



Preserving Tamil Brahmi Letters on Ancient Inscriptions: a Novel Preprocessing Technique for Diverse Applications

K Poornimathi, V Muralibaskaran and L Priya

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

December 16, 2023

Preserving Tamil Brahmi Letters on Ancient Inscriptions: A Novel Preprocessing Technique for Diverse Applications

Poornimathi K¹, Muralibhaskaran V², Priya L³

^{1,2,3}Rajalakshmi Engineering College, Chennai, India

poornimathi.k@rajalakshmi.edu.in

muralibhaskaran.v@rajalakshmi.edu.in

priya.l@rajalakshmi.edu.in

Abstract. Inscriptions play a crucial role in preserving historical, cultural, and linguistic information. The identification and analysis of patterns in Tamil letters found in inscriptions provide valuable insights into the evolution of the Tamil language and its script. However, manual analysis of these inscriptions is time-consuming and prone to errors. In recent years, deep learning techniques have shown promising results in pattern recognition tasks, motivating the exploration of various strategies to identify the patterns of Tamil letters on inscriptions. This paper focuses on leveraging deep learning algorithms for the automated identification of Tamil letter patterns in inscriptions. Firstly, a dataset of digitized Tamil inscriptions is collected, consisting of high-resolution images representing a wide range of letter variations. Preprocessing techniques are employed to enhance the quality and clarity of the images, removing noise and artifacts. Various deep learning models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), are then trained using the preprocessed image dataset. CNNs excel in extracting spatial features from images, enabling the recognition of letter shapes and contours. RNNs, on the other hand, capture temporal dependencies within sequences of letters, aiding in deciphering the structure and connectivity of the inscriptions. To improve the performance of the models, data augmentation techniques are employed to increase the dataset size and enhance its diversity. However, preprocessing plays a major role in sharpening the features present in the image. Hence, this paper addresses the preprocessing techniques such as Image Blur, Binarization and Edge Detection with respect to inscription. Preprocessing techniques were identified and tested with the inscription image. Based on the results and response time, it is suggested that the Median filter with canny edge detection is working well for inscription images. After preprocessing the results have been tested with edge detection and it is found that, Median filter with canny edge detection gives best accuracy in comparison with other algorithms.

Keywords: Character Recognition, Pre-processing, Image processing, Computer vision.

1 INTRODUCTION

Father of inscriptions is Samudragupta. However, Chandragupta Maurya was the one who has used inscriptions to communicate with his subjects. In olden days inscriptions are the only format to communicate the human insights. The Dravidian literary languages include Tamil, Telugu, Kannada, and Malayalam. The eldest member of the Dravidian language family is the Tamil language. It has been in this world for more than 5,000 years. Based on the market survey, the Tamil-only periodicals are numbered more than 1800. Based on a meticulous examination of grammatical and lexical variances, these periodicals have been categorized into three distinct groups: Old Tamil (spanning approximately from 450 BCE to 700 CE), Middle Tamil (from 700 to 1600 CE), and Modern Tamil (since 1600 CE).

Tamil-Brahmi, also known as Tamili, represents a variant of the Brahmi script that was used in southern India. It served as the script for inscriptions in the early form of Old Tamil. The Tamil-Brahmi script's historical and stratigraphic dating places it between the third century BCE and the first century CE. It is the earliest known writing system found in numerous regions encompassing Tamil Nadu, Kerala, Andhra Pradesh, and Sri Lanka. Inscriptions in Tamil Brahmi have been discovered on various surfaces, including cave entrances, stone beds, potsherds, jar burials, coins, seals, and rings.

Starting from the 5th century CE, Tamil began to be written in Vatteluttu script in the Chera and Pandya regions, while the Chola and Pallava regions adopted the Grantha or Tamil script for writing. Vatteluttu is an ancient script that was used to write the Tamil language and several other languages in South India. The term "Vatteluttu" means "rounded script" in Tamil, referring to the rounded or circular shapes of its characters. This script has a rich history and is of great importance in the development of writing systems in the region. It evolved from the earlier Tamil-Brahmi script, which was influenced by the Brahmi script of North India. Vatteluttu script further developed into

other scripts such as the Grantha script, which in turn influenced the modern Tamil script. Grantha script is an ancient writing system primarily used for writing the Sanskrit language and other languages of the South Indian region. It holds significant historical and cultural importance, particularly in the context of Tamil Nadu and Kerala. Both Vatteluttu and Grantha scripts are derived from the Brahmi script, but they belong to different categories and have distinct characteristics. Vatteluttu is associated with Tamil and other Dravidian languages, while Grantha script is associated with Sanskrit and Tamil in a religious context.

