Journal of Theoretical and Applied Information Technology

31st August 2023. Vol.101. No 16
© 2023 Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

GRASSHOPPER OPTIMIZATION-BASED NEUTROSOPHICAL FUZZY CONVOLUTIONAL NEURAL NETWORK FOR ENHANCED MOVING OBJECT DETECTION

S SARAVANAKUMAR [1], M. LINGARAJ [2]

[1] Research Scholar, Department of Research and Development, Bharathiar University, Coimbatore, India
[2] Associate Professor, Department of Computer Science, Sankara College of Science and Commerce, Coimbatore, India
Email-id :[1]ssk.saravanakumar@gmail.com, [2]maillinga123@gmail.com

ABSTRACT

Detecting moving objects is a cornerstone of computer vision research and has many practical uses in security, robotics, video analysis, and virtual reality. This paper presents a novel approach, the Grasshopper Optimization based Neutrosophical Fuzzy Convolutional Neural Network (NFCNN), for enhanced moving object detection. The proposed approach integrates the Grasshopper Optimization Algorithm (GOA), neutrosophic principles, and fuzzy logic into a Convolutional Neural Network (CNN) architecture to improve moving object detection accuracy, robustness, and efficiency. The GOA is employed to optimize the parameters of the NFCNN, enabling adaptive learning and feature extraction from input data. Neutrosophic principles are integrated into the NFCNN to handle uncertain and imprecise information, capturing the nuances and contradictions in moving object detection. Fuzzy logic is incorporated to manage the imprecision and uncertainties inherent in object detection tasks. The proposed GOA-NFCNN is evaluated on benchmark datasets, and existing practices are compared to the outcomes. The experimental results demonstrate the superiority of the Grasshopper Optimization-based Neutrosophical Fuzzy Convolutional Neural Network's accuracy, robustness, and computational efficiency. Integrating GOA, neutrosophic principles, and fuzzy logic in the NFCNN yields significant improvements in moving object detection. The proposed approach enhances the ability to handle complex motion patterns, occlusions, and variations in lighting conditions, resulting in more accurate and reliable object detection in dynamic environments.

Keywords: *Moving Object Detection, Grasshopper Optimization Algorithm, Neutrosophical Fuzzy Logic, Convolutional Neural Network (CNN), Enhanced Object Detection, Computer Vision*

1. INTRODUCTION

Detecting objects in motion is a critical component of computer vision, involving identifying and tracking objects in motion within a given scene or video. It finds wide application in surveillance systems, autonomous vehicles, video analysis, and augmented reality [1]. This section delves into the concept of moving object detection, exploring its techniques and highlighting its significance in computer vision. The primary goal of moving object detection algorithms is to differentiate between the static background and the moving objects present in a video or image sequence. They aim to extract pertinent information about the moving objects, including their precise location, shape, size, and trajectory, while effectively filtering out the static background clutter [2]. This is typically accomplished through motion analysis, object tracking, and background modeling[3].

There are several advantages to using moving object detection techniques:

- Enhanced Surveillance: Moving object detection is a fundamental component of surveillance systems. By accurately detecting and tracking moving objects, it enables the identification of potential security threats, such as intruders or suspicious activities, in real-time [4]. This capability enhances the effectiveness of security measures and allows for prompt intervention when needed.
- Traffic Monitoring: Moving object detection is crucial for traffic monitoring systems in urban areas and highways. By identifying and tracking vehicles and pedestrians, it enables

traffic flow analysis, congestion detection, and the implementation of intelligent transportation systems. This data is invaluable to controlling traffic better, making roads safer, and increasing transportation efficiency [5].

- Autonomous Vehicles: Moving object detection plays a vital role in developing autonomous driving systems. To make educated judgments and guarantee safe navigation, it is essential to accurately identify and track things in the vehicle's surroundings, such as other cars, people, and barriers. This capability is of utmost importance in ensuring the reliability and safety of autonomous driving technology[6].
- Video Analysis: Moving object detection finds extensive application in video analysis tasks such as action recognition, behaviour understanding, and video summarization. By identifying and tracking moving objects, meaningful information can be extracted from video sequences, enabling higher-level analysis and interpretation of visual content. This enables researchers and analysts to gain valuable insights from video data and extract relevant information for various applications. Despite its numerous advantages, moving object detection also presents several challenges [7, 8]:
- Variability in Motion Patterns: Objects in motion can exhibit a wide range of motion patterns, including different speeds, directions, and types of motion. The challenge lies in developing algorithms that can accurately detect and track objects with varying motion patterns while handling occlusions, scale changes, and complex interactions between multiple objects.
- Real-Time Performance: Many applications of moving object detection, such as surveillance systems and autonomous vehicles, require real-time performance to ensure timely and responsive actions.
- Illumination Changes and Dynamic Backgrounds: The performance of moving object identification algorithms is often negatively impacted by lighting and backdrop changes. Shadows, lighting variations, and moving background objects can introduce noise and false positives, making it challenging to separate foreground objects from the background accurately.
- Computational Efficiency: Algorithms for detecting moving objects must instantly process massive volumes of visual input, which requires efficient computational techniques. Developing algorithms that balance accuracy and computational efficiency is crucial for practical implementation in resource-constrained environments.

The ability to recognize and follow moving objects is a critical application of computer vision technology. It has many applications, including surveillance systems, autonomous vehicles, video analysis, and augmented reality. Traditional approaches to moving object detection often face challenges in handling complex motion patterns, occlusions, and variations in lighting conditions. Researchers have been exploring advanced techniques and algorithms to overcome these challenges and enhance the accuracy and robustness of moving object detection. Bio-inspired Optimization [9–22], [34],[35] has several potential to solve various research issues.

Neutrosophics, Convolutional Neural Networks (CNNs), and the Grasshopper Optimization Algorithm (GOA) are three concepts that can contribute to improving moving object detection. Neutrosophics can handle uncertainties and imprecise information inherent in moving object detection [23]. Neutrosophic logic allows for representing and processing uncertain and contradictory data, enabling a more comprehensive understanding of the complexities involved in the detection process. Regarding computer vision, convolutional neural networks (CNNs) have been a game-changer, particularly in image analysis tasks. With their hierarchical structure and weight sharing, CNNs learn and extract meaningful features from images [24]. They have shown remarkable performance in object detection, making them well-suited for moving object detection. The Grasshopper Optimization Algorithm (GOA) is a nature-inspired optimization algorithm that emulates the collective behavior of grasshoppers. This algorithm has shown promise in optimizing the parameters of moving object detection algorithms, enabling them to effectively address challenges commonly encountered in real-world scenarios, including occlusions and lighting variations. The GOA's ability to explore the search space efficiently and balance exploration and exploitation makes it a promising approach for enhancing moving object detection [25].

1.1. Problem Statement

The goal of this study is to improve upon current techniques of moving object recognition by designing new algorithms that are both more efficient and more accurate, efficiency, and robustness of object detection in dynamic environments. The motivation for this research stems from the need to unlock the full potential of computer vision systems in various applications, including surveillance, autonomous systems, video analysis, and augmented reality. The specific problem is to address the challenges current moving object detection algorithms face, such as difficulties in accurately identifying and tracking objects with complex motion patterns, occlusions, and variations in lighting conditions. The goal is to develop novel techniques and methodologies to effectively handle these challenges and provide more reliable and precise object detection results in real-time scenarios.

Furthermore, the problem also involves addressing the computational complexity and resource limitations associated with moving object detection. Efficient algorithms and architectures must be designed to process large amounts of visual data in real-time while minimizing computational overhead and optimizing resource utilization. By addressing these challenges, the research aims to advance the field of moving object detection and enable the development of more accurate, efficient, and robust algorithms. The ultimate objective is to enhance safety, security, and efficiency in various domains, such as surveillance systems, autonomous vehicles, video analysis, and augmented reality applications.

1.2. Motivation

Moving object detection is critical in computer vision with significant real-world applications. Accurately identifying and tracking objects in dynamic environments are essential for various fields, including surveillance, autonomous systems, video analysis, and augmented reality. However, existing moving object detection algorithms often face challenges that limit their performance and applicability in practical scenarios. The motivation for this research stems from the need to overcome the limitations of current methods and improve the accuracy, efficiency, and robustness of moving object detection. By addressing these challenges, we can unlock the potential of computer vision systems in numerous domains, leading to enhanced safety, security, and efficiency in various applications. Improved moving object detection algorithms can have a transformative impact on surveillance systems. They can enable more effective threat detection, prompt intervention in security-sensitive areas, and proactive monitoring of public spaces. This can significantly enhance public safety and provide invaluable support to law enforcement agencies.

1.3. Research Objective

The objectives of this research are as follows:

- Develop a Grasshopper Optimization Algorithm for optimizing the parameters of the Neutrosophical Fuzzy Convolutional Neural Network.
- Design and implement a Neutrosophical Fuzzy Convolutional Neural Network architecture to handle uncertainties and imprecise information in moving object detection.
- Measure the method's performance on standard datasets, and see how it stacks up against other popular solutions.
- Conduct an extensive analysis of the results to demonstrate the advantages and effectiveness of the proposed Grasshopper Optimization-based Neutrosophical Fuzzy Convolutional Neural Network in enhancing moving object detection.

1.4. Organization of the Paper

The structure of the remaining paper is as follows: Section 2 reviews related works in moving object detection, optimization algorithms, and fuzzy logic, identifying gaps and limitations in the existing literature. Section 3 presents the proposed methodology, explaining the Grasshopper Optimization Algorithm (GOA) and its application in optimizing the Neutrosophical Fuzzy Convolutional Neural Network (NFCNN) parameters for enhanced moving object detection. Section 4 details the dataset used in the study, including its source and characteristics. Metrics for gauging how well the suggested technique works are introduced in Section 5. Compared to standard practices, experimental findings achieved using the suggested methodology are presented and analyzed in Section 6. Section 7 concludes the paper, summarizing the essential findings and contributions, discussing the significance of the proposed methodology, and suggesting future research directions.

