

Evaluating Handwritten Character Recognition with Hu Moments and K-Nearest Neighbors Algorithm

Bojan Despodov¹, Done Stojanov¹, Cveta Martinovska Bande¹

¹ Faculty of Computer Science, Goce Delcev University, Krste Misirkov, 10A, 2000 Stip, North Macedonia

Abstract – Handwritten character recognition remains a challenging task in various domains, including handwriting analysis and document digitization. This task poses a significant challenge because of the variability and complexity of handwriting. To address this, it is crucial to develop different methodologies that enhance accuracy and efficiency in recognition. This research evaluates the discrimination capacity of a classifier that combines together the Hu Moments features extraction technique and the k-nearest neighbors (KNN) algorithm for the purpose of handwritten character recognition. The authors worked with handwritten characters from the English alphabet where each character is represented in two handwritten forms. The average accuracy of recognition was 82.69%, that conforms the efficiency of the proposed classifier. This research contributes to the field of automated handwritten recognition and highlights the necessity for improved techniques in character analysis, which has implications for handwriting recognition in different contexts, linguistic studies, document digitization, and other applications where character recognition is crucial. This is the first study that aims to apply the combination of Hu Moments features extraction technique in combination with the KNN algorithm for purpose of handwritten character recognition.

Keywords – Handwritten Character Recognition, Hu Moments, K-Nearest Neighbors (KNN).

1. Introduction

Handwritten character recognition, also known as handwriting recognition or HCR, is a technology that aims to convert handwritten text into a machine-readable digital text [23], [25]. It is a subset of optical character recognition (OCR) [19], which is employed in various applications, such as digitizing handwritten documents, processing forms, and enabling natural input methods for electronic devices. The performance of HCR primarily depends of quality of the source data [19] and the choice of features used to describe input samples [20]. Handwritten character recognition systems typically involve the use of pattern recognition algorithms to analyze and interpret the shape and structure of handwritten characters. HCR includes three stages: preprocessing, feature extraction, and classification prior to the actual recognition [20].

There are two types of handwritten recognition: online and offline [22]. Offline handwritten character recognition extends the challenge of handwritten character recognition by considering instances where the handwriting is not constrained to predefined areas or templates. Unlike online recognition, which involves capturing the dynamic process of writing (e.g., through stylus movements on a digital tablet), offline recognition deals with static images of handwritten text, such as scanned documents or photographs [22]. The challenges in offline handwritten character recognition are increased due to the lack of temporal information about the writing process [16]. Variability in writing styles, intra-class variations, and contextual dependencies become more pronounced when dealing with static images. As an effort to maximize inter-class differences and minimize intra-class differences, [6] used conditional log-likelihood loss function. [30] presented loss-based metric learning character recognition algorithm in addition to ResNet network for the same purpose.

DOI: 10.18421/TEM133-10

<https://doi.org/10.18421/TEM133-10>


Corresponding author: Bojan Despodov,
Faculty of Computer Science, Goce Delcev University,
Krste Misirkov, 10A, 2000 Stip, North Macedonia
Email: bojan.31021@student.ugd.edu.mk

Received: 13 March 2024.

Revised: 03 July 2024.

Accepted: 18 July 2024.

Published: 27 August 2024.

 © 2024 Bojan Despodov, Done Stojanov & Cveta Martinovska Bande; published by UIKTEN. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License.

The article is published with Open Access at <https://www.temjournal.com/>

Additionally, offline recognition systems must contend with the complexities of noise, distortions, and variations in lighting conditions that are inherent in real-world documents [1]. Efforts in this field focus not only on improving recognition accuracy but also on developing systems that can handle diverse languages, scripts, and writing styles. Despite significant progress, achieving human-level performance in offline handwritten character recognition remains a complex problem due to the inherent variability and intricacies of handwritten text. Ongoing research aims to refine existing models, explore novel techniques, and leverage diverse datasets to push the limits of what can be achieved in this challenging domain.

Since 1962, Hu's moments [13] have been used as a major computational technique for objects features extraction. They rely on the basic principles of algebraic invariants theory that the features of the object do not alter on the operations: translation, rotation, scaling, and reflection [3]. As such, they are powerful features extraction technique, which either in its native or modified form, have many successful implementations. Hu moments were used for digits recognition [29], human body shape recognition [5], sign language recognition [21]. In order to perform object recognition, Hu moments are often combined with powerful classifiers [12], such as: support vector machine (SVM), k-nearest neighbors (KNN) [15] or convolutional neural network [26]. Guided by the successful implementations of the k-nearest neighbors (KNN) algorithm for face detection [8] and MRI segmentation [24], it is assumed that the same algorithm may be also applied for handwritten character recognition.

This study aims to evaluate the discrimination ability of Hu moments features extraction technique [13] combined with the k-nearest neighbors (KNN) algorithm [7] for the purpose of automated handwritten character recognition. The English alphabet as a source dataset is used. Each letter was written in capital, presented with two forms, providing an input set of 52 samples, organized into one image. The authors extracted characters from the input image and applied the steps of: preprocessing, features extraction and classification. Apart from the grayscale conversion and later on inversion, adaptive thresholding was applied [2], [4], [28], which seemed to be better option than fixed thresholding [18], as it provided better demarcation of the shapes of characters out of the background. Additionally, Gaussian blur [10] was applied on the image resulting from adaptive thresholding and inversion. The seven moments of Hu were used to model each character features, thus having each character represented as a unique vector of seven real numbers - Hu moments. In this fashion, same-letter characters,

even written into an alternative form, are expected to be represented by Hu vectors at minimum distance. Manhattan distance [17] to compute the inter-vector distance in addition to the KNN algorithm that searched for the nearest neighbor ($k=1$), thus performing character classification. The leave-one-out cross validation [27] is used as a widely accepted technique for classifier performance evaluation in the absence of abundant labeled data. During the cross-validation, one character is used as a test sample and the remaining 51 characters as a training set. The results obtained from this method are promising, with on average accuracy of character recognition of 82.69%.

