Chapter One

Introduction

1.1 Background to the Study

Robust object detection and recognition are fundamental aspects of grasping, robot manipulation, human-robot interaction and augmented reality. However, cluttered environments, occlusion between objects, lighting conditions and small deformable objects remain as challenges. Furthermore, objects may appear in different scale and forms depending on the camera view point and calibration. Therefore, accurate scene understanding including object detection and pixel-wise semantic segmentation is crucial for practical interaction with real-world objects[1]. Object detection is a fundamental visual recognition problem in computer vision and has been widely studied in the past decades. Visual object detection aims to find objects of certain target classes with precise localization in a given image and assign each object instance a corresponding class label[2].

The task of object tracking is crucial in the field of computer vision. It is used to track changes in an object's presence, location, size, and other attributes over time and space in a video sequence[3]. Visual surveillance is highly identified research area including wide area applications in human activity monitoring, public safety in places like banks, shopping malls and private places, automated identification of events of interest, people counting, augmented reality, motion based recognition, autonomous robot navigation and other commercial areas[4].

Videos are basically sequences of images, each of which is called a frame, displayed in fast enough frequency so that human eyes can percept the continuity of its content. However, the contents of two consecutive frames are closely related.

Hence two adjacent frames can be used to track status of objects in the scene as moving or still[5]. It goes without saying that all image processing methods can be used to analyze film in individual frames. Three crucial steps are involved in video analysis: finding intriguing moving objects, following them from frame to frame, and analyzing the object tracks to determine the objects' behaviors[6]. Since object detection typically precedes object tracking, it is frequently necessary to repeatedly detect an object in subsequent image sequences to aid and validate object tracking[4].

Convolution Neural Networks (CNNs) has been widely used in visual recognition from 2012 due to its high capability in correctly classifying images[8]. CNNs were used to estimate object poses in an RGB-D scene in order to represent the mass 3D models. Some used deep CNNs for object recognition on images lacking low-level cues, such as realistic object texture, pose, or background[1]. Combining CNN features is required to produce object proposals, extract CNN feature maps, and execute support vector machine classification[11].

Swarm Intelligence (SI) is inspired by the collective intelligence of decentralized, self-organized systems. A swarm is a population of interacting individuals that can optimize global objectives through collaborative search of the space[9]. The intelligence relies on the networks of interactions among individuals, and between individuals and the environment. There is a general stochastic tendency in a swarm for individuals to move toward a center of mass in the population on critical dimensions, resulting in convergence on an optimum[10]. One of the main techniques in swarm intelligence is Particle Swarm Optimization (PSO). PSO has most often been used as feature selection at feature level fusion[12].

The distribution of all publications and publication per year from 2010 to 2015

with respect to Swarm Intelligence (SI)-based algorithms such as Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Differential Evolution (DE), Bacterial Foraging Optimization (BFO), Artificial Bee Colony (ABC), Glowworm Swarm Optimization (GSO) and Bat Algorithm (BA) method was respectively presented in a Chart[12]. Zhang discovered that the number of total publications related to PSO was higher than the sum of six other algorithms, and the number of publications per year related to PSO was the highest among all seven SI-based algorithms. The study suggested that PSO is the most prevalent SI-based optimization algorithms[13].

Particle Swarm Optimization (PSO) is a meta-heuristic, population-based optimization technique aimed at finding a solution to an optimization problem in a search space. The PSO algorithm was first described by Kennedy and Eberhart in 1995. The principal objective of PSO is to optimize a given function called the fitness function[14]. It utilizes a population of particles with a meta-heuristic procedure to search for an optimum value by trial and error. This procedure has a trade-off in randomness and local search. There is no guarantee that the PSO will be able to obtain the best solution. In addition, the solution is dependent on the searching time. PSO comprises two phases: exploration and exploitation. In the exploration phase, particles are spread so that they can explore the search space. The phase reduces the risk of particles to be trapped at the local optimum, which however results in a slower convergence rate. In the exploitation phase, particles are spread only in a local area that finds the best solution at that time. The phase aims to obtain the optimal solution to achieve a higher convergence rate with the risk of being trapped in the local optimum[15].

For the purpose of object detection and tracking in video datasets, a hybrid paradigm that combined the relative strengths and minimized the relative weaknesses of PSO and CNN, called hybridized Swarm Intelligence Convolution Neural Network (CNN-HPSO) was employed in this study. On the collected video datasets and on a real-time basis in AVi and MP4 video formats, various image processing operations including video preprocessing, frame display, background subtraction, segmentation, and tracking were carried out. The foreground and background elements of the videos were separated into individual frames, and color conversion was done using a preprocessing technique. To determine whether there was a total or abrupt shift in intensity in the videos, background subtraction was used. An enhanced swarm intelligent Convolution Neural Networks technique was used to complete the segmentation and tracking module.

1.2 Statement of the Problem

In recent years, analysis and interpretation of video sequences to detect and track objects of interest had become an active research field in computer vision and image processing. Detection and Tracking includes extraction of moving object from frames and continuous tracking it thereafter forming persistent object trajectories over time. Despite significant efforts in object detection and tracking, an efficient method which detect and track fast moving objects; and provides high computational efficiency (high localization accuracy on small objects under partial occlusions, low processing time, high precision, high sensitivity, high specificity, low false positive rate) have not been developed[16].

Convolution Neural Networks (CNNs) are widely being used in various domains including object detection and tracking but still has overfitting problem and

difficulty in handling object occlusions. One fundamental problem of PSO is that it usually gets stuck in local optima, leading to suboptimal solutions and also suffer from premature convergence. Furthermore, PSO is faced with the challenge of selecting appropriate values for its parameters (the learning rate, inertia weight, and swarm size)[17, 18].

Therefore, in this work a hybrid paradigm that combined the relative strengths and minimized the relative weaknesses of PSO and CNN, called hybridized Swarm Intelligence Convolution Neural Network (CNN-HPSO) was evolved, and was used to detect and track moving objects addressing fast moving objects, low localization accuracy on small objects under partial occlusions, high processing time, low precision, low sensitivity, low specificity, and high false positive rate which are some of the problems associated with the existing methods.

1.3 Aim and Objectives of the Study

The aim of this research is to develop a Swarm Intelligence Convolution Neural Network model that can efficiently detect and track objects.

The specific objectives are to:

i. acquire video sequence file from a standard database online and on real time basis
ii. formulate a hybrid Particle Swarm Optimization Convolution Neural Network model for object detection and tracking system
iii. implement and test the performance of the developed model in (ii) above
iv. evaluate the performance of the developed model against existing models using accuracy, precision, false positive rate, sensitivity, specificity and computation time.

1.4 Significance of the Study

There are many uses for object detection, including in augmented reality, human-computer interface, security and surveillance, video communication and compression, traffic management, medical imaging, and video editing[19, 20]. Object detection is an important aspect in any surveillance applications such as video analysis, video communication, traffic control, medical imaging, and military service[21].

The swarm intelligent CNN technique ensures that improvement of localization accuracy of small objects was achieved, false detection of objects was avoided and only interested objects were classified as objects, else the remaining ones were concluded as a noise or background image. Also, it was believed that the technique ensure efficient detection and tracking of object that is accurate, precise and less computationally expensive[22, 23].

1.5 Justification for the Study

In today's highly computerized digital world the matter of security poses serious concern to organizations, homes and country. With the increase in crime and terror rate globally, automated video surveillance, is the need of the hour. Surveillance along with the detection and tracking has become extremely important. Human detection and tracking are ideal, but the random nature of human movement makes it extremely difficult to track and classify as suspicious activities[24, 25]. The primary objective for this is to detect the suspiciously abandoned object recorded by most of the Close Circuit Television Cameras (CCTV) as a result of fast motion

challenge, occlusion, localization accuracy of small object and time of detection[26].

Object detection, is one of the most fundamental and challenging problems in computer vision, which seeks to locate object instances from a large number of predefined categories in natural images. Deep learning techniques have emerged as a powerful strategy for learning feature representations directly from data and have led to remarkable breakthroughs in the field of generic object detection[27, 28]. Though these CNNs are powerful, they often require high computational costs due to substantial high storage and computational resources. The study was able to fine-tune the parameters of CNN using enhanced particle swarm optimization in order to reduce the computational time, improved accuracy and resolved fast motion challenge.

1.6 Scope of the Study

This research is limited to developing a hybrid paradigm for object detection and tracking system using video datasets acquired from standard database online and real-time basis via YouTube in AVi and MP4 video formats while solving the problem of inability to detect and track moving objects as well as low computational efficiency (low localization accuracy on small objects under partial occlusions, high processing time, low precision, low sensitivity, low specificity, high false positive rate) associated with these existing methods using a technique that combined the relative strengths and minimized the relative weaknesses of PSO and CNN, called Hybrid Swarm Intelligence Convolution Neural Network (CNN-HPSO). The developed hybrid model and existing models (CNN and CNN-PSO) were implemented using MatLab (R2016a) software.

1.7 Limitations of the Study

This study does not take care of eliminating the shadow of the moving objects as well as the reduction or mitigation of noise in the aspect of video preprocessing. Also, it does not address the problem of detecting and tracking multiple objects.

1.8 Operational Definition of Terms

Neural Network: Any system of neurons, whether natural or artificial, is referred to as a neural network.

Convolutional Neural Network (CNN): Feed-forward neural network variations with a unique architecture are known as convolutional neural networks.

Artificial Neural Network (ANN): a subset of machine learning models based on the connectionism-discovered principles of neuronal organization in the organic neural networks that make up animal brains.

Computer Vision: Computer vision is a branch of computer science that focuses on giving computers the ability to recognize and comprehend people and objects in pictures and movies. Computer vision, like other forms of AI, aims to carry out and automate tasks that mimic human abilities.

Moving Objects: An object changing its position with regard to time is said to be moving.

Occlusion: When one object in an image obscures a portion of another object, it is said to be occluded.

Object Detection: Using computer vision, object detection can be used to find instances of objects in pictures or movies.

Object Tracking: In the computer vision application known as "object tracking," a program identifies objects and then keeps track of their movements in space or across

several camera angles.

Swarm: A swarm is a group of interconnected people who work together to search the environment in order to maximize overall goals.

Swarm Intelligent: Swarm intelligence is the collective brainpower of an autonomous, decentralized system.

Particle Swarm Optimization (PSO): The swarm intelligence algorithm's optimization method known as Particle Swarm Optimization is employed in feature level fusion as a feature selection method.

Object Segmentation: Object segmentation is the process of assigning a specific class to each pixel value in a picture.

Video Frame: A video frame is a solitary still image that, when seen alongside the other frames of the movie, causes motion to appear on the playback surface.

Endnotes

[1]B. Daulet, A. Zhilisbayev, A. Kuzdeuov, A. Oleinikov, D. Fadeyev, Z. Makhataeva & H. A. Varol. "*Deep Learning Based Object Recognition Using Physically-Realistic Synthetic Depth Scenes*". **Machine Learning and Knowledge Extraction** 1. no. 3, 2019. 883-903.

[2]W. Xiongwei, D. Sahoo& S. C. Hoi. "*Recent Advances In Deep Learning for Object Detection*". **Neurocomputing. 396**. 2020. 39-64.

[3]W. Pradnya, M. S. Kale, & V. M. Thakare. "*Edge Detection for Moving Object Tracking*". **International Journal**. 5, no 4,2015. 16-19.

[4]P. Divyani & H. J. Galiyawala. "*A Review on Moving Object Detection and Tracking*".**International Journal of Computer Application** 5, no. 3 2015. 168-175

[5]M. Kaushal , B. S. Khehra& A. Sharma. "*Soft Computing Based Object Detection and Tracking Approaches: State-of-the-Art Survey*"**.Applied Soft Computing**. 70, 2018, 423-464

[6]S. A. Daneshyar& N. M. Charkari. "*Biogeography Based Optimization Method for Robust Visual Object Tracking*". **Applied Soft Computing**. 122, 2022, 108802

[7]Y. Zhou, Y. Chen & D. Pi. "*Discovery of Stay Area in Indoor Trajectories of Moving Objects*". **Expert Systems with Applications.** 2021. 114501

[8]T. Shijian & Y. Yuan. "*Object Detection Based on Convolutional Neural Network*". **IEEE Transactions on Pattern Analysis and Machine Intelligence**. 2015.1-5.

[9]T. Thenmozhi& A.M. Kalpana. **"***Adaptive Motion Estimation and Sequential Outline Separation Based Moving Object Detection in Video Surveillance System***". Microprocessors and Microsystems**.76, 2020, 103084

[10]B. Munjal, A. R. Aftab, S. Amin, M. D. Brandlmaier, F. Tombari& F. Galasso. *"Joint Detection and Tracking in Videos with Identification Features"*.**Image and Vision Computing**. 100, 2020, 103932

[11]G. Ross, J. Donahue, T. Darrell& J. Malik. "*Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation*". **In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**, 2014. 580-587.

[12]X. Hongyu, X. Lv, X. Wang, Z. Ren, N. Bodla& R. Chellappa. "*Deep Regionlets for Object Detection*". **In Proceedings of the European Conference on Computer Vision (ECCV)**, 2018. 798-814.

[13]Z. Yudong, S. Wang& G. Ji. "*A Comprehensive Survey on Particle Swarm Optimization Algorithm and its Applications*". **Mathematical Problems in Engineering.** 2015. 1-20

[14]S. Saptarshi, S. Basak& R. A. Peters. "*Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives*". **Machine Learning and Knowledge Extraction** 1, no.1 2019. 157-191.

[15]J. WikanKuncara& L. K. Muslim. "*Optimization of Decision Tree Algorithm in Text Classification of Job Applicants Using Particle Swarm Optimization*". **In 2020 3rd International Conference on Information and Communications Technology (ICOIACT**). 2020. 201-205.

[16]L. Qing, S. Hu, K. Shimasaki& I. Ishii. "*An Active Multi-Object Ultrafast Tracking System with CNN-Based Hybrid Object Detection*". **Sensors** 23, no. 8, 2023. 41-50.

[17]Z. Jinghua, C. Li, Y. Yin, J. Zhang & M. Grzegorzek. "*Applications of Artificial Neural Networks in Microorganism Image Analysis: AComprehensive Review from Conventional Multilayer Perceptron to Popular Convolutional Neural Network and Potential Visual Transformer*". **Artificial Intelligence Review** 56, no 2, 2023. 1013-1070.

[18]K. Najwa, F. BenSaid, R. Fdhila, R. Fourati, A. Hussain& A. M. Alimi. "*A Novel Approach of Many-Objective Particle Swarm Optimization with Cooperative Agents Based on an Inverted Generational Distance Indicator*". **Information Sciences** 623, 2023. 220-241.

[19]S. Balakrishna&A. A. Mustapha. "*Progress in Multi-Object Detection Models: A Comprehensive Survey*". **Multimedia Tools and Applications.** 82, 2023, 22405-22439

[20]Y. Liu, P. Sun, N. Wergeles& Y. Shang. *"A Survey and Performance Evaluation of Deep Learning Methods for Small Object Detection".***Expert Systems with Applications**. 172, 2021, 114602

[21]A. Parakh, S. S. Kahlon, N. Bisht, P. Dash, S. Ahuja& A. Goyal. "*Abandoned Object Detection and Tracking Using CCTV Camera*". **In Information and Communication Technology for Sustainable Development. Springer, Singapore**, 2018. 483-492

[22]L. Li, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu& M. Pietikäinen. "*Deep Learning for Generic Object Detection: A survey*." **International Journal of Computer Vision.** 128, no. 2, 2020. 261-318.

[23]X. Zhao, G. Wang, Z. He&H. Jiang. "*A Survey of Moving Object Detection Methods: A Practical Perspective*.**Neurocomputing**. 503, 2022, 28-48.

[25]S.D. Roy &M. K. Bhowmik."*A Comprehensive Survey on Computer Vision Based Approaches for Moving Object Detection*".**In Proceedings of the IEEE Region 10 Symposium (TENSYMP), Dhaka, Bangladesh**,5–7June2020;1531–1534.

[26]Y. Yu, L. Kurnianggoro, &K.H. Jo. "*Moving Object Detection for a Moving Camera Based on Global Motion Compensation and Adaptive Background Model"*.**Int. J. Control Autom. Syst.**17, 2019, 1866-1874.

[27]S. Wu,O. Oreifej& M. Shah."*Action Recognition in Videos Acquired by a Moving Camera Using Motion Decomposition of LagrangianParticleTrajectories*".**InProceedingsoftheIEEEInternationalConferenceonComputerVision(ICCV),Barcelona,Spain,**6-13, November2011;1419-1426.

[28]W. Rahmaniar, W. J Wang& H. C. Chen."*Real-Time Detection and Recognition of Multiple Moving Objects for Aerial Surveillance*".**Electronics.** Vol 8. 2019, 1373.

Chapter Two

Literature Review

2.1 Introduction

Both commercial and military electronics make extensive use of object detection and tracking, the topic has been extensively researched for many years. Therefore, there are a wide variety of motion detection and tracking methods in the literature. Some of the algorithms are well developed and have a very satisfactory performance; nevertheless, still there are some unsolved problems in the area. Noise in images is one of the problems for a typical tracking system.

In real life scenarios, the input video may be noisy and a robust tracking system should be tolerant to noise up to some extent. Desired features on the image might be lost due to blurring. Changes in illuminations are another challenging situation for surveillance applications. Due to the angle of the light source and different type of whether conditions, pixels of the same scene may change dramatically. Thus a robust tracking system should withstand such kinds of variations.

The job of object tracking is crucial in the area of computer vision. There is a lot of interest in object tracking algorithms due to the ubiquity of powerful computers, the accessibility of high-quality, low-cost video cameras, and the growing need for automatic video analysis. The issue of estimating an object's trajectory in the image plane as it moves around a scene is the basic definition of tracking[1].

Furthermore, visual surveillance has been a very active research topic in the last few years due to the growing importance for security in public places. A typical automated visual surveillance system consists of moving object detection, object classification, object tracking, activity understanding, and semantic description[2].

Moving object detection is not only useful for object tracking initialization for a visual surveillance system, it is always the first step of many different computer vision applications, for example, automated visual surveillance, video indexing, video compression, and human machine interaction. Since subsequent processes are greatly dependent on the performance of this stage, it is important that the classified foregrounds pixels accurately correspond to the moving objects of interests[1].

A lot of research effort is devoted to detecting moving object with various processing steps and core algorithms. Due to its simplicity image differencing is a popular method for motion detection[2,3]. The reference image is subtracted from the current input picture to create the difference image, and the resulting image is thresholded to detect moving objects. Many researchers conducted surveys and published experiments on a variety of selection factors to meet application-specific standards for false alarms and misses.

2.2 Challenges in Detecting and Tracking Moving Objects

There are a lot of common difficulties and problems encountered when performing moving object detection. Some examples of these are illumination change and repetitive motion from clutter such as waving tree leaves. Due to these problems with dynamic environmental conditions, moving object detection from the background becomes very challenging. The major challenges for background subtraction are how to update the background model, and how to determine the threshold for classification of foreground and background pixels[4]. Similarly, most of the generated background models are not applicable in some scenes with some specified issues, including but not limited to six terms[4]:

1. Flexibility to illumination change: The background model should adapt to gradual illumination changes.
2. Dynamic textures change: The background model should be able to adapt to dynamic background movements, which are not of interest for visual surveillance, such as moving curtains.
3. Noise acceptance: The background model should demonstrate appropriate noise immunity.
4. Susceptible to clutter motion: The background model should not be sensitive to repetitive clutter motion.
5. Bootstrapping: The background model should be properly generated at the beginning of the sequence.
6. Expedient implementation: The background model should be able to be set up fast and reliably.

The inability of most techniques developed by various researchers to meet the listed requirement makes it very difficult to implement them in real life application. Recently, several contributions have been proposed and successfully demonstrated for foreground detection and tracking. However, these algorithms need to resolve the difficulties such as radical changes and target drift encountered during tracking process. Main challenge involved in motion tracking algorithm is to estimate object motion as more precisely and efficiently as possible[5].

2.3 Concept of Object Detection and Tracking

Object detection is broken down into several phases, which are depicted in Figure 2.1. To obtain more precise results in video surveillance, there are various methods. For various environmental conditions, each step has a variety of algorithms[4].

There are many ways to get exact results in the shortest amount of time because security is involved. Shadow object detection becomes a challenging and important issue as a result of environmental factors like lighting shifts.

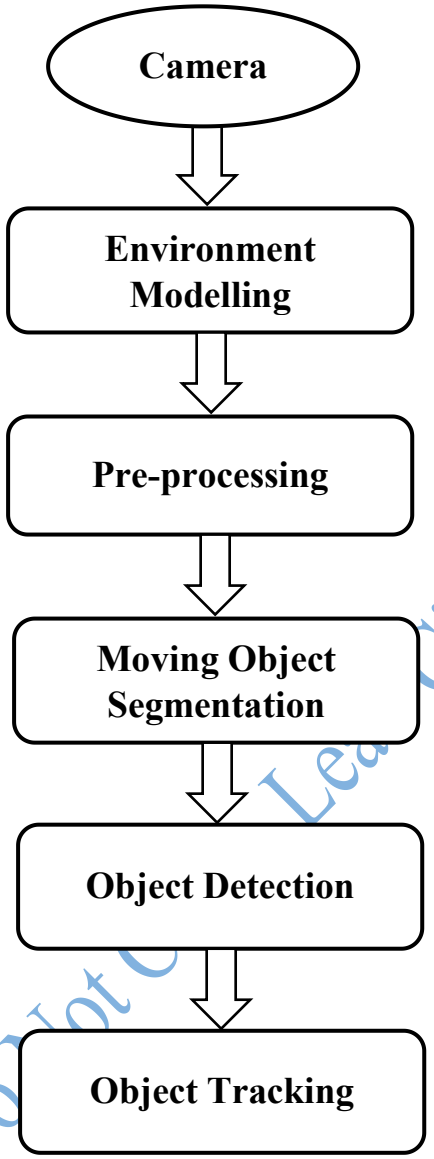


Figure 2.1: General Block Diagram of Object Detection in a video surveillance system[6]

Utilizing data from a single frame is a well-known method for object identification. Now, the descriptions of each step are as follows:

2.3.1 Environmental Modelling

In video surveillance systems, rigid and non-rigid objects are the fundamental categories for object detection. Algorithm implementation varies depending on the nature of the object. The study being reviewed presents a survey of algorithms for human and vehicle detection and monitoring. Additionally, there are 2D and 3D models for environment rendering. But in this case, the author only looked at 2D images[3]. Different methods for frame subtraction are used in scene modeling. Here, the background value is changed after subtracting each video frame. To find and follow moving items in the video, a tracking algorithm based on adaptive background subtraction is used.

Here, moving object detection is accomplished by determining which portions of each video frame are stationary or moving by subtracting the background picture from each frame. The background image is constantly changing as a result of shifting moving items and other environmental disturbances.

The method offers precisely defined object bounds. Using a Gaussian Mixture model (GMM), the object and background are modeled in this case, and an approximate contour that takes into account the object edge feature is extracted. Following that, during form context matching, the object's states, including translation, rotation, and scale, are estimated. There are three different kinds of background subtraction: pixel level, region level, and frame level[3].

2.3.2 Pre-Processing:

Real-time recognition in video requires pre-processing. Pre-processing is required because there are many possibilities for noise in real-time detection systems. Real-time systems need extremely accurate algorithms for detection because they are affected by environmental factors and lighting conditions. The image must first be filtered before the object detection methods are used. Additionally, noise causes small gaps in the images, so we must eliminate it to obtain a precise extracted image for additional processing[7].

2.3.3 Moving Object Segmentation

Segmenting the background or motion is another crucial step in the object recognition process. The challenging job in a video surveillance system is to precisely extract the moving object. After the object has been extracted, various algorithms are used to identify it. The technique of frame differencing is used to find the object. To find the moving item, the MATLAB image acquisition toolbox was used. But this approach also has the obstruction issue. In order to extract the moving object from a continuous video frame, the adjacent frame difference technique is used[7]. Following that, a Canny edge detector is developed in MATLAB to determine the object's boundaries. This also implements object rearrangement. Background subtraction, temporal differencing, and optical flow method are the three main kinds of video differencing.

Various other techniques are also useful in video surveillance[4]. Here, motion segmentation is a challenging job due to the cameras being mounted on the intersections. Occlusion is a serious issue because so many moving objects span the road[1]. As a result, it was unable to distinguish the items, because there are so many

different algorithms for correctly detecting an object, including blob-based, contour-based, shape-based, and other approaches. The study of video surveillance is a very broad subject. As security concerns grow, video algorithms and exact software can produce precise results more quickly. One needs object detection techniques that can find things quickly because time is the most valuable resource in video surveillance.

2.3.4 Object Detection

Tracking and detecting an object in a video surveillance system is a challenging job. It can be used to identify an object in surveillance footage. Numerous environmental variables exist, including sunlight, traffic lights, weather impacts, etc. As it had been observed, the background removal phase of object recognition and tracking is the initial stage. For object identification, a centroid weighted kalman filter is employed[4].

This algorithm uses the centroid to determine the precise location of the object after subtracting the background picture from the foreground image. Although the kalman filter is a better choice for object detection than other algorithms, it has lower accuracy. Pedestrians are regarded as objects. The primary concern is regarded to be the safety of pedestrians. A 6-D vision algorithm is employed to identify pedestrians. In their work, points are used to determine an object's or pedestrian's depth value, and pedestrians are then identified using the spatial position data. To obtain the collision avoidance alert, situation analysis and the vehicle control modules are combined. Lateral control for escape is also used when using vehicle control. Because it combines two algorithms, it produces precise results with no false detection.

Motion recognition, segmentation, and object classification are all parts of object detection[5]. In order to identify real object findings, these two are combined.

There are numerous methods available for classifying and segmenting motion. As has been explained, motion segmentation is basically background subtraction or frame subtraction. Once the object or the motion is extracted, the classifiers are used to detect a particular object. Object detection methods are classified basically as: Motion based classification and Shape based classification.

Skin color based object detection is performed. In various lighting conditions, skin color is extracted for better result. Hough Transform is used to detect pedestrian's head. But it cannot be implemented in crowded scenario because probability of occlusion is more in crowded environment. Blob based and contour based techniques are also satisfying.

Blob of a human is taken from human cyclic motion analysis. This method uses bayes classifier to get better result for the position of the person. The combination of blob motion and HOG techniques give some good results. Contour based detection is also one of the good options for accurate result. Firstly shape matching approach is implemented using contour extraction method. Canny edge detector is used to get the edges of the object after that the contour is extracted which is rough plot of the object. The SVM classifier is also good classifier for the object detection.

2.3.5 Object Tracking

Object tracking is the next step to the object detection. A very important question in the field of intelligent transportation system is to prevent pedestrians from being hit by vehicles. So, a vision-based approach that can identify pedestrians resourcefully and automatically is in need. Video surveillance is having very wide

range of applications which includes different techniques to recognize the objects at different environment conditions[8].

Object tracking is useful as a part of safety of the pedestrian or vehicles. Therefore detection of the object is applicable at crossroad, in the crowded environment, at traffic signals, various lighting conditions, and etcetera. Several algorithms are available to get accurate results. In object tracking, objects are tracked in every consecutive frame sequences for different application in video surveillance system[9]. The basic object tracking is divided into three categorize: region-based tracking, contour-based tracking, feature-based tracking, and model-based tracking.

2.3.5.1 Region-Based Tracking

Colour and texture characteristics of the item are taken into account when tracking it. Utilizing a stable background model and motion segmentation, tracking is performed in this case. In order to track the items, human bodies or vehicles are treated as blobs and background subtraction techniques are used. To address the silhouette and occlusion issue, color and gradient data are used. Blob-based monitoring is put into practice. Here, the background subtraction method-a precise technique for extracting the object-is used to identify an object. Extraction of the foreground blob yields the precise item. After the detection, mean-shift algorithm is applied using particle filter. The mean shift algorithm uses an iterative process to find the density extrema or modes of a given distribution. For a better understanding of the tracking outcome, the findings from both algorithms are displayed in blob and HOG. This approximates accuracy, but there are other methods that can produce more exact findings[10].

For object tracking, an adaptive background subtraction technique is used. Here, the background subtraction technique is employed to identify an object. To determine the object's moving location, the current image is compared to the background image. Then, only moving areas are taken into account for target detection. To find linked regions in binary images, labeling of connected components is used. Based on pixel connectivity, it scans a picture and divides its pixels into component parts. To identify and locate the object, a centroid is used.

2.3.5.2 Contour-Based Tracking

This technique makes use of the background pictures' borders or edges. Boundary characteristics offer more accurate shape data. The information given by the object boundaries is the foundation of this technique. There are numerous methods available to find an object's boundary or edges, including the Harris corner detector and Canny edge detector. The contour is extracted depending on shape. Using a clever edge detection technique, a rough contour is first extracted using edge pixel values. The parameters are then approximated using shape matching after that. A discrete collection of points sampled from a shape's contours serves as its representation[5]. The contours of these spots may be internal or external. Tracking is ultimately completed by elastic shape matching for extracting the precise contour. The advantages of this strategy include translation, biased occlusion management, and rotation. However, in a congested outdoor setting, a more robust approach is needed. The shape of the item is detected and the entire object is tracked using contour-based tracking[11].

2.3.5.3 Feature-Based Tracking

To find and follow the object in different environmental conditions, various features are used. The process of feature-based tracking involves extracting various characteristics, classifying them, and comparing them to the expected outcomes. Different characteristics, such as height, width, area, and so forth, are taken into account. Using the background subtraction technique, moving region segmentation removes the background image from the current image. For a multicolor world, GMM (Gaussian mixture model) is used here. The method of extracting a phrase by filtering it through the following process. The spatial location, which is the average position of all the pixels in the area, is provided by the centroid of the bounding box. To extract the object, factors like height, area, breadth, and aspect ratio are used. The process of tracking involves anticipating the pedestrian search window and then comparing the detected pedestrians to the present context. The Kalman filter is utilized for object detection. Although these techniques can manage occlusion, they take a moderately shorter amount of time to detect and track the object[12].

When shape-based methods are unreliable, feature-based tracking is helpful. Weather and environmental variations have an impact on the formulas. We get more precise results when we measure an object's characteristics while it is in use. Shapes change based on an object's location at various levels. However, it has a drawback in that the 3-D model does not produce satisfactory outcomes.

2.3.5.4 Model-Based Tracking:

Basic models that are taken into account for object recognition and tracking are 2-D and 3-D models. Construction of human body models, representation of previous knowledge of motion models and motion constraints, prediction, and search

strategies are all problems that are generally involved in model-based human body tracking. A physical copy is also used as a guide. The shape of the vehicles is preset in vehicle detection. As a consequence, the current results are compared to the predicted results' template on that basis. One can specify the shape of the detected object, just like in model-based method. As a result, tracking the item is made simpler. The human body can also be used for the same purpose[12].

Since the human body's shapes, such as its head, arms, and legs, are fixed, we can train the classifier according to those shapes and more precisely apply the tracking algorithm. The only challenge with the model-based method is the extremely challenging construction of the human shape in 3-D models.

2.4 Tracking Process

Finding an object's motion path over time by determining its location in each frame of the video is the process of tracking an object in a video. One of the most well-liked areas of study in the field of computer vision is object tracking. Despite being a well-studied issue, it is still difficult in many ways. The development of trackers for particular object classes, such as faces or humans, has advanced greatly over time, but trackers for generic objects remain a challenging field. For example, noise in images, complex motion, and complex object shapes are some examples of objects whose appearance dramatically changes due to, for example, distension or changes in light[13].

Imagine observing a random moving object that abruptly modifies its motion or external characteristics like form or color. Along with numerous other scenery elements that might exhibit comparable behaviors. Even for the human eye, it would

typically be challenging to keep watch of the object. In light of this, it appears that a computer would find it nearly impossible to handle this job. Having said that, there are methods to make tracking easier by putting some restrictions in place. Nearly all algorithms make the assumption that the object being monitored moves smoothly and without any abrupt changes. Other methods of streamlining the job include knowledge of the object's appearance in advance and constant velocity or acceleration.

Object representation, object detection, and object tracking are the three main stages in the process of creating an object tracker. The focus of this task is object tracking, and this section will review various methods for doing so. However, since these are crucial steps in the process of successful object tracking, it will also quickly discuss the area of object representation and object detection. In this manner, the reader is given a better foundation and comprehension of the procedure for creating an object tracker[13].

2.4.1 Object Representation

An object of interest must first be represented in a manner that is understandable to a computer in order to be tracked. Typically, properties pertaining to appearance and shape serve as the foundation for the depiction. Both appearance representation and shape representation can be used independently or in combination to describe an object. The application domain, goal, and other external factors decide how the object or objects should be represented. The representation in turn dictates which tracking method is most appropriate. To recapitulate, an object's representation is typically said to be made up of its shape and/or appearance[13].

$$\text{Object representation} = \text{Shape} + \text{Appearance} \tag{2.1}$$

2.4.2 Shape Representation

There are various ways to characterize or describe an object's shape so that computations for detection and tracking can be made; some representations are better suited for particular kinds of objects. Every type of representation model has benefits and drawbacks, and these frequently change based on the application domain and object type. Points, geometric figures, silhouettes, contours, articulated shape models, and skeletal models are examples of common methods to represent shapes[12]. The following introduces these shape representations along with a succinct explanation of their intended use.

2.4.2.1 Points

Figure 2.2(a) and 2.2(b) both show the object of interest as a solitary point or as a collection of points. When tracking multiple objects in a video and there is interaction between the objects, such as partial or complete occlusion, using a collection of points to represent the objects can be problematic. Misdetections are readily caused by the inability to keep track of which point belongs to which object. As a result, it works better for small, straightforward things that can be represented by a single point[12].

2.4.2.2 Geometric Shapes

Figures 2.2(c) and 2.2(d) show how primitive geometric shapes like a rectangle or an ellipse are used to depict shapes. A popular strategy for both rigid and non-rigid objects is to use simple shapes as representations, though it works best for straightforward, rigid objects. Usually, the objects in movies are not quite as precise and straightforward as these kinds of shapes[12]. As a result, it is frequent for portions

of the objects to be left outside or for portions of the backdrop to be incorporated into the shape template, which could lead to tracking issues[14].

2.4.2.3 Silhouette and Contour

It is known as contour representation to use an object's outline or boundary to symbolize it (see Figures 2.2(g) and 2.2(h)). Using the area inside the contour is known as silhouette depiction (Figure 2.2(i)). It is simpler to depict complex and/or non-rigid objects using a contour or silhouette. It is a versatile model that can depict a wide variety of object forms[14].

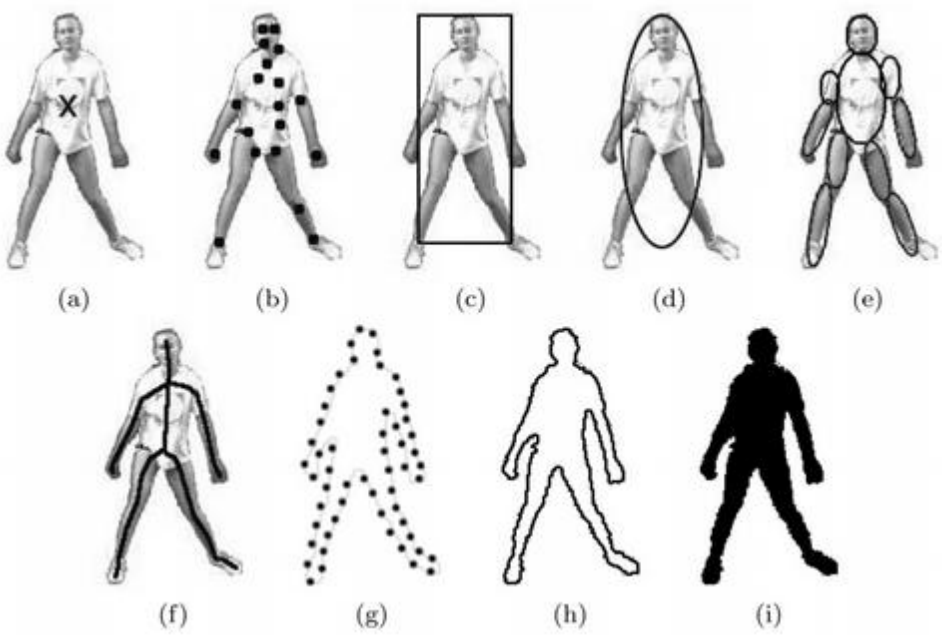


Figure 2.2: Different approaches regarding shape representation[14]

2.4.2.4 Articulated Shape Models

An articulated object is one that is made up of various components that are connected by joints. In Figure 2.2(e), a human body with its head, torso, limbs, legs,

hands, and feet is an example of how this could be used. Simple geometric shapes like ellipses, which is a model discussed above, can be used to symbolize each component.

2.4.2.5 Skeletal Model

It is possible to derive the skeleton of an object by using the object's silhouette and the medial axis transform (Figure 2.2(f)). Although this approach is frequently employed in object recognition, it is less frequently mentioned in object tracking literature.

2.4.3 Appearance Representation

There are many ways to depict an object by appearance, much like the shape representation. Probability densities, templates, active appearance models, and multi view appearance models are a few popular methods of depicting an object[14]. Below is a summary and explanation of these.

2.4.3.1 Probability Densities of Object Appearance

The chance that a random variable will fall within a specific range of values is expressed by a probability density function. An estimation of the probability densities of object appearance features can be computed using the interior area of an image defined by the shape model, such as by a contour. For instance, color or texture are examples of appearance characteristics. The probability distributions can either be nonparametric, like histograms, or parametric, like the Gaussian distribution[8] (or normal distribution).

2.4.3.2 Templates

Templates are created using silhouettes or basic geometric forms. The method has the benefit that the templates can store both spatial and graphical information. Since templates only encode the appearance features from one view, using them to

model objects whose poses don't change can be problematic for objects that dramatically differ in appearance from different views. Problems can also occur when the appearance features change noticeably while the tracking is being done. One illustration of this is the sensitivity of color features to variations in lighting.

2.4.3.3 Active Appearance Models

The shape of an object is determined by a collection of landmarks, which may be located on the object boundary or inside the object area. By storing an appearance vector for each landmark, the object appearance is concurrently modeled. This may manifest itself in terms of color, texture, or varying strength. This model does need a training phase where a set of samples is used to learn the shape and related appearance.

2.4.3.4 Multiview Appearance Models

These models encode various views of the item, unlike templates. There are various methods for doing this; one illustration is the creation of a region using the provided views. Principal Component Analysis (PCA) and Independent Component Analysis (ICA) are two examples of subspace methods that have been applied in this context[8].

2.5 Feature Selection

The selection of an object's distinguishing characteristics is an essential component of object monitoring. Given their close relationship, the feature selection is significantly influenced by the choosing of object representations[12]. For instance, contour-based representation frequently uses object boundaries as features. The following is a summary of some popular feature choices.

2.5.1 Edges

An object's border and the backdrop are typically clear to see. In other words, it is typically not difficult for the human eye to identify the edges of things in an image. This is as a result of boundaries producing significant shifts in image intensities. When edges are used as the representing feature, tracking algorithms that follow an object's boundaries can detect these variations in intensity[15]. Compared to, say, hue, edges are less sensitive to changes in illumination.

2.5.2 Optical Flow

This refers to what is sometimes referred to as a motion feature—apparent motion of brightness patterns in a visual image. Almost all people encounter optical flow as a visual occurrence every day. Looking out the window while driving gives the impression that things, like trees and structures, are moving backwards. Optic movement causes this motion to appear. By tracking each pixel's velocity between frames, the apparent motion can be calculated. These computations make use of the brightness constraint, which guarantees that corresponding pixels have the same brightness across frames. Figure 2.3 offers an instance of this[16].

2.5.3 Colour

Different color spaces can be used to hold the data from various frames. In picture processing, color is typically represented by RGB (red, green, and blue), though YCbCr and HSV are also occasionally used. Since the apparent color of an item is directly influenced by the illumination factor, one issue with using color as a feature representation is its sensitivity to changes in illumination[9].

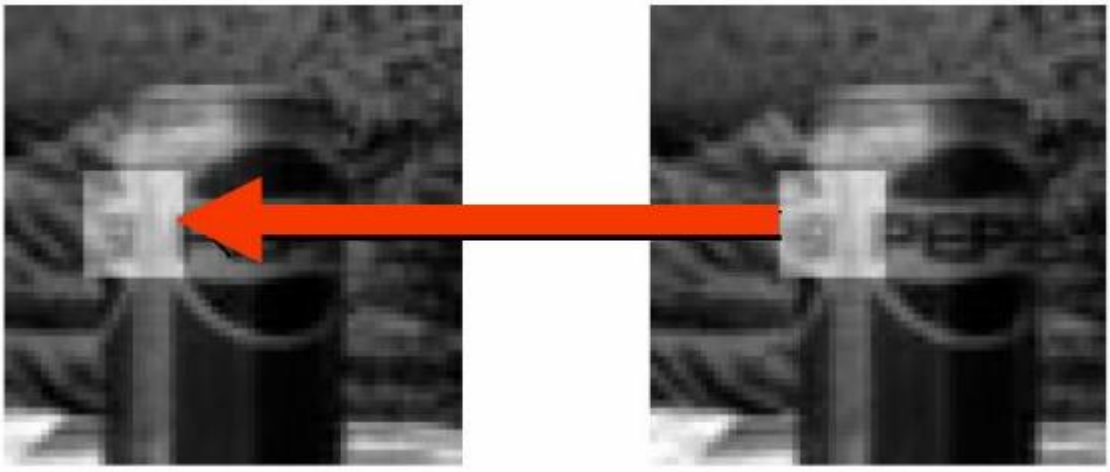


Figure 2.3: Despite possible changes in pixel location, brightness in a particular area stays constant[16]

In addition to illumination, the object's reflectance characteristics also have an impact on apparent hue.

2.5.4 Texture

A surface's intensity variation can be used to quantify pertinent properties like smoothness and regularity. The descriptors, which can be generated in a variety of ways, must go through a processing phase. A texture measure, one type of texture descriptor, uses filters to describe level, border, spot, wave, and ripple.

2.6 Segmentation Techniques for Object Detection and Tracking

The division of an image into a number of segments or areas is called segmentation. The generated segments will collectively cover the complete image and share perceptual characteristics, such as texture or color. This is useful for identifying borders and objects. The goal of motion segmentation is to break down a movie into its background and moving elements[16]. This breakdown is a prerequisite for many computer vision processes. In the processing and analysis of image sequences, segmentation of the items in the sequence is crucial. Once the moving items have

been located or removed, they can be used for a variety of tasks. The three major problems with a motion-based segmentation algorithm are usually present. Data primitives, also known as areas of support, are the first problem. These primitives can be single pixels, corners, lines, blocks, or regions. Motion models or motion depictions are the second issue. Segmentation criteria are the final problem. Motion segmentation is so dependent on motion. One of two methods for classifying video-based segmentation algorithms is based on how they depict motion. There are different approaches for this method. Some important segmentation methods for moving object detection and tracking includes: background subtraction method, active contour model and threshold, temporal and spatial, Edge Detection and Optical Flow[3, 16].

2.6.1 Background Subtraction Method

A common class of methods for separating foreground objects from the background in a series of video frames is background subtraction. The process of subtracting the experimental picture from the estimated image and thresholding the outcome to produce the foreground objects is known as "background subtraction," and it is a straightforward process. Fundamental logic is that difference between current frame and a reference frame. Reference frame also called background image. The selection of background (background modeling) can be classified into two categories which are recursive and non-recursive techniques. Recursive technique includes frame differencing, linear predictive filter, median filter, and nonparametric model. Non-recursive technique uses method of sliding window approach for background estimation[13]. Non-recursive techniques are highly adaptive they do not depend on history beyond those frames stored in buffer. This method is not suited for the

background is dynamic, illumination changes or in the presence of shadow. This technique can be used to find moving targets in a security video. This technique allows us to precisely monitor or identify the object. Under this background subtraction technique, there are four steps:

1. Preparation
2. Historical modeling
3. Background recognition, and
4. Data verification

In this technique, a series of video frames is divided into the foreground and background using some logic, the Gaussian Mixture Model and Frame Variation[13].

2.6.1.1 Gaussian Mixture Model

For every background subtraction method, background modeling is crucial. The backdrop model protects against abrupt background changes, but it is sufficient to recognize all moving objects in a series of video frames. The values of a particular pixel are shown as a mixture of Gaussians. At each Iteration Gaussians are evaluated, determine which one is mostly likely compared to the background. Pixels that do not match "background Gaussians" are classified as foreground object.

2.6.1.2 Frame Difference

Frame differencing is the process of determining the moving objects from the present frame and a reference frame using a background image. This method identifies the presence of moving object by considering the difference between two consecutive frames. By subtracting second image from the first image frame using image subtraction operator in consecutive frame get the desired output. It is very efficient method for detecting gray level changes between images by using frame

differencing algorithm[13]. The algorithm may be subdivided into three parts. Initial step is the selection of perfect reference or background. Second step is the arithmetic subtraction operation and the last or third step is the selection of a suitable threshold. Reference image can be selected as a frame which is temporally adjacent image from a dynamic sequence. This method lacks in obtaining the complete contour of the object. The benefits of frame differencing are as follows:

1. For each pixel, we can choose a distinct threshold.
2. Adapting to time.
3. Offers quick healing.

2.6.2 Contour and Threshold Method

2.6.2.1 Active Contour Model

An object shape can be defined using an active contour model, also known as snakes, in a series of video frames. It will have a dynamic depiction of the bounding contour. Moving target segmentation begins with the initial ACM (active contour model) investigation, which is accomplished using the image difference technique. The edges of moving regions can be used as the initial ACM once the moving regions in the image have been identified. For the extraction of contours from moving objects, active contour methods have been used[15]. Many segmentation algorithms can benefit from using this contour method to plainly capture moving objects. For instance, the background subtraction technique uses contours to plainly show the foreground object. It offers the following benefits[15]:

1. An object with a clear background can be segmented.
2. It can be applied to monitor moving objects in space.

2.6.2.2 Threshold Method

The thresholding technique is the most basic form of object segmentation. The selection of the threshold value is the key to this technique. The threshold algorithm separates the background pixels from the motion object pixels based on the variations in their gray values[15]. The simple method is to create a pixel-by-pixel difference between real-time images that are being captured frame by frame and an image that has the same background as the real-time image above but no moving objects, i.e., the gray value of each real-time image's pixel minus the gray value g(i,j) of the corresponding pixel of the background image, to produce an image with a white background. When the threshold value is too high, too many target spots are mistakenly classified as background; when the threshold value is too low, it works in the reverse way. But it has the following benefits[10]:

1. It's fine and in excellent shape.
2. Can easily distinguish moving objects.

2.6.3 Temporal and Spatial Segmentation

For the purpose of comprehending images and effectively manipulating image data, image segmentation offers a potent semantic account of video imagery. For image coding applications, where video data is simply described as a collection of moving layers, spatial-temporal segmentation is used to generate a layered image representation of the video. For instance, spatial and temporal segmentation are required for identifying and measuring human movement. In essence, the spatial segmentation is a tracking procedure that generates a motion vector for each frame that contains a collection of joint angles. The CHMR (Continuous Human Movement

Recognition) system used for temporal segmentation makes an effort to infer the movement technique that could have generated the observed sequence of motion vectors. The video sequence is divided into "scenes or shots" by the standard methods to temporal segmentation, which are primarily based on abrupt changes in image appearance. Other methods segment the video into smaller sequences that capture various events or activities and are behavior-based. A shot is a continuous series of photos captured by a single camera.

With the aid of contour-based techniques like the Canny and Sobel Edge Detector, spatial segmentation is accomplished. With the aid of clustering points in the moving object, temporal segmentation handles. The following are some benefits of spatial-temporal segmentation: 1. Locate and follow a moving item and 2. Segmented items will be clean.

2.6.4 Edge Detection Algorithm

In computer vision, edge detection is a well-established discipline. Given that there is frequently a spiky adjustment in intensity at the region borders, region boundaries and edges are closely linked. As a result, another segmentation approach was built on edge detection methods. However, one requires closed region boundaries in order to segment an object from a movie. The boundaries between such things are the sought-after edges. Edges obtained from edge detectors can also be subjected to segmentation techniques. The segmentation procedure heavily relies on edge information. Any segmentation approach can be used with these methods to track and identify the object in the video. The difficult but crucial job of segmenting moving objects falls under computer vision. There are numerous uses for it, including video communication, traffic monitoring, people tracking, content-based picture coding, and

image compression. Many segmentation techniques for moving objects are built on moving-edge detection; this includes[3, 17]:

1. Sobel Edge Detector
2. Canny Edge Detector

2.6.4.1 Sobel Edge Detection:

The most widely used edge detection methods are for contour-based segmentation[10]. Due to their versatility and effectiveness, Canny and Sobel operators are the most frequently used edge detection techniques. Up until the creation of edge detection methods with a formal foundation, like Canny edge detection, the 2-D Sobel operator was the most popular edge detection operator. It gained popularity because, generally, it performed better than other edge detection operators available at the time, like the Prewitt operator. A root mean square (RMS) estimate of the noise is also used to calculate the thresholds for the 2-D Sobel to produce the binary image. Sobel edge detection has the following advantages[3]:

1. Easy to track 2D object and match the moving objects.
2. Used in many segmentation techniques to avoid the noise in the moving frames.

2.6.4.2 Canny Edge Detection

For moving object segmentation, moving-edge detection has captured people's interest. The processing of the background edge map, present frame edge map, and frames difference edge map results in the generation of moving-edge points. With the help of the Canny edge detector, which uses Gaussian convolution to reduce noise, these spatial domain edge maps were created[3].

For images, the algorithm is composed of 5 steps:

1. Smoothing: picture blurring to reduce the amount of noise.

2. Finding gradients: Where the gradient of the image has high magnitudes, the edges should be noticeable.
3. Non-maximum suppression: The only edges that should stand out are local peaks.
4. Double thresholding: By thresholding, potential edges become invincible.
5. Edge tracking by hysteresis: All edges that are not linked to a very specific (strong) edge are suppressed to arrive at the final edges.

The basic idea of canny operator is to use the first order derivative of 2-D Gaussian function in any direction as a noise filter. The Canny algorithm has difficulty in treating images which contain the salt and pepper noise, and it does not have the ability to adapt in the variance of the Gaussian filtering[9].

The canny edge detector is used to identify the edges of the object and their traces to detect the object. It is the most common and frequent method used for the object detection for its curve let transforms property. It determines the edges of the object more accurately than other operators. Because of the canny edge detector is susceptible to noise in raw unprocessed image data, it uses a filter based on a Gaussian, where the image is convolved with a Gaussian filter**.** The result will be a blurred version of the original which is not affected by a single noisy pixel to any significant degree. An edge in an image may point in various directions, so the canny edge algorithm uses four filters to detect vertical, horizontal, and diagonal edges in the image[3,10].

The Canny operator works in a multi-stage process. First of all, the image is smoothed by Gaussian convolution. Then a simple 2-D first derivative operator (somewhat like the Roberts Cross) is applied to the smoothed image to highlight

regions of the image with high first spatial derivatives. Edges give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as non-maximal suppression. The tracking process exhibits hysteresis controlled by two thresholds: T1 and T2with T1 > T2. Tracking can only begin at a point on a ridge higher than T1. Tracking then continues in both directions out from that point until the height of the ridge falls below T2. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments. Canny Edge detection techniques has the following advantages:

1. Less Sensitive to noise**:** Its uses Gaussian filter which removes noise at great extent as compared to above filters. Hence, noise can be removed cleanly.
2. Remove streaking problem: The classical operators 'like Robert uses single thresholding technique but it results into streaking. Streaking means, if the edge gradient just above and just below the set threshold limit it removes the useful part of connected edge, and leave the disconnected final edge. To overcome from this drawback canny detector uses 'hysteresis' technique which uses two threshold values t-low and t-high as discussed above in canny algorithm.
3. Good localization**:** LoG operators cannot find edge orientation while canny operator provides edge gradient orientation which results into good localization. The moving objects are simple to follow and connect as a result. This study examines all of its benefits for detecting moving objects.

2.6.5 Optical Flow Segmentation Method

The job of segmenting moving objects from a video sequence is crucial because it has uses in areas like object recognition, video compression, and video

surveillance. By using optical flow, it is possible to identify each point's apparent motion in relation to the image plane. The camera's movement in relation to the scene and the movement of the corresponding 3D point in the scene both contribute to this motion. A vector motion field called optical flow (OF) describes how the apparent velocities of the brightness patterns in a sequence are distributed. Optical flow is the distribution of apparent velocities of movement of brightness patterns in an image. Optical flow can arise from relative motion of objects and the viewer. Consequently, optical flow can give important information about the spatial arrangement of the objects viewed and the rate of change of this arrangement. Discontinuities in the optical flow can help in segmenting images into regions that correspond to different objects[11].

The high sensitivity to noise and the high computational cost of such methods were historically their major drawbacks. Currently, optical flow is widely used due to the fast processing speeds of computers and advancements made by study.

One of the most important areas of optical flow study now involves motion estimation and video compression[17]. While it may appear that the optical flow field is similar to a dense motion field derived from motion estimation techniques, optical flow is the study of not only the determination of the optical flow field itself, but also its use in robotics researchers in many areas such as: object detection and tracking, image dominant plane extraction, movement detection, and robot navigation. Finally, optical flow is a good indicator of motion segmentation hypothetically. Nevertheless, optical flow on its own is insufficient because it cannot address occlusions and consecutive stopping. These techniques are also very sensitive to noise and light variations. Between image frames, it depicts coherent motion of points or

characteristics. Segmentation is carried out by classifying motion vectors into groups that move coherently. Without technology, it cannot be done in real-time. It has the following advantages[17]:

1. Its good in under light condition.

2. Can track moving object clearly.

2.6.5.1 Optical Flow in Motion Analysis

Optical flow gives a description of motion and can be a valuable contribution to image interpretation even if no quantitative parameters are obtained from motion analysis. Optical flow can be used to study a large variety of motions moving observer and static objects, static observer and moving objects, or both moving[17]. Optical flow analysis does not result in motion trajectories instead, more general motion properties are detected that can significantly increase the reliability of complex dynamic image analysis[18]. Motion, as it appears in dynamic images, is usually some combination of four basic elements:

I. Translation at constant distance from the observer.

II. Translation in depth relative to the observer.

III. Rotation at constant distance about the view axis.

IV. Rotation of a planar object perpendicular to the view axis.

Optical-flow based motion analysis can recognize these basic elements by applying a few relatively simple operators to the flow. Motion form recognition is based on the following facts:

a) Translation at constant distance is represented as a set of parallel motion vectors.

b) Translation in depth forms a set of vectors having a common focus of expansion.

c) Rotation at constant distance results in a set of concentric motion vectors.

d) Rotation perpendicular to the view axis forms one or more sets of vectors starting from straight line segments.

Exact determination of rotation axes and translation trajectories can be computed, but with a significant increase in difficulty of analysis[18]. Figure 2.4 depicts the motion form recognition and focus of expansion.

2.6.5.2 Optical Flow Computation

Optical flow computation is based on two assumptions:

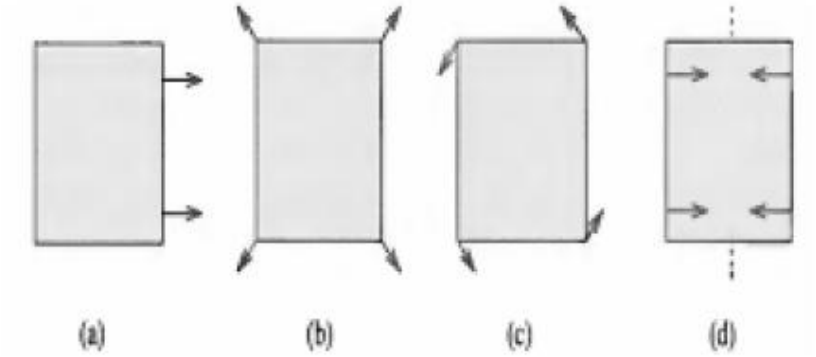
1. The observed brightness of any object point is constant over time.

2. Nearby points in the image plane move in a similar manner (the velocity smoothness constraint). Suppose we have a continuous image; $f(x, y, t)$ refers to the gray-level of (x, y) at time t; representing a dynamic image as a function of position and time permits it to be expressed. As refer to equation 2.2:

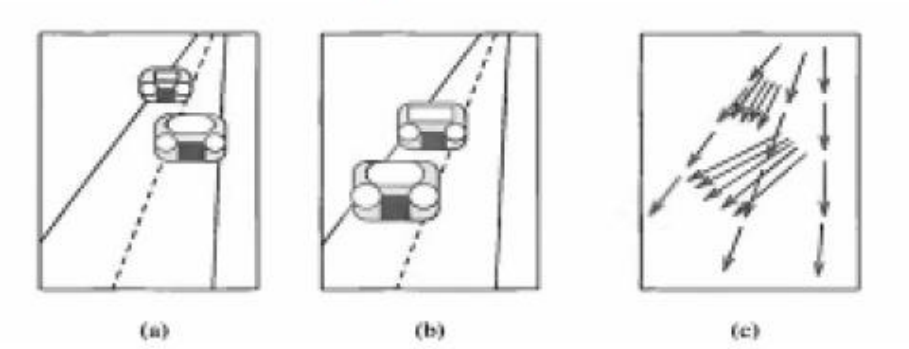
$$f(x + dx, y + dy, t + dt) = f(x, y, t) + fxdx + fydy + ftdt \quad (2.2)$$

Where fx , fy , ft denote the partial derivatives of f . We can assume that the immediate neighborhood of (x, y) is translated some small distance (dx, dy) during the interval dt that is, we can finddx, dy, dt as refer to equation 2.3 :

$$f(x + dx, y + dy, t + dt) = f(x, y, t) \quad (2.3)$$



Motion form recognition, (a) Translation at constant distance, (b) Translation in depth, (c) Rotation at constant distance, (d) Planar object rotation perpendicular to the view



Focus of expansion, (a) Time t1 (b) Time t2. (c) Optical flow

Figure 2:4: Motion form recognition and Focus of expansion[18]

The majority of optical flow approaches use gradient-based techniques, or block matching based methods[18]. Speed and efficiency factors have led to the preference of gradient-based methods. To identify the optical flow, gradient-based methods analyze the change in gradient and intensity using partial spatial and temporal derivatives. Block matching-based techniques depend on figuring out how closely the two images match. In order to calculate how much a region has moved, this usually entails matching "blocks" of one image to "blocks" of the other.

Both approaches rely on the assumptions of constant luminance and spatial continuity, and they work best when determining flow at or near clearly specified features.

As a consequence, errors in the optical flow output can happen when objects are not clearly defined (possibly because of clutter) or when the lighting circumstances change. When attempting to calculate the flow for uniform areas with little to no texture, performance is also affected.

A robust approach built on a robust estimation framework was proposed to attempt to get around the drawbacks of the current approaches[18]. By reducing the outliers brought on by motion discontinuities and violations of the constant luminance premise, the estimation framework. This strategy, however, is unsuitable for surveillance uses because it is too slow for a real-time system. Because they will catch every movement, optical flow images can be constrictive for surveillance purposes. Trees swaying in the wind can be eliminated using background modeling methods, but optical flow will always pick up on this motion. As a result, a lot of extra processing might be needed to filter the motion and determine what is brought on by the scene's target objects.

Another technique that is frequently used to track objects is optical flow. It is frequently used as an alternative to motion detection to find moving objects in a scene. Although optical flow offers extra information in the form of the direction of movement, it is more noise-prone than motion detection. This can be applied to segment occlusions between moving objects moving in various directions and help forecast future positions.

The algorithm for using particle filters to detect objects was employed. The particle tracker's probabilistic method works well for optical flow because it can easily handle the sparse or inaccurate data that optical flow estimates provide. In the condensation framework, an observation model was put forth to track using optical

flow contours. The accuracy of the match between the model and the input picture is assessed using the flow discontinuities along the contour between the inside and outside of the contour of the object being tracked. The optical flow inside should be similar to what the model predicts, whereas the optical flow outside should be very different. Optical flow algorithms work best in the vicinity of well-defined features, and they frequently manage sparsely detailed areas poorly. Within the bounding box of the object, uniformly bright areas are identified using optical flow to find general areas of motion.

2.7 Optimization Techniques

Swarm intelligence is a system of collective agent that interacts with the surrounding environment that perform global pattern. This intelligence compose base for the evaluation, comparing and imitation. Swarm intelligence system is act as in their coordinated without external disturbance. In years, the numbers of swarm base optimization have increased such as Particle Swarm Optimization (PSO), Artificial Bee Colony optimization (ABC) and Firefly Algorithm (FA) for image processing[19].

Swarm Intelligence principles have been successfully applied in a variety of problem domains including function optimization problems, finding optimal routes, scheduling, structural optimization, and image and data analysis. Computational modeling of swarms has been further applied to a wide-range of diverse domains, including machine learning, bioinformatics and medical informatics, dynamical systems and operations research; they have been even applied in finance and business[19].

2.7.1 Firefly Algorithm

Firefly algorithm (FA) is a biologically inspired meta-heuristic optimization algorithm. This algorithm is inspired by the flashing behaviour of tropical fireflies. In firefly algorithm, there are two important variables, which is the light intensity and attractiveness. Firefly is attracted toward the other firefly that has brighter flash than itself. The attractiveness is depended with the light intensity. The Firefly Algorithm (FA) is based on the communication behaviour of tropical fireflies and the idealized behaviour of the flashing patterns[2].

2.7.2 Particle Swarm Optimization

When compared with genetic search, Particle Swarm Optimization (PSO) is a relatively recent optimization technique of the swarm intelligence paradigm**.** PSO is a population-based optimization technique**,** inspired by the behaviour of schools of fish, herds of animals or flocks of birds. Particle Swarm Optimization is somewhat similar to genetic algorithms because both of them are population based. In PSO the system is initialized with a population of random solutions called particles[19]. These particles move through the problem space in search of the global minima or maxima. Each particle keeps track of its past best performance/fitness and its neighbours (Specified proximity radius) best performance to decide on its next move. Also, the swarm is aware of the global best achieved by all the particles[5].

The biological examples of the swarming, flocking, and herding phenomena in vertebrates served as the foundation for the development of the innovative distributed intelligent paradigm known as swarm intelligence (SI), which was created to solve optimization issues. Particle Swarm Optimization (PSO), from which the concept was derived, includes swarming behaviors seen in flocks of birds, schools of fish, or bee swarms, as well as human social behavior. PSO is a population-based optimization

tool that can be used to solve a variety of issues involving function optimization or problems that can be converted to function optimization problems. The primary advantage of PSO as an algorithm is its quick convergence, which compares favorably to other global optimization algorithms like Simulated Annealing (SA), Genetic Algorithms (GA), and others[5].

Because they rely directly on function values rather than derivative information, some population-based algorithms like the Genetic Algorithm (GA) and Simulated Annealing (SA) are expensive. They are, however, prone to premature convergence. An empirical approach to global optimization is particle swarm optimization. Before they discover the location where they can find food, the birds are either dispersed or group together during their search. While the birds are moving from one location to another in search of food, there is always one bird that can smell the food very well, which means the bird is aware of the location where the food can be found and has greater knowledge of the food resource[24].

The birds will eventually swarm to the location where food can be found because they are constantly transmitting information, particularly good information, while looking for food from one place to another. In terms of the particle swarm optimization algorithm, the solution swarm is like a flock of birds moving from one location to another; good information is like the most optimistic solution; and food resources are like the most optimistic solution over the course of the entire course. With the help of each person, the particle swarm optimization algorithm can determine the most optimistic answer. The simple behavioral pattern is controlled for each particle to demonstrate the complexity of the entire particle swarm. The particle without quality and volume, acts as each individual[5, 19].

Particles, which are possible solutions in PSO, follow the current optimum particles as they move through the problem space. Every particle maintains a record of the coordinates in the problem space that correspond to the best answer (fitness) found thus far. The name of this number is P_{best}. The best value so far attained by any particle in the particle's neighbors is another best value that the particle swarm optimizer keeps note of. The name of this number is lbest. The best value is a global best and is referred to as gbest when a particle uses the entire population as its topological neighbors. The idea behind particle swarm optimization is to change (accelerate) each particle's velocity toward its P_{best} and lbest at each time increment. (for lbest version). With distinct random numbers being produced for acceleration towards the P_{best} and lbest locations, acceleration is weighted by random term. The particle updates its velocity and positions using the following equations after determining the optimal values. Algorithm 2.1 expressed the standard particle swarm optimization[5, 17].

$$\vec{v}_i(k+1) = \omega\vec{v}_i(k) + c_1 r_1\big(P_b - \vec{x}_i(k)\big) + c_2 r_2\big(G_b - \vec{x}_i(k)\big) \quad (2.4)$$

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1) \quad (2.5)$$

Where,

$\vec{v}_i(k)$: is velocity of particle i at iteration k.

$\vec{x}_i(k)$: is the position of particle i at iteration k.

$\vec{v}_i(k+1)$: is velocity of particle i at iteration $k+1$.

$\vec{x}_i(k+1)$: is the position of particle i at iteration $k+1$.

r is an arbitrary integer between $(0,1)$

c_1 is the cognitive acceleration coefficient

c_2 is the social acceleration coefficient

The pseudocode of the initial version of PSO for real valued variables is given as follows[5]:

1. For each particle
2. initialize particle
3. End For
4. Do
5. For each particle
6. Calculate fitness value
7. if the fitness value is better than the best fitness value (P_{best}) in history set the current value as the new P_{best}
8. End
9. Choose the particle with the best fitness value of all the particles as the gBest
10. For each particle
11. calculate particle velocity according to equation (2.4)
12. update particle position according to equation (2.5)
13. End
14. While maximum iterations or minimum error criteria is not attained.

The principles of the PSO algorithm using the examples of synthetic inversion for velocity calibration in micro seismic monitoring were introduced[20]. The study explored the inevitable limitations tackling with complex models. Then, the work described the staged shrinkage strategy (SSS) for improving the PSO algorithm, which avoids premature and speeds up the convergence rate. Finally, the work

presented simulation results to demonstrate the superiority of our proposed SSS-PSO algorithm. The work also investigated the influence of the built-in velocity clamping factor on the reliability and efficiency for the PSO and SSS-PSO algorithms. Algorithm 2.2 described the Modified Particle Swarm Optimization used in the work.

Algorithm 2.1: Standard Particle Swarm Optimization[20]

Step 1:Generate random population of N, Set parameter ω_{min}, ω_{max}, c_1 and c_2 of PSO

Step 2: Initialize population of particles having positions x_j and velocities v_j

Step 3: Set iteration $k = 1$

Step 4: Calculate fitness of particles $F_{ij}(t) = f(\vec{x}_{ij}(t))$ and find the index of the best particle b

$$\vec{x}_{ij}(t) = Minimize \sum_{i}^{n}\sum_{j}^{m} e^{C_{ij}X_{ij}}$$

Step 5: Select $Pbest_{ij}(t) = \vec{x}_{ij}(t)$ and $Gbest_{ij} = x_{bj}(t)$

Step 6:$\omega = \omega_{min} + (\omega_{max} - k + 1) \times \frac{\omega_{max} - \omega_{min}}{Max_{no}}$ where, k is the current iteration, ω_{max} is the final weight, ω_{min} is the initial weight ω is the inertia weight employed to overcome the problem of premature convergence, Max_{no} is the maximum number of iterations.

Step 7: Update velocity and position of particles

$$\vec{v}_{ij}(t+1) = X\omega\vec{v}_{ij}(t) + c_1 r_1\left(P_{best} - \vec{x}_{ij}(t)\right) + c_2 r_2\left(P_{best} - \vec{x}_{ij}(t)\right) + c_3 r_3\left(G_{best} - \vec{x}_{ij}(t)\right)$$

$$\vec{x}_{ij}(t+1) = \vec{x}_{ij}(t) + \vec{v}_{ij}(t+1)$$

Step 8: Evaluate fitness $F_{ij}(t) = f(\vec{x}_{ij}(t+1))$ and find the index of the best particle b_1

Step 9: Update $Pbest$ of population

If $F_{ij}(t+1) < F_{ij}(t)$ then $Pbest_{ij}(t+1) = \vec{x}_{ij}(t+1)$ else

$$Pbest_{ij}(t+1) = Pbest_{ij}(t)$$

Step 10: Update $Gbest$ of population

If $F_{bj}(t+1) < F_{bj}(t)$ then $Gbest_j(t+1) = Pbest_{bj}(t+1)$ and set $b = b_1$ else

$$Gbest_{bj}(t+1) = Gbest_j(t)$$

Step 11: If $k < Max\_no$ then $k = k + 1$ and goto step 2 else goto step 11

Step 12: Output optimum solution as $Gbest_{bj}$.

$$Gbest_{bj} = \vec{x}_{bj}(t)$$

Algorithm 2.2: Modified Particle Swarm Optimization[20]

Step 1: Generate random population of N, Set parameter ω_{min}, ω_{max}, c_1 and c_2 of PSO

Step 2: Initialize population of particles having positions x_j and velocities v_j

Step 3: Set iteration $t = 1$

Step 4: Evaluate the objective function values of particles as $f(x) = f(x_i^t)$

$$f(x) = \sum_{i=1}^{m}\sum_{j=1}^{n} \Delta\left(W_{i,j}^{m,n}\right)\left((x_i) - (x_j)\right)$$

Where, x_i^t represent the s at $i = 1,2, \ldots, n$ and $k = 2, 3, \ldots, m$.

Where, $\Delta\left(W_{i,j}^{m,n}\right)((x_i) - \left(x_j\right)$ is the change in weight of input pixel x along the row and column.

Step 5: Find the cognitive best for each particle as $P_{best,iD}^t = x_i^t$ and global best as $G_{best,\,iD} = min\{P_{best,iD}^t\}$

Step 7: Find the new values by updating the velocity and position of the i^{th} in the D^{th} Dim

$V_{max,\,D} = \delta(x_{max,\,D} - x_{min,\,D})$

$V_{min,\,D} = \delta(x_{min,D} - x_{max,\,D})$

where $V_{max,\,D}$ is the allowed maximum velocity of particles, D is the velocity clamping factor, and $x_{max,\,D}$ and $x_{min,\,D}$ are the maximum and minimum location values of particles at D^{th} −dimension.

$V_{max,\,D} = \delta(x_{max,\,D} - x_{min,\,D})$

where, $V_{max,\,D}$ is the allowed maximum velocity of particles, D is the velocity clamping factor, and $x_{max,\,D}$ is the maximum location values of particles at D^{th} dimension.

$$V_{i,D}(t+1) = \begin{cases} V_{max,D}, & \text{if, } V_{i,D}(t) > V_{max,\,D} \\ V_{i,D}(t+1), & \text{if, } V_{i,D}(t) < V_{max,\,D} \end{cases}$$

$$V_{i,D}{}^{t+1} = \gamma . V_{i,D}{}^{t} + c_1^t . r_1 \left[P_{best,iD}^t - x_i{}^t \right] + c_2^t . r_2 \left[G_{best,iD}^t - x_i{}^t \right]$$

$$x_i{}^{t+1} = x_i{}^t + V_{i,D}{}^{t+1}$$

where $r_1$and $r_2$are random numbers from uniform distribution $U(0,1)$

Step 8: Find the objective function values of x_b^{t+1} as $f(x) = f(x_b^{t+1})$ and find the index of the best particle b

Step 9: Update P_{best} of population

$$P_{best,bD}^{t+1} = \begin{cases} P_{best,iD}^t & \text{if,} \quad f(x_b^{t+1}) > P_{best,iD}^t \\ x_b^{t+1} & \text{if,} \quad f(x_b^{t+1}) \leq P_{best,iD}^t \end{cases}$$

Step 10: Update $G_{best}$of population

$$G_{best,\ bD} = \begin{cases} \min\ (P_{best,iD}^t) & \text{if,} \quad f(x_b^{t+1}) > P_{best,iD}^t \\ \min\ (P_{best,bD}^{t+1}) & \text{if,} \quad f(x_b^{t+1}) \leq P_{best,iD}^t \end{cases}$$

Step 11: If t< Max_{iter} then t= $t + 1$ and GOTO step 1 else GOTO step l2

Step 12: Output optimum weight selected solution as $Gbest_{bD}$.

$$Gbest_{bD} = x_b$$

2.8 Convolution Neural Networks (CNN)

CNNs are a variant of feed-forward neural networks with a special architecture. The architecture of CNNs usually contains a convolution followed by a pooling operation. Every neuron in a convolution layer is connected to some region in the input, which is called a local receptive field. Unlike other types of feed-forward neural networks, all weights are shared based on the position within a receptive field. The shared weights are also called filters[1]. The convolutions operation can be formalized as follows:

$$(f * g)(z) = \sum_{x}\sum_{y} f(x,y) \cdot g(z-x, z-y) \tag{2.6}$$

where, $f(x,y)$ is the input image at position (x,y) and $g(z-x, z-y)$ is a trainable filter. Further, a pooling layer is used to generate translation invariant features by computing statistics of the convolution activations from different positions along specific windows. One pooling layer that is commonly used in the CNN implementation is the max pooling layer, which takes the maximum value over a processed region. For the two-dimensional case, the max pooling operation with a pooling size of $j \times k$ becomes:

$$MP\,(f(x,y)_{jk} = \max_{\substack{m=j-x,\ \ldots,j+x \\ n=k-y,\ \ldots k+y}} \{f(m,n)\} \tag{2.7}$$

Convolution Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimize through learning[1]. Each neuron will still receive an input and perform an operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single perceptive score function (the weight). The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply. The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in the field of pattern recognition within images. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks - whilst further reducing the parameters required to set up the model. One of the largest limitations of traditional

forms of ANN is that they tend to struggle with the computational complexity required to compute image data[1, 17].

2.8.1 CNN Architecture

CNNs primarily focus on the basis that the input was comprised of images. This focuses the architecture to be set up in a way to best suit the need for dealing with the specific type of data. One of the key differences is that the neurons that the layers within the CNN are comprised of neurons organized into three dimensions, the spatial dimensionality of the input (height and the width) and the depth[1, 21]. The depth does not refer to the total number of layers within the ANN, but the third dimension of an activation volume. Unlike standard ANNS, the neurons within any given layer will only connect to a small region of the layer preceding it.

CNNs are comprised of three types of layers. These are convolutional layers, pooling layers and fully-connected layers. When these layers are stacked, a CNN architecture has been formed. A simplified CNN architecture for MNIST classification is illustrated in Figure 2.2:

The basic functionality of the example CNN above can be broken down into four key areas.

1. As found in other forms of ANN, the input layer will hold the pixel the image.

2. The convolutional layer will determine the output of neurons of which are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume.

The rectified linear unit (commonly shortened to ReLu) aims to apply an 'elementwise' activation function such as sigmoid to the output of the activation produced by the previous layer.

3. The pooling layer will then simply perform down sampling along the spatial dimensionality of the given input, further reducing the number of parameters within that activation.

4. The fully-connected layers will then perform the same duties found in standard ANNs and attempt to produce class scores from the activations to be used for classification. It is also suggested that ReLu may be used between these layers, as to improve performance.

Through this simple method of transformation, CNNs are able to transform the original input layer by layer using convolutional and down-sampling techniques to produce class scores for classification and regression purposes.

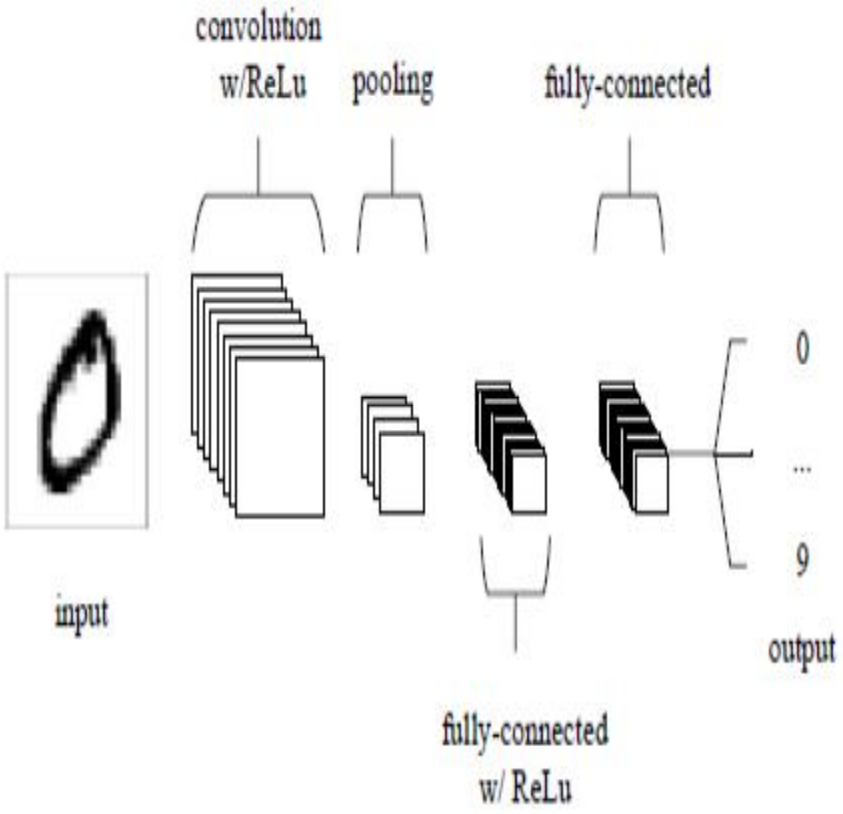


Figure 2.5: A simple CNN architecture, comprising five layers[21]

2.8.2 Performance Metrics

Performance metrics are accurac 57 sitivity, and specificity. A confusion matrix is used to calculate these three measures. The confusion matrix is a matrix representation of the classification results, the upper-left cell denotes the number of samples classifies as true while they were true (i.e., true positives), and lower right cell denotes the number of samples classified as false while they were actually false (i.e., true-false). The other two cells (lower left cell and upper right cell) signifies the number of samples misclassified. Specifically, the lower-left cell denoting the number of samples classified as false while they actually were true (i.e., false negatives), and the upper right cell denoting the number of samples classified as true while they actually were false (i.e., false positives)[21].

A confusion matrix for classification of a k class is k x k contingency table whose cells [i,j] (i=1,...,k, j=1,...,k) present frequencies of observations with real class Ci and inferred class Cj. A binary confusion matrix is a special case when there are only two classes: C (positive class) and not-C (negative class). A k x k confusion matrix can be represented as a set of k binary confusion matrices, one for each class Ci. In a binary confusion matrix, observations classified correctly into the positive class are called true positives and observations classified correctly into the negative class are called true negatives. Occurrences of the positive class classified falsely as negative are called false negatives and examples of negative class classified falsely as positive are called false positives. Numbers of true positive, false positive, true

negative and false negative observations are identified with TP, FP, TN, and FN. From these frequencies, one can calculate classification performance indicators that reflect how the classifier performs in detecting the given class. The most common of such indicators are discussed[22].

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \; x\,100\% \tag{2.7}$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP+FN}} \; x\,100\% \tag{2.8}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}} \; x\,100\% \tag{2.9}$$

$$\text{False Positive Rate} = \frac{\text{FP}}{\text{TN+FP}} = 1 - Specificity \tag{2.10}$$

$$\text{Overall Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \; x\,100\% \tag{2.11}$$

$$\text{Average recognition time} = \frac{\text{Total Recognition Time}}{\text{Number of recognized foods}} \tag{2.12}$$

2.9 Related Works

The purpose of this study is to examine the scalability of the incremental deep learning approach for visual identification, particularly for applications involving quick object detection. Comparing training-at-once versus gradual learning with information transfer and distillation, the experimental investigation shows that training-at-once has a higher computing cost and lower knowledge retention. The experiment used a cutting-edge object detector as its foundation and extended it to include knowledge transfer and distillation to compare three training methods: training-at-once, transfer learning without distillation, and transfer learning with distillation. The experimental results and analysis compared the accuracy of new classes, knowledge retention of old classes, data storage, calculation time, and

memory use while modifying several important parameters, examining the pros and cons of each training technique. While using the most storage and memory, training-at-once (the baseline) produced the highest accuracy of both new and old classes. Both transfer learning methods reduced the amount of storage needed by 73% when compared to the baseline, but at the cost of a 53% increase in computing time. Distillation's ability to handle long-term incremental learning is shown by its ability to retain 96% accuracy with previous classes, demonstrating the need of transfer learning for information retention. Transfer learning without distillation was able to produce slightly higher accuracy with new classes (53% vs. 60%), less memory utilization (65% vs. +26%), but at the risk of forgetting existing classes (100%). This study proved that distillation loss could assist retain the advantages of incremental learning while balancing the accuracy of the old and new object classes. Training batch size and the number of assigned classes are crucial for preserving the accuracy of new classes, maintaining the knowledge of existing classes, and lowering computing costs, according to research utilizing different critical parameters across all training methodologies[23].

It is difficult to implement high-performance one-stage object detectors on applications with limited resources. This study examines the variables influencing the computing complexity of one-stage detectors and suggests LEYOLO, a compact and effective detection framework. This framework was used to create a lightweight backbone network for the detection task that consisted of a number of effective feature extraction modules and a new channel attention module. A lightweight multiscale feature fusion structure with a weighted fusion algorithm was suggested to effectively integrate the features retrieved from the backbone without incurring the

expense of dimensionality reduction and downsampling. Additionally, based on the framework, two detectors (LEYOLOs and LEYOLOm) were created. Given only modest computing budgets, experimental results demonstrate that LEYOLO accomplished state-of-the-art trade-offs between performance and complexity[24].

Deep learning is essential for the real-world task of object detection. The You Only Look Once (YOLO) object identification model accurately and quickly identifies interesting areas in photos. In this essay, object detection is used to enhance security and combat terrorism. By identifying and detecting the guns on closed-circuit television (CCTV), people are shielded from harm. This study provides a YOLO v4-based real-time detection method for CCTV autonomous weaponry. The authors suggested using the YOLO v4 backbone with Spatial Cross Stage Partial-ResNet (SCSP-ResNet) to address the features of CCTV scenarios. The receptive field improvement module, meanwhile, was demonstrated to be able to capture subtle semantic details of high-dimensional tiny objects. The experimental results show that the proposed detection model reduces inference time and mAP (mean Accuracy Precision) by 7.37% and 4.2%, respectively[25].

The goal of this study was to develop a new Multi-Object Tracking-by-Detection (MOT-bD) framework using a Deep Convolutional Neural Network (DCNN) detector and a spatiotemporal interlaced video model. "Interlaced images" are images that have had the spatiotemporal variation of objects between photos stored. Since interlaced objects were constructed to increase overlap during the association step, which improved the MOT performance over the same detector/association algorithm applied on non-interlaced images, a specialized

"interlaced object" convolutional deep detector was trained to detect objects in interlaced images and a classical association algorithm to perform the association between detected objects. The results of the experiments show that interlacing video has many benefits for tracking performances, including precision and accuracy, and they show that the "power of video-interlacing" outperforms several cutting-edge tracking frameworks when tracking multiple objects[26].

Different feature integration in data association is still a challenge. For instance, focusing solely on the appearance feature may result in inaccurate association results when intra-frame objects have similar looks, while relying too much on the motion feature may result in failure to perform the essential data association when object movements are complex. The writers redesigned the integration of motion and appearance in order to achieve a better trade-off between those two features. The online technique casts each object's position and motion into adaptive search windows, and within those search windows, matching is limited to the similarity of visual attributes. The motion feature is formed as spatiotemporal constraints and used to refine tracklets produced by the online approach in the offline method. Numerous MOT datasets were subjected to experiments from a variety of angles, including different motion speeds, different lighting conditions, different item classifications, etc., to confirm that the approach performs reliably. The technique also shows its efficacy by outperforming the prior first-place finishes in two CVPR 2020 MOT tasks[27].

This study introduces a novel network architecture that can increase object detection accuracy by utilizing the spatiotemporal data seen in videos. First,

connection suggestions that originate from the same anchor box in surrounding frames are used to associate and aggregate box properties. In order to utilize long-term spatio-temporal information, the authors next created a new attention module that collects short-term enhanced box properties. For the first time in the field of video object detection, this module makes use of geometrical features over a lengthy period of time. Finally, aggregated data that takes into consideration both the short- and long-term temporal context is supplied into a spatio-temporal double head together with spatial information from the reference frame. In order to demonstrate the proposal's durability over a wide range of situations, it was evaluated on five video object detection datasets with wildly varied features. The method performs better than the state-of-the-art, according to non-parametric statistical testing[28].

The authors of this work suggested a brand-new deep learning-based traffic sign identification system. The "you look only once" (YOLO) v5 structure for feature selection was used in conjunction with the method, known as the feature-selection-based attentional deconvolution detector (FSADD). The network separates the acquired feature maps from the convolution layer into similar and non-similar feature maps when a detection technique uses feature selection. The outputs of filters with random weights are typically used to create the feature maps that are acquired after the convolution layers. The network receives numerous types of feature maps with extraneous components as a result of the filter's randomization, which reduces the effectiveness of detection. The sizes of the receptive fields have also been altered in the FSADD model for better traffic sign detection performance. Due to the tiny sizes of these signs in the photos, many of the generic detection techniques are not suited for the German traffic sign detection benchmark (GTSDB). Experimental

comparisons with the GTSDB were made to demonstrate that the FSADD is on par with the state-of-the-art for detecting 29 different types of traffic signs with classification performances that are 73.9% accurate[29].

Due to their limited visual qualities, it is theoretically challenging to detect small moving targets made up of one or a few pixels. Some bio-inspired models have been created as a result of natural inspiration. The present models' heavy reliance on the input videos' sampling rates, however, creates a bottleneck that severely impedes their use in real-time in the physical world. This is due to the fact that high-sampling-frequency videos demand a significant amount of computer resources to collect and process. While there are considerable spatial inaccuracies and poor responses in low-sampling-frequency films, model detection performance. The authors suggested an STMD-based visual neural model with a fractional-order difference operator for small target motion detection to overcome these problems. The rising and falling brightness components are further separated by the STMD network before being time-aligned, multiplied, and used to forecast the positions of moving tiny targets. The created model locates the small moving targets accurately and robustly in low sampling frequencies because of the instantaneous information's quick response and the addition of memory information. The created model greatly increases the detection performance for low-sampling-frequency videos, according to numerical testing[30].

Due to their great effectiveness, low cost, and superior mobility, unmanned aerial vehicles (UAVs) have been used to check in a variety of situations. But since they are so much smaller and denser than regular objects, objects in aerial photos make it challenging for current object detection techniques to get the desired

outcomes. A previous improved Transformer network (PETNet) based on YOLO was suggested in this research as a solution to this problem. A one-to-many feature fusion (OMFF) method and a unique previous improved Transformer (PET) module are specifically suggested for integration into the network. The shallow feature maps receive two more detecting heads. In order to increase the network's capacity for expressiveness, improved global information was captured using PET in this study. In order to reduce the information loss of small objects, the OMFF seeks to fuse multiple types of characteristics. Additionally, the additional detection heads increase the likelihood of finding smaller-scale items, and the expanded multi-head parallel detection is better suited to detecting things in aerial photos at several scales. The constructed PETNet delivers cutting-edge results on the VisDrone-2021 and UAVDT databases with average precision (AP) of 35.3 and 21.5, respectively, demonstrating that the network is better suited for aerial image detection and is of great reference value[31].

Unmanned aerial vehicle (UAV) computer vision analysis is a significant future development. One of the key technologies to assess scene information in UAV photos is object detection. The small and dense targets in the UAV image, however, make missed detection errors quite easy to make. In order to achieve high quality detection of small targets, the authors of this research use a dual neural network review technique that swiftly sorts through the missed targets in the one-stage detection by identifying the secondary properties of the suspected target regions. First, the one-stage detector identifies UAV images, and the target is detected when the result is detected with confidence more than or equal to the threshold. Results below the threshold are regarded as potential regions with missed aims. Second, the VGG

backbone extracts the feature map from the UAV image. The location data of the suspicious regions and the feature map were combined for secondary identification. The dual network's features are then fused late, and the re-identified outcomes serve as the first confidence addition's compass. Regions with confidence above the threshold are regarded as targets after the addition. In order to acquire the final detection findings, the targets of initial detection and secondary identification were finally synthesized. The technique offers ground-breaking performance on the VisDrone, UAVDT, and MS COCO datasets, according to experimental results[32].

The main difficulty for harvesting robots is reliably detecting crops, and deep learning techniques are frequently utilized for this. These techniques, however, need a lot of training data, which can be a problem. In order to increase the volume of training data and improve the effectiveness of the detection models, data augmentation is commonly advised. In this study, the authors suggested using geometric, photometric, and partial occlusion changes to RandAugment (RA) to enhance crop identification performance. On Tomato datasets, the authors tested with various transformation lists utilizing well-known detection networks. The maximum accuracy was attained by YOLOv3 with geometric modifications and partial occlusion at 76.42, an improvement of 3.47 over the baseline model without augmentation. Additionally, RA increased the detection accuracy in other open datasets like Apple, Kiwi, and Mango. For each network, a different set of alterations was the ideal combination to get the best performance. Additionally, the authors used YOLOv3 and the ideal transformation list to show the robotic tomato harvesting system, and they were able to reduce the processing time from getting the detection data to less than 60 ms[33].

This work proposed a general viewpoint that unifies existing region feature extraction methods and a novel method that is end-to-end learnable. The method removes most heuristic choices and outperforms its Region of Interest (ROI) pooling counter-parts. It moves further towards fully learnable object detection[34].

The authors of this research work formulated object detection as a problem of graph structure inference, where given an image the objects are treated as nodes in a graph and relationships between the objects are modeled as edges in such graph.It presented a so-called Structure Inference Network (SIN), a detector that incorporates into a typical detection framework (e.g. Faster R-CNN) with a graphical model which aims to infer object state. Comprehensive experiments on PASCAL VOC and MS COCO datasets indicated that scene context and object relationships truly improve the performance of object detection with more desirable and reasonable outputs[35].

Another approach on how to modify Faster R-CNN for the task of small object detection in optical remote sensing images was investigated by the researchers. First, the study not only modify the RPN stage of Faster R-CNN by setting appropriate anchors but also leverage a single high-level feature map of a fine resolution by designing a similar architecture adopting top-down and skip connections. In addition, the study incorporated context information to further boost small remote sensing object detection performance while the study applied a simple sampling strategy to solve the issue about the imbalanced numbers of images between different classes. At last, the study introduced a simple, yet effective data augmentation method named 'random rotation' during training. Experimental results show that our modified Faster R-CNN algorithm improves the mean average precision by a large margin on detecting small remote sensing objects[36].

A deep object recognition framework using a synthetic depth image dataset was developed and validated by the authors. The study synthetically generated a depth image dataset of 22 objects randomly placed in a 0.5 m × 0.5 m × 0.1 m box, and automatically labeled all objects with an occlusion rate below 70%. Faster Region Convolutional Neural Network (R-CNN) architecture was adopted for training using a dataset of 800,000 synthetic depth images, and its performance was tested on a real-world depth image dataset consisting of 2000 samples. Deep object recognizer has 40.96% detection accuracy on the real depth images and 93.5% on the synthetic depth images. Training the deep learning model with noise-added synthetic images improves the recognition accuracy for real images to 46.3%. The object detection framework can be trained on synthetically generated depth data, and then employed for object recognition on the real depth data in a cluttered environment. Synthetic depth data-based deep object detection has the potential to substantially decrease the time and human effort required for the extensive data collection and labeling[37].

A review on deep learning-based object detection frameworks was addressed provided and provided for in this research work. The review begins with a brief introduction on the history of deep learning and its representative tool, namely Convolutional Neural Network (CNN). Then the study focused on typical generic object detection architectures along with some modifications and useful tricks to improve detection performance further. As distinct specific detection tasks exhibit different characteristics, the study also briefly survey several specific tasks, including salient object detection, face detection and pedestrian detection. Experimental analyses were also provided to compare various methods and draw some meaningful conclusions. Finally, several promising directions and tasks are provided to serve as

guidelines for future work in both object detection and relevant neural network-based learning systems[38].

This study proposed a comprehensive survey of recent advances in visual object detection with deep learning. By reviewing a large body of recent related work in literature, the study systematically analyzed the existing object detection frameworks and organize the survey into three major parts: firstly, detection components, secondly, learning strategies, and thirdly, applications and benchmarks. In the survey, it covered a variety of factors affecting the detection performance in detail, such as detector architectures, feature learning, proposal generation, sampling strategies, and so on. Finally, it discussed several future directions to facilitate and spur future research for visual object detection with deep learning[39].

A comprehensive review of the recent deep learning-based object detection progress in both the computer vision and earth observation communities was provided for in this study. The authors proposed a large scale, publicly available benchmark for object Detection in Optical Remote sensing images, which the study name as DIOR. The dataset contains 23,463 images and 192,472 instances, covering 20 object classes. The proposed DIOR dataset which has large-scale on the object categories, on the object instance number, and on the total image number; secondly, it has a large range of object size variations, not only in terms of spatial resolutions, but also in the aspect of inter- and intra-class size variability across objects; thirdly, it holds big variations as the images are obtained with different imaging conditions, weathers, seasons, and image quality; and fourthly, it has high inter-class similarity and intra-class diversity. The proposed benchmark can help the researchers to develop and validate their data driven methods. Finally, the study evaluated several state-of-the-art approaches on our

DIOR dataset to establish a baseline for future research[40].

This work on the mainstream object detection algorithms was compared and analyzed and proposed a multi-scaled deformable convolutional object detection network. The study used deep convolutional networks to obtain multi-scaled features and add deformable convolutional structures to overcome geo-metric transformations. It then fused the multi-scaled features by up sampling, to implement the final object recognition and region regress. The experiments improved the accuracy of detecting small target objects with geometric deformation, showing significant improvements in the trade-off between accuracy and speed[41].

The overhead view of person tracking object using faster region convolutional neural network (Faster-RCNN) in combination withGeneric Object Tracking Using Regression Networks (GOTURN) architecture was developed by the researchers. Individual person was first identified in overhead view video sequences and then trackedusing a GOTURN tracking algorithm. Faster-RCNN detection model achieved the true detection rate ranging from 90%to 93% with a minimum false detection rate up to 0.5%. The GOTURN tracking algorithm achieved similar results withthe success rate ranging from 90% to 94%[42].

The authors of this work implemented object tracking in a video using Convolutional Neural Network (CNN) architecture. The study used ALOV 300+ datasetwhich included frames of videos of different classes (Light, Surface Cover, Specularity, Transparency, Shape, Motion Smoothness, Motion Coherence and Clutter) with their corresponding target object bounding boxes. It used two Inception V3 architectures that contain the CNN framework. Most of the generic object trackers were trained from scratch online and do not benefit from many videos that were

available for offline training. The method used extensive offline training of neural networks for tracking different objects at high speed in real time. The algorithm learned a general relationship between an object's appearance and its motion in an offline manner, allowing the network to track new objects. The model showed 78% accuracy on the train set of 10,000 images and an accuracy of 76% on the test set. It also increasing the dropout factor from 0.5 to 0.7 variance and was reduced by 2% whereas regularization helped reduce the variance by 0.1%[43].

The tracker learned to track objects by reusing the features from the image object detector, which is a light-weighted increment to the detector, with only a slight speed drop for the video object detection task to augment a well-trained image object detector with an efficient and effective class-agnostic convolutional regression tracker for the video object detection task. The performance of the model as proposed by the authors was evaluated on the large-scale ImageNet VID dataset. It improved the mean average precision (mAP) score for the image object detector by around 5% and around 3% for the image object detector plus Seq-NMS post-processing[44].

A novel and accurate object detection method for garlic using a convolutional neural network was developed by the researchers of this work. The you-only-look-once (YOLO) algorithm, which is based on lightweight and transfer learning, is the most advanced computer vision method for single large object detection. To detect the bulb, the YOLOv2 model was modified using an inverted residual module and residual structure[45]. The modified model was trained based on images of bulbs with varied brightness, surface attachment, and shape, which enabled sufficient learning of the detector. The optimum minibatches and epochs were obtained by comparing the test results of different training parameters. Research shows that IRM-YOLOv2 is

superior to the SqueezeNet, ShuffleNet, and YOLOv2 models of classical neural networks, as well as the YOLOv3 and YOLOv4 algorithm models. The confidence score, average accuracy, deviation, standard deviation, detection time, and storage space of IRM-YOLOv2 were 0.98228, 99.2%, 2.819 pixels, 4.153, 0.0356 s, and 24.2 MB, respectively. In addition, this study provides an important reference for the application of the YOLO algorithm in food research. Computational time were not taken into consideration[46, 47].

The authors proposed a robust new algorithm to detect and track objects from natural scenes captured with real-time cameras. This work aims to create a detection and tracking algorithm that is responsive to actual and fundamental changes. This algorithm is characterized by the detection of multiple moving creatures, limited resources, and different challenges. This algorithm combines principal component analysis and deep learning networks to make the most of these two approaches' advantages to achieve an intelligent detection and tracking system that works in real-time. It is done adaptively between the two approaches to enhance performance compared to the existing detection and tracking algorithms. The experimental results showed the new algorithm's effectiveness and efficiency by comparing it with other detection and tracking systems and obtaining good detection and classification accuracy[48, 49].

A quality-aware object association (QOA) strategy to leverage the quality score as an important reference factor for achieving robust association was developed in this study. Despite its simplicity, extensive experiments indicate that the strategy significantly boosts tracking performance by 2.2% Average Multi-object Tracking Accuracy(AMOTA) and the authors' method outperforms all existing state-of-the-art

works on nuScenes by a large margin. Moreover, QTrack achieves 48.0% and 51.1% AMOTA tracking performance on the nuScenes validation and test sets, which significantly reduces the performance gap between pure camera and LiDAR based tracker[50, 51].

A new solution to moving object detection and tracking using an event frame from bio-inspired event cameras was developed by the researchers. First, an object detection method was designed using a combined event frame and a standard frame in which the detection is performed according to probability and color, respectively[52]. Then, a detection-based object tracking method is proposed using an event frame and an improved kernel correlation filter to reduce missed detection. Further, a distance measurement method is developed using event frame-based tracking and similar triangle theory to enhance the estimation of distance between the object and camera. Experiment results demonstrate the effectiveness of the proposed methods for moving object detection and tracking[53, 54].

3D detection and 3D tracking from only monocular videos in an end-to-end manner was jointly proposed and trained by the authors of this work. The key component is a novel spatial-temporal information flow module that aggregates geometric and appearance features to predict robust similarity scores across all objects in current and past frames. Specifically, we leverage the attention mechanism of the transformer, in which self-attention aggregates the spatial information in a specific frame, and cross-attention exploits relation and affinities of all objects in the temporal domain of sequence frames. The affinities are then supervised to estimate the trajectory and guide the flow of information between corresponding 3D objects[55]. In addition, the authors proposed a temporal -consistency loss that explicitly involves 3D

target motion modeling into the learning, making the 3D trajectory smooth in the world coordinate system. Time3D achieves 21.4% AMOTA, 13.6% AMOTP on the nuScenes 3D tracking benchmark, surpassing all published competitors, and running at 38 FPS, while Time3D achieves 31.2% mAP, 39.4% NDS on the nuScenes 3D detection benchmark[56, 57].

A hybrid, high-temporal-resolution object detection and tracking approach that combines learned and classical methods using synchronized images and event data was presented in this study. Off-the-shelf frame-based object detectors are used for initial object detection and classification. Then, event masks, generated per detection, are used to enable inter-frame tracking at varying temporal resolutions using the event data. Detections are associated across time using a simple, low-cost association metric[58]. Moreover, we collect and label a traffic dataset using the hybrid sensor DAVIS 240c. This dataset is utilized for quantitative evaluation using state-of-the-art detection and tracking metrics. We provide ground truth bounding boxes and object IDs for each vehicle annotation. Further, the authors generated high-temporal-resolution ground truth data to analyze tracking performance at different temporal rates. Our approach shows promising results, with minimal performance deterioration at higher temporal resolutions (48–384 Hz) when compared with the baseline frame-based performance at 24 Hz[59, 60].

In order to reduce the number of traffic accidents caused by jaywalking each year, a video anomaly detection algorithm for the activity was suggested in this paper. The suggested model has been testing with one of the newest street scene datasets and its various versions. It is a novel variation of the InceptionV3 deep CNN model. Two distinct subsystems built on the pre-trained InceptionV3 architecture make up our

model. The first subsystem, Anomaly Detector, analyzes a video frame and determines whether a jaywalking event is there. When a jaywalking (anomalous) video frame is supplied, the second subsystem, Anomaly Localizer, predicts the bounding box labels to identify the jaywalking item in the frame. We use the Street Scene Dataset, which was released in February 2020 and provides a variety of jaywalking instances, to analyze[61]. In this study, we test various pre-trained CNN models, including VGG16, ResNet50, and InceptionV3, to assess how well they perform anomaly detection. We also evaluate the model's performance for various dataset sizes with both an even and an uneven distribution of anomalous and non-anomalous frames. Finally, we evaluate how well the model predicts various kinds of jaywalking incidents. The study demonstrates that the detection accuracy of our suggested model is remarkably high[62].

The inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids with marginal extra cost was exploited by the researchers. A top-down architecture with lateral connections was developed for building high-level semantic feature maps at all scales. This architecture, called a Feature Pyramid Network (FPN), showed significant improvement as a generic feature extractor in several applications. Using FPN in a basic Faster R-CNN system, the method achieves state-of-the-art single-model results on the COCO detection benchmark without bells and whistles, surpassing all existing single-model entries including those from the COCO 2016 challenge winners. In addition, the method adopted by the study can run at 5 FPS on a GPU and thus is a practical and accurate solution to multi-scale object detection[63, 64].

The task of unsupervised learning to detect and segment foreground objects in

single images was addressed by the authors. The study achieved researchers goal by training a student pathway, consisting of a deep neural network that learns to predict, from a single input image, the output of a teacher pathway that performs unsupervised object discovery in video. The approach used is different from the published methods that perform unsupervised discovery in videos or in collections of images at test-time. The study moved the unsupervised discovery phase during the training stage, while at test time the study apply the standard feed-forward processing along the student pathway. This has a dual benefit: firstly, it allows, in principle, unlimited generalization possibilities during training, while remaining fast at testing. Secondly, the student not only becomes able to detect in single images significantly better than its unsupervised video discovery teacher, but it also achieves state of the art results on two current benchmarks, YouTube Objects and Object Discovery datasets. At test time, the system was two orders of magnitude faster than other previous methods[65, 66].

A unified implementation of the Faster R-CNN, R-FCN and SSD systems, which the study viewed as "meta-architectures" and trace out the speed/accuracy trade-off curve created by using alternative feature extractors and varying other critical parameters such as image size within each of these meta-architectures was presented in this study[67]. On one extreme end of this spectrum where speed and memory are critical, the study present a detector that achieves real time speeds and can be deployed on a mobile device. On the opposite end in which accuracy is critical, the study presented a detector that achieves state-of-the-art performance measured on the COCO detection task[68].

A brand-new automatic method for spotting small objects in traffic sequences was presented in this paper. In the first stage, super-resolution algorithms and pre-

trained object identification networks were used to build automatically, offline, the vehicle patterns identified from a sequence of frames. The object detection model was then retrained using the previously acquired data, customizing it to the scene under analysis. The retrained model was then used in the remaining traffic sequence or the camera-generated video stream since it is already live and in real-time. The GRAM and NGSIM datasets have both been used to test this architecture[69,70].

This work focused on a multi-convolutional neural network (CNN)-based system for the detection, monitoring, and recognition of dog emotions in surveillance films. This system tracks the dogs in the video, finds them in each frame, and detects their emotions. The system detects dogs using a YOLOv3 model. A deep association metric model (DeepDogTrack) that uses a Kalman filter in conjunction with a CNN for processing tracks the dogs in real time. Then, based on manual evaluations by veterinary professionals and custom dog breeders, the dogs' emotional behaviors are divided into three categories: angry (or aggressive), happy (or excited), and neutral (or general). The long short-term deep features of dog memory networks model (LDFDMN) is used to detect the emotions of the dogs after the system extracts sub-pictures from films of dogs, assesses whether the images are sufficient to recognize the emotions of the dogs, and concludes whether they are. Two picture datasets were used for the dog detection trials to test the model's efficacy, and the detection accuracy rates were 97.59% and 94.62%, respectively. When a dog's face features were hidden, when it was a particular breed, when its body was covered, or when the dog region was insufficient, detection errors happened. Three video files with one or more dogs each were used in the dog-tracking trials. When there was just one dog in the video, the tracking accuracy rate was greatest (93.02%), and when there were

several dogs in the film, the tracking accuracy rate was highest (86.45%). When a dog entered or left the screen, the area covered by its body grew, which caused tracking errors and tracking loss. Two video datasets were used for the tests on canine emotion recognition. The accuracies of emotion recognition were 81.73% and 76.02%, respectively[71, 72].

Object classification and detection using cifar-10 data set with intended classification and detection of air-plain images was presented in this work. It used convolutional neural network on keras with tensor flow support, the experimental results showed the time required to train, test and create the model in limited computing system. The study trained the system with 60,000 images with 25 epochs each epoch is taking 722to760 seconds in training step on tensor flow Central Processing Unit (CPU) system. At the end of 25 epochs the training accuracy is 96 percentage and the system can recognition input images based on train model and the output is respective label of images[73, 74].

A novel object detection framework named "Deep Regionlets" by establishing a bridge between deep neural networks and conventional detection schema for accurate generic object detection was proposed by the authors. Motivated by the abilities of regionlets for modeling object deformation and multiple aspect ratios, the study incorporated regionlets into an end-to-end trainable deep learning framework. The deep regionlets framework consists of a region selection network and a deep region let learning module. Specifically, given a detection bounding box proposal, the region selection network provided guidance on where to select regions to learn the features from. The regionlet learning module focuses on local feature selection and transformation to alleviate local variations. To this end, the study first realized non-

rectangular region selection within the detection framework to accommodate variations in object appearance. Moreover, the study design a “gating network" within the regionlet leaning module to enable soft regionlet selection and pooling. The Deep Regionlets framework is trained end-to-end without additional efforts. It performed ablation studies and conduct extensive experiments on the PASCAL VOC and Microsoft COCO datasets. The proposed framework outperforms state-of-the-art algorithms, such as RetinaNet and Mask R-CNN, even without additional segmentation labels[75, 76].

The applications of object detection for video and digital images as well as the detection of objects by web cameras for improved accuracy and real-time implementation were discussed in this research work[77]. For the purpose of preventing occlusion, using a Convolutional Neural Network (CNN) has the drawback of producing several feature maps; nevertheless, this doesn't improve performance. So, in order to solve those problems, the authors proposed a more sophisticated and improved neural network model. In order to perform better than existing approaches while maintaining a relatively high level of accuracy, this research effort has created object detection to analyze digital image/video utilizing a web camera and deep learning method[78].

The authors of this work posited that in order to address the problems of inadequate positioning data and poor target recognition accuracy (convolutional neural network), an optimized model of moving target identification based on CNN must be created. In this paper, the target detection model and the depth semantic segmentation model are combined to provide the target classification information and

semantic location information[79]. The simultaneous fusion of image features carrying different types of information and a pyramid structure of multiscale image features provides the classification and positioning portion of the target detection model, allowing it to use the matched image fusion characteristics to detect targets of different sizes and shapes. Experimental results show that the accuracy rate of this method is 0.941, which is 0.189 greater than the accuracy rate of the LSTM-NMS algorithm. This method has excellent robustness and improves the scene adaptation of feature extraction as well as the precision of moving target position identification through the migration of CNN and the learning of context information[80].

Using RCNN-SR, point tracker, and Kalman filter, a motion information-based detection and tracking technique was proposed in this study. All of the centroid locations of the localized bare hand are mapped to create the trajectory of the gesticulated characters. The original-existence based gesticulated character recognition model is built in this study by using prior knowledge about the characters in order to recognize this gesticulated trajectory. This is an attempt to account for pattern, style, scale, rotation, and illumination differences. By including the boundary information, we have also addressed the case sensitivity between the English lowercase and uppercase alphabets. NITS R-Net, NITS Hand Gesture Database VIIIB, and NITS Gesture Image Databases were also suggested in this work. Several benchmark databases are taken into consideration to assess proposed models[81].

This work proposed AlexNet and region of interest (colour, movement). In comparison to the baseline models, an improvement of 14% is made. A detection and tracking technique that makes use of AlexNet and point-tracker is suggested for

tracking the bare hand. This model improves on the baseline models by about 10%. Oxford hand, OUHands, and EgoHands databases, as well as NITS hand gesture (I-IV, VII), are used to evaluate the proposed models for detection and tracking. Deep convolutional neural networks are used to identify gesture trajectories. This model is able to outperform the baseline recognition models with a relative improvement of about 7%. The NITS hand gesture (IIV, VII), MNIST, and SVHN databases are utilized to assess the performance of the recognition model. We also present the NITS-Net database, which contains photos taken with and without hands[82].

Despite the fact that background subtraction is a relatively mature topic, extensive research has been needed to address open problems and advance the development of a generalized moving object recognition framework for real-time applications. The effectiveness of background subtraction is completely dependent on the effectiveness of the subsequent steps in higher level video analytical activities[83, 84].

In terms of foreground identification, the convolutional neural network (CNN) has achieved significant advancements. The foreground detection top-rank background subtraction techniques still have a lot of drawbacks, though. It is difficult to distinguish the real foreground from the complicated background[85, 86].

The essential goal of background modeling in computer vision is removing a scene's static background in order to retrieve the foreground objects. This procedure is a precondition for many computer vision applications, such as object tracking, motion detection, and video monitoring. Many methods have been proposed for backdrop modeling, ranging from simple threshold-based procedures to intricate deep learning models. This study proposes a strategy for the variable illumination problem that

starts with the K.M.M. baseline model pipeline and two pre-processing methods. Additionally, we discuss backdrop modeling's challenges, such as lighting fluctuations, camera jitter, and PTZ, and we identify several promising areas for additional research[86, 87].

The difficult problem of detecting moving objects in a variety of demanding situations has been the focus of numerous research teams. The first uses include static cameras, but as mobile sensors have grown in popularity, studies on moving cameras have developed throughout time. The authors suggested identifying and classifying the many existing approaches found in the literature in this survey. In order to achieve this, the article suggests categorizing various methods based on the choice of scene representation: one plane or numerous portions. The methods are divided into these two groups based on eight different approaches: multi-planes, multi-cameras, motion segmentation, subspace segmentation, plane+parallax, motion compensation, and panoramic background subtraction[88, 89].

When carried out by vertebrate retinas, the detection of moving objects is a simple operation, but it is a challenging computer vision task. Three major contributions from this PhD dissertation project include: a multi-hierarchical spiking neural network (MHSNN) architecture for detecting horizontal and vertical movements; a hybrid sensitive motion detector (HSMD) algorithm for detecting object motion; and a real-time neuromorphic implementation of the HSMD algorithm, the neuromorphic hybrid sensitive motion detector (NeuroHSMD)[90].

Both the military and the civilian worlds have made extensive use of moving object detection in remote sensing image sequences. The small sizes of moving

objects and the rich backgrounds found in remote sensing photos, however, make efficient recognition extremely challenging. The research suggests a real-time moving object recognition approach for remote sensing image sequences to address this issue. This technique combines the motion data from many frames recovered by the motion detection branch with the semantic data from a single image extracted by the object detection branch[91, 92].

A growing interest in finding abnormalities within dynamic situations captured by moving cameras has been prompted by the proliferation of small, affordable cameras, such as dash cameras, body cameras, and cameras mounted on robots. Existing evaluations, however, mostly focus on Video Anomaly Detection (VAD) techniques that assume static cameras. The VAD literature for moving cameras is still fragmented and hasn't received thorough assessments. This work attempts to fill this gap by providing the first thorough analysis of Moving Camera Video Anomaly Detection (MC-VAD). The study examines the research publications on MC-VAD, exposing their flaws and emphasizing associated difficulties[93, 94].

The primary difficult problem in computer vision is moving object detection and tracking (MODT), which is crucial for numerous applications such as robotics, surveillance, navigation systems, armies, and environmental monitoring. There are a number of methods that have been utilized to locate and follow moving objects in surveillance systems. Therefore, it is essential to create new or improved algorithms that are reliable enough to function during both the day and the night. A modified BGS approach was put forth in this work.The video was first divided into its frame-by-frame equivalent, and then these frames were subjected to a modified background

removal technique with adaptive threshold, resulting in the detection of objects. The object that was discovered was tracked using the Kalman filter technique. When compared to existing methodologies, the experimental findings demonstrate that the suggested method can accurately and efficiently detect and track moving objects with less processing time[95, 96].

Zero shot learning (ZSL) locates hidden objects for which there are no training images. Traditional ZSL methods are limited to recognition settings where each test image is assigned to one of several classes of invisible objects. As unseen objects typically only appear as a small portion of a larger scene, we contend that this setting is inappropriate for real-world applications and necessitates both "recognition" and "localization" of the invisible category. We offer a unique "Zero-Shot Detection" (ZSD) problem setting to overcome this issue. ZSD attempts to simultaneouslyidentifyand locate object instances that belong to novel categories without the need of training data[97].

The research offered a comprehensive approach to the ZSD problem that jointly models the intricate interactions between information from the visual and semantic domains. An end-to-end trainable deep network for ZSD is shown in the paper that successfully reduces noise in unsupervised semantic descriptions. In order to accomplish this, we use the idea of meta-classes to create a unique loss function that combines semantic domain clustering and max-margin class separation. We suggest an experimental strategy for the massive ILSVRC dataset that abides by practical challenges, such as the fact that rare classes are more likely to be the unseen ones, in order to establish a benchmark for ZSD. In addition, the study offers a

baseline strategy that is expanded from traditional recognition to the ZSD context. On the ImageNet detection, MSCOCO, and FashionZSD datasets, our thorough tests demonstrate a considerable improvement in performance (in terms of mAP and Recall) on the crucial yet challenging ZSD task[98].

Zero shot learning (ZSL) locates hidden objects for which there are no training images. Traditional ZSL methods are limited to recognition settings where each test image is assigned to one of several classes of invisible objects. This arrangement, according to the article, is unsuitable for real-world applications when unseen things only appear as a discrete component of a larger scene, necessitating the "recognition" and "localization" of the unseen category. This study's new "Zero-Shot Detection" (ZSD) problem setting, which aims to simultaneously recognize and locate object instances belonging to unique categories, without any training data, was introduced to solve this restriction.The research offered a comprehensive approach to the ZSD problem that jointly models the intricate interactions between information from the visual and semantic domains. An end-to-end trainable deep network for ZSD is investigated in the paper, and it is shown to be successful in reducing noise in unsupervised semantic descriptions[99].

Finding instances of objects from a large number of specified categories in real images is the goal of object detection, one of the most fundamental and difficult problems in computer vision. Feature representations may now be learned directly from data using deep learning approaches, which have produced amazing advances in the field of general object detection. The purpose of this work is to present a thorough assessment of the most recent developments in this subject brought about by deep

learning techniques during this period of rapid evolution. This survey includes more than 300 research contributions that span a wide range of general object detection topics, such as detection frameworks, object feature representation, object proposal generation, context modeling, training methods, and evaluation metrics[100].

Deep learning (DL) has a significant impact on many areas of science and has solidified itself as a flexible approach for tackling new problems in the Earth Observation (EO) community. Despite this, the entry barriers for EO researchers are high because the field is crowded and evolving quickly, primarily due to developments in computer vision (CV). This paper provides an overview of the development of DL with a focus on image segmentation and object detection in convolutional neural networks (CNN) in order to remove the hurdles for EO researchers. The study runs from late 2012 through late 2019, when CNN established new benchmarks for image recognition.In order to make it easier to evaluate contemporary DL models, the paper highlights the relationships between the most significant CNN architectures and CV foundational ideas. Additionally, this work provides a brief history of the most prevalent DL frameworks as well as a list of EO datasets. The article reduces the distance between theoretical notions from CV and their practical application in EO by exploring effective DL architectures on these datasets and reflecting on advancements made in CV and their implications on future research in EO[101].

One of the most important and difficult challenges for finding objects in images and videos is object detection. More research on computer vision tasks like object classification, object counting, and object monitoring has recently attracted a

lot of interest. This paper examines object detection strategies and gives a thorough assessment of the relevant literature. The results of the current research work have been summarized using a systematic review, which has also been used to explore seven relevant research topics about object detection. The analysis of conventional, two-stage, one-stage object identification approaches, dataset preparation and the usage of a standard dataset, annotation tools, and performance evaluation metrics were the main areas of interest in the study. Additionally, it has been determined through comparative studies that the offered techniques differ in terms of their design, optimization function, and training methodologies. The performance of the detectors has increased as a result of deep neural networks' astounding achievement in object detection. Future directions for object detection and several research problems have also been explored[102].

Applications for moving object detection and tracking include surveillance, anomaly detection, vehicle navigation, etc. There is a sufficient amount of research on object detection and tracking, as well as several important survey publications. However, due to the difficulty of the issue, there has only been a limited amount of research on camouflage object recognition and tracking. The existing research on this issue has relied either on biological traits of the camouflaged objects or computer vision methods. The article examines existing computer vision methods for tracking and detecting camouflaged objects from a theoretical standpoint. This work also discusses a number of pertinent concerns and the path that this field may go in the future[103, 104].

One of the best neural networks for classification, segmentation, natural language processing (NLP), and video processing is the convolutional neural network (CNN). The CNN is made up of numerous structural layers. Convolution layers, pooling layers, and fully connected layers comprise CNN's architectural framework. Due to CNN's capacity to automatically learn features from images, which requires a vast quantity of training data and powerful processing resources like GPUs, this application has become the most difficult. Numerous CNN architectures have been reported as a result of the availability of the aforementioned resources. This study focuses on how convolution, pooling, and the fully connected layers of the CNN architecture work, as well as the origins of reported architectures, their limitations and benefits, and a comparison of contemporary architecture in terms of the quantity of parameters, architectural depth, and significant contributions[105, 106].

A neural network that has excelled in solving computer vision issues is the convolutional neural network (CNN). CNNs are regarded as the best for extracting information from images and have displayed exceptional performance in image classification, segmentation, and detection. Today, CNN is used to solve the majority of image processing and computer vision-related issues[107]. The capacity of CNN to operate on raw data without any prior knowledge is what accounts for its superior performance in the aforementioned challenge. CNN is an artificial neural network (ANN) with biological inspiration. In CNN, information moves in a feed-forward network in a single direction. Its architecture is identical to that of the visual cortex in the human brain, with sophisticated and simple cells based on several layers[108, 109].

Convolutional Neural Networks (CNNs) have emerged as the de facto standard for many Computer Vision and Machine Learning operations over the past ten years. CNNs are alternating convolutional and subsampling layers feed-forward artificial neural networks (ANNs). If trained on a sizable visual database with ground-truth labels, deep 2D CNNs with millions of parameters and numerous hidden layers can learn complex objects and patterns. They may be used as the principal tool for many engineering applications for 2D signals, such as photos and video frames, with the right training. However, this could not be an effective alternative in many applications compared to 1D signals, particularly if the training data is limited or application-specific. 1D CNNs have recently been proposed as a solution to this problem and have already attained cutting-edge performance levels in a number of applications, including the classification and early diagnosis of personalized biomedical data, the monitoring of structural health, the identification and detection of anomalies in power electronics, and the detection of electrical motor faults. A real-time and affordable hardware implementation is possible as a result of the straightforward and compact configuration of 1D CNNs, which simply carry out 1D convolutions (scalar multiplications and adds). The overall architecture and operating principles of 1D CNNs, as well as their main technical applications, are covered in detail in this paper, with a special emphasis on recent advancements in the field. Finally, their distinctive qualities are highlighted, capping off their cutting-edge performance. On a special website, the benchmark datasets and the main 1D CNN software utilized in those applications are also shared with the public[110,111].

Summary of Gaps in Literature

A review of the above related works with literature analysis revealed that most

of the works adopted CNN as solution model to solve object detection and tracking problems but with or no efforts to address the problem of inability to detect and track fast moving objects and low computational efficiency (low localization accuracy on small objects under partial occlusions, high processing time, low precision, low sensitivity, low specificity, high false positive rate) associated with these existing methods, and as well to enhance CNN technique for object detection and tracking. Therefore, this work evolved a hybrid paradigm that combined the relative strengths and minimized the relative weaknesses of PSO and CNN, called a Hybrid Swarm Intelligence Convolution Neural Network (CNN-HPSO) which was used to detect and track moving objects addressing fast moving objects, low localization accuracy on small objects under partial occlusions, high processing time, low precision, low sensitivity, low specificity, and high false positive rate which are some of the problems associated with the existing models.

Endnotes

[1]J. P. Bhimavarapu, S. Ramaraju, D. Nagajyothi& I. V. Rao. "*Convolutional Neural Network Based Object Detection System for Video Surveillance Application*"**. Concurrency and Computation: Practice and Experience.** Vol. 35, Issue3. 2023.

[2]J. Li, X. Wei, B. Li & Z. Zeng. "*A Survey on Firefly Algorithms*". **Neurocomputing**. 500, 2022, 662-678

[3]R. Sun, T. Lei, Q. Chen, Z. Wang, X. Du, W. Zhao & A. K. Nandi. "*Survey of Image Edge Detection*". **Horizons in Signal Processing.**2. 2022. 826967

[4]A. M. Shiddiqi, E. D. Yogatama & D. A. Navastara. **"***Resource-Aware Video Streaming (RAViS) Framework for Object Detection System Using Deep Learning Algorithm*"**. MethodsX.** 11, 2023, 102285

[5]A. G. Gad. "*Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review*". **Archives of Computational Methods in Engineering.** 29, 2022. 2531-2561

[6]A.S Dave, M. Nagmode & A. Jahagirdar. "*Statistical Survey on Object Detection and Tracking Methodologies*." **International Journal of Scientific & Engineering Research** 4, no. 3 2013. 1-6.

[7]K. Tong, Y. Wu & F. Zhou. **"***Recent Advances in Small Object Detection Based on Deep Learning: A Review*"**. Image and Vision Computing**. 97, 2020, 103910

[8]F. Yang, S. Odashima, S. Yamao, H. Fujimoto, S. Masui&S. Jiang. "*A Unified Multi-view Multi-person Tracking Framework*". **Computational Visual Media**, 2023. 03820

[9]S. M. Al-Jaberi, A. Patel & A. N. AL-Masri. *"Object Tracking and Detection Techniques under GANN Threats: A Systemic Review"*. **Applied Soft Computing.** 139, 2023, 110224

[10]F. Yang, X. Chang, S. Sakti, Y. Wu & S. Nakamura. *"ReMOT: A Model-Agnostic Refinement for Multiple Object Tracking"*.**Image and Vision Computing**. 106, 2021, 104091

[11]B. Munjal, A. R. Aftab, S. Amin, M. D. Brandlmaier, F. Tombari & F. Galasso. *"Joint Detection and Tracking in Videos with Identification Features"*.**Image and Vision Computing**. 100, 2020, 103932

[12]M. Kaushal , B. S. Khehra & A. Sharma. "*Soft Computing Based Object Detection and Tracking Approaches: State-of-the-Art Survey*"**.Applied Soft Computing**.Vol. 70, 2018, 423-464

[13]G. Jemilda, S. Baulkani, D. G. Paul, & J. B. Rajan. "*Tracking Moving Objects in Video*." **JCP** 12, no. 3 2017, 221-229.

[14]A. Y. Omar& M. Shah. "*Object Tracking: A Survey*." **ACMComputing Surveys (CSUR)**no.38, 4. 2006, 1-8.

[15]R. Alagarsamy & D. Muneeswaran. "*Multi-Object Detection and Tracking Using Reptile Search Optimization Algorithm with Deep Learning*"**. *Symmetry.*** *15, 6*, 2023, 1194.

[16]F. Lorenzo, T. A. Johansen, & T. I. Fossen. "*Dead Reckoning of a Fixed-Wing UAV with Inertial Navigation Aided by Optical Flow*." **International Conference on Unmanned Aircraft Systems-ICUAS,** IEEE 2017. 1250-1259.

[17]F. Perez-Hernandez, S. Tabik, A. Lamas, R. Olmos, H. Fujita & F. Herrera. *"Object Detection Binary Classifiers Methodology Based on Deep Learning to Identify Small Objects Handled Similarly: Application in Video Surveillance"*. **Knowledge-Based Systems.** Vol. 194. 2020. 105590

[18]S. A. Akramin, F. Hafiz, & M. H. Ali. "*Motion Detection Techniques Using Optical Flow*." **World Academy of Science, Engineering and Technology** 56 2009: 559-561.

[19]N. A. Al-Thanoon, O. S. Qasim& Z. Y. Algamal. "*A New Hybrid Firefly Algorithm and Particle Swarm Optimization for Tuning Parameter Estimation in Penalized Support Vector Machine with Application in Chemometrics*".

Chemometrics and Intelligent Laboratory Systems. Vol. 184, 2019, 142-152

[20]Y. Yue, J. Wen & X. Chen. "*Improvements on Particle Swarm Optimization Algorithm for Velocity Calibration in Microseismic Monitoring.*" **Earthquake Science**28, no. 4, 2015, 263-273.

[21]O. Keiron, & R. Nash. "*An Introduction to Convolutional Neural Networks.*" **arXiv preprint arXiv***:1511.08458* 2015, 1-5.

[22]Y. Liu, P. Sun, N. Wergeles & Y. Shang. *"A Survey and Performance Evaluation of Deep Learning Methods for Small Object Detection".***Expert Systems with Applications**. Vol. 172, 2021, 114602

[23]E. I. Yuwono, D. Tjondonegoro, G. Sorwar & A. Alaei. **"***Scalability of Knowledge Distillation in Incremental Deep Learning for Fast Object Detection***". Applied Soft Computing.** 129, 2022, 109608

[24]J. Huang, J. Chen & H. Wang. *"A Lightweight and Efficient One-Stage Detection Framework".* **Computers and Electrical Engineering.** Vol. 105. 2023. 108520

[25]G. Wang, H. Ding, M. Duan, Y. Pu, Z. Yang & H. Li. "*Fighting Against Terrorism: A Real-Time CCTV Autonomous Weapons Detection Based on Improved YOLO v4*". **Digital Signal Processing.** 132. 2023. 103790

[26]A. Mhalla, T. Chateau, N. Essoukri & B. Amara. "*Spatio-Temporal Object Detection by Deep Learning: Video-Interlacing to Improve Multi-Object Tracking*". **Image and Vision Computing**. Vol. 88, 2019, 120-131.

[27]F. Yang, Z. Wang, Y. Wu, S. Sakti & S. Nakamura.*"Tackling Multiple Object Tracking with Complicated Motions-Re-Designing the Integration of Motion and Appearance".* **Image and Vision Computing.**Vol. 124, 2022, 104514

[28]D. Cores, V. M. Brea &M. Mucientes. **"***Short-Term Anchor Linking and Long-Term Self-Guided Attention for Video Object Detection".***Image and Vision Computing.** Vol. 110, 2021, 104179

[29]J. Chung, S. Park, D. Pae, H. Choi&M. Lim**. "***Feature-Selection-Based Attentional-Deconvolution Detector for German Traffic Sign Detection Benchmark".* **Electronics**. 12, no3, 2023, 725

[30]M. Xu, H. Wang, H. Chen, H. Li &J. Peng. "*A Fractional-Order Visual Neural Model for Small Target Motion Detection*". **Neurocomputing**. Vol. 550, 2023, 126459

[31]T. Wang, Z. Ma, T. Yang& S. Zou. "*PETNet: A YOLO-Based Prior Enhanced Transformer Network for Aerial Image Detection*".**Neurocomputing**. Vol. 547, 2023, 126384

[32]G. Tian, J. Liu & W. Yang. "*A Dual Neural Network for Object Detection in UAV Images*". **Neurocomputing**. Vol. 443, 2021, 292-301

[33]G. Lee, P. Yonrith, D. Yeo &A. Hong. **"***Enhancing Detection Performance for Robotic Harvesting Systems Through RandAugment***". Engineering Applications of Artificial Intelligence**. Vol.123,Part C, 2023, 106445

[34]G. Jiayuan, H. Hu, L. Wang, Y. Wei, & J. Dai. "*Learning Region Features for Object Detection*." **In Proceedings of the European Conference on Computer Vision (ECCV)**, 2018, 381-395.

[35]L. Yong, R. Wang, S. Shan & X. Chen. "*Structure Inference Net: Object Detection Using Scene-Level Context and Instance-Level Relationships*." **In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**, 2018, 6985-6994.

[36]R. Yun, C. Zhu& S. Xiao. "*Small Object Detection in Optical Remote Sensing Images via Modified Faster R-CNN*." **Applied Sciences.** 8, no. 5 2018, 813.

[37]B. Daulet, A. Zhilisbayev, A. Kuzdeuov, A. Oleinikov, D. Fadeyev, Z. Makhataeva& H. A. Varol. "*Deep Learning Based Object Recognition using Physically-Realistic Synthetic Depth Scenes*." **Machine Learning and Knowledge Extraction.** 1, no. 3, 2019, 883-903.

[38]Z. Zhong-Qiu, P. Zheng, S. Xu & X. Wu. "*Object Detection with Deep Learning: A Review*." **IEEE Transactions on Neural Networks and Learning Systems.** 30, no. 11, 2019, 3212-3232.

[39]W. Xiongwei, D. Sahoo& S. CH Hoi. "*Recent Advances in Deep Learning for Object Detection"*.**Neurocomputing.** 396, 2020, 39-64.

[40]L. Ke, G. Wan, G. Cheng, L. Meng& J. Han. "*Object Detection in Optical Remote Sensing Images: A Survey and a New Benchmark*." **ISPRS Journal of Photogrammetry and Remote Sensing.** 159, 2020, 296-307.

[41]C. Danyang, Z. Chen & L. Gao. "*An Improved Object Detection Algorithm Based on Multi-Scaled and Deformable Convolutional Neural Networks*." **Human-Centric Computing and Information Sciences.** 10, no. 1, 2020, 1-22.

[42]A. Misbah, I. Ahmed, F. A. Khan, F. Qayum& H. Aljuaid. "*Convolutional Neural Network-Based Person Tracking Using Overhead Views*." **International Journal of Distributed Sensor Networks.** 16, no. 6, 2020, 12-20.

[43]S. Upasna, A. Saini& R. Domala. "*Object Tracking in Videos Using CNN*." **In ICDSMLA 2019, Springer, Singapore**, 2020, 520-527.

[44]L. Ye, M. Y. Yang, G. Vosselman&G. Xia. "*Video Object Detection with a*

Convolutional Regression Tracker." **ISPRS Journal of Photogrammetry and Remote Sensing.** 176. 202, 139-150.

[45]Y. Ke, B. Peng, F. Gu, Y. Zhang, S. Wang, Z. Yu& Z. Hu. "*Convolutional Neural Network for Object Detection in Garlic Root Cutting Equipment*." **Foods.** 11, no. 15, 2022, 2197.

[46]A. Bochkovskiy, C.Y Wang& H.Liao."*YOLOv4: Optimal Speed and Accuracy of Object Detection*".**arXiv** 2020,arXiv:2004.10934.

[47]C. Li, L, Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie,etal."*YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications"*.**arXiv** 2022,arXiv:abs/2209.02976.

[48]M.N. Chapel,&T. Bouwmans."*Moving Objects Detection with a Moving Camera: A Comprehensive Review*".**Computer Science Review.** 38, 2020. 100310.

[49]S. Minaeian, J. Liu & Y. J. Son."*Effective and Efficient Detection of Moving Targets from a UAV's Camera*".**IEEE Trans. on Intell. Transp. Syst.**19, 2018, 497-506.

[50]Y.Wu, X. He,& T. Q. Nguyen."*Moving Object Detection with a Freely Moving Camera via Background Motion Subtraction*".**IEEE Trans. Circuits Syst. Video Technology**.27, 2017, 236-248.

[51]A. H. Nuha, & H. N. Abdullah. "*A Novel Real-Time Multiple Objects Detection and Tracking Framework for Different Challenges*." **Alexandria Engineering Journal.** 61, no. 12, 2022, 9637-9647.

[52]Y. Jinrong, E. Yu, Z. Li, X. Li &W. Tao. "*Quality Matters: Embracing Quality Clues for Robust 3D Multi-Object Tracking*." **arXiv preprint arXiv.** 2208. 2022.10976

[53]Z. Jiang, S. Ji, Z. Cai, Y. Zeng& Y. Wang. "*Moving Object Detection and Tracking by Event Frame from Neuromorphic Vision Sensors*." **Biomimetics.**7, no. 1, 2022, 31.

[54]L. Peixuan& J. Jin. "*Time3D: End-to-End Joint Monocular 3D Object Detection and Tracking for Autonomous Driving.*" **In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.** 2022, 3885-3894.

[55]C. Zuo, J. Qian, S. Feng, W. Yin, Y. Li, P. Fan, J. Han, K. Qian & Q. Chen."*Deep Learning in Optical Metrology: A Review*".**Light Sci. Appl**.11, 2022, 39

[56]L. Huang, R. Luo, X. Liu& X. Hao."*Spectral Imaging with Deep Learning"*.**Light Sci. Appl.**11, 2022, 61.

[57]C. Xiao, Q. Yin, X. Ying, R. Li, S. Wu, M. Li, L. Liu, W. An,& Z. Chen."*DSFNet: Dynamic and Static Fusion Network for Moving Object Detection in Satellite Videos*".**IEEE Geoscience and Remote Sensing Letters**. 19, 2022, 1-5.

[58]H. Zhu, X. Yan, H. Tang, Y. Chang, B. Li,& X. Yuan."*Moving Object Detection with Deep CNNs*".**IEEE Access**. 8, 2020,29729-29741.

[59]J. Zhu, Z. Wang, S. Wang,& S. Chen."*Moving Object Detection Based on Background Compensation and Deep Learning*".**Symmetry**. 12, 2020, 1965.

[60]D. Li, B. Mo&J. Zhou."*Boost Infrared Moving Aircraft Detection Performance by Using Fast Homography Estimation and Dual Input Object Detection Network*".**Infrared Phys. Technol**. 123, 2022, 104182.

[61]E. S. Zaid& S. A. Rawashdeh. "*High-Temporal-Resolution Object Detection and Tracking Using Images and Events.*" **Journal of Imaging**8, no. 8, 2022, 210.

[62]F.H. Chan, Y.T. Chen, Y. Xiang&M. Sun, "Anticipating Accidents in Dashcam Videos".**In Proceedings of Asian ConferenceonComputerVision(ACCV)**, 2016, pp. 136–153.

[63]M. Zaheer, A. Mahmood, M. Khan, M. Segu, F. Yu&S. Lee. "*Generative Cooperative Learningfor Unsupervised Video Anomaly Detection".***In Proceedings of the IEEE Conference onComputerVisionandPatternRecognition(CVPR)**, 2022, pp. 14744-14754

[64]Y. Zhang, H. Lu, L. Zhang, X. Ruan, & S. Sakai, "*Video Anomaly Detection Based on Locality Sensitive Hashing Filters*".**Pattern Recognition**, vol. 59, 2016 302-311

[65]G. Pang, C. Shen, L. Cao& A. V. D. Hengel, "*Deep Learning for Anomaly Detection: A Review*," **ACM Computing Surveys (CSUR)**,vol. 54, no. 2, 2021 1-38

[66]S. Chandrakala, K. Deepak& G. Revathy, "*Anomaly Detection in Surveillance Videos: AThematic Taxonomy of Deep Models, Review and Performance Analysis*," **Artificial Intelligence Review**, 2, 3, 2022, 1-50

[67]P. Pareek & A. Thakkar, "*A Survey on Video-Based Human Action Recognition: Recent Updates, Datasets, Challenges, and Applications".* **Artificial Intelligence Review**, vol. 54, no. 3, 2021, 2259-2322

[68]F. Lateef, M. Kas& Y. Ruichek."*Temporal Semantics Auto-Encoding Based Moving Objects Detection in Urban Driving Scenario*".**In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Nagoya, Japan**,11–17July2021;1352–1358.

[69]S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz& D. Terzopoulos. "*Image Segmentation Using Deep Learning: A Survey*". **arXiv**2020, arXiv:2001.05566.

[70]Y. Liu, Y. Wang, S. Wang, T. Liang, Q. Zhao, Z. Tang & H. Ling. "*CBNet: A Novel Composite Backbone Network Architecture for Object Detection*". **arXiv**2019, arXiv:1909.03625.

[71]B. Aarti & R. Kaushal. "*Jaywalking Detection and Localization in Street Scene Videos Using Fine-Tuned Convolutional Neural Networks*." **Multimedia Tools and Applications**. 2023, 1-21.

[72]L. Tsung-Yi, P. Dollár, R. Girshick, K. He, B. Hariharan& S. Belongie. "*Feature Pyramid Networks for Object Detection*." **In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.** 2017, 2117-2125.

[73]C. Ioana, S. V. Bogolin&M. Leordeanu. "*Unsupervised Learning from Video to Detect Foreground Objects in Single Images*." **In Proceedings of the IEEE International Conference on Computer Vision.** 2017. 4335-4343.

[74]H. Jonathan, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer et al. "*Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors*." **In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.** 2017. 7310-7311.

[75]G. A. Iván, J. García-González, R. M. Luque-Baena& E. López-Rubio. "*Automated Labeling of Training Data for Improved Object Detection in Traffic Videos by Fine-Tuned Deep Convolutional Neural Networks*." **Pattern Recognition Letters.** 167. 2023, 45-52.

[76]C. Huan-Yu, C. H. Lin, J. W. Lai&Y. Chan. "*Convolutional Neural Network-Based Automated System for Dog Tracking and Emotion Recognition in Video Surveillance*." **Applied Sciences.** 13, no. 7, 2023, 4596.

[77]D.P. Sudharshan & S. Raj. "*Object Recognition in Images Using Convolutional Neural Network.*" **In 2018 2nd International Conference on Inventive Systems and Control (ICISC), IEEE.** 2018. 718-722.

[78]X. Hongyu, X. X. Wang, Z. Ren, N. Bodla& R. Chellappa. "*Deep Regionlets for Object Detection*." **In Proceedings of the European Conference on Computer Vision (ECCV).** 2018, 798-814.

[79]K. R. Kumar, P. S. Reddy, A. Katuri, K. S. S Rama Reddy, D. Pavan Vamsi& G. Amulya. "*Object Detection Mechanism using Deep CNN Model*." **In 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), IEEE**, 2023, 1354-1362.

[80]Y. Zhe, Z. Bu&Y. Pan. "*Optimization Algorithm of Moving Object*

Detection Using Multiscale Pyramid Convolutional Neural Networks." **Computational Intelligence and Neuroscience.** 2023.

[81]Y. K. Singh, A. M. Kirupakaran, R. H. Laskar& M. K. Bhuyan. "*Detection, Tracking, and Recognition of Isolated Multi-Stroke Gesticulated Characters*." **Pattern Analysis and Applications**. 2023, 1-26.

[82]Y. K. Singh, K. A. Monsley&R. H. Laskar. "*Gesture Objects Detection and Tracking for Virtual Text Entry Keyboard Interface*." **Multimedia Tools and Applications.** 82, no. 4, 2023, 5317-5342.

[83]R. Kalsotra &S. Arora. **"***Background Subtraction for Moving Object Detection: Explorations of Recent Developments and Challenges***". The Visual Computer. 38, 2022, 4151-4178. https://doi.org/**10.1007/s00371-021-02286-0

[84]B. W. Zhang, L. M Wang, Z. Wang, Y. Qiao & H. L. Wang. "*Real-Time Action Recognition with Deeply Transferred Motion Vector CNNs*". **IEEE Trans. Image Process**. 2018, 27, 2326–2339.

[85]R. Kalsotra&S. Arora. "*Performance Analysis of U-Net with Hybrid Loss for Foreground Detection***". Multimedia Systems**. 2022.

[86]Y. Su, J. Liu, F. Xu, X. Zhang&Y. Zuo. "*A Novel Anti-Drift Visual Object Tracking Algorithm Based on Sparse Response and Adaptive Spatial-Temporal Context-Aware*". **Remote Sensing**. 13. 2021, 4672.

[87]B.G. Vishruth, S. J. Brahmadev&A. T. Sivateja. "*Improved Background Subtraction with Histogram Equalization and Adaptive Thresholding*". **IJARCCE** 12(6), 2023, 12654.

[88]K. Nguyen, C. Fookes, S. Sridharan, Y. Tian, F. Liu, X. Liu, and A. Ross, "*The state of aerial surveillance: A survey*," **arXiv preprint** arXiv:2201.03080, 2022. 3

[89]M. Chapel&T. Bouwmans. "*Moving Objects Detection with a Moving Camera: A Comprehensive Review*". **Computer Science Review**. 38 No 3, 2020, 100310

[90]P. Machado. "*Computational Models of Object Motion Detectors Accelerated Using FPGA Technology*". Nottingham Trent University. 2021. DOI:10.13140/RG.2.2.18739.81441

[91]B. Wang, J. Liu, S. Zhu&F. Xu. "*A Dual-Input Moving Object Detection Method in Remote Sensing Image Sequences via Temporal Semantics*". **Remote Sensing**. 15(9), 2023. 2230. DOI:10.3390/rs15092230

[92]S. D. Khan & H. Ullah, "*A Survey of Advances in Vision-Based Vehicle Re-Identification*".**Computer Vision and Image Understanding**, vol. 182, 2019, 50-63

[93]R. Jiao, Y. Wan, F. Poiesi & Y. Wang. **"***Survey on Video Anomaly Detection in Dynamic Scenes with Moving Cameras"***. Arxiv 2023. 07050.** https://doi.org/10.48550/arxiv.2308.07050

[94]B. Ramachandra, M. J. Jones & R. R. Vatsavai. "*A Survey of Single-Scene Video Anomaly Detection,*" **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 44, no. 5, 2022. 2293-2312

[95]S. H. Jeevith& S.Lakshmikanth. "*Detection and Tracking of Moving Object Using Modified Background Subtraction and Kalman Filter*". **International Journal of Electrical and Computer Engineering (IJECE)**. 11(1): 2021. 217. DOI:10.11591/ijece.v11i1.pp217-223

[96]T. Hoeser & C. Kuenzer. "*Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends*". **Remote Sensing**. Vol. 12, issue 10, 2020, 1667 https://doi.org/10.3390/rs12101667

[97]H.S.G.Supreeth & C. M. Patil. "*Efficient Multiple Moving Object Detection and Tracking Using Combined Background Subtraction and Clustering*". **Signal, Image and Video Processing**. vol.12, no.6, 2018. 1097-1105

[98]L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin& B. A. Johnson. "*Deep Learning in Remote Sensing Applications: A Meta-Analysis and Review.* **Remote Sens**.152, 2019, 166-177.

[99]S. Rahman, S. H. Khan&F. Porikli. "*Zero-Shot Object Detection: Joint Recognition and Localization of Novel Concepts*". **International Journal of Computer Vision**. Vol 128, 2020, 2979-2999

[100]L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu&M. Pietikäinen. "*Deep Learning for Generic Object Detection: A Survey"*.**International Journal of Computer Vision**.Vol. 128, 2019, 261-318

[101]K. Li, G. Wan, G. Cheng, L. Meng&J. Han. "*Object Detection in Optical Remote Sensing Images: A survey and ANew Benchmark"*. **Remote Sensing**. 159, **2020**, 296-307.

[102]J. Kaur&W. Singh. "*Tools, Techniques, Datasets and Application Areas for Object Detection in an Image: AReview*". **Multimedia Tools and Applications**. volume 81, 2022.38297-38351

[103]A. Mondal. "*Camouflaged Object Detection and Tracking: A Survey*". **International Journal of Image and Graphics**. Vol. 20, No. 04, 2020. 2050028https://doi.org/10.1142/S021946782050028X

[104]N. Senan, M. Aamir, R. Ibrahim, N. Taujuddin, & W. W. Muda, "*An Efficient Convolutional Neural Network for Paddy Leaf Disease and Pest Classification*".**International Journal of Advanced Computer Science and Applications**, vol. 11, 2020.

[105]N. Senan, F. Wahid, M. Aamir, A. Samad &M. Khan. "*Comparative Analysis of Recent Architecture of Convolutional Neural Network*". **Mathematical Problems in Engineering**. 2022, 7313612

[106]S. Zhang, W. Huang, and C. Zhang, "*Three-channel convolutional neural networks for vegetable leaf disease recognition*," **Cognitive Systems Research**, vol. 53, 2019, 31-41

[107]A. Khan, A. Sohail, U. Zahoora & A. S. Qureshi, "*A Survey of the Recent Architectures of Deep Convolutional Neural Networks*," **Artificial Intelligence Review**, vol. 53, no. 8, 2020, 5455-5516

[108]S. Loussaief & A. Abdelkrim, "*Convolutional Neural Network Hyper-Parameters Optimization Based on Genetic Algorithms*," **International Journal of Advanced Computer Science and Applications**, vol. 9, no. 10, 2018, 252-266

[109]S. Kiranyaz, T. Ince, A. Iosifidis& M. Gabbouj. "*Operational Neural Networks*", **Neural Computing and Applications (Springer-Nature)**. 2020, 1-24, 10.1007/s00521-020-04780-3

[110]S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbnj & D. J. Inman. *"1D Convolutional Neural Networks and Applications: A Survey"*.**Mechanical Systems and Signal Processing**, Volume 151, 2021, 107398

[111]O. Avci, O. Abdeljaber, S. Kiranyaz, M. Hussein, M. Gabbouj& D. Inman. "*A Review of Vibration-Based Damage Detection in Civil Structures: From Traditional Methods to Machine Learning and Deep Learning Applications".* **Mechanical Systems and Signal Processing**, 147. 2021, 107077, 10.1016/j.ymssp.2020.107077

Chapter Three

Methodology

3.1 Research Approach

The goal of this study is to use CNN-HPSO to detect and track objects. Figure 3.1 shows the process for object detection and tracking in a video frame and enumerated as follows:

i. Video sequence datasets were acquired from a standard database online. Multiple frames were sampled from the video sequence. The more frames that were grabbed or sampled, the better, as this improved the system's sensitivity and precision. This made it possible to identify any insignificant movement that took place during the video sequence.

ii. The original picture was converted into a grayscale image using the frame rate display after the video had been divided into frames. The same preprocessing procedures were applied to all of the frames. This was done to lower the pictures' noise level and enhance the outcomes of further processing.

iii. The algorithm known as Background Subtraction was used to identify any sudden or total changes in intensity in the footage.

iv. An Enhanced Particle Swarm Optimization (HPSO) technique was formulated by including parameters such as modified acceleration coefficient and modified velocity component into the standard PSO to avoid premature convergence and imbalance between exploration and exploitation.

v. CNN-HPSO technique was used for edge detection and extraction of the boundary of the image and the object tracking was finally carried out.

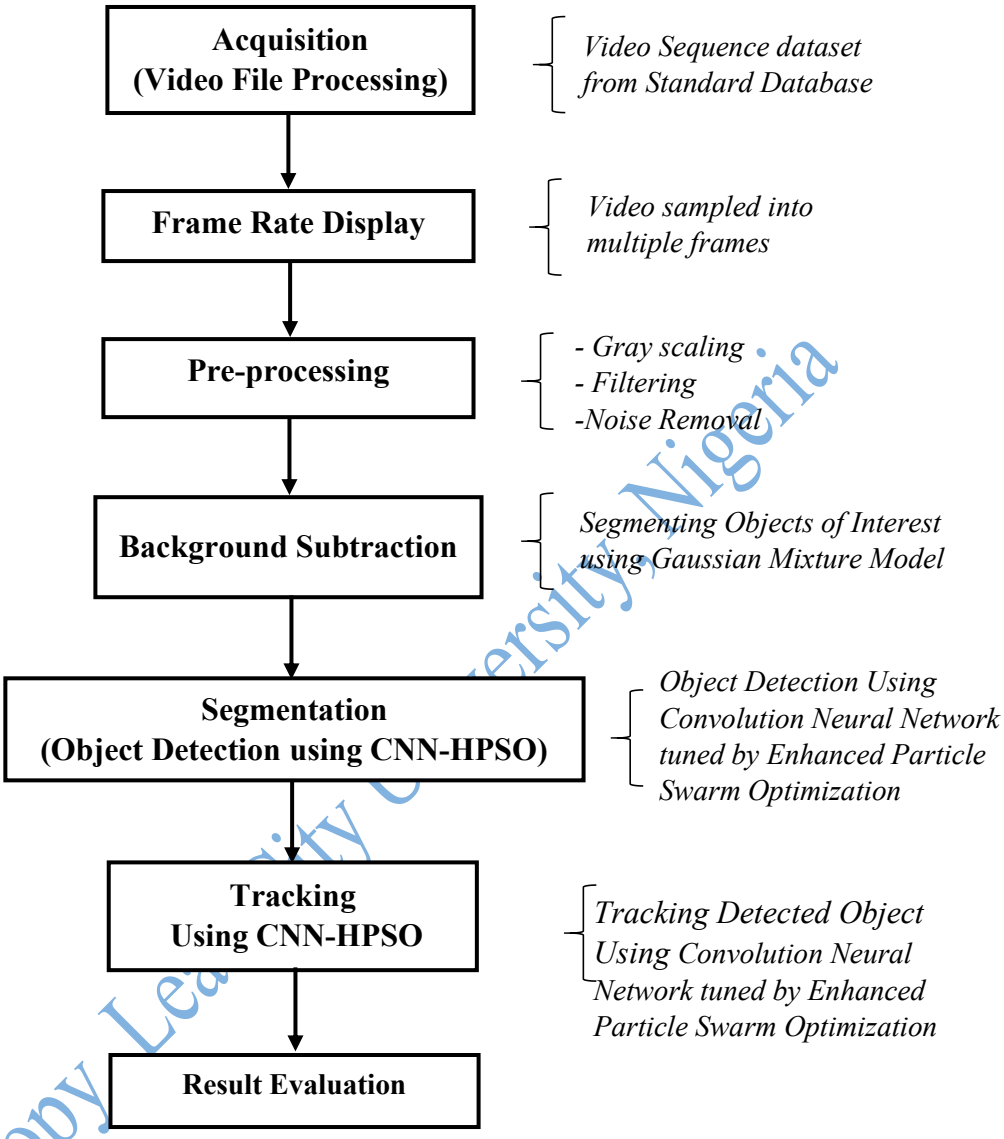


Figure 3.1: The Structure of the Object Detection and Tracking System
(Researcher, Kareem A.E. 2023)

3.2 Acquisition of Video Sequence file

The first steps in moving object detection and tracking is the acquisition of video sequence file either on a real time basis or from an online standard database. For this work, a video file was acquired from online standard database[1], and seven sample video files were acquired on real-time basis via YouTube in AVi and MP4 video formats from the cities of Ogbomoso, Ile-Ife, Ibadan and Lagos[2, 3, 4,5].

3.2.1 Frame Display

The frequency at which an imaging device creates distinct consecutive images is known as the frame rate or frame frequency, and it applies equally to motion capture systems, video cameras, film cameras, and compute graphics. The most common way to describe frame rate is in frames per second, though progressive scan monitors also use the Hertz unit. (Hz). The block that determines and shows the average update rate of the input signals is called the frame rate display block. The video sequence is sampled into numerous frames using this technique, also known as object representation. This makes it possible to identify any very slight movement that may take place during the video sequence. This block was also used to control the stated number of video frames by monitoring the simulation's video frame rate.

3.2.2 Pre-processing

Pre-processing is any signal processing technique used to enhance or modify the raw quality of an image or video. Preprocessing is primarily done to better the raw image by reducing noise and boosting contrast. Additionally, it isolates any interesting items in the image. Preprocessing serves to enhance the video or image so that it increases the likelihood that other processes will be successful. The preprocessing stage involves two major activities: conversion to gray scale and removing noise.

(a) Conversion to Gray Scale

Each frame underwent the same processing steps after being captured. Gray scaling, or grayscale with 0-255 values, is the first step. A grayscale digital picture is one in which each pixel's value in computing is a single sample. Although the samples could theoretically be displayed as shades of any color or even coded with different

colors for different intensities, displayed images of this type are typically made of shades of gray, ranging from black at the weakest intensity to white at the strongest. In order to record 256 intensities, usually on a non-linear scale, grayscale images meant for visual display are typically stored with 8 bits per sampled pixel. This is essential for time and space management.

(b) Noise Reduction

Noise reduction is an essential preprocessing steps to ensure that accurate detection and tracking of objects. This is also important because the Canny edge detector uses a filter based on the first derivative of a Gaussian, because it is susceptible to noise exists in raw unprocessed image data. Thus, at first the raw image is convolved with a Gaussian filter. The result is a slightly blurred version of the original which is not affected by a single noisy pixel to any significant degree.

(c) Background Subtraction

The area of a video frame that differs greatly from a background model is where moving objects can be identified using background subtraction. The basic idea behind background subtraction is to create the objects of interest by thresholding the outcome after subtracting the observed image from the estimated image. The locations of the items of interest are indicated by the regions of the image plane where there is a noticeable difference between the observed and estimated images. In this study, Gaussian Mixture Model was used to segment objects of interest. In this method, a pixel in the current frame is checked against the background model by comparing it with every Gaussian in the model until a same Gaussian is found. When same Gaussian is found the mean and variance of the Gaussian are updated, otherwise a newest Gaussian with the mean equal to the current pixel color and some initial

variance is introduced into the mixture. Every pixel is classified based on whether the matched distribution represents the background process.

3.3 Development of a Hybrid Particle Swarm Optimization Convolution Neural Network Model for Object Detection and Tracking System

The following processes were involved in the development of a Hybrid Particle Swarm Optimization Convolution Neural Network model for Object Detection and Tracking system:

3.3.1 Development of an Enhanced Particle Swarm Optimization

There are some parameters in standard PSO algorithm that affect its performance. The basic PSO parameters are number of iterations, velocity components, and acceleration coefficients. In addition, PSO is also influenced by inertia weight, velocity clamping, and velocity constriction. Some of the deficiencies of PSO are premature convergence, high computational complexity, slow convergence, and sensitivity to parameters. Reasons for such challenges may fall within the fact that PSO does not appropriately handle the relationship between exploitation (local search) and exploration (global search), so it usually converges to a local minimum quickly[6, 7].

As a result, this study was formulated and an enhanced PSO from standard PSO which selects the best combination of weights at the convolution layers and classification layer of the CNNs. The optimal weights selected improved the computational time and accuracy of CNNs network during training and classification process and as well overcome the problem of over fitting. However, PSO parameters such as acceleration coefficient and velocity component were modified to produce HPSO which was believed to handle the relationship between exploitation (local

search) and exploration (global search), to avoid being trapped at local minimum quickly and achieving optimal weights.

3.3.2 Formulation of Objective Function for Assignment of Optimal Weight

The general formulations of an optimal weight determination problem used in this study are as follows:

$$\min_{P_{best,}, G_{best,}, W_f^d} \emptyset(y(W_f^d))$$

Subject to: $C_1: 0 \leq (wt, Obj, P_{best,}, G_{best,}, W^d) \leq 1 \quad W^d \in W_e$ (3.1)

$$C_2: Obj = \begin{cases} 1 & \text{if,} \quad Obj \leq \overline{Obj} \\ 0 & \text{otherwise,} \quad Obj > \overline{Obj} \end{cases}$$

where $wt \in R^n$ is the vectors of randomized weights. $\overline{Obj}$ is the mean square error for Obj. The entire state vector is denoted as $y = [wt]$, where wt is the set of the weights of CNN. The problem will be defined on the weight's horizon $W_e = [W_o^d W_f^d]$. Where W_e consists of original weights of W_o^d of y and final weight W_f^d selected which is equivalent to optimal weight that will be achieved in Algorithm 3.2.

The local and global best position weight-dependent control variables $P_{best}, G_{best} \in R^n$ and possibly the final feature W_f^d are decision variables for optimization. The goal of the optimization is to find the optimal set of decision variables to minimize the objective function $\emptyset$, that is, $\emptyset(y(W_f^d))$.

Constraints (C_1 and C_2) that describe the proper error fitness and weight parameter requirements to be satisfied during the determination of optimal weight, respectively, limit the search space for the optimum. Weight restriction C_1 and objective constraint C_2 were taken into account in the study. C_1 makes ensuring that

the values fall within the 0 to 1 range. C_2 attests that the optimal weights' objective value was marked as 1, while the other weights' objective value was tagged as 0.

3.3.3 Formulation of Acceleration Coefficient and Velocity Component in Standard PSO

The standard PSO has inertia weight ω which controls the momentum of particles by weighting the contribution of the previous velocity and how much memory of the previous flight influences the new velocity. The goal of inertia weight was to eliminate the premature convergence but could not eliminate the effect.

$$\omega = \omega_{min} + (\omega_{max} - iter + 1) \times \frac{\omega_{max} - \omega_{min}}{Max_{iter}} \quad (3.2)$$

where, $iter$ is the current iteration, ω_{max} is the final weight, ω_{min} is the initial weight $\boldsymbol{\omega}$ is the inertia weight employed to overcome the problem of premature convergence, Max_{iter} is the maximum number of iterations.

In this research, modified acceleration coefficients c_1^t, c_2^t and constriction factors were introduced to the standard PSO as against the inertial coefficient ω. This was to prevent the divergence of the particles during the search for solutions in problem space. The coefficient was used to fine-tune the convergence of particle swarm optimization as described in Equation 3.3.

$$c_1^t = c_1 - \frac{t}{Max_{iter}}(c_1)$$
$$c_2^t = c_2 - \frac{t}{Max_{iter}}(c_2) \quad (3.3)$$

Where c_1 is the initialcognitive acceleration coefficient, c_2 is the initial social acceleration coefficient, c_1^t is the modified cognitive acceleration coefficient in the current generation, c_2^t is the modified social acceleration coefficient in the current

generation, Max_{iter} is the maximum number of iterations, and t is the current generation.

The constriction factor (γ) was applied to standard PSO to overcome the problem of premature convergence. Incorporating the factor ensures the quality of convergence and the stability of the generation process for particle swarm optimization.

$$\gamma = \frac{2}{(\rho-2)+\sqrt{\rho^2-4\rho)}} \tag{3.4}$$

where $\rho = c_1+c_2$ depends on the cognitive and social parameters and it is conditioned to$\rho>4$ which ensures the efficiency of the constriction coefficient.

Modified Velocity Component was introduced to reduce the blind spots of the standard PSO by clamping the particle velocity such that the velocity remains within the limits of (V_{std}, V_{mean}), and penalizing the particle velocity, if the sum of the velocity vector and position vector results in the new position of particle outside the boundary limits of the search space. These two modifications ensured that the velocity of the particle was within the confined limits along with the position of the particle within the boundary limits of the search space. The V_{std} and V_{mean} parameters are obtained using the following expression,

$$V_{std,D} = \delta(x_{std,\ D} - x_{mean,D}) \tag{3.5}$$

$$V_{mean,D} = \delta(x_{mean,D} - x_{std,\ D}) \tag{3.6}$$

where, $V_{std,D}$ is the allowed standard deviation velocity of particles, D is the velocity clamping factor, and $x_{std,\ D}$and $x_{mean,D}$ are the standard deviation and mean location values of particles at D^{th} dimension.

So, if the velocity update results in a step that is too large, the standard

deviation velocity limits the velocities as follows:

$$V_{i,D}(t+1) = \begin{cases} V_{std,D}, & \text{if,} \quad V_{i,D}(t) > V_{std,\,D} \\ V_{i,D}(t+1), & \text{if,} \quad V_{i,D}(t) < V_{std,\,D} \end{cases} \tag{3.7}$$

Obviously, unlike acceleration constants that balance the local and global search, the velocity clamping factor controls the convergence speed. A larger velocity clamping factor allows big steps and contributes to a fast convergence rate, but increases the probability that the method will get trapped in local optima. A smaller velocity clamping factor constrains the particle step size, slows down the optimization process, and increases the particles diversity. Essentially, the acceleration constants and velocity clamping factor have corporative roles in the convergence of the PSO algorithm.

The velocity and position of HPSO are given in equations (3.8) and (3.9) as follows.

$${V_{i,D}}^{t+1} = \gamma.[{V_{i,D}}^{t} + c_1^t \ .r_1[P_{best,iD}^t - {x_i}^t] + c_2^t.r_2[G_{best,D}^t - {x_i}^t]] \tag{3.8}$$

$${x_i}^{t+1} = {x_i}^t + {V_{i,D}}^{t+1} \tag{3.9}$$

Where, ${V_{i,D}}^{t+1}$ is modified velocity of particle, ${V_{i,D}}^{t}$ is the current velocity of particle in the D^{th} dimension with(V_{std}, V_{mean}), ${x_i}^{t+1}$ is the modified position of particle, ${x_i}^t$ is the current position of individual particle, γ is inertial weight parameter, $c_1^t \ and \ c_2^t$ is the cognitive and social acceleration factor in the current generation, $r_1 \ and \ r_2$ are uniform random number between $[0,1]$, $P_{best,iD}^t$ is best position of individual $i$in the D^{th} dimension with (V_{std}, V_{mean}) until iteration t, and $G_{best,D}^t$ is the global best position of the group in the D^{th} dimension with(V_{std}, V_{mean}) until iteration t.

The number of particles N is the swarm size and is equivalent to the size of the weight solution. The acceleration constricted coefficients $c_1^t \ and \ c_2^t$ at each of the iterations, together with the random values $r_1 \ and \ r_2$, maintained the stochastic

influence of the cognitive and social components of the particle's velocity respectively, while c_1 and c_2 are two acceleration constants. The constant c_1 expresses how much confidence a particle has, while c_2 expresses how much confidence a particle has in its neighbours. Position and Velocity components will be initialized and are very important for updating particle's position and velocity. The cognitive best position $P^t_{best,iD}$ corresponds to the position in search space where particle iD has the smallest value at an iteration t as determined by the objective function, considering a minimization problem. In addition, the position yielding the lowest value amongst all the cognitive best $P^t_{best,iD}$ is called the global best position which is denoted by $G^t_{best,D}$.

The term $\gamma.{V_{i,D}}^t$ is called inertia weight component that provides a memory of the previous flight direction. The inertia weight was used to balance the global exploration and local exploitation which is updated according to Equation 3.10. The term $c^t_1.r_1\left[P^t_{best,iD} - {x_i}^t\right]$ is called cognitive component which measures the performance of the particles relative to past performances. This part appears to be the particle's finest memory of its position on an individual level. The term $c^t_2.r_2\left[G^t_{best,D} - {x_i}^t\right]$ for $G^t_{best,D}$ EPSO is called social component which measures the performance of the particles relative to a group of particles or neighbours. Equations 3.14 and 3.15 were updated to reflect the acceleration coefficients or factors, which are positive parameters that measure cognitive and social development. Finally, because of the global best position (G_{best}), it returned the ideal CNN weight when the maximum number of iterations had been achieved; otherwise, the procedure was

repeated. Algorithm 3.1 outlines the algorithmic stages for the HPSO technique to achieve optimized CNN weight.

Algorithm 3.1: Enhanced Particle Swarm Optimization (Researcher, Kareem A

Step1: Choose the number of particles

Step 2: Initialize population having positions x_{ij}^0 and velocities v_{ij}^0, set cognitive and social parameter c_1, c_2., $D = max.$ no of dimensions, $n = max.$ no of particles, Max_{iter} =max. no of iteration

Step 3: Set iteration $t = 1$

Step 4: Evaluate the objective function values of particles as $f(x) = f(x_i^t)$

$$f(x) = \sum_{i=1}^{m}\sum_{j=1}^{n} \Delta\left(W_{i,j}^{m,n}\right)\left((x_i) - (x_j)\right)$$

where, x_i^t represent the s at $i = 1,2, \ldots, n$ and $k = 2,3, \ldots, m$

where, $\Delta\left(W_{i,j}^{m,n}\right)\left((x_i) - (x_j)\right)$ is the change in weight of input pixel x along the row and column.

Step 5: Find the cognitive best for each particle as $P_{best,iD}^t = x_i^t$ and global best as: $G_{best,iD}^t = min\{P_{best,iD}^t\}$

Step 6: Calculate constriction coefficient $(\gamma) = \frac{2}{(\rho-2)+\sqrt{\rho^2-4\rho)}}$

where $\rho = c_1 + c_2$ depends on the cognitive and social parameters and the condition $\rho > 4$ assured the efficiency of the constriction coefficient

$c_1^t = c_1 - \frac{t}{Max_{iter}}(c_1), c_2^t = c_2 - \frac{t}{Max_{iter}}(c_2)$

Step 7: Find the new values by the velocity and position of the i^{th} particle in the D^{th} dimension

$V_{std,D} = \delta(x_{std,D} - x_{mean,D})$

$V_{mean,D} = \delta(x_{mean,D} - x_{std,D})$

Where, $V_{std,D}$ is the allowed standard deviation velocity of particles, D is the

velocity clamping factor, and $x_{std,\ D}$ and $x_{mean,\ D}$ are the standard deviation and mean location values of particles at D^{th} dimension.

$$V_{std,\ D} = \delta(x_{std,\ D} - x_{mean,\ D})$$

Where, $V_{std,\ D}$ is the allowed standard deviation velocity of particles, D is the velocity clamping factor, and $x_{max,\ D}$ is the standard deviation location values of particles at D^{th} dimension.

$$V_{i,D}(t+1) = \begin{cases} V_{std,D}, & \text{if, } V_{i,D}(t) > V_{std,\ D} \\ V_{i,D}(t+1), & \text{if, } V_{i,D}(t) < V_{std,\ D} \end{cases}$$

$$V_{i,D}{}^{t+1} = \gamma . V_{i,D}{}^{t} + c_1^t . r_1 \left[P_{best,iD}^t - x_i{}^t\right] + c_2^t . r_2 \left[G_{best,iD}^t - x_i{}^t\right]$$

$$x_i{}^{t+1} = x_i{}^t + V_{i,D}{}^{t+1}$$

Where, $\boldsymbol{r_1}$and $r_2$are random numbers from uniform distribution $U(0,1)$

Step 8: Find the objective function values of x_b^{t+1} as $f(x) = f(x_b^{t+1})$ and find the index of the best particle b

Step 9: Update $Pbest$ of population

$$P_{best,bD}^{t+1} = \begin{cases} P_{best,iD}^t & \text{if, } f(x_b^{t+1}) > P_{best,iD}^t \\ x_b^{t+1} & \text{if, } f(x_b^{t+1}) \leq P_{best,iD}^t \end{cases}$$

Step 10: Update G_{best} of population

$$G_{best,bD} = \begin{cases} \min\left(P_{best,iD}^t\right) & \text{if, } f(x_b^{t+1}) > P_{best,iD}^t \\ \min\left(P_{best,bD}^{t+1}\right) & \text{if, } f(x_b^{t+1}) \leq P_{best,iD}^t \end{cases}$$

Step 11: If t < Max_{iter} then t= $t+1$ and GOTO step 1 else GOTO step l2

Step 12: Output optimum weight selected solution as $G_{best_{bD}}$.

$$G_{best_{bD}} = x_b$$

3.3.4 The Procedural Steps in Achieving the Detection and Tracking Process with CNN-HPSO

The procedural steps in achieving the detection and tracking process with CNN-HPSO is as follows:

Step 1: Forward pass: output of neuron of row k, column y in the l_{th} convolution layer and k^{th} feature pattern in equation (3.10) among them, f is the number of convolution cores in a feature pattern, output of neuron of row x, column y in the l_{th} sub sample layer and k^{th} feature pattern in equation (3.11), the output of the j_{th} neuron in l_{th} hidden layer H in equation (3.12), among them, s is the number of feature patterns in sample layer. output of the i_{th} neuron l_{th} output layer Fin equation (3.5).

$$O_{x,y}^{(l,k)} = \tanh\left(\sum_{t=0}^{f-1}\sum_{r=0}^{K_h}\sum_{c=0}^{K_w} W_{(r,c)}^{(k,t)} O_{(x+r,\ x+c)}^{(l-1,k)} + Bias^{(l,k)}\right) \quad (3.10)$$

$$O_{x,y}^{(l,k)} = \tanh\left(W^{(k)}\sum_{r=0}^{S_k}\sum_{c=0}^{S_w} O_{(x*S_h+r,\ y*S_w+c)}^{(l-1,k)} + Bias^{(l,k)}\right) \quad (3.11)$$

$$O_{(l,f)} = \tanh\left(\sum_{k=0}^{s-1}\sum_{x=0}^{S_h}\sum_{y=0}^{S_w} W_{(x,y)}^{(j,k)} O_{(x,\ y)}^{(l-1,k)} + Bias^{(l,j)}\right) \quad (3.12)$$

$$O_{(l,i)} = \tanh\left(\sum_{j=0}^{H} O_{(l-1,\ j)} W_{(i,\ j)}^{l} + Bias^{(l,i)}\right) \quad (3.13)$$

Step 2: Back propagation: output deviation of the k_{th} neuron in output layer O:

$$d(O_k^o) = y_k - t_k \quad (3.14)$$

Step 3: input deviation of the k_{th} neuron in output layer:

$$d(I_k^o) = (y_k - t_k)\varphi(v_k) = \varphi(v_k)d(O_k^o) \quad (3.15)$$

Step 4: weight and bias variation of k_{th} neuron in output O:

$$\Delta\left(W_{k,x}^o\right) = d(I_k^o)y_{k,x} \quad (3.16)$$

$$\Delta(Bias_k^o) = d(I_k^o) \quad (3.17)$$

Step 5: output bias of kth neuron in hidden layer H, *where th is the threshold*:

$$d\left(O_k^H\right) = \sum_{i=0}^{i<th} d(I_i^o)W_{i,k} \quad (3.18)$$

Step 6: input bias of *k*th neuron in hidden layer *H*:

$$d\left(I_k^H\right) = \varphi(v_k)d(O_k^H) \quad (3.19)$$

Step 7: weight and bias variation in row *x*, column *y* in the *m*th feature pattern, a former layer in front of *k* neurons in hidden layer *H*

$$\Delta\left(W_{m,x,y}^{H,k}\right) = d(I_k^H)y_{x,y}^m \quad (3.20)$$

$$\Delta\left(Bias_k^H\right) = d(I_k^H) \quad (3.21)$$

Step 8: output bias of row *x*, column *y* in *m*th feature pattern, sub sample layer *S*

$$d\left(O_{x,y}^{S,m}\right) = \sum_k^{170} d(I_{m,x,y}^H)W_{m,x,y}^{H,k} \quad (3.22)$$

Step 9: input bias of row *x*, column *y* in *m*th feature pattern, sub sample layer S

$$d\left(I_{x,y}^{S,m}\right) = \varphi(v_k)d(O_{x,y}^{S,m}) \quad (3.23)$$

Step 10: weight and bias variation of row x, column y in k^{th} feature pattern, sub sample layer *S*

$$\Delta(W^{S,m}) = \sum_{x=0}^{fh}\sum_{y=0}^{fw} d(I_{\lfloor x/2\rfloor,\lfloor y/2\rfloor}^{S,m})O_{x,y}^{C,m} \quad (3.24)$$

among them, *C* represents convolution layer.

$$\Delta(Bias^{S,m}) = \sum_{x=0}^{fh}\sum_{y=0}^{fw} d(O_{x,y}^{S,m}) \quad (3.25)$$

Step 11: output bias of row x, column y in k^{th} feature pattern, convolution layer C

$$d\left(O_{x,y}^{C,k}\right) = d(I_{\lfloor x/2\rfloor,\lfloor y/2\rfloor}^{S,k})W^k \quad (3.26)$$

Step 12: input bias of row *x*, column y in k^{th} feature patter, convolution layer C

$$d\left(I_{x,y}^{C,k}\right) = \varphi(v_k)d(O_{x,y}^{C,k}) \tag{3.27}$$

weight variation of row r, column c in m the convolution core corresponding to k^{th} feature pattern in l^{th} layer, convolution C.

$$\Delta\left(W_{r,c}^{k,m}\right) = \sum_{x=0}^{fh}\sum_{y=0}^{fw} d\left(I_{x,y}^{C,k}O_{x+r,y+c}^{l-1,m}\right) \tag{3.28}$$

total bias variation of the convolution core

$$\Delta\left(Bias^{C,k}\right) = \sum_{x=0}^{fh}\sum_{y=0}^{fw} d(O_{x,y}^{C,k}) \tag{3.29}$$

Step 13: Evaluate Objective Function of HPSO based on initial optimal weights.

$$fit = \sum_{i=1}^{m}\sum_{j=1}^{n}\Delta\left(W_{i,j}^{m,n}\right)\left((x_i) - (x_j)\right) \tag{3.30}$$

where, $\Delta\left(W_{i,j}^{m,n}\right)\left((x_i) - (x_j)\right)$ is the change in weight of input pixel x along the row and column.

There are many layers in the architectures but three of the layers contains weight matrices. These three layers requires the training phase. Wepsoi is the optimal weight of the convolution layer, it is used by convolution filters for image processing, Wepso2 Wepsoi contain connection optimal weights of classification neural network. Figure 3.2 described the architecture of the CNN-HPSO. The weight parameters at the feature extraction layer was optimized with HPSO and the weight parameters at the classification layer was optimized with HPSO. Figure 3.3 defined the flowchart of the CNN-HPSO.

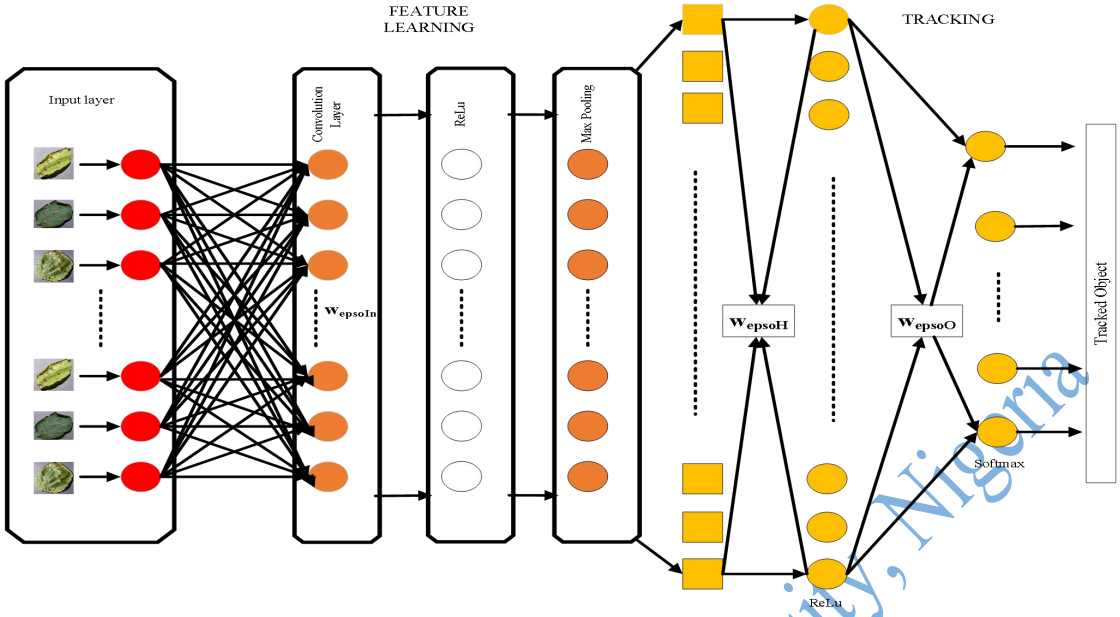


Figure 3.2: Architecture of the CNN-HPSO (Researcher: Kareem A.E. 2023)

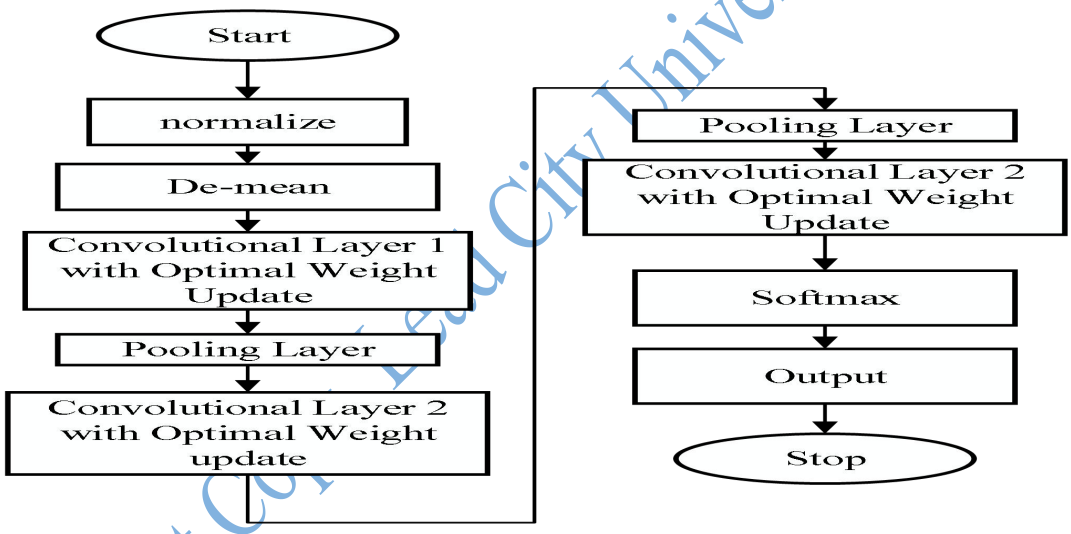


Figure 3.3: Flowchart of the CNN-HPSO (Researcher: Kareem A.E. 2023)

3.4 Implementation of Object Detection and Tracking System

There were two phases in the implementation of the developed technique. This includes the learning (training) phase and the recognition (testing) phase. The performance of the technique was evaluated according to performance metrics. MATLAB R2016 was used as the implementation tool. An interactive Graphic

User Interface (GUI) was developed. Figure 3.4 depicts the flowchart of the technique used.

3.5 Evaluation Measures

The performance of the CNN, CNN-PSO and CNN-HPSO techniques for the detection and tracking of moving objects was evaluated based on accuracy, false acceptance rate, false rejection rate, precision, and computation time. Confusion matrix was used to determine the value of the effectiveness of the performance metrics. False Positive (FP), True Positive (TP), False Negative (FN), and True Negative (TN) are all present.

The foregrounds which are correctly detected are called true positives (TP), whereas the undetected foregrounds are termed as false negatives (FN). The falsely detected objects are referred to as false positives (FP). The true negatives (TN) are the objects which are not wrongly detected as background. The performance metrics such as false alarm rate, precision and accuracy were calculated using these terms.

$$\text{False Acceptance Rate} = \frac{\text{FP}}{\text{TN+FP}} \times 100 \tag{3.31}$$

$$\text{False Rejection Rate} = \frac{\text{FN}}{\text{TP+FN}} \times 100 \tag{3.32}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \times 100 \tag{3.33}$$

$$\text{Overall Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \times 100 \tag{3.34}$$

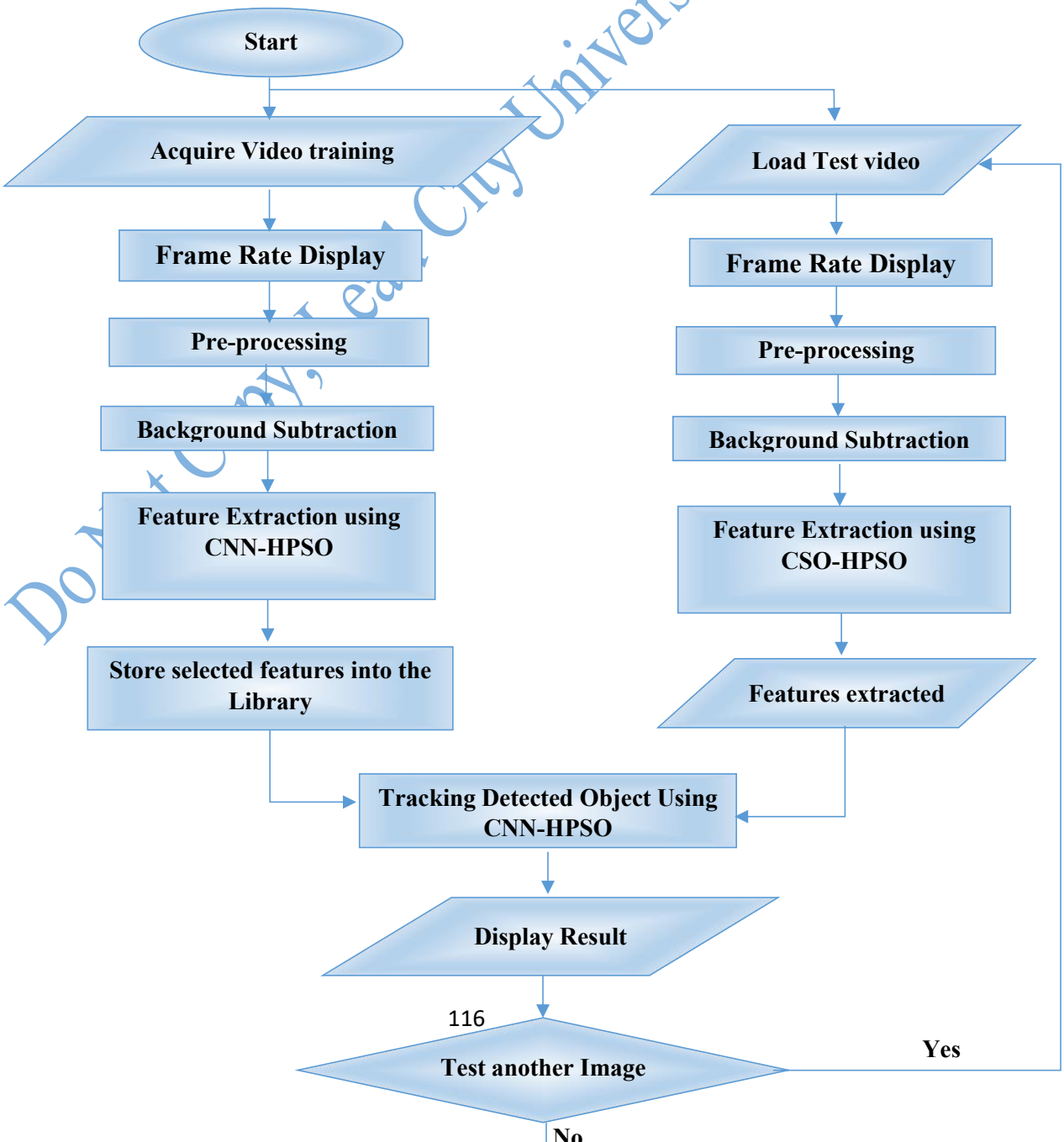


Figure 3.4: Flowchart showing the techniques for Detection and Tracking of Object (Researcher: Kareem A.E. 2023)

Endnotes

[1]Plastonick. "*Motion Object Tracking Sample".* **Plastonick**. 2016. https://www.youtube.com/watch?v=P2toRLWq2Xs. (Available online)

[2]Heternity Media. "*Ogbomoso, Oyo State, Nigeria*". **Heternity Media**. 2020.https://www.youtube.com/watch?v=lyBtCodi55s. (Available online)

[3]Abinibi Hub. "*Ile-Ife, Nigeria: "The Yoruba City of Culture and Traditions*". **Abinibi Hub**. 2022.https://www.youtube.com/watch?v=uOT2jcKAtkk. (Available online)

[4]Top. "*A beautiful Aerial View of Oshodi, Lagos, Nigeria*". **Top: The Official Photography**. 2021.https://www.youtu.be/8fGEQf52e-U?si=VEPxQpr924mchuBY(Available online)

[5]Episode 1. "*See Ibadan From Above*". **HD Drone Footage**. 2021. https://www.youtu.be.ks5CQkzE7kA?si=S8VE-e_vmN729poO

[6]A. Raphael, J. R. Tapamo & A. O. Adewumi. "*Age Estimation via Face Images: ASurvey*." **EURASIP Journal on Image and Video Processing.** no. 1, 2018. 1-35.

[7]Z. Yudong, S. Wang& G. Ji. "*A Comprehensive Survey on Particle Swarm Optimization Algorithm and its Applications*." **Mathematical Problems in Engineering**. 2015. 1-20

Chapter Four

Results and Discussion

4.1 Results

The study was implemented on the computer with Intel(R) Core(TM) i7-480M CPU 2.7 GHz and 8.00G memory. Only four videos with two different formats (Mp4 and Avi) were used to evaluate the techniques in this study. The software is programmed using MATLAB R2016a. The results are presented in this chapter. An extensive evaluation on the acquired videos with some people as moving object based on the CNN-HPSO, CNN-PSO and CNN method. Plates 4.1 - 4.8 show the graphical user interface of the simulation results.

4.2 Results for Video 1

Table 4.1a, Table 4.1b, Table 4.1c, Table 4.1d, Table 4.1e, Table 4.1f, Table 4.1g and Table 4.1h show the performance of CNN, CNN-PSO and CNN-HPSO. In video 1, which is the outcome shown in Table 4.1a, CNN detected 13 objects and 10 moving objects in the Mp4 format. In the Avi format, CNN also found 12 objects and 9 moving objects. For the Mp4 format, CNN-PSO found 15 objects and 13 moving objects, while for the Avi format, it found 13 objects and 11 moving objects. In the Mp4 file, CNN-HPSO detected 16 objects and 14 moving objects, while in the Avi format, it detected 14 objects and 12 moving objects.

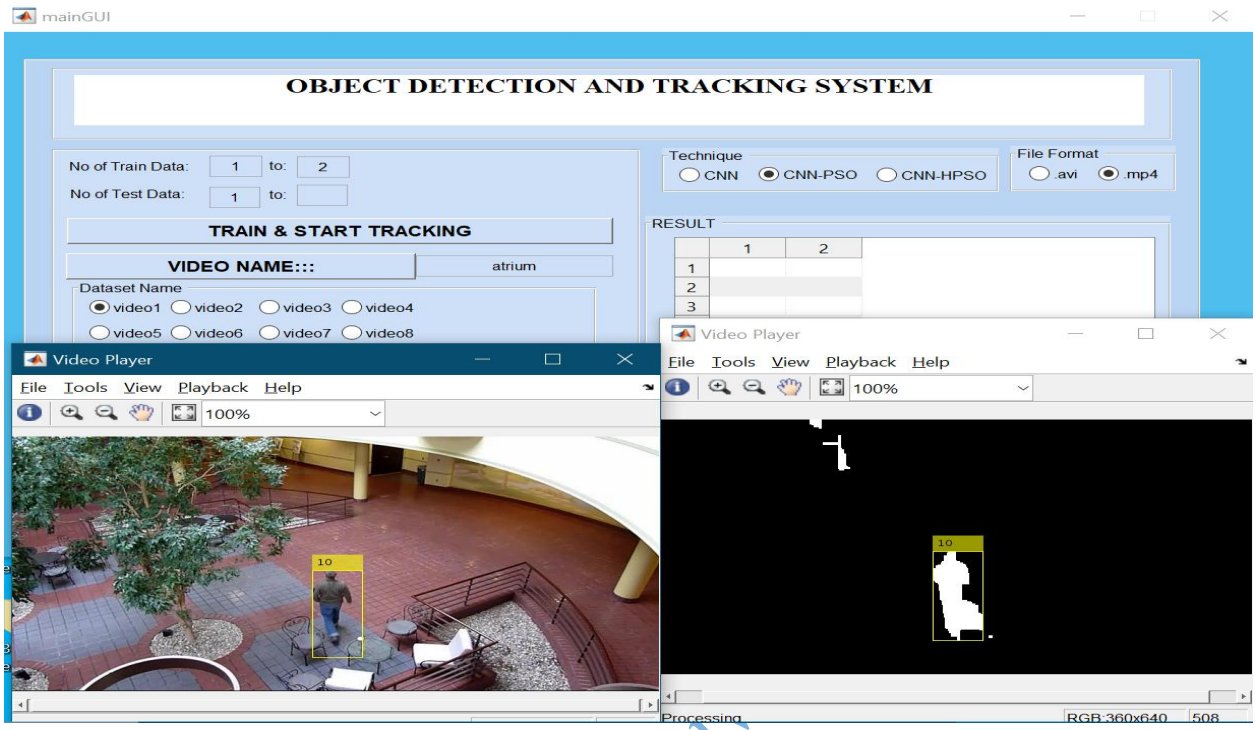


Plate 4.1: Graphical User Interface by CNN, CNN-PSO and CNN-HPSO techniques during Training and Detection with online data-Video1 (Researcher, Kareem A.E. 2023)

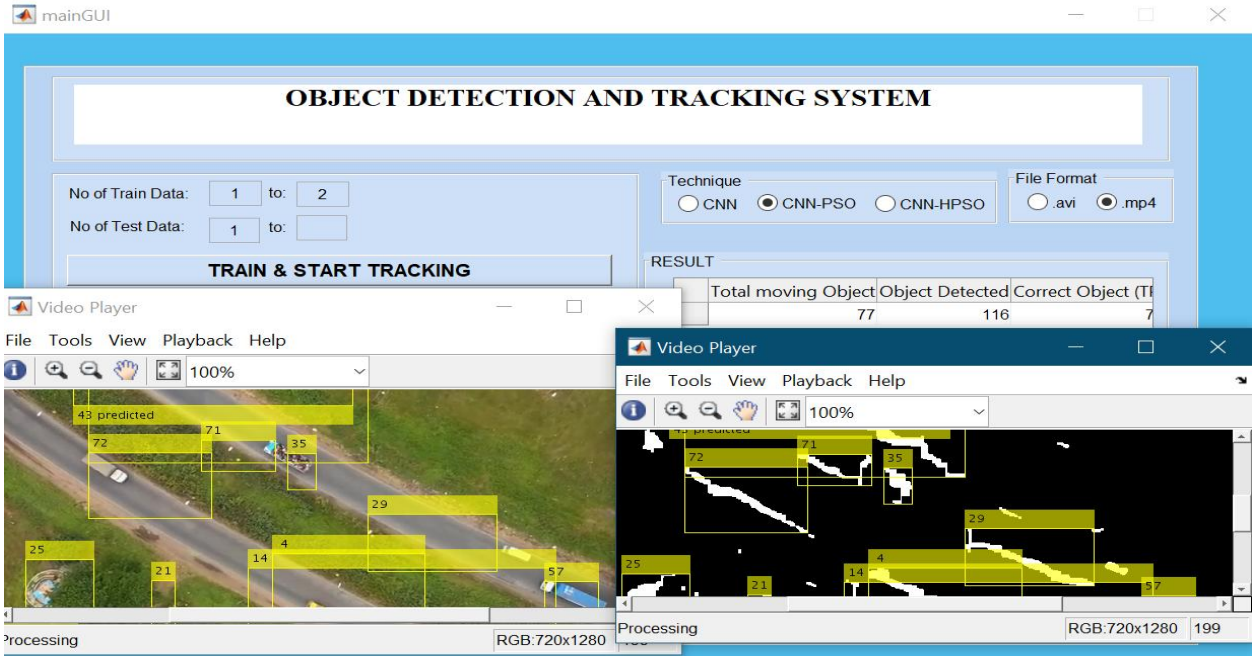


Plate 4.2: Graphical User Interface by CNN, CNN-PSO and CNN-HPSO techniques during Training and Detection with local acquired data for Video2 (Researcher, Kareem A.E. 2023)

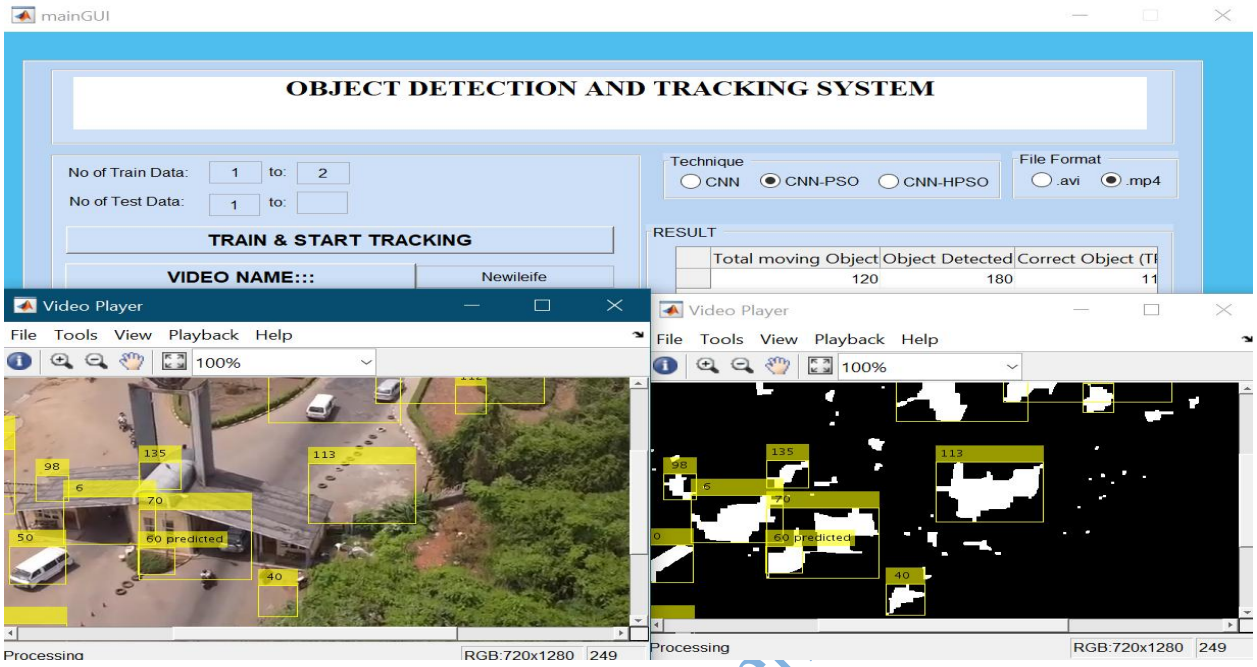


Plate 4.3: Graphical User Interface by CNN, CNN-PSO and CNN-HPSO techniques during Training and Detection with local acquired data for Video3 (Researcher, Kareem A.E. 2023)

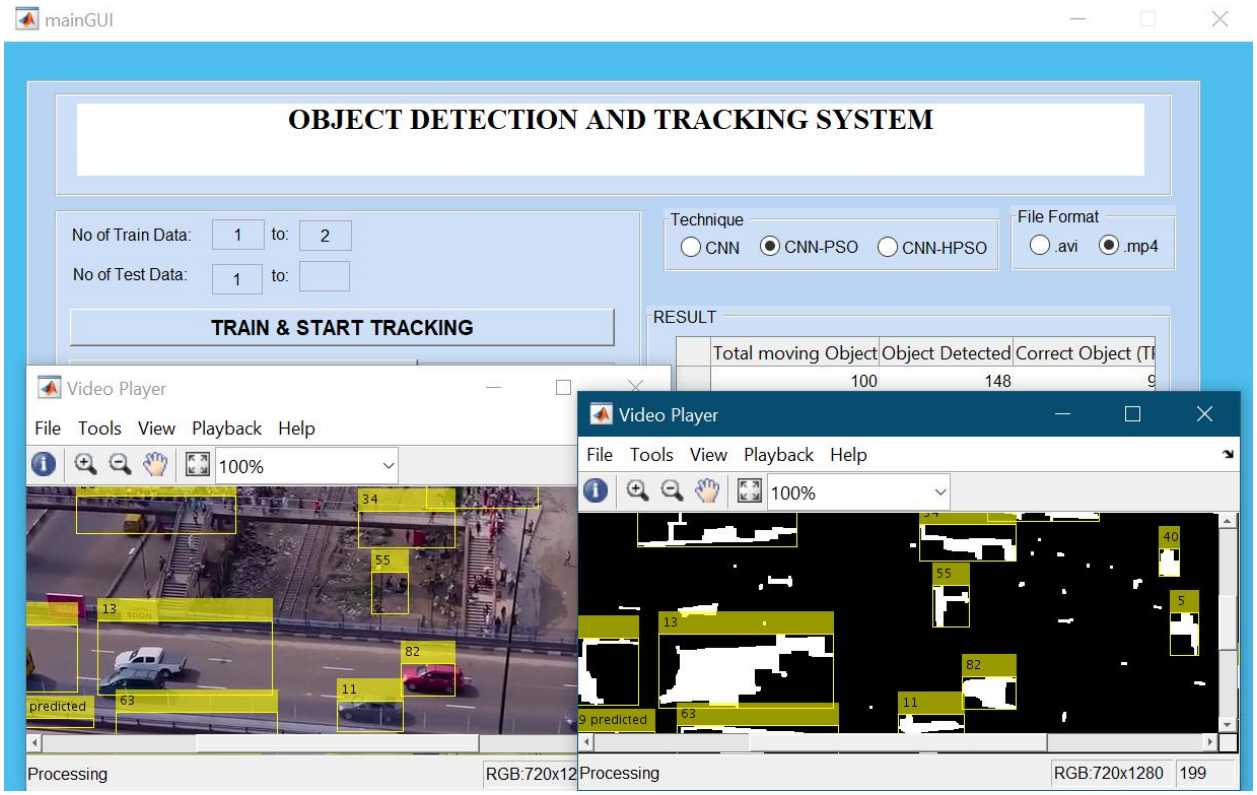


Plate 4.4: Graphical User Interface by CNN, CNN-PSO and CNN-HPSO techniques during Training and Detection with local acquired data for Video4 (Researcher, Kareem A.E. 2023)

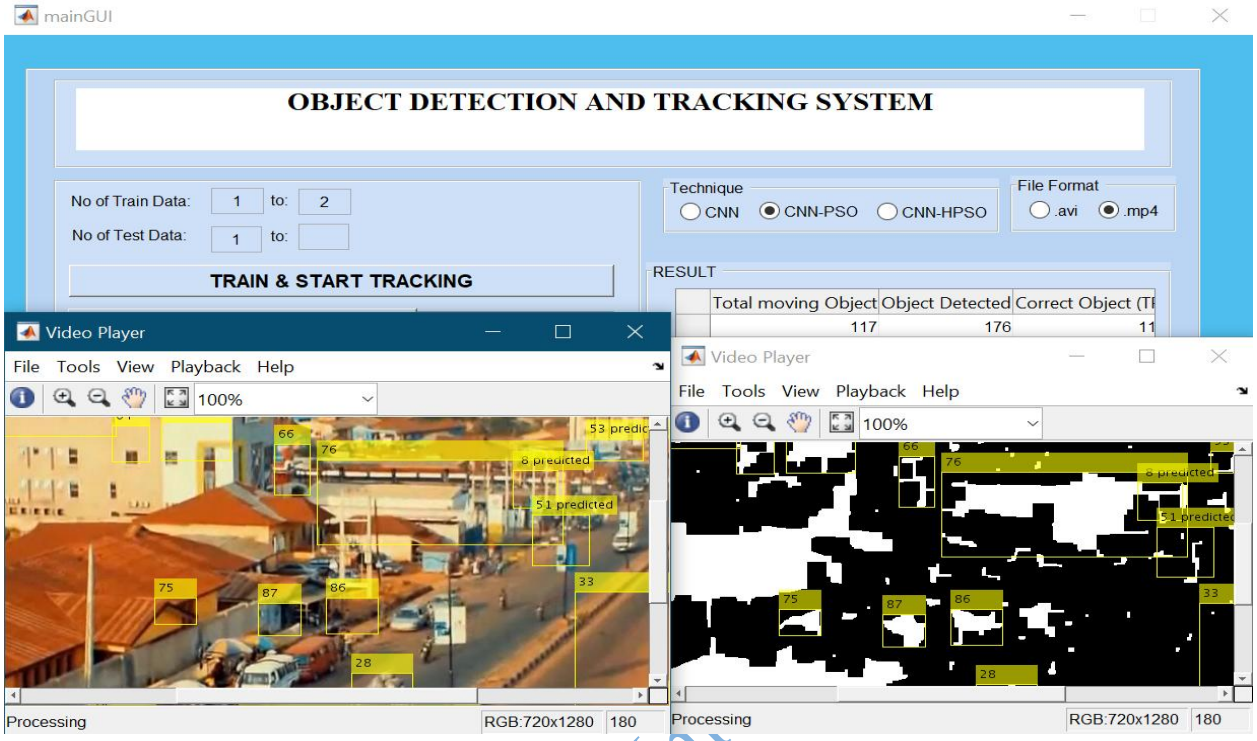


Plate 4.5: Graphical User Interface by CNN, CNN-PSO and CNN-HPSO techniques during Training and Detection with local acquired data for Video5 (Researcher, Kareem A.E. 2023)

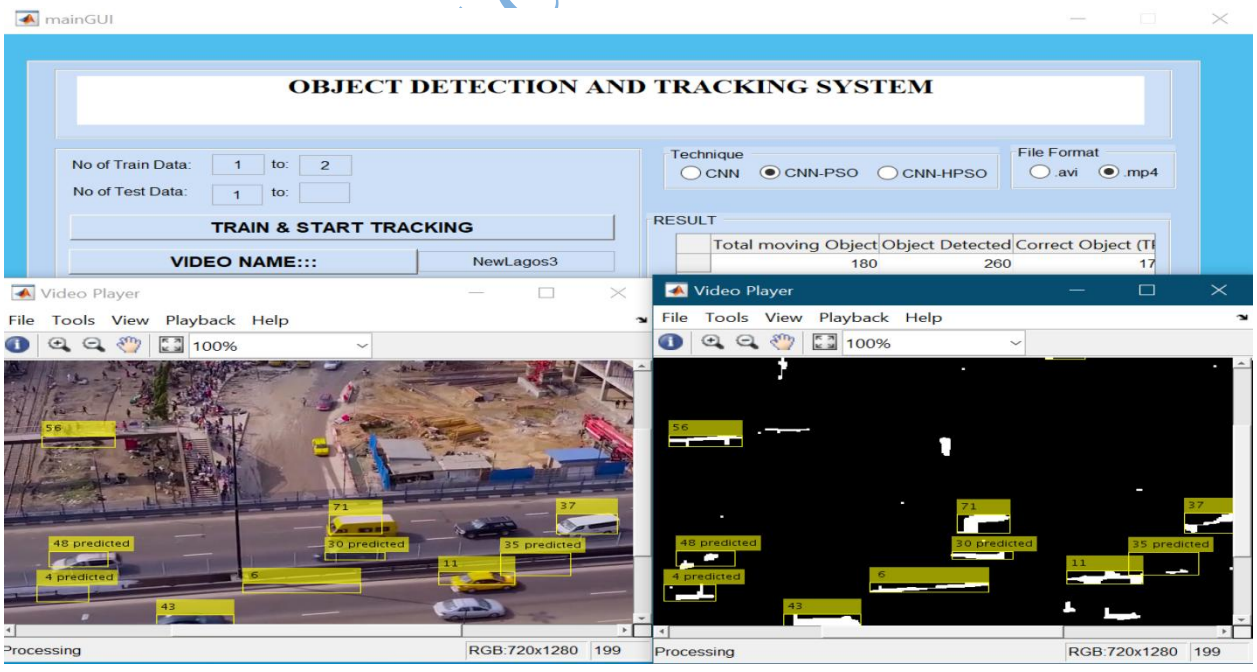


Plate 4.6: Graphical User Interface by CNN, CNN-PSO and CNN-HPSO techniques during Training and Detection with local acquired data for Video6 (Researcher, Kareem A.E. 2023)

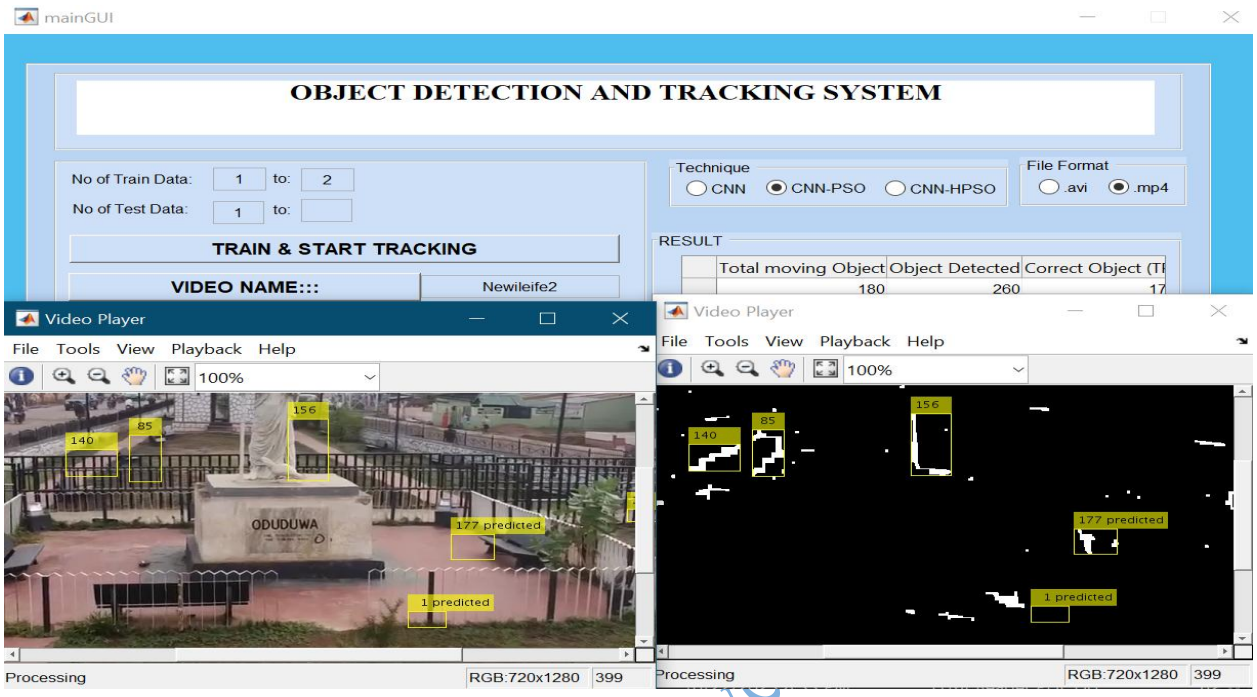


Plate 4.7: Graphical User Interface by CNN, CNN-PSO and CNN-HPSO techniques during Training and Detection with local acquired data for Video7 (Researcher, Kareem A.E. 2023)

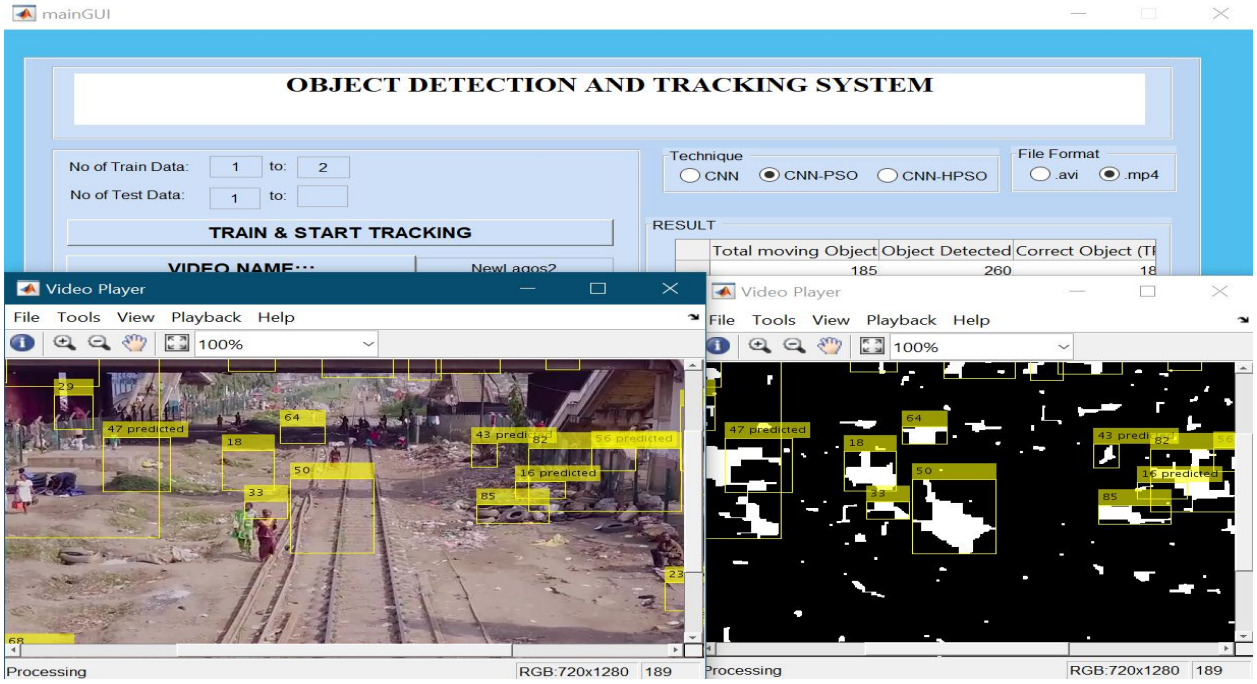


Plate 4.8: Graphical User Interface by CNN, CNN-PSO and CNN-HPSO techniques during Training and Detection with local acquired data for Video8 (Researcher, Kareem A.E. 2023)

Table 4.1: The Performance of CNN, CNN-PSO and CNN-HPSO
(Researcher, Kareem A.E. 2023)

a. Video 1

Technique Used	CNN		CNN-PSO		CNN-HPSO	
Video Type	**Mp4**	**Avi**	**Mp4**	**Avi**	**Mp4**	**Avi**
Total moving Object	10	9	13	11	14	12
Object Detected	13	12	15	13	16	14
Correct Object (TP)	7	7	11	9	13	11
Misclassified Correct Object (FN)	3	2	2	2	1	1
False Static Object (TN)	1	1	1	1	1	1
Misclassified Static Object (FP)	2	2	1	1	1	1
Accuracy (%)	61.54	66.67	80.00	76.92	87.50	85.71
Precision (%)	77.78	77.78	91.67	90.00	92.86	91.67
FPR (%)	66.67	66.67	50.00	50.00	50.00	50.00
Sensitivity (%)	70.00	77.78	84.62	81.82	92.86	91.67
Specificity (%)	33.33	33.33	50.00	50.00	50.00	50.00
Time	38.18	49.11	33.16	41.37	30.93	37.39

b. Video 2

Technique Used	CNN		CNN-PSO		CNN-HPSO	
Video Type	**Mp4**	**Avi**	**Mp4**	**Avi**	**Mp4**	**Avi**
Total moving Object	73	68	77	72	80	77
Object Detected	112	108	116	110	120	115
Correct Object (TP)	66	62	73	69	79	75
Misclassified Correct Object (FN)	7	6	4	3	1	2
False Static Object (TN)	35	37	37	37	39	37
Misclassified Static Object (FP)	4	3	2	1	1	1
Accuracy (%)	90.18	91.67	94.83	96.36	98.33	97.39
Precision (%)	94.29	95.38	97.33	98.57	98.75	98.68
FPR (%)	10.26	7.50	5.13	2.63	2.50	2.63
Sensitivity (%)	90.41	91.18	94.81	95.83	98.75	97.40
Specificity (%)	89.74	92.50	94.87	97.37	97.50	97.37
Time	129.70	159.23	123.43	139.05	113.89	128.49

c. Video 3

Technique Used	CNN		CNN-PSO		CNN-HPSO	
Video Type	**Mp4**	**Avi**	**Mp4**	**Avi**	**Mp4**	**Avi**
Total moving Object	111	105	120	112	127	117
Object Detected	169	158	180	171	186	178
Correct Object (TP)	106	102	116	109	124	114
Misclassified Correct Object (FN)	5	3	4	3	3	3
False Static Object (TN)	51	47	55	56	58	60
Misclassified Static Object (FP)	7	6	5	3	1	1
Accuracy (%)	92.90	94.30	95.00	96.49	97.85	97.75
Precision (%)	93.81	94.44	95.87	97.32	99.20	99.13
FPR (%)	12.07	11.32	8.33	5.08	1.69	1.64
Sensitivity (%)	95.50	97.14	96.67	97.32	97.64	97.44
Specificity (%)	87.93	88.68	91.67	94.92	98.31	98.36
Time	159.22	168.48	150.94	156.15	137.66	147.83

d. Video 4

Technique Used	CNN		CNN-PSO		CNN-HPSO	
Video Type	**Mp4**	**Avi**	**Mp4**	**Avi**	**Mp4**	**Avi**
Total moving Object	95	90	100	93	102	97
Object Detected	143	132	148	137	151	142
Correct Object (TP)	90	86	96	90	100	95
Misclassified Correct Object (FN)	5	4	4	3	2	2
False Static Object (TN)	44	38	46	43	48	44
Misclassified Static Object (FP)	4	4	2	1	1	1
Accuracy (%)	93.71	93.94	95.95	97.08	98.01	97.89
Precision (%)	95.74	95.56	97.96	98.90	99.01	98.96
FPR (%)	8.33	9.52	4.17	2.27	2.04	2.22
Sensitivity (%)	94.74	95.56	96.00	96.77	98.04	97.94
Specificity (%)	91.67	90.48	95.83	97.73	97.96	97.78
Time	129.70	148.63	123.55	139.81	115.26	130.44

e. Video 5

Technique Used	CNN		CNN-PSO		CNN-HPSO	
Video Type	**Mp4**	**Avi**	**Mp4**	**Avi**	**Mp4**	**Avi**
Total moving Object	113	108	117	112	120	117
Object Detected	172	168	176	170	180	175
Correct Object (TP)	106	102	113	109	119	115
Misclassified Correct Object (FN)	7	6	4	3	1	2
False Static Object (TN)	55	57	57	57	59	57
Misclassified Static Object (FP)	4	3	2	1	1	1
Accuracy (%)	93.60	94.64	96.59	97.65	98.89	98.29
Precision (%)	96.36	97.14	98.26	99.09	99.17	99.14
FPR (%)	6.78	5.00	3.39	1.72	1.67	1.72
Sensitivity (%)	93.81	94.44	96.58	97.32	99.17	98.29
Specificity (%)	93.22	95.00	96.61	98.28	98.33	98.28
Time	264.30	324.48	251.51	283.35	232.07	261.84

f. Video 6

Technique Used	CNN		CNN-PSO		CNN-HPSO	
Video Type	**Mp4**	**Avi**	**Mp4**	**Avi**	**Mp4**	**Avi**
Total moving Object	171	165	180	172	187	177
Object Detected	249	238	260	251	266	258
Correct Object (TP)	166	162	176	169	184	174
Misclassified Correct Object (FN)	5	3	4	3	3	3
False Static Object (TN)	71	67	75	76	78	80
Misclassified Static Object (FP)	7	6	5	3	1	1
Accuracy (%)	95.18	96.22	96.54	97.61	98.50	98.45
Precision (%)	95.95	96.43	97.24	98.26	99.46	99.43
FPR (%)	8.97	8.22	6.25	3.80	1.27	1.23
Sensitivity (%)	97.08	98.18	97.78	98.26	98.40	98.31
Specificity (%)	91.03	91.78	93.75	96.20	98.73	98.77
Time	288.31	305.08	273.32	282.77	249.28	267.69

g. Video 7

Technique Used	CNN		CNN-PSO		CNN-HPSO	
Video Type	**Mp4**	**Avi**	**Mp4**	**Avi**	**Mp4**	**Avi**
Total moving Object	160	155	165	158	167	162
Object Detected	226	215	231	220	234	225
Correct Object (TP)	155	151	161	155	165	160
Misclassified Correct Object (FN)	5	4	4	3	2	2
False Static Object (TN)	62	56	64	61	66	62
Misclassified Static Object (FP)	4	4	2	1	1	1
Accuracy (%)	96.02	96.28	97.40	98.18	98.72	98.67
Precision (%)	97.48	97.42	98.77	99.36	99.40	99.38
FPR (%)	6.06	6.67	3.03	1.61	1.49	1.59
Sensitivity (%)	96.88	97.42	97.58	98.10	98.80	98.77
Specificity (%)	93.94	93.33	96.97	98.39	98.51	98.41
Time	261.24	299.37	248.86	281.61	232.16	262.73

h. Video 8

Technique Used	CNN		CNN-PSO		CNN-HPSO	
Video Type	**Mp4**	**Avi**	**Mp4**	**Avi**	**Mp4**	**Avi**
Total moving Object	180	175	185	178	187	182
Object Detected	255	244	260	249	263	254
Correct Object (TP)	175	171	181	175	185	180
Misclassified Correct Object (FN)	5	4	4	3	2	2
False Static Object (TN)	71	65	73	70	75	71
Misclassified Static Object (FP)	4	4	2	1	1	1
Accuracy (%)	96.47	96.72	97.69	98.39	98.86	98.82
Precision (%)	97.77	97.71	98.91	99.43	99.46	99.45
FPR (%)	5.33	5.80	2.67	1.41	1.32	1.39
Sensitivity (%)	97.22	97.71	97.84	98.31	98.93	98.90
Specificity (%)	94.67	94.20	97.33	98.59	98.68	98.61
Time	242.88	278.34	231.37	261.82	215.85	244.27

For video1 in the Mp4 and Avi formats, CNN accurately sensed seven (7) and seven (7) moving objects respectively, misclassified three (3) and two (2) correct objects respectively, detected one (1) and one (1) false static object respectively and misclassified two (2) and two (2) false objects respectively. For video1 in Mp4 format, CNN-PSO accurately identified 11 moving objects, misclassified two (2) correctly identified objects, detected one (1) false static object, and correctly identified one (1) false object. Similarly, for video1 in Avi format, CNN-PSO accurately identified nine (9) moving objects, misclassified two (2) correctly identified objects, detected one (1) false static object, and correctly identified one (1) false object. Additionally, for the Mp4 and Avi formats, CNN-HPSO detected 16 and 14 objects, respectively. For videos in the Mp4 format, the CNN-HPSO technique correctly identified 13 moving objects, misclassified one (1) correctly identified object, incorrectly identified one (1) false static object, and successfully falsely identified one (1) false object. Similarly, for video in the Avi format, CNN-HPSO accurately identified 11 moving objects, misclassified one (1) correctly identified object while incorrectly identifying one (1) false static object and one (1) misclassified false static object.

Additionally, according to Table 4.1, for video1 in Mp4 format, CNN and CNN-PSO achieved accuracy of 61.54% and 80.00%, precision of 77.78% and 91.67%, FPR of 66.67% and 50.00%, sensitivity of 70.00% and 84.62%, and specificity of 33.33% and 50.00%, respectively. Also, for video1 in Avi format, CNN and CNN-PSO obtained accuracy ratings of 66.67% and 76.92%, precision ratings of 77.78% and 90.00%, FPR ratings of 66.67% and 50.00%, sensitivity ratings of 77.78% and 81.82%, and specificity ratings of 33.33% and 50.00%, respectively. Moreover, for video1 in Mp4 format, CNN-HPSO obtained accuracy of 87.50%,

precision of 92.86%, FPR of 50.00%, sensitivity of 92.86%, and specificity of 50.00%. Similar results were obtained for video1 in Avi format using the CNN-HPSO method, which included accuracy of 85.71%, precision of 91.67%, FPR of 50.00%, sensitivity of 91.67%, and specificity of 50.00%.

4.3 Results for Video 2

From the information depicted in Table 4.1b, specifically from video2, CNN detected 112 objects, 73 of which were moving in the Mp4 format, and 108 objects, 68 of which were moving in the Avi format. CNN-HPSO observed 120 objects and 80 moving objects in the Mp4 format and perceived 115 objects and 77 moving objects in the Avi formats respectively, while CNN-PSO detected 116 objects and 77 moving objects for Mp4 format and detected 110 objects and 72 moving objects for Avi format. For video2 in Mp4 and Avi formats, CNN correctly sensed 66 and 62 moving objects; incorrectly classified seven (7) and six (6) correctly detected objects; identified 35 and 37 static objects, and incorrectly classified four (4) and three (3) false static objects, respectively. Correspondingly, CNN-PSO correctly detected 69 moving objects, incorrectly classified three (3) correctly detected objects, detected 37 false static objects, and correctly misclassified one (1) false object in Avi format; it correctly detected 73 moving objects, incorrectly classified four (4) correctly detected objects, detected 37 false static objects, and correctly misclassified two (2) false object in Mp4 format. Additionally, the CNN-HPSO method for videos in the Mp4 file correctly identified 79 moving objects, misclassified one (1) correctly identified object, incorrectly identified 39 static objects, and falsely identified one (1) false static object. Similarly, it properly identified 75 moving objects, misclassified two (2)

correctly identified object, incorrectly identified 37 static objects, and falsely identified one (1) false static object for video in Avi file.

Additionally, according to Table 4.1b, for video2 in Mp4 format, CNN and CNN-PSO achieved accuracy of 90.18% and 94.83%, precision of 94.29% and 97.33%, FPR of 10.26% and 5.13%, sensitivity of 90.41% and 94.81%, and specificity of 89.74% and 94.87%. For video2 in Avi format, CNN and CNN-PSO achieved accuracy of 91.67% and 96.36%, precision of 95.38% and 98.57%, FPR of 7.50% and 2.63%, sensitivity of 91.18% and 95.83%, specificity of 92.50% and 97.37% respectively. Additionally, CNN-HPSO achieved in video2 in Mp4 format, accuracy of 98.33%, precision of 98.75%, FPR of 2.50%, sensitivity of 98.75%, and specificity of 97.50%. Similar results were obtained for video2 in Avi format using the CNN-HPSO technique, which included accuracy of 97.39%, precision of 98.68%, FPR of 2.63%, sensitivity of 97.40%, and specificity of 97.37%.

4.4 Results for Video 3

Based on the information in Table 4.1c, CNN identified 169 objects, 111 of which were moving from video3 in Mp4 file format, and 158 objects, 105 of which were moving from video3 in Avi file format. For Mp4 format, CNN-PSO detected 180 objects out of which 120 were moving objects, while it detected 171 objects out of which 112 were moving objects for Avi format. CNN-HPSO discovered 186 objects and 127 moving objects in the Mp4, while 178 objects and 117 moving objects were found in in the Avi format. For video3 in the Mp4 and Avi formats, CNN accurately identified 106 and 102 moving objects respectively, correctly recognized 5 and 3 false static objects, respectively, while incorrectly identifying 51 and 47 static objects, respectively. However, it incorrectly classified seven (7) and six

(6) of the correctly discovered objects for Mp4 and Avi formats. For video3 in Mp4 format, CNN-PSO accurately identified 116 moving objects, misclassified four (4) correctly identified objects, discovered 55 false static objects, and correctly identified five (5) false objects. The CNN-PSO algorithm correctly identified 109 moving objects in video3 in Avi format, misclassified three (3) correctly identified objects, correctly identified 56 false static objects, and accurately misidentified three (3) false objects.

Additionally, for the Mp4 format, CNN-HPSO correctly identified 124 moving objects, while three (3) correctly identified objects were misclassified, 58 static objects were incorrectly identified, and one (1) false object was mistakenly identified. Also in the Avi format, CNN-HPSO correctly identified 114 moving objects, misclassified three (3) of them, found 60 false static objects, and incorrectly identified one (1) false object.

The accuracy, precision, FPR, sensitivity, and specificity for video3 in Mp4 format were also reached by CNN and CNN-PSO as shown in Table 4.1c, at 92.90% and 95.00%, 93.81% and 95.87%, 12.07% and 8.33%, 95.50% and 96.67%, 87.93% and 91.67% respectively. CNN and CNN-PSO achieved for video3 in Avi format accuracy of 94.30% and 96.49%, precision of 94.44% and 97.32%, FPR of 11.32% and 5.08%, sensitivity for video3 of 97.14% and 97.32%, and specificity of 88.68% and 94.92% respectively. In addition, CNN-HPSO obtained accuracy of 97.85%, precision of 99.20%, FPR of 1.69%, sensitivity of 97.64%, and specificity of 98.31% for a video3 in Mp4 format. Similar outcomes were attained for video3 in Avi format using the CNN-HPSO method, with accuracy, precision, FPR, sensitivity and specificity of 97.75%, 99.13%, 1.64%, 97.44%, and 98.36% respectively.

4.5 Results for Video 4

Based on the information in Table 4.1d, specifically from the video4 in Mp4 and Avi file format, CNN recognized 143 objects in the Mp4 format, 95 of which were moving, whereas in Avi file format, 132 objects were detected, 90 of which were moving. CNN-PSO found 148 objects and 100 moving objects for Mp4 and 137 objects and 93 moving objects for Avi, but CNN-HPSO perceived 151 objects and 102 moving objects for Mp4 and 142 objects and 97 moving objects for Avi. For video4 in the Mp4 and Avi formats, CNN properly identified 90 and 86 moving objects, misclassified correct objects of 5 and 4 objects, false static object of 44 and 38, and while wrongly identified 4 and 4 static objects respectively. For video4 in Mp4 format, CNN-PSO successfully identified 96 moving objects, misclassified 4 correctly identified objects, discovered 46 false static objects, and correctly identified 2 false objects. CNN-PSO correctly identified 90 moving objects in video4 in Avi format, but incorrectly classified 3 correctly identified objects, correctly identified 43 false static objects, and correctly identified 1 false object. Additionally, for the Mp4 and Avi formats, CNN-HPSO found 151 and 142 objects respectively. 100 moving objects in the Mp4 format were successfully detected by the CNN-HPSO approach, whereas 2 correctly identified objects were misclassified, 48 static objects were incorrectly identified, and 1 false object was wrongly identified. Similarly, in video4 in the Avi format, CNN-HPSO correctly identified 95 moving objects, misclassified 2 of them, found 44 false static objects, and incorrectly identified 1 false object.

According to Table 4.1d, CNN and CNN-PSO also achieved accuracy, precision, FPR, sensitivity and specificity with respective values of 93.71% and

95.95%, 95.74% and 97.96%, 8.33% and 4.17%, 94.74% and 96.00%, and 91.67% and 95.83% in the video4 in Mp4 formats. For video4 in Avi format, CNN and CNN-PSO achieved accuracy of 93.94% and 97.08%, precision of 95.56% and 98.90%, FPR of 9.52% and 2.27%, sensitivity of 95.56% and 96.77%, and specificity of 90.48% and 97.73%. Additionally, for a video4 in Mp4 format, CNN-HPSO attained accuracy of 98.01%, precision of 99.01%, FPR of 2.04%, sensitivity of 98.04%, and specificity of 97.96%. The CNN-HPSO approach produced similar results for video4 in Avi format, with accuracy, precision, FPR, sensitivity and specificity of 97.89%, 98.96%, 2.22%, 97.94%, and 97.78%, respectively.

4.6 Results for Video 5

In Table 4.1e of video 5, CNN detected 172 objects and 113 moving objects in the Mp4 format. In the Avi format, CNN also found 168 objects and 1089 moving objects. For the Mp4 format, CNN-PSO found 176 objects and 117 moving objects, while for the Avi format, it found 170 objects and 112 moving objects. In the Mp4 file, CNN-HPSO detected 180 objects and 120 moving objects, while in the Avi format, it detected 175 objects and 117 moving objects.

For video5 in the Mp4 and Avi formats, CNN accurately sensed 106 and 102 moving objects respectively, misclassified seven (7) and six (6) correct objects respectively, detected 55 and 57 false static object respectively and misclassified 4 and 3 false objects respectively. For video5 in Mp4 format, CNN-PSO accurately identified 113 moving objects, misclassified 4 correctly identified objects, detected 57 false static object, and correctly identified 2 false object. Similarly for video1 in Avi format, CNN-PSO accurately identified 109 moving objects, misclassified 3 correctly identified objects, detected 57 false static object, and correctly identified one (1) false

object. For videos in the Mp4 format, the CNN-HPSO technique correctly identified 119 moving objects, misclassified one (1) correctly identified object, incorrectly identified 59 false static object, and successfully falsely identified one (1) false object. Similarly, for video in the Avi format, CNN-HPSO accurately identified 115 moving objects, misclassified 2 correctly identified object while incorrectly identifying 57 false static object and one (1) misclassified false static object.

Additionally, according to Table 4.1, for video5 in Mp4 format, CNN and CNN-PSO achieved accuracy of 93.60% and 96.59%, precision of 96.36% and 98.26%, FPR of 6.78% and 3.39%, sensitivity of 93.81% and 96.58%, and specificity of 93.22% and 96.61% respectively. Also, for video5 in Avi format, CNN and CNN-PSO obtained accuracy ratings of 94.64% and 97.65%, precision ratings of 97.14% and 99.09%, FPR ratings of 5.00% and 1.72%, sensitivity ratings of 94.44% and 97.32%, and specificity ratings of 95.00% and 98.28% respectively. Moreover, for video5 in Mp4 format, CNN-HPSO obtained accuracy of 98.89%, precision of 99.17%, FPR of 1.67%, sensitivity of 99.17%, and specificity of 98.33%. Similar results were obtained for video5 in Avi format using the CNN-HPSO method, which included accuracy of 98.29%, precision of 99.14%, FPR of 1.72%, sensitivity of 98.29%, and specificity of 98.28%.

4.7 Results for Video 6

From the information depicted in Table 4.1f, specifically from video6, CNN detected 249 objects, 171 of which were moving in the Mp4 format, and 238 objects, 165 of which were moving in the Avi format. CNN-HPSO observed 266 objects and 187 moving objects in the Mp4 format and perceived 258 objects and 177 moving objects in the Avi formats respectively, while CNN-PSO detected 260 objects and 180

moving objects for Mp4 format and detected 251 objects and 172 moving objects for Avi format. For video6 in Mp4 and Avi formats, CNN correctly sensed 166 and 162 moving objects; incorrectly classified five (5) and three (3) correctly detected objects; identified 71 and 67 static objects, and incorrectly classified seven (7) and six (6) false static objects respectively. Correspondingly, CNN-PSO correctly detected 169 moving objects, incorrectly classified three (3) correctly detected objects, detected 76 false static objects, and correctly misclassified three (3) false object in Avi format; it correctly detected 176 moving objects, incorrectly classified four (4) correctly detected objects, detected 75 false static objects, and correctly misclassified five (5) false object in Mp4 format. Additionally, the CNN-HPSO method for videos in the Mp4 file correctly identified 184 moving objects, misclassified three (3) correctly identified object, incorrectly identified 78 static objects, and falsely identified one (1) false static object. Similarly, it properly identified 174 moving objects, misclassified three (3) correctly identified object, incorrectly identified 80 static objects, and falsely identified one (1) false static object for video in Avi file.

Additionally, according to Table 4.1f, for video6 in Mp4 format, CNN and CNN-PSO achieved accuracy of 95.18% and 96.22%, precision of 95.95% and 97.24%, FPR of 8.97% and 6.75%, sensitivity of 97.08% and 97.78%, and specificity of 91.03% and 93.75%. For video6 in Avi format, CNN and CNN-PSO achieved accuracy of 96.22% and 97.61%, precision of 96.43% and 98.26%, FPR of 8.22% and 3.80%, sensitivity of 98.18% and 98.26%, specificity of 91.75% and 96.20% respectively. Additionally, CNN-HPSO achieved in video6 in Mp4 format, accuracy of 98.50%, precision of 99.46%, FPR of 1.27%, sensitivity of 98.40%, and specificity of 98.73%. Similar results were obtained for video6 in Avi format using the CNN-

HPSO technique, which included accuracy of 98.45%, precision of 99.43%, FPR of 1.23%, sensitivity of 98.31%, and specificity of 98.77%.

4.8 Results for Video 7

Based on the information in Table 4.1g, CNN identified 226 objects, 160 of which were moving from video7 in Mp4 file format, and 215 objects, 155 of which were moving from video7 in Avi file format. For Mp4 format, CNN-PSO detected 231 objects out of which 165 were moving objects, while it detected 220 objects out of which 158 were moving objects for Avi format. CNN-HPSO discovered 234 objects and 167 moving objects in the Mp4, while 225 objects and 162 moving objects were found in in the Avi format. For video7 in the Mp4 and Avi formats, CNN accurately identified 155 and 151 moving objects respectively, correctly recognized 62 and 56 false static objects, respectively, while incorrectly identifying four (4) and four (4) static objects respectively. However, it incorrectly classified five (5) and four (4) of the correctly discovered objects for Mp4 and Avi formats. For video7 in Mp4 format, CNN-PSO accurately identified 161 moving objects, misclassified four (4) correctly identified objects, discovered 64 false static objects, and correctly misclassified two (2) static objects. The CNN-PSO algorithm correctly identified 155 moving objects in video7 in Avi format, misclassified three (3) correctly identified objects, correctly identified 61 false static objects, and accurately misidentified one (1) static object.

Additionally, for the Mp4 format, CNN-HPSO correctly identified 165 moving objects, while two (2) correctly identified objects were misclassified, 66 static objects were incorrectly identified, and one (1) false object was mistakenly identified. Also in the Avi format, CNN-HPSO correctly identified 160 moving objects,

misclassified two (2) of them, found 62 false static objects, and incorrectly identified one (1) false object.

The accuracy, precision, FPR, sensitivity, and specificity for video3 in Mp4 format were also reached by CNN and CNN-PSO as shown in Table 4.1g, at 96.02% and 97.40%, 97.48% and 98.77%, 6.06% and 3.03%, 96.88% and 97.58%, 93.94% and 96.97% respectively. CNN and CNN-PSO achieved for video7 in Avi format accuracy of 96.28% and 98.18%, precision of 97.42% and 99.36%, FPR of 6.67% and 1.61%, sensitivity for video7 of 97.42% and 98.10%, and specificity of 93.33% and 98.39% respectively. In addition, CNN-HPSO obtained accuracy of 98.72%, precision of 99.40%, FPR of 1.49%, sensitivity of 98.80%, and specificity of 98.51% for a video7 in Mp4 format. Similar outcomes were attained for video7 in Avi format using the CNN-HPSO method, with accuracy, precision, FPR, sensitivity and specificity of 98.67%, 99.38%, 1.59%, 98.77%, and 98.41% respectively.

4.9 Results for Video 8

Based on the information in Table 4.1h, specifically from the video8 in Mp4 and Avi file format, CNN recognized 255 objects in the Mp4 format, 180 of which were moving, whereas in Avi file format, 244 objects were detected, 175 of which were moving. CNN-PSO found 260 objects and 185 moving objects for Mp4 and 249 objects and 178 moving objects for Avi, but CNN-HPSO perceived 263 objects and 187 moving objects for Mp4 and 254 objects and 182 moving objects for Avi. For video8 in the Mp4 and Avi formats, CNN properly identified 175 and 171 moving objects, misclassified correct of 5 and 4 objects, false static object of 71 and 65 and while wrongly identified 4 and 4 static objects respectively. For video8 in Mp4 format, CNN-PSO successfully identified 181 moving objects, misclassified 4 correctly

identified objects, discovered 73 false static objects, and correctly identified 2 false objects. CNN-PSO correctly identified 175 moving objects in video8 in Avi format, but incorrectly classified 3 correctly identified objects, correctly identified 70 false static objects, and correctly identified 1 false object. Additionally, for the Mp4 and Avi formats, CNN-HPSO found 185 and 180 objects respectively. whereas 2 correctly identified items were misclassified, 75 static objects were incorrectly identified, and 1 false object was wrongly identified. Similarly, in video8 in the Avi format, CNN-HPSO correctly identified 180 moving objects, misclassified 2 of them, found 71 false static objects, and incorrectly identified 1 false object.

According to Table 4.1h, CNN and CNN-PSO also achieved accuracy, precision, FPR, sensitivity and specificity with respective values of 96.47% and 97.69%, 97.77% and 98.91%, 5.33% and 2.67%, 97.22% and 97.84%, and 94.67% and 97.33% in the video8 in Mp4 formats. For video8 in Avi format, CNN and CNN-PSO achieved accuracy of 96.72% and 98.39%, precision of 97.71% and 99.43%, FPR of 5.80% and 1.41%, sensitivity of 97.71% and 98.31%, and specificity of 94.20% and 98.59%. Additionally, for a video8 in Mp4 format, CNN-HPSO attained accuracy of 98.86%, precision of 99.46%, FPR of 1.32%, sensitivity of 98.93%, and specificity of 98.68%. The CNN-HPSO approach produced similar results for video8 in Avi format, with accuracy, precision, FPR, sensitivity and specificity of 98.82%, 99.45%, 1.39%, 98.90%, and 98.61%, respectively.

4.10 Processing Time for the Techniques

The processing times for CNN, CNN-PSO, and CNN-HPSO methods are shown in Table 4.2. According to the results shown in the table, CNN-HPSO achieves processing times of 30.93 seconds for Mp4 video format and 37.39 seconds for Avi

video format, while CNN-PSO achieves processing times of 33.16 seconds and 41.37 seconds for Mp4 and Avi video formats, respectively and CNN technique realizes processing times of 38.18 seconds and 49.11 seconds for Mp4 video format and Avi video format respectively, using the first video. Additionally, CNN-HPSO technique achieved processing times of 113.89 seconds for Mp4 video format and 128.49 seconds for Avi video format, while CNN-PSO technique achieved processing times of 123.43 seconds and 139.05 seconds for Mp4 and Avi video formats and CNN technique achieved processing times of 129.70 seconds and 159.23 seconds for Mp4 video format and Avi video format respectively, using the second video. Furthermore, for video3, CNN-PSO produced processing times of 150.94 seconds and 156.15 seconds for Mp4 and Avi video formats respectively, and CNN technique produced processing times of 159.22 seconds and 168.48 seconds for Mp4 and Avi video formats respectively, while CNN-HPSO realized processing times of 137.66 seconds and 147.83 seconds for Mp4 and Avi video formats respectively. In addition, for video4, CNN-HPSO technique had processing times of 115.26 seconds for Mp4 video format and 130.44 seconds for Avi video format, while CNN-PSO technique had processing times of 123.55 seconds and 139.81 seconds for Mp4 and Avi video formats, and CNN technique had processing times of 129.70 seconds and 148.63 seconds for Mp4 and Avi video formats respectively.

In video5, CNN-HPSO achieves processing times of 232.07 seconds for Mp4 video format and 261.84 seconds for Avi video format, while CNN-PSO achieves processing times of 251.51 seconds and 283.35 seconds for Mp4 and Avi video formats respectively and CNN technique realizes processing times of 264.30 seconds and 324.48 seconds for Mp4 video format and Avi video format respectively.

Additionally, in video6, CNN-HPSO technique achieved processing times of 249.28 seconds for Mp4 video format and 267.69 seconds for Avi video format, while CNN-PSO technique achieved processing times of 273.32 seconds and 282.77 seconds for Mp4 and Avi video formats and CNN technique achieved processing times of 288.31 seconds and 305.08 seconds for Mp4 video format and Avi video format respectively. Furthermore, for video7, CNN-PSO produced processing times of 248.86 seconds and 281.61 seconds for Mp4 and Avi video formats respectively, and CNN technique produced processing times of 261.24 seconds and 299.37 seconds for Mp4 and Avi video formats respectively, while CNN-HPSO realized processing times of 232.16 seconds and 262.73 seconds for Mp4 and Avi video formats respectively. In addition, for video8, CNN-HPSO technique had processing times of 215.85 seconds for Mp4 video format and 244.27 seconds for Avi video format, while CNN-PSO technique had processing times of 231.37 seconds and 261.82 seconds for Mp4 and Avi video formats, and CNN technique had processing times of 242.88 seconds and 278.34 seconds for Mp4 and Avi video formats respectively.

As a consequence, moving object detection and tracking take less time to process when PSO and HPSO are used in conjunction with CNN.

Table 4.2: Processing Time for the Techniques (Researcher, Kareem A.E. 2023)

Video Format	**Mp4**			**Avi**		
Techniques	**CNN**	**CNN-PSO**	**CNN-HPSO**	**CNN**	**CNN-PSO**	**CNN-HPSO**
Video 1	38.18	33.16	30.93	49.11	41.37	37.39
Video 2	129.70	123.43	113.89	159.23	139.05	128.49

Video 3	159.22	150.94	137.66	168.48	156.15	147.83
Video 4	129.70	123.55	115.26	148.63	139.81	130.44
Video 5	264.30	251.51	232.07	324.48	283.35	261.84
Video 6	288.31	273.32	249.28	305.08	282.77	267.69
Video 7	261.24	248.86	232.16	299.37	281.61	262.73
Video 8	242.88	231.37	215.85	278.34	261.82	244.27

4.11 Discussion of Results

In comparison to CNN and CNN-PSO according to the results of the research, CNN-HPSO techniques were more effective at detecting and tracking of moving objects. A demonstration of the effectiveness of the methods for segmenting and tracking moving objects is provided by the graphical representation of the simulation produced by the technique. The shadows cast by the people moving in the video were also included in the segmented picture, as shown by the binary representation of moving objects. The processing times for each method, based on various video formats, are shown in Figure 4.1. When PSO and HPSO are used in conjunction with CNN, moving objects are quickly detected and tracked, resulting in a faster processing time than with CNN alone. The processing period for Avi video formats is longer than for Mp4 video formats. This is because the Avi video format is superior to the Mp4 format in terms of quality, size, and lossless compression. This supports the forensic study of video file formats hypothesis that high processing is necessary for Avi video formats because of its size and high quality[4].

Additionally, this result confirmed the literature that various video file formats are usually processed at different speeds and that the processing time increases with file size and quality[5]. As a consequence, when compared to CNN-PSO and CNN, CNN-HPSO has a faster processing time according to the results of this study.

Figures 4.1- 4.6 show graphs illustrating the effectiveness of the study's methods based on performance metrics. According to the graphs, CNN-HPSO technique had the best accuracy, precision, FPR, sensitivity, and specificity.

Figure 4.1 depicts the processing times achieved by CNN-HPSO, CNN-PSO and CNN techniques with respect to the video formats: Video1.mp4, Video2.mp4, Video3.mp4, Video4.mp4, Video5.mp4, Video6.mp4, Video7.mp4, Video8.mp4, Video1.avi, Video2.avi, Video3.avi, Video4.avi, Video5.avi, Video6.avi, Video7.avi and Video8.avi respectively, it was found that, compared to the two methods, CNN produced the highest processing time.

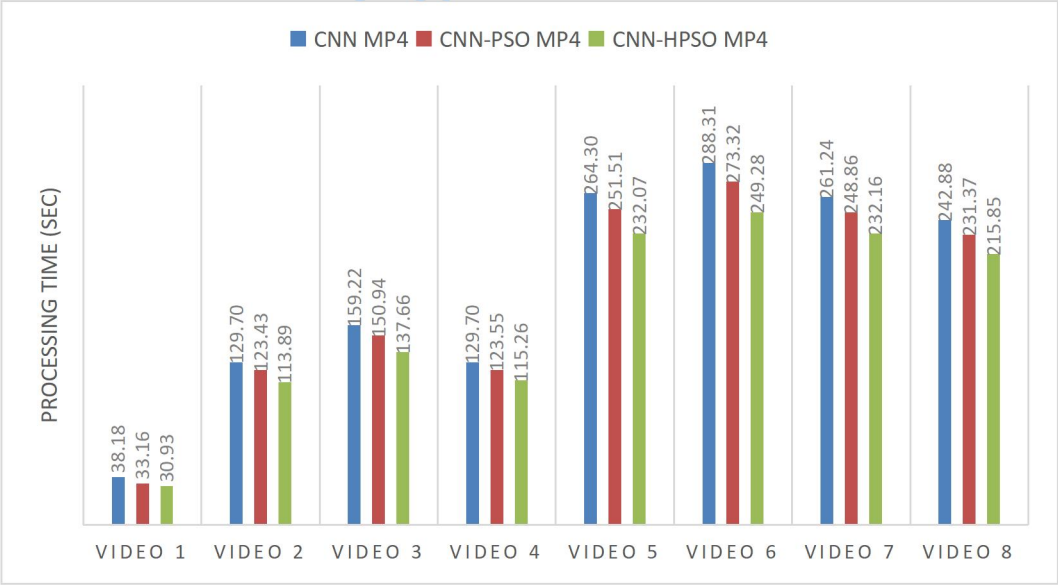


Figure 4.1(a): Graph of Processing Time against each technique with Mp4
(Researcher, Kareem A.E. 2023)

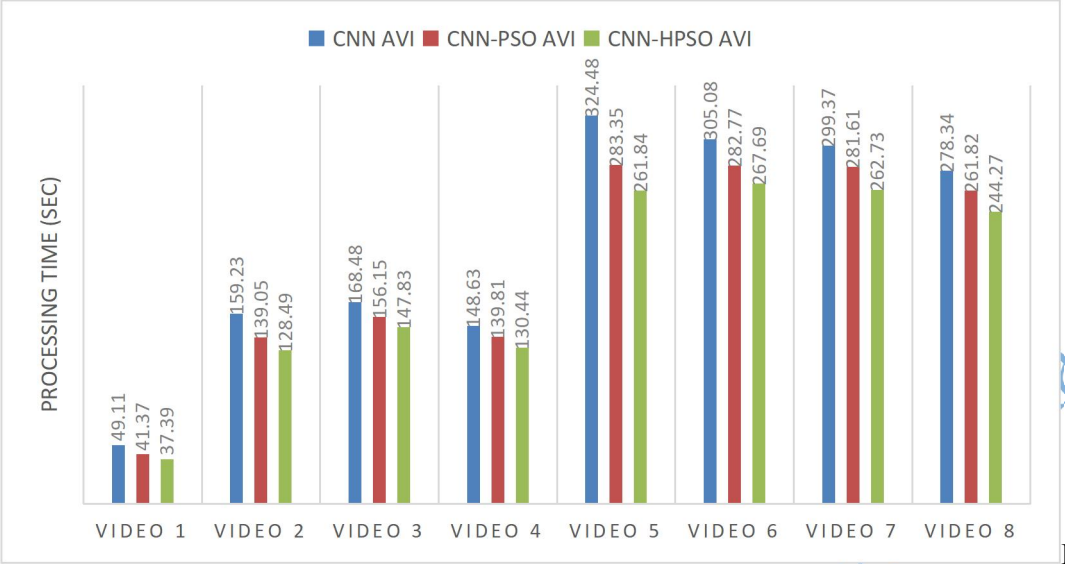


Figure 4.1(b): Graph of Processing Time against each technique with Avi (Researcher, Kareem A.E. 2023)

The accuracy attained by CNN-HPSO, CNN-PSO, and CNN techniques in relation to the video formats is similarly shown in Figure 4.2. In comparison to CNN-PSO and CNN techniques for the following videos: Video1.mp4, Video2.mp4, Video3.mp4, Video4.mp4, Video5.mp4, Video6.mp4, Video7.mp4, Video8.mp4, Video1.avi, Video2.avi, Video3.avi, and Video4.avi, Video5.avi, Video6.avi, Video7.avi and Video8.avi respectively, the CNN-HPSO technique provided the highest (best) accuracy. The use of PSO and HPSO techniques, which can reduce noise and preserve edge information for simple tracking using CNN-PSO and CNN-HPSO methods, is what has led to an improvement in accuracy. The suggested solution in this research ensures accurate detection of moving objects[1, 3].

Figure 4.3 shows the precision for the video format for CNN-HPSO, CNN-PSO, and CNN methods. In comparison to CNN-PSO and CNN techniques for all the video formats, the CNN-HPSO technique provided a slight rise in precision. The outcome demonstrates that the CNN-PSO method performs reasonably well even when PSO technique is used.

Figure 4.4 shows the FPR for the video format for CNN-HPSO, CNN-PSO, and CNN methods. For all the video formats, the CNN-HPSO method outperformed the CNN-PSO and CNN methods in terms of FPR. The outcome showed that using the CNN-HPSO method enhanced the FPR[2].

Figure 4.5 also shows the sensitivity to the video formats attained by CNN-HPSO, CNN-PSO, and CNN methods. In comparison to CNN-PSO and CNN techniques, CNN-HPSO provided the best sensitivity for all the video formats.

Figure 4.6 shows the specificity for the video format for CNN-HPSO, CNN-PSO, and CNN methods. For all the video formats, the CNN-HPSO method outperformed the CNN-PSO and CNN methods in terms of specificity. The outcome showed that using the CNN-HPSO method enhanced the specificity of moving objects.

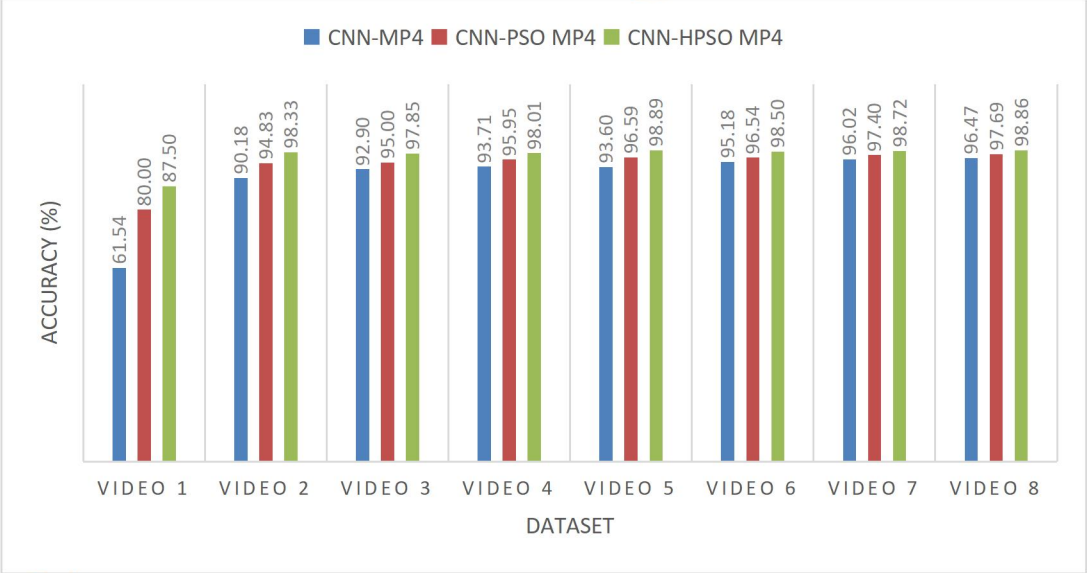


Figure 4.2(a): Graph showing the Accuracy of the Techniques with Mp4
(Researcher, Kareem A.E. 2023)

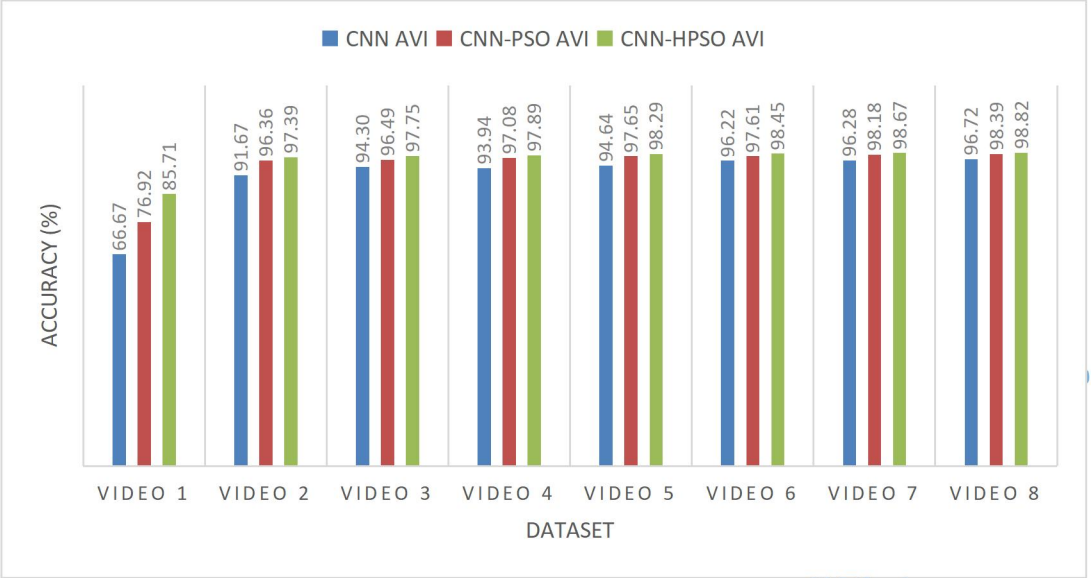


Figure 4.2(b): Graph showing the Accuracy of the Techniques with Avi (Researcher, Kareem A.E. 2023)

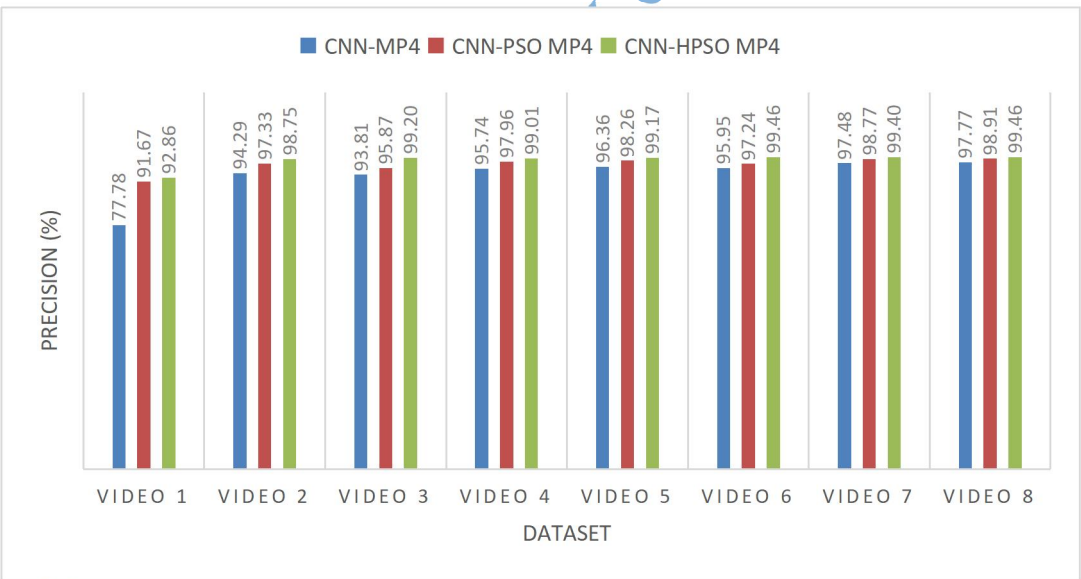


Figure 4.3(a): Graph showing the Precision of the Techniques with Mp4 (Researcher, Kareem A.E. 2023)

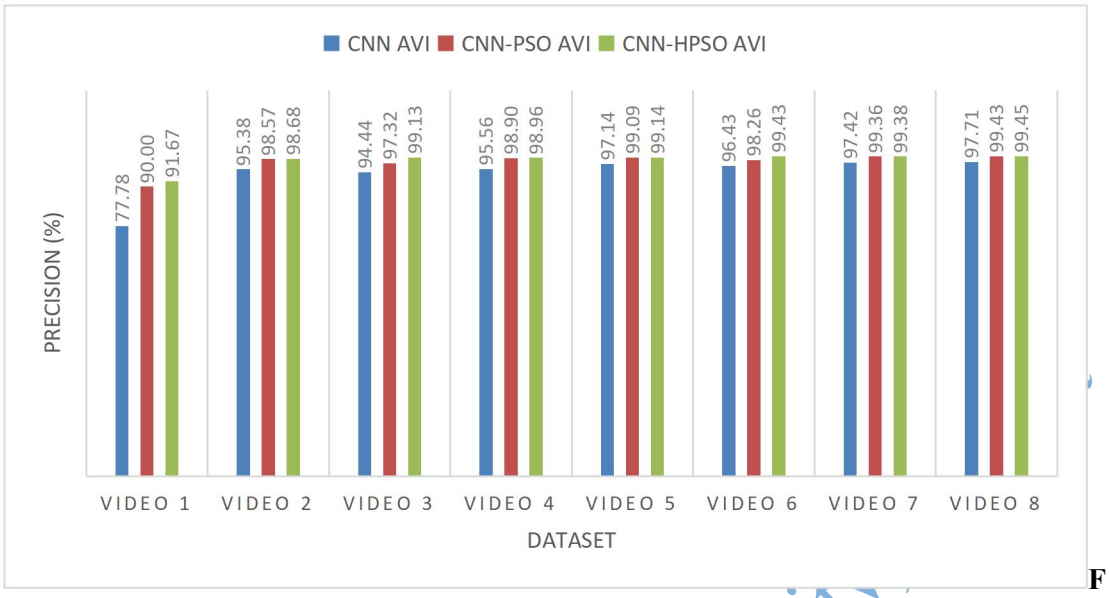


Figure 4.3(b): Graph Showing the Precision of the Techniques with Avi
(Researcher, Kareem A.E. 2023)

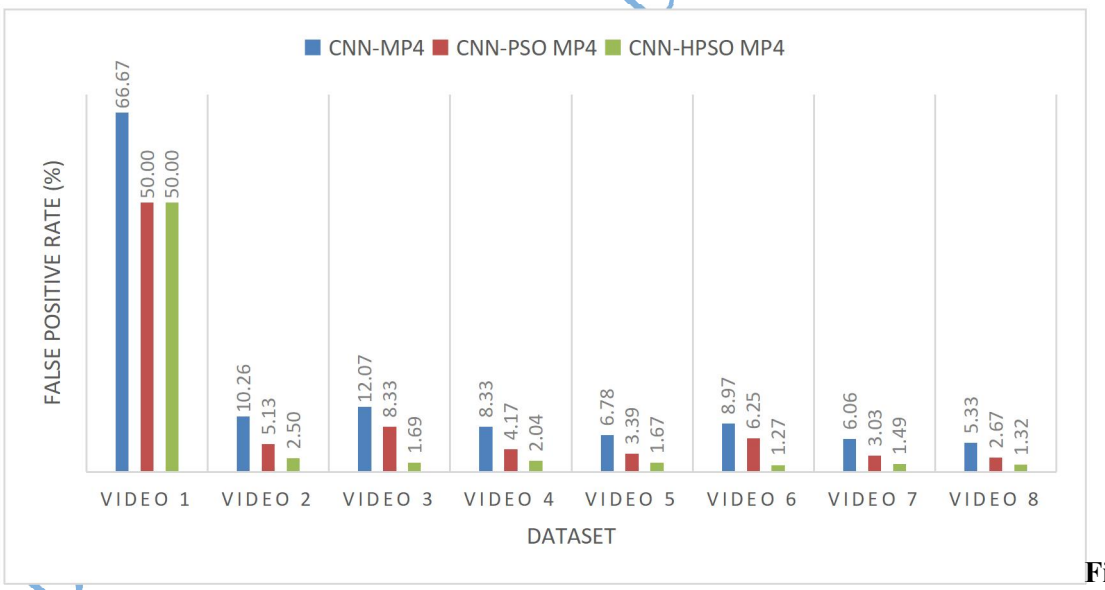


Figure 4.4(a): Graph showing the FPR of the Techniques with MP4
(Researcher, Kareem A.E. 2023)

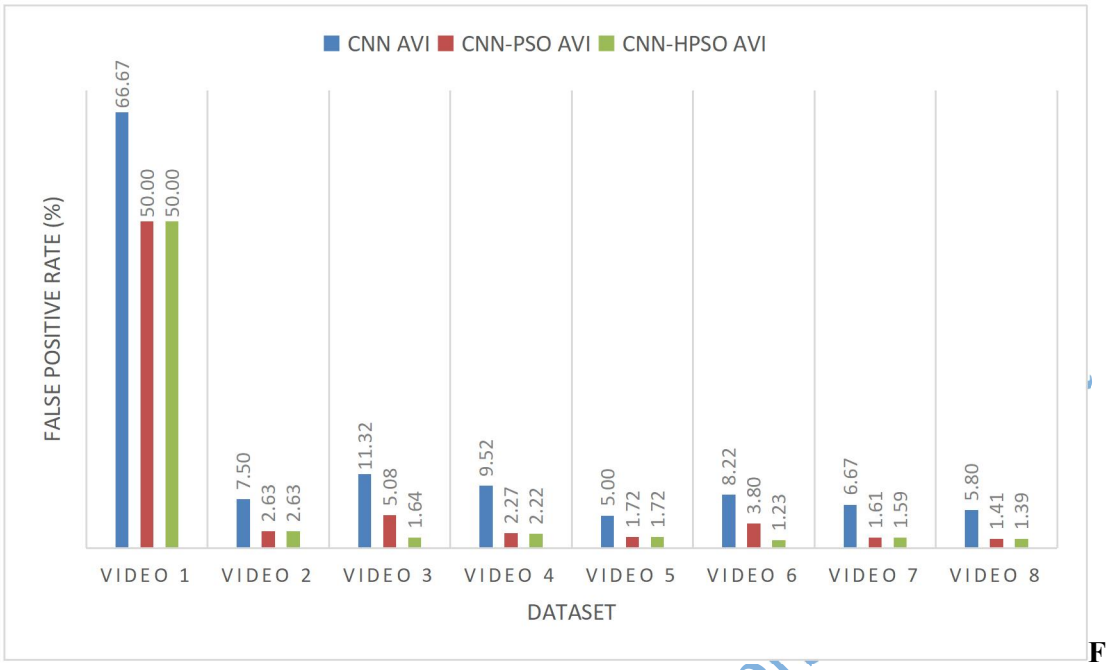


Figure 4.4(b): Graph showing the FPR of the Techniques with AVi
(Researcher, Kareem A.E. 2023)

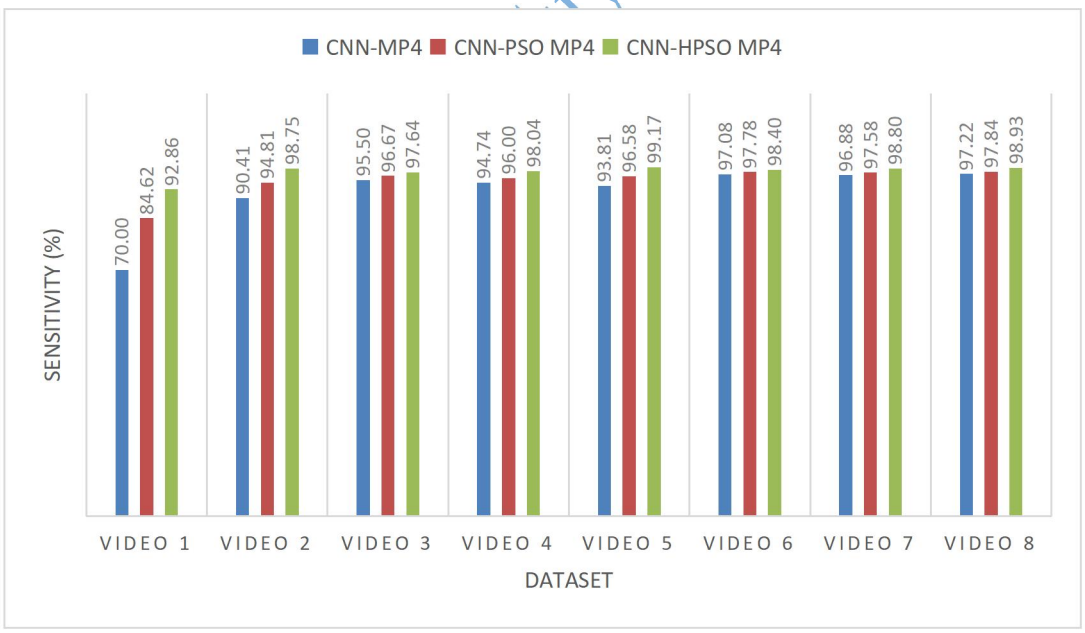


Figure 4.5(a): Graph showing the Sensitivity of the Techniques with Mp4
(Researcher, Kareem A.E. 2023)

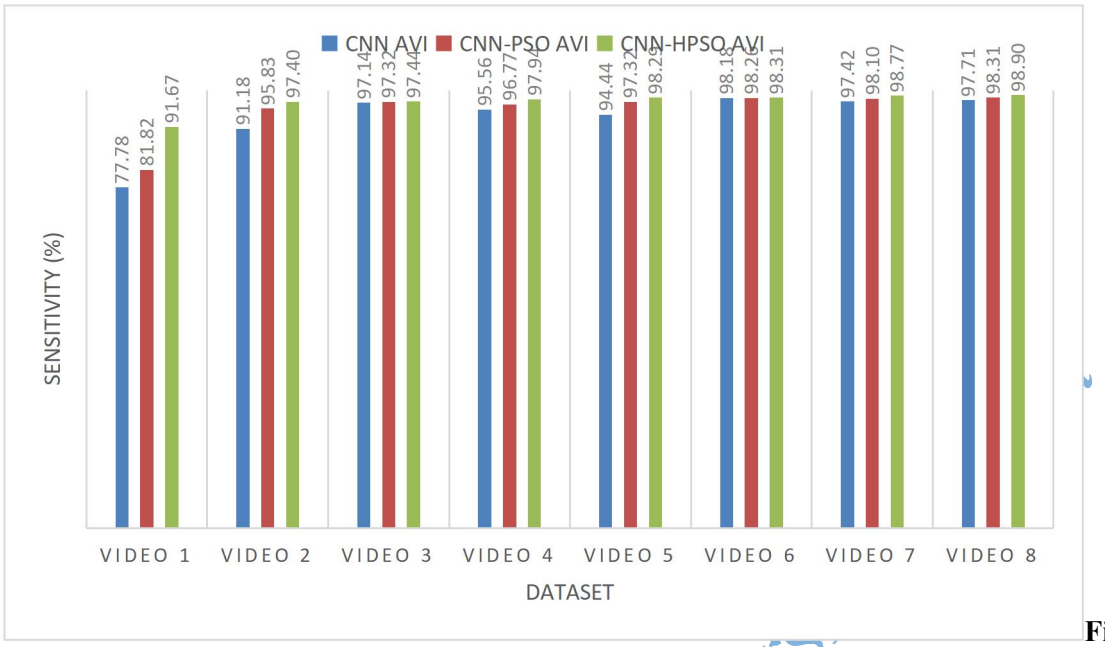


Figure 4.5(b): Graph showing the Sensitivity of the Techniques with Avi
(Researcher, Kareem A.E. 2023)

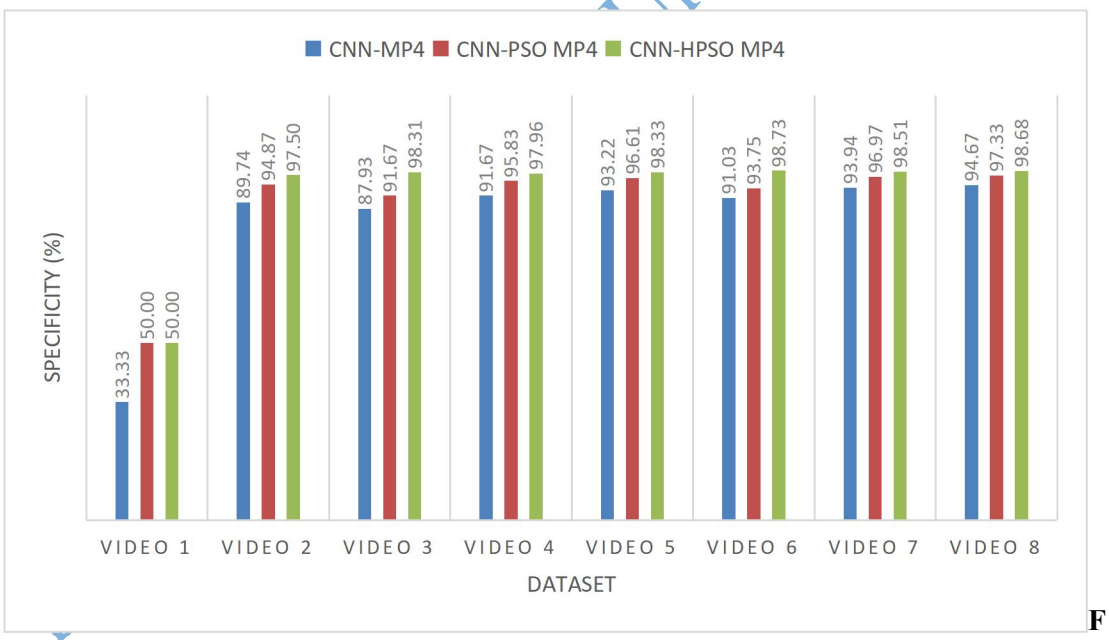


Figure 4.6(a): Graph showing the Specificity of the Techniques with Mp4
(Researcher, Kareem A.E. 2023)

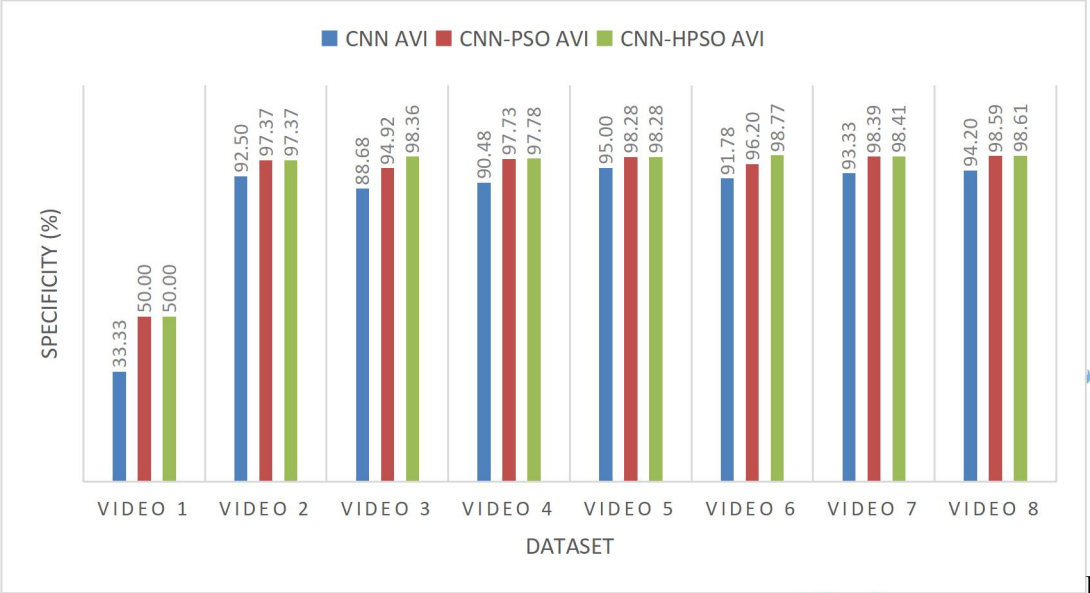


Figure 4.6(b): Graph showing the Specificity of the Techniques with Avi (Researcher, Kareem A.E. 2023)

Table 4.3: Average of Various Techniques for the Eight Videos (Researcher, Kareem A.E. 2023)

Video Formats	**Mp4**			**Avi**		
Techniques	**CNN**	**CNN-PSO**	**CNN-HPSO**	**CNN**	**CNN-PSO**	**CNN-HPSO**
Proc. Time(s)	189.19	179.52	165.89	216.59	198.24	185.09
Accuracy (%)	89.95	94.25	97.08	91.30	94.83	96.62
Precision (%)	93.64	96.99	98.41	93.98	97.62	98.23
FPR (%)	15.56	10.37	7.75	15.09	8.56	7.80
Sensitivity(%)	91.95	95.23	97.82	93.67	95.46	97.34

Specificity(%)	84.44	89.62	92.25	84.91	91.43	92.19

Table 4.3 above shows the average of CNN-HPSO, CNN-PSO and CNN in MP4 and AVi formats for the eight videos. The average results of CNN-HPSO, CNN-PSO and CNN on the videos with MP4 format yielded processing time, accuracy, precision, FPR, sensitivity and specificity of 165.89s, 97.08%, 98.41%, 7.75%, 97.82%, and 92.25%; 179.52s, 94.25%, 96.99%, 10.37%, 95.23% and 89.62%; and 189.19s, 89.95%, 93.64%, 15.56%, 91.95.33% and 84.44% respectively. For the videos in AVi format, CNN-HPSO, CNN-PSO and CNN produced similar average results with processing time, accuracy, precision, FPR, sensitivity, and specificity of 185.09s, 96.62%, 98.23%, 7.80%, 97.34% and 92.19%; 198.24s, 94.83%, 97.62%, 8.56%, 95.46% and 91.43%; and 216.59s, 91.30%, 93.98%, 15.09%, 93.67% and 84.91% respectively.

According to the findings of this research, the CNN-HPSO technique outperforms the CNN-PSO and CNN techniques in terms of accuracy, precision, FPR, specificity and processing time.

Endnotes

[1]Y. Chen,& Q. Wu. "*Moving Vehicle Detection Based on Optical Flow Estimation of Edge*". **In 2015 11th International Conference on Natural Computation** (ICNC), 2015. 754-758.

[2]A. Mahabalagiri, K. Ozcan& S. Velipasalar. "*A Robust Edge-Based Optical Flow Method for Elderly Activity Classification with Wearable Smart Cameras*". **In 2013 Seventh International Conference on Distributed Smart Cameras** (ICDSC), 2013. 1-6.

[3]J. Revaud, P. Weinzaepfel, Z. Harchaoui& S. C. Epicflow. "*Edge-Preserving Interpolation of Correspondences for Optical Flow*". **In Proceedings of**

the IEEE Conference on Computer Vision and Pattern Recognition. 2015. 1164-1172.

[4]T. Gloe, A. Fischer&M. Kirchner. "*Forensic Analysis of Video File Formats*". **Digital Investigation,** 11, 2014. 68-76.

[5]A. L. S. Orozco, C. Q. Huamán, D. P. Álvarez&L. J. G. Villalba. "*A Machine Learning Forensics Technique to Detect Post-Processing in Digital Videos*". **Future Generation Computer Systems**. 2020. 199-212.

Chapter Five

Conclusion

5.1 Summary of Results

The CNN-HPSO technique was used in this study to detect and track moving objects. Videos obtained from locally and online sources were used to assess the effectiveness of the developed technique. The findings show that the developed

technique outperformed the CNN-PSO and CNN technique in terms of accuracy, precision, FPR, sensitivity, specificity and processing time. The findings of the statistical analysis confirmed that there is a significant difference in how they performed.

The evaluation results demonstrated that in terms of object detection and tracking accuracy, the CNN-HPSO outperforms the CNN-PSO and CNN techniques. Precise detection and tracking were made feasible by the developed method's reasonable processing time. The outcomes of this research thus lend credence to the assertion that the HPSO and PSO methods, when combined with CNN technology, enhanced the detection and tracking of moving objects while also accelerating processing. With the developed technique, moving object detection and tracking in surveillance systems could be greatly improved.

5.2 Recommendations

In terms of the efficacy of the method, CNN-HPSO can be used to detect and track moving objects in order to build an accurate and computationally effective intelligent visual surveillance system that can aid human operators in spotting unusual events in the video sequence and responding to them quickly. Also, it could be adopted for various applications in computer vision such as video surveillance, video compression, vision-based control, human-computer interfaces, robotics, medical imaging, and augmented reality.

5.3 Contribution to Knowledge

The research has contributed to the body of knowledge in the following ways:

i. The study has produced a hybrid model (CNN-HPSO) which can be used for various applications in Computer Vision and other related Artificial Intelligence applications.

ii. The CNN-HPSO developed is suitable for detecting and tracking fast moving objects and partial occluded objects.

iii. The CNN-HPSO model for object detection and tracking developed in this work is featured with high localization, low processing time, high precision, high sensitivity, high specificity and low false positive rate.

5.4 Suggestion for Further Studies

The following are recommended for additional research in light of the findings of this work.

i. More work in the aspect of video preprocessing with the developed technique should be carried out with the aim of eliminating the shadow of moving objects. This will help to ascertain if an improvement upon the results obtained can be achieved.

ii. Upcoming work such as reduction or mitigation of noise from videos should be looked into and possibly addressed.

iii. Also, the problems of detecting and tracking multiple objects should be addressed.

Bibliography

Conference Papers

Agarwal, P., Sanaj, S. K., Nikhil, B., Pritam, D., Sanjay, A. & Ayush, G. "*Abandoned Object Detection and Tracking Using CCTVCamera*". In Information and communication technology for sustainable development. Springer, Singapore, 2018. 483-492

Chen, Y., & Wu, Q. "*Moving Vehicle Detection Based on Optical Flow Estimation of Edge*". In 2015 11th International Conference on Natural Computation (ICNC), 2015: 754-758.

Croitoru, I., Simion-Vlad, B., & Marius, L. "*Unsupervised Learning from Video to Detect Foreground Objects in Single Images*." In Proceedings of the IEEE International Conference on Computer Vision. 2017. 4335-4343.

Chan, F. H., Chen, Y. T., Xiang, Y& Sun, M. "Anticipating Accidents in Dashcam Videos," In Proceedings of Asian Conference on Computer Vision (ACCV), 2016, pp. 136–153.

Fusini, L., Tor A. J., & Thor I. F. "*Dead Reckoning of a Fixed-Wing UAV with Inertial Navigation Aided by Optical Flow*." International Conference on Unmanned Aircraft Systems-ICUAS,IEEE 2017. 1250-1259.

Girshick, R., Jeff, D., Trevor D. & Jitendra M. "*Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation*". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014. 580-587.

Gu, J., Han H., Liwei W., Yichen W. & Jifeng D. "*Learning Region Features for Object Detection*." In Proceedings of the European Conference on Computer Vision (ECCV), 2018, 381-395.

Huang, J., Vivek R., Chen S., Menglong Z., Anoop K., Alireza F. & Ian F. "*Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors*." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. 7310-7311.

Jati, W. & Muslim L. K. "*Optimization of Decision Tree Algorithm in Text Classification of Job Applicants Using Particle Swarm Optimization*". In 2020 3rd International Conference on Information and Communications Technology (ICOIACT).2020. 201-205.

Kandagatla, R. K., Sreenivasa R. P. S, Akhila K., Rama R. KSS, Pavan, V. & Amulya, G. "*Object Detection Mechanism using Deep CNN Model*." In 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), IEEE, 2023, 1354-1362.

Lateef, F.; Kas, M. & Ruichek, Y. "*Temporal Semantics Auto-Encoding Based Moving Objects Detection in Urban Driving Scenario*". In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Nagoya, Japan, 11–17 July 2021; 1352–1358.

Li, P.& Jieyu J. "*Time3D: End-to-End Joint Monocular 3D Object Detection and Tracking for Autonomous Driving.*" In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022, 3885-3894.

Lin, T. –Yi., Piotr, D., Ross G., Kaiming H., Bharath H. & Serge B. "*Feature Pyramid Networks for Object Detection*." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, 2117-2125.

Liu, Y., Ruiping W., Shiguang S. & Xilin, C. "*Structure Inference Net: Object Detection Using Scene-Level Context and Instance-Level Relationships*".In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, 6985-6994.

Mahabalagiri, A., Ozcan, K.& Velipasalar, S. "*A Robust Edge-Based Optical Flow Method for Elderly Activity Classification with Wearable Smart Cameras*". In 2013 Seventh International Conference on Distributed Smart Cameras (ICDSC), 2013: 1-6.

Revaud, J., Weinzaepfel, P., Harchaoui, Z., & Schmid, C. E. "*Edge-Preserving Interpolation of Correspondences for Optical Flow"*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015: 1164-1172.

Roy, S.D. & Bhowmik, M.K. "*A Comprehensive Survey on Computer Vision Based Approaches for Moving Object Detection*". In Proceedings of the IEEE Region 10 Symposium (TENSYMP), Dhaka, Bangladesh, 5–7 June 2020; 1531–1534.

Singh, U., Anjali S., & Ravi D. "*Object Tracking in Videos Using CNN*." In ICDSMLA 2019, Springer, Singapore, 2020, 520-527.

Sudharshan, D. P., & Swathi R. "*Object Recognition in Images Using Convolutional Neural Network*." In 2018 2nd International Conference on Inventive Systems and Control (ICISC), IEEE. 2018. 718-722.

Wu, S., Oreifej, O., & Shah, M. "*Action Recognition in Videos Acquired by a Moving Camera Using Motion Decomposition of Lagrangian Particle Trajectories*". In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Barcelona,Spain, 6-13 November 2011; 1419-1426.

Xu, H., XutaoL., Xiaoyu, W., Zhou R., Navaneeth B., & Rama C. "*Deep Regionlets for Object Detection*". In Proceedings of the European Conference on Computer Vision (ECCV), 2018. 798-814.

Zaheer, M., Mahmood, A., Khan, M., Segu, M., Yu, F.& Lee, S. "*Generative cooperative learning for unsupervised video anomaly detection*," InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 14744-14754

Journal Papers

Abdulghafoor, N. H., & Hadeel N. A. "*A Novel Real-Time Multiple Objects Detection and Tracking Framework for Different Challenges*." **Alexandria Engineering Journal.** 61, no. 12, 2022, 9637-9647.

Ahmad, M., Imran, A., Fakhri A. K., Fawad Q., & Hanan A. "*Convolutional Neural Network-Based Person Tracking Using Overhead Views*." **International Journal of Distributed Sensor Networks.** 16, no. 6, 2020, 12-20.

Ahmed, G. G. "*Particle Swarm Optimization Algorithm and Its Applications: A SystematicReview*". **Archives of Computational Methods in Engineering.** 29, 2022. 2531-2561

Ajoy M. "*Camouflaged Object Detection and Tracking: A Survey*". **International Journal of Image and Graphics**. Vol. 20, No. 04, 2020.2050028https://doi.org/10.1142/S021946782050028X

Ala, M., Thierry C., Najoua E.& Ben A. "*Spatio-Temporal Object Detection by Deep Learning: Video-Interlacing to Improve Multi-Object Tracking*". **Image and Vision Computing**. Vol. 88, 2019, 120-131.

Alper, Y. O. & Mubarak S. "*Object Tracking: A Survey*." **ACMComputing Surveys (CSUR).** no.38, 4. 2006, 1-8.

Angulu, R., Jules R. T. & Aderemi O. A. "*Age Estimation via Face Images: ASurvey*." **EURASIP Journal on Image and Video Processing**. no. 1, 2018, 1-35.

Ary, M. S., Edo D. Y. &Dini A. N. **"***Resource-Aware Video Streaming (RAViS) Framework for Object Detection System Using Deep Learning Algorithm*"**.** **MethodsX.** 11, 2023, 102285

Avci, O., Abdeljaber, O., Kiranyaz, S., Hussein,M., Gabbouj, M. & Inman, D. "*A Review of Vibration-Based Damage Detection in Civil Structures: From Traditional Methods to Machine Learning and Deep Learning Applications".* **Mechanical Systems and Signal Processing**, 147. 2021, 107077, 10.1016/j.ymssp.2020.107077

Baimukashev, D., Alikhan Z., Askat K., Artemiy O., Denis F., ZhanatM. & Huseyin A. V. "*Deep Learning Based Object Recognition Using Physically-Realistic Synthetic Depth Scenes*". **Machine Learning and Knowledge Extraction.** 1, no. 3, 2019, 883-903.

Bala, A. & Rishabh K. "*Jaywalking Detection and Localization in Street Scene Videos Using Fine-Tuned Convolutional Neural Networks*." **Multimedia Tools and Applications**. 2023, 1-21.

Bharti M., Abdul R. A., Sikandar A., Meltem D. B., Federico T. & Fabio G.*"Joint Detection and Tracking in Videos with Identification Features".***Image and Vision Computing**. 100, 2020, 103932

Bochkovskiy, A.; Wang, C.Y. & Liao, H. "*YOLOv4: Optimal Speed and Accuracy of Object Detection*". **arXiv** 2020, arXiv:2004.10934.

Bo W., Jinghong L., Shengjie Z.&Fang X. "*A Dual-Input Moving Object Detection Method in Remote Sensing Image Sequences via Temporal Semantics*". **Remote Sensing**. 15(9), 2023. 2230. DOI:10.3390/rs15092230

Cao, D., Zhixin C. & Lei G. "*An Improved Object Detection Algorithm Based on Multi-Scaled and Deformable Convolutional Neural Networks*." **Human-centric Computing and Information Sciences.** 10, no. 1, 2020, 1-22.

Chandrakala, S., Deepak, K. &Revathy, G. "*Anomaly Detection in Surveillance Videos: A Thematic Taxonomy of Deep Models, Review and Performance Analysis*".**Artificial Intelligence Review**, 2, 3, 2022, 1-50

Chapel, M.N. & Bouwmans, T. "*Moving Objects Detection with aMoving Camera: A Comprehensive Review*". **Computer Science Review**. 38, 2020. 100310.

Chen, H. –Y., ChuenH. L., JyunW. L. & YungK. C. "*Convolutional Neural Network-Based Automated System for Dog Tracking and Emotion Recognition in Video Surveillance*." **Applied Sciences.** 13, no. 7, 2023, 4596.

Daniel C., Víctor M. B.&Manuel M. **"***Short-Term Anchor Linking and Long-Term Self-Guided Attention for Video Object Detection***.Image and Vision Computing.** Vol. 110, 2021, 104179

Dave, S. A., Nagmode, M. & Aditi J. "*Statistical Survey on Object Detection and Tracking Methodologies*". **International Journal of Scientific & Engineering Research** 4, no. 3 2013. 1-6.

Elizabeth I. Y., Dian T., Golam S. &Alireza A.**"***Scalability of Knowledge Distillation in Incremental Deep Learning for Fast Object Detection***". Applied Soft Computing.** 129, 2022, 109608

El-Shair, Z. & Samir A. R. "*High-Temporal-Resolution Object Detection and Tracking Using Images and Events.*" **Journal of Imaging**8, no. 8, 2022, 210.

Fan Y., Shigeyuki O., Sosuke Y., Hiroaki F., Shoichi M.&Shan J. "*A Unified Multi-View Multi-Person Tracking Framework"*. **Computational Visual Media**, 2023. 03820

Fan Y., Xin C., Sakriani S., Yang W.& Satoshi N. *"ReMOT: A Model-Agnostic Refinement for Multiple Object Tracking"*.**Image and Vision Computing**. 106, 2021, 104091

Fan Y., Zheng W., Yang W., Sakriani S.& Satoshi N. "*Tackling Multiple Object Tracking with Complicated Motions-Re-Designing the Integration of Motion and Appearance"***. Image and Vision Computing.**Vol. 124, 2022, 104514

Francisco P. H., Siham T., Alberto L., Roberto O., Hamido F.&Francisco H. *"Object Detection Binary Classifiers Methodology Based on Deep Learning to Identify Small Objects Handled Similarly: Application in Video Surveillance"*. **Knowledge-Based Systems.** Vol. 194. 2020. 105590

Gangyi Tian, Jianran Liu and Wenyuan Yang. "*A dual neural network for object detection in UAV images*". **Neurocomputing**. Vol. 443, 2021, 292-301

García-Aguilar, I., Jorge G., Rafael M. L. B. & Ezequiel L. R. "*Automated Labeling of Training Data for Improved Object Detection in Traffic Videos by Fine-Tuned Deep Convolutional Neural Networks*." **Pattern Recognition Letters.** 167. 2023, 45-52.

Giwan L., Phayuth Y., Doyeob Y. &Ayoung H. **"***Enhancing Detection Performance for Robotic Harvesting Systems Through RandAugment***". Engineering Applications of Artificial Intelligence**. Vol.123,Part C, 2023, 106445

Gloe, T., Fischer, A. & Kirchner, M. "*Forensic Analysis of Video File Formats*". **Digital Investigation,** 11, 2014: 68-76.

Guanbo W., Hongwei D., Mingliang D., Yuanyuan P., Zhijun Y.& Haiyan L. "*Fighting Against Terrorism: A Real-Time CCTV Autonomous Weapons Detection Based on Improved YOLO v4*". **Digital Signal Processing.** 132. 2023. 103790

Huang, L.; Luo, R.; Liu, X. & Hao, X. "*Spectral Imaging with Deep Learning"*. **Light Sci. Appl.** 11, 2022, 61.

Jaskirat K.&Williamjeet S. "*Tools, Techniques, Datasets and Application Areas for Object Detection in an Image: A Review*". **Multimedia Tools and Applications**. volume 81, 2022.38297-38351

Jeevith S. H.&Lakshmikanth S. "*Detection and Tracking of Moving Object Using Modified Background Subtraction and Kalman Filter*". **International Journal of Electrical and Computer Engineering (IJECE)**. 11(1): 2021. 217. DOI:10.11591/ijece.v11i1.pp217-223

Jemilda, G., S. Baulkani, D. George Paul, & Benjamin R. J. "*Tracking Moving Objects in Video*." **JCP** 12, no. 3 2017, 221-229.

Jianchen H., Jun C. & Han W.*"A Light Weight and Efficient One-Stage Detection Framework"*. **Computers and Electrical Engineering.** Vol. 105. 2023. 108520

John, P. B., Sriharsha, R., Dimmita N.& Inumula V. R. "*Convolutional Neural Network Based Object Detection System for Video Surveillance Application***". Concurrency and Computation: Practice and Experience.** Vol. 35, Issue3. 2023.

Junho C., Sangkyoo P., Dongsung P., Hyunduck C.&Myotaeg L. **"***Feature-Selection-Based Attentional-Deconvolution Detector for German Traffic Sign Detection Benchmark"*. **Electronics**. 12, no3, 2023, 725

Jun L., Xiaoyu W., Bo L.& Zhigao Z. "*A Survey on Firefly Algorithms*". **Neurocomputing**. 500, 2022, 662-678

Kang T., Yiquan W. & Fei Z. **"***Recent Advances in Small Object Detection Based on Deep Learning: A Review***". Image and Vision Computing**. 97, 2020, 103910

Khan, A., Sohail, A., Zahoora, U.& Qureshi, A. S. "*A survey of the Recent Architectures of Deep Convolutional Neural Networks*," **Artificial Intelligence Review**, vol. 53, no. 8, 2020, 5455-5516

Khan S. D&Ullah, H. "*A Survey of Advances in Vision-Based Vehicle Re-Identification*," **Computer Vision and Image Understanding**, vol. 182, 2019, 50-63

Kim, B. & Neville, C. "*Accuracy and Feasibility of a Novel Fine Hand Motor Skill Assessment Using Computer Vision Object Tracking*. **Scientific Reports**. 13. 2023, 1813.

Kiranyaz, S., Ince, T., Iosifidis, A. & Gabbouj, M. "*Operational Neural Networks*", **Neural Computing and Applications (Springer-Nature)**. 2020, 1-24, 10.1007/s00521-020-04780-3

Kouka, N., Fatma B., Raja F., Rahma F., Amir H. & Adel M. A. "*A Novel Approach of Many-Objective Particle Swarm Optimization with Cooperative Agents Based on an Inverted Generational Distance Indicator*". **Information Sciences.** 623, 2023. 220-241.

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M. & Nie, W. "*YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications"*. **arXiv** 2022, arXiv:abs/2209.02976.

Li, D., Mo, B. & Zhou, J. "*Boost Infrared Moving Aircraft Detection Performance by Using Fast Homography Estimation and Dual Input Object Detection Network*". **Infrared Phys. Technol**. 123, 2022, 104182.

Li, K., Gang W., Gong C., Liqiu M. & Junwei H. "*Object Detection in Optical Remote Sensing Images: A Survey and ANew Benchmark*." **ISPRS Journal of Photogrammetry and Remote Sensing.** 159, 2020, 296-307.

Li, K.; Wan, G.; Cheng, G.; Meng, L. & Han, J. "*Object Detection in Optical Remote Sensing Images: A survey and a New Benchmark*. **Remote Sens**.159, 2020. 296–307.

Li L.,Wanli O., Xiaogang W., Paul F., Jie C., Xinwang L.&Matti P. "*Deep Learning for Generic Object Detection: A Survey".***International Journal of Computer Vision**.Vol. 128, 2019, 261-318

Li, Q., Shaopeng H., Kohei S. & Idaku I. "*An Active Multi-Object Ultrafast Tracking System with CNN-Based Hybrid Object Detection*". **Sensors** 23, no. 8, 2023. 41-50.

Liu, L., Wanli O., Xiaogang W., Paul F., Jie C., Xinwang L. &Matti P. "*Deep Learning for Generic Object Detection: A Survey*." **International Journal of Computer Vision.** 128, no. 2, 2020. 261-318

Liu, Y., Wang, Y., Wang, S., Liang, T., Zhao, Q., Tang, Z. & Ling, H. "*CBNet: A Novel Composite Backbone Network Architecture for Object Detection*". **arXiv**2019, arXiv:1909.03625.

Loussaief S & Abdelkrim, A. "*Convolutional Neural Network Hyper-Parameters Optimization Based on Genetic Algorithms*," **International Journal of Advanced Computer Science and Applications**, vol. 9, no. 10, 2018, 252-266

Lyu, Y., Michael Y., Yang, G. V. & Gui-Song X. "*Video Object Detection with a Convolutional Regression Tracker*." **ISPRS Journal of Photogrammetry and Remote Sensing.** 176. 2021, 139-150.

Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G.& Johnson, B.A. "*Deep Learning in Remote Sensing Applications: A Meta-Analysis and Review.* **Remote Sens**.2019, 152, 2019, 166-177.

Manisha K., Baljit S. K. & Akashdeep S. "*Soft Computing Based Object Detection and Tracking Approaches: State-of-the-Art Survey*".**Applied Soft Computing**. 70, 2018, 423-464

Marie-Neige C.&Thierry B. "*Moving Objects Detection with a Moving Camera: A Comprehensive Review*". **Computer Science Review**. 38 No 3, 2020, 100310

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N. & Terzopoulos, D. "*Image Segmentation Using Deep Learning: A Survey*". **arXiv**2020, arXiv:2001.05566.

Minaeian, S., Liu, J. & Son, Y.J. "*Effective and Efficient Detection of Moving Targets from a UAV's Camera*". **IEEE Trans. on Intell. Transp. Syst.** 19, 2018, 497-506.

Mingshuo X., Hongxin W., Hao C., Haiyang L. &Jigen P. "*A Fractional-Order Visual Neural Model for Small Target Motion Detection*". **Neurocomputing**. Vol. 550, 2023, 126459

Nguyen, K., Fookes, C., Sridharan, S., Tian, Y., Liu, F., Liu, X &Ross, A. "*The State of Aerial Surveillance: A Survey*," **arXiv preprint** arXiv:2201.03080, 2022. 3

Niam, A. A., Omar, S. Q. & Zakariya, Y. A. "*A New Hybrid Firefly Algorithm and Particle Swarm Optimization for Tuning Parameter Estimation in Penalized Support Vector Machine with Application in Chemometrics*". **Chemometrics and Intelligent Laboratory Systems**. Vol. 184, 2019, 142-152

Norhalina S., Fazli W., Muhammad A., Ali S. &Mukhtaj K. "*Comparative Analysis of Recent Architecture of Convolutional Neural Network*". **Mathematical Problems in Engineering**. 2022, 7313612

Orozco, A. L. S., Huamán, C. Q., Álvarez, D. P. & Villalba, L. J. G. "*A Machine Learning Forensics Technique to Detect Post-Processing in Digital Videos*". **Future Generation Computer Systems**. 2020:199-212.

O'Shea, K. & Ryan N. "*An Introduction to Convolutional Neural Networks.*" **arXiv preprint arXiv***:1511.08458* 2015, 1-5.

Pang, G., Shen, C., Cao, L. & Hengel, A.V.D. "*Deep Learning for Anomaly Detection: A Review*," **ACM Computing Surveys (CSUR)**, vol. 54, no. 2, 2021 1-38

Pareek, P & Thakkar, A. "*A Survey on Video-Based Human Action Recognition: Recent Updates, Datasets, Challenges, and Applications*", **Artificial Intelligence Review**, vol. 54, no. 3, 2021, 2259-2322

Prajapati, D., & Hiren J. G. "*A Review on Moving Object Detection and Tracking*".**International Journal of Computer Application** 5, no. 3 2015. 168-175

Rahmaniar, W., Wang, W.-J. & Chen, H.-C. "*Real-Time Detection and Recognition of Multiple Moving Objects for Aerial Surveillance*". **Electronics.** Vol 8. 2019, 1373.

Ramachandran A. & Dhamodaran M."*Multi-Object Detection and Tracking Using Reptile Search Optimization Algorithm with Deep Learning*". ***Symmetry.*** *15, 6*, 2023. 1194.

Ramachandra, B., Jones, M. J. & Vatsavai, R. R. "*A Survey of Single-Scene Video Anomaly Detection,*" **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 44, no. 5, 2022. 2293-2312

Ren, Y., Changren Z., & Shunping X. "*Small Object Detection in Optical Remote Sensing Images via Modified Faster R-CNN.*" **Applied Sciences.** 8, no. 5 2018, 813.

Rudrika K.&Sakshi A. "*Performance Analysis of U-Net with Hybrid Loss for Foreground Detection*".**Multimedia Systems**. 2022.

Rudrika K. & Sashi A. **"***Background Subtraction for Moving Object Detection: Explorations of Recent Developments and Challenges***". The Visual Computer.** 38, 2022, 4151-4178 https://doi.org/10.1007/s00371-021-02286-0

Rui S., Tao L., Qi C., Zexuan W., Xiaogang D., Weiqiang Z. & Asoke K. N. "*Survey of Image Edge Detection*". **Horizons in Signal Processing.**2. 2022. 826967

Runyu J., Yi W., Fabio P. & Yiming W. **"***Survey on Video Anomaly Detection in Dynamic Scenes with Moving Cameras"***. Arxiv 2023. 07050.** https://doi.org/10.48550/arxiv.2308.07050

Saeed M. A. J., Asma P. & Ahmed N. A. *"Object Tracking and Detection Techniques Under GANN Threats: A Systemic Review"*. **Applied Soft Computing.** 139, 2023, 110224

Senan, N., Aamir, M., Ibrahim, R., Taujuddin, N & Muda, W. W "*An Efficient Convolutional Neural Network for Paddy Leaf Disease and Pest Classification*," **International Journal of Advanced Computer Science and Applications**, vol. 11, 2020.

Sengupta, S., Sanchita B. & Richard A. P. "*Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives*". **Machine Learning and Knowledge Extraction** 1, no.1 2019. 157-191.

Serkan K., Onur A., Osama A., Turker I., Moncef G. &Daniel J. I.*"1D Convolutional Neural Networks and Applications: A Survey".***Mechanical Systems and Signal Processing**. Volume 151, 2021, 107398

Seyed A. D. & Nasrollah M. C. "*Biogeography Based Optimization Method for Robust Visual Object Tracking*". **Applied Soft Computing**. 122, 2022, 108802

Shafie, A. A., Fadhlan H. & Ali, M. H. "*Motion Detection Techniques Using Optical Flow*." **World Academy of Science, Engineering and Technology** 56 2009: 559-561.

Shafin R., Salman H. K.&Fatih P. "*Zero-Shot Object Detection: Joint Recognition and Localization of Novel Concepts*". **International Journal of Computer Vision**. Vol 128, 2020, 2979-2999

Sivadi B.&Ahmad A. M. "*Progress in Multi-Object Detection Models: AComprehensive Survey*". **Multimedia Tools and Applications.** 82, 2023, 22405-22439

Su, Y., Liu, J., Xu, F., Zhang, X. & Zuo, Y. "*A Novel Anti-Drift Visual Object Tracking Algorithm Based on Sparse Response and Adaptive Spatial-Temporal Context-Aware*". **Remote Sensing**. 13. 2021, 4672.

Supreeth, H.S.G & Chandrashekar M. P. "*Efficient Multiple Moving Object Detection and Tracking Using Combined Background Subtraction and Clustering*". **Signal, Image and Video Processing**. vol.12, no.6, 2018. 1097-1105

Tang, S.& Ye Y. "*Object Detection Based on Convolutional Neural Network*". **IEEE Transactions on Pattern Analysis and Machine Intelligence**. 2015.1-5.

Thenmozhi, T. &Kalpana, A. M. **"***Adaptive Motion Estimation and Sequential Outline Separation Based Moving Object Detection in Video Surveillance System***". Microprocessors and Microsystems**.76, 2020, 103084

Thorsten H.& Claudia K. "*Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends*". **Remote Sensing**. Vol. 12, issue 10, 2020, 1667 https://doi.org/10.3390/rs12101667

Tianyu W., Zhongjing M., Tao Y.& Suli Z. "*PETNet: A YOLO-Based Prior Enhanced Transformer Network for Aerial Image Detection*".**Neurocomputing**. Vol. 547, 2023, 126384

Vishruth, B. G.,Surya, J. B.&Sivateja A. T. "*Improved Background Subtraction with Histogram Equalization and Adaptive Thresholding*". **IJARCCE** 12(6), 2023, 12654.

Warade, P. P., Kale, M. S & Thakare, V. M. "*Edge Detection for Moving Object Tracking*". **International Journal**. 5, no 4,2015. 16-19.

Wu, X., Doyen S. & Steven C. H. "*Recent Advances in Deep Learning for Object Detection*". **Neurocomputing.** 396, 2020, 39-64.

Wu, Y.; He, X. & Nguyen, T.Q. "*Moving Object Detection with a Freely Moving Camera via Background Motion Subtraction*". **IEEE Trans. Circuits Syst. Video Technology**. 27, 2017, 236-248.

Xiao, C.; Yin, Q.; Ying, X.; Li, R.; Wu, S.; Li, M.; Liu, L.; An, W. & Chen, Z. "*DSFNet: Dynamic and Static Fusion Network for Moving Object Detection in Satellite Videos*". **IEEE Geoscience and Remote Sensing Letters**. 19, 2022, 1-5.

Yadav, K. S., Anish M. K., Rabul H. L., & Bhuyan, M. K. "*Detection, Tracking, and Recognition of Isolated Multi-Stroke Gesticulated Characters*." **Pattern Analysis and Applications**. 2023, 1-26.

Yadav, K. S., Anish M. K & Rabul H. L. "*Gesture Objects Detection and Tracking for Virtual Text Entry Keyboard Interface*." **Multimedia Tools and Applications.** 82, no. 4, 2023, 5317-5342.

Yang, J., En Y., Zeming L., Xiaoping L. & Wenbing T. "*Quality Matters: Embracing Quality Clues for Robust 3D Multi-Object Tracking*." **arXiv preprint arXiv.** 2208. 2022.10976

Yang, K., Baoliang P., Fengwei G., Yanhua Z., Shenying W., Zhaoyang Y.& Zhichao H. "*Convolutional Neural Network for Object Detection in Garlic Root Cutting Equipment*." **Foods.** 11, no. 15, 2022, 2197.

Yang L., Peng S., Nickolas W. & Yi S. *"A Survey and Performance Evaluation of Deep Learning Methods for Small Object Detection".***Expert Systems with Applications**. Vol. 172, 2021, 114602

Yang, Y., Jian W. &Xiaofei C. "*Improvements on Particle Swarm Optimization Algorithm for Velocity Calibration in Microseismic Monitoring*." **Earthquake Science**28, no. 4, 2015, 263-273.

Yang, Z., Ziyu, B. & Yexin P. "*Optimization Algorithm of Moving Object Detection Using Multiscale Pyramid Convolutional Neural Networks*." **Computational Intelligence and Neuroscience.** 2023.

Yang Z., Yi C. & Dechang P. "*Discovery of Stay Area in Indoor Trajectories of Moving Objects*". **Expert Systems with Applications.** 2021. 114501

Yazdi, M. & Bouwmans, T. "*New Trends on Moving Object Detection in Video Images Captured by a Moving Camera: A Survey*". **Computer Science Review.** 28, 2018, 157-177.

Yu, Y.; Kurnianggoro, L. & Jo, K.-H. "*Moving Object Detection for a Moving Camera Based on Global Motion Compensation and Adaptive Background Model"*. **Int. J. Control Autom. Syst.** 17, 2019, 1866-1874.

Zhang, B.W., Wang, L.M., Wang, Z., Qiao, Y., & Wang, H.L. "*Real-Time Action Recognition with Deeply Transferred Motion Vector CNNs*". **IEEE Trans. Image Process**. 2018, 27, 2326–2339.

Zhang, J., Chen L., Yimin Y., Jiawei Z.& Marcin G. "*Applications of Artificial Neural Networks in Microorganism Image Analysis: A Comprehensive Review from Conventional Multilayer Perceptron to Popular Convolutional Neural Network and Potential Visual Transformer*". **Artificial Intelligence Review** 56, no 2, 2023. 1013-1070.

Zhang, S., Huang, W.& Zhang, C. "*Three-Channel Convolutional Neural Networks for Vegetable Leaf Disease Recognition*".**Cognitive Systems Research**, vol. 53, 2019, 31-41

Zhang, Y., Lu, H., Zhang, L., Ruan, X & Sakai, S. "*Video Anomaly Detection Based on Locality Sensitive Hashing Filters*," **Pattern Recognition**, vol. 59, 2016 302-311

Zhang, Y., Shuihua W.& Genlin J. "*A Comprehensive Survey on Particle Swarm Optimization Algorithm and its Applications*". **Mathematical problems in engineering.** 2015. 1-20

Zhao, J., Shilong J., Zhihao C., Yiwen Z. & Yingxun W.. "*Moving Object Detection and Tracking by Event Frame from Neuromorphic Vision Sensors*." **Biomimetics.**7, no. 1, 2022, 31.

Zhao, X.; Wang, G.; He, Z. & Jiang, H. "*A Survey of Moving Object Detection Methods: A Practical Perspective*. **Neurocomputing**. 503, 2022, 28-48.

Zhao, Z., Peng Z., Shou-tao X. & Xindong W. "*Object Detection with Deep Learning: A Review*." **IEEE Transactions on Neural Networks and Learning Systems.** 30, no. 11, 2019, 3212-3232.

Zhu, H.; Yan, X.; Tang, H.; Chang, Y.; Li, B. & Yuan, X. "*Moving Object Detection with Deep CNNs*". **IEEE Access**. 8, 2020, 29729-29741.

Zhu, J.; Wang, Z.; Wang, S. & Chen, S. "*Moving Object Detection Based on Background Compensation and Deep Learning*". **Symmetry**. 12, 2020, 1965.

Zuo, C.; Qian, J.; Feng, S.; Yin, W.; Li, Y.; Fan, P.; Han, J.; Qian, K. & Chen, Q. "*Deep Learning in Optical Metrology: A Review*". **Light Sci. Appl**. 11, 2022, 39

Thesis

Pedro M."*Computational Models of Object Motion Detectors Accelerated Using FPGA Technology*". **Nottingham Trent University**. 2021. DOI:10.13140/RG.2.2.18739.81441

Electronic source

Abinibi Hub. "*Ile-Ife, Nigeria:The Yoruba City of Culture and Traditions*". **Abinibi Hub**. 2022. https://www.youtube.com/watch?v=uOT2jcKAtkk. (Available online)

Episode 1. "*See Ibadan From Above*". **HD Drone Footage**. 2021. https://www.youtu.be.ks5CQkzE7kA?si=S8VE-e_vmN729poO

Heternity Media. "*Ogbomoso, Oyo State, Nigeria*". **Heternity Media**. 2020.https://www.youtube.com/watch?v=lyBtCodi55s. (Available online)

Plastonick. "*Motion Object Tracking Sample".* **Plastonick**. 2016. https://www.youtube.com/watch?v=P2toRLWq2Xs. (Available online)

Top. "*A beautiful Aerial View of Oshodi, Lagos, Nigeria*". **Top: The Official Photography**. 2021.https://www.youtu.be/8fGEQf52e-U?si=VEPxQpr924mchuBY(Available online)

Appendix i.

CNN-PSO

```
function out = PSO(problem, params)

    %% Problem Definiton

    CostFunction = problem.CostFunction;  % Cost Function

    nVar = problem.nVar;        % Number of Unknown (Decision) Variables

    VarSize = [1 nVar];         % Matrix Size of Decision Variables

    VarMin = problem.VarMin;    % Lower Bound of Decision Variables
    VarMax = problem.VarMax;    % Upper Bound of Decision Variables

    %% Parameters of PSO

    MaxIt = params.MaxIt;   % Maximum Number of Iterations

    nPop = params.nPop;     % Population Size (Swarm Size)

    w = params.w;           % Intertia Coefficient
    wdamp = params.wdamp;   % Damping Ratio of Inertia Coefficient
    c1 = params.c1;         % Personal Acceleration Coefficient
    c2 = params.c2;         % Social Acceleration Coefficient

    % The Flag for Showing Iteration Information
    ShowIterInfo = params.ShowIterInfo;

    MaxVelocity = 0.2*(VarMax-VarMin);
    MinVelocity = -MaxVelocity;
    %% Initialization
    % The Particle Template
    empty_particle.Position = [];
    empty_particle.Velocity = [];
```

```
empty_particle.Cost = [];
empty_particle.Best.Position = [];
empty_particle.Best.Cost = [];
empty_particle.expvel = [];
empty_particle.featposition=[];
empty_particle.Best.featposition=[];

% Create Population Array
particle = repmat(empty_particle, nPop, 1);

% Initialize Global Best
GlobalBest.Cost = inf;

% Initialize Population Members
for i=1:nPop

    % Generate Random Solution
    particle(i).Position = unifrnd(VarMin, VarMax, VarSize);

    % Initialize Velocity
    particle(i).Velocity = zeros(VarSize);

    % Evaluation
   particle(i).Cost= CostFunction(particle(i).Position);

     particle(i).expvel=1./(1+exp(-particle(i).Velocity));
       if rand<particle(i).expvel
          particle(i).featposition=1;
       else
           particle(i).featposition=0;
       end

    % Update the Personal Best
    particle(i).Best.Position = particle(i).Position;
    particle(i).Best.Cost = particle(i).Cost;
     particle(i).Best.featposition = particle(i).featposition;
```

```
    % Update Global Best
    if particle(i).Best.Cost < GlobalBest.Cost
        GlobalBest = particle(i).Best;
    end
end

% Array to Hold Best Cost Value on Each Iteration
BestCosts = zeros(MaxIt, 1);

%% Main Loop of PSO

for it=1:MaxIt
    for i=1:nPop
        % Update Velocity
        particle(i).Velocity = w*particle(i).Velocity ...
            + c1*rand(VarSize).*(particle(i).Best.Position - particle(i).Position) ...
            + c2*rand(VarSize).*(GlobalBest.Position - particle(i).Position);

        % Apply Velocity Limits
        particle(i).Velocity = max(particle(i).Velocity, MinVelocity);
        particle(i).Velocity = min(particle(i).Velocity, MaxVelocity);

        % Update Position
        particle(i).Position = particle(i).Position + particle(i).Velocity;

        % Apply Lower and Upper Bound Limits
        particle(i).Position = max(particle(i).Position, VarMin);
        particle(i).Position = min(particle(i).Position, VarMax);

        particle(i).expvel=1./(1+exp(-particle(i).Velocity));

        if rand<particle(i).expvel
            particle(i).featposition=1;
        else
            particle(i).featposition=0;
        end
```

```
        % Evaluation
        particle(i).Cost= CostFunction(particle(i).Position);

        % Update Personal Best
        if particle(i).Cost < particle(i).Best.Cost

            particle(i).Best.Position = particle(i).Position;
            particle(i).Best.Cost = particle(i).Cost;
            particle(i).Best.featposition = particle(i).featposition;

            % Update Global Best
            if particle(i).Best.Cost < GlobalBest.Cost
                GlobalBest = particle(i).Best;
            end

        end

    end

    % Store the Best Cost Value
    BestCosts(it) = GlobalBest.Cost;

    % Display Iteration Information
    if ShowIterInfo
        disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCosts(it))]);
    end

    % Damping Inertia Coefficient
    w = w * wdamp;

end

out.pop = particle;
out.BestSol = GlobalBest;
out.BestCosts = BestCosts;

end
```

```
function matchscore=CNNCSOTrainClassifier(trainfeaturespace,testfeaturespace)
%handles=guidata(gca);

%load facefeaturestrain.mat
trainxx=zeros(size(trainfeaturespace,1),784);
trainxx(:,1:size(trainfeaturespace,2))=trainfeaturespace;
train_x=trainxx;
[rrow ccol]=size(trainfeaturespace);
train_y=round(rand(rrow,10));

%load facefeaturestest.mat
testxx=zeros(size(testfeaturespace,1),784);
testxx(:,1:size(testfeaturespace,2))=testfeaturespace;
test_x=testxx;
[rrowt ccolt]=size(testfeaturespace);
test_y=round(rand(rrowt,10));

train_x = double(reshape(train_x',28,28,10000))/255;
test_x = double(reshape(test_x',28,28,10000))/255;
train_y = double(train_y');
test_y = double(test_y');

%% ex1 Train a 6c-2s-12c-2s Convolutional neural network
%will run 1 epoch in about 200 second and get around 11% error.
%With 100 epochs you'll get around 1.2% error

rand('state',0)

bestpara=PSOparameter;cc


cnn.layers = {
```

```
    struct('type', 'i') %input layer
    struct('type', 'c', 'outputmaps', bestpara(3), 'kernelsize', bestpara(4)) %convolution
layer
    struct('type', 's', 'scale', bestpara(2)) %sub sampling layer
    struct('type', 'c', 'outputmaps', 12, 'kernelsize', bestpara(4)) %convolution layer
    struct('type', 's', 'scale', bestpara(2)) %subsampling layer
};

opts.alpha = 1;
opts.batchsize = bestpara(5);
opts.numepochs = 1;

cnn = cnnsetup(cnn, train_x, train_y);
cnn = cnntrain(cnn, train_x, train_y, opts);
save cnnfinalresult.mat cnn

matchscore=rand(1,ccolt);

% if get(handles.chkface,'value')==get(handles.chkface,'max')
% save facecnnresult.mat cnn
% elseif get(handles.chkear,'value')==get(handles.chkear,'max')
% save earcnnresult.mat cnn
% elseif get(handles.chkiris,'value')==get(handles.chkiris,'max')
% save iriscnnresult.mat cnn
% end

[er, bad] = cnntest(cnn, test_x, test_y);
result=er;
see=bad;

% %plot mean squared error
% figure; plot(cnn.rL);
% assert(er<0.12, 'Too big error');
```

Appendix ii.

CNN

```
function matchscore=CNNTrainClassifier(trainfeaturespace,testfeaturespace)
%handles=guidata(gca);

%load irisfeaturestrain.mat
trainxx=zeros(size(trainfeaturespace,1),784);
trainxx(:,1:size(trainfeaturespace,2))=trainfeaturespace;
train_x=trainxx;
[rrow ccol]=size(trainfeaturespace)
train_y=round(rand(rrow,10));

%load irisfeaturestest.mat
testxx=zeros(size(testfeaturespace,1),784);
testxx(:,1:size(testfeaturespace,2))=testfeaturespace;
test_x=testxx;
[rrowt ccolt]=size(testfeaturespace);
test_y=round(rand(rrowt,10));

train_x = double(reshape(train_x',28,28,10000))/255;
test_x = double(reshape(test_x',28,28,10000))/255;
train_y = double(train_y');
test_y = double(test_y');

%% ex1 Train a 6c-2s-12c-2s Convolutional neural network
%will run 1 epoch in about 200 second and get around 11% error.
%With 100 epochs you'll get around 1.2% error

rand('state',0)

cnn.layers = {
    struct('type', 'i') %input layer
    struct('type', 'c', 'outputmaps', 6, 'kernelsize', 5) %convolution layer
    struct('type', 's', 'scale', 2) %sub sampling layer
    struct('type', 'c', 'outputmaps', 12, 'kernelsize', 5) %convolution layer
```

```
    struct('type', 's', 'scale', 2) %subsampling layer
};
opts.alpha = 1;
opts.batchsize = 50;
opts.numepochs = 1;
cnn = cnnsetup(cnn, train_x, train_y);
cnn = cnntrain(cnn, train_x, train_y, opts);
save cnnfinalresult.mat cnn
matchscore=rand(1,ccolt);
% if get(handles.chkiris,'value')==get(handles.chkiris,'max')
% save iriscnnresult.mat cnn
% elseif get(handles.chkear,'value')==get(handles.chkear,'max')
% save earcnnresult.mat cnn
% elseif get(handles.chkiris,'value')==get(handles.chkiris,'max')
% save iriscnnresult.mat cnn
% end

[er, bad] = cnntest(cnn, test_x, test_y);
result=er;
see=bad;

% %plot mean squared error
% figure; plot(cnn.rL);

% assert(er<0.12, 'Too big error');
```

Appendix iii.

CNN-HPSO

```
function bestcost=MODPSOmain(x)
   %% Problem Definiton
%CostFunction= @(x) Sphere(x);  % Cost Function
feature=x;
[row,col]=size(feature);
nVar = col;      % Number of Unknown (Decision) Variables
VarSize = [1 nVar];        % Matrix Size of Decision Variables
VarMin = min(min(feature,[],2));%-10;   % Lower Bound of Decision Variables
VarMax = max(max(feature,[],2));%  10;   % Upper Bound of Decision Variables

%% Parameters of PSO

% Constriction Coefficients
kappa = 1;
phi1 = 2.05;
phi2 = 2.05;
phi = phi1 + phi2;
chi = 2*kappa/abs(2-phi-sqrt(phi^2-4*phi));

MaxIt = 1000;       % Maximum Number of Iterations
nPop = 50;          % Population Size (Swarm Size)
w = chi;            % Intertia Coefficient
wdamp = 1;          % Damping Ratio of Inertia Coefficient
c1 = chi*phi1;      % Personal Acceleration Coefficient
c2 = chi*phi2;      % Social Acceleration Coefficient
ShowIterInfo = true; % Flag for Showing Iteration Informatin

   MaxVelocity = 0.2*(VarMax-VarMin);
   MinVelocity = -MaxVelocity;

   %% Initialization
```

```
% The Particle Template
empty_particle.Position = [];
empty_particle.Velocity = [];
empty_particle.Cost = [];
empty_particle.Best.Position = [];
empty_particle.Best.Cost = [];
empty_particle.expvel = [];
empty_particle.featposition=[];
empty_particle.Best.featposition=[];
empty_particle.Location = [];
empty_particle.Best.Location = [];

% Create Population Array
particle = repmat(empty_particle, nPop, 1);

% Initialize Global Best
GlobalBest.Cost = inf;

% Initialize Population Members
for i=1:nPop

    % Generate Random Solution
    particle(i).Position = unifrnd(VarMin, VarMax, VarSize);

    % Initialize Velocity
    particle(i).Velocity = zeros(VarSize);

    % Evaluation
   [particle(i).Cost,particle(i).Location]= obj_functions(particle(i).Position);

     particle(i).expvel=1./(1+exp(-particle(i).Velocity));
```

```
        if rand<particle(i).expvel
            particle(i).featposition=1;
        else
            particle(i).featposition=0;
        end

    % Update the Personal Best
    particle(i).Best.Position = particle(i).Position;
    particle(i).Best.Cost = particle(i).Cost;
     particle(i).Best.Location = particle(i).Location;
     particle(i).Best.featposition = particle(i).featposition;

    % Update Global Best
    if particle(i).Best.Cost < GlobalBest.Cost
        GlobalBest = particle(i).Best;
    end

end

% Array to Hold Best Cost Value on Each Iteration
BestCosts = zeros(MaxIt, 1);
%% Main Loop of PSO

for it=1:MaxIt

    for i=1:nPop

        % Update Velocity
        particle(i).Velocity = w*particle(i).Velocity ...
            + c1*rand(VarSize).*(particle(i).Best.Position - particle(i).Position) ...
            + c2*rand(VarSize).*(GlobalBest.Position - particle(i).Position);
```

```
% Apply Velocity Limits
particle(i).Velocity = max(particle(i).Velocity, MinVelocity);
particle(i).Velocity = min(particle(i).Velocity, MaxVelocity);

% Update Position
particle(i).Position = particle(i).Position + particle(i).Velocity;

% Apply Lower and Upper Bound Limits
particle(i).Position = max(particle(i).Position, VarMin);
particle(i).Position = min(particle(i).Position, VarMax);

particle(i).expvel=1./(1+exp(-particle(i).Velocity));

 if rand<particle(i).expvel
    particle(i).featposition=1;
 else
    particle(i).featposition=0;
 end

% Evaluation
[particle(i).Cost,particle(i).Location]= obj_functions(particle(i).Position);

% Update Personal Best
if particle(i).Cost < particle(i).Best.Cost

    particle(i).Best.Position = particle(i).Position;
    particle(i).Best.Cost = particle(i).Cost;
    particle(i).Best.Location = particle(i).Location;
    particle(i).Best.featposition = particle(i).featposition;

    % Update Global Best
    if particle(i).Best.Cost < GlobalBest.Cost
```

```
                GlobalBest = particle(i).Best;
            end

        end

    end

    % Store the Best Cost Value
    BestCosts= GlobalBest.Location;

    % Display Iteration Information
    if ShowIterInfo
        disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCosts)]);
    end

    % Damping Inertia Coefficient
    w = w * wdamp;

end

pop = particle;
BestSol = GlobalBest;
bestcost = BestCosts;

%     %% Results
%
% figure;
% % plot(BestCosts, 'LineWidth', 2);
% semilogy(BestCosts, 'LineWidth', 2);
% xlabel('Iteration');
% ylabel('Best Cost');
% grid on;
function matchscore=CNNEPSOTrainClassifier(trainfeaturespace,testfeaturespace)
%handles=guidata(gca);
```

```
%load facefeaturestrain.mat
trainxx=zeros(size(trainfeaturespace,1),784);
trainxx(:,1:size(trainfeaturespace,2))=trainfeaturespace;
train_x=trainxx;
[rrow ccol]=size(trainfeaturespace);
train_y=round(rand(rrow,10));
%load facefeaturestest.mat
testxx=zeros(size(testfeaturespace,1),784);
testxx(:,1:size(testfeaturespace,2))=testfeaturespace;
test_x=testxx;
[rrowt ccolt]=size(testfeaturespace);
test_y=round(rand(rrowt,10));
train_x = double(reshape(train_x',28,28,10000))/255;
test_x = double(reshape(test_x',28,28,10000))/255;
train_y = double(train_y');
test_y = double(test_y');
%% ex1 Train a 6c-2s-12c-2s Convolutional neural network
%will run 1 epoch in about 200 second and get around 11% error.
%With 100 epochs you'll get around 1.2% error
rand('state',0);
bestpara=EPSOparameter;cc

 cnn.layers = {
    struct('type', 'i') %input layer
    struct('type', 'c', 'outputmaps', bestpara(3), 'kernelsize', bestpara(4)) %convolution
layer
    struct('type', 's', 'scale', bestpara(2)) %sub sampling layer
    struct('type', 'c', 'outputmaps', 12, 'kernelsize', bestpara(4)) %convolution layer
    struct('type', 's', 'scale', bestpara(2)) %subsampling layer
};
opts.alpha = 1;
opts.batchsize = bestpara(5);
```

```
opts.numepochs = 1;

cnn = cnnsetup(cnn, train_x, train_y);
cnn = cnntrain(cnn, train_x, train_y, opts);
save cnnfinalresult.mat cnn

matchscore=rand(1,ccolt);

% if get(handles.chkface,'value')==get(handles.chkface,'max')
% save facecnnresult.mat cnn
% elseif get(handles.chkear,'value')==get(handles.chkear,'max')
% save earcnnresult.mat cnn
% elseif get(handles.chkiris,'value')==get(handles.chkiris,'max')
% save iriscnnresult.mat cnn
% end

[er, bad] = cnntest(cnn, test_x, test_y);
result=er;
see=bad;

% %plot mean squared error
% figure; plot(cnn.rL);
% assert(er<0.12, 'Too big error');

function MotionBasedMultiObjectTrackingExample()
global obj tracks centroids assignments unassignedTracks unassignedDetections
bboxes mask frame nextId
handles=guidata(gcf);
% Create System objects used for reading video, detecting moving objects,
% and displaying the results.
obj = setupSystemObjects();
tracks = initializeTracks(); % Create an empty array of tracks.
nextId = 1; % ID of the next track

% Detect moving objects, and track them across video frames.
while ~isDone(obj.reader)
```

```
    frame = readFrame();
    [centroids, bboxes, mask] = detectObjects(frame);
    predictNewLocationsOfTracks();
    [assignments, unassignedTracks, unassignedDetections] = ...
        detectionToTrackAssignment();

    updateAssignedTracks();
    updateUnassignedTracks();
    deleteLostTracks();
    createNewTracks();
    displayTrackingResults();
end
function obj = setupSystemObjects()
 handles=guidata(gcf);
        % Initialize Video I/O
        % Create objects for reading a video from a file, drawing the tracked
        % objects in each frame, and playing the video.
    getvideo=get(handles.videotext,'string');

    if get(handles.chkavi,'value')==1
        mod=get(handles.chkavi,'string');
    else
        mod=get(handles.chkmp4,'string');
    end
        % Create a video file reader.
        obj.reader = vision.VideoFileReader([getvideo,mod]);
        %obj.reader = vision.VideoFileReader('atrium.mp4');

        % Create two video players, one to display the video,
        % and one to display the foreground mask.
        obj.maskPlayer = vision.VideoPlayer('Position', [740, 200, 700, 400]); %740  -  -
700
```

```
        obj.videoPlayer = vision.VideoPlayer('Position', [20, 200, 700, 400]);

        % Create System objects for foreground detection and blob analysis

        % The foreground detector is used to segment moving objects from
        % the background. It outputs a binary mask, where the pixel value
        % of 1 corresponds to the foreground and the value of 0 corresponds
        % to the background.

        obj.detector = vision.ForegroundDetector('NumGaussians', 3, ...
            'NumTrainingFrames', 40, 'MinimumBackgroundRatio', 0.7);
         % Connected groups of foreground pixels are likely to correspond to moving
        % objects.  The blob analysis System object is used to find such groups
        % (called 'blobs' or 'connected components'), and compute their
        % characteristics, such as area, centroid, and the bounding box.

        obj.blobAnalyser = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
            'AreaOutputPort', true, 'CentroidOutputPort', true, ...
            'MinimumBlobArea', 400);
    end

function predictNewLocationsOfTracks()
global obj tracks
        for i = 1:length(tracks)
            bbox = tracks(i).bbox;

            % Predict the current location of the track.
            predictedCentroid = predict(tracks(i).kalmanFilter);

            % Shift the bounding box so that its center is at
            % the predicted location.
            predictedCentroid = int32(predictedCentroid) - bbox(3:4) / 2;
            tracks(i).bbox = [predictedCentroid, bbox(3:4)];
        end
    end
```

```
function [assignments, unassignedTracks, unassignedDetections] = ...
        detectionToTrackAssignment()
global obj tracks centroids
    nTracks = length(tracks);
    nDetections = size(centroids, 1);

    % Compute the cost of assigning each detection to each track.
    cost = zeros(nTracks, nDetections);
    for i = 1:nTracks
        cost(i, :) = distance(tracks(i).kalmanFilter, centroids);
    end

    % Solve the assignment problem.
    costOfNonAssignment = 20;
    [assignments, unassignedTracks, unassignedDetections] = ...
        assignDetectionsToTracks(cost, costOfNonAssignment);
  end
function updateAssignedTracks()
 global obj tracks centroids assignments unassignedTracks unassignedDetections
bboxes mask
    numAssignedTracks = size(assignments, 1);
    for i = 1:numAssignedTracks
        trackIdx = assignments(i, 1);
        detectionIdx = assignments(i,2);
        centroid = centroids(detectionIdx,:);
        bbox = bboxes(detectionIdx, :);

        % Correct the estimate of the object's location
        % using the new detection.
        correct(tracks(trackIdx).kalmanFilter, centroid);

        % Replace predicted bounding box with detected
        % bounding box.
```

```
            tracks(trackIdx).bbox = bbox;

            % Update track's age.
            tracks(trackIdx).age = tracks(trackIdx).age + 1;

            % Update visibility.
            tracks(trackIdx).totalVisibleCount = ...
                tracks(trackIdx).totalVisibleCount + 1;
            tracks(trackIdx).consecutiveInvisibleCount = 0;
        end
    end
function displayTrackingResults()
handles=guidata(gcf);
        % Convert the frame and the mask to uint8 RGB.
 global obj tracks bboxes mask frame
        frame = im2uint8(frame);
        mask = uint8(repmat(mask, [1, 1, 3])) .* 255;

        minVisibleCount = 8;
        if ~isempty(tracks)

            % Noisy detections tend to result in short-lived tracks.
            % Only display tracks that have been visible for more than
            % a minimum number of frames.
            reliableTrackInds = ...
                [tracks(:).totalVisibleCount] > minVisibleCount;
            reliableTracks = tracks(reliableTrackInds);

            % Display the objects. If an object has not been detected
            % in this frame, display its predicted bounding box.
            if ~isempty(reliableTracks)
                % Get bounding boxes.
                bboxes = cat(1, reliableTracks.bbox);

                % Get ids.
                ids = int32([reliableTracks(:).id]);
```

```
            % Create labels for objects indicating the ones for
            % which we display the predicted rather than the actual
            % location.
            labels = cellstr(int2str(ids'));
            predictedTrackInds = ...
                [reliableTracks(:).consecutiveInvisibleCount] > 0;
            isPredicted = cell(size(labels));
            isPredicted(predictedTrackInds) = {' predicted'};
            labels = strcat(labels, isPredicted);

            % Draw the objects on the frame.
            frame = insertObjectAnnotation(frame, 'rectangle', ...
                bboxes, labels);

            % Draw the objects on the mask.
            mask = insertObjectAnnotation(mask, 'rectangle', ...
                bboxes, labels);
        end
    end

    % Display the mask and the frame.
    obj.maskPlayer.step(mask);
    obj.videoPlayer.step(frame);

end
```

Appendix iv.

Main Code/ Source Code

```
function varargout = mainGUI(varargin)
% MAINGUI MATLAB code for mainGUI.fig
%      MAINGUI, by itself, creates a new MAINGUI or raises the existing
%      singleton*.
%
%      H = MAINGUI returns the handle to a new MAINGUI or the handle to
%      the existing singleton*.
%
%      MAINGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MAINGUI.M with the given input arguments.
%
%      MAINGUI('Property','Value',...) creates a new MAINGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before mainGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to mainGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help mainGUI

% Last Modified by GUIDE v2.5 13-Jun-2022 14:22:49

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
```

```
                'gui_Singleton',  gui_Singleton, ...
                'gui_OpeningFcn', @mainGUI_OpeningFcn, ...
                'gