Dheepika PS
Umadevi V

A NOVEL HYBRID DEEP LEARNING APPROACH FOR 3D OBJECT DETECTION AND TRACKING IN AUTONOMOUS DRIVING

Abstract *Recently Object detection and tracking using fusion of LiDAR and RGB camera for the autonomous vehicle environment is a challenging task. The existing works initiates several object detection and tracking frameworks using Artificial Intelligence (AI) algorithms. However, they were limited with high false positives and computation time issues thus lacking the performance of autonomous driving environment. The existing issues are resolved by proposing Hybrid Deep Learning based Multi Object Detection and Tracking (HDL-MODT) using sensor fusion methods. The proposed work performs fusion of solid state LiDAR, Pseudo LiDAR, and RGB camera for improving detection and tracking quality. At first, the multi-stage preprocessing is done in which noise removal is performed using Adaptive Fuzzy Filter (A-Fuzzy). The pre-processed fused image is then provided for instance segmentation to reduce the classification and tracking complexity. For that, the proposed work adopts Lightweight General Adversarial Networks (LGAN). The segmented image is provided for object detection and tracking using HDL. For reducing the complexity, the proposed work utilized VGG-16 for feature extraction which forms the feature vectors. The features vectors are then provided for object detection using YOLOv4. Finally, the detected objects were tracked using Improved Unscented Kalman Filter (IUKF) and mapping the vehicles using time based mapping by considering their RFID, velocity, location, dimension and unique ID. The simulation of the proposed work is carried out using MATLAB R2020a simulation tool and performance of the proposed work is compared with several metrics that show that the proposed work outperforms than the existing works.*

Keywords 3D object detection, object tracking, hybrid deep learning, pre-processing, segmentation, sensor image fusion

Citation Computer Science 25(3) 2024: 435–467

1. Introduction

In recent days, autonomous vehicles have developed rapidly, in which three-dimensional (3D) multiobject detection and tracking are performed [10]. This provides vital information by estimating traffic scale over time, and orientation. Various types of sensors are used to detect objects and perform tracking such as optical RADAR, cameras, and LiDAR in autonomous vehicles especially, Light Detection and Ranging (LiDAR) are equipped in autonomous vehicles to recognize and detect multiple 3D objects which provide deep measurements [2, 21]. RGB images are also captured from cameras with high resolution to detect the objects [27, 28]. Pseudo-LiDAR is used in several approaches to extract depth from the images and transform the image into 3D cloud points [19]. However, several challenges occurred in Pseudo LiDAR due to high sparse points and complexity [8, 23]. To overcome this challenge, RGB-D images and LiDAR cloud points are combined to perform multi-object detection by considering obstacles, static and dynamic objects. However, environmental and other factors increase the noise which reduces the detection accuracy [17, 22]. Bounding box estimation is performed by considering orientation and location of all objects present in the frame to detect the objects. In some works, image features are considered to estimate bounding boxes. However, the appearance and geometrical factors of the image are ignored which reduces the accurate object detection when it is in dynamic state (i.e., motorcycles, pedestrians, etc.) [18]. Segmentation is performed to improve classification accuracy. Various types of features are extracted from the point cloud images and RGB images which are under the category of spatial and temporal related features [3]. In several works, ground extraction is performed initially then classify the non-ground points based on several classes which reduce the complexity of object segmentation and provide high accuracy. However, it reduces the speed which is not applicable for real-time scenarios. Various deep learning neural networks are used to detect and track objects such as convolutional neural networks (CNN), residual neural network (ResNet), you only look once (YOLO), single-shot detector (SSD), etc., in which YOLO and SSD are single-stage classification algorithms whereas CNN, ResNet algorithms are two-stage algorithms [9]. Initially, these algorithms are used to detect 2D objects. Later the 2D cloud points are converted into 3D voxels using 3D CNN for detecting 3D objects [13]. The point cloud images are projected in bird's eye view in some works to know the dense of the image. However, it consists of several challenges such as occlusion of objects and perspective-related problems. However, single-stage algorithms do not detect small objects accurately whereas two-stage algorithms do not applicable for real-time scenarios due to low speed [31].

Acronyms

LIDAR	–	Light Detection and Ranging
RGB-D	–	Red Green Blue-Depth
HDL-MODT	–	Hybrid Deep Learning based Multi Object Detection and Tracking

A FUZZY	–	Adaptive Fuzzy Filter
MSO	–	Moth Swarm Optimization
LGAN	–	Lightweight General Adversarial Networks
VGG	–	Visual Geometry Group
HDL	–	Hybrid Deep Learning
YOLO	–	You Only Look Once
IUKF	–	Improved Unscented Kalman Filter
RFID	–	Radio-Frequency IDentification
RADAR	–	Radio Detection And Ranging
CNN	–	Convolutional Neural Networks
ResNet	–	Residual Neural Network
SSD	–	Single-Shot Detector
KITTI	–	Karlsruhe Institute of Technology and Toyota Technological Institute

1.1. Motivation and objectives

The 3D object detection and tracking method faced many shortcomings in terms of Less detection, classification accuracy, and less tracking accuracy. The existing works provided some approaches however, they failed to provide precise results. Some of the shortcomings faced by the existing works are:

- **Less Detection Accuracy.** The existing works directly acquire images from the datasets and undergo further processes which limit them with less detection accuracy as the directly acquired images suffer from noise and poor quality. Also, the existing works employ single stage detector for object detection, which also limits the detection accuracy.
- **Complexity in Object Classification.** High complexity in classification affects classification accuracy. The existing works are limited with high complexity during classification as they extract raw features directly from the images without performing segmentation.
- **Poor Object Tracking.** The existing works lack poor tracking accuracy as they consider only current and previous time stamps for object tracking however, some of the object tracking-related metrics are not considered.

The above major pitfalls motivated us to deliver a robust, reliable, and accurate framework with the aim of detecting and tracking the objects with high detection accuracy and rapid classification using Solid-state LiDAR, Pseudo LiDAR, and RGB-D images. In addition, various problems are addressed in this research based on sparsity of cloud points, false positive rates, etc.

The foremost objective of this work is to detect and track the 3D objects using solid-state LiDAR, pseudo-LiDAR, and RGB-D images with high detection accuracy and rapid classification for efficient tracking. The remaining objectives of this proposed work are sorted below.

- To enhance the quality of LiDAR and RGB-D images, pre-processing is performed for removing the noise and equalizing the luminance effect which increases the accuracy of the detection results.
- To improve the viewport prediction of the input image by performing image rotation and instance segmentation for detecting the objects' pose accurately even for small objects which improves the tracking reliability.
- To reduce the false-positive rate, extraction of multiple features is performed which extracts numerous features to increase the accuracy of detection and tracking.

1.2. Research contribution

Designing a highly accurate 3D object detection and tracking model for autonomous driving using deep learning algorithm is the major aim of this work. Some of the research contributions are provided below:

- The problem of image quality, noise factors, and less contrastness are resolved by performing multi stage pre-processing. The existing works performs only noise removal as pre-processing technique. The proposed work performs multi-stage pre-processing as A-Fuzzy based noise removal, MSO based contrast enhancement, and point to voxel conversion.
- The complexity issues during object detection and tracking are resolved by performing instance segmentation using LGAN algorithm. Most of the existing works provides the raw data to the classifier which increases the computation of object detection and classification respectively.
- The accuracy of the object detection and tracking is improved by adopting HDL algorithm in which VGG 16 is utilized for feature extraction, and YOLOv4 is utilized for object detection and classification. Further, the object tracking is achieved by adopting IUKF algorithm based on RFID, unique ID, location, velocity, and dimension.

1.3. Paper organization

The rest of this paper is organized as follows; the section II provides the literature survey along with the existing gaps. Section III emphases the problem statement which shows the major research works and their corresponding problems. Section IV details the proposed work with detailed explanation along with diagrams and pseudocodes. Section V explains the experimental analysis in which four sub-sections provides such as simulation setup, dataset description, experimental analysis, and research summary. Section VI concludes the proposed work.

2. Literature survey

This section emphasis, the existing literatures and gaps associated with them in object detection and tracking for autonomous driving. Furthermore, this section also subdivided into three sections which are also listed below.

2.1. Object detection approaches

Authors in this work introduce camera image-based 3D object detection [29]. This work extracts Pseudo LiDAR points from stereo images and performs object classification which give low cost for object detection however, the Pseudo LiDAR points are prone to high sparsity.

Authors in this work perform segmentation of foreground-based object detection from the LiDAR cloud points [24]. Here raw LiDAR point clouds were taken as input for real-time object detection however, the LiDAR sensor is prone to noise and environmental conditions which leads to less detection accuracy.

Authors in this work perform autonomous vehicle detection by employing LiDAR point clouds [5]. This work extracts feature from the raw LiDAR point clouds however, the extraction of features from the raw LiDAR point clouds leads to high complexity in object classification.

The 3D object detection for autonomous vehicles was performed using fusion of LiDAR and camera data approach was discussed in [32]. This work utilized convolutional neural network for 3D object detection however, the convolutional neural network is limited with feature redundancy which leads to high complexity during object classification.

2.2. Object tracking approaches

Authors in this paper introduced 3D probabilistic object tracking model for autonomous driving [4]. This work considers both camera and LiDAR images for 3D object tracking. Based on the matching result, the object tracking was performed and unmatched results were further provided to initialization of tracking phase.

Authors in this work perform a 3D object tracking framework by introducing SIMTRACK [15]. This work performs both object detection and classification by considering only LiDAR images as input. This work attains less detection accuracy and poor tracking as they considered only raw LiDAR point clouds for object detection, and considered only current and previous time stamps for object tracking respectively.

In this paper [25], author proposed an approach to perform tracking of multiple targets for autonomous vehicle environment using YOLOv3. Experimental analysis is performed using two datasets namely KITTI and UA-DETRAC datasets in terms of processing speed and accuracy. Here, YOLOv3 based object detection and tracking was performed. However, it cannot able to detect small objects in an efficient manner which reduces the tracking efficiency.

Authors in this work [16], perform camera fusion methods for tracking objects using 3D in space. This work fused the imaging modalities such as radar and 3D camera. This work directly provides the fused data to the center fusion network without pre-processing it, that leads to lesser tracking accuracy.

2.3. Object detection and tracking approaches

Authors in this paper [20], proposed object detection and tracking for moving objects using 360-degree view camera. Here, moving object is detected based on the position and velocity, however direction is also an important metric for object tracking, hence this research obtains less performance in moving object tracking that reduces tracking accuracy.

Authors in this work [7], utilized Kalman filter method for performing object detection and tracking for autonomous vehicles. The object detection and tracking model was highly suitable for pedestrians, bicycles, and cars. The results shows that the fusion of LiDAR and radar gains better results than the radar and LiDAR only modalities.

In [1], authors perform fusion methodology for enabling object detection and tracking for autonomous vehicles. The detected objects were tracked using radar which used gaussian mixture probability hypothesis density filtering algorithm based on three phases such as booting, prediction, and update. The gaussian mixture probability density hypothesis filtering was highly linear that did not suit for real time environment.

As same as the aforementioned papers, authors in [14] also performed fusion of camera and radar for joint object detection and tracking. This work utilized faster regional convolutional neural network for object detection whereas the radar information was utilized for object tracking. Here, the utilization of faster regional convolutional neural network limits with higher time consumption and less convergence.

3. Problem statement

The major problems associated with the specific prior works are provided in this section. Furthermore, this section also provides the brief research solutions for the mentioned problems.

An accurate and effective 3D object detection framework for autonomous vehicles was introduced in this work [30]. This work consists of three phases namely fusion phase, voxel-wise feature encoder phase, and 3D backbone network phase. This work performs 3D object detection by aggregating the RGB image and LiDAR point cloud image. The LiDAR point cloud image images were voxelized in order to extract the point-wise features, and the RGB image features are extracted directly from the RGB images. Both the extracted features are aggregated in the fusion phase. Finally, the extracted voxel features are fed to 3D backbone network which consists of 3D sparse convolutional layers and performs bounding box classification.

Authors in this work introduces an object detection framework in real-time environment using LiDAR [6]. This work consists of phases such as input data acquisition, segmentation, and classification. Initially, the LiDAR point cloud data were acquired from the data set from which LiDAR point cloud map was formed. The LiDAR point cloud map was provided for segmentation in which three sub-phases are involved namely hierarchical segmentation, hierarchical merge, and extraction of ground. Finally, the Yolov4 classifies and detects the objects which were represented in 3D bounding boxes.

The major limitations associated with those works are listed below:

- This work employs LiDAR sensors for acquiring LiDAR point cloud images which were effective and accurate however, the LiDAR sensors are limited with high cost and prone to environmental conditions.

- Here, feature extraction was done during classification phase in which only limited features are extracted however, this attains poor classification as they considered only limited features (i.e., only textual features).

- The adoption of single-stage detector (i.e., Yolov4) in [6] was used for feature extraction and object detection which also limits with less detection accuracy as it did not withstand with heavy features.

A joint object detection and object tracking framework using LiDAR point clouds was introduced in this work was discussed in [26]. This work consists of four phases namely feature extraction phase, association phase, refinement phase, and trajectory phase. Initially, the two LiDAR point cloud points are taken inputs, that were provided to the feature extraction phase in which point-wise features are extracted for both the images and 3D bounding boxes were assigned. The output of the feature extraction phase was provided to association phase in which feature fusion and foreground removal were taken place. The refinement phase was used to refine the aggregated features and also provides the tracking displacement information of both frames. Finally, the trajectory phase matches the displacement of both frames and tracks the image in bird's eye view.

Authors in this work [12] utilized LiDAR and camera, joint object tracking and classification were introduced in this work. This work consists of two stages namely detection stage, and classification stage. Initially, the camera and LiDAR point cloud images with current and previous time stamps are provided to combine networks in which both the temporal and spatial information are combined. Based on the combining result, heat map was formed. The fusion network fuses both the information provided for object detection in which regional proposals and refinement of the proposals were made and performs object detection. Based on the detection and time stamps, spatiotemporal graphs were constructed. Finally, based on the graphs, the object tracking was performed in adjacent network.

The major problems centred in this work are listed below:

- Here, the raw input LiDAR input images were taken for feature extraction and object detection however, the acquisition of raw LiDAR was prone to noise, environmental conditions, and also achieves increased complexity.
- The object tracking was performed based on the current and previous timestamps by using graph neural networks however, the tracking accuracy was affected by not considering some of the tracking-related attributes (location, dimension, orientation, etc.).
- The 3D object detection was performed based on the regional proposals and refinement for the fused features however, the features fed to the classifier was not effective as it holds unnecessary background information which increases the complexity.

3.1. Research solutions

The aforementioned research problems are resolved by proposing 3D multi-objective object detection and tracking method using deep learning algorithm. The proposed work fused the three input images such as RGB-D, pseudo-LiDAR cloud points, and solid-state LiDAR for improving the accuracy of object detection and tracking. At first, the images acquired from the dataset are preprocessed in which the proposed work performs multi stage pre-processing which includes noise removal using A-Fuzzy, contrast enhancement using MSO, and point to voxel conversion. The pre-processed fused images are fed for segmentation to reduce the classifier complexity. As the fused images are of different orientations, the proposed work tends to manage them by rotating the images into four degrees such as 10°, 90°, 180°, and 270°. Once, all the images are properly oriented the instance segmentation is performed by L-GAN algorithm. From the segmented part, the feature extraction and classification is performed using hybrid deep learning algorithm named VGG 16 and YOLOv4 respectively. The VGG 16 is utilized for feature extraction whereas the YOLOv4 for enabling high speed and precise classification of moving object. The detected objects are then tracked based on several metrics using IUKF. Furthermore, the reliability of tracking is improved by performing mapping in location and time respectively.

4. Proposed work

Accurate 3D object detection and tracking are mainly focused in this research for autonomous vehicle environments. For this purpose, we take input images from Solid-state LiDAR, Pseudo LiDAR, and RGB-D images. The adoption of intelligence algorithms in this work is to ensure the precision, reliability, and timeliness of the proposed framework. The proposed work adopts Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) dataset for effectively train the proposed deep learning algorithm for autonomous driving environment. The system model of the proposed work is shown in Figure 1. This proposed work consists of three sequential processes which are described as follows.

