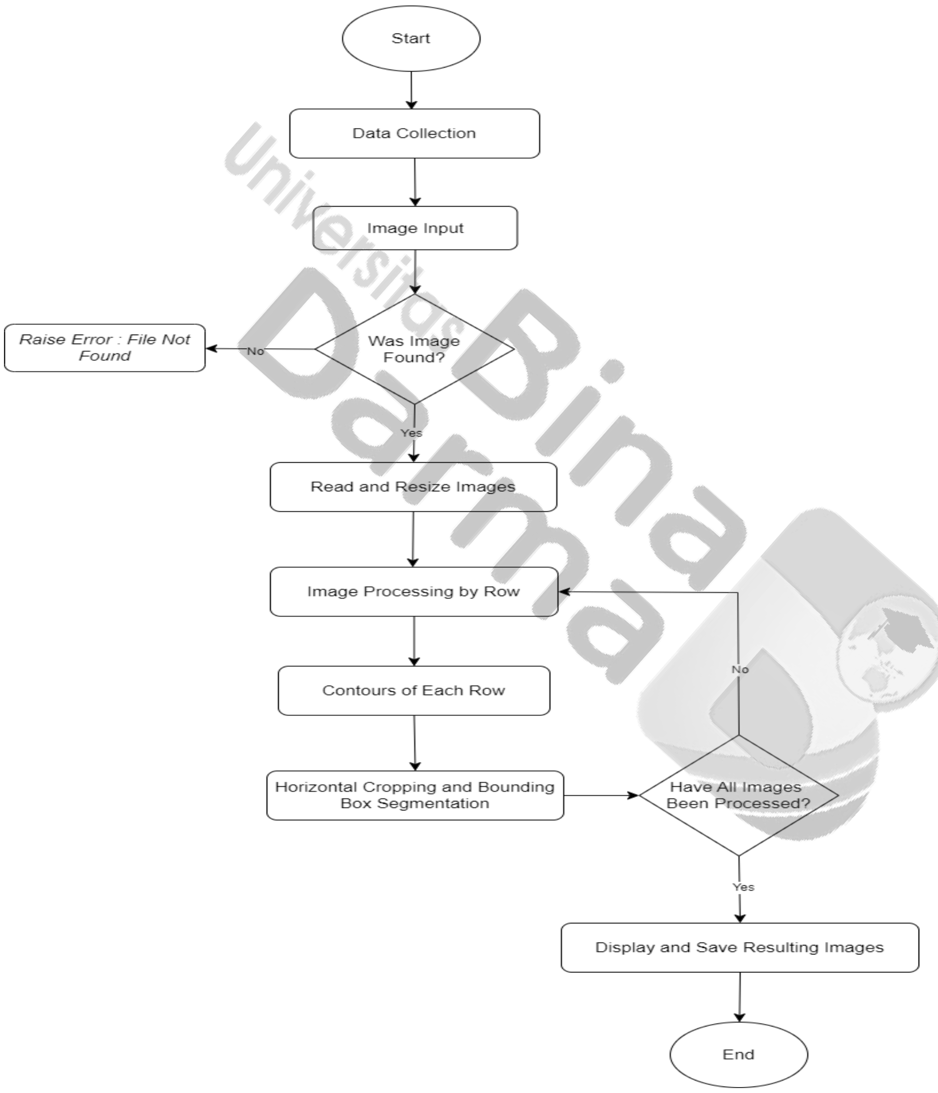

Start
Data Collection
Image Input
Was Image Found?
No
Raise Error : File Not Found
Yes
Read and Resize Images
Image Processing by Row
Contours of Each Row
Horizontal Cropping and Bounding Box Segmentation
Have All Images Been Processed?
No
Yes
Display and Save Resulting Images
End

Figure 1. Flowchart

2.1. Data collection

A questionnaire is an instrument used to measure an event or occurrence, containing a collection of questions or items that need to be filled out to obtain

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

information related to the research being conducted [5]. Previously, the team from the ISRG (Intelligent Systems Research Group) research group had collected initial data on the script. The team gathered basic information about the OKU Timur script. Additional data obtained from the cultural leader and script expert aim to complement and deepen the existing understanding of the OKU Timur script.

This data collection stage is derived from script artifacts found in the OKU Timur region. This process then involves various sources. The OKU Timur script will be processed into a questionnaire for the Image Processing stage. For one page, there are 24 OKU Timur script characters, where in image (a) is a page that has not been filled in by respondents, and image (b) is a questionnaire that has been filled in by respondents. Below is one of the script pages that has been processed into a questionnaire with the aim of identifying the unique writing styles of each respondent, as not all respondents have the same handwriting.