This study contributes to advancements in handwritten character recognition, a field where improvements have far-reaching implications for historical script analysis, document digitization, and various other applications. In the sections that follow, the methodology, experimental setup, and findings of this research will be delved into, providing a comprehensive understanding of the process and results of handwritten character recognition methodology.

2. Materials and Methods

Figure 1 shows the input dataset used in the experiment. The English alphabet was used, having each letter written twice in different shape, thus working with an input set of 52 characters. During the later on performance evaluation, each letter was used as a test sample and the remaining 51 characters to train the classifier.

A A
 B B
 C C
 D D
 E E
 F F
 G G
 H H
 I I
 J J
 K K
 L L
 M M
 N N
 O O
 P P
 Q Q
 R R
 S S
 T T
 U U
 V V
 W W
 X X
 Y Y
 Z Z

Figure 1. Input image of the handwritten English alphabet

The two distinct handwritten forms of each character are notably more similar to one another than to different characters, making them a test case for handwritten character recognition. Between the characters, there are differences in translation, rotation, scale, reflection, style, orientation, size, stroke thickness, curvature, and slant. These diverse variations contribute to the complexity of character representation and pose challenges for the handwritten character recognition system.

2.1. Image Pre-processing

Image preprocessing plays a central and foundational role in the world of character recognition. It serves as the base upon which the processes of feature extraction and character classification is built. Its significance lies in making sure the data that is worked with is of high quality and suitable for character recognition. Character recognition systems have the challenging task of understanding handwritten characters. This task is tricky because people have different ways of writing, and the quality of handwriting varies widely. Moreover, documents can come in all sorts of conditions. Image preprocessing steps aim to cope with these challenges, cleaning and simplifying the initial data, making it easier for computers to understand. The image preprocessing process includes steps like turning the image into grayscale, setting thresholds, inverting colors, applying gaussian blur, and finding contours. These steps are crucial for later classification. In the following sections the preprocessing steps that have been applied to the input (Figure 1) will be discussed, aimed to improve the overall quality of the source data, that is a prerequisite for successful classification.

2.2. Greyscale Conversion

For the reason that the subsequent step of adaptive thresholding requires an 8-bit single channel grayscale image but the input image is 16-bit single channel grayscale image, the input image was explicitly converted to 8 bits in grayscale (Figure 2). This step, though seemingly straightforward, holds profound importance in the context of character recognition. The purpose of grayscale conversion is to streamline the complexity of the input image. Grayscale conversion is focusing solely on character contours and variations in intensity. This simplification process yields a single-channel image where variations in shading and contrast take center stage. In grayscale images, each pixel is typically represented by a single intensity value.

This reduction in dimensionality means that computational operations, memory usage, and the overall efficiency of subsequent processing are greatly enhanced. The impact of grayscale conversion extends to feature extraction, which forms the core of character recognition. In this study, the focus is on character edges, shape, and contours extractions, which are captured for further analysis.

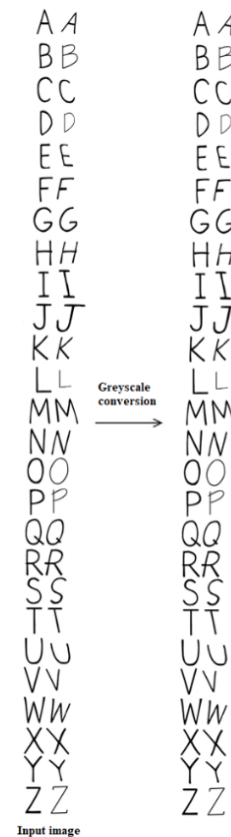


Figure 2. Greyscale conversion of the input image

2.3. Adaptive Thresholding and Inversion

Next, adaptive thresholding and inversion was applied to the image obtained from grayscale conversion. This step aims to separate the image into two parts: the main characters and the background. The aim to demarcate the shape (contour) of the characters out of the background by adaptively adjusting the threshold value for a pixel as the mean of the pixel intensities in the local neighborhood [28]. Through the process of adaptive thresholding, the grayscale image undergoes a transformation, transforming into a binary representation. In this binary environment, characters emerge as stark black shapes against a pristine white canvas, where black and white offer a clear and uniform distinction. This binary representation illuminates the path for characters to stand out prominently against their background.

The dynamic determination of the threshold value is pivotal, as it defines the criteria that pixels must meet to qualify as character and what remains as background.

In this research, employing adaptive thresholding with a local neighborhood of size 11x11 pixels was found to result in improved performance compared to a fixed threshold. The choice of adaptive thresholding parameters is crucial, as they define how pixels are classified as characters or background based on their local context [2].

In tandem with thresholding, inversion plays a pivotal role in ensuring that characters are consistently portrayed as white against the black background, regardless of the initial ink color. This consistency holds paramount importance for subsequent feature extraction and classification stages, as these processes rely on character contours

and shapes best captured against a uniform black backdrop. By inverting the colors, the original white background is turned into black, making character shapes stand out even more. The final result is a uniform representation, where characters are showcased in white against a black background. This clear and consistent depiction not only distinguishes characters from the background but also accentuates character shapes, facilitating precise feature extraction and character classification. Similar way of image preprocessing was also used in [9]. Both thresholding and inversion seamlessly collaborate to craft a binary image that presents characters with clarity, showcasing them in white against a black background for effective feature extraction and classification. The image, resulting from adaptive thresholding and inversion, is presented in Figure 3.

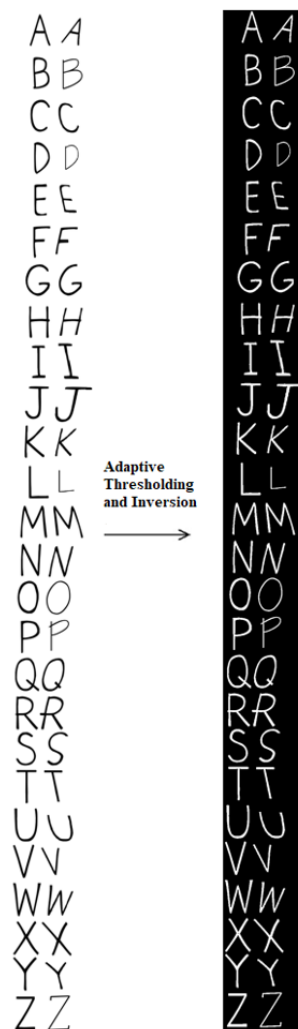


Figure 3. Resulting image from adaptive thresholding and inversion

2.4. Gaussian Blur

A Gaussian blur with a kernel of size 5x5 was applied as an additional preprocessing step to the binary image obtained from adaptive thresholding and inversion (Figure 4).

The Gaussian blur operation operates as a smoothing filter. It reduces high-frequency noise in the image, which could be caused by various factors. The kernel, in this context, is a mathematical matrix that serves as a weight distribution.

The convolution process averages pixel values, effectively smoothing the image and suppressing noise to create a more homogeneous appearance [14]. Smoothing the image through gaussian blur before contour detection can enhance the continuity of pixel intensities along edges [10].

The Gaussian kernel determines the weights applied during convolution, and this research has found that a kernel with a size of 5x5 best contributes to the blurring effect. It helps mitigate small irregularities and gaps in the image, providing a more reliable representation of character boundaries for accurate contour detection.

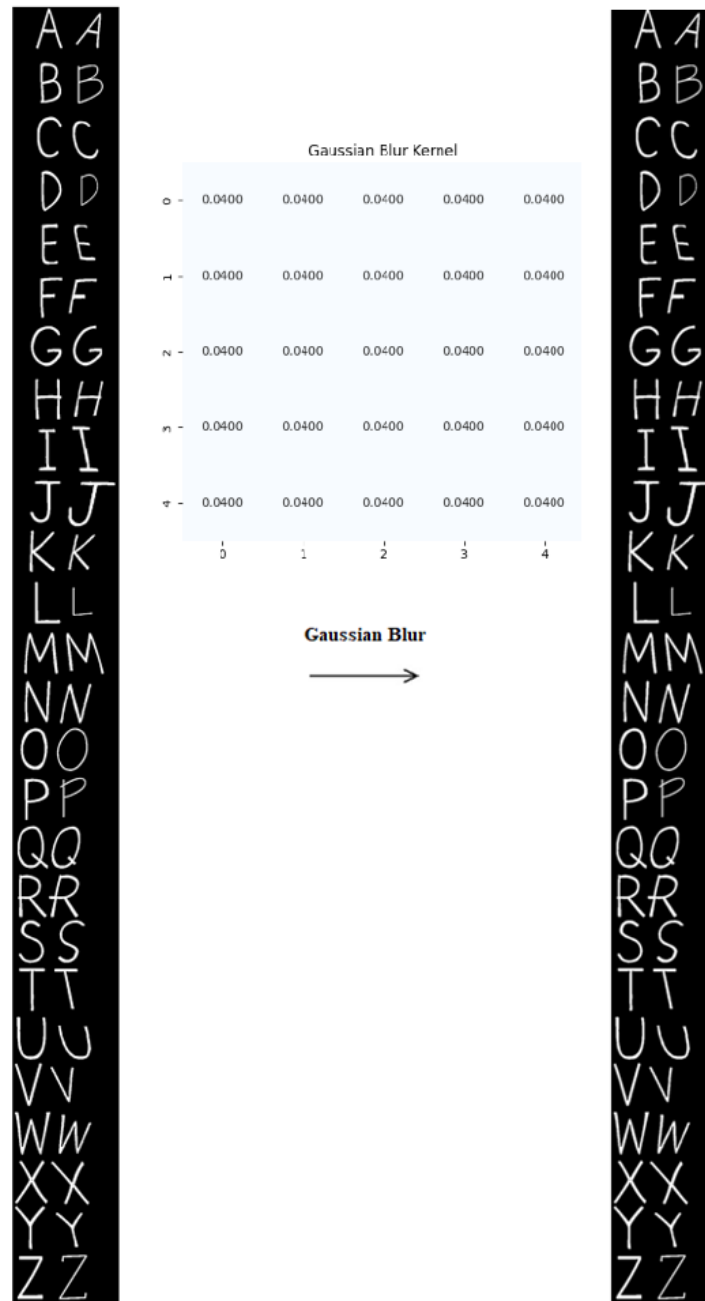


Figure 4. Blurred image after applying gaussian blur with kernel of size 5x5

3. Contour Detection and Features Extraction

Contour detection and feature extraction are fundamental tasks in image processing and computer vision. These techniques play a crucial role in identifying and delineating object boundaries and extracting meaningful patterns or descriptors from images. Contours represent the outline or boundary of objects in an image, highlighting changes in

intensity or color. Feature extraction involves identifying and quantifying important aspects of these contours or regions, such as shape, texture, or other distinctive characteristics. These extracted features serve as inputs for various higher-level algorithms, including object recognition, image segmentation, and pattern analysis. Effective contour detection and feature extraction are essential for different image processing related tasks.