2. LITERATURE REVIEW

"Fusion Representation Learning (FRL)" [26] combines spatial, temporal, and appearance-based representations so that more foreground motion may be seen. It involves extracting features from the input data and fusing them using feature-level or decision-level fusion techniques. The approach aims to capture both spatial and temporal cues and the appearance characteristics of moving objects. Integrating these different representations enhances the accuracy and robustness of moving object detection, especially in challenging scenarios with occlusions and complex motion patterns. "Novel RPCA with Nonconvex Logarithm and Truncated Fraction" [27] introduces a new method that improves the accuracy of moving object detection. The approach effectively separates foreground moving objects from complex backgrounds using Robust Principal Component Analysis (RPCA) with nonconvex logarithms and truncated fraction norms. These innovative techniques enhance the detection process's robustness by modelling the dynamic nature of moving objects and handling noise and outliers. Experimental evaluations demonstrate the approach's effectiveness in accurately identifying and separating moving objects in complex scenes.

The "Deep-Learning-Assisted Versatile Electret Sensor" [28] approach combines deep-learning techniques with an electret sensor to improve the detection of moving objects. By integrating deep learning algorithms with sensor technology, the approach enhances the accuracy and versatility of object detection in different scenarios. The method addresses the challenges associated with accurately detecting and classifying moving objects. Electret, an electrostatic sensor, is utilized to capture relevant data, while deep learning algorithms are employed to analyze and classify the sensor data. The proposed approach demonstrates promising results in accurately detecting and identifying moving objects in various environments. "Learning Dynamic Background" [29] enhances moving object detection by learning dynamic backgrounds using weak supervision. Training a deep learning model, it adapts to identify dynamic elements in complex backgrounds, improving the accuracy of detecting and tracking moving objects. The approach utilizes techniques such as background differencing and foreground saliency analysis to identify regions deviating from the learned dynamic background. By incorporating weak supervision, the model progressively improves its ability to distinguish moving objects from the dynamic background. Experimental evaluations validate the effectiveness of this approach, making it a valuable solution for moving object detection tasks where fully annotated training data is limited or unavailable. The method significantly enhances the robustness and accuracy of moving object detection systems.

"Full-Spectrum Light Sources" [30] aim for better object-detecting precision algorithms in challenging weather scenarios. This approach enhances the visibility of moving objects in conditions such as rain, fog, and low-light environments by utilizing full-spectrum light sources that emit light across a wide range of wavelengths. The goal is to develop robust algorithms to detect and track objects even when visibility is compromised. By leveraging the advantages of full-spectrum light sources, this approach addresses the challenges of adverse weather conditions, enabling more accurate and reliable moving object detection. "Atanassov's Intuitionistic 3D Fuzzy Histon Roughness Index" [31] combines the advantages of both methods to improve the accuracy and reliability of moving object detection. By utilizing Atanassov's Intuitionistic 3D Fuzzy Histogram Roughness Index, which captures the roughness of the image histogram along with texture features, the approach effectively detects and distinguishes moving objects from the background. This fusion-based approach enhances the robustness of moving object detection, enabling accurate and reliable results in various scenarios.

"2D LiDAR-based object detection" [32] utilizes a 2D LiDAR sensor to detect and track objects in the vehicle's vicinity. This approach enhances the vehicle's perception capabilities by leveraging the laser-based technology of LiDAR to create a detailed and accurate representation of the surrounding environment. The 2D LiDAR sensor scans the surrounding area, measuring distances to objects and generating a point cloud. Object detection algorithms identify and classify objects such as pedestrians, vehicles, and obstacles by analyzing the point cloud data. The tracking component then continuously tracks the detected objects over time, enabling the vehicle to maintain situational awareness and make informed decisions. "Simultaneous Denoising (SD)" [33] combines denoising and object detection algorithms to improve the accuracy of detecting moving objects in the presence of noise. The method uses a low-rank approximation to differentiate between a scene's background and foreground elements. Using a low-rank approximation, the approach approximates a low-rank matrix as the scene's backdrop. This low-

rank approximation helps remove the noise and preserve the static elements in the scene. The remaining residual matrix after low-rank approximation is then analyzed to identify and classify the moving objects.

3. GRASSHOPPEROPTIMIZATION - BASED NEUTROSOPHICALFUZZYCONVOLUTIONAL NEURAL NETWORK

3.1. Moving Object Detection

Moving object detection is a fundamental task in computer vision, playing a pivotal role in various real-world applications. It encompasses the challenging process of identifying and tracking objects within dynamic scenes captured by imaging devices. Computer vision systems can enable surveillance, autonomous navigation, video analysis, and activity recognition by accurately detecting and tracking moving objects. Moving object detection aims to differentiate foreground objects from the background, which may contain stationary elements or other irrelevant information. Several elements, such as shifting illumination, occlusions, distracting backdrops, and rapid object motion, make this a complex process. Effective moving object detection algorithms should be able to handle these complexities and uncertainties.

The process of moving object detection involves several steps. Initially, a background model is created to represent the stationary elements in the scene. The current frame is then compared to the backdrop model to pick out any pixels or areas that are drastically different. These deviations are considered potential foreground objects. Subsequently, post-processing techniques such as noise removal and object refinement are applied to improve the accuracy of the detection results. Finally, object-tracking algorithms can track the identified objects across subsequent frames. Accurate moving object detection is crucial for various applications. Surveillance systems enable the detection of intruders or suspicious activities in monitored areas. In autonomous vehicles, moving object detection is essential for collision avoidance and pedestrian detection. In video analysis, it assists in recognizing and tracking objects of interest, facilitating tasks such as action recognition and behaviour analysis.

3.2. Neutrosophical Fuzzy Logic

Neutrosophical Fuzzy Logic is a mathematical framework that combines neutrosophy and fuzzy logic to handle indeterminacy, uncertainty, and vagueness in decision-making and reasoning processes. It provides a comprehensive approach to dealing with real-world problems' complexities and ambiguities. Neutrosophy, introduced by FlorentinSmarandache, deals with the study of indeterminacy and considers the existence of three components: truth, falsity, and indeterminacy. It recognizes that many real-world situations and phenomena possess inherent uncertainties, where elements can simultaneously have degrees of truth, falsity, and indeterminacy. On the other hand, fuzzy logic is a mathematical approach that allows for the representation and manipulation of incomplete or imprecise information. It is based on the concept of fuzzy sets, which assign membership degrees to elements rather than strict binary classifications.

Neutrosophical Fuzzy Logic offers a more flexible and expressive decision-making framework by integrating neutrosophy and fuzzy logic. It allows for representing and handling uncertainties, vagueness, and contradictions that often arise in complex real-world problems. In Neutrosophical Fuzzy Logic, membership functions assign membership degrees to elements based on their truth, falsity, and indeterminacy degrees. These membership degrees are combined using neutrosophic fuzzy operators such as conjunction, disjunction, and implication to perform reasoning and make informed decisions. Neutrosophical Fuzzy Logic finds applications in various fields, including artificial intelligence, pattern recognition, image processing, decision support systems, and control systems. It enables the modelling and analysis of problems that involve imprecise or uncertain information, allowing for more robust and adaptive solutions.

3.2.1. Neutrosophical fuzzy sets for moving object detection

In the context of moving object detection, a digital image *"I"* can be represented as a matrix of pixels. Each pixel, denoted as $p(i,j)$, corresponds to a specific location in the image. To handle uncertainties and indeterminacies associated with each pixel, we can associate a neutrosophical fuzzy set $A(i,j)$ with it. The neutrosophical fuzzy set $A(i,j)$ can be defined as Eq.(1).

$$\boldsymbol{A(i,j)} = \{\boldsymbol{(x, \mu T(i,j,x), \mu F(i,j,x), \mu I(i,j,x))} | \boldsymbol{x \in X}\} \quad (1)$$

Here, i and j represent the pixel coordinates, X is the set of possible pixel values, and

$(x, \mu T(i,j,x), \mu F(i,j,x), \mu I(i,j,x))$ denotes an element within the neutrosophical fuzzy set $A(i,j)$. Each element in the set consists of a specific pixel value x, and its associated membership degrees of truth (μT), falsity (μF), and indeterminacy (μI).

The membership degrees $\mu T(i,j,x), \mu F(i,j,x)$, and $\mu I(i,j,x)$ quantify the degree to which the pixel $p(i,j)$ belongs to the truth, falsity, and indeterminacy components, respectively. These membership degrees are determined based on specific criteria or measures that assess the pixel's characteristics or properties, such as intensity, colour, texture, or motion. By representing pixels as neutrosophical fuzzy sets, we can handle the inherent uncertainties and indeterminacies in moving object detection. This mathematical framework enables the fusion of different sources of information, adaptive thresholding, informed decision-making, and robust object tracking, leading to more accurate and reliable results in detecting and tracking moving objects within dynamic scenes.

3.2.2. Membership Functions for Moving Object Detection

Membership functions are crucial in Neutrosophical Fuzzy Logic as they determine the membership degrees associated with pixel characteristics. In moving object detection, these membership functions can incorporate motion information and evaluate the degree to which a pixel belongs to different components: truth, falsity, and indeterminacy.