Tamil inscription images can be easily read and analyzed if the preprocessing is done effectively. Hence, this paper presented the various preprocessing techniques and its results for Tamil inscription images. The inscription images represent politics, religious, economy, location, tradition, astronomy, history, culture and many others.

Some of the ancient inscription images are shown below in Fig 1 for a better understanding of inscription images.



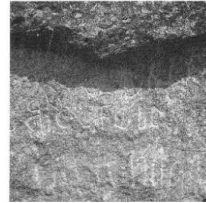
a. Ashokan Kalvettu



b. Inscription at vikkiramankalam
(விக்கிரமங்கலம்)



c. Inscription at
muthalaikulam
(முதலைக்குளம்)



d. Inscription at Aanaimalai
(ஆனைமலை)

Fig. 1. Sample Images of Tamil Inscription.

2 LITERATURE REVIEW

In this section, a thorough discussion and analysis were conducted on the research work related to preprocessing techniques utilized for inscriptions, as well as the technique employed for the recognition of letters within inscriptions.

To identify prehistoric languages, signs, and fonts, Naresh et al. (2022) used an approach that integrates an artificial neural network (ANN) with the Opposition-based Grey Wolf optimization Algorithm (OGWA). The authors emphasize the importance of the weights and connections between layers in ANN system efficacy. This paper investigates the implementation of various optimization algorithms, including Opposition-based Grey Wolf optimization, Particle Swarm optimization, and Grey Wolf optimization, to the ANN system in order to determine these weights effectively. In addition, the researchers declare their intention to reconfigure the ANN's structure in future studies in order to improve the system's predictive performance. [1]

Dhivya et al. (2021) [2] presented a character recognition system for ancient Tamil script, including training a Convolutional Neural Network (CNN) from scratch with a SoftMax classifier, implementing transfer learning using Mobile Net, and incorporating CNN and Support Vector Machines (SVM). The system achieved high accuracy rates for various datasets, such as 98.1% for vowels, 97.7% for consonants, and 97.5% for consonant vowels. Data collection involved contributions from approximately 1,600 engineering students at the SRM Institute of Science and Technology.

Sukanthi et al. (2021) [3] introduced a modified bi-level thresholding (MBET) algorithm for binarizing images of terrestrial and underwater stone inscriptions. This adaptive local thresholding method aimed to reduce noise and accurately extract object margins in both terrestrial and underwater images, and it was compared to several existing thresholding algorithms. Brindha et al. (2021) [4] proposed a technique for converting ancient Tamil script into its modern form. This method involved a new feature extraction approach using a chi-square test to assess feature independence, followed by neural networks for character recognition. The system achieved an accuracy rate of

91.3%.Neha Gautam et al. (2020) [5] introduced a deep convolutional neural network (CNN) with dropout for recognizing Brahmi words. Their study achieved a recognition rate of 92.47% on a Brahmi image database and conducted experiments to contrast different methods and parameter tuning for optimal results.In Merline et al. (2020) [6], training of an 18-layer CNN was conducted for character recognition in the Tamil script. The CNN architecture employed ReLU activation functions and autonomously learned image-based features in a hierarchical fashion, demonstrating good results through supervised learning.Suriya et al. (2020) [7] utilized a Convolutional Neural Network to recognize characters in challenging conditions, addressing issues like low resolution, blur, and low contrast. The study involved the use of HP Labs India's Isolated Handwritten Tamil Character dataset, with various CNN models producing different results.Lalitha et al. (2019) [8] focused on enhancing optical character recognition techniques for 7th- to 12th-century Tamil script. Their approach included binarization, a two-dimensional CNN, and the integration of Tesseract with text-to-speech output. They aimed to develop a universally applicable OCR system with audio output, and future work could involve expanding the dataset and improving segmentation and accuracy.Merline et al. (2019) [9,13] detailed the identification of ancient Tamil characters on stone inscriptions using OCR technology. They employed ensemble learning, K-nearest neighbors (KNN), and Unicode mapping for character classification. Pre-processing included noise removal, segmentation, and feature extraction.In Durga et al. (2018) [11,12], a system for improving stone inscription images was proposed, involving the use of IBF to eliminate noise while preserving character edges. The fuzzy system aimed to predict character and background pixel uncertainty.These studies and methods offer a diverse range of techniques for character recognition in the context of ancient Tamil script, providing valuable insights and opportunities for further research and development.

3 METHODOLOGY FOR INSCRIPTION TRANSLATION

A method has been presented in this paper from the detailed study of literature present in this area. This method is used for the translation of the ancient south Indian languages into contemporary language using stone inscription images from multiple geographical locations.

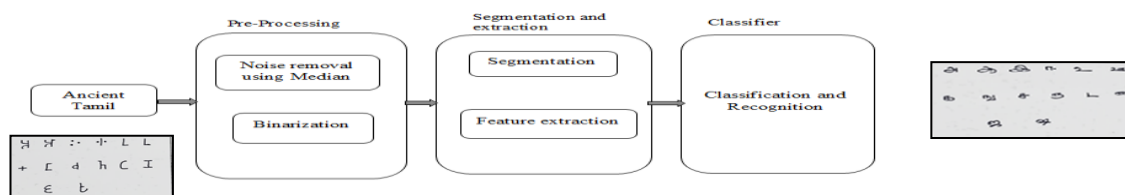


Fig. 2. The proposed system's block diagram.

The ancient Tamil letters from the stone inscription images are being taken as the input. Noise levels present in the input images are pre-processed using suitable algorithms and then the image is getting binarized. Pre-processed image is segmented and the features of the letters are extracted and based on the training model it has been classified and recognized as the Tamil letters which are in practice now.

This paper mainly focuses on the preprocessing techniques involved in the inscription image translation are:

- a. Noise Removal
- b. Gray Scale Image
- c. Thresholding
- d. Thinning and Skeletonization
- e. Skew Correction
- f. Normalization
- g. Image Scaling

Out of the above Preprocessing techniques, the following techniques are being implemented and tested for Tamil inscriptions.

- a. Image Blurring
- b. Binarization
- c. Edge Detection

3.1 Image Blurring

It is a technique used in image processing to reduce the sharpness or clarity of an image intentionally. It involves applying a blurring filter to the image, which results in a smoother or less detailed appearance. Blurring can serve various purposes, such as noise reduction, hiding sensitive information, or creating artistic effects.

There exists four different blurring techniques which are widely used for inscriptions and are listed below:

- i) Average
- ii) Gaussian
- iii) Median and
- iv) Bilateral

Gaussian Blur : It applies a Gaussian distribution-based kernel to the image. It provides a smooth blurring effect while preserving the overall structure and reducing noise.

Median Blur : It replaces each pixel with the median value of its neighboring pixels. It is effective for reducing salt-and-pepper noise while preserving edges and fine details.

Bilateral Blur : It considers both spatial proximity and pixel intensity similarity to perform blurring. It preserves edges and details while reducing noise, resulting in a smoother image with preserved structure.

code snippet for preprocessing is given below for better understanding.

```
import cv2
from google.colab.patches import cv2_imshow
from matplotlib import pyplot as plt
img = cv2.imread('content/kalvettu2.png')
print('Original Image')
cv2_imshow(img)
# Gaussian Blurring
gausBlur = cv2.GaussianBlur(img, (3, 3), 0)
print('Gaussian Blurring')
cv2_imshow(gausBlur)
cv2.waitKey(0)
# Median blurring
medBlur = cv2.medianBlur(img, 3)
print('Median Blurring')
cv2_imshow(medBlur)
cv2.waitKey(0)
# Bilateral Filtering
bilFilter = cv2.bilateralFilter(img, 3, 25, 50)
print('Bilateral Filtering')
```