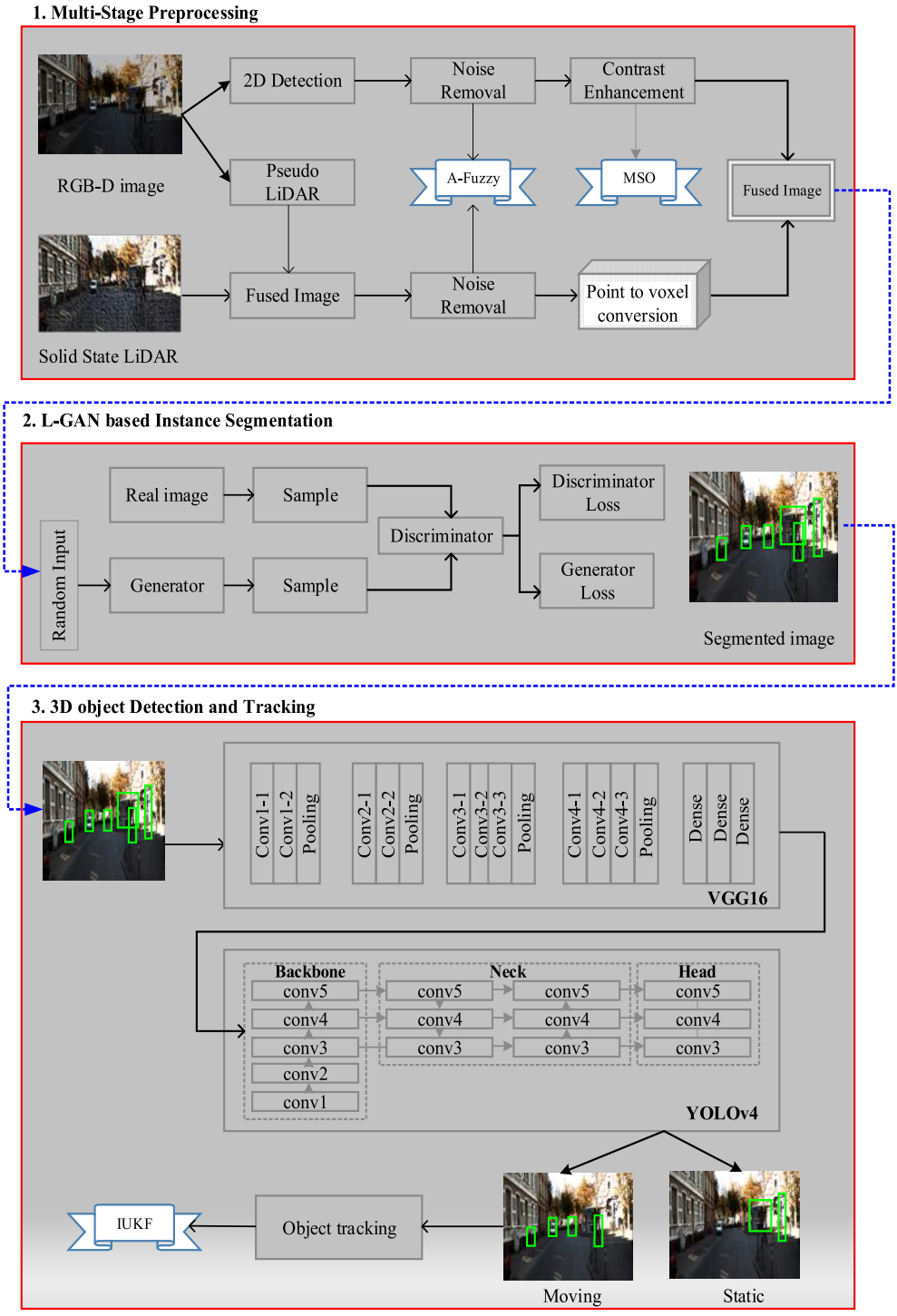


Figure 1. Overall architecture of proposed 3D object & tracking model

4.1. Multi stage pre-processing

Initially, we have taken three types of inputs, they are RGB-D images, Pseudo LiDAR cloud points, and Solid-state LiDAR cloud points in which the pseudo LiDAR cloud points are obtained from the RGB-D images. Solid-state LiDAR and Pseudo LiDAR are fused to decrease the sparsity. The Solid-state LiDAR is the emerging technology in 3D object detection that overcomes the LiDAR in terms of low cost, high speed, and better accuracy. The fusion of three inputs, cancels the disadvantages of one another thereby achieving better representation of real time scenes. The Table 1 represents the comparison of upsides and downsides of RGB-D images, Pseudo LiDAR cloud points, and Solid-state LiDAR cloud points. In addition, it illuminates the scenario with high speed.

Table 1
Comparison of proposed inputs

Proposed inputs	Upsides	Downsides
Solid-state LiDAR	– Compact size and structure – Free from calibration and effectively capture important features – Less cost than conventional LiDAR	– Not with stand with bad climatic situations – Not suitable for omnidirectional scanning
Pseudo-LiDAR	– Clearer depth information – Low power when compared to LiDAR	– High depth estimation error – Ineffective in capturing local features
RGB-D	– Provides orientation and poses of the objects – Provides omnidirectional view of the environment	– Problem of interference when two cameras capture same scene – Capturing and measuring range is limited

Multi-stage preprocessing is classified into three stages which are explained as follows.

4.1.1. Noise removal

Generally, RGB-D images and LiDAR cloud points have more noises which reduce the quality of the images. To overcome this issue, noise removal is performed using A-Fuzzy with two stages. In the initial stage, the pixels are classified as noisy and good. The noisy pixels are taken in the second stage to remove the noise. In addition, this filter preserves the edge which increases the detection accuracy efficiently.

Stage 1: For pixel (pix_{ji}) of an input image, the pixel set of neighborhoods (nei_{ji}^{W}) is assessed which is associated with the $pix_{ji} \in K$ in which K is the input image

among that W is half filter window. Therefore, nei_{ji}^{W} can be formulated as,

$$nei_{ji}^{W} = \{pix_{j+n,i+r} \ \ \forall n, r \in [-W, W]\} \tag{1}$$

From the above equation, the size of nei_{ji}^{W} is $(2W+1)\cdot(2W+1)$. For instance, if k_m is the pixel element in nei_{ji}^{W} then $m = 1, 2, ..., M$ in which $M = (2W+1)\cdot(2W+1)$. Once, the nei_{ji}^{W} is computed, the membership function for every nei_{ji}^{W} element is computed. In this stage, the noise intensity value is determined with range of 0 to 1 i.e. $pix_{ji} \notin \{0, 1\}$. For every element k_m of nei_{ji}^{W} the membership function mem_{ji}^{W} is determined based on gaussian membership functions $\delta_{(mem_{ji}^{W})}(k_m) : nei_{ji}^{W} \to [0, 1]$. For the type-3 fuzzy set, the $\widetilde{(mem)}_{ji}^{W}$ is defined by the $\delta_{\widetilde{mem}_{ji}^{W}}(k_m, \delta_{mem_{ji}^{W}})$. The association of mem_{ji}^{W} with the gaussian membership function can be formulated as,

$$\delta_{(mem)_{ji}^{(W,n)}}(k_m) = e^{-(k_m - \vartheta_{ji}^{(W,n)})^2/2(\rho_{ji}^{W})^2} \tag{2}$$

where, $\vartheta_{ji}^{(W,n)}$ is the mean function that varies based on n, and ρ_{ji}^{W} is the variance that is set as constant. The formulation of $\vartheta_{ji}^{(W,n)}$ and ρ_{ji}^{W} can be provided below,

$$\vartheta_{ji}^{(W,n)} = \varpi_{\beth}(nei_{ji}^{W}), \beth = 1, 2, 3, ..., h \tag{3}$$

$$\rho_{ji}^{W} = \varpi_{h}(R_{ji}^{W}) \tag{4}$$

From the above equations,$\varpi_{\beth}$ is the mean of $\beth$ middle, and ϖ_h is the standard mean. Utilizing $\iota_1$norm, the R_{ji}^{W} can be formulated as follows,

$$\begin{aligned} R_{ji}^{W} &= \{|k_m - av_{\vartheta}|, \forall \in k_m \in nei_{ji}^{W}\} \\ av_{\vartheta} &= \frac{1}{h}\sum_{n=1}^{h} \vartheta_{ji}^{(W,n)} \end{aligned} \tag{5}$$

where, the average mean can be defined as av_{ϑ} from the mean of $\beth$ middle. From the $\delta_{mem_{ji}^{(W,n)}}$, mean, and variance the membership matrix (∇_{ji}) is constructed with size of $h \times M$ that composed values of membership function of k_m.

From the matrix, the threshold value (th^n) is determined for pixel classification that can be formulated as,

$$th^n = min(max(\nabla_{ji})) \tag{6}$$

From the above equation, *min* and *max* denotes the minimum and maximum operators respectively. In the ∇_{ji}, the column wise operation includes the association of mem_{ji}^{W} with nei_{ji}^{W} that can be expressed as,

$$\delta_{(mem)_{ji}^{W}} = \frac{\sum_{n=1}^{h} \nabla_{ji}}{h} \ \ \forall \ \ j = 1, ..., M \tag{7}$$

From the Equations (6) and (7), the pixel quality is determined. If the $\delta_{(mem)_{ji}^{W}} > th^{n}$ then the pixel is considered as good otherwise considered as noisy pixels.

Stage 2: In this stage, the noisy pixels of the input images were denoised based on the good pixels. The good pixels sets were denoted as β with mapping function of [0,1] by the membership function δ_{β}. The mean gaussian membership function is computed for the β based on mean of $\exists$ middle. From which the average of $\exists$ – values is taken from the set of β. Therefore, the avg_m and ρ_{β} of δ_{β} can be formulated as,

$$avg_n = \frac{\sum_{n=1}^{h} m_n}{h}; \rho_{\beta} = |\beta - avg_m| \tag{8}$$

$$\delta_{\beta}(g_j) = e^{-(g_j-(avg)_m)^2/2\rho_{\beta}^2} \tag{9}$$

From the above Equation (9), m_n is the n-th good pixels mean $(n = 1, 2, ..., h)$. The denoise intensity pixels can be formulated as,

$$de_{pix} = \frac{\sum_{\forall g_j \in \beta} \mathbb{I}_j g_j}{L}; L = \sum_{j=1}^{de} I_j \tag{10}$$

where, the weight of the good pixel can be denoted as $I_j \in \delta_{\beta}$, and the normalized term is denoted as L. The pseudo code for the proposed noise removal step using A-Fuzzy in multi stage pre-processing is provided below.

Algorithm 1 Pseudocode for Noise Removal Using A-Fuzzy

Input: Noisy 2D Image
Output: Denoised Image
Begin
//Noise Removal//
for all input images **do**
    for every pix_{ji} in K **do**
        Determine the neighborhood pixels nei_{ji}^{W} (1)
        Compute gaussian membership function (2)
        Determine mean $(\vartheta_{ji}^{(W,n)})$ and variance (ρ_{ji}^{W}) (3), (4)
        Construct membership matrix ∇_{ji}
        Determine threshold and $\delta_(mem)_{ji}^{W}$ (6) and (7)
        if $\delta_{(mem)_{ji}^{W}} > th^{n}$ **then**
            Good Pixel
        else
            Bad Pixel
        end if
    end for
end for
End

4.1.2. Contrast enhancement

After removing the noise, contrast enhancement is performed to improve the quality of RGB-D images using MSO algorithm which equalizes the histogram of the images by improving the visibility based on brightness adjustment in an efficient manner. This equalizes the luminance to increase the detection accuracy.

The noise removed image is denoted as $de(j,i)$ in which $j = 1,2,...,N$ and $i = 1,2,...,R \in z^{NxR}$ in which the (j,i) is the gray location in the image of size $N \cdot R$. At first, the given denoised image is segmented into non-overlapping segments as $S = \{s_1, s_2, ..., s_n\}$. Furthermore, the segments are divided into blocks that can be represented as $B = \{b_1, b_2, ..., b_{n-1}\}$. From that, the Contrastness Measure (CM) is determined. For computing CM, the Contrast Value Factor (CVF) is determined that can be formulated as,

$$CVF(l) = CM_{mw} + CM_{wD}, \quad l \in [1,2,...,n] \tag{11}$$

Where, CM_{mw} is the contrast measure with mean window, and CM_{wD} is the contrast measure with window deviation. Both are computed based on the non-overlapping segments. At last, the contrast score is determined by,

$$con_{sc} = \frac{1}{n}\sum_{l}^{n} CVF(l), \quad l \in [1,...,n] \tag{12}$$

From the Equation (12), the given image con_{sc} can be determined based on the probability value from high to low contrast. If the given image has lower contrast, then the proposed work utilized MSO algorithm to enhance the contrast based on equalizing the histogram. In our work, the less contrast pixels in the images are considered as moths (q_i) and their histogram is $F(q_i)$. For every iteration, the contrast of pixels is improved. The positions of the less contrast pixels are initialized as follows,

$$q_{iv} = Rnd[0,1] \ \cdot \ (q_v^{maxi} - q_v^{mini}) + q_v^{mini} \tag{13}$$

where, $i \in \{1,2,...,q\}, v \in \{1,2,...,d\}$ in which q denotes the pixel population, d is the problem dimension, and Rnd is the random value. Whereas, the q_v^{mini} and q_v^{maxi} are the lower and upper limits respectively. The objective function of moth swarm optimization based contrast enhancement is shown in Equation (12). From that, the probability of updation can be formulated as,

$$Pr_u = \frac{F(q_i)_u}{\sum_{u=1}^{qu} F(q_i)_u} \tag{14}$$

For optimizing the histogram of the pixels for reducing the luminance, the following conditions must be satisfied based on the contrast score (i.e., objective function) that can be formulated as,

$$F(q_i)_u = \begin{cases} \frac{1}{1+con_{sc}}, con_{sc} \geq 0 \\ 1 + |con_{sc}|, con_{sc} < 0 \end{cases} \tag{15}$$

The updation of low contrast pixels to high contrast pixels after contrast enhancement can be formulated as,

$$q_i^{j+1} = q_i^j + 0.001 \cdot \alpha[q_i^j - q_i^{mini}, q_i^{maxi} - q_i^j] + \\ (1 - \aleph/\alpha) \cdot Rnd_1 \cdot (bes_u^i - q_i^j) + 2\aleph/\alpha - Rnd_2 \cdot (bes_\aleph^i - q_i^j) \tag{16}$$

Where, Rnd_1 and Rnd_2 are the random numbers of interval [0,1]. $\aleph/\alpha$, and $2\aleph/\alpha$ are the environmental factors affecting the contrast enhancement. During the end of current iteration, the contrastness of the pixels are refined for next iteration which iterates until the desired solution had met.

4.1.3. Points to voxel conversion

The enhanced LiDAR 2D cloud points are converted into 3D voxels for improving the perception view of the object to increase the detection accuracy of the 3D objects. For converting the 2D LiDAR to 3D voxels, the maximum and minimum points are traversed in three dimensional directions (i.e., X, Y, and Z). The maximum traversed point is $(maxi_x, maxi_y, maxi_z)$, and the minimum traversed point is $(mini_x, mini_y, mini_z)$. The voxel grid can be computed by rounding operation based on the voxel size that can be formulated as,

$$\left\lceil \frac{maxi_x - mini_x}{VoxS} \right\rceil \cdot \left\lceil \frac{maxi_y - mini_y}{VoxS} \right\rceil \cdot \left\lceil \frac{maxi_z - mini_z}{VoxS} \right\rceil \tag{17}$$

From the obtained voxel gird, the 3D voxel grid coordinates for every point clouds can be determined as,

$$\begin{cases} Vox_{i.X} = \left\lceil \frac{Pt_i.X - mini_x}{VoxS} \right\rceil \\ Vox_{i.Y} = \left\lceil \frac{Pt_i.Y - mini_y}{VoxS} \right\rceil \quad Pt_i \in pointcloud \\ Vox_{i.Z} = \left\lceil \frac{Pt_i.Z - mini_z}{VoxS} \right\rceil \end{cases} \tag{18}$$

From the above equation, Pt_i denotes the i-th point in the 2D LiDAR point cloud. The voxelized image and the contrast enhanced image are fused to form high refined 3D image.

4.2. L-GAN based instance segmentation

The voxelized LiDAR cloud points and pre-processed RGB-D images are fused before performing segmentation for achieving efficient results in detection of 3D objects. The angle of the images is changed from one another which reduces the detection accuracy. To overcome this issue, we rotate the images in terms of several degrees such as 10°, 90°, 180°, and 270°. After rotating the images, instance segmentation

is performed for the fused images using Lightweight Generative Adversarial Network (L-GAN) algorithm which provides precise segmentation when compared with other state-of-the-art models in terms of having channel and position attention modules respectively (Ch_{att} & Po_{att}).

The generator (Gen) in the GAN is trained in the way of mapping function from input image to segmented image. The Gen composed of encoder and decoder architecture. The training of Gen is carried out using loss functions from discriminator (Dis) to Gen. For instance, the input object image is denoted as 'a' and the ground-truth image is denoted as 'b'. A random variable 'E' is introduced to reduce the overfitting at the decoder layer. Therefore, outputs of Gen and Dis can be represented as $Gen(a, E)$ and $Dis(a, Gen(a, E))$. With that, the generator loss function can be formulated as,

$$\begin{gathered} Gen_{loss}(Gen, Dis) = \mathbb{E}_{a,b,E}(-\log(Dis(a, Gen(a, E)))) \; + \\ \gamma\mathbb{E}_{a,b,E}(L1_{loss}(b, Gen(a, E))) \;\; + \;\; \varphi\mathbb{E}_{a,b,E}(jacc_{loss}(b, Gen(a, E))) \end{gathered} \tag{19}$$

where, γ and φ are the factor of weights. Our work considers three losses such as Jaccard loss, $L1$ loss, and adversarial loss. The reason for adopting three loss functions is that, as the adversarial loss might slow down the learning process so that $L1$ loss is utilized for preserving the object boundaries and Jaccard loss is utilized for improving the relationship among the original and segmented image.

On the other side, the discriminator Dis composed of four layers such as convolutional, position attention, channel attention, and activation layer respectively for robustly finds the generated images into real or fake. The loss function associated with the Dis can be formulated as,

$$\begin{gathered} Dis_{loss}(Gen, Dis) = \mathbb{E}_{a,b,E}(-log(Dis(a, Z))) + \\ \mathbb{E}_{a,b,E}(-log(1 - Dis(a, Gen(a, E)))) \end{gathered} \tag{20}$$

From the Dis_{loss}, the loss of binary entropy can be effectively determined by two mathematical terms such as $-\log(Dis(a, Z))$ (i.e., ground-truth image) and $-\log(1 - Dis(a, Gen(a, E)))$ (i.e., predicted image). The optimizer in the Dis performs minimization and maximization of loss function for predicted and ground truth images with classes of 0 and 1 respectively.

The attention modules in the encoder and decoder of Gen is utilized for learning both the high and low level features respectively. The Ch_{att} is utilized for learning the high level features by learning the feature interdependencies. From the features $\cup \in \mathbb{K}^{(C \cdot He \cdot Wi)}$, the Ch_{att} generates the channel attention map $\mathbb{X} \in \mathbb{K}^{C \cdot C}$ in which the C, He, and Wi denotes the channel, height, and width of the given image.

Utilizing softmax function, the $\mathbb{X} \in \mathbb{K}^{(C \cdot C)}$ is created as follows,

$$y_{ji} = \frac{exp(U_i.U_j)}{\sum_{i=1}^{C} exp(U_i.U_j)} \tag{21}$$

From the above equation, $U_i.U_j$ denotes the transpose of matrix multiplication, y_{ji} denotes the impact of i-th channel on j-th channel. The multiplicative results are reshaped to $\mathbb{K}^{(C \cdot He \cdot Wi)}$ that is again multiplied by χ (a scalar parameters).

After that, element wise addition is undergone to provide the output is $E \in \mathbb{K}^{(C \cdot He \cdot Wi)}$ as,

$$E_j = \chi \sum_{i=1}^{C} (y_{ji} U_i) + U_j \tag{22}$$

The final feature representation is the sum of weights of features of all channels which can provides the semantic dependencies and enhance the decimator functions.

Once, the important features are obtained from the Ch_{att}, the contextual information are obtained by the Po_{att}. In simple words, the Po_{att} encodes the contextual information to local features and represents them to local feature maps $\in \mathbb{K}^{(C \cdot He \cdot Wi)}$.