Membership degree of x in the truth component:

$$\boldsymbol{\mu T(i,j,x) = f_{motion}(x)} \quad (2)$$

In Eq.(2), the function $f_{motion}(x)$ represents a measure of motion for the pixel value x. It quantifies the degree to which the pixel value is associated with motion. Implementing $f_{motion}(x)$ can vary depending on the chosen motion detection algorithm or technique.

Membership degree of x in the falsity component:

$$\boldsymbol{\mu F(i,j,x) = 1 - \mu T(i,j,x)} \quad (3)$$

In Eq.(3), the membership degree in the falsity component is obtained as the complement of the membership degree in the truth component. A higher value of $\mu F(i,j,x)$ indicates a stronger association with the falsity component, suggesting a lack of motion.

Membership degree of x in the indeterminacy component:

$$\boldsymbol{\mu I(i,j,x) = 1 - \mu T(i,j,x) - \mu F(i,j,x)} \quad (4)$$

In Eq.(4), the membership degree in the indeterminacy component represents the remaining degree of membership not accounted for by the truth or falsity components. It quantifies the uncertainty or ambiguity associated with the pixel value x.

Moving object detection decision rule:

$$\boldsymbol{\mu T(i,j,x) > \mu F(i,j,x)} \quad (5)$$

$$\boldsymbol{\mu T(i,j,x) > \mu I(i,j,x)}: \boldsymbol{Pixel\ p(i,j)} \quad (6)$$

If Eq.(5) and Eq.(6) get true, it is classified as a moving object. This decision rule determines whether a pixel is classified as a moving object based on the relative values of its membership degrees in the truth, falsity, and indeterminacy components. If the membership degree in the truth component is higher than those in the falsity and indeterminacy components, the pixel is classified as a moving object.

3.2.3. Neutrosophical Fuzzy Fusion for Moving Object Detection

Fusion operators are essential in Neutrosophical Fuzzy Logic as they enable the combination of information from multiple sources. Neutrosophical Fuzzy Logic offers fusion operators such as neutrosophic fuzzy conjunction (NFC) and neutrosophic fuzzy disjunction (NFD), which allow the merging of membership degrees obtained from different neutrosophical fuzzy sets.

For two given neutrosophical fuzzy sets $A(i,j)$ and $B(i,j)$ associated with a pixel $p(i,j)$, the NFC operator can be defined as Eq.(7).

$$\boldsymbol{NFC(A(i,j), B(i,j)) = \left\{ \begin{array}{c} \left(x, min(\mu T_{A(i,j,x)}, \mu T_{B(i,j,x)})\right), \\ \left(min(\mu F_{A(i,j,x)} \mu F_{B(i,j,x)})\right), \\ \left(min(\mu I_{A(i,j,x)}, \mu I_{B(i,j,x)})\right) \end{array} \right\}} \quad (7)$$

In Eq.(7), $(x, \mu T_{A(i,j,x)}, \mu F_{A(i,j,x)}, \mu I_{A(i,j,x)})$ represents an element of the neutrosophical fuzzy set $A(i,j)$, while $(x, \mu T_{B(i,j,x)}, \mu F_{B(i,j,x)}, \mu I_{B(i,j,x)})$ represents an element of the neutrosophical fuzzy set $B(i,j)$. The NFC operator combines these two sets by taking the minimum membership degrees for

each component (truth, falsity, and indeterminacy) at each pixel value x.

Similarly, the NFD operator can be defined as Eq.(8):

$$NFD(A(i,j),B(i,j)) = \begin{Bmatrix} \left(x, max\left(\mu T_{A(i,j,x)}, \mu T_{B(i,j,x)}\right)\right), \\ \left(max\left(\mu F_{A(i,j,x)} \mu F_{B(i,j,x)}\right)\right), \\ \left(max\left(\mu I_{A(i,j,x),} \mu I_{B(i,j,x)}\right)\right) \end{Bmatrix} \quad (8)$$

The NFD operator combines the membership degrees by taking the maximum values for each component (truth, falsity, and indeterminacy) at each pixel value x. Utilizing these fusion operators allows the membership degrees of different neutrosophical fuzzy sets to be effectively combined, facilitating the integration of information from diverse sources. The min-max operations employed in these operators ensure that the resulting fusion retains the characteristics of the original sets, enabling the representation of uncertainties and indeterminacies more comprehensively.

3.2.4. Neutrosophical Fuzzy Decision-Making for Moving Object Detection

Decision rules in moving object detection can be formulated based on membership degrees obtained from different sources, utilizing logical operators such as neutrosophic fuzzy conjunction (NFC) and neutrosophic fuzzy disjunction (NFD). By applying these operators, we can construct rules to determine the membership of a pixel in a moving object. Let's consider sets $A(i,j)$ and $B(i,j)$ representing different information sources associated with a pixel $p(i,j)$. The criterion for making a call might be stated as follows:

If NFC$(A(i,j),B(i,j))$ is greater than or equal to the neutrosophical fuzzy threshold $T(i,j)$, then pixel $p(i,j)$ belongs to the moving object. Here, NFC$(A(i,j),B(i,j))$ represents the fusion of sets $A(i,j)$ and $B(i,j)$ using the NFC operator. The NFC operator combines the membership degrees of truth, falsity, and indeterminacy obtained from the two sets, resulting in a new neutrosophical fuzzy set representing the degree of agreement between the information sources regarding the pixel's association with the moving object.

The neutrosophical fuzzy threshold $T(i,j)$ is obtained through the fusion of threshold values associated with sets $A(i,j)$ and $B(i,j)$. It serves as a criterion to decide whether the pixel is considered part of the moving object based on the combined information. We can decide the pixel's membership in the moving object by comparing the NFC fusion result with the threshold value. If the NFC result is greater than or equal to the threshold, the pixel is classified as belonging to the moving object. These decision rules, constructed using fusion operators and thresholds, provide a framework to integrate and evaluate information from diverse sources. Neutrosophical Fuzzy Logic enables effective decision-making in moving object detection by incorporating uncertainties and indeterminacies, leading to more accurate and reliable results in complex scenarios.

3.2.5. Neutrosophical Fuzzy Tracking for Moving Object Detection

Tracking moving objects over time is a challenging task requiring considering the dynamic changes in their states. Neutrosophical Fuzzy Logic provides a robust framework to handle uncertainties and indeterminacies inherent in the tracking process. By representing the position and velocity of the object as neutrosophical fuzzy sets, tracking algorithms can effectively update the object's state using fusion operators such as neutrosophic fuzzy conjunction (NFC) and neutrosophic fuzzy disjunction (NFD) at each time step.

In a tracking scenario, the position and velocity of the object can be described using neutrosophical fuzzy sets. Let's denote the position set as $P(i,j)$ and the velocity set as $V(i,j)$, where i and j represent the object's coordinates. These sets can be defined as Eq.(9) and Eq.(10):

$$P(i,j) = \{\{(x, \mu T_{P(i,j,x)}, \mu F_{P(i,j,x),} \mu I_{P(i,j,x)})\}\} \quad (9)$$

$$V(i,j) = \{(v, \mu T_{V(i,j,x)}, \mu F_{V(i,j,x),} \mu I_{V(i,j,x)})\} \quad (10)$$

Here, x denotes the position value, v represents the velocity value, and $\mu T, \mu F$and μI denote the membership degrees in the truth, falsity, and indeterminacy components.

To track the object over time, the state update process involves fusing the information from the previous time step with the new measurements. This fusion can be achieved using the NFC and NFD operators. For instance, the fusion of position sets $P(i,j)$ and $P'(i,j)$ at time t can be expressed as Eq.(11):

$$NFC\big(P(i,j),P'(i,j)\big)=\left\{\begin{pmatrix}x,min\big(\mu T_{P(i,j,x)},\mu T_{P'(i,j,x)}\big),\\min\big(\mu F_{P(i,j,x)},\mu F_{P'(i,j,x)}\big),\\min\big(\mu I_{P(i,j,x)},\mu I_{P'(i,j,x)}\big)\end{pmatrix}\right\} \quad (11)$$

Similarly, the fusion of velocity sets $V(i,j)$ and $V'(i,j)$can be defined as Eq.(12):

$$NFC\big(V(i,j),V'(i,j)\big)=\left\{\begin{pmatrix}v,min\big(\mu T_{V(i,j,v)},\mu T_{V'(i,j,v)}\big),\\min\big(\mu F_{V(i,j,v)},\mu F_{V'(i,j,v)}\big),\\min\big(\mu I_{V(i,j,v)},\mu I_{V'(i,j,v)}\big)\end{pmatrix}\right\} \quad (12)$$

Utilizing these fusion operators, the tracking algorithm can update the object's position and velocity sets at each time step. These fused sets represent the updated state of the object, considering the uncertainties and indeterminacies associated with the measurements and the tracking process itself.

Algorithm 1: Neutrosophical Fuzzy Fusion for Moving Object Detection

Step 1: Initialize

Load the digital image I.

Define the set of possible pixel values X.

Set the threshold values for decision-making.

Step 2: For each pixel $p(i,j)$ in the image I:

a) Calculate the membership degrees for truth, falsity, and indeterminacy:

- Evaluate the degree of motion for the pixel value using a motion detection algorithm.
- Determine the percentage of truth-value membership for each pixel.
- Calculate the membership degree of the pixel value in the falsity component.
- By calculating their values, determine the percentage of pixels that belong to the indeterminacy component.

b) Determine if the pixel is classified as a moving object:

- Compare the membership degree in the truth component with those in the falsity and indeterminacy components.
- If the membership degree in the truth component is higher, classify the pixel as a moving object.

Step 3: Apply fusion operators for moving object detection

a) Combine membership degrees from different sources using fusion operators:

- For each pixel$p(i,j)$, combine the membership degrees from different neutrosophical fuzzy sets using the fusion operator (e.g., NFC or NFD).
- Take the minimum or maximum values for each component (truth, falsity, indeterminacy) at each pixel value.
- Generate a new neutrosophical fuzzy set representing the fused membership degrees.

Step 4: Perform decision-making for moving object detection

- Compare the fused membership degrees with the threshold values.
- If the fused membership degrees are greater than or equal to the thresholds, classify the pixel as part of the moving object.

Step 5: Repeat Steps 2 to 4 for each pixel in the image.

Step 6: Optional: Track moving objects over time using neutrosophical fuzzy tracking:

- Define neutrosophical fuzzy sets for the position and velocity of the object.
- Update the object's state at each time step by fusing information from previous time steps with new measurements.
- Use fusion operators (e.g., NFC) to combine the membership degrees of position and velocity sets.
- Obtain the updated position and velocity sets representing the object's state.

Step 7: Repeat the tracking process for subsequent time steps.

Step 8: **Output the results of moving object detection and tracking.**

3.3. Convolutional Neural Networks

3.3.1. Convolution Operation

The convolution operation in CNNs involves sliding a filter (known as a kernel) across the input image and then calculating the dot product of the filter weights with each input patch. Mathematically, the convolution operation for a specific output feature map position (i, j) and channel (c) can be defined as Eq.(13).

$$Y[i,j,c] = \Sigma\Sigma\Sigma X[a,b,d] * F[i-a, j-b, d, c] \quad (13)$$

$Y[i,j,c]$ represents the output value at position (i, j) and channel c in the feature map. $X[a,b,d]$ represents the input value at position (a, b) and channel d in the input image.$F[i-a, j-b, d, c]$ represents the filter weight at the position $(i-a, j-b)$, input channel d, and output channel c.

The summation is performed over all possible positions (a, b) and channel d, which are determined by the size of the filter and the number of input channels. We compute the output feature map Y by sliding the filter over the entire input image and applying this equation.

3.3.2. Activation Function

Using an activation function, the network may describe intricate interactions by making the convolutional layer's output non-linear. In convolutional neural networks (CNNs), the Rectified Linear Unit (ReLU) activation function is defined in Eq.(14).

$$A = max(0, Z) \quad (14)$$

where A represents the activation output, and Z represents the input to the activation function. The ReLU function returns the maximum between 0 and the input value Z. If Z is positive, the output A will equal$Z$. If Z is negative, the output A will be 0. This function effectively "activates" the neuron when the input is positive and keeps it inactive (outputting 0) when the input is negative.

An element-by-element application of the ReLU activation function on a convolutional layer's output feature map, we introduce non-linearity and allow the network to learn complex patterns and representations.

3.3.3. Pooling Operation

Pooling is a downsampling operation that reduces the feature maps' spatial dimensions, helping extract the essential information while reducing computational complexity. Max pooling is a commonly used technique that selects the maximum value within each pooling window. Consider a specific output feature map position (i, j) and channel c. The pooling operation computes the maximum value within a pooling window of size $P\ x\ P$ in the input feature map, X which can be mathematically represented as Eq/(15).

$$Y[i,j,c] = max(X[Pi: P(i+1), Pj: P(j+1), c]) \quad (15)$$

$Y[i,j,c]$ denotes the output value at position (i, j) and channel c in the pooled feature map.$X[Pi: P(i+1), Pj: P(j+1), c]$ represents the input values within the pooling window located at position (i, j), and channel c.P is the pooling size, indicating the dimensions of the pooling window.

To create the output feature map, the max pooling operation picks the input value that is the highest inside the pooling window. We obtain the pooled feature map Y by sliding the pooling window over the input feature map.

3.3.4. Fully Connected Layers

After several convolutional and pooling layers, CNNs often include one or more fully connected layers to learn high-level representations. These layers connect every neuron to every neuron in the subsequent layer, allowing for complex mappings. Let's denote the output of the last pooling layer as a vector V. The fully connected layer can be represented mathematically as Eq.(16).

$$Y = W * V + B \quad (16)$$

where Y represents the ultimately linked layer's output vector, W denotes the weight matrix that defines relationships between neurons in one layer and those in the next. The dimensions of W are typically (M, N), where M is the number of neurons in the current layer, and N is the number of neurons in the subsequent layer.V is the input vector obtained by flattening what comes out of the final layer of pooling. Its dimensions are typically $(N, 1)$. B is the bias vector with dimensions $(M, 1)$ is added element-wise to the weighted sum.

The input vector V is multiplied by the weight matrix W, and then the result is added to the bias vector B element by element in the fully connected layer. The resultant vector Y is the ultimately linked layer's output.

3.3.5. Softmax Function

To transform raw scores into class probabilities, a softmax function is typically used to output the final fully connected layer in a classification job. The softmax function normalizes the output scores, ensuring that they sum up to 1 and can be interpreted as probabilities. Let's consider the input vector X with dimensions $(C, 1)$, where C is the number of classes. The softmax function can be mathematically defined as Eq.(17).

$$Y[i] = exp\big(X[i]/\Sigma(X[j])\big) \quad (17)$$

where $Y[i]$ represents the output probability for class i, and $X[i]$ is the raw score or logit for class i. The sum in the denominator is computed over all classes j. The softmax function exponentiates each raw score and divides it by the sum of all exponentiated scores, ensuring that the output probabilities are non-negative and sum up to 1. This allows us to interpret the output probability Y as the likelihood of the input belonging to each class.

3.3.6. Backpropagation and Optimization Algorithms

Backpropagation and optimization algorithms are crucial in training CNNs by updating the network's parameters to minimize the loss function. The input data is propagated through the network during the forward pass, and the output predictions are computed. Then, during the backward pass, the gradients of the loss concerning the network's parameters are calculated using the chain rule of calculus. These gradients represent the sensitivity of the loss to changes in the parameters and indicate the direction in which the parameters should be adjusted.

Optimization algorithms, such as gradient descent and its variants, utilize these gradients to update the network's parameters iteratively. The objective is to determine the best values for the parameters to maximize the model's prediction ability while minimizing the loss function. Gradient descent, a widely used optimization algorithm, adjusts the weights and biases in the network by taking steps proportional to the negative of the gradients. The magnitude of these steps is determined by the learning rate, which controls the convergence speed. Stochastic gradient descent (SGD), a variant of gradient descent, further enhances the optimization process by randomly selecting a subset of training samples, called a mini-batch, to compute the gradients and update the parameters. This introduces noise that helps escape local minima and can lead to faster convergence.

3.3.7. Regularization Techniques

Overfitting is avoided in CNNs because to regularization methods like $L1$ and $L2$. A dropout is a specific form of regularization that enhances the model's robustness.

L1 Regularization

$L1$ regularization introduces an additional term in the loss function that penalizes the absolute values of the weights. Let's consider a specific layer in a CNN with weights represented by the matrix W. The L1 regularization term can be written as Eq.(18).

$$L1 = \lambda * \Sigma|W| \quad (18)$$

where λ is the regularization parameter and $|W|$ represents the element-wise absolute values of the weights. The overall loss function with $L1$ regularization can be written as Eq.(19).

$$Loss = CrossEntropyLoss + L1 \quad (19)$$

During backpropagation, the gradients of the $L1$ regularization term concerning the weights are computed as Eq.(20).

$$\partial L1/\partial W = \lambda * sign(W) \quad (20)$$

These gradients are then added to the gradients of the cross-entropy loss during backpropagation to update the weights.

L2 Regularization

In $L2$ regularization, often called weight decay, a component is added to the loss function that penalizes squared weights. Let's consider a specific layer in a CNN with weights represented by the matrix W. The $L2$ regularization term can be written as Eq/(21).

$$L2 = \lambda * \Sigma W^2 \quad (21)$$

where λ is the regularization parameter, and W^2represents the element-wise squared values of the weights. The overall loss function with$L2$ regularization can be written as Eq.(22).

$$Loss = CrossEntropyLoss + L2 \quad (22)$$

During backpropagation, the gradients of the $L2$ regularization term concerning the weights are computed as Eq.(23).

$$\partial L2/\partial W = 2 * \lambda * W \quad (23)$$

These gradients are then added to the gradients of the cross-entropy loss during backpropagation to update the weights.

3.3.8. Dropout

A dropout is a regularization approach that sets a small percentage of activations in each training layer to zero at random. Let's consider a specific layer in a CNN with activations represented by matrix A. Dropout is applied by multiplying the activations with a binary mask matrix D, where each element of D is set to 0 or 1 with a certain probability. During forward propagation, the dropout mask is applied element-wise to the activations. The masked activations, A_{masked}, are computed asEq.(24).

$$A_{masked} = A * D \quad (24)$$

where A represents the initial activations, and the masked activations are then passed to the subsequent layers.

During backpropagation, the gradients from the subsequent layers are multiplied element-wise with the dropout mask, ensuring that only the active neurons receive gradients. Let $\partial L/\partial A_{masked}$ be the gradients concerning the masked activations. The gradients concerning the original activations, ∂L/∂A, are calculated as Eq.(25).

$$\partial L/\partial A = \partial L/\partial A_{masked} * D \quad (25)$$

where * represents element-wise multiplication.

Using regularization strategies like $L1$ and $L2$ regularization, CNNs can learn more robust and generalized representations while mitigating overfitting risks. The regularization terms affect the loss function and gradients during backpropagation, providing a penalty on the weights. Dropout introduces stochasticity by randomly dropping out activations, which helps prevent over-reliance on specific activations and encourages the network to learn more robust features. Together, these techniques contribute to improved generalization and model performance in CNNs.

