A Novel Preprocessing Technique for the Preservation of Tamil Brahmi Letters on Ancient Inscriptions in Different Application Domain

Poornimathi.K[1],Muralibhaskaran.V[2],Priya.L[3]
Assistant Professor(SG) [1],
Professor[2,3],
RajalakshmiEngineering College[1,2,3]
[1]poornimathi.k@rajalakshmi.edu.in ,
[2]muralibhaskaran.v@rajalakshmi.edu.in ,
[3]priya.l@rajalakshmi.edu.in
+91 9790719507

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.***

I. 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. These periodicals have been classified into three categories based on the thorough analyses of grammatical and lexical variations, namely: Old Tamil (from about 450 BCE to 700 CE), Middle Tamil (700–1600), and Modern Tamil (since 1600). Tamil-Brahmi, also known as Tamili, was a variant of the Brahmi script in southern India. It was used to write inscriptions in the early form of Old Tamil. The Tamil-Brahmi script has been paleo graphically and stratigraphically dated between the third century BCE and the first century CE, and it constitutes the earliest known writing system evidenced in many parts of Tamil Nadu, Kerala, Andhra Pradesh and Sri Lanka. Tamil Brahmi inscriptions have been found on cave entrances, stone beds, potsherds, jar burials, coins, seals, and rings.

From the 5th century CE onwards Tamil is written in Vatteluttu in the Chera and Pandya country and Grantha or Tamil script in the Chola and Pallava country. 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 (□□□□□□□□□□□□□□□)

c.Inscription at muthalaikulam (□□□□□□□□□□□□□)

d.Inscription at Aanaimalai (□□□□□□)

Fig. 1 *Sample Images of Tamil Inscription*

II. 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 S et al (2021) suggested a few steps for efficient character recognition. Training the CNN model from scratch with a SoftMax classifier in a sequential model. Using Mobile Net: Transfer learning paradigm from a pre-trained model on a Tamizhi dataset. Building a model with CNN and SVM. SVM for evaluating the best accuracy to recognize handwritten Brahmi characters The accuracy of the trained and tested Mobile Net model for the datasets of vowels (8 classes), consonants (18 classes), and consonant vowels (26 classes) is 98.1%, 97.7%, and 97.5%, respectively. Data Collection Techniques: Approximately 1,600 engineering students from the SRM Institute of Science and Technology contributed 1,090,000 isolated samples to the character database. The information was stored in CSV format. The handwritten typefaces had a grid-like layout. Each grid cell was segmented using the Hough Transform method. Using OPENCV, the image is segmented into distinct scripts. The algorithm for character segmentation is used to extract handwritten characters. Images of segmented characters are classified into 209 classifications. To evaluate the dataset, a CNN model and SoftMax classifier are combined to recognize the script. This research may convince historians, anthropologists, and scholars from other fields to utilize the source material and gain a deeper comprehension of ancient Tamil culture.[2]

The diverse thresholding techniques (Sukanthi et al., 2021) The binarization of images of terrestrial and underwater stone inscriptions is preceded by contrast enhancement and followed by edge-based filtering that reduces noise and sharpens edges. The modified bi-level thresholding (MBET) algorithm is proposed and compared to several existing thresholding algorithms, including the Otsu method, Niblack method, Sauvola method, Bernsen method, and Fuzzy C means method. The proposed MBET algorithm, with its adaptive local thresholding feature, is expected to minimize noise and extract the margins of the objects in both terrestrial and underwater images flawlessly.[3]

The technique (Brindha et al., 2021) for deriving the features and converting the ancient Tamil script to its modern form is proposed. The processed image is subjected to a new feature extraction technique in which the system uses a chi-square test to determine whether or not all the zoning feature values are independent or dependent. Neural networks recognize the characters using the extracted features. The featured image is trained with NNTool, and the data are compared to the database in order to recognize Brahmi characters. The recognized characters are converted into modern Tamil letters, and the resulting HTML output is displayed. The image features are extracted using Zernike moments and zoning features, two novel feature extractions. To ascertain whether the values of the vectors are independent or dependent, a chi-square test is conducted. The accuracy rate of 91.3% is achieved using the confusion matrix. By employing sophisticated algorithms, the method can be expanded to larger datasets, online character recognition, and improved precision.[4]