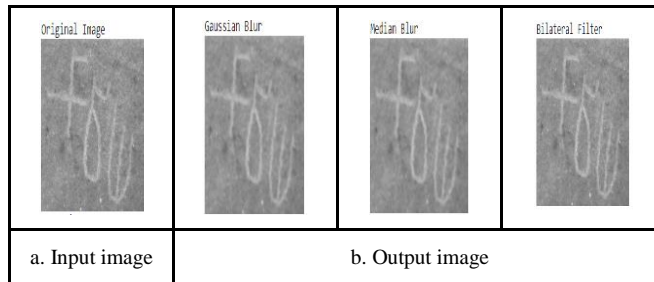


Fig. 3. Image Blurring

Performance metrics have been analyzed using the following expressions. The Mathematical Relationship used to calculate the performance metrics has been given in Table 1

Table 1. Mathematical Relationship to Calculate Metrics

Performance Metrics	Mathematical Relationship
Peak Signal-to-Noise Ratio (PSNR)	$PSNR=10 * \log_{10}(\frac{L^2}{MSE})$
Structural Similarity Index Measure (SSIM)	$SSIM=(L^2 * C * S)^{\alpha} \beta$
Root Mean Square Error (RMSE)	$RMSE=\sqrt{MSE}$
Mean Squared Error (MSE)	$MSE=\frac{1}{MN} \sum_{x,y=1}^{MN} [I(x,y) - B(x,y)]^2$
Precision(P)	$TP / (TP + FP)$
Recall(R)	$TP / (TP + FN)$

The representation used in the above-mentioned mathematical relationship table description are given below. The original image (denoted as I) and the blurred image (denoted as B). M and N are the dimensions of the image.

Luminance Comparison (L): It measures the similarity of the luminance or brightness of the two images. This component focuses on how well the intensity information is preserved in the distorted image.

Contrast Comparison (C): It assesses the similarity of contrast or the difference between light and dark areas in the two images. This component evaluates whether the contrast is maintained.

Structure Comparison (S): It measures the similarity of the image structure, which includes the pattern and texture. This component quantifies how well the structural details are preserved.

True Positives (TP): The number of correctly detected edges.

False Positives (FP): The number of edges detected by the algorithm but not present in the ground truth.

False Negatives (FN): The number of edges present in the ground truth but missed by the algorithm.

Precision measures the fraction of the detected edges that are true edges, while recall measures the fraction of true edges that were correctly detected.

Performance Metrics of Image Blur

The assessment of image blurring techniques involves evaluating how well a method replicates or eliminates blur in an image. Multiple metrics are available to gauge the performance, including Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), Root Mean Square Error (RMSE), Normalized Cross-Correlation (NCC), and more.

Table 2. Performance Metrics of Image Blur Techniques

Performance Metrics	Image Blur Techniques		
	Gaussian Blur	Median Blur	Bilateral Blur
Peak Signal-to-Noise Ratio (PSNR)	32.52	31.81	38.98
Structural Similarity Index Measure (SSIM)	0.8044	0.7097	0.9650
Root Mean Square Error (RMSE)	6.03	6.55	2.87

3.2 Binarization

The goal of image binarization is to segment an image into foreground and background regions by assigning a threshold value to each pixel. If the pixel's intensity value is above the threshold, it is assigned the value 1 (white), indicating it belongs to the foreground. If the intensity value is below the threshold, the pixel is assigned the value 0 (black), indicating it belongs to the background. It is used to simplify an image and focus on specific regions or objects of interest. Particularly useful in various image analysis tasks, such as character recognition, document analysis, object detection, and feature extraction. Some common methods for image binarization include global thresholding, adaptive thresholding, Otsu's thresholding, and local thresholding techniques.

Global Thresholding

This technique is straightforward and commonly utilized, involving the application of a single threshold value to the entire image. Pixels with intensity values exceeding the threshold are designated as part of the foreground, while those falling below the threshold are categorized as part of the background. This approach relies on the assumption of a bimodal histogram, where a distinct demarcation exists between the intensities of the foreground and background.

```
If pixel_intensity >= threshold_value:  
    Set pixel_value = foreground_value  
else:  
    Set pixel_value = background_value
```

Adaptive Thresholding

It is a technique that adjusts the threshold value locally based on the properties of small neighbor hoods within the image. It is useful when dealing with images with uneven illumination or variations in object appearance. Common adaptive thresholding methods include:

Adaptive Mean Thresholding: This method calculates the threshold for each pixel based on the mean intensity of its local neighborhood.

Steps to be followed:

1. *Define* the size of the local neighborhood (typically a square or rectangular window) centered around each pixel.
2. For *each pixel* in the image:

- a. **Calculate** the mean intensity of the pixel's local neighborhood.
 - b. **Determine** the threshold value for the pixel by subtracting a constant (often referred to as the offset) from the calculated mean intensity.
 - c. **Compare** the pixel's intensity with the threshold value:
3. If the intensity is greater than or equal to the threshold, assign it as foreground (white). and If the intensity is less than the threshold, assign it as background (black).

Adaptive Gaussian Thresholding: Here, the threshold is determined by the weighted mean of the pixel's local neighborhood using a Gaussian window.

Steps to be followed:

1. **Define** the size of the local neighborhood (typically a square or rectangular window) centered around each pixel.
2. **Calculate** the Gaussian weight matrix based on the size of the local neighborhood. The Gaussian weight matrix assigns higher weights to the pixels closer to the center of the neighborhood and lower weights to the pixels farther away.
3. For **each pixel** in the image:
 - a. Extract the local neighborhood centered around the pixel.
 - b. Multiply the pixel intensities of the neighborhood with the corresponding weights from the Gaussian weight matrix.
 - c. Calculate the weighted mean intensity of the neighborhood.
 - d. Determine the threshold value for the pixel by subtracting a constant (offset) from the calculated weighted mean intensity.
 - e. Compare the pixel's intensity with the threshold value:
 - If the intensity is greater than or equal to the threshold, assign it as foreground (white).
 - If the intensity is less than the threshold, assign it as background (black).

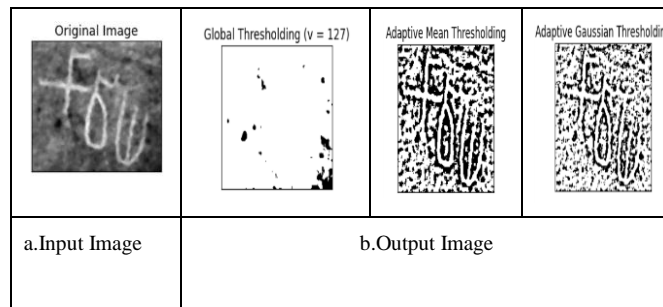


Fig. 4. Adaptive Thresholding.

Otsu's Thresholding

This method automatically determines the optimal threshold value by maximizing the between-class variance. It computes the threshold that minimizes the intra-class variance within the foreground and background regions, resulting in a better separation of objects from the background. Otsu's method is particularly useful for images with uneven illumination and non-bimodal histograms.

For each possible **threshold_value**:

Calculate the probabilities of pixels being in foreground and background classes

Calculate the mean intensities of the foreground and background classes

Calculate the between-class variance

Select the **threshold_value** that maximizes the between-class variance

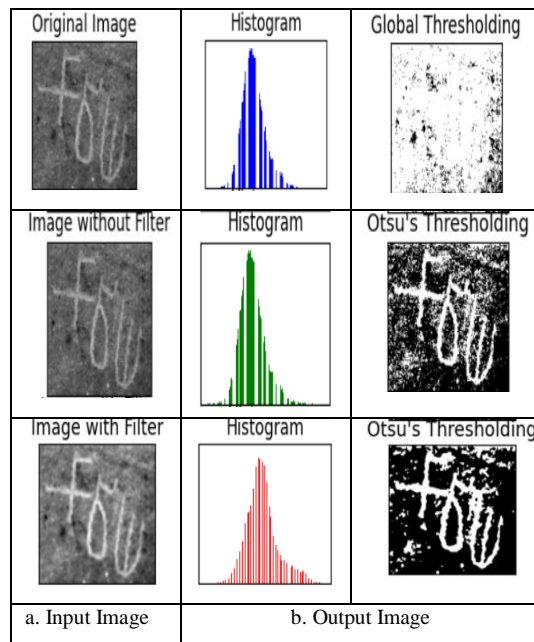


Fig. 5. Otsu's Thresholding.