Algorithm 2: CNN

Step 1: Initialize the network parameters, including the filter weights and biases, for each layer in the CNN.

Step 2: Perform a forward pass through the network:

a) Convolution operation: We implement the activation function by sliding the filters across the input moving objects from a video file and computing the dot product of the filter weights and the input patches.
b) Pooling operation: Picking maximum values inside pooling windows might help you reduce the feature maps' spatial dimensions.
c) To retrieve a vector from the last pooling layer's output, we must flatten it.
d) Fully connected layers: The input vector should be multiplied by the weight matrix, and then the bias vector should be added to it, element by element. Invoke the activation procedure.
e) Softmax function: Convert the raw scores into class probabilities.

Step 3: Compute the loss function, typically using cross-entropy loss, by comparing the predicted class probabilities with the accurate labels.

Step 4: Backpropagation may be used to compute the loss function gradients for the network parameters.

Step 5: Update the network parameters using an optimization algorithm:

a) Initialize the parameter update values.
b) Compute the gradients for the parameters.
c) Update the parameters by taking steps proportional to the negative gradients, scaled by the learning rate.
d) Repeat steps b and c for a specified number of iterations or until a convergence criterion is met.

Step 6: To avoid overfitting, use regularization methods like $L1$ or $L2$ regularization to add terms to the reduction function and tweak the gradients in backpropagation.

Step 7: Repeat steps 2 to 6 for a fixed number of epochs or until the desired level of accuracy is achieved.

Step 8: Evaluate the trained CNN on a separate test dataset to assess its performance.

Step 9: Adjust the hyperparameters, such as learning rate and regularization strength, based on the evaluation results and repeat steps 2 to 8 if necessary.

3.4. Grasshopper Optimization based Neutrosophical Fuzzy Logic CNN

The proposed Grasshopper Optimization-based Neutrosophical Fuzzy Logic CNN leverages a modified architecture that combines the strengths of traditional Convolutional Neural Networks (CNNs) with the optimization capabilities of the Grasshopper Optimization algorithm and the flexibility of Neutrosophical Fuzzy Logic. This section provides a mathematical exploration of the architectural design of the proposed CNN. The architecture of the proposed CNN consists of several key components, incorporating pooling layers, fully connected layers, and convolutional layers. These components combine to extract hierarchical features from the input data and perform classification tasks.

3.4.1. Convolutional Layers

Convolutional layers capture local features and spatial relationships in the input data. Let's denote the input to the k-th convolutional layer as X_k, which has dimensions $H_k \times W_k \times D_k$, where H_k represents the height, W_k represents the width, and D_k represents the number of channels. The convolution operation is performed by applying a set of learnable filters $F_k$with dimensions $F_h \times F_w \times D_k \times N_k$, where$F_h$ and F_w denote the filter height and width, respectively, and N_k represents the number of filters in the k-th layer.The output feature maps of the k-th convolutional layer, denoted as Z_k, can be computed using the convolution operation described in Eq.(26).

$$Z_{k[i,j,n]} = activation\left(\Sigma\Sigma\Sigma X_{k[a,b,d]} * F_{k[i-a,j-b,d,n]}\right) \quad (26)$$

where $Z_{k[i,j,n]}$ represents the activation at position (i,j) of the n-th filter in the k-th layer, $X_{k[a,b,d]}$ represents the input activation at position (a,b) in the channel d and activation() is the activation function, which introduces non-linearity.

3.4.2. Pooling Layers

The computational burden is lightened and some translation invariance is introduced by using pooling layers for down sampling each of the spatial dimensions of the feature maps. Max pooling is the most typical form of pooling. Given an input feature map X_k, the max pooling operation with a pooling size of $P_k \times P_k$ can be defined as Eq.(27).

$$Y_{k[i,j,n]} = max\left(X_k\left[P_{ki}: P_{k(i+1)}, P_{kj}: P_{k(j+1)}, n\right]\right) \quad (27)$$

where $Y_{k[i,j,n]}$ represents the output activation at position (i,j) of the n-th filter in the k-th layer.

3.4.3. Fully Connected Layers

After several convolutional and pooling layers, the feature maps are typically flattened into a vector and fed to one or more fully connected layers to learn high-level representations. Let's denote the input vector to the l-th fully connected layer as V_1, which has dimension$M_1 \times 1$, where M_1 represents the number of neurons in the l-th layer. The fully connected layer can be represented as Eq.(28).

$$Y_1 = W_1 * V_1 + B_1 \quad (28)$$

where Y_1 represents the output vector, W_1denotes the weight matrix with dimensions $M_{\{1+1\}} \times M_1$, and B_1 represents the bias vector with dimensions $M_{\{1+1\}} \times 1$. The weight matrix $W_1$and bias vector B_1 are learnable parameters that are updated during the training process.

3.4.4. Neutrosophical Fuzzy Logic-based Activation Function:

Neutrosophical Fuzzy Logic-based activation functions are employed in the proposed CNN instead of traditional ones. These functions introduce uncertainty and contradiction into the activation values, allowing for greater flexibility in the network. Let's denote the input activation to the l-th layer as A_1, which has dimension $M_1 \times 1$. The Neutrosophical Fuzzy Logic-based activation function for the l-th layer can be defined as Eq.(29).

$$\begin{aligned} A_{1[i]} = {} & \left(1-\mu_{1[i]}\right) * \left(1-\gamma_{1[i]}\right) * A_{1[i]} \\ & + \mu_{1[i]} * \gamma_{1[i]} * A_{1[i]} \\ & + \mu_{1[i]} * \left(1-\gamma_{1[i]}\right) \\ & * max\left(A_{1[i]}, 0\right) \\ & + \left(1-\mu_{1[i]}\right) * \gamma_{1[i]} \\ & * min\left(A_{1[i]}, 0\right) \end{aligned} \quad (29)$$

where $A_{1[i]}$represents the i −th element of the input activation vector to the l-th layer, $\mu_{1[i]}$denotes the truth membership value, and $\gamma_{1[i]}$ represents the indeterminacy membership value. These membership values allow for capturing the uncertain and contradictory nature of the activation values.

The truth membership value $\mu_{1[i]}$ and indeterminacy membership value $\gamma_{1[i]}$ can be

computed based on Neutrosophic Fuzzy Logic principles, considering the input activation and certain parameters specific to the layer. By incorporating the Neutrosophical Fuzzy Logic-based activation function, the proposed CNN introduces a new level of adaptability and robustness, allowing it to handle uncertain and contradictory information effectively.

3.4.5. Grasshopper Optimization-based Weight Update

The Grasshopper Optimization algorithm is a nature-inspired optimization algorithm that mimics the behaviour of grasshoppers in their search for optimal food sources. In the context of the proposed Grasshopper Optimization based Neutrosophical Fuzzy Logic CNN, the algorithm is adapted to update the network weights during training. Let's denote the network weights as W, including the weight matrices in the convolutional and fully connected layers. The goal is to optimize these weights based on the grasshoppers' fitness values and movement equations.

Initialization

The initialization phase in Grasshopper Optimization-based Weight Update involves setting up the initial conditions for the algorithm. Firstly, the population size, denoted as N, represents the number of grasshoppers. This determines the diversity of solutions explored during the optimization process. Next, the maximum number of iterations is determined, indicating the termination criterion for the algorithm. The population of grasshoppers is initialized randomly within the search space. Each grasshopper is assigned a position vector that represents a potential solution. By randomly distributing the grasshoppers, the algorithm covers a broader area of the search space, ensuring a comprehensive exploration of possible solutions. Additionally, the weight values, W_{min} and W_{max}, are set to define the range within which the weight factor will be adjusted. These values can be predefined based on the specific problem requirements. The decay rate parameter, γ, controls the rate at which the weight decays over iterations, influencing the balance between exploration and exploitation in the algorithm.

Fitness Evaluation

Fitness evaluation is a crucial step in the Grasshopper Optimization-based Weight Update. It involves assessing the quality of each grasshopper's position in the population based on the objective function of the optimization problem. The objective function captures the specific goals and constraints of the problem. For each grasshopper, the objective function is applied to its position vector, producing a fitness value that quantifies how well the grasshopper's solution performs. The fitness evaluation guides the algorithm by providing a measure of the quality or desirability of each solution. The objective function is problem-dependent and can vary widely across different optimization scenarios. It encapsulates the performance metrics, constraints, and goals of the problem domain. By evaluating the fitness of each grasshopper, the algorithm gains insights into the performance landscape and can make informed decisions about the search direction. The fitness values obtained during this step are vital in subsequent phases, such as selecting the best grasshopper and updating the global best position. The fitness evaluation step facilitates the identification of promising solutions and guides the algorithm towards the optimal solution in the search space.

Update of Best Grasshopper

After the fitness evaluation, the next step in the Grasshopper Optimization-based Weight Update is determining the best grasshopper in the population. This is done by identifying the grasshopper with the highest fitness value, representing the global best grasshopper. By comparing the fitness values of all grasshoppers, the algorithm selects the one that exhibits the most promising solution. This global best grasshopper represents the currently known optimal solution found during the iterations. The global best grasshopper is crucial in guiding the movement and exploration of other grasshoppers in subsequent steps. It serves as a reference for desirable positions and helps direct the search towards regions more likely to contain better solutions.

Weight Update

In Grasshopper Optimization-based Weight Update, the weight value, $W(t)$, is dynamically adjusted at each iteration to balance exploration and exploitation. The weight update equation is as Eq.(30).

$$W(t) = W_{min} + [(W_{max} - W_{min}) * exp(-\gamma * t)] \quad (30)$$

The weight value $W(t)$ determines the influence of the globally best position found so far, X_{best}, on the movement of grasshoppers. It regulates the exploitation aspect of the algorithm, encouraging the grasshoppers to converge towards the globally

best solution. The weight update equation ensures that the weight value starts from the minimum weight value, W_{min}, and decays exponentially over iterations according to the decay rate parameter γ. This decay factor allows the algorithm to gradually reduce the impact of the global best position, leading to a more diverse exploration of the search space in the early iterations. To achieve a happy medium between exploration and exploitation during optimization, the algorithm constantly modifies its behaviour by changing the weight value dependent on the current iteration. This adaptive weight update mechanism enhances the algorithm's convergence rate and improves its ability to find high-quality solutions.

Position Update

In the Grasshopper Optimization-based Weight Update, the position update step plays a crucial role in guiding the movement of grasshoppers towards potentially better solutions. Each grasshopper in the population's position is updated using Eq.(31).

$$X(t+1) = X(t) + R * exp(-\beta * t) * \left(best\_grasshopper\_position - X(t)\right) + W(t) * \left(X_{best} - X(t)\right) \quad (31)$$

Here, $X(t)$ represents the current position of the grasshopper at iteration t, $best\_grasshopper\_position$ denotes the position of the best grasshopper found so far, and X_{best} represents the global best position obtained during the optimization process.

The position update equation consists of two components: exploration and exploitation. The term $R * exp(-\beta * t) * \left(best\_grasshopper\_position - X(t)\right)$ introduces exploration by adding a random factor R and a decaying exponential term that directs the grasshoppers towards potentially unexplored regions of the search space.The term $W(t) * \left(X_{best} - X(t)\right)$ represents exploitation, as it guides the grasshoppers towards the globally best position found so far. The weight factor $W(t)$ determines the strength of exploitation, which is dynamically adjusted based on the weight update equation.

Boundary Handling

A boundary-handling mechanism is employed to ensure that the updated positions of grasshoppers remain within the boundaries of the search space. During the position update, if a grasshopper's new position violates the predefined boundaries, it is adjusted accordingly. Boundary handling techniques can vary depending on the problem domain. Common approaches include reflection, where the grasshopper's position is mirrored back into the feasible region, and random reinitialization, where the grasshopper is assigned a new position randomly within the boundaries. The purpose of boundary handling is to maintain the feasibility of solutions throughout the optimization process. Boundary handling ensures that the algorithm focuses on valid and meaningful solutions by preventing the grasshoppers from venturing outside the allowed search space.

Fitness Update

Following the position update, the fitness of the new positions of grasshoppers is re-evaluated using the objective function. The objective function calculates the fitness value based on the updated positions and represents the quality or desirability of each grasshopper's solution. By updating the fitness values, the algorithm captures the improvements or deterioration in the grasshoppers' solutions. This step allows for continuous monitoring and assessment of the population's performance, enabling the algorithm to adapt and adjust its search direction. The fitness update is vital for comparing the current solutions with the previously identified best solution. Suppose any grasshopper exhibits a better fitness value than the best global grasshopper. The global best grasshopper is updated accordingly. This ensures that the algorithm progresses towards the most promising solutions discovered throughout the optimization process. By iteratively evaluating and updating the fitness of the grasshoppers, the algorithm can refine its search and converge towards optimal or near-optimal solutions in the search space. The fitness update step is integral to the continuous improvement and refinement of the population during the optimization process.

Termination Condition

The termination condition determines when the Grasshopper Optimization-based Weight Update algorithm should stop iterating and conclude the optimization process. There are two common termination conditions: Maximum Number of Iterations: The algorithm can run for a predefined maximum number of iterations. Once this limit is reached, the algorithm terminates, regardless of whether an optimal solution has been found. This termination condition ensures the algorithm does not run indefinitely and allows for a controlled runtime. The algorithm can terminate if a certain fitness

threshold is achieved. This threshold represents a desired level of fitness that is considered satisfactory for the optimization problem. If the fitness of the global best grasshopper surpasses or meets this threshold, the algorithm stops iterating. Termination conditions help prevent unnecessary computations and allow for efficient resource utilization. They ensure that the algorithm terminates when it has either reached a satisfactory solution or has exhausted the predefined computational resources.

Once the termination condition is met, the Grasshopper Optimization-based Weight Update algorithm produces the final output, which consists of the global best position and its corresponding fitness value. This output represents the optimal or near-optimal solution found during the optimization process. The globally best position indicates the solution that achieved the highest fitness value throughout the iterations. It serves as the recommended solution to the given optimization problem. The fitness value associated with the globally best position quantifies the quality or desirability of the solution. The output provides valuable insights into the problem domain, allowing decision-makers to make informed choices based on the discovered optimal or near-optimal solution. Depending on the problem, the output can be further utilized for decision-making and analysis or as input for subsequent processes.

Algorithm 3:Grasshopper Optimization-based Weight Update

Step 1: **Initialization**
Set the population size and the number of possible repetitions.
Initialize the population of grasshoppers randomly within the search space.
Set the weight values, W_{min} and, W_{max}, and the decay rate parameter γ.

Step 2: **Fitness Evaluation**
Evaluate the fitness of each grasshopper in the population based on the objective function of the optimization problem.

Step 3: **Update of Best Grasshopper**
Identify the grasshopper with the best fitness value as the global best grasshopper.

Step 4: **Weight Update**
Calculate the weight value, $W(t)$, at each iteration

Step 5: **Position Update**
For each grasshopper in the population, update its position

Step 6: **Boundary Handling**
Ensure that the updated positions of grasshoppers lie within the boundaries of the search space. If a position violates the boundaries, adjust it accordingly.

Step 7: **Fitness Update**
Evaluate the fitness of the new positions of grasshoppers using the objective function.

Step 8: **Update the Best Grasshopper**
If any grasshopper has a better fitness value than the global best grasshopper, update the global best grasshopper accordingly.

Step 9: **Termination Condition**
Repeat steps 4 to 8 until the maximum number of iterations is reached or a termination criterion is satisfied (e.g., achieving a desired fitness threshold).

Step 10: **Output**
Return the global best position and corresponding fitness value as the optimal solution.

4. ABOUT AAU RAINSNOW TRAFFIC SURVEILLANCE DATASET

The Rain, Snow, and Bad Weather in Traffic Surveillance dataset focuses on the challenges of adverse weather conditions in traffic surveillance. It addresses the limitations of vision-based image analysis algorithms when visibility is impaired by factors such as rain, snow, haze, and fog. The dataset consists of 22 five-minute videos captured from seven different traffic intersections. These videos precisely capture rainfall and snowfall scenarios, representing varying lighting conditions from daylight to twilight and night. The scenes include challenging elements such as glare from car headlights, reflections from puddles, and blurring caused by raindrops on the camera lens. Data collection involved using a conventional RGB colour camera and a thermal infrared camera. Combining these two modalities, the dataset aims to facilitate robust detection and classification of road users even in difficult weather conditions. The dataset offers valuable features and attributes for training and evaluating object detection and classification algorithms under challenging weather conditions. These features include:

- **Weather Conditions:** The dataset focuses on two adverse weather conditions: rainfall and snowfall. This allows researchers to assess the performance of algorithms in

scenarios with impaired visibility due to precipitation.

- **Lighting Variations:** The dataset captures scenes with varying lighting conditions, ranging from broad daylight to twilight and night. This variation in illumination enables the evaluation of algorithms across different lighting scenarios, which is crucial for real-world deployment.
- **Challenging Elements:** The dataset includes challenging elements commonly encountered in bad weather, such as glare from car headlights, reflections from puddles, and blurring caused by raindrops on the camera lens. These elements mimic real-world conditions and provide realistic challenges for object detection algorithms.
- **Dual Modalities:** Data collection uses a conventional RGB colour camera and a thermal infrared camera. The availability of dual modalities enables researchers to explore the benefits of combining visual and thermal information for robust detection and classification of road users under adverse weather conditions.
- **Per-Pixel, Instance-Level Annotations:** The dataset provides per-pixel, instance-level annotations for road users in 100 frames randomly selected from each video sequence. This level of annotation granularity allows for detailed analysis and evaluation of algorithms, enabling precise localization and categorization of objects in challenging weather scenarios.
- **Annotation Statistics:** The dataset contains 2,200 annotated frames, with annotations for 13,297 objects. These annotation statistics provide sufficient labelled data for training and evaluating object detection and classification algorithms.

5. Performance Metrics

- **Precision:** It measures the accuracy of the detected moving objects. It is the proportion of successfully recognized moving items (true positives) over all detected moving objects (true positives plus false positives).
- **Recall:** It is also known as sensitivity, quantifies the completeness of the detected moving objects. It is determined by dividing the number of correct identifications by the sum of the correct identifications and the number of false negatives (moving objects missed).
- **Classification Accuracy:** In moving object detection, classification accuracy measures the algorithm's ability to classify moving objects and background regions correctly. It is the fraction of assessed areas that have been appropriately categorized.
- **F-Measure:** The F-measure, often known as the $F1$ score, is a composite measure of accuracy and recall that is used to rank algorithms. Precision measures the fraction of moving items accurately recognized out of the total number of detected objects, whereas recall measures the same fraction out of the total number of real-world moving objects.