Neha Gautam et al. (2020) propose using a deep convolutional neural network with dropout to recognize Brahmi words. In addition to proposing a DCNN for Brahmi word recognition, this study conducts a series of experiments on a conventional Brahmi dataset. On a publicly accessible Brahmi image database, the practical operation of this method was systematically tested, obtaining a 92.47% recognition rate by CNN with dropout. This study contrasted the outcomes of various methods in order to make recommendations based on parameter tuning. In addition, determining the optimal parameters for producing error-free results is a research problem. Similarly, complex future tasks such as character recognition of rotated, mirror-text, and chaotic images could benefit from the extraction of novel features.[5]

In this paper (Merline et al., 2020), training of an 18-layer CNN for 73 class character recognition problems was conducted. This CNN architecture is trained to extract sample features using the ReLU activation function. CNN can autonomously learn a unique set of image-based features in a hierarchical fashion. We attained the Segmentation Rate and Recognition Rate using our framework by mapping the From Ancient Tamil to Modern Tamil characters rate. The proposed work concentrates on the Simple Convolution Neural Network for image classification, which incurs less computational expense. The results demonstrate that CNN is capable of obtaining good results on the Tamil Dataset through supervised learning alone. We did not use an unsupervised, pre-trained network, despite the fact that we believe it would have facilitated the work.[6]

Using a Convolutional Neural Network, the proposed method (Suriya et al., 2020) is able to recognize characters in challenging conditions where traditional character recognition systems fail, such as in the presence of low resolution, substantial blur, low contrast, and other distortions. The following significant challenges can be investigated further in the future: This work utilizes HP Labs India's Isolated Handwritten Tamil Character data set. As a result, it has been determined that among the proposed algorithms, various CNN models produce different results, and the one that provides the highest recognition accuracy is superior. [7]

The proposed work (Lalitha et al., 2019) concentrates on enhancing optical character recognition techniques for 7th- to 12th-century Tamil script. After the image is binarized using the Otsu thresholding method, a two-dimensional convolutional neural network is defined and used to train, classify, and recognise ancient Tamil characters. The neural network is attached to Tesseract via the Python Pytesseract library in order to implement optical character recognition techniques. This work incorporates Google's text-to-speech voice engine to generate an audio output of the digitized text as an added feature. This research endeavored to develop a universally applicable OCR system with audio output for the ancient Tamil script. Using CNN and Image Recognition techniques, an operational system for modern and ancient Tamil was developed. The effectiveness of OCR techniques on ancient Tamil scripts can be enhanced through the addition of more data. The development of a language parser to aid in the segmentation of the digitized script and enhance the accuracy of the audio output is another possible extension of our work.[8]

The proposed work (Merline et al., 2019) detailed the historical events of the Chola period in the 12th century. Using OCR technology, ancient Tamil characters carved into stones are identified. Ensemble learning and KNN are used to classify the characters, and Unicode is then used to match the classified characters. The images undergo pre-processing to remove noise using median filters, segmentation using bounding boxes, and extraction of features. The extracted features are applied to the Ensemble learning classifier. Using Unicode, the modern Tamil character is mapped. [9,13]

A new system for improving stone inscription images is proposed by Durga et al., 2018.IBF is used to eliminate extraneous noise while maintaining character with edges. The proposed fuzzy system aids in predicting character and background pixel uncertainty. [11,12]

Summary of Literature is shown below for the user's understanding.

TABLE I
Existing techniques- Summary

Author	Year	Technique Used	Limitation
Naresh et. al .,	2022	Artificial Neural Network (ANN) with the Opposition-based Grey Wolf Optimization Algorithm (OGWA)	Better performance by ANN
Dhivya S et. al.,	2021	Convolutional Neural Network (CNN) with Mobile Net and SVM	Identification of Handwritten inscription only.
Sukanthi et.al.,	2021	Edge-based filtering, modified bi-level Entropy thresholding (MBET)	PSNR and SD for the terrestrial stone surface-based images with 43% and 39% on an average and for underwater stone inscription images 49% and 39% on an average.
Brindha et. al.,	2021	Neural Network (NN) tool	The 91.3% accuracy rate is achieved using the confusion matrix.
Merline et. al.,	2020	Convolutional Neural Network (CNN)	Improve the efficiency by considering the strokes, style and size of characters.
Suriya et. al.,	2020	Convolutional Neural Network (CNN)	Handwritten recognition only.