3.1. Contour Detection

Following preparatory steps, contour detection is reached, a crucial phase in the character recognition process. After applying Gaussian blurring with a kernel of size 5x5, the external contours are extracted from the image resulting from the Gaussian blur. In essence, contour detection serves as the gateway to character separation and the in-depth analysis of their structure. Contours not only differentiate characters from the background but also provide a detailed roadmap of their shapes and boundaries. This forms the bedrock for subsequent feature extraction and classification stages, where the character's structural attributes take center stage. With contour detection, the aim is to delineate the defining boundaries that outline each character's structural identity. Contours are pivotal for distinguishing characters from the background and capturing the fundamental shape and structure of the characters themselves. In contour detection, the focus is on identifying the external contours of the characters, which allow us to concentrate on the primary outlines and external

boundaries, emphasizing the character's core shape, while disregarding intricate details or nested contours within the character. This method ensures that all the fine-grained details along the contour are preserved without simplification. This precision is especially valuable when dealing with handwritten characters that exhibit unique and intricate shapes, allowing us to faithfully capture the full character form. In essence, contour detection serves as the gateway to character separation and the in-depth analysis of their structure. These contours not only differentiate characters from the background but also provide a detailed roadmap of their shapes and boundaries.

Figure 5 shows the extracted external contours from the preprocessed image with 52 characters. This forms the bedrock for subsequent feature extraction and classification stages, where the character's structural attributes take center stage. With the characters now distinctly outlined and their structural features unveiled through contour detection, the next phase of feature extraction using Hu Moments [13] is ready to advance.

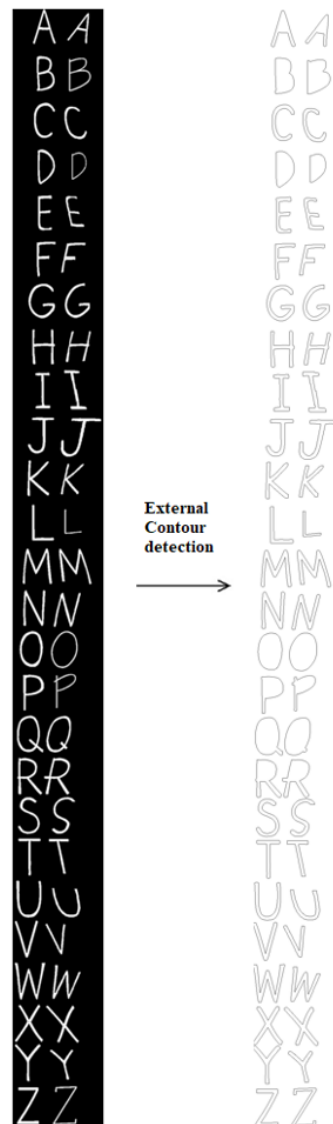


Figure 5. External contours of the handwritten characters

3.2. Features Extraction Using Hu Moments

Feature extraction is a crucial phase that involves distilling the most salient attributes from the characters. These attributes will serve as the foundation for handwritten character recognition. Having successfully applied the preprocessing steps, feature extraction is performed by exploring the remarkable domains of Hu moments. They are a set of seven distinctive moments employed to capture the essential characteristics of the characters. Feature extraction using Hu moments plays a pivotal role in character recognition, offering a quantitative and robust representation of characters' shape and structure. These moments are computed from raw moments, central moments, and normalized central moments. Each of the seven Hu moments conveys specific information about the character's geometry, such as its size, orientation, and distribution of pixel intensities, and are calculated as follows:

$$h_1 = \eta_{20} + \eta_{02} \quad (1)$$

$$h_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \quad (2)$$

$$h_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \quad (3)$$

$$h_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \quad (4)$$

$$h_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (5)$$

$$h_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \quad (6)$$

$$h_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (7)$$

Vector of seven Hu moments is used that encapsulate the unique attributes of each character. The Hu moments are a set of seven scale and rotation invariant features derived from the central moments of an image, denoted by symbols such as η_{20} , η_{02} , η_{11} , η_{30} , η_{12} , η_{21} , and η_{03} . These central moments are integral to the calculation of the Hu moments, facilitating the extraction of invariant features for pattern recognition and image analysis. Breaking down each central moment, η_{20} represents the central moment involving the second power of the horizontal distance from the mean, measuring the concentration of pixel values along the horizontal axis. Similarly, η_{02} involves the vertical distance from the mean and measures the concentration along the vertical axis.