Kernel or Filter

It is an essential parameter in image processing algorithms, especially in operations like filtering and convolution. The kernel, also known as a filter or a mask, is a small matrix used to process the pixels of an image. It plays a crucial role in determining the nature and extent of the image processing operation applied. The importance of kernel size in image processing are Spatial Extent, Detail Preservation, Computational Complexity, Feature Extraction, Artifact Removal and Trade-off between Smoothing and Localization. It is essential to understand the characteristics and requirements of the image processing operation to select an appropriate kernel size that balances the desired outcome with computational efficiency and the preservation of relevant image details.

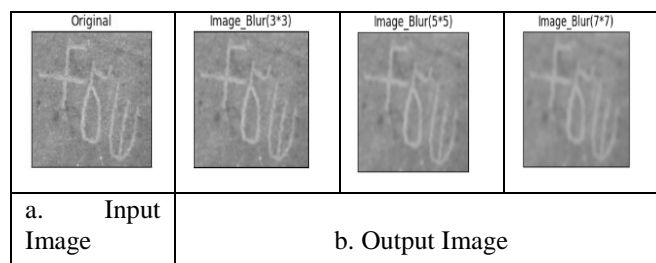


Fig. 6. Kernel Size

3.3 Edge Detection

It is used to identify boundaries or edges between different objects or regions within an image. It is performed using various algorithms or operators, such as the Sobel operator, Canny edge detector, or Laplacian of Gaussian (LoG), among others. These algorithms analyze the gradients or changes in pixel intensities to locate and highlight edges within an image. Edges represent significant transitions in image intensity and by detecting edges, subsequent algorithms can focus on analyzing and interpreting the structural information within the image. Edge detection is an important preprocessing step that sets the foundation for subsequent image analysis and interpretation tasks in image processing pipelines. Most commonly used types of edge detection methods:

Steps Followed:

Sobel Operator:

It is a popular edge detection method that uses a convolutional kernel to compute the gradient magnitude of an image. It approximates the gradient of the image intensity at each pixel to identify edges. The Sobel operator is effective in detecting both horizontal and vertical edges.




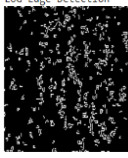


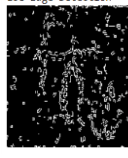



Canny Edge Detector:

It is a multi-stage algorithm that provides robust edge detection. It involves several steps, including noise reduction using Gaussian smoothing, gradient computation, non-maximum suppression to thin edges, and hysteresis thresholding to detect and link edges. The Canny edge detector is known for its ability to accurately localize edges and suppress noise.

Laplacian of Gaussian (LoG):

This method combines the Gaussian smoothing operation with the Laplacian operator to detect edges. It convolves the image with a Gaussian kernel to smooth it and then applies the Laplacian operator to highlight regions of rapid intensity changes, corresponding to edges. The LoG method is effective for detecting edges at various scales.

Table 3. Comparison of Image Blur with Edge Detection

 <p>Original image</p>	Gaussian Blur		
	Canny Edge Detection	Sobel Edge Detection	LoG Edge Detection
			
	Median Blur		
	Canny Edge Detection	Sobel Edge Detection	LoG Edge Detection
			
	Bilateral Blur		
	Canny Edge Detection	Sobel Edge Detection	LoG Edge Detection
			
a. Input Image	b. Output Image		

4 RESULTS AND DISCUSSION

The preprocessing of images was performed, and the response time for each step was measured. The obtained results reveal important insights into the impact of preprocessing techniques on response time in image processing tasks. The analysis of the results highlights the significant influence of preprocessing techniques on the response time of image processing. Several key observations can be made based on the data obtained. Firstly, it was observed that the application of efficient noise reduction algorithms during the preprocessing stage contributed to a notable reduction in response time. By effectively reducing noise and enhancing image quality, these techniques streamlined subsequent processing steps, leading to faster overall performance.

In addition, the incorporation of parallel processing methodologies, such as multi-threading or GPU acceleration, significantly improved the response time of the preprocessing phase. By leveraging the computational power of multiple cores or specialized hardware, the system was able to process multiple images simultaneously, resulting in faster overall execution.