Merline et. al.,	2019	OCR and KNN	The performance metric of segmentation & recognition rates can be improved
Naresh et. al.,	2019	OCR, Advanced Maximally stable extremal regions and Affine Invariant Intensity Extreme Based (AMSER)	The geometric features such as edges and blob can be considered for efficient segment. Light illumination needs to be added.
Durga et. al.,	2018	Modified Fuzzy Entropy-based Adaptive Thresholding (MFEAT) with degree of Gaussian membership function and iterative bilateral filter (IBF).	84.98% of accuracy in extracting the characters from stone inscriptions

III. 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.

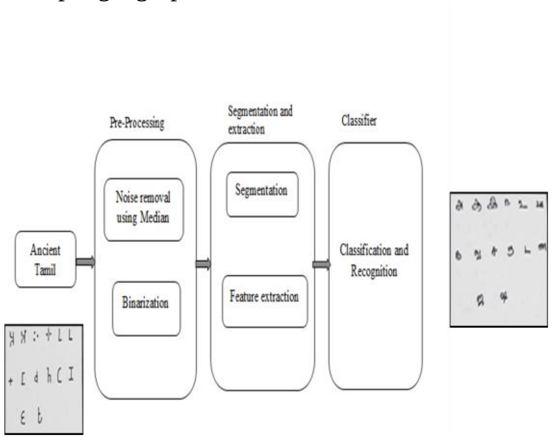

Pre-Processing
Segmentation and extraction
Classifier
Ancient Tamil
Noise removal using Median
Binarization
Segmentation
Feature extraction
Classification and Recognition

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

a. 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.

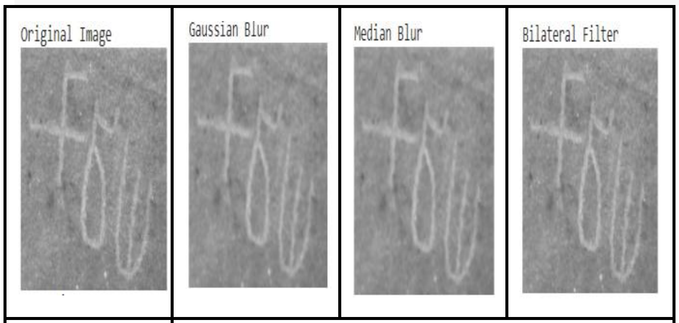
Bilateral Blur : It considers both spatial proximity and pixel intensity similarity to perform blurring. It preserves edges and details while reducing noise, resulting in a smoother image with preserved structure.

code snippet for preprocessing is given below for better understanding.

a. Image Blurring

```
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')
cv2_imshow(bilFilter)
```



a. Input image	b. Output image

Fig.3 Image Blurring

b. 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

It is a simple and widely used technique where a single threshold value is applied to the entire image. Pixels with intensity values above the threshold are classified as foreground, while those below the threshold are considered background. This method assumes a bimodal histogram, where there is a clear separation between foreground and background intensities.

```
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.

<u>Steps to be followed:</u>

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.

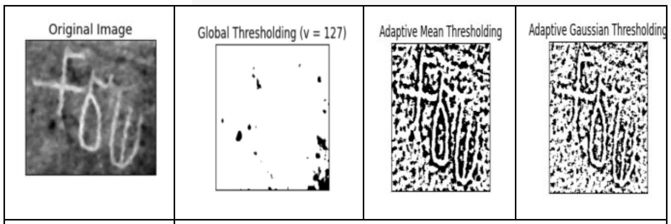
<u>Steps to be followed:</u>

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).

Codesnippet:

```
#Adaptive Thresholding
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread('/kalvettu2.png',
cv.IMREAD_GRAYSCALE)
assert img is not None, "file could not be read, check with
os.path.exists()"
img = cv.medianBlur(img,5)
ret,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
th2 =
cv.adaptiveThreshold(img,120,cv.ADAPTIVE_THRESH_M
EAN_C,\
 cv.THRESH_BINARY,11,2)
```

```
th3 =
cv.adaptiveThreshold(img,120,cv.ADAPTIVE_THRESH_G
AUSSIAN_C,\
 cv.THRESH_BINARY,11,2)
titles = ['Original Image', 'Global Thresholding (v = 127)',
 'Adaptive Mean Thresholding', 'Adaptive Gaussian
Thresholding']
images = [img, th1, th2, th3]
for i in range(4):
 plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
 plt.title(titles[i])
 plt.xticks([]),plt.yticks([])
plt.show()
```


Original Image
Global Thresholding (v = 127)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

a.Input Image b.Output Image

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
```