The feature maps are then provided to the consecutive convolutional layers for generating the other two feature maps that is represented as $(B, C) \in \mathbb{K}^{C \cdot He \cdot Wi}$.

The feature maps are then reshaped and fed to softmax layer for generating the spatial feature map that can be formulated as,

$$SP_{ji} = \frac{exp(B_i, C_j)}{\sum_{i=1}^{N} exp(B_i, C_j)} \tag{23}$$

Where, Sp_{ji} refers to j-th spatial position interaction on i-th position. The association among the feature maps is ensured by the softmax layer. For instance, the U is provided to the convolutional layers for generating a new feature map $D \in \mathbb{K}^{C \cdot N}$.

The output from the Po_{att} can be computed by multiplying the transpose of Sp_{ji} and D that can be formulated as:

$$E_j = \xi \sum_{i=1}^{N} (Sp_{ji} D_i) + U_j \tag{24}$$

Where, the scalar constraint is represented as ξ. The Po_{att} output is sum of weight of neighbor features which represents the context information of local features through spatial map representation.

Figure 2 represents the diagrammatic view of L-GAN based instance segmentation of objects.

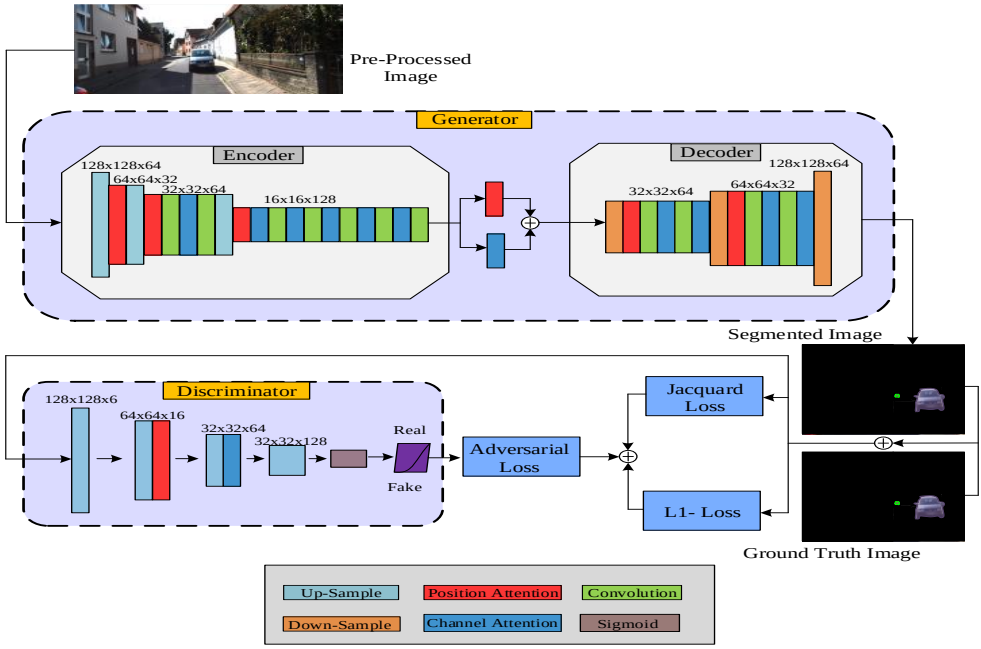


Figure 2. L-GAN based Instance Segmentation

4.3. 3D object detection and tracking

After segmenting the images, we perform feature extraction from the segmented region for classification of objects. Feature extraction and classification are performed using Hybrid Deep Learning algorithm which consists of YOLOv4 and VGG16 algorithms. YOLOv4 is mainly implemented to increase the classification speed and detect small objects. VGG16 is implemented to increase detection accuracy. In this proposed, we extract numerous features such as spatial, temporal, textural, visual, and auditory features using VGG16, and classification is performed using YOLOv4 which provides four classes such as ground, vehicles, pedestrians, and obstacles. Figure 3 represents the process of 3D object detection.

4.3.1. Feature extraction-VGG 16

The segmented image from the LGAN of input size 224×224 is provided with R, G, and B channels. The input size of the image is reduced for every pixel for achieving the desired results. Once, the images are passed over the ReLU activations, the resultant image is provided to the stack of two consecutive convolutional layers with area size of 3×3 and have 64 filters. The image is processed at 1 pixel padding and convolutional stride is also at 1 pixel. The two consecutive layer preserves the spatial resolution with pooling of two pixel of window size 2×2, so that the activation window size is reduced to half.

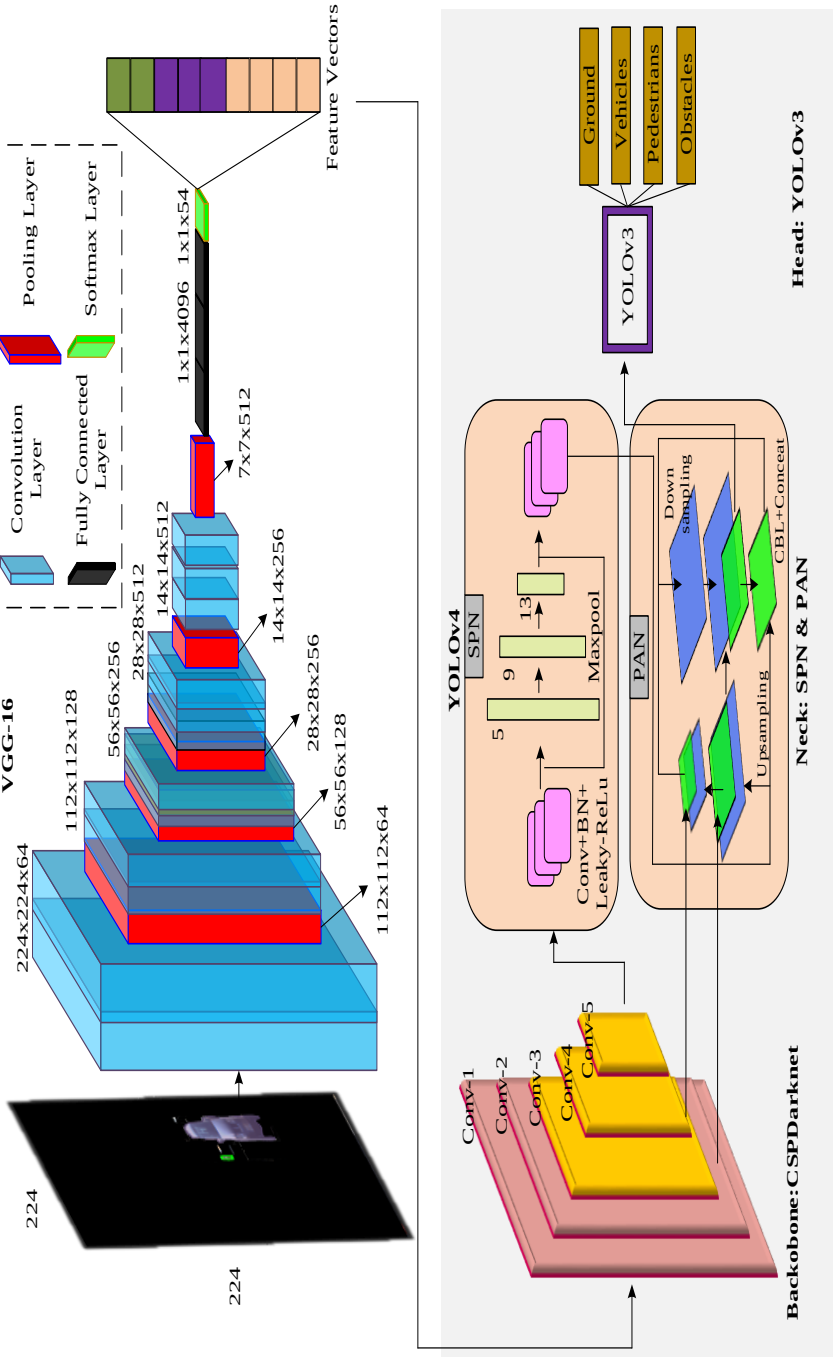


Figure 3. VGG-16 & YOLOv4 based Feature Extraction and Classification

The output activation function from the first stack of convolution is then provided to second stack of activation and convolutional layers in which the activation layer 128 filters with size of $56 \times 56 \times 128$, and three convolutional layers has 256 filter with size of $56 \times 56 \times 256$. In similar manner, the filter in the consecutive three layers is increased to 512 and convolution layer size is reduced to of $28 \times 28 \times 512$, $14 \times 14 \times 512$, and $7 \times 7 \times 512$. The output images from the final max pooling layer are then provided to the 3 fully connected layer in which the first two fully connected layers composed of 4096 channels and last one layer has 1000 channels. Finally, the softmax layer provides the feature vectors of the segmented images that includes feature representation vectors of features such as spatial, temporal, textural, visual, and auditory features which can be represented as,

$$Fe = \begin{bmatrix} Fe_1 \\ Fe_2 \\ Fe_3 \\ Fe_4 \\ Fe_5 \end{bmatrix} \tag{25}$$

where, Fe_1, Fe_2, Fe_3, Fe_4 and Fe_5 represents the features such as spatial, temporal, textural, visual, and auditory respectively.

4.3.2. Object classification – YOLOv4

The extracted feature vectors are then provided to the YOLOv4. The YOLOv4 is developed by improving the YOLOv3 form improving the speed, detailed, and stable results. The YOLOv4 composed of backbone network, neck, and head for processing the input features and provides the desirable output. To be clearer, the proposed work used backbone network as Cross Scale Partial Darknet-53 (CSPDarknet-53), the neck structure used are Spatial Pyramidal Pooling (SPN) and Path Aggregation Network (PAN), the head structure has YOLOv3 for enabling speedy object classification.

The extracted feature vectors are provided to the CSPDarknet-53 for representing the deep features using five ResNet blocks. The ResNet blocks composed of fifty-three convolutional layers of sizes 3×3 and 1×1 with connection to batch normalization, and mesh activation layer respectively. For reducing the computation complexity, the conventional ReLU is substituted with the leaky ReLU. The represented features from the CSPDarknet-53 are then provided to the neck that consist of SPN and PAN. The SPN composed of several maxpooling layers of different sizes such as 13, 9, and 5 to normalize the features sizes through cross minibatch normalization. The normalizes features are provided to the PAN for continuously extracting the features in repeated fashion in top down and bottom down approach. The extracted deep features, are finally provided to the YOLO head. The proposed YOLO head utilized YOLOv3 for object detection with the size of 76×76 to detect and classify the object of varying sizes.

The classification result can be represented as:

$$YOLOv4_{Head} = \begin{cases} Ground \\ Vehicles \\ Pedestrains \\ Obstacles \end{cases} \tag{26}$$

Finally, loss of the YOLOv4 is computed which consist of three losses such as object classification, localization, and offset loss respectively. The formulation of loss function is computed below,

$$L^{oss} = \Xi_1 L^{cl} + \Xi_2 L^{loc} + \Xi_3 L^{con} \tag{27}$$

Where, L^{cl}, L^{loc}, and L^{con} states the classification, localization, and confidence loss respectively. The Ξ_1, Ξ_2, and Ξ_3 are the balancing factors of the respective loss functions. The formulation of individual loss functions can be formulated as,

$$L^{cl} = -\Sigma_{i \in box} \Sigma_{j \in class} (ob_{ij} ln(pr_{ij}) + (1 - ob_{ij}) ln(1 - pr_{ij}) \tag{28}$$

$$L^{loc} = 1 - InOU(Pre, GnT) + \frac{d^2_{Pre,GnT}(Pre_{cen}, GnT_{cen})}{\iota^2} + \eth \tag{29}$$

$$L^{con} = -\Sigma(ob_i ln(pr_i) + (1 - ob_i) ln(1 - pr_i)) \tag{30}$$

From the Equation (28), pr_{ij} and ob_{ij} represents the i-th object class in the boundary box prediction i. From Equation (29), InOU is the intersection over union, Pre, GnT denotes the predicted and ground truth results respectively, Pre_{cen}, GnT_{cen} defines the center point euclidean distance, and $\eth$ denotes the facet ratio. From the Equation (30), ob_i represents whether there is any object in the bounding box [0,1], and pr_{ij} probability of the object in the bounding box.

4.3.3. Object tracking-IKF

After classification of objects, tracking is performed only for moving objects by considering RFID, unique ID, dimension, and orientation using Improved Unscented Kalman Filter (IUKF) which reduces the variance and tracks the objects with high accuracy by considering the object's velocity and location. Time-based mapping is performed by considering previous and current time, location from the RFID to increase the tracking reliability.

At a 3D plane, let us consider the detected object at the previous stage is moving at a uniform speed. The state of the moving object is denoted as $d[u]$ with time u. The position of the object in 3D plane time u is denoted as $pos_x[u]$, $pos_y[u]$, $pos_z[u]$. The detected bound box aspect ratio asp[u], height hei[u], object velocity

$vel_x[u], vel_y[u], vel_z[u]$, location of the object loc[u], and its unique id $obj[ID_u]$. The complete details are denoted as,

$$\begin{aligned} d[u] = (&pos_x[u], pos_y[u], pos_z[u], vel_x[u], vel_y[u], vel_z[u], \\ &asp[u], hei[u], loc[u], obj[ID_u])^T \in \wedge^{10} \end{aligned} \tag{31}$$

The state of an object at time $u+1$ can be formulated as,

$$d[u+1] = Fd[u] + Ng\Psi[u] \tag{32}$$

Where, F denotes the transfer matrix for the previous object state, Ng is the noise matrix, and $\Psi[u]$ represents the noise vector of a system at time u. Based on the mentioned object motion model, the IUKF algorithm is utilized for object state tracking. At first, the sigma points group are constructed as,

$$\Gamma_i = \begin{cases} \bar{d}[u] + (\sqrt{(dim_{sv} + \Upsilon)err[u]}, i = 1, 2, ..., L \\ \bar{d}[u] - (\sqrt{(dim_{sv} + \Upsilon)err[u]}, i = L+1, ..., 2L \\ \bar{d}[u], \qquad\qquad\qquad\qquad i = 0 \end{cases} \tag{33}$$

Where, $err[u]$ represents the covariance error matrix, dim_{sv} is the state vector dimension, and Υ is the distance parameter of sigma points. The sigma points are substituted to the non-linear equation that can be formulated as,

$$y_i = h(\Gamma_i), i = 0, 1, ..., 2dim_{sv} \tag{34}$$

From the y, mean and variance are computed that can be formulated as follows,

$$\bar{y} \approx \Sigma_{i=0}^{2dim_{sv}} W_i^{(m)} y_i \tag{35}$$

$$err_{\mathfrak{v}} \approx \Sigma_{i=0}^{2dim_{sv}} W_i^{(c)} (y_i - \bar{y})(y_i - \bar{y})^T \tag{36}$$

The computation of $W_i^{(m)}$, and $W_i^{(c)}$ is provided as follows,

$$W_0^{(m)} = u/(dim_{sv} + u) \tag{37}$$

$$W_0^{(c)} = u/(dim_{sv} + u) + (1 + \tau^2) \tag{38}$$

$$W_i^{(m)} = W_i^{(c)} = u/[2(dim_{sv} + u)], i = 1, ..., 2dim_{sv} \tag{39}$$

In order to regularize the UKF, the correctness factor Γ^* is introduced to improve the UKF. The adoption of Γ^* reduces the chance of wrong measurements during object tracking. The formulation of Γ^* in the proposed IUKF can be provided as,

$$\Gamma^* = \bar{d}[u] + u(y_u - \hat{y}_{\bar{u}}) \tag{40}$$

Once the correction is completed, state of the object can be formulated as,

$$f = min(\bar{y}_{u+1} - \bar{y}_{u+1})^T (\bar{y}_{u+1} - \bar{y}_{u+1}), ob_{lm} \leq \tau \leq ob_{up} \tag{41}$$

where, ob_{lm} is the lower limit of the moving object, and ob_{up} is the upper limit of the moving object. The tracking continues until the maximum number of steps $(u+1)$. The pseudocode denotes the IUKF based 3D object tracking.

Algorithm 2 Pseudocode for IUKF Object Tracking

Input: Detected Object with Bounding Box
Output: Object Tracking
Begin
Initialize the object tracking model (31)
Formulate the object consecutive states (32)
//IUKF based Object Tracking//
for all detected objects **do**
    Construct the sigma points Γ_i (33)
    Compute the $y_i \rightarrow h(.)$ (34)
    Compute mean and variance (35)–(36)
    Compute $W_i^{(m)}$ and $W_i^{(c)}$ (37)–(38)
    Regularize $UKF \rightarrow \Gamma^*$ (40)
    Obtain object state (41)
    Track until (u+1)
end for
End

Pseudocode explanation

Step 1: The Object detected with bounding boxes will be given as input.

Step 2: Initialize the object tracking model in 3D Plane with the complete details of Position pos, velocity vel, aspect ratio asp, height hei, location loc, unique id obj[ID].

Step 3: The State of an object with time $\mathrm{u}+1$ will be formulated with noise matrix Ng, transfer matrix from the previous state F, noise vector $\Psi[u]$ at time u.

Step 4: After all the objects has been detected, construct the sigma points group with error matrix err[u], dimension and the distance.

Step 5: Substitute the values of sigma points to non-linear equation Y_i.

Step 6: From Y_i, Calculate mean and variance $\bar{y}$ and err_y

Step 7: Compute the value of $W_i^{(m)}$, and $W_i^{(c)}$, which is calculated during the findings of mean and variance.

Step 8: The correctness factor Γ^* is given to reduce the chance of wrong measurements during tracking.

Step 9: After Correction, state of object can be formulated.

Step 10: Tracking continues until the maximum number of steps reached $(\mathrm{u}+1)$.

5. Experimental results

This section provides the detailed view of simulation, implementation, and comparative results. For diminishing the readers difficulty, separate sections are provided for simulation and implementation results, dataset description, comparative results, and summary.

5.1. Simulation setup

The proposed Hybrid Deep Learning based Multi Object Detection and Tracking (HDL-MODT) is simulated using MATLAB tool of version R2020a. The simulation results show that, the proposed work outperforms than the existing work. The proposed simulation is packaged with pre-processing, segmentation, classification, and tracking. To achieve better performance, some of the system configurations must be adjusted. The adjusted system configurations are mentioned in Table 2. Furthermore, the simulation results of the proposed work also provided in the Figure 4a–4d.

Table 2
System configurations

Hardware Configuration	RAM	500GB
	Hard Disk	8GB
Software Configuration	Simulation Tool	MATLAB R2020a
	OS	Windows-10(64-bit) OS
	Processor	Intel(R) Core(TM) i5-4590S CPU@3.00GHz

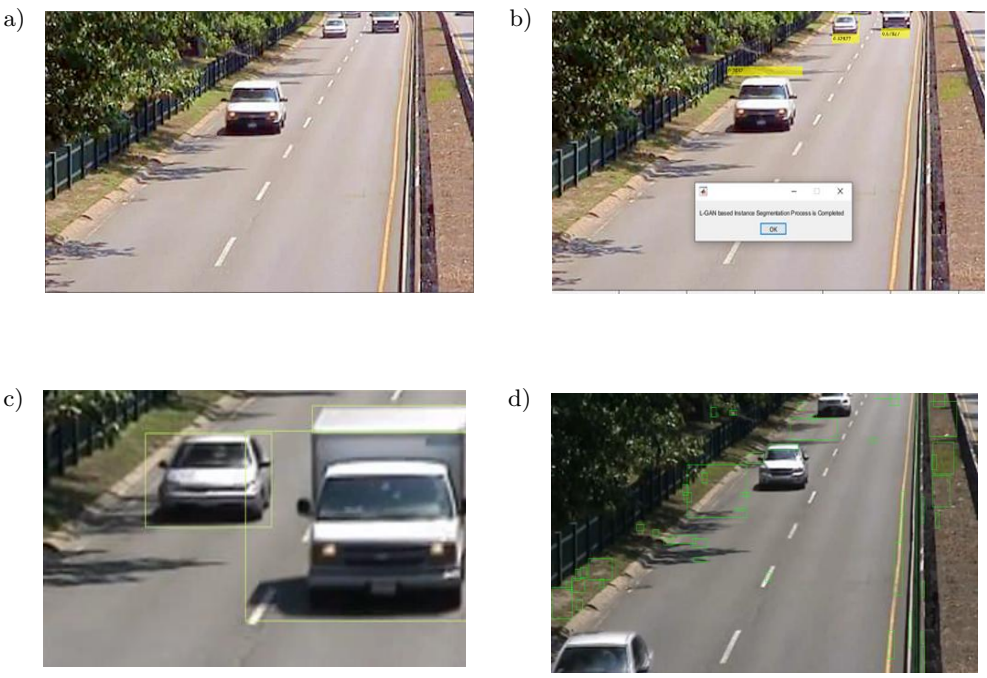


Figure 4. Multi Stage Noise Removal (a); L-GAN based Instance Segmentation (b); Moving Object Detection & Tracking (c); Static Object Detection (d)

5.2. Dataset description

The performance of the proposed work is evaluated by performing quantitative and qualitative experiments using Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) dataset. This dataset [11] holds the images from 3D LiDAR and stereo RGB camera from the autonomous vehicles. The capturing of LiDAR frames by using an LiDAR sensor named HDL-64E. The points generated by the sensor is one million. The frames provided for testing and training is 7518 and 7481 respectively. The proposed work divides the dataset as 25% for validation and 75% for training. The dataset contains 52,979 labels with nine categories such as 'don't care objects', 'miscellaneous', 'tram', 'sitting person', 'truck', 'van', 'cyclist', 'pedestrian', and 'car'.

5.3. Comparative analysis

This sub-section explains the comparative results proposed HDL-MODT with existing works such as PointTrackNet [26], and STR-ODT [12] respectively. The validation metrics taken such as accuracy, precision, recall, f-score, and computation. The brief explanation of the proposed comparative results are defined below.

5.3.1. Accuracy comparison

Accuracy is defined as the sum of True Positive (TP) and True Negative (TN) to the ratio of sum of TP, TN, False Positive (FP), and False Negative (FN) respectively. The formulation of accuracy is,

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{42}$$

Figure 5 represents the comparison of accuracy of proposed and existing works with respect to number of frames.

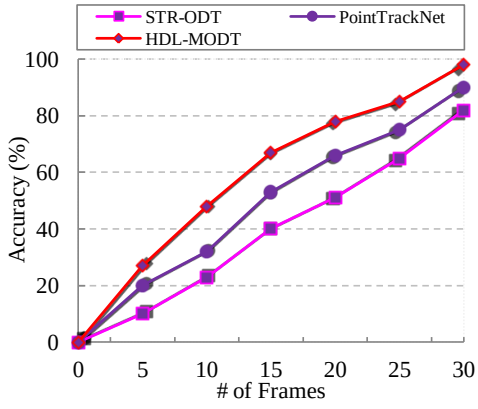


Figure 5. Number of Frames vs Accuracy

From the inference, it is shown that when the number of frames increases the accuracy rate also increases. The reason for such higher increment in accuracy is that, the proposed work utilized hybrid deep learning algorithm named VGG-16 and YOLOv4 for feature extraction and classification respectively. On contrary, the existing work PointTrackNet lacks with extracting optimal features and poor classifier for object detection leads to less accuracy. On the whole, the graphical inference shows that, our proposed work achieves higher accuracy than the existing works.

The numerical results show that, the proposed work achieves higher accuracy of 98% when the frames increased to 30 whereas the existing works PointTrackNet and STR-ODT achieves lesser accuracy of 90% and 82% respectively. Overall, the proposed work achieves higher accuracy of 8–16% than the existing works.

5.3.2. Precision comparison

The precision is defined as the ratio of TP to the sum of TP and FP respectively. In other words, its also define how precisely the proposed work classifies and tracks the objects. The formulation of precision is provided as below,

$$Pre = \frac{TP}{TP + FP} \tag{43}$$

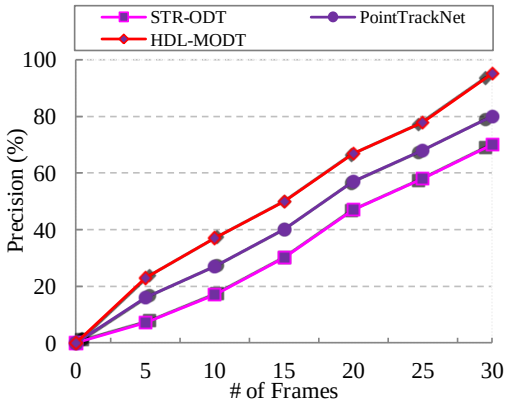


Figure 6. Number of frames vs precision

Figure 6 shows the comparison precision with proposed and existing in terms of number of frames. The precision rate increased with increase in number of frames. Among that our proposed work achieves higher precision than the existing works. The reason for such higher precision rate is that, the proposed work performs improved deep learning based segmentation named LGAN based instance segmentation, and IUKF based object tracking. In IUKF based objects, the detected moving objects are tracked by considering metrics such as velocity, location, RFID, dimension, and

unique ID. Furthermore, the proposed work also utilized time-based mapping to precisely track down the objects. The existing work lacks with less precision rate, as they were not performing segmentation which increase the higher false positive rates. In addition to that, the existing object tracking feature was not plausible that also affects the precision in object tracking.

The numerical results show that, the proposed work achieves higher precision of 95% when the frames increased to 30 whereas the existing works PointTrackNet and STR-ODT achieves lesser precision of 80% and 70% respectively. Overall, the proposed work achieves higher precision of 5–25% than the existing works.

5.3.3. Recall comparison

The recall rate is defined as the ratio of TP to the sum of TP and FN respectively. The proposed work defines the recall rate by computing the amount of positively detected samples. The formulation of proposed recall rate is as follows,

$$Rec = \frac{TP}{TP + FN} \tag{44}$$

The comparison of recall rate with respect to number of frames for the proposed and existing works is shown in Figure 7. The figure shows that, the recall rate increases with increase in frame rate. The major reason for such higher recall rate is that, the proposed work performs multi stage pre-processing method and thereby the rate of correctly classifying the samples is increased. The proposed pre-processing method firmly increases the performance of accuracy and precision respectively. The existing works PointTrackNet and STR-ODT limits with pre-processing of acquired images thereby they achieve deprived performance on further processes. Hence, the probability of positively classifying the samples is less in the existing works.

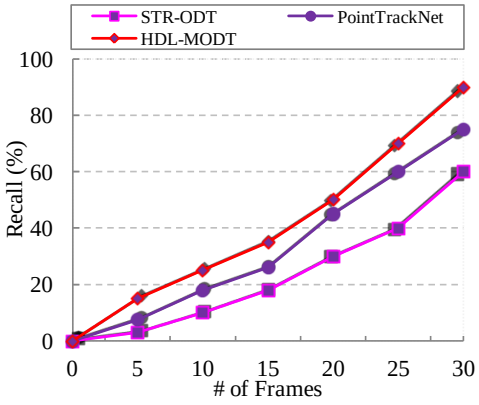


Figure 7. Number of frames vs Recall

The numerical results show that, the proposed work achieves higher recall rate of 90% when the frames increased to 30 whereas the existing works PointTrackNet and STR-ODT achieves lesser recall rate of 75% and 60% respectively. Overall, the proposed work achieves higher recall rate of 20–30% than the existing works.

5.3.4. F-Score comparison

The *F-Score* is defined as the harmonic mean of precision and recall rates respectively. To be clear, the mathematical illustration of *F-Score* can be formulated as,

$$F\text{-}Score = 2 \cdot \frac{Pre \cdot Rec}{Pre + Rec} \tag{45}$$

The comparison of *F-Score* rate of proposed and existing works with respect to number of frames is shown in Figure 8. The figure shows that when the number of frames increases the *F-Score* rate also increases. From which the proposed work achieves higher *F-Score* rate than the existing works. As the proposed work performs effective pre-processing, and segmentation respectively. The proposed work adopts L-GAN for segmenting the objects in which it performs instance segmentation to merely classify the objects to reduce the unwanted discrepancies during classification thereby improving the *F-Score* rate. In contrast, the existing works PointTrackNet and STR-ODT achieves less *F-Score* rate as they lack with pre-processing by directly provides the images for further process, and also performs ineffective segmentation which reduced the *F-Score* rate.

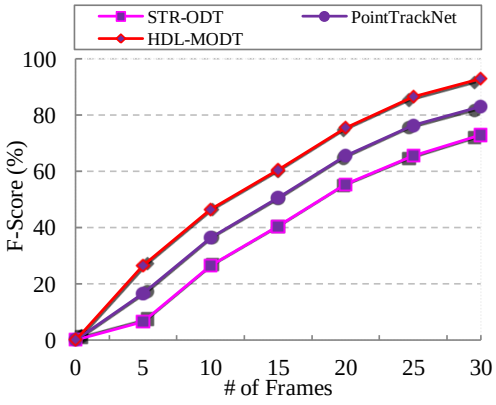


Figure 8. Number of frames vs *F-Score*

The numerical results show that, the proposed work achieves higher *F-Score* rate of 93% when the frames increased to 30 whereas the existing works PointTrackNet and STR-ODT achieves lesser *F-Score* rate of 83% and 73% respectively. Overall, the proposed work achieves higher *F-Score* rate of 10–20% than the existing works.

5.3.5. Computation time

The computation time is defined as the amount of time taken to complete as process. The mathematical formulation of computation time is defined as the ratio of overall computation time to the time taken for computation,

$$CT = \frac{CT_{Time}}{Ov_{Time}} \tag{46}$$

where, CT_{Time} is the computation time taken, and Ov_{Time} is the overall computation time.

Figure 9 shows the comparison of computation time of proposed and existing works with respect to number of frames respectively. From the graphical inference, the computation time of proposed work decreases with increase in number of frames. The reason for such less computation time is that, the proposed work adopts multi stage pre-processing and hybrid deep learning based object detection respectively. The object detection is performed using hybrid deep learning algorithm named VGG-16 and YOLOv4 respectively. The proposed process reduces those complexity by increasing the computation time. On the other hand, the existing works PointTrackNet and STR-ODT gains with higher computation time as they lack with effective preprocessing and classification respectively thereby time for computation was high.

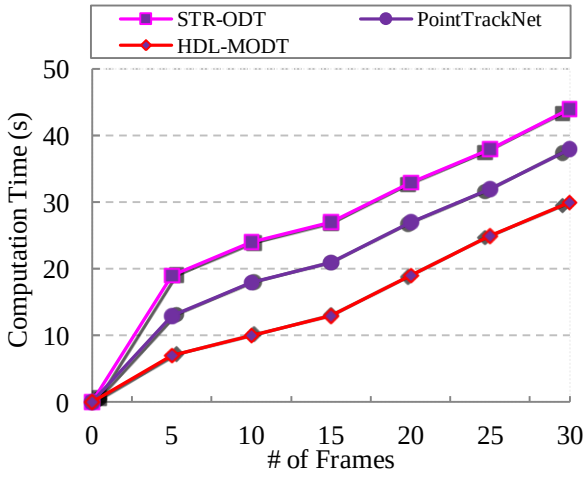


Figure 9. Number of frames vs computation time

The numerical results show that, the proposed work achieves lesser computation time of 30 s when the frames increased to 30 whereas the existing works PointTrackNet and STR-ODT achieves higher computation time of 38 s and 44 s respectively. Overall, the proposed work achieves lesser computation time of 8 s–14 s than the existing works.

5.4. Research summary

The summary of the experimental results section is provided in this supplementary section. The proposed work composed of sequential processes such as multi stage pre-processing, instance segmentation, and 3D object detection & tracking. The simulation of proposed work is carried out using MATLAB R2020a in which the simulation results are provided in Figures 4a–d. The comparative results of the proposed work and existing works in various simulation metrics are shown in Figures 5–9. The average simulation results comparison of proposed and existing also shown in Table 3 Some of the major highlights of the proposed work are given below:

- For enhancing the quality of the images (i.e., solid-state LiDAR, pseudo-LiDAR, and RGB-D), we perform multi-stage preprocessing in which noise is removed by A-Fuzzy filter and contrast enhancement using MSO algorithm. In addition, point to voxel conversion is performed for achieving efficient object detection results.
- For improving the viewport of the image by performing image rotation and instance segmentation. The L-GAN algorithm is implemented to perform instance segmentation that maximizes the accuracy of object detection which increases the reliability of tracking.
- For increasing the accuracy and speed of 3D object detection, we perform multiple feature extraction and classification by Hybrid deep learning algorithm which detects the objects with low false positive rate and high speed.
- For increasing the tracking reliability, we implement IUKF filter by considering numerous metrics and performing time-based mapping which increases the reliability of tracking with high accuracy.

Table 3
Average comparison of proposed vs existing

Metrics	HDL-MODT	PointTrackNet	STR-ODT
Accuracy [%]	57.57	48	38.72
Precision [%]	50	41.142	32.71
Recall [%]	40.71	33.072	23
F-Score [%]	55.5	46.93	38.22
Computation Time [s]	14.85	21.28	26.42

6. Conclusion

High false positive rates, high computation time, and less QoS are the major issues in the 3D object detection and localization. So that, we tend to resolve that issue by proposing HDL-MODT method. The proposed work adopts KITTI dataset for training and testing the classifiers. Initially, the images captured from the RGB-D cameras

and Solid-State LiDAR are pre-processed in multi stages. The proposed work performs three stages of pre-processing such as noise removal using A-Fuzzy, contrast enhancement using MSO, and point to voxel conversion respectively. The pre-processed image is fused to improve the image quality. Secondly, the fused image is provided for instance segmentation using L-GAN in which position and channel attention are adopted for segmenting the possible objects in the input images. The fused images are then provided for object detection, classification, and tracking. The VGG-16 is utilized for feature extraction which extracts the optimal features such as spatial, temporal, textural, visual, and auditory features. The extracted features are represented in form of feature vectors. The feature vectors are provided as an input to the YOLOv4 classifier for object detection and classification task which classifies the objects into four classes such as ground, vehicles, pedestrians, and obstacles and tow categories as static and moving objects. For the moving objects, we perform tracking using IUKF algorithm based on metrics such as RFID, unique ID, location, dimension, and velocity. The time based mapping is also performed to enhance the tracking accuracy. The simulation of proposed work is carried out using MATLAB R2020a simulation tool and performance of the proposed work is validated by considering metrics such as accuracy, precision, recall, F-score, and computation time.

Acknowledgements

The authors thank the reviewer(s) for their scholarly comments and suggestions. The authors also express their gratitude to the Editor-in-Chief (Jacek Kitowski), the Editor, and the Editorial Office Assistant(s) of this journal for managing this manuscript.

References

[1] Bai J., Li S., Huang L., Chen H.: Robust detection and tracking method for moving object based on radar and camera data fusion, *IEEE Sensors Journal*, vol. 21(9), pp. 10761–10774, 2021. doi: 10.1109/jsen.2021.3049449.

[2] Bashar M., Islam S., Hussain K.K., Hasan M.B., Ashikur Rahman A.B.M., Kabir M.H.: Multiple object tracking in recent times: A literature review, *arXiv preprint arXiv:220904796*, 2022. doi: 10.48550/arXiv.2209.04796.

[3] Bescos B., Campos C., Tardós J.D., Neira J.: DynaSLAM II: Tightly-coupled multi-object tracking and SLAM, *IEEE Robotics and Automation Letters*, vol. 6(3), pp. 5191–5198, 2021. doi: 10.1109/lra.2021.3068640.

[4] Chiu H.K., Li J., Ambruş R., Bohg J.: Probabilistic 3D multi-modal, multi-object tracking for autonomous driving. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14227–14233, IEEE, 2021. doi: 10.1109/icra48506.2021.9561754.

[5] Choi H., Jeong J., Choi J.Y.: Rotation-Aware 3D Vehicle Detection from Point Cloud, *IEEE Access*, vol. 9, pp. 99276–99286, 2021. doi: 10.1109/access.2021.3095525.

[6] Fan Y.C., Yelamandala C.M., Chen T.W., Huang C.J.: Real-Time Object Detection for LiDAR Based on LS-R-YOLOv4 Neural Network, *Journal of Sensors*, vol. 2021, pp. 1–11, 2021. doi: 10.1155/2021/5576262.

[7] Farag W.: Kalman-filter-based sensor fusion applied to road-objects detection and tracking for autonomous vehicles, *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 235(7), pp. 1125–1138, 2021. doi: 10.1177/0959651820975523.

[8] Huang C., He T., Ren H., Wang W., Lin B., Cai D.: OBMO: One bounding box multiple objects for monocular 3D object detection, *IEEE Transactions on Image Processing*, vol. 32, pp. 6570–6581, 2023. doi: 10.1109/tip.2023.3333225.