η_{11} , on the other hand, represents the central moment involving the product of the horizontal and vertical distances from the mean, quantifying the correlation between pixel values along both axes. η_{30} captures the third power of the horizontal distance from the mean, providing information about the skewness along the horizontal axis. Moving forward, η_{12} involves the product of the horizontal and vertical distances, reflecting both correlation and elongation of the pixel distribution. Similarly, η_{21} , with reversed roles of horizontal and vertical axes, also reflects correlation and elongation patterns. Lastly, η_{03} captures the third power of the vertical distance from the mean, revealing information about the skewness along the vertical axis. The specific combinations of these central moments in the Hu moment formulas contribute to achieving robustness to various transformations such as translation, rotation, and scaling. In the context of the hu moments, h_1 represents the sum of squared differences between the horizontal and vertical second-order central moments, providing information about the concentration of pixel intensities along both axes. h_2 involves differences in the distribution of pixels along both axes and their correlation, emphasizing patterns and relationships between pixel values. Similarly, h_3 captures the skewness of the distribution of pixels, h_4 focuses on the image's symmetry, and h_5 provides a comprehensive measure of the variation in pixel distribution. Additionally, h_6 and h_7 capture information related to the image's elongation, orientation, and higher-order shape moments. These moments are crucial in achieving robustness to translation, rotation, and scaling, making them valuable for pattern recognition and handwritten character analysis. The ability to recognize patterns despite variations is a key characteristic for applications where invariant features are essential. The feature vector, derived from the binary character images and calculated from the contour information obtained during the contour detection phase, becomes the character's distinctive fingerprint. Hu moments are applied due to the robustness of the character recognition system. Since they are insensitive to various transformations, this model can effectively recognize characters regardless of slight variations in size, orientation, or writing style. This invariance, combined with the comprehensive information encapsulated by Hu moments, results in a feature vector with strong discriminative power. In the process of calculating Hu moments, raw moments, central moments, and normalized central moments are fundamental. The raw moment of a digital image is calculated as follows:

$$M_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q \cdot f(x, y) \quad (8)$$

The values of p and q determine the order of the moment. For example, M_{00} is the zero-order moment, M_{10} is the first-order moment in the x-direction, M_{01} is the first-order moment in the y-direction, and so on. X and y are the spatial coordinates of the pixels in the digital image. The summation of Eq. (8):

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1}$$

goes over all the pixel coordinates in the image, where M is the number of rows and N is the number of columns. $f(x, y)$ represents the intensity or brightness of the pixel at coordinates (x, y) in the digital image. It is the pixel value at a specific location. The formula calculates the raw moment by summing up the product of the pixel intensity $f(x, y)$ and the spatial coordinates x^p and y^q for all pixels in the image. This calculation is performed for all x and y within the specified ranges (from 0 to M-1 for x and from 0 to N-1 for y). The raw moments are useful in image analysis and processing for various applications, such as image recognition, object detection, and feature extraction. They provide information about the distribution of pixel intensities in the image. The choice of p and q determines the specific order of the moment and influences the characteristics captured by the moment.

Central moments are obtained by subtracting the centroid coordinates (\bar{x}, \bar{y}) from the pixel coordinates in the raw moments:

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \bar{x})^p (y - \bar{y})^q \cdot f(x, y) \quad (9)$$

This part of Eq. (9):

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1}$$

denotes a double summation over all pixel coordinates (x, y) in the image. The image has dimensions M (rows) by N (columns), and this sum is taken over all the pixels. These terms: $(x - \bar{x})^p (y - \bar{y})^q$ represent the differences between the pixel coordinates (x, y) and the centroid coordinates (\bar{x}, \bar{y}) raised to the powers p and q, respectively. The centroid coordinates are the mean values of the pixel coordinates in the image.

The part $f(x, y)$ represents the pixel intensity or the value of the image at the pixel coordinates (x, y). The formula calculates central moments by taking the sum of the pixel intensities, each multiplied by the differences between the pixel coordinates and the centroid coordinates raised to certain powers. These moments provide information about the spatial distribution and shape of the pixel intensities in the image, and central moments are often used to describe features like variance, skewness, and kurtosis in image analysis. The use of central moments (subtracting the centroid) helps to make the moments invariant to translation, making them useful for various image processing applications.

Normalized central moments are obtained by dividing the central moments by a scaling factor that makes them scale-invariant. The scaling factor is typically a power of the zeroth central moment (μ_{00}).

$$\mu_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\frac{(p+q)}{2+1}}} \quad (10)$$

In Eq. (10) μ_{00} is the zeroth-order central moment, often denoted as the total mass or total intensity of the image. It is calculated as the sum of all pixel intensities in the image. This expression $\frac{(p+q)}{2+1}$ is the exponent to which the zeroth central moment is raised. It is a scaling factor that helps make the central moments scale-invariant. The factor of (2+1) in the denominator is a common choice in scale-invariant moment calculations. The formula essentially divides the central moment μ_{pq} by a scaling factor involving the zeroth-order central moment. This scaling factor is introduced to achieve scale invariance, meaning that the normalized central moments are less affected by changes in the scale or size of the image. By dividing by μ_{00} raised to the power of $\frac{(p+q)}{2+1}$, the normalized central moments become less sensitive to changes in the overall intensity or size of the image. Normalized central moments are often used in image analysis and pattern recognition because they provide a more robust representation of the shape and distribution of pixel intensities in an image, independent of factors such as translation, rotation, and scale.

In the way just explained, vectors of seven Hu moments were computed for each character, which were used as shape descriptors for the characters. The purpose of the KNN algorithm was to identify the character in the training set whose vector of Hu moments is at minimum Manhattan distance in respect to the vector of Hu moments for the test character.

4. Classification

Following the Hu moments feature extraction phase, characters' classification is performed. For that purpose, the KNN algorithm, a versatile and intuitive algorithm for distinguishing characters based on their feature vectors [12], is used. KNN is a classification algorithm that operates on the principle of proximity, specifically using distance metrics such as Manhattan distance [12]. It classifies an object, in this case a handwritten character of the English alphabet, by comparing it to its K nearest neighbor in the feature space. The purpose of KNN in this methodology is for a given character, based on the Manhattan distance calculation between Hu vectors for the specific character and the others in the training set, to select the character at the smallest distance. This is based on the expectation that characters of similar shape, such as two versions of the letter A, will have the smallest mutual deviation in the values contained in their Hu vectors. In some research support vector machine algorithm is used for classification in combination with the feature extraction using Hu moments [21], [29]. Another notable strength of KNN is its simplicity. It does not rely on complex assumptions or mathematical models. Instead, it makes predictions based on the majority class among its nearest neighbors through a similarity comparison with Manhattan distance. This straightforward approach is particularly advantageous when dealing with handwritten characters, where writing styles can be diverse and intricate. The KNN classifier was chosen because the input image consists of a set of 52 samples, representing a small dataset. Simpler models, like KNN, may perform well with smaller datasets, especially if the underlying assumptions align well with the characteristics of the data. On the other

hand, classifiers with more complex models may often require larger datasets to effectively capture and generalize patterns. These models have a large number of parameters, and training them on a small dataset might lead to overfitting. After experimenting with various distance metrics, Manhattan distance was chosen for the KNN model [11]. Manhattan distance measures the cumulative distance traveled along the axes independently, reflecting the sum of absolute differences in coordinates along each dimension. This characteristic makes it well-suited for characterizing the intricate nuances in writing styles exhibited in diverse versions of handwritten characters within the English alphabet.

To evaluate the performance of the classifier, a leave-one-out cross-validation (LOOCV) methodology is employed [12]. This involves iteratively using one character as a test sample and training the KNN classifier on the remaining characters, and then using it to predict the label of the character that was left out. This process is repeated for each character in the dataset, allowing us to assess the performance of the classifier. Selecting an appropriate K value is crucial. A smaller K may make the algorithm sensitive to noise, while a larger K may result in over-smoothing. This research has determined that K=1 is an optimal setting, effectively balancing the trade-off between noise reduction and capturing character variations. KNN has the ability to handle multi-class classification, which aligns with this study of the whole English alphabet. It determines the character's label based on the majority class among its nearest neighbors. With the KNN classifier in action, the experiments and results will be explored in the following section, shedding light on the efficacy of the character recognition methodology. Figure 6 summarizes the proposed methodology for handwritten character recognition.

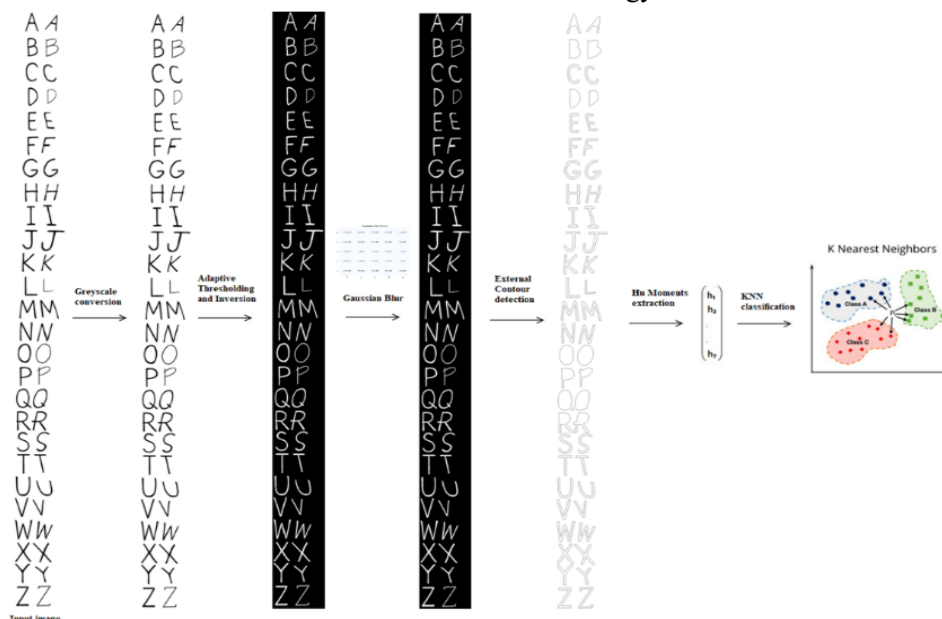


Figure 6. Proposed methodology for handwritten character recognition

5. Results and Discussion

The accuracy of character recognition using the proposed methodology is evaluated using a confusion matrix (Figure 7), which depicts the trade-off between true and wrong predictions for each character. The diagonal elements in a confusion matrix represent the instances that were correctly classified for each class. Specifically, each element $C[i, i]$ on the diagonal corresponds to the number of instances of class i that were correctly predicted as class i . In other words, it shows the true positives for each class, indicating how many instances were accurately classified by the model for each class. Based on the confusion matrix, the total number of correct classifications is 43 for a total of 52 handwritten characters. The total number of characters that were misclassified is 9. The elements that are not on the diagonal of a confusion matrix represent instances that were misclassified. These off-diagonal elements can be categorized into two types of errors: False Positives (FP) and False Negatives (FN).

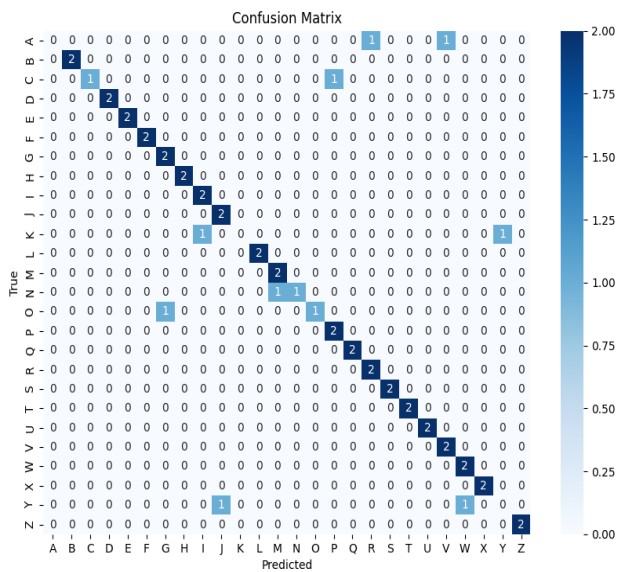


Figure 7. Confusion matrix of the classification

Figure 8 shows classifier precision, recall and f1-score on single letter predictions and on average for the integral dataset.