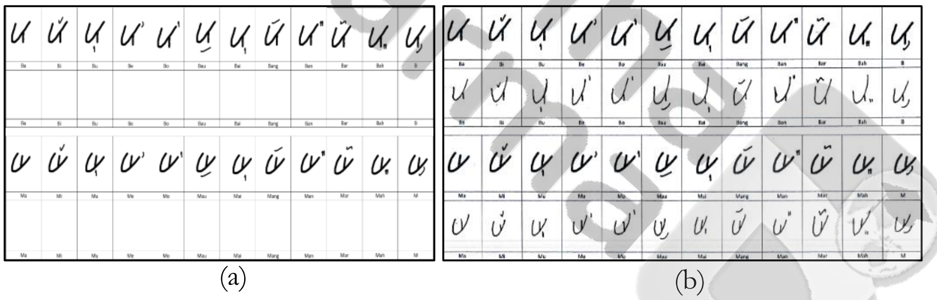


Figure 2. The questionnaire that will be used as a dataset

The data collection is carried out by distributing a questionnaire consisting of 225 classes of the OKU Timur script. This will be given to 102 respondents. The questionnaire is structured in such a way as to cover various character variations of the OKU Timur script. The questionnaire will be processed into a dataset by performing character-wise image cropping using the bounding box method and grouping them based on the name of each letter. The total number of images to be processed is 23,224 images, which will take place during the preprocessing data stage.

2.2 Image Processing

The process of reading digital images to enhance image quality or extract useful information. This process involves various techniques such as Contour detection, Bounding Box, and Image Cropping. At this stage, it is carried out through image processing within the scope of Google Colaboratory by utilizing the capabilities of the Matplotlib, OS, and OpenCV libraries. This technique improves image

p-ISSN: 2656-5935 http://journal-isi.org/index.php/isi e-ISSN: 2656-4882

quality and prepares it for the next stage, ensuring the effectiveness of the images in further processing steps.

2.3 Bounding Box Segmentation

Before applying Bounding on the script characters in Image 2, a horizontal cropping process is carried out to facilitate the Bounding box process, allowing noise in the image to be focused on the desired characters. 'The Bounding Box technique serves as an approach to perform character image segmentation by creating specific boundaries around certain objects in the image using pixel coordinates indicated by upper-right (UR), lower-left (LL), lower-right (LR), and upper-left (UL)' [7]. In this study, the Bounding Box is used to expedite the process of extracting objects within the OKU Timur script questionnaire form. By using the Bounding Box, each character can be extracted as a separate entity.

3. RESULTS AND DISCUSSION

3.1 Image input

The OKU Timur script, which was originally discovered as an ancient artifact, has undergone a binarization process previously conducted by the ISRG team. This pre-processing step was crucial as it allowed researchers to effectively proceed with the segmentation process, making it possible to accurately extract image objects from the script. The entire procedure begins by reading the image file from a specified path, which serves as the initial input for the analysis. In cases where the image is not located in the given path, the system will generate an error message stating "File not found," and subsequently, the process will be terminated to prevent further errors. This validation step is essential to ensure that the required data is available and accessible before moving forward with the more advanced stages of image processing. Ensuring the presence of the image data not only streamlines the workflow but also guarantees the accuracy and efficiency of the segmentation process in this research.

```
# Jalankan fungsi utama dengan parameter crop_top dan crop_bot
image_path = "/content/Aksara.jpg"
main(image_path, scale_factor=1.0, crop_top=-30, crop_bot=40)
```

The advantage of this step is its ability to prevent larger errors in the future by ensuring that only valid images will be processed. However, the drawback is its dependency on the image files being present; if the image file is unavailable or the path is incorrect, the process will stop and require good error handling to avoid disruption.

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

3.2 Image Reading

```
# Fungsi untuk membaca dan mengubah ukuran gambar
def read_and_resize_image(image_path, max_width=2000):
    img = cv2.imread(image_path)
    if img is None:
        raise FileNotFoundError(f"Image '{image_path}' not found.")
    h, w = img.shape[:2]
    if w > max_width:
        img = cv2.resize(img, (max_width, int(h * max_width / w)),
interpolation=cv2.INTER_AREA)
    return cv2.cvtColor(img, cv2.COLOR_BGR2RGB
```

After the image is found, the next step is to resize the image if necessary. This function reads the image from the specified path and resizes it if the image width exceeds the predefined maximum limit. The resizing is done while maintaining the aspect ratio to avoid distortion and to optimize image processing efficiency by ensuring the image is within the appropriate size.