[9] Jiang P., Ergu D., Liu F., Cai Y., Ma B.: A Review of Yolo algorithm developments, *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022. doi: 10.1016/j.procs.2022.01.135.

[10] Kim A., Ošep A., Leal-Taixé L.: EagerMOT: 3D Multi-Object Tracking via Sensor Fusion, *CoRR*, vol. abs/2104.146822104.14682, 2021. doi: 10.1109/icra48506.2021.9562072. 2104.14682.

[11] KITTI DataSet, https://universe.roboflow.com/sebastian-krauss/kitti-9amcz/DATASET/2.

[12] Koh J., Kim J., Yoo J.H., Kim Y., Kum D., Choi J.W.: Joint 3D object detection and tracking using spatio-temporal representation of camera image and LiDAR point clouds. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 1210–1218, 2022. doi: 10.1609/aaai.v36i1.20007.

[13] Lee E., Nam M., Lee H.: Tab2vox: CNN-based multivariate multilevel demand forecasting framework by tabular-to-voxel image conversion, *Sustainability*, vol. 14(18), 11745, 2022. doi: 10.3390/su141811745.

[14] Liu Z., Cai Y., Wang H., Chen L., Gao H., Jia Y., Li Y.: Robust target recognition and tracking of self-driving cars with radar and camera information fusion under severe weather conditions, *IEEE Transactions on Intelligent Transportation Systems*, vol. 23(7), pp. 6640–6653, 2021. doi: 10.1109/tits.2021.3059674.

[15] Luo C., Yang X., Yuille A.: Exploring simple 3D multi-object tracking for autonomous driving. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10488–10497, 2021. doi: 10.1109/iccv48922.2021.01032.

[16] Nabati R., Harris L., Qi H.: CFTrack: Center-based radar and camera fusion for 3D multi-object tracking. In: *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, pp. 243–248, IEEE, 2021. doi: 10.1109/ivworkshops54471.2021.9669223.

[17] Pal S.K., Pramanik A., Maiti J., Mitra P.: Deep learning in multi-object detection and tracking: state of the art, *Applied Intelligence*, vol. 51, pp. 6400–6429, 2021. doi: 10.1007/s10489-021-02293-7.

[18] Pang Z., Li Z., Wang N.: SimpleTrack: Understanding and rethinking 3D multi-object tracking. In: L. Karlinsky, T. Michaeli, K. Nishino (eds.), *Computer Vision – ECCV 2022 Workshops. Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*, pp. 680–696, Springer, 2022. doi: 10.1007/978-3-031-25056-9_43.

[19] Park D., Ambruş R., Guizilini V., Li J., Gaidon A.: Is pseudo-lidar needed for monocular 3D object detection? In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3142–3152, 2021. doi: 10.1109/iccv48922.2021.00313.

[20] Premachandra C., Ueda S., Suzuki Y.: Detection and tracking of moving objects at road intersections using a 360-degree camera for driver assistance and automated driving, *IEEE Access*, vol. 8, pp. 135652–135660, 2020. doi: 10.1109/access.2020.3011430.

[21] Qian R., Lai X., Li X.: 3D object detection for autonomous driving: A survey, *Pattern Recognition*, vol. 130, 108796, 2022. doi: 10.1016/j.patcog.2022.108796.

[22] Shreyas E., Sheth M.H., Mohana: 3D object detection and tracking methods using deep learning for computer vision applications. In: *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, pp. 735–738, IEEE, 2021. doi: 10.1109/rteict52294.2021.9573964.

[23] Simonelli A., Bulò S.R., Porzi L., Kontschieder P., Ricci E.: Are we Missing Confidence in Pseudo-LiDAR Methods for Monocular 3D Object Detection? In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3205–3213, 2021. doi: 10.1109/iccv48922.2021.00321.

[24] Wang B., Zhu M., Lu Y., Wang J., Gao W., Wei H.: Real-time 3D object detection from point cloud through foreground segmentation, *IEEE Access*, vol. 9, pp. 84886–84898, 2021. doi: 10.1109/access.2021.3087179.

[25] Wang K., Liu M.: YOLOv3-MT: A YOLOv3 using multi-target tracking for vehicle visual detection, *Applied Intelligence*, vol. 52(2), pp. 2070–2091, 2022. doi: 10.1007/s10489-021-02491-3.

[26] Wang S., Sun Y., Liu C., Liu M.: PointTrackNet: An End-to-End Network for 3-D Object Detection and Tracking From Point Clouds, *IEEE Robotics and Automation Letters*, vol. 5(2), pp. 3206–3212, 2020. doi: 10.1109/lra.2020.2974392.

[27] Wang Y., Guizilini V.C., Zhang T., Wang Y., Zhao H., Solomon J.: DETR3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries. In: A. Faust, D. Hsu, G. Neumann (eds.), *Conference on Robot Learning, 8–11 November 2021, London, UK*, Proceedings of Machine Learning Research, vol. 164, pp. 180–191, PMLR, 2022. https://proceedings.mlr.press/v164/wang22b.html.

[28] Wang Y., Wang C., Long P., Gu Y., Li W.: Recent advances in 3D object detection based on RGB-D: A survey, *Displays*, vol. 70, 102077, 2021. doi: 10.1016/j.displa.2021.102077.

[29] Wang Y., Yang B., Hu R., Liang M., Urtasun R.: PLUMENet: Efficient 3D object detection from stereo images. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3383–3390, IEEE, 2021. doi: 10.1109/iros51168.2021.9635875.

[30] Wen L.H., Jo K.H.: Fast and accurate 3D object detection for lidar-camera-based autonomous vehicles using one shared voxel-based backbone, *IEEE Access*, vol. 9, pp. 22080–22089, 2021. doi: 10.1109/access.2021.3055491.

[31] Xie X., Cheng G., Wang J., Yao X., Han J.: Oriented R-CNN for object detection. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3520–3529, 2021. doi: 10.1109/iccv48922.2021.00350.

[32] Zhao X., Sun P., Xu Z., Min H., Yu H.: Fusion of 3D LIDAR and camera data for object detection in autonomous vehicle applications, *IEEE Sensors Journal*, vol. 20(9), pp. 4901–4913, 2020. doi: 10.1109/jsen.2020.2966034.

Affiliations

Dheepika PS
Nehru Memorial College (Affiliated to Bharathidasan University), Department of Computer Science, Tiruchirapalli 621007, India, psdheepika@gmail.com

Umadevi V
Nehru Memorial College (Affiliated to Bharathidasan University), Department of Computer Science, Tiruchirapalli 621007, India, yazh1999@gmail.com

Received: 07.07.2023
Revised: 26.04.2024
Accepted: 26.04.2024

Computer Science • 25(3) 2024 https://doi.org/10.7494/csci.2024.25.3.5651

Heba F. Eid
Erik Cuevas

ENHANCED BONOBO OPTIMIZER FOR OPTIMIZING DYNAMIC PHOTOVOLTAIC MODELS

Abstract *Bonobo optimizer (BO) is a novel metaheuristic algorithm motivated by the social behaviour of the bonobos. This paper presents a quantum behaved bonobo optimization algorithm (QBOA) employing an innovative metaheuristic based on the reproductive strategies and social behavior of bonobos. Whereby, the quantum mechanics are embedded into the bonobo optimizer to direct the search agents through the search space. Accordingly, under this quantum-behaved movement, the proposed QBOA's exploitation capability is promoted. The performance of the proposed QBOA is exhibited on CEC2005 and CEC2019 benchmarks. Moreover, the QBOA algorithm was adapted to optimize the dynamic photovoltaic models parameters. QBOA exhibits the efficiency and adequacy to solve various optimization problems based on experimental and comparison findings, as well as its ability to implement competitive and promising results optimizing dynamic photovoltaic models.*

Keywords bonobo optimization algorithm, quantum behaved mechanism, photovoltaic model, dynamic photovoltaic model, parameters estimation

Citation Computer Science 25(3) 2024: 469–493

1. Introduction

Meta-heuristics algorithms have been effectively employed to solve a variety of real-world optimization problems, due to their reliable, robust and computationally efficient in avoiding local minima [11]. Various well-known meta-heuristics algorithms that have been proposed in the literature are Particle swarms [10], Coyote Optimization Algorithm (COA) [24], Dragonfly Algorithm (DA) [21], Particle Swarm Optimization (PSO) [10], Gravitational Search Algorithm (GSA) [27], Moth-Flame Optimization (MFO) [20], Genetic algorithm (GA) [17]. The bonobo optimizer (BO) is a novel meta-heuristic algorithm that is modeled on bonobo social behaviour [6]. The BO's search capabilities was made more resilient and efficient by using a fission-fusion approach for selection, four well-developed methods for producing offspring, and the application of two distinctive phases.

Despite the fact that population based approaches can grant promising solutions for optimization problems, as the dimension of the search space grows, they experience a series of challenges. A major key challenges is that, population based approaches frequently become stuck in local optimum when dealing with multi-modal complicated problems [3]. Correspondingly,to achieve high performance on complicated optimization problems, the exploration and exploitation stages should be well balanced [12].

The photovoltaic (PV) solar model presents one of the most exciting themes that has led to an increase in researchers' interest [13, 28]. However, one of the key challenges for researchers is to insure that the PV model captures the maximum amount of available power [4]. Different PV solar models have been presented including the static and dynamic PV models. Nevertheless, in the static PV model the representation of the load connection, switching and variation is not taken into account. Therefore, dynamic PV model has been proposed to overcome the drawbacks of the static PV models by representing the load connection in the PV model [8, 16]. The accuracy of dynamic PV models is essentially impacted by the precision of their parameters values which are obtained under various operating conditions. Accordingly, an accurate identification of the PV parameters is vital to gain the maximum power of the dynamic PV model.

Several conventional methods are applied for the PV parameters identification [14,29]. Meanwhile, for dynamic PV models, only the non-linear least square approaches and least square have been utilized for parameters identification [1,8]. Nevertheless, conventional methods based on classical numerical/analytical tools might be unable to accurately fit with the PV model, in consequence of the multi-modal and non-linear nature of the problem, which leads to negatively impact the maximum available power optimal capturing.

Motivated from the above discussion, a quantum behaved bonobo optimization algorithm (QBOA) is proposed. Whereby, quantum mechanics are adopted in this paper to integrate a quantum behavior in the bonobo optimization algorithm. For which, the proposed quantum-behaved bonobo optimization exploitation mechanism

absorbs the character of Quantum-behaved method. Aiming to evaluate the robustness and coherence of the QBOA, the proposed QBOA performance is evaluated on CEC2019 and CEC2005 benchmark. Furthermore, the QBOA algorithm was adapted to optimize the dynamic photovoltaic models parameters.

The major contributions of this paper are as follows:

1. A quantum behaved bonobo optimization algorithm is proposed (QBOA), which is combining the advantages of the BOA and quantum mechanics to direct the search agents through the search space.
2. Several tests are conducted over unimodal and multimodal benchmark functions that are adopted for assessing the effectiveness of the proposed QBOA algorithm.
3. The proposed QBOA algorithm is used for optimizing integral and fractional dynamic photovoltaic models. The experimental results ensure that the QBOA algorithm is efficient enough in the identification of the dynamic PV model parameters.

The remainder of the paper is organized as follows: In Section 2, the Bonobo Optimization algorithm brief description is given. In Section 3, the proposed QBOA is detailed. In Section 5, the efficiency of the proposed QBOA algorithm on CEC2019 and CEC2005 benchmarks, as well as comparative analysis of the QBOA versus several optimization algorithms are presented. The dynamic photovoltaic models are provided in Section 4. The simulation results and analysis of the QBOA of dynamic PV models are discussed in Section 5.4. Lastly, in Section 6, The paper's key findings are discussed.

2. Bonobo optimization algorithm

Das and Pratihar presented the Bonobo Optimizer (BO) as a new metaheuristic algorithm [6]. The BO algorithm simulates the reproductive approaches social behavior of bonobos. BO mimics the fission-fusion process, which focuses on segmenting the community into multiple subgroups of varying compositions and sizes, then rejoining them with the rest of the community. Restricting, promiscuous, consortship and extra-group mating are the four types of bonobo strategies. The BO algorithm's working principle is depicted in detail as follows:

Non-user initial parameters are set: the positive and negative phase count $ppc = 0, npc = 0$, the change in phase $cp = 0$, the extra-group mating probability $p_{xgm} = p_{xgm-intial}$, the sizing factor of the temporary sub-group $tsgs_{factor} = tsgs_{factor-initial}$, the directional probability $p_d = 0.5$ and the phase probability $p_p = 0.5$.

Inspired by the fission-fusion social group technique [7], the p^{th} bonobo is chosen for pairing with the i^{th} bonobo. For which, the temporary subgroup maximum size $tsgs_{max}$ is calculated using the following equation:

$$tsgs_{max} = max(2, tsgs_{factor} \cdot N) \tag{1}$$

Where N is the size of the population. According to Equation (1), the temporary subgroup size is between 2 and $tsgs_{max}$; and is constructed by randomly selecting non-repeats bonobos from the $N-1$ population, eliminating the i^{th} bonobo. The optimal solution from the subgroup is picked as the p^{th} bonobo if its fitness is higher than the i^{th} bonobo; contrarily, a randomly bonobo from the subgroup is selected as the p^{th} bonobo. The chosen p^{th} bonobo then begins mating in order to generate the offspring.

In the bonobo society, four different types of mating have been observed: extra-group mating, promiscuous, consortship and restrictive mating. The mating method differs according to whether the phase is positive or negative. The possibility of restricted and promiscuous matings is high during a positive phase. On the other hand, in a negative phase, extra-group and consortship matings are perceived as high. A mating method is utilized using the phase probability parameter p_p. Whereby, a random number $r \in [1, 0]$ is generated, and determined to be either equal or less than p_p. A new_bonobo is created using the following equation:

$$\begin{aligned} new\_bonobo_k = bonobo_k^i + r_1 \cdot scab \cdot (\alpha_k^{bonobo} - bonobo_k^i) + \\ (1 - r_1) \cdot flag \cdot scsb \cdot (bonobo_k^i - bonobo_k^p) \end{aligned} \tag{2}$$

where r_1 is a random number in [0,1] and k is the optimization problem decision variable number. While, scsb and scab are the p^{th} bonobo and the alpha bonobo sharing coefficients, respectively. The parameters scab and scsb are predetermined constants that affect the balance between explorative and exploitative tendencies. The flag parameter can have two possible values: 1 or -1 for promiscuous mating or restrictive mating, respectively.

Sharing coefficients for the alpha-bonobo and pth-bonobo are represented using scab and scsb, respectively.

On the contrary, extra-group mating is employed to create an offspring when r is bigger than p_p, as shown below:

$$\beta_1 = e^{\left(r_4 + r_4^2 - 2/r_4\right)} \tag{3}$$

$$\beta_2 = e^{\left(2r_4 - r_4^2 - 2/r_4\right)} \tag{4}$$

$$new\_bonobo_k = bonobo_k^i + \beta_1 \cdot (UB_k - bonobo_k^i) \tag{5}$$

$$new\_bonobo_k = bonobo_k^i - \beta_2 \cdot (bonobo_k^i - LB_k) \tag{6}$$

$$new\_bonobo_k = bonobo_k^i - \beta_1 \cdot (bonobo_k^i - LB_k) \tag{7}$$

$$new\_bonobo_k = bonobo_k^i + \beta_2 \cdot (UB_k - bonobo_k^i) \tag{8}$$

where, LB_k and UB_k are the lower and upperboundary, respectively; while, $r4 \neq 0$ is a random number.

If a random number $r2$ is greater than $pxgm$, the consortship mating method is utilized to generate an offspring, as follows:

$$new\_bonobo_k = \begin{cases} bonobo_k^i + flag \cdot e^{-r_5} \cdot \left(bonobo_k^i - bonobo_k^p\right), \\ \qquad\qquad \text{if } (r_6 \leq p_d || \text{ flag } = 1) \\ \\ bonobo_k^p, \qquad\qquad \text{otherwise} \end{cases} \tag{9}$$

When the new bonobo's fitness is discovered to be better than the parent's, or when a random number $r \in [0, 1]$ is equal to or less than p_{xgm}, the new bonobo is accepted. Furthermore, if the new_bonobo fitness is shown to be superior to the alpha fitness, the new_bonobo is designated as the alpha-bonobo.

The BO's controlling parameters are modified as follows when the newly obtained alpha bonobo in the current iteration is detected to be an improved solution.

$$npc = 0,\ cp = min(0.5, ppc \cdot rcpp),\ ppc = ppc + 1$$

$$p_{xgm} = p_{xgm-initial},\ p_d = p_p,\ p_p = cp + 0.5$$

$$tsgs_{factor} = min(tsgs_{factor-max}, (tsgs_{factor-initial} + ppc \cdot rcpp^2))$$

Where $rcpp$ is the change rate of the phase probability. On the other hand, the following updates have been made to the controlling parameters:

$$npc = 1 + npc,\ ppc = 0,\ cp = -min(0.5, rcpp \cdot npc)$$

$$p_p = cp + 0.5,\ p_{xgm} = min(0.5, (p_{xgm-initial} - rcpp^2 \cdot npc))$$

$$tsgs_{factor} = min(0, (tsgs_{factor-initial} - npc \cdot rcpp^2)),\ p_d = p_p$$

3. Proposed quantum-behaved Bonobo optimization algorithm (QBOA)

In Bonobo optimizer, the bonobos are characterized by their location and position vector, which constitute the bonobo particle's trajectory. In accordance with Newtonian mechanism particles moves along a predetermined trajectory. However, due to the principle of uncertainty, it is not possible to estimate both distance and position simultaneously in reality.

Accordingly, quantum mechanics are adopted in this study to integrate quantum behaviour in the bonobo optimization algorithm. For which, the proposed quantum-behaved bonobo optimization exploitation mechanism absorbs the character

of Quantum-behaved method. The mechanism setting places the bonobo in quantum mechanics space, utilizes the wave function to describe the bonobo's position and regulated the bonobo's state change process according to the Schrodinger equation [18].