Classification Report:				
	precision	recall	f1-score	support
A	0.00	0.00	0.00	2
B	1.00	1.00	1.00	2
C	1.00	0.50	0.67	2
D	1.00	1.00	1.00	2
E	1.00	1.00	1.00	2
F	1.00	1.00	1.00	2
G	0.67	1.00	0.80	2
H	1.00	1.00	1.00	2
I	0.67	1.00	0.80	2
J	0.67	1.00	0.80	2
K	0.00	0.00	0.00	2
L	1.00	1.00	1.00	2
M	0.67	1.00	0.80	2
N	1.00	0.50	0.67	2
O	1.00	0.50	0.67	2
P	0.67	1.00	0.80	2
Q	1.00	1.00	1.00	2
R	0.67	1.00	0.80	2
S	1.00	1.00	1.00	2
T	1.00	1.00	1.00	2
U	1.00	1.00	1.00	2
V	0.67	1.00	0.80	2
W	0.67	1.00	0.80	2
X	1.00	1.00	1.00	2
Y	0.00	0.00	0.00	2
Z	1.00	1.00	1.00	2
accuracy			0.83	52
macro avg	0.78	0.83	0.78	52
weighted avg	0.78	0.83	0.78	52

Figure 8. Precision, recall and f1-score for the 26 classes of handwritten characters

Precision is a measure of the accuracy of the positive predictions made by a classifier. It is the ratio of true positive predictions to the total number of positive predictions made by the model. Precision is calculated using Eq. (11):

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (11)$$

Precision provides insights into the model's ability to avoid falsely labeling negative instances as positive. Recall (Sensitivity or True Positive Rate) is a measure of the ability of a classifier to capture all the relevant instances of a particular class. It is the ratio of true positive predictions to the total number of actual positive instances. Recall is calculated using Eq. (12):

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (12)$$

Recall is particularly important in scenarios where the cost of missing positive instances is high. The F1-Score is the harmonic mean of precision and recall. It provides a balanced measure that takes both false positives and false negatives into account. F1-Score is calculated using Eq. (13):

$$F1score = \frac{2x(Precision \times Recall)}{Precision + Recall} \quad (13)$$

F1-Score is useful when there is an uneven class distribution (imbalanced classes) and provides a single metric that combines precision and recall.

Based on the obtained class-wise metrics shown in Figure 8, the characters B, D, E, F, H, L, Q, S, T, U, X, Z exhibit perfect precision, recall, and F1-score of 1.00. The model accurately identifies and classifies instances for these classes without any false positives or negatives. The model excels in these classes, demonstrating a robust ability to correctly classify instances and maintain a perfect balance between precision and recall. Characters C, G, I, J, M, N, O, P, R, V, W show reasonable performance with precision and recall values ranging from 0.67 to 1.00. The characters A, K, Y have precision, recall, and F1-equal to 0.00, indicating poor performance. The model struggles to correctly classify instances for these classes. The accuracy and error rate for the whole dataset are shown on Figure 9. The developed hybrid model predicted the correct label of the characters in 82.69% of the cases.

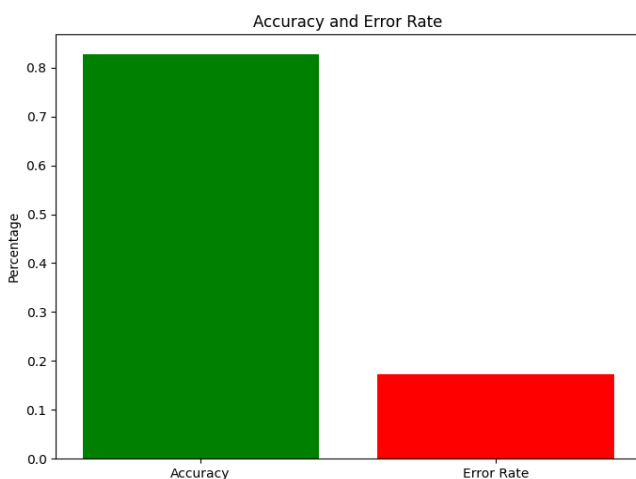


Figure 9. Accuracy and error rate of the classification

6. Conclusion

In the presented research, a variety of preprocessing techniques were employed and the seven Hu moments were utilized to characterize handwritten characters. This resulted in representing each character as a unique vector consisting of seven real numbers, corresponding to the Hu moments. It is assumed that characters of the same letter, even rendered in alternative forms, are described with Hu vectors at minimum Manhattan distance. The Hu vector at minimum distance is identified by the KNN algorithm, which actually performs the task of character recognition. The results of this approach yielded a high accuracy rate of 82.69%. This is a pioneering work in the field of pattern recognition that applies for the first time the Hu moments with KNN for handwritten character recognition.