All performance indicators are calculated using the words TP, TN, FP, and FN, which are widely used to assess the efficacy of an object identification system. The brief definition of each term:

- **True Positive (TP):** A true positive is when the algorithm correctly detects and identifies a moving object as present in the scene. In other words, it indicates that the algorithm has accurately identified a positive instance (the presence of a moving object) when it is indeed present.
- **True Negative (TN):** A true negative occurs when the algorithm correctly identifies and labels a region as background or non-object, with no moving object in that region. It represents the correct rejection of the absence of a moving object.
- **False Positive (FP):** A false positive is when the algorithm incorrectly identifies a region containing a moving object, but no object is present. It indicates a false alarm, where the algorithm detects an object where none exists.
- **False Negative (FN):** A false negative occurs when the algorithm fails to detect a moving object in the scene. It represents a missed detection, where the algorithm fails to identify the presence of an object that should have been detected.

6. RESULTS AND DISCUSSION

6.1. Precision and Recall Analysis

The analyses of precision and recall for the three approaches are shown in Figure 1: FRL, SD, and GOA-NFCNN. The result values of Figure 1 are provided in Table 1.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

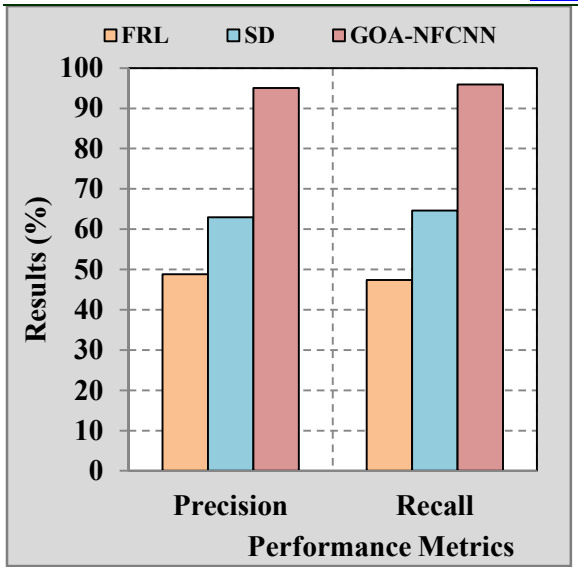

Figure 1. Precision And Recall

Firstly, regarding the precision results in Figure 1, FRL achieves a precision of 48.83%, indicating that 48.83% of the detected moving objects by this method are true positives. This means there is a relatively high chance of false positives, where background regions are mistakenly classified as moving objects. SD performs better with a precision of 62.93%, implying a higher accuracy in identifying true moving objects. However, the proposed GOA-NFCNN method surpasses both existing works with an impressive precision of 95.04%. This indicates a significantly lower rate of false positives and a higher accuracy in identifying moving objects. Secondly, regarding the recall results in Figure 1, FRL achieves a recall of 47.40%, indicating that only 47.40% of the true moving objects in the scene are correctly detected by this method. This implies a high rate of false negatives, where actual moving objects are missed. SD performs slightly better with a recall of 64.60%, indicating a higher ability to detect a more significant proportion of the true moving objects. However, GOA-NFCNN outperforms FRL and SD with a recall of 95.94%, indicating its superior capability to detect more true moving objects.

The precision and recall analysis results reveal essential insights into the performance of the different methods. FRL exhibits relatively low precision and recall values, indicating its limitations in accurately detecting moving objects. SD demonstrates better performance in terms of precision and recall, suggesting its ability to achieve a reasonable balance between accuracy and completeness. However, the proposed GOA-NFCNN method outshines FRL and SD, showcasing its exceptional precision and recall values. GOA-NFCNN demonstrates a significantly higher accuracy in identifying moving objects while maintaining a high level of completeness.

These findings highlight the effectiveness of the GOA-NFCNN approach in enhancing moving object detection. Fusing grasshopper optimization-based techniques with the neutrosophical fuzzy convolutional neural network contributes to exceptional precision and recall values. The GOA-NFCNN method achieves an excellent trade-off between accurately identifying moving objects and minimizing false positives and negatives.

Table 1. Precision And Recall Results

	Precision	Recall
FRL	48.83	47.40
SD	62.93	64.60
GOA-NFCNN	95.04	95.94

6.2. Classification Accuracy and F-Measure Analysis

Figure 2 illustrates the classification accuracy and F-measure analysis results for three different methods: FRL, SD, and GOA-NFCNN.

FRL achieves a relatively lower classification accuracy of 47.83% and an F-measure of 48.10%. This can be attributed to its working mechanism, which may not effectively capture moving objects' complex and diverse characteristics. FRL may struggle to handle variations in lighting conditions, occlusions, and background clutter, leading to misclassifications and imbalanced performance. FRL's fusion representation learning approach might not sufficiently capture the discriminative features necessary for accurate object detection.

SD performs better with a classification accuracy of 64.33% and an F-measure of 63.76%. The working mechanism of SD, which combines denoising and moving object detection, allows for more effective removal of noise and enhances object classification accuracy. By leveraging low-rank approximation techniques, SD can better extract relevant information and reduce the impact of noise and interference. This leads to improved classification accuracy and a more balanced F-measure than FRL.

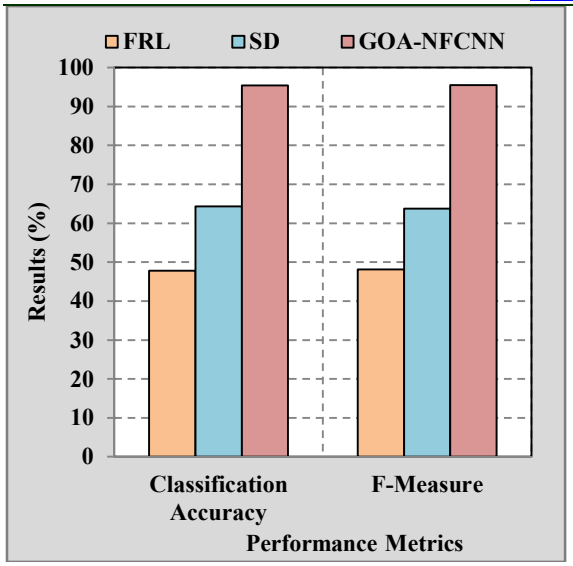

Figure 2. Classification Accuracy And F-Measure

GOA-NFCNN outperforms both FRL and SD with a classification accuracy of 95.38% and an F-measure of 95.49%. The impressive results can be attributed to the unique working mechanism of GOA-NFCNN. The integration of grasshopper optimization and neutrosophical fuzzy convolutional neural network allows for enhanced feature extraction, robust object representation, and precise classification. The grasshopper optimization algorithm optimizes the network parameters, enabling better adaptability to different scenarios and improving the accuracy of moving object detection. The neutrosophical fuzzy framework handles uncertainties and imprecise data, ensuring a more balanced performance in terms of precision and recall. This comprehensive approach enables GOA-NFCNN to achieve exceptional classification accuracy and a highly balanced F-measure.

The superior performance of GOA-NFCNN can also be attributed to its ability to handle various challenges in moving object detection, such as complex backgrounds, illumination changes, and occlusions. The fusion of grasshopper optimization and neutrosophical fuzzy techniques allows for effective representation learning, noise reduction, and robust feature extraction, leading to more accurate object detection and classification. Additionally, the deep learning capabilities of the neural network component in GOA-NFCNN enable it to learn and adapt to complex patterns and variations in moving objects, further enhancing its performance.

The achieved results in Figure 2 can be attributed to the working mechanisms of the methods. FRL struggles with accurate classification and balanced performance, while SD improves upon FRL by incorporating denoising techniques. However, GOA-NFCNN surpasses both methods by leveraging grasshopper optimization and neutrosophical fuzzy concepts, leading to significantly higher classification accuracy and a well-balanced F-measure. The unique working mechanism of GOA-NFCNN enables it to handle various challenges and extract robust features, resulting in superior performance in moving object detection tasks. The result values of Figure 2 are provided in Table 2.

Table 2. Classification Accuracy And F-Measure Results

	Classification Accuracy	F-Measure
FRL	47.83	48.10
SD	64.33	63.76
GOA-NFCNN	95.38	95.49

7. CONCLUSION

This research presents a novel approach called the Grasshopper Optimization-based Neutrosophical Fuzzy Convolutional Neural Network (GOA-NFCNN) for enhanced moving object detection. The proposed method leverages the fusion of grasshopper optimization and Neutrosophical fuzzy techniques to address the challenges associated with accurate object classification. The GOA-NFCNN method aims to overcome limitations in existing approaches and improve moving object detection under various weather conditions. The integration of grasshopper optimization and neutrosophical fuzzy concepts offers a promising solution to handle uncertainties, inaccurate data, and complex scenarios, thereby enhancing the accuracy and robustness of the detection process. The classification accuracy result further validates the effectiveness of the proposed GOA-NFCNN method. With a high classification accuracy of 95.38%, GOA-NFCNN demonstrates its capability to accurately classify moving objects, even under challenging conditions such as adverse weather, lighting variations, and occlusions. This remarkable accuracy showcases the potential of the proposed approach to address real-world scenarios and applications where precise object detection is crucial. The superior classification accuracy achieved by GOA-NFCNN indicates its ability to

extract relevant features, optimize network parameters through grasshopper optimization, and leverage the power of neutrosophical fuzzy techniques for robust object classification. This signifies the significant contributions of the proposed method in improving the performance of moving object detection systems. The results highlight the potential of GOA-NFCNN for various practical applications such as traffic surveillance, video analysis, and autonomous driving systems, where accurate and reliable moving object detection is essential. The utilization of GOA-NFCNN can lead to improved safety, efficiency, and decision-making in these domains.

REFERENCES:

[1]. Jadallah, H., Al Aghbari, Z.: SwapQt: Cloud-based in-memory indexing of dynamic spatial data. Futur. Gener. Comput. Syst. 106, 360–373 (2020). https://doi.org/10.1016/j.future.2020.01.009.

[2]. Niemiec, S.L.S., Wagas, R., Vigen, C.L.P., Blanchard, J., Barber, S.J., Schoenhals, A.: Preliminary User Evaluation of a Physical Activity Smartphone App for Older Adults. Heal. Policy Technol. 11, (2022). https://doi.org/10.1016/j.hlpt.2022.100639.

[3]. Dou, J., Li, J., Qin, Q., Tu, Z.: Moving object detection based on incremental learning low rank representation and spatial constraint. Infrared Phys. Technol. 124, 205–212 (2022). https://doi.org/https://doi.org/10.1016/j.jvcir.2018.09.009.

[4]. Mei, J., Ding, Y., Zhang, W., Zhang, C.: Fast detection, position and classification of moving objects on production line. Optik (Stuttg). 121, 2176–2178 (2010). https://doi.org/https://doi.org/10.1016/j.ijleo.2009.11.003.

[5]. Ahmed, S.A., Topalov, A. V, Shakev, N.G., Popov, V.L.: Model-Free Detection and Following of Moving Objects by an Omnidirectional Mobile Robot using 2D Range Data. IFAC-PapersOnLine. 51, 226–231 (2018). https://doi.org/https://doi.org/10.1016/j.ifacol.2018.11.546.

[6]. Christy Jeba Malar, A., Deva Priya, M., Janakiraman, S.: A Hybrid Crow Search and Gray Wolf Optimization Algorithm-based Reliable Non-Line-of-Sight Node Positioning Scheme for Vehicular Ad hoc Networks. Int. J. Commun. Syst. 34, e4697 (2021). https://doi.org/10.1002/dac.4697.

[7]. Resmi, H.B., Deepambika, V.A., Rahman, M.A.: Symmetric Mask Wavelet Based Detection and Tracking of Moving Objects Using Variance Method. Procedia Comput. Sci. 58, 58–65 (2015). https://doi.org/https://doi.org/10.1016/j.procs.2015.08.012.

[8]. Elafi, I., Jedra, M., Zahid, N.: Unsupervised detection and tracking of moving objects for video surveillance applications. Pattern Recognit. Lett. 84, 70–77 (2016). https://doi.org/https://doi.org/10.1016/j.patrec.2016.08.008.

[9]. Ramkumar, J., Vadivel, R.: Improved frog leap inspired protocol (IFLIP) – for routing in cognitive radio ad hoc networks (CRAHN). World J. Eng. 15, 306–311 (2018). https://doi.org/10.1108/WJE-08-2017-0260.

[10]. Ramkumar, J., Vadivel, R.: Performance Modeling of Bio-Inspired Routing Protocols in Cognitive Radio Ad Hoc Network to Reduce End-to-End Delay. Int. J. Intell. Eng. Syst. 12, 221–231 (2019). https://doi.org/10.22266/ijies2019.0228.22.

[11]. D. Jayaraj, J. Ramkumar, M. Lingaraj, B. Sureshkumar, "AFSORP: Adaptive Fish Swarm Optimization-Based Routing Protocol for Mobility Enabled Wireless Sensor Network", International Journal of Computer Networks and Applications(IJCNA), 10(1), PP: 119-129, 2023, DOI: 10.22247/ijcna/2023/218516.

[12]. Jaganathan, R., Vadivel, R.: Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks. Int. J. Comput. Digit. Syst. 10, 1063–1074 (2021). https://doi.org/10.12785/ijcds/100196.

[13]. Vadivel, R., Ramkumar, J.: QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications. Inc. Internet Things Healthc. Appl. Wearable Devices. 109–121 (2019). https://doi.org/10.4018/978-1-7998-1090-2.ch006.

[14]. Ramkumar, J., Vadivel, R.: CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks. In: Advances in Intelligent Systems and Computing. pp. 145–153. Springer Verlag (2017). https://doi.org/10.1007/978-981-10-3874-7_14.

[15]. Lingaraj, M., Sugumar, T.N., Felix, C.S., Ramkumar, J.: Query aware routing protocol for mobility enabled wireless sensor network.

Int. J. Comput. Networks Appl. 8, 258–267 (2021). https://doi.org/10.22247/ijcna/2021/209192.

[16]. Ramkumar, J., Vadivel, R.: Whale optimization routing protocol for minimizing energy consumption in cognitive radio wireless sensor network. Int. J. Comput. Networks Appl. 8, 455–464 (2021). https://doi.org/10.22247/ijcna/2021/209711.

[17]. Ramkumar, J., Samson Dinakaran, S., Lingaraj, M., Boopalan, S., Narasimhan, B.: IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion. Presented at the (2023). https://doi.org/10.1007/978-981-19-8353-5_2.

[18]. Ramkumar, J., Vadivel, R Meticulous Elephant Herding Optimization based Protocol for Detecting Intrusions in Cognitive Radio Ad Hoc Networks. Int. J. Emerg. Trends Eng. Res. 8, 4548–4554 (2020). https://doi.org/10.30534/ijeter/2020/828820 20.

[19]. Ramkumar, J., Vadivel, R.: Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks. Wirel. Pers. Commun. 120, 887–909 (2021). https://doi.org/10.1007/s11277-021-08495-z.

[20]. Ramkumar, J.: Bee inspired secured protocol for routing in cognitive radio ad hoc networks. Indian J. Sci. Technol. 13, 2159–2169 (2020). https://doi.org/10.17485/ijst/v13i30.1152.

[21]. Ramkumar, J., Kumuthini, C., Narasimhan, B., Boopalan, S.: Energy Consumption Minimization in Cognitive Radio Mobile Ad-Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol. 2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022. 1–6 (2022). https://doi.org/10.1109/ICACTA54488.2022.9752899.

[22]. Menakadevi, P., Ramkumar, J.: Robust Optimization Based Extreme Learning Machine for Sentiment Analysis in Big Data. 2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022. 1–5 (2022). https://doi.org/10.1109/ICACTA54488.2022.9753203.

[23]. Sharma, M., Kandasamy, I., Vasantha, W.B.: Comparison of neutrosophic approach to various deep learning models for sentiment analysis[Formula presented]. Knowledge-Based Syst. 223, 107058 (2021). https://doi.org/10.1016/j.knosys.2021.107058.

[24]. Munigadiapa, P., Adilakshmi, T.: MOOC-LSTM: The LSTM Architecture for Sentiment Analysis on MOOCs Forum Posts. (2023). https://doi.org/10.1007/978-981-19-3391-2_21.

[25]. Zhang, Y., Zheng, J., Zhang, C., Li, B.: An effective motion object detection method using optical flow estimation under a moving camera. J. Vis. Commun. Image Represent. 55, 215–228 (2018). https://doi.org/https://doi.org/10.1016/j.jvcir.2018.06.006.

[26]. Wang, P., Wu, J., Fang, A., Zhu, Z., Wang, C., Ren, S.: Fusion representation learning for foreground moving object detection. Digit. Signal Process. 138, 104046 (2023). https://doi.org/https://doi.org/10.1016/j.dsp.2023.104046.

[27]. Yang, Y., Yang, Z., Li, J.: Novel RPCA with nonconvex logarithm and truncated fraction norms for moving object detection. Digit. Signal Process. 133, 103892 (2023). https://doi.org/https://doi.org/10.1016/j.dsp.2022.103892.

[28]. Wang, L., Hu, M., Kong, K., Tao, J., Ji, K., Dai, Z.: A deep-learning-assisted versatile electret sensor for moving object detection. Nano Energy. 104, 107934 (2022). https://doi.org/https://doi.org/10.1016/j.nanoen.2022.107934.

[29]. Zhang, Z., Chang, Y., Zhong, S., Yan, L., Zou, X.: Learning dynamic background for weakly supervised moving object detection. Image Vis. Comput. 121, 104425 (2022). https://doi.org/https://doi.org/10.1016/j.imavis.2022.104425.

[30]. Rebai Boukhriss, R., Fendri, E., Hammami, M.: Moving object detection under different weather conditions using full-spectrum light sources. Pattern Recognit. Lett. 129, 205–212 (2020). https://doi.org/https://doi.org/10.1016/j.patrec.2019.11.004.

[31]. Giveki, D.: Robust moving object detection based on fusing Atanassov's Intuitionistic 3D Fuzzy Histon Roughness Index and texture features. Int. J. Approx. Reason. 135, 1–20 (2021). https://doi.org/https://doi.org/10.1016/j.ijar.2021.04.007.

[32]. Soitinaho, R., Moll, M., Oksanen, T.: 2D LiDAR based object detection and tracking on a moving vehicle. IFAC-Papers OnLine.

55, 66–71 (2022). https://doi.org/https://doi.org/10.1016/j.ifacol.2022.11.116.

[33]. B., S., Tom, A.J., George, S.N.: Simultaneous denoising and moving object detection using low rank approximation. Futur. Gener. Comput. Syst. 90, 198–210 (2019). https://doi.org/https://doi.org/10.1016/j.future.2018.07.065.

[34]. A. Senthilkumar, J. Ramkumar, M. Lingaraj, D. Jayaraj, B. Sureshkumar, "Minimizing Energy Consumption in Vehicular Sensor Networks Using Relentless Particle Swarm Optimization Routing", International Journal of Computer Networks and Applications (IJCNA), 10(2), PP: 217-230, 2023, DOI: 10.22247/ijcna/2023/220737.

[35]. L. Mani, S. Arumugam, and R. Jaganathan, "Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol," ACM Int. Conf. Proceeding Ser., pp. 1–5, Dec. 2022, doi: 10.1145/3590837.3590907.