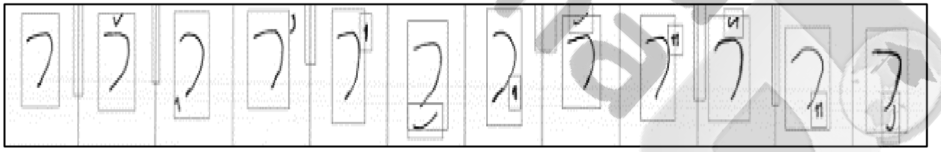


Figure 3. The size result is not optimal

In Figure 3, the result shows an image with pixels that are too large, which is why this process can sometimes lead to branching and overlapping bounding boxes, resulting in suboptimal outcomes. This stage can be likened to trying to print an oversized document on smaller paper. If you attempt to print a giant map on an A4 sheet, you need to reduce the map's size to fit the page without losing readability or important information. Similarly, a large image sourced externally may need to be resized to be processed efficiently without compromising quality. This resizing is done in a way that maintains a balance between efficiency and quality, just like adjusting a map on smaller paper while keeping it informative. The advantage of this process is that reducing the image size can speed up processing and reduce memory usage. However, the downside of this step is the potential loss of detail and image quality, especially if the original image is very large and requires significant reduction.

4.3 Image Processing

After the image size is optimized, the image will be processed row by row for further analysis. This process involves several steps.

4.3.1 Contour Of Each Row

```
# Fungsi utama untuk proses thresholding, dilasi, dan mendapatkan bounding
box
def process_image(image, min_contour_area=500, margin=10, scale_factor=1.5,
save_path='hasil_crop'):
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    thresh = cv2.threshold(gray, 80, 255, cv2.THRESH_BINARY_INV)[1]
    dilated = cv2.dilate(thresh, np.ones((5, 5), np.uint8), iterations=1)
```

The image is converted to grayscale to simplify the analysis, as transforming a color image (RGB) into grayscale means reducing the complexity of the image data. A color image has three types of colors: red, green, and blue, which means each pixel has three intensity values for each color type. In grayscale, the image has only one intensity channel, which speeds up processing and reduces computational complexity without losing relevant information for contour detection. Then, thresholding is applied to highlight the contours, creating a binary image (black and white) that emphasizes the boundaries between objects and the background, making it easier to detect contours around objects. The contours found indicate the boundaries of the objects in the image. Adjustments to the min_contour_area also affect the bounding box process, so some tuning is necessary to ensure that the bounding results are detected as a whole and do not branch.

```
contours, _ = cv2.findContours(dilated, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    result_image = image.copy()
```

After thresholding is applied, contours can be found using contour-finding algorithms such as "cv2.findContours()" in "OpenCV," which then detects the edges of objects by following significant changes in intensity. This approach is particularly useful for images with high contrast between objects and the background, such as black text on a white background.

This method works very well; however, if the image contains noise, color gradients, or uneven lighting, simple thresholding may not be sufficient, and the contour results may become inaccurate. Overlapping contours or objects that are very close together can pose challenges in separation. In conclusion, this approach is often used because it is straightforward, especially for tasks such as edge detection, object segmentation, or processing letters and characters in images.

4.3.2 Bounding Box Segmentation

With this function, the image can be cropped based on specific rows. It is assumed that the image consists of six evenly spaced rows, and the user can select particular rows for further processing. Crop_bot and crop_top margins are added to ensure more accurate crop boundaries. This function is designed to divide the image into several parts according to the specified rows, based on the assumption that the image is divided into six equal vertical sections. The process begins by calculating the total height of the image, and then the height of each row is determined by dividing the image height by six. If the image height is too small for cropping, the system will return an error to inform that cropping is not possible.