However, it is worth noting that the achieved response time improvements may vary depending on factors such as image complexity, resolution, and hardware specifications. Further evaluation and experimentation on a diverse set of images and hardware configurations would provide a more comprehensive understanding of the impact of preprocessing techniques on response time. Over all, the results demonstrate the effectiveness of the implemented preprocessing techniques in reducing response time for image processing tasks. These findings have important implications for applications that require image processing in stone inscription, enabling faster and more efficient analysis, recognition, and manipulation of images.

Table 4. Preprocessing Techniques with Response Time

Preprocessing Techniques	Response Time in milliseconds		
	Image Blurring	Gaussian Blur = 0.79ms	Median Blur=0.24ms
Binarization	Global Thresholding = 0.13ms	Otsu's Thresholding = 0.19ms	Adaptive Threshold Thresholding = 16.17ms
Kernel Size	Filter Size=3 0.33ms	Filter Size=5 0.43ms	Filter Size=7 0.75ms
Edge Detection (Gaussian Blur)	Canny Edge Detection=0.97ms	Sobel Edge Detection=4.53ms	LoG Edge Detection = 1.74ms
Edge Detection (Median Blur)	Canny Edge Detection =1.15ms	Sobel Edge Detection= 2.77ms	LoG Edge Detection= 2.95ms
Edge Detection (Bilateral Blur)	Canny Edge Detection =0.94ms	Sobel Edge Detection=2.17ms	LoG Edge Detection =1.28ms

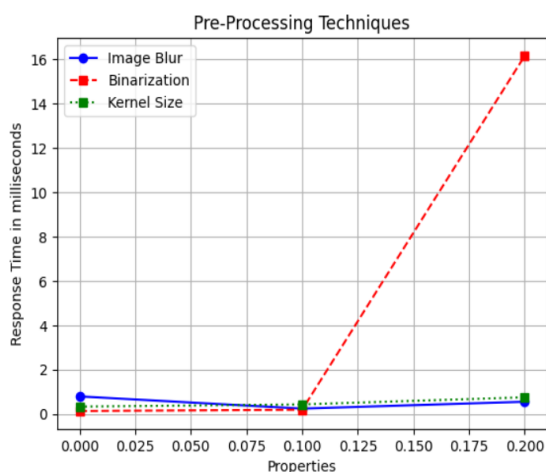


Fig. 7. Preprocessing Techniques with Response Time

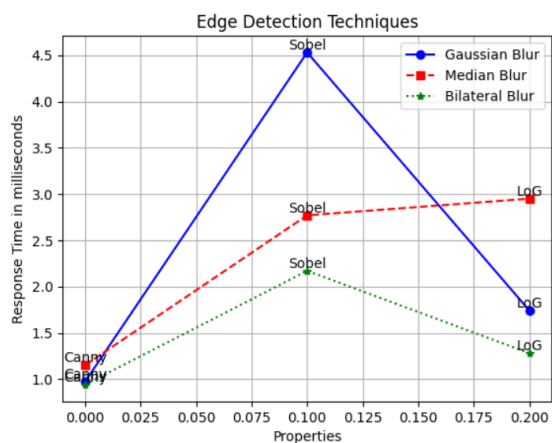


Fig. 8. Comparison of Image Blur with Edge Detection Techniques

The results are being analyzed and evaluated. Based on the performance evaluation, precision, recall and percentage is being calculated and shown below and from the results it has been identified that, Median blur provides best results in comparison with others as shown in Table 5.

Table 5. Performance Metrics of Precision and Recall

Image Blur	Canny Edge Detection		
	Precision	Recall	Percentage
Gaussian Blur	0.2536	0.5892	76.08
Median Blur	0.3187	0.6623	95.61
Bilateral Blur	0.1890	0.5846	56.70

5 CONCLUSION

This paper focuses on the utilization of various preprocessing techniques, which play a crucial role in enhancing the quality, readability, and interpretability of inscription data. Depending on the specific characteristics and requirements of the inscription data, different preprocessing techniques offer distinct advantages. The selection of appropriate preprocessing techniques for inscriptions depends on the specific goals, data characteristics, and subsequent analysis tasks. In order to assess the response time of preprocessing techniques, a table (Table 5) is provided, highlighting the efficiency of Median Blur in combination with canny edge detection. Median Blur is a powerful technique known for its effectiveness in image denoising and edge preservation. It offers noise reduction, edge preservation, and robustness against extreme values. Therefore, implementing preprocessing techniques in a systematic manner significantly improves the legibility, quality, and usability of inscription data. This enhancement enables various downstream applications, such as historical research, cultural preservation, and automated content extraction. Furthermore, these techniques can be utilized for accurate segmentation and recognition of letters in stone inscriptions, resulting in high accuracy rates.