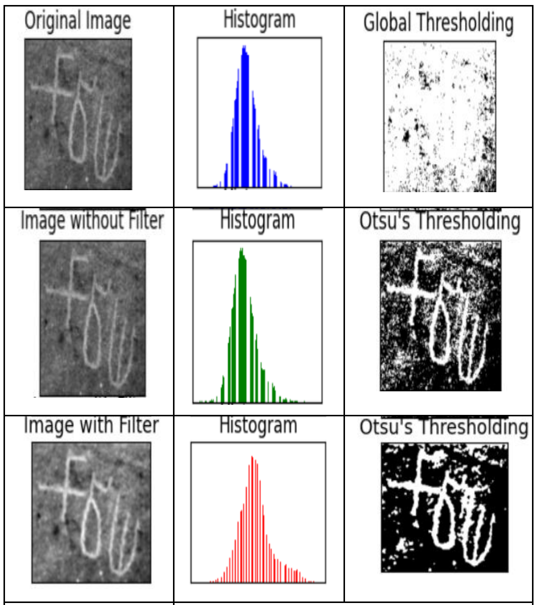
Code snippets

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread('/kalvettu2.png',
cv.IMREAD_GRAYSCALE)
assert img is not None, "file could not be read, check
with os.path.exists()"
# global thresholding
ret1,th1 =
cv.threshold(img,127,255,cv.THRESH_BINARY)
# Otsu's thresholding
ret2,th2 =
cv.threshold(img,0,255,cv.THRESH_BINARY+cv.T
HRESH_OTSU)
# Otsu's thresholding after Gaussian filtering
blur = cv.GaussianBlur(img,(5,5),0)
```

```
ret3,th3 =
cv.threshold(blur,0,192,cv.THRESH_BINARY+cv.T
HRESH_OTSU)
# plot all the images and their histograms
images = [img, 0, th1, img, 0, th2, blur, 0, th3]
color = ['b','g','r']
titles = ['Original Image','Histogram','Global
Thresholding',
 'Image without Filter','Histogram',"Otsu's
Thresholding",
 'Image with Filter','Histogram',"Otsu's
Thresholding"]
for i in range(3):

plt.subplot(3,3,i*3+1),plt.imshow(images[i*3],'gray'
)
 plt.title(titles[i*3]), plt.xticks([]), plt.yticks([])
plt.subplot(3,3,i*3+2),plt.hist(images[i*3].ravel(),2
56,color=color[
  i])
 plt.title(titles[i*3+1]), plt.xticks([]), plt.yticks([])

plt.subplot(3,3,i*3+3),plt.imshow(images[i*3+2],'g
ray')
 plt.title(titles[i*3+2]), plt.xticks([]), plt.yticks([])
plt.show()
```



a. Input Image b. Output Image

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.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
image = cv2.imread('/content/kalvettu2.png')
# making filter of 3 by 3 filled with 1 divide by 9 for normalization
blur_filter1 = np.ones((3, 3), float)/(9.0)
# making filter of 5 by 5 filled with 1 divide by 25 for normalization
blur_filter2 = np.ones((5, 5), float)/(25.0)
# making filter of 7 by 7 filled with 1 divide by 49 for normalization
blur_filter3 = np.ones((7, 7), float)/(49.0)

image_blur1 = cv2.filter2D(image, -1, blur_filter1)
image_blur2 = cv2.filter2D(image, -1, blur_filter2)
image_blur3 = cv2.filter2D(image, -1, blur_filter3)

plt.subplot(2,2,1),plt.imshow(image,cmap = 'gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,2),plt.imshow(image_blur1,cmap = 'gray')
plt.title('Image_Blur(3*3)'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,3),plt.imshow(image_blur2,cmap = 'gray')
plt.title('Image_Blur(5*5)'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,4),plt.imshow(image_blur3,cmap = 'gray')
plt.title('Image_Blur(7*7)'), plt.xticks([]), plt.yticks([])

cv2.waitKey(0)
cv2.destroyAllWindows()
```

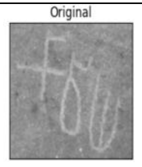
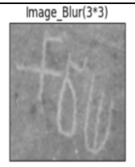
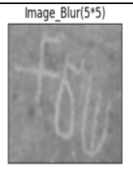
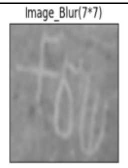
			
a. Input Image	b. Output Image		

Fig.6 Kernel Size

c. 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:

<u>Steps Followed:</u>

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. II
Comparison of Image Blur with Edge Detection