In quantum mechanics, the fundamental time dependent Schrodinger equation is defined by:

$$i\hbar\frac{\partial\Psi}{\partial t} = -\hat{H}(X)\Psi \tag{10}$$

The wave function Ψ described the quantum state of the bonobo and only depends on its position. $\hat{H}(X)$ is a time independent Hamiltonian operator given by:

$$\hat{H}(X) = -\frac{\hbar^2}{2m}\nabla^2 + V(X) \tag{11}$$

Where $\hbar$ is the Planck's constant, $V(X)$ is the potential energy distribution and m is the bonobo mass. In a three dimensional space, the probability density of the bonobo in a position to appear is given by:

$$|\Psi|^2 dxdydz = Qdxdydz \tag{12}$$

Where, Q is the probability density function that meets the normalization condition:

$$\int_{-\infty}^{+\infty} |\Psi|^2 dxdydz = \int_{-\infty}^{+\infty} Qdxdydz = 1 \tag{13}$$

Moreover, the positions of local attractor for each bonobo can be defined as:

$$b_k = scab \cdot r \cdot \alpha_k^{bonobo} + scsb \cdot (1-r) \cdot bonobo_k^p \tag{14}$$

The statistical justification for the wave function is shown by Equations (12) and (13) where the integration is carried out over the full space. Each bonobo in the proposed algorithm has assumed a spin-less movement with a specific potential energy in D-dimensional Hilbert space. The bonobo is pulled using this field in accordance with a position specified by Equation (14).

For D-dimensional Hilbert space, where each bonobo position is bounded by delta potential well; a new_bonobo is created using the following equation:

$$new\_bonobo_k = b_k + flag \cdot \alpha_k^{bonobo} \cdot |bonobo_k^i - W_k| \cdot \ln\left(\frac{1}{r}\right) \tag{15}$$

Where r is a random number in [0,1] and W is the average positions of the bonobo at iteration t. The pseudo code of the QBOA algorithm is presented in Algorithm 1.

Algorithm 1 Pseudocode of the QBOA

Input:
Total population number N_{pop}
Optimization iterations number Max_Iter
Output:
Optimal alpha Bonobo

Initialize the bonobo parameters
Initialize the bonobo population positions randomly.
Calculate the objective values for each search agent and dictate the α bonobo
while $t \leq Max\_Iter$ **do**
  Calculate $tsgs_{max}$ using equation 1
  for i=1:N_{pop} **do**
    Determine the temporary subgroup size
    #Apply the fission-fusion Technique
    Select Flag value
    if $r <= p_p$ **then**
      #Update the position of the bonobo using the quantum mechanism
      Calculate the positions of local attractor for each bonobo using equation 14
      Create new_bonobo using equation 15
      Apply the boundary limiting conditions
    else
      for k=1:d **do**
        if $r_2 < p_{xgm}$ **then**
          if $\alpha_k^{bonobo} >= bonobo_k$ **then**
            if $r_3 <= p_p$ **then**
              Create new_bonobo_k using equation 5
            else
              Create new_bonobo_k using equation 6
            end if
          else
            if $r_4 <= p_p$ **then**
              Create new_bonobo_k using equation 7
            else
              Create new_bonobo_k using equation 8
            end if
          end if
        else
          Create new_bonobo_k using equation 9
        end if
        Apply the boundary limiting conditions
      end for
      Evaluate the new_bonobo_k fitness value
      if $fitness(new\_bonobo_k) < fitness(\alpha^{bonobo})$ **then**
        $\alpha^{bonobo} = new\_bonobo_k$
      end if
    end if
  end for
  Update the controlling parameters
  t=t+1
end while
return α^{bonobo}

4. Dynamic photovoltaic models

4.1. Integral dynamic photovoltaic model

The considered integral dynamic PV model [8] is a second-order model that account the junction capacitance and conductance, along with the inductive effects, as shown in Figure 1a. The PV model and its associated load are valid in the area of the current-voltage curve which lies between the near constant voltage region and the open circuit voltage [9,15]. The circuit in Figure 1a has a linear behavior; accordingly, it is possible to reduce the PV static part to a series resistance Rs and a constant voltage source Voc, yielding the circuit in Figure 1b. While, the dynamic part of the integral PV model is represented by conductance Rc, capacitor C for junction capacitance, and inductance L for cabling and connection inductance.

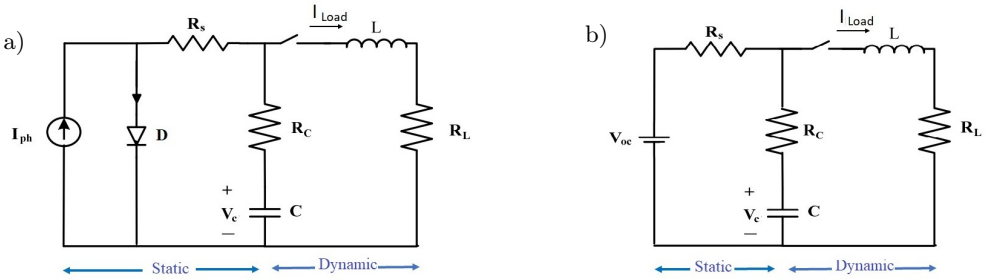


Figure 1. Integral-dynamic PV model: PV model (a); equivalent PV model in the linear voltage region (b)

4.2. Fractional dynamic photovoltaic model

The inductor and capacitor in the fractional dynamic PV model are alternated with fractional equivalents of orders β and α, respectively, as illustrated in Figure 2.

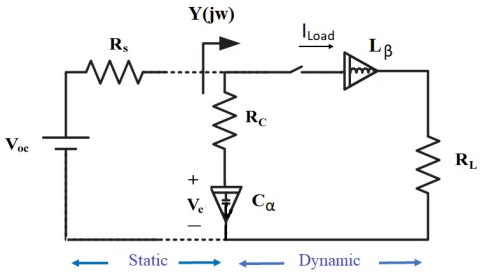


Figure 2. Fractional-dynamic PV model

Whereby, the fractional capacitor's effect is visible in low value of the resistor R_c; due to, the real frequency dependence on fractional capacitance impedance [26].

5. Simulation results and analysis

With the goal of examining the performance and capabilities of the proposed Quantum-Behaved Bonobo Optimization Algorithm QBOA, Matlab R2018b was adjusted for simulation purposes. All evaluation experiments were conducted on: Intel(R), Core i7 4910MQ CPU@2.90GHz and 16GB RAM.

5.1. Performance estimation with CEC 2005

With the aim to evaluate the proposed QBOA algorithm performance, multiple optimization test problems solved over various runs to obtain a reliable conclusion. QBOA is estimated on 23 test functions extracted from the CEC 2005 [19]. Accordingly, the test functions are separated into two categories based on their characteristics: $(F1 - F7)$ unimodal and $(F8 - F23)$ multimodal functions. Whereby, the test objective functions are denoted as: "differentiable, non-differentiable, discontinuous, continuous, scalable, non-scalable,non-separable and separable". Since they include no local optima and a single global optimum, unimodal functions are used to estimate the exploitative potential of the meta-heuristic algorithm. Multimodal functions, on the other hand, have several local optimal and one global optimum. As a result, they can be used to assess the meta-heuristic algorithm's capability for exploration and escape from local optima.

The BOA and proposed QBOA internal parameters are adapted as: total population number $N = 50$, stopping criterion of $30000 \cdot d$, where d is the optimization problems dimension, $p_{xgm-initial} = 1/d$, rcpp=0.0036, $tsgs_{factor-max} = 0.02$, and the sharing coefficients scab = 1.3 and scsb = 1.4. To conduct an unbiased comparison, the statistical results for each test function are calculated across 30 independent runs with completely random initial conditions. Through which, four distinct evaluation factors are taken into account: the minimum (best) solution, the average (mean) solution, the maximum (worst) solution and the standard deviation (St.dev). The worst, best and mean metrics examine the accuracy of the solution, while the St.dev estimates the obtained solution's robustness.

The experimental finding of the BOA and proposed QBOA; on the unimodal and multimodal test optimization functions are recorded in Table 1. From Table 1, it is clear that the proposed quantum-behaved BO surpass the BO algorithm in term of mean, best and worst results, except for test function F12. Both QBOA and BOA could continuously attain the global optimal for F5 and F6. Furthermore, compared to the BOA, the QBOA was able to locate the global optima with less standard deviation for all test functions, which demonstrate the QBOA's robustness in locating the global optimal.

Table 1
Statistical results of BOA and QBOA algorithm on CEC 2005

Function	BOA				QBOA			
	Best	Mean	Worst	St.dev	Best	Mean	Worst	St.dev
F1	9.5176E–57	4.51667E–46	1.35358E–44	2.4712E–45	1.3325E–141	2.0989E–118	5.3473E–117	9.7943E–118
F2	9.18109E–29	3.80198E–26	2.25682E–25	6.34863E–26	5.29453E–72	3.15854E–60	9.45349E–59	1.72583E–59
F3	3.01466E–56	5.08291E–50	6.43562E–49	1.33984E–49	9.6229E–136	1.8777E–109	5.633E–108	1.0284E–108
F4	4.27144E–28	3.31868E–25	2.49976E–24	6.22758E–25	9.13827E–72	6.93219E–57	2.05336E–55	3.74755E–56
F5	0	0	0	0	0	0	0	0
F6	0	0	0	0	0	0	0	0
F7	1.17E–04	1.49E–03	4.23E–03	1.09E–03	2.84E–06	7.58E–04	2.34E–03	5.87E–04
F8	–418.9828873	–411.0869983	–300.5445527	30.0487686	–4189.982887	–4182.982887	–4111.982869	3.31728E–06
F9	0	0.033165302	0.994959057	0.18165384	0	0	0	0
F10	8.88178E–16	8.88178E–16	8.88178E–16	0	8.88178E–16	8.88178E–16	8.88178E–16	0
F11	0	0.001644112	0.009864672	0.003739194	0	0	0	0
F12	4.71163E–31	4.71163E–31	4.71163E–31	8.90784E–47	7.81257E–26	3.69146E–08	1.10744E–06	2.0219E–07
F13	4.69E–27	3.0545E–06	6.62661E–05	1.28046E–05	1.34978E–32	1.34978E–32	1.34978E–32	5.5674E–48
F14	0.998003838	1.776170636	12.67050581	2.961408668	0.998003838	0.998003838	0.998003838	1.00999E–16
F15	3.08E–04	4.58E–03	2.04E–02	7.51E–03	3.09E–04	3.30E–03	2.04E–02	6.81E–03
F16	–1.031628453	–1.031628453	–1.031628453	5.21556E–16	–1.03	–1.03	–1.03	0
F17	0.397887358	0.397887358	0.397887358	0	0.397887358	0.39788736	0.397887	0
F18	3	3.9	30	4.929503018	3	3	3	1.30E–15
F19	–3.862779787	–3.837012599	–3.089764134	0.141132703	–3.86	–3.86	–3.86	7.05E–15
F20	–3.042457738	–3.017878829	–2.981002427	0.030617435	–3.042423011	–3.025796432	–2.975483131	0.026197581
F21	–10.15319876	–9.984784532	–5.100772055	0.922442691	–10.15319876	–10.1531982	–10.15318981	2.12097E–16
F22	–10.40282204	–10.05122207	–5.128822495	1.338056572	–10.40282204	–10.40232042	–10.39505238	0.001910066
F23	–10.5362903	–9.998653738	–5.128480623	1.640498374	–10.5362903	–10.53629027	–10.53628927	1.87452E–17

The proposed QBOA was analyzed versus six well known met-heuristic algorithms: Coyote Optimization Algorithm (COA) [24], Dragonfly Algorithm (DA) [21], Particle Swarm Optimization (PSO) [10], Gravitational Search Algorithm (GSA) [27], Moth-Flame Optimization (MFO) [20], Genetic algorithm (GA) [17], as shown in Table 2. The initial controlling parameters of the optimization algorithms are listed in Table 3.

Table 2

Comparison results attained for QBOA and different optimization algorithms

Function		QBOA	COA	PSO	DA	GSA	GA	MFO
F1	Mean	**2.0989E–118**	25.32456	1.36E–04	5.30E–01	2.53E–16	8.00E–04	1.65E–31
	St.dev	**9.7943E–118**	9.284834	2.02–7	1.318	9.67E–17	8.70E–04	4.91E–31
F2	Mean	**3.15854E–60**	0.7051718	0.042144	2.392	0.055655	3.00E–03	2.69E–19
	St.dev	**1.72583E–59**	0.1208514	0.045421	3.912	0.194074	1.80E–03	6.22E–19
F3	Mean	**1.8777E–109**	2252.63547	70.12562	215.45	896.5347	13.213	2.05E–11
	St.dev	**1.0284E–108**	825.3839	22.11924	935.17	318.9559	8.042	4.21E–11
F4	Mean	**6.93219E–57**	24.4519057	1.086481	1.153	7.35487	0.209	5.79E–06
	St.dev	**3.74755E–56**	3.6646211	0.317039	2.702	1.741452	5.80E–02	3.17E–05
F5	Mean	**0**	2592.44628	96.71832	6784.5	67.54309	66.9	133.11
	St.dev	**0**	1925.75963	60.11559	21974.5	62.22534	22.6	555.57
F6	Mean	**0**	27.835087	0.000102	2.2023	2.50E–16	7.50E–04	4.78E–32
	St.dev	**0**	12.9163617	8.28E–05	5.528	1.74E–16	7.20E–04	1.27E–31
F7	Mean	**7.58E–04**	6.72E–02	1.23E–01	6.90E–03	0.089441	8.10E–04	1.20E–03
	St.dev	5.87E–04	2.40E–02	4.50E–02	7.60E–03	0.04339	**5.50E–04**	7.20E–04
F8	Mean	–4182.982887	**–12299.311**	–4841.29	–3213.66	–2821.07	–3692.39	–3329.13
	St.dev	**3.31728E–06**	105.679038	1152.814	431.748	493.0375	182.42	288.317
F9	Mean	**0**	20.11984	46.70423	11.561	25.96841	3.80E–04	12.8372
	St.dev	**0**	2.22484365	11.62938	10.177	7.470068	3.20E–04	7.352
F10	Mean	**8.88E–16**	4.0391476	2.76E–01	3.14E–05	6.21E–02	**8.88E–16**	**8.88E–16**
	St.dev	**0**	0.95061505	5.09E–01	1.70E–04	2.36E–01	1.00E–31	1.00E–31
F11	Mean	**0**	1.1976333	9.22E–03	0.3846	27.70154	8.88E–16	1.78E–01
	St.dev	**0**	7.64E–02	7.72E–03	0.3826	5.040343	1.00E–31	8.43E–02
F12	Mean	**3.69146E–08**	2.0935602	6.92E–03	0.5296	1.799617	5.73E–05	3.11E–02
	St.dev	**2.0219E–07**	0.809141215	2.63E–02	0.6912	0.95114	1.40E–04	9.49E–02
F13	Mean	**1.34978E–32**	11.91776	6.68E–03	0.5292	8.899084	6.21E–05	1.10E–03
	St.dev	**5.5674E–48**	3.7510395	8.91E–03	0.7173	7.126241	1.10E–04	3.33E–03
F14	Mean	**0.998**	**0.998**	3.627168	1.1	3.4	**0.998**	1.03
	St.dev	1.00999E–16	**4.12E–17**	2.560828	0.303306	2.578637	8.83E–14	0.181483682
F15	Mean	3.30E–03	**3.07E–04**	5.77E–04	1.34E–03	1.80E–03	8.40E–04	8.37E–04
	St.dev	6.81E–03	**7.70E–09**	2.22E–04	5.11E–04	4.90E–04	2.90E–04	2.54E–04
F16	Mean	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**
	St.dev	**0**	6.58E–16	6.25E–16	2.55E–11	**0**	5.02E–10	**0**
F17	Mean	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**
	St.dev	**0**	**0**	**0**	7.60E–13	**0**	4.73E–07	1.13E–16
F18	Mean	**3**	**3**	**3**	**3**	**3**	**3**	**3**
	St.dev	**1.30E–15**	1.36E–15	1.33E–15	1.38E–06	4.17E–15	1.21E–08	1.95E–15
F19	Mean	**–3.86**	–3.86277	–3.8628	**–3.86**	–3.8628	**–3.86**	**–3.86**
	St.dev	7.05E–15	3.12E–15	**2.58E–15**	1.59E–03	2.29E–15	2.20E–03	2.71E–15

Table 2 cont.

Function		QBOA	COA	PSO	DA	GSA	GA	MFO
F20	Mean	–3.025796432	–3.04245773	–3.26634	–3.25	–3.31778	**–3.32**	**–3.22**
	St.dev	2.62E–02	**2.21E–13**	6.05E–02	6.72E–02	2.31E–02	2.17E–02	4.51E–02
F21	Mean	–10.1531982	–10.153199	–9.31	–9.81	–9.95512	**–10.2**	–7.56
	St.dev	**2.12097E–16**	7.23E–15	1.925505	1.280913	3.737079	4.84E–04	3.323037
F22	Mean	**–10.40232042**	**–10.40232042**	–9.52	–10.4	–9.68447	–9.93	–9.35
	St.dev	**1.91E–03**	0.962917757	2.00228	0.192434	2.014088	1.822252	2.423664
F23	Mean	**–10.536**	**–10.536**	**–10.536**	**–10.536**	**–10.536**	–9.61	–10.3
	St.dev	**1.87452E–17**	1.22342651	1.635722	1.060781	2.60E–15	2.405191	1.39948

Table 3
Parameter settings of the optimization algorithms

Optimization Algorithm	Parameter Setting
Quantum-Behaved Bonobo Optimization Algorithm (QBOA)	rcpp = 0.0036, $tsgs_{factor-max} = 0.02$, scab = 1.25, scsb = 1.3
Particle Swarm Optimization (PSO)	c1 = c2 = 2, $w_{max} = 0.9, w_{min} = 0.2$
Coyote Optimization Algorithm (COA)	packs number $N_p = 10$, coyotes number $N_c = 10$
Gravitational Search Algorithm (GSA)	$G_0 = 100, alpha = 20$
Genetic Algorithm (GA)	Roulette wheel selection, crossover = 0.7, mutation = 0.3
Dragonfly Algorithm (DA)	$\beta = 0.5$
Moth-Flame Optimization (MFO)	b = 1, a decreased linearly from –1 to –2

As reported in Table 2, the proposed QBOA algorithm surpassed the comparison algorithms for all optimization test functions except for F8, where COA presents better mean and QBOA presents the second best; and for F15 where COA presents better standard deviation and mean measure. Compared to the GA and MFO algorithm, QBOA finds the second best results for function F20. Moreover, for functions F5, F6, F9, and F11, the theoretical global optimum could be consistently located through QBOA. While, for test function F10 QBOA was able to attain the theoretical global optimal similar to GA and MFO algorithms, however with the best St.dev.

5.2. Performance estimation with CEC 2019

Extra examinations of the proposed quantum behaved BO on the CEC2019 benchmarks are undertaken in this sub-section. CEC2019 [25] signifies a test environment, including ten different functions with various characteristics. CEC01, CEC02, and CEC03 are the first three functions, with dimensions of: $[-8192, 8192]$, $[-16384, 16384]$ and $[-4, 4]$, respectively. While, the test functions CEC04 to CEC10 are shifted and rotated in the range $[-100, 100]$. The evaluation findings are reported in Table 4. From Table 4, the proposed QBOA algorithm provides the best results in term of best, worst, mean and standard deviation for all test functions.

Table 4
Statistical results of BOA and QBOA algorithm on CEC 2019

Function	BOA				QBOA			
	Best	Mean	Worst	St.dev	Best	Mean	Worst	St.dev
CEC01	5.05E+09	4.39E+10	8.01646E+11	1.45922E+11	9.44E+07	3.82E+10	1.67726E+11	35343962073
CEC02	17.54313222	29.24788963	147.0449426	27.2284062	17.34285715	17.34394919	17.37353053	0.000158971
CEC03	12.70240428	12.70240486	12.70241987	2.88462E–06	12.70240422	12.70240826	12.70242195	1.13E-07
CEC04	196.933194	1224.912295	4116.441557	835.165023	32.78727204	188.1486484	1342.768471	24.1737793
CEC05	1.462741151	1.963403865	2.282512869	0.18092835	1.041741434	1.105880945	1.935413404	0.021160053
CEC06	8.32434066	11.14097169	13.38913543	1.333579119	8.028343674	9.275948809	13.29664279	1.024430079
CEC07	230.2766639	841.5797016	1358.830208	290.1988289	105.5116356	410.001414	1263.376387	186.1911856
CEC08	5.80548777	6.799254532	7.18450629	0.990889863	3.71901424	5.099504448	7.585492774	0.401191137
CEC09	5.030062803	53.47166279	205.4756953	52.53316654	2.075989319	2.163408587	5.849287551	0.008198
CEC10	20.2806579	20.57014903	20.78783798	0.135932697	20.26194419	20.13350873	20.77914636	0.115771177

In addition, the QBOA is examined against four well known algorithms in the literature: Particle Swarm Optimization (PSO) [10], Dragonfly Algorithm (DA) [21], Whale optimization algorithm (WOA) [23] and Salp swarm algorithm (SSA) [22], Table 5. From Table 5, the proposed QBOA Obtains better results for CEC02, CEC05, CEC06, CEC08 and CEC09. Furthermore, QBOA provides the better mean for CEC10, and produces the second-best outcomes for test function CEC07 in comparison to the PSO algorithm.

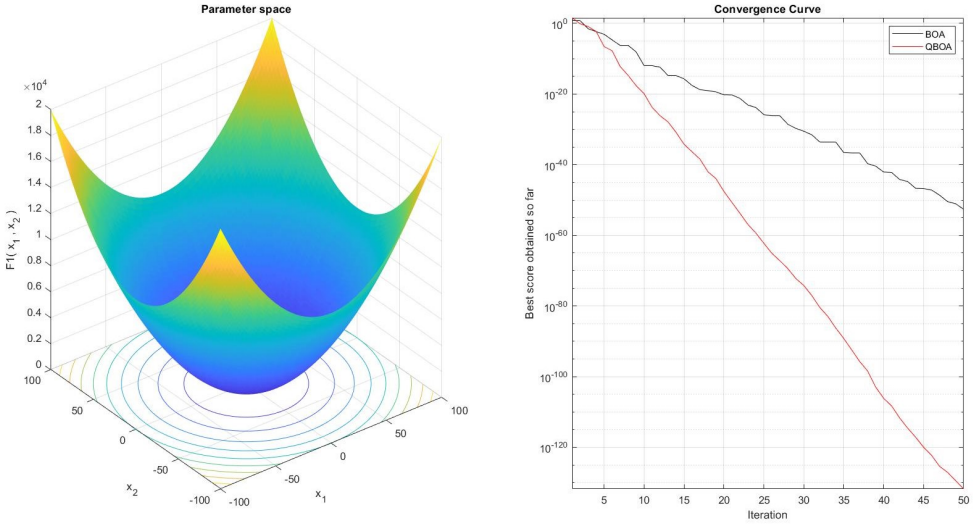
Table 5
Comparison results attained for QBOA and different optimization algorithms on CEC2019

Function		QBOA	PSO	DA	WOA	SSA
CEC01	Mean	3.82E+10	1.471E+12	5.43E+10	4.11E+10	**6.05E+09**
	St.dev	3.53E+10	1.324E+12	6.69E+10	5.42E+10	**4.75E+09**
CEC02	Mean	**17.34394919**	15183.91348	78.0368	17.3495	18.3434
	St.dev	**0.000158971**	3729.553229	87.7888	0.0045	0.0005
CEC03	Mean	**12.7024**	**12.7024**	13.7026	13.7024	13.7025
	St.dev	1.13E-07	9.03E-15	0.0007	**0**	0.0003
CEC04	Mean	188.1486484	**16.80077558**	344.3561	394.6754	41.6936
	St.dev	24.1737793	**8.199**	414.0982	248.5627	22.2191
CEC05	Mean	**1.105880945**	1.138264	2.5572	2.7342	2.2084
	St.dev	**0.021160053**	0.08938	0.3245	0.2917	0.1064
CEC06	Mean	**9.275948809**	9.30531	9.8955	10.7085	6.0798
	St.dev	**1.024430079**	1.69	1.6404	1.0325	1.4873
CEC07	Mean	410.001414	**160.686**	578.9531	490.6843	410.3964
	St.dev	186.1911856	**104.203**	329.3983	194.8318	290.5562
CEC08	Mean	**5.099504448**	5.2241	6.8734	6.909	6.3723
	St.dev	**0.401191137**	0.7867	0.5015	0.4269	0.5862
CEC09	Mean	**2.163408587**	2.373279	6.0467	5.9371	3.6704
	St.dev	**0.008198**	0.01843	2.871	1.6566	0.2362
CEC10	Mean	**20.13350873**	20.2806	21.2604	21.2761	21.04
	St.dev	0.115771177	0.12853	0.1715	**0.1111**	0.078

5.3. Convergence analysis

The proposed QBOA and BOA's convergence behaviour are explored. The cost function for F1–F4, F7, F9, F10, F13, F15, and F21 test problems, as well as their associated convergence curves, are shown in Figures 3–6. As shown in Figure 3, the QBOA algorithm exhibits abrupt changes in the early stages of iterations, which decreased gradually throughout the course of iterations. This behaviour demonstrates that, the proposed QBOA algorithm gains from a well balance of exploitation and exploration, accordingly enables the QBOA to avoid being locked into local optimals.

a)



b)

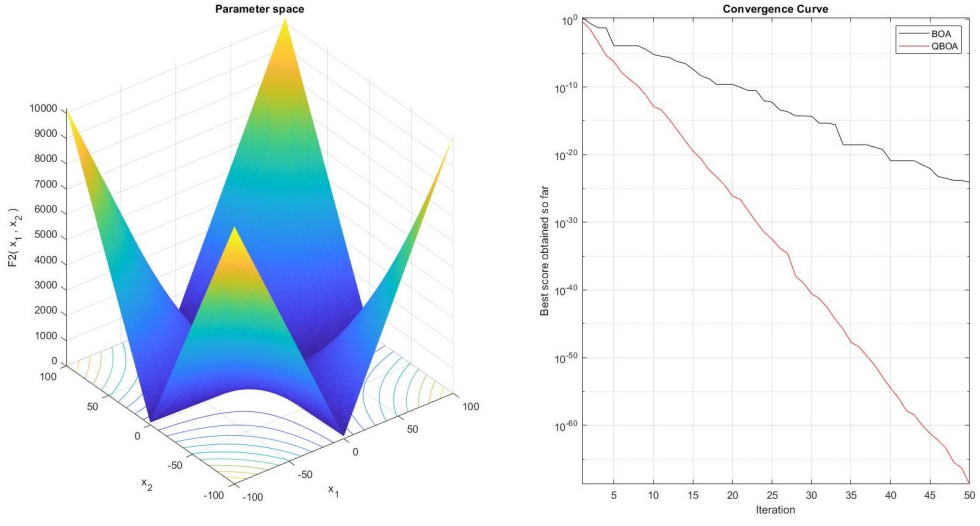
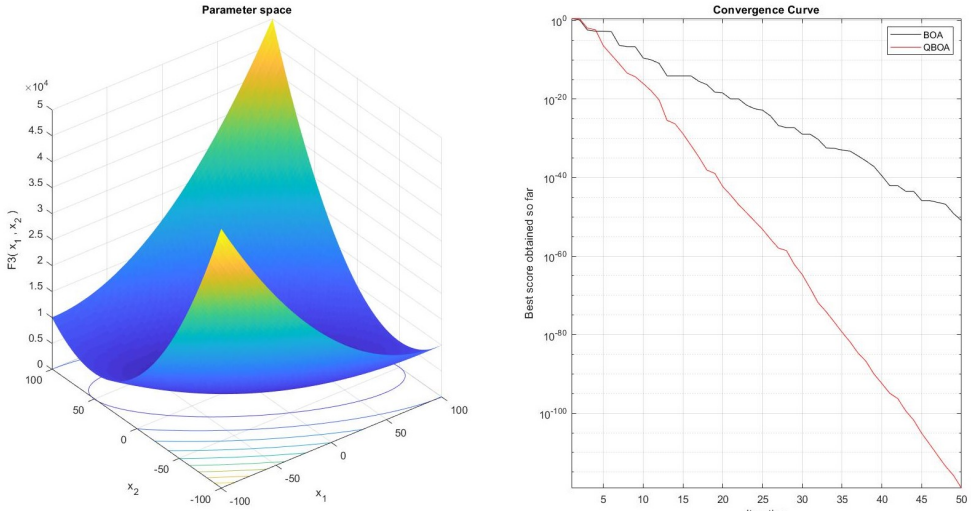


Figure 3. Best fitness convergence curves: a) $F1$; b) $F2$

a)



b)

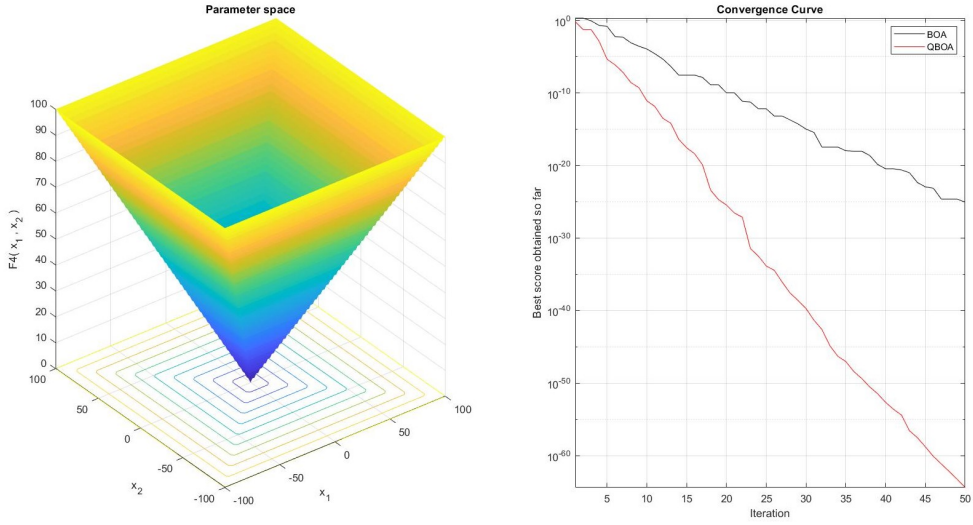
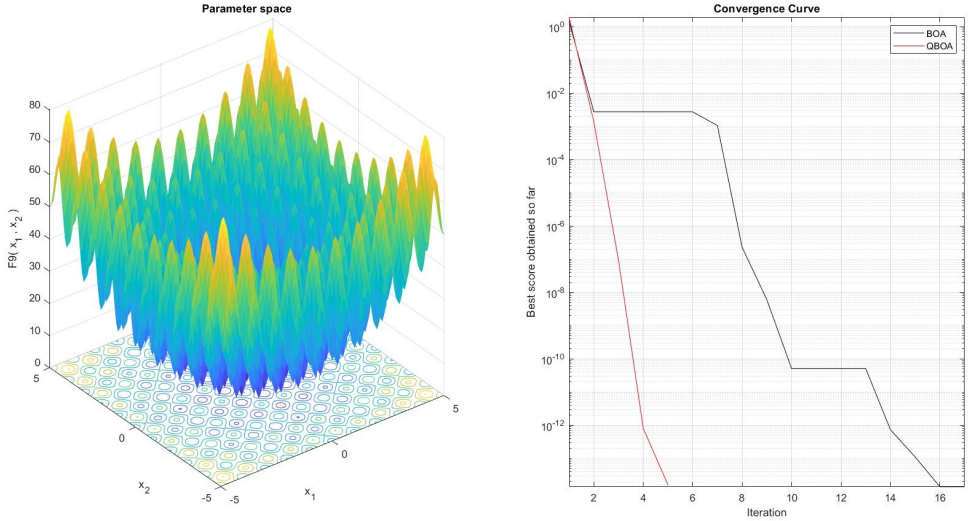


Figure 4. Best fitness convergence curves (cont.): a) $F3$; b) $F4$

a)



b)

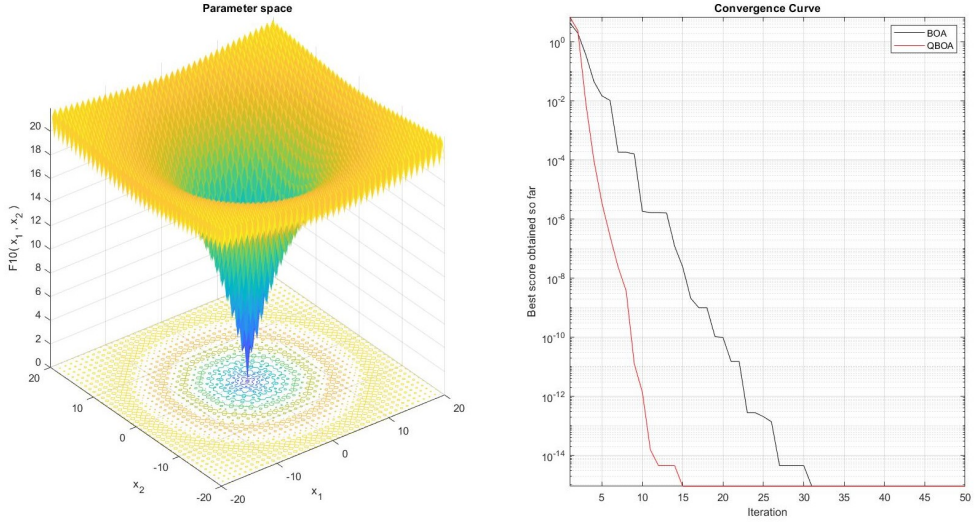
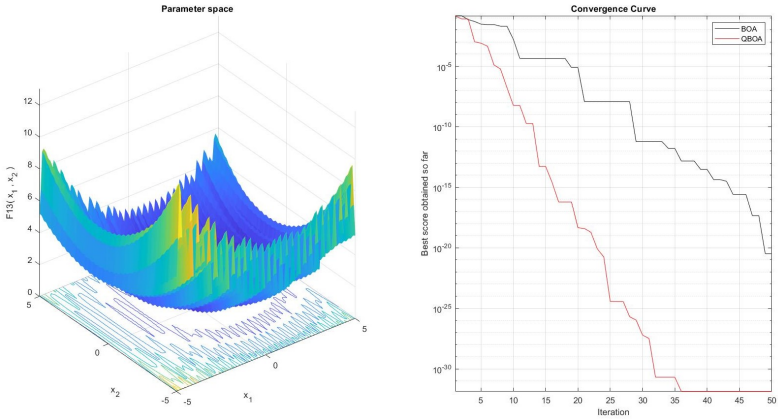
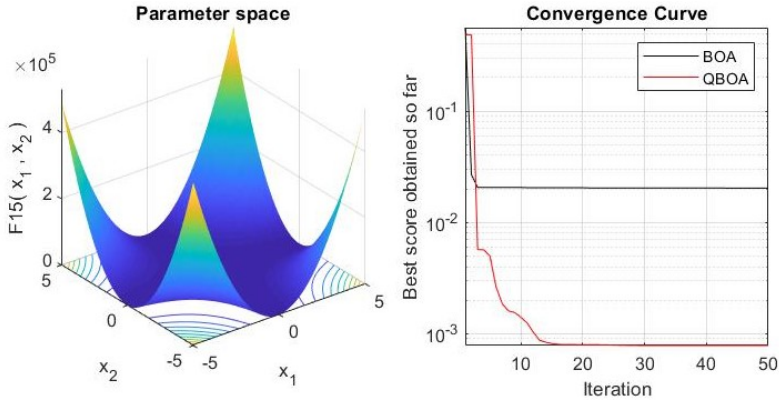


Figure 5. Best fitness convergence curves (cont.): a) $F9$; b) $F10$

a)



b)



c)

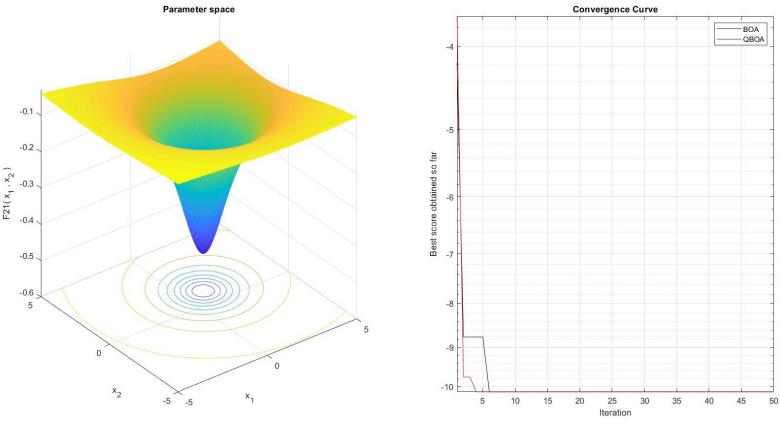


Figure 6. Best fitness convergence curves (cont.): a) $F13$; b) $F15$; c) $F21$

5.4. Dynamic PV models' parameters estimation

This subsection is concerned with the application of the proposed QBOA for parameter optimization of the integral and fractional dynamic PV models. The dynamic data sets were collected from PV module at an irradiance level of 655 W/m^2 and a temperature of 25°C, with connected load $R_L = 23.1\ \Omega$ [8]. Whereby, for the integral dynamic PV model, three unknown parameters R_c, L and C should be estimated, while in the fractional dynamic PV model five parameters $R_c, L_\beta, C_\alpha, \beta$ and α should be estimated.