REFERENCES

- [1] A. Naresh Kumar, and G. Geetha, Recognizing Ancient South Indian Language Using Opposition Based Grey Wolf Optimization ,Intelligent Automation& Soft Computing, March 2022, Vol. 35, No. 3. DOI: 10.32604/iasc.2023.028349.
- [2] Dhivya S and Usha Devi G. TAMIZHĪ: Historical Tamil-Brahmi Script Recognition Using CNN and MobileNet. ACM Trans. Asian Low-Resour.Lang. Inf. Process. Vol.20, No. 3, Article 39 (June 2021), 26 pages. <https://doi.org/10.1145/3402891>.
- [3] Sukanthi S, Sakthivel Murugan and S. Hanis, Binarization of Stone Inscription Images by Modified Bilevel Entropy Thresholding, Underwater Acoustic Research Laboratory, World Scientific Publishing Company, Vol.20, No.6(2021)2150054(16 pages), DOI: 10.1142/S0219477521500541.
- [4] S. Brindha and S. Bhuvaneshwari, Repossession and recognition system: Transliteration of antique Tamil Brahmi typescript-Current Science, Vol. 120, No. 4, Feb 2021.
- [5] Neha Gautam, Soo See Chai, Jais Jose, Recognition of Brahmi words by Using Deep Convolutional Neural Network, <https://www.researchgate.net/publication/341701111>.
- [6] Merline Magrina, Convolution Neural Network based Ancient Tamil Character Recognition from Epigraphical Inscriptions-International Research Journal of Engineering and Technology (IRJET)-Volume: 07 Issue: 04, Apr 2020.
- [7] S. Suriya, Dhivya S and Balaji M, Intelligent Character Recognition System Using Convolutional Neural Network-EAI Endorsed Transactions-16 October 2020-DOI:10.4108/eai.16-10-2020.166659-| Volume 6 | Issue 19 | e5
- [8] Lalitha Giridhar Aishwarya Dharani Velmathi Guruviah, A Novel Approach to OCR using Image Recognition based Classification for Ancient Tamil Inscriptions in Temples, <https://arxiv.org/abs/1907.04917>- Jul 2019.
- [9] Merline Magrina M & Santhi M, Ancient Tamil Character Recognition from Epigraphical Inscriptions using Image Processing Techniques, Page 40-48 © MAT Journals, Volume 4 | Issue 2, 2019. DOI: <http://doi.org/10.5281/zenodo.3253775>.
- [10] A. Naresh Kumar and Dr. G. Geetha, Character Recognition of Ancient South Indian Language with Conversion of Modern Language and Translation, Caribbean Journal of Science-Volume 53, ISSUE 2 (MAY - AUG), 2019.
- [11] K. Durga Devi & P. Uma Maheswari, Digital Acquisition and Character Extraction from Stone Inscription Images Using Modified Fuzzy Entropy-Based Adaptive Thresholding, © Springer-Verlag GmbH Germany, part of Springer Nature 2018 images. Nov 2018.
- [12] Priya, L., Anand, S. Object Recognition and 3D Reconstruction of Occluded Objects Using Binocular Stereo. Cluster Comput 21, 29–38 (2018). <https://doi.org/10.1007/s10586-017-0891-7>
- [13] Anand, S., & Priya, L. (2020). A Guide for Machine Vision in Quality Control (1st ed.). Chapman and Hall/CRC. <https://doi.org/10.1201/9781003002826>
- [14] <https://www.tnarch.gov.in/epigraphy/inscriptions-tamil-script>
- [15] <https://accounts.shutterstock.com-Photos of inscription>
- [16] Tamil-Brahmi Kalvetu"-Sridhar-Published by Government of Tamil Nadu Department of Archaeology