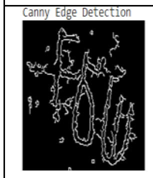
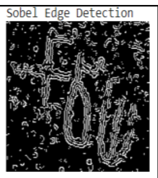
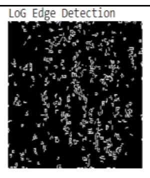
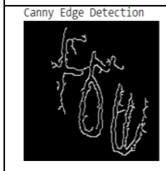
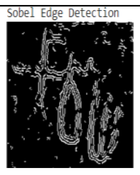
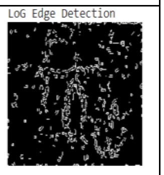
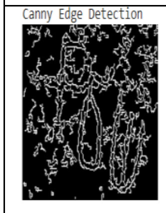
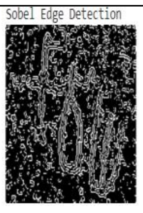
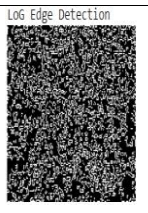
	Gaussian Blur		
			
	Median Blur		
			
	Bilateral Blur		
			
a. Input Image	b. Output Image		

Fig.6 ***Comparison of Image Blur with Edge Detection***

IV.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 III
Preprocessing Techniques with Response Time
2

Preprocessing Techniques	**Response Time in milliseconds**		
Image Blurring	Gaussian Blur=0.79ms	Median Blur=0.24ms	Bilateral Blur=0.55ms
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

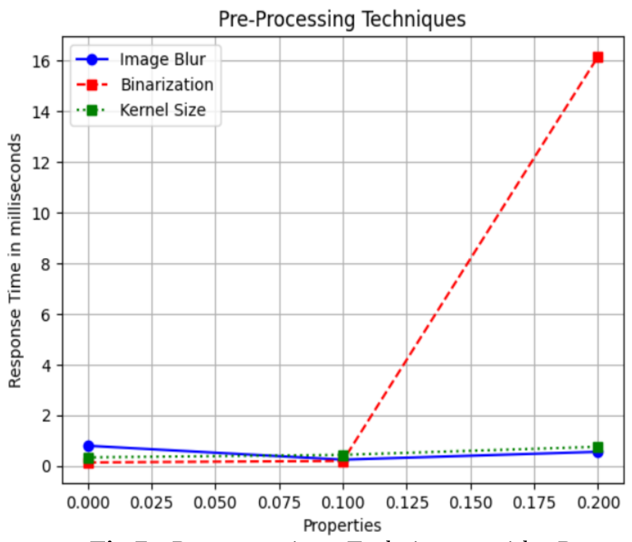


Fig.7 *Preprocessing Techniques with Response Time*

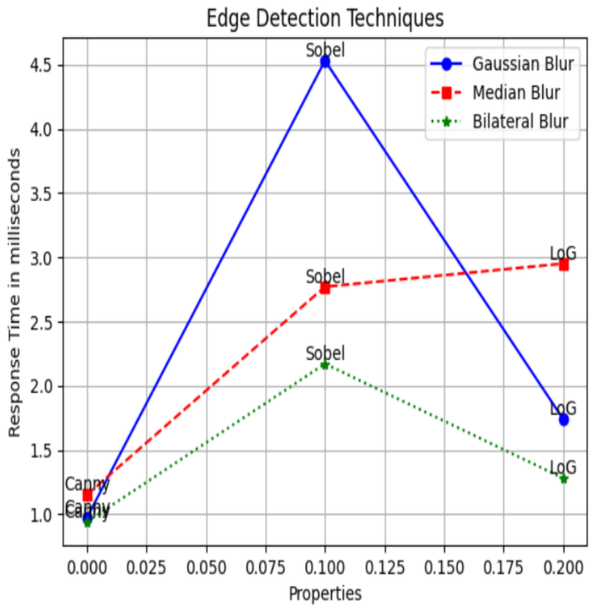


Fig.8 *Comparison of Image Blur with Edge Detection Techniques*

V.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 III) 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.TAMIZHİ: 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 InscriptionImagesbyModifiedBilevelEntropyThresholding,Underwater AcousticResearchLaboratory,WorldScientificPublishingCompany,Vol. 20,No.6(2021)2150054(16 pages),DOI: 10.1142/S0219477521500541.

[4] S. Brindha and S. Bhuvaneswari, 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 byUsingDeepConvolutionalNeuralNetwork,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 SystemUsingConvolutional Neural Network-EAI Endorsed Transactions-16October2020-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 AncientTamilInscriptionsinTemples,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,Page40-48©MATJournals,Volume4|Issue2,2019.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 Kalvettu”-Sridhar-Published by Government of Tamil Nadu Department of Archaeology