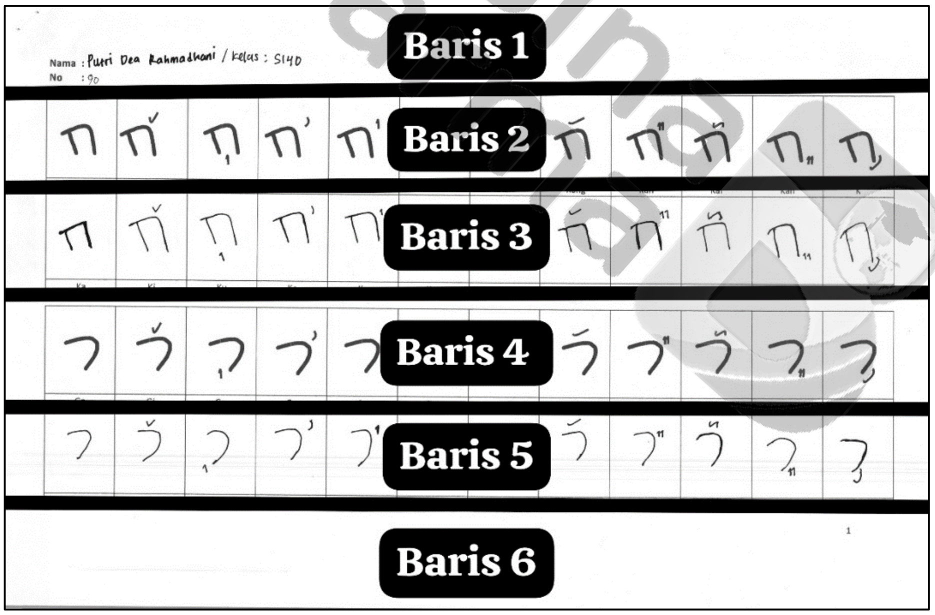

Nama : Putri Dea Rahmadhani / kelas : S14D
No : 90

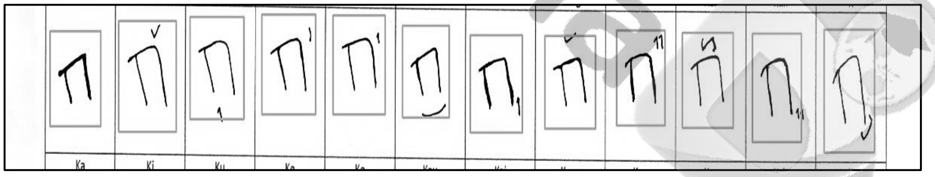
Figure 4. Assumption of Row Design on the Form

After validation is complete, the image is cropped based on the row indices provided as parameters. The vertical positions of the top and bottom parts of the image for each row are determined through calculations that account for the margin adjustments at the top and bottom of the image. Subsequently, the created image segments are stored in a list, and the last segment of the image ensures that any remaining uncropped part is also included. Finally, each image segment produced by this process is returned as output, which can be further processed or saved.

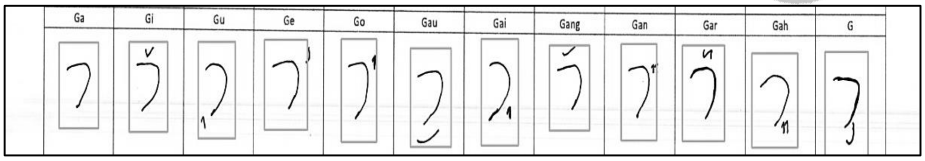
When cropping the image based on the bounding boxes generated from contour detection, each part detected as an object within the image is cropped separately, with the images inside those bounding boxes being extracted and saved as new image files. The output results in images that are split horizontally into sections containing the object segments from the processed image, which are saved in a directory such as save_path. The advantage of this step is the ability to save processing results in an easily accessible and organized format. However, the drawbacks include the potential for numerous output files that require additional management and storage, as well as the potential for longer processing times if the number of objects in the image is large.

```
for idx, (x, y, w, h) in enumerate([cv2.boundingRect(c) for c in contours if
cv2.contourArea(c) > min_contour_area], start=1):
        # Sesuaikan bounding box berdasarkan scale_factor
        new_w, new_h = int(w * 1.2), int(h * 1.5)
        center_x, center_y = x + w // 2, y + h // 2
        new_x, new_y = max(center_x - new_w // 2, 0), max(center_y - new_h //
2, 0)

        # Crop setiap objek di dalam bounding box
        crop = image[max(new_y-margin, 0):min(new_y+new_h+margin,
image.shape[0]), max(new_x-margin, 0):min(new_x+new_w+margin,
image.shape[1])]
        crops.append(crop)  # Simpan crop dalam list

        # Simpan hasil crop sebagai gambar terpisah
        crop_filename = f'{save_path}/{idx}.jpg'
        cv2.imwrite(crop_filename, cv2.cvtColor(crop, cv2.COLOR_RGB2BGR))

        # Gambarkan bounding box pada gambar asli (opsional)
        cv2.rectangle(result_image, (new_x-margin, new_y-margin),
(new_x+new_w+margin, new_y+new_h+margin), (0, 255, 0), 2)

    # Menampilkan semua hasil crop secara terorganisir
    num_crops = len(crops)
    cols = min(6, num_crops)  # Menentukan jumlah kolom (max 6)
    rows = (num_crops + cols - 1) // cols  # Menghitung jumlah baris
    plt.figure(figsize=(8, 8))
    for i, crop in enumerate(crops, start=1):
        plt.subplot(rows, cols, i)
        plt.imshow(crop)
        plt.title(f'Crop {i}')
        plt.axis('off')

    plt.tight_layout()
    plt.show()

    return result_image
```