References:

- [1]. Ahmed, S. S., Mehmood, Z., Awan, I. A., & Yousaf, R. M. (2023). A novel technique for handwritten digit recognition using deep learning. *Journal of Sensors*, 2023(1), 2753941.
- [2]. Akinbade, D., Ogunde, A. O., Odim, M. O., & Oguntunde, B. O. (2020). An adaptive thresholding algorithm-based optical character recognition system for information extraction in complex images. *Journal of Computer Science*, 16(6), 784-801.
- [3]. Anandhalli, M., Tanuja, A., & Baligar, P. (2022). Geometric invariant features for the detection and analysis of vehicle. *Multimedia tools and applications*, 81(23), 33549-33567.
- [4]. Bradley, D., & Roth, G. (2007). Adaptive thresholding using the integral image. *Journal of graphics tools*, 12(2), 13-21.
- [5]. Chen, W., & Lu, J. (2017). Human Shape Recognition Algorithm Design Based on Hu Moments and Zernike Moments. In *2016 4th International Conference on Machinery, Materials and Information Technology Applications*, 858-864. Atlantis Press.
- [6]. Cheng, C., Zhang, X. Y., Shao, X. H., & Zhou, X. D. (2016). Handwritten Chinese character recognition by joint classification and similarity ranking. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 507-511. IEEE.
- [7]. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1), 21-27.
- [8]. Dino, H. I., & Abdulrazzaq, M. B. (2019). Facial expression classification based on SVM, KNN and MLP classifiers. In *2019 International Conference on Advanced Science and Engineering (ICOASE)*, 70-75. IEEE.
- [9]. Fernando, M., & Wijayanayake, J. (2015). Novel approach to use HU moments with image processing techniques for real-time sign language communication. *International Journal of Image Processing (IJIP)*, 9(6).
- [10]. Gedraite, E. S., & Hadad, M. (2011). Investigation on the effect of a Gaussian Blur in image filtering and segmentation. In *Proceedings ELMAR-2011*, 393-396. IEEE.
- [11]. Golubitsky, O., & Watt, S. M. (2010). Distance-based classification of handwritten symbols. *International Journal on Document Analysis and Recognition (IJDAR)*, 13(2), 133-146.
- [12]. Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, 2, 1-758. New York: springer.
- [13]. Hu, M. K. (1962). Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2), 179-187.
- [14]. Hummel, R. A., Kimia, B., & Zucker, S. W. (1987). Deblurring gaussian blur. *Computer Vision, Graphics, and Image Processing*, 38(1), 66-80.

- [15]. Jusman, Y., Anam, M. K., Puspita, S., & Saleh, E. (2021). Machine learnings of dental caries images based on hu moment invariants features. In *2021 International Seminar on Application for Technology of Information and Communication (iSemantic)*, 296-299. IEEE.
- [16]. Khobragade, R. N., Koli, N. A., & Lanjewar, V. T. (2020). Challenges in recognition of online and off-line compound handwritten characters: a review. *Smart Trends in Computing and Communications: Proceedings of SmartCom 2019*, 375-383.
- [17]. Kour, G., & Saabne, R. (2014). Fast classification of handwritten on-line arabic characters. In *2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, 312-318. IEEE.
- [18]. Milgram, D. L., Rosenfeld, A., Willet, T., & Tisdale, G. (1978). Algorithms and hardware technology for image recognition. *Final Report to US Army Night Vision Laboratory*.
- [19]. Mithe, R., Indalkar, S., & Divekar, N. (2013). Optical character recognition. *International journal of recent technology and engineering (IJRTE)*, 2(1), 72-75.
- [20]. Mohamad, M. A., Hassan, H., Nasien, D., & Haron, H. (2015). A review on feature extraction and feature selection for handwritten character recognition. *International Journal of Advanced Computer Science and Applications*, 6(2).
- [21]. Otiniano-Rodríguez, K. C., Cámara-Chávez, G., & Menotti, D. (2012, November). Hu and Zernike moments for sign language recognition. In *Proceedings of international conference on image processing, computer vision, and pattern recognition*, 1-5.
- [22]. Priya, A., Mishra, S., Raj, S., Mandal, S., & Datta, S. (2016). Online and offline character recognition: A survey. In *2016 International conference on communication and signal processing (ICCSP)*, 967-970. IEEE.
- [23]. Sampath, A., Tripti, C., & Govindaru, V. (2012). Freeman code based online handwritten character recognition for Malayalam using backpropagation neural networks. *Advanced Computing*, 3(4), 51.
- [24]. Stojanov, D., & Koceski, S. (2014). Topological MRI prostate segmentation method. In *2014 Federated Conference on Computer Science and Information Systems*, 219-225. IEEE.
- [25]. Vinotheni, C., & Pandian, S. L. (2023). End-to-end deep-learning-based tamil handwritten document recognition and classification model. *IEEE Access*, 11, 43195-43204.
- [26]. Wan, K. W., Wong, C. H., Ip, H. F., Fan, D., Yuen, P. L., Fong, H. Y., & Ying, M. (2021). Evaluation of the performance of traditional machine learning algorithms, convolutional neural network and AutoML Vision in ultrasound breast lesions classification: A comparative study. *Quantitative imaging in medicine and surgery*, 11(4), 1381.
- [27]. Wong, T. T. (2015). Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern recognition*, 48(9), 2839-2846.
- [28]. Yan, F., Zhang, H., & Kube, C. R. (2005). A multistage adaptive thresholding method. *Pattern recognition letters*, 26(8), 1183-1191.
- [29]. Zekovich, S., & Tuba, M. (2013). Hu moments based handwritten digits recognition algorithm. *Recent Advances in Knowledge Engineering and Systems Science*, 98-103.
- [30]. Zhang, R., Wang, Q., & Lu, Y. (2017, November). Combination of ResNet and center loss based metric learning for handwritten Chinese character recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 5, 25-29. IEEE.