Thirty independent runs of the PV optimization problems were performed. The BOA and proposed QBOA internal parameter values for optimizing the dynamic PV models were kept the same as follows: $N = 50$, $p_{xgm-initial} = 1/d$, $rcpp = 0.0036$, $tsgs_{factor-max} = 0.02$, $scab = 1.3$ and $scsb = 1.4$. The upper and lower boundaries of the fractional and integral dynamic PV models are given in Table 6.

Table 6
Dynamic PV model parameters boundaries

Model	Parameters	Lower bound (LB)	Upper bound (UB)
Integral PV	R_c	0	20
	C	2.0E–08	600 E–07
	L	5.0 E–06	100 E–06
Fractional PV	R_c	0	20
	C_α	2.0E–08	600 E–07
	L_β	5.0 E–06	100 E–06
	α	0.8	1.1
	β	0.8	1.1

The three optimized parameters $(R_c, L, C,)$ for the integral PV model, the five optimized parameters $(R_c, C_\alpha, \alpha, L_\beta, \beta)$ for the fractional PV model and their corresponding objective function (RMSE) are reported in Table 7 and Table 8 . Moreover, the proposed QBOA's RMSE findings are compared to BOA algorithm and various well-known and recently developed optimizers, including Gradient-Based Optimizer (GBO) [2], artificial ecosystem based optimization (AEO) [30] and jellyfish search optimizer (JS) [5]. The tabulated results revealed that, the proposed QBOA produced the best RMSE values.

For more comprehensive validation of the proposed QBOA, the load current curve estimated by QBOA is generated and contrasted with that of the measured data for the integral and fractional PV model in Figure 7 and Figure 8, respectively. Additionally, the absolute error curves between the measured and estimated load current curves are given in Figure 9 and Figure 10. The visual comparisons validate the efficiency of the proposed QBOA in the identification of the dynamic PV model parameters.

Table 7
Identified parameters of integral dynamic PV model

Algorithm	R_c	C	L	RMSE
GBO	5.624748753	8.16E–06	7.47 E–06	0.008493067
AEO	5.624748647	8.16E–06	7.47 E–06	0.0084931
JS	5.624749	8.15726 E–06	7.47323 E–06	0.008493067
BOA	7.314974895	3.81307E–07	7.3251E–06	0.0084805
proposed QBOA	7.314974219	3.81307E–07	7.3251E–06	0.0084505

Table 8
Identified parameters of fractional dynamic PV model

Algorithm	R_c	C_α	L_β	α	β	RMSE
GBO	5.00598	5.04 E–06	1.35 E–05	1.026120535	0.957165925	0.0082360
AEO	4.55020	1.46 E–05	1.73 E–05	0.917230623	0.940654537	0.0081960
JS	4.69892	4.08 E–05	1.44 E–05	0.833404373	0.953192785	0.007995872
BOA	3.91281E–05	1.80215E–06	7.20724E–05	0.947239708	0.846072368	0.006168565
proposed QBOA	1.00E–05	1.83738E–06	6.60382E–05	0.94299238	0.852227908	0.006155832

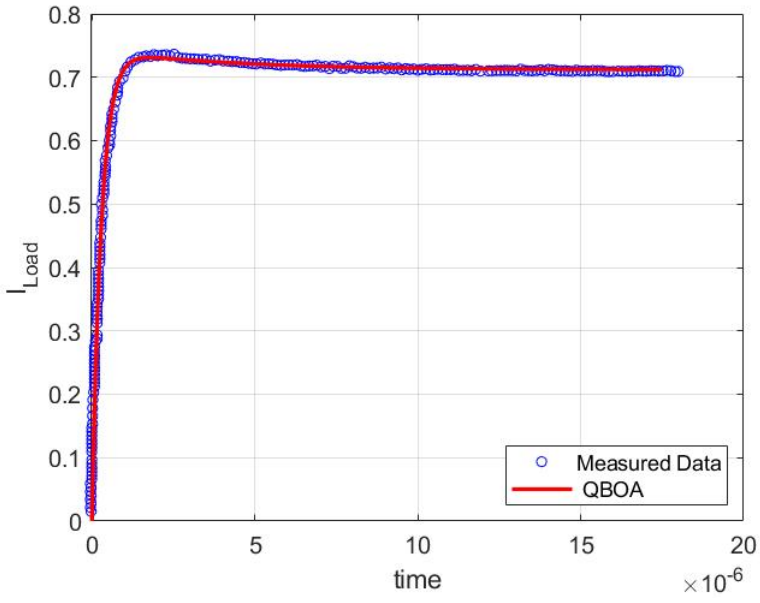


Figure 7. Integral dynamic PV model load current fitting

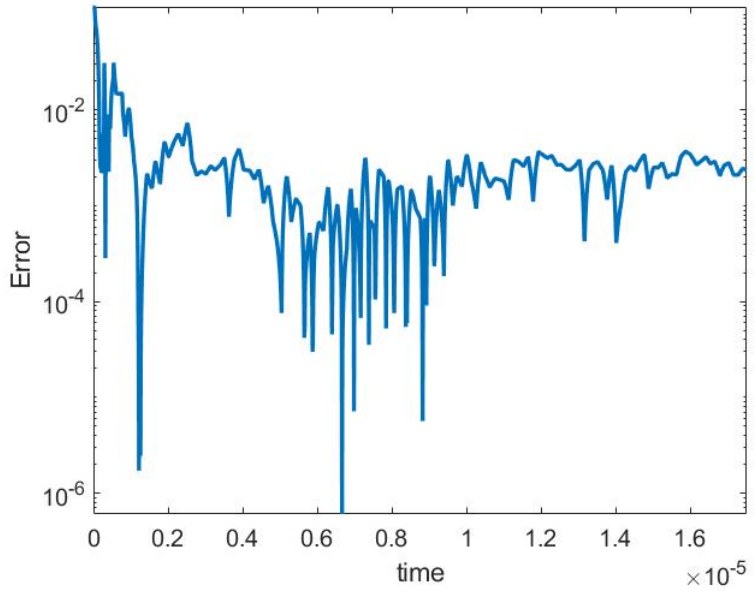


Figure 8. Integral dynamic PV model absolute error

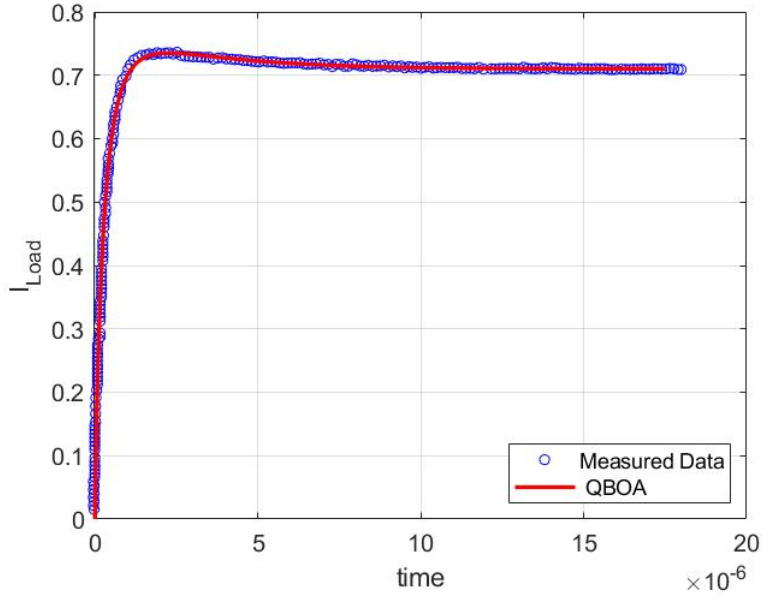


Figure 9. Fractional dynamic PV model load current fitting

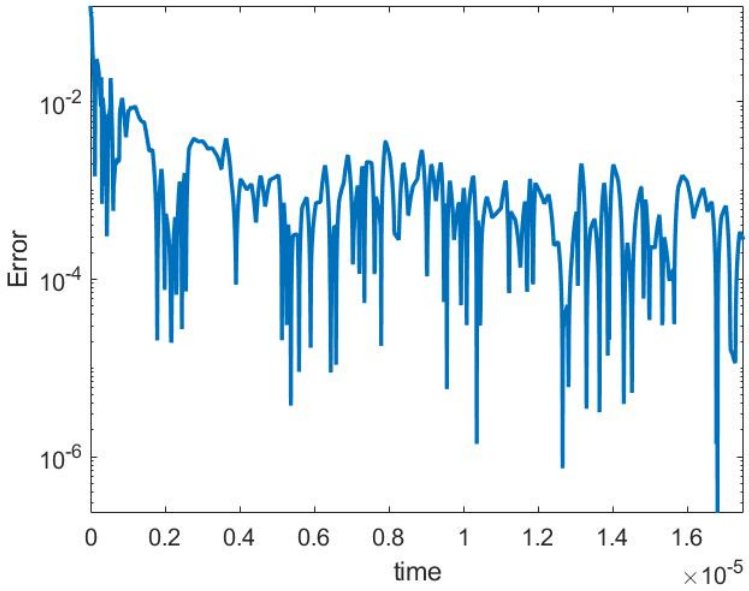


Figure 10. Fractional dynamic PV model absolute error

6. Conclusion

The finite supply of non-renewable resources combined with the world population's rapid increase is driving up demand for energy. Due to this circumstance, there is a risk of environmental pollution and climate change. Therefore, researchers' attention towards renewable energy sources, especially solar energy, have taken the spotlight. Several photovoltaic models have been proposed, where designing a high performance photovoltaic system requires addressing the problem of simulating a solar module and identifying its parameter. This paper employs a quantum behaved meta-heuristic named QBOA for addressing various optimization problems and dynamic photovoltaic models parameter identification. QBOA simulates the reproductive strategies and social behavior of bonobos. Quantum mechanics incorporated into the QBOA algorithm to direct the search agents through the search space. Correspondingly, the exploitation potential of the proposed QBOA algorithm is promoted by this integration. To determine the robustness and coherence of the QBOA, its performance has been examined using the CEC2019 and CEC2005 benchmarks. Additionally, the proposed QBOA is presented to optimize dynamic photovoltaic model's parameter. From the experimental and simulations results, it can be designated that the quantum behaved QBOA sustains the competitiveness on solving different optimization problems and optimizing dynamic photovoltaic models.

References

[1] AbdelAty A.M., Radwan A.G., Elwakil A.S., Psychalinos C.: Transient and Steady-State Response of a Fractional-Order Dynamic PV Model Under Different Loads, *Journal of Circuits, Systems and Computers*, vol. 27(02), 1850023, 2018. doi: 10.1142/S0218126618500238.

[2] Ahmadianfar I., Bozorg-Haddad O., Chu X.: Gradient-based optimizer: A new metaheuristic optimization algorithm, *Information Sciences*, vol. 540, pp. 131–159, 2020. doi: 10.1016/j.ins.2020.06.037.

[3] Ali M.Z., Awad N.H., Suganthan P.N., Duwairi R.M., Reynolds R.G.: A novel hybrid Cultural Algorithms framework with trajectory-based search for global numerical optimization, *Information Sciences*, vol. 334–335, pp. 219–249, 2016. doi: 10.1016/j.ins.2015.11.032.

[4] Ayang A., Wamkeue R., Ouhrouche M., Djongyang N., Salomé N.E., Pombe J.K., Ekemb G.: Maximum likelihood parameters estimation of single-diode model of photovoltaic generator, *Renew Energy*, vol. 130, pp. 111–121, 2019. doi: 10.1016/j.renene.2018.06.039.

[5] Chou J.S., Truong D.N.: A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean, *Applied Mathematics and Computation*, vol. 389, 125535, 2021. doi: 10.1016/j.amc.2020.125535.

[6] Das A.K., Pratihar D.K.: A new Bonobo Optimizer (BO) for real-parameter optimization. In: *2019 IEEE Region 10 Symposium (TENSYMP)*, pp. 108–113, 2019. doi: 10.1109/tensymp46218.2019.8971108.

[7] De Waal F.B.: Bonobo Sex and Society, *Scientific American*, vol. 272, pp. 82–88, 1995. doi: 10.1038/scientificamerican0395-82.

[8] Di Piazza M.C., Luna M., Vitale G.: Dynamic PV Model Parameter Identification by Least-Squares Regression, *IEEE Journal of Photovoltaics*, vol. 3, pp. 799–806, 2013. doi: 10.1109/jphotov.2012.2236146.

[9] Di Piazza M.C., Vitale G.: Photovoltaic field emulation including dynamic and partial shadow conditions, *Applied Energy*, vol. 87, pp. 814–823, 2010.

[10] Eberhart R., Kennedy J.: A new optimizer using particle swarm theory. In: *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, Nagoya, Japan, 1995.

[11] Eid H.F., Abraham A.: Solving unconstrained, constrained optimization and constrained engineering problems using reconfigured water cycle algorithm, *Evolutionary Intelligence*, vol. 16, pp. 633–649, 2023. doi: 10.1007/s12065-021-00688-6.

[12] Eid H.F., Muda A.K.: Adjustive Reciprocal Whale Optimization Algorithm for Wrapper Attribute Selection and Classification, *International Journal of Image, Graphics and Signal Processing*, vol. 11, pp. 18–26, 2019. doi: 10.5815/ijigsp.2019.03.03.

[13] El-Fergany A.: Efficient tool to characterize photovoltaic generating systems using mine blast algorithm, *Electric Power Components and Systems*, vol. 43(8–10), pp. 890–901, 2015. doi: 10.1080/15325008.2015.1014579.

[14] Et-torabi K., Nassar-eddine I., Obbadi A., Errami Y., Rmaily R., Sahnoun S., El fajri A., Agunaou M.: Parameters estimation of the single and double diode photovoltaic models using a Gauss–Seidel algorithm and analytical method: A comparative study, *Energy Conversion and Management*, vol. 148, pp. 1041–1054, 2017. doi: 10.1016/j.enconman.2017.06.064.

[15] Gil-Arias O., Ortiz-Rivera E.I.: General purpose tool for simulating the behavior of PV solar cells modules and arrays. In: *2008 11th Workshop on Control and Modeling for Power Electronics*, pp. 1–5, 2008. doi: 10.1109/compel.2008.4634686.

[16] Go S.I., Choi J.H.: Design and Dynamic Modelling of PV-Battery Hybrid Systems for Custom Electromagnetic Transient Simulation, *Electronics*, vol. 9, 1651, 2020. doi: 10.3390/electronics9101651.

[17] Holland J.H.: Genetic algorithms, *Scholarpedia*, vol. 7(12), 1482, 2012. doi: 10.4249/scholarpedia.1482.

[18] Levin F.S.: *An introduction to quantum theory*, Cambridge University Press, 2002.

[19] Liang J.J., Suganthan P.N., Deb K.: Novel composition test functions for numerical global optimization. In: *Proceedings 2005 IEEE Swarm Intelligence Symposium, SIS 2005*, pp. 68–75, 2005. doi: 10.1109/SIS.2005.1501604.

[20] Mirjalili S.: Moth-flame optimization algorithm: a novel natureinspired heuristic paradigm, *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015. doi: 10.1016/j.knosys.2015.07.006.

[21] Mirjalili S.: Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete and multi-objective problems, *Neural Computing and Applications*, vol. 27, pp. 1053–1073, 2016. doi: 10.1007/s00521-015-1920-1.

[22] Mirjalili S., Gandomi A.H., Mirjalili S.Z., Saremi S., Faris H., Mirjalili S.M.: Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems, *Advances in Engineering Software*, vol. 114, pp. 163–191, 2017. doi: 10.1016/j.advengsoft.2017.07.002.

[23] Mirjalili S., Lewis A.: The whale optimization algorithm, *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016. doi: 10.1016/j.advengsoft.2016.01.008.

[24] Pierezan J., Dos Santos Coelho L.: Coyote optimization algorithm: a new metaheuristic for global optimization problems. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2633–2640, Brazil, Rio de Janeiro, 2018. doi: 10.1109/cec.2018.8477769.

[25] Price K.V., Awad N.H., Ali M.Z., Suganthan P.N.: *Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization*, Technical report, Nanyang Technological University, Singapore, 2018.

[26] Radwan A.G., Salama K.N.: Fractional-order, *RC* and and *RL* circuits, *Circuits, Systems, and Signal Processing*, vol. 31, pp. 1901–1915, 2012. doi: 10.1007/s00034-012-9432-z.

[27] Rashedi E., Nezamabadi-Pour H., Saryazdi S.: GSA: a gravitational search algorithm, *Information Sciences*, vol. 179(13), pp. 2232–2248, 2009. doi: 10.1016/j.ins.2009.03.004.

[28] Ridha H.M., Heidari A.A., Wang M., Chen H.: Boosted mutation-based Harris hawks optimizer for parameters identification of single-diode solar cell models, *Energy Conversion and Management*, vol. 209, 112660, 2020. doi: 10.1016/j.enconman.2020.112660.

[29] Tossa A.K., Soro Y.M., Azoumah Y., Yamegueu D.: A new approach to estimate the performance and energy productivity of photovoltaic modules in real operating conditions, *Solar Energy*, vol. 110, pp. 543–560, 2014. doi: 10.1016/j.solener.2014.09.043.

[30] Zhao W., Wang L., Zhang Z.: Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm, *Neural Computing & Applications*, vol. 32, pp. 9383–9425, 2020. doi: 10.1007/s00521-019-04452-x.

Affiliations

Heba F. Eid
Al-Azhar University, Faculty of Science, Cairo, Egypt, heba.fathy@azhar.edu.eg

Erik Cuevas
University of Guadalajara, Department of Electronics, Mexico, erik.cuevas@cucei.udg.mx

Received: 30.07.2023
Revised: 2.04.2024
Accepted: 2.04.2024