Image processing is performed using thresholding and dilation, followed by the calculation of bounding boxes based on the contours found, with adjustments made using a predetermined scale factor. These bounding boxes serve to mark the areas surrounding objects in the image. To assess the accuracy of the bounding boxes, a comparison is made between the number of detected bounding boxes and the number of bounding boxes that should be present.



(a)



(b)

Figure 5. Bounding Box Results

The image above shows the results of the crop function that divides the image into two horizontal parts and the bounding box function. Given that each row contains 12 characters that must be identified by the bounding box algorithm on a questionnaire page, it is expected that the image processing results for the second and fourth rows will yield 24 detected bounding boxes. Ideally, each character in these rows will be detected separately, so the number of bounding boxes produced matches the number of characters present.

a. Second Row, The results of the twelve characters filled in by the respondents have been identified and detected using bounding boxes.
b. Fourth Row, The results of the twelve characters filled in by the respondents have been identified and detected using bounding boxes.

4.4 Output Image Display

```
# Fungsi utama
def main(image_path, scale_factor=1.0, crop_top=0, crop_bot=30):
    image = read_and_resize_image(image_path)

    # Tampilkan gambar asli
    plt.imshow(image)
    plt.title("Gambar Asli")
    plt.show()

    # Potong gambar berdasarkan baris ke-2 dan ke-4
    rows = process_rows(image, row_indices=[3, 5], crop_top=crop_top, 
crop_bot=crop_bot)

    for idx, row in enumerate(rows, start=2):
        processed_row = process_image(row, scale_factor=scale_factor, 
save_path=f'baris_ke_{idx}')
        cv2.imwrite(f'Baris_{idx}.jpg', cv2.cvtColor(processed_row, 
cv2.COLOR_RGB2BGR))

        # Tampilkan hasil pemotongan
        plt.figure(figsize=(10,10))
        plt.imshow(processed_row)
        plt.title(f"Baris ke-{idx}")
        plt.show()
```

The cropped results from each row are meticulously displayed in a grid format using Matplotlib, which not only provides an organized and aesthetically pleasing visualization but also significantly enhances the analysis of the cropping outcomes. This structured presentation facilitates a more detailed examination of each individual image segment, allowing for a comprehensive assessment of the effectiveness of the cropping process. Furthermore, the final results are diligently saved as files for future reference and thorough documentation, ensuring that the workflow is transparent and easily retrievable. This section is designed to deliver a holistic visual representation that encapsulates the entirety of the image processing journey once all cropping tasks are completed. It serves the dual purpose of showcasing the processed images while also preserving the final versions that have undergone rigorous transformations. The application of this approach spans across the entire image or a collection of images that have been meticulously processed, culminating in the presentation of final outputs that include both the cropped segments and the original images adorned with bounding boxes. The focus during the image results stage is primarily on providing a visual synthesis of the overall results from the entire processing sequence, highlighting not only the individual cropped images but also the original images that have been annotated with bounding boxes, thereby enhancing the interpretability and significance of the results.

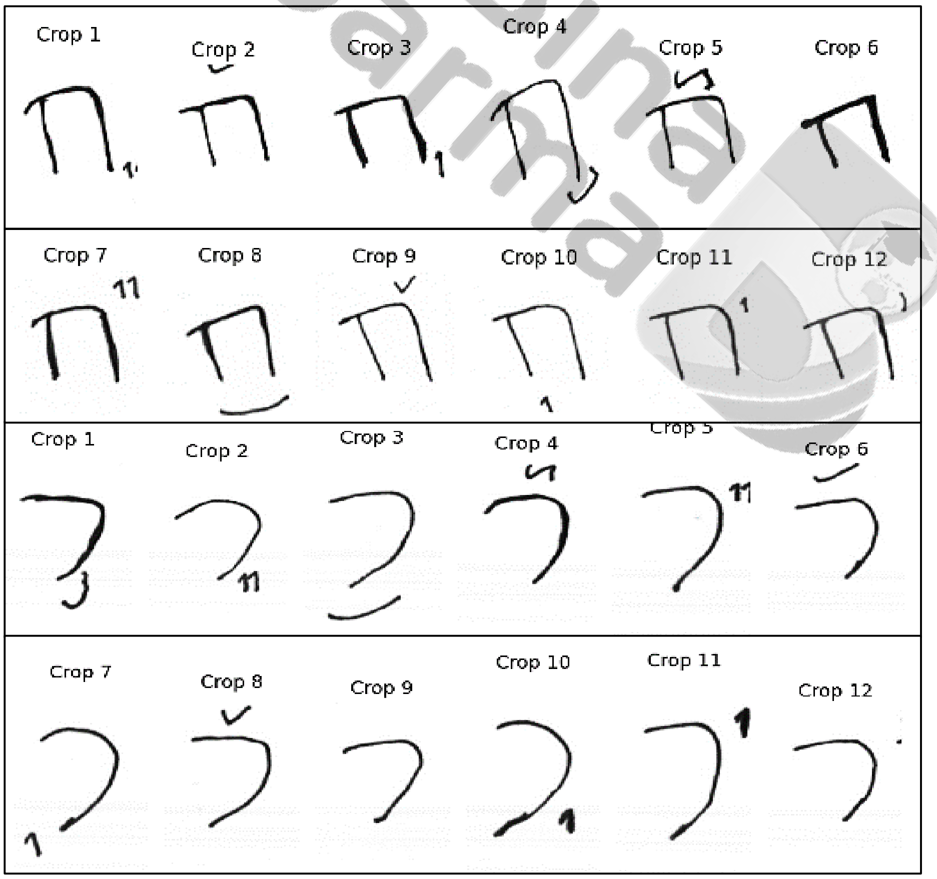


Figure 6. Characters in Grid Format

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

4. CONCLUSION

In this research, the bounding box method in the segmentation process for extracting image objects from the OKU Timur script is useful for detecting the desired script characters. The stages involved include data collection into a dataset, image processing leading to the bounding box stage. The final result is images of the script that have been successfully segmented using bounding boxes, providing a clear visualization of the characters, as well as the extraction of script objects one by one, which will be clustered based on their respective characters for training by a model. Although the results show promising levels, there is still room for improvement.

REFERENCES

[1] Giwang Sumsel, 2022, Dinas Kebudayaan dan Pariwisata Provinsi Sumatera Selatan, https://giwang.sumselprov.go.id/budaya/detail/412 (Accessed on May 27, 2024).

[2] Arifin, M. S. (2019, September). Penggunaan Budaya Visual Aksara Kaganga Sebagai Inovasi Industri Huruf Modern Reinvensi Pragmatis Untuk Inovasi Industri Kreatif Berbasis Budaya Visual Nusantara. In *Seminar Nasional Seni dan Desain 2019* (pp. 243-247). State University of Surabaya.

[3] Alhapizi, R., Nasir, M., & Effendy, I. (2020). Penerapan Data Mining Menggunakan Algoritma K-Means Clustering Untuk Menentukan Strategi Promosi Mahasiswa Baru Universitas Bina Darma Palembang. In *Journal of Software Engineering Ampera* (Vol. 1, Issue 1).https://journal-computing.org/index.php/journal-sea/index.

[4] Mardiah, H. S. (2020). Segmentasi Citra Untuk Pencarian Kode Warna Cat Menggunakan Metode Thershold Hsv. *Bulletin of Information Technology (BIT)*, *1*(3), 134-143.

[5] Nur Amalia, R., Setia Dianingati, R., & Annisaa, E. (2022). PENGARUH JUMLAH RESPONDEN TERHADAP HASIL UJI VALIDITAS DAN RELIABILITAS KUESIONER PENGETAHUAN DAN PERILAKU SWAMEDIKASI. *Generics : Journal of Research in Pharmacy Accepted : 4 Mei*, *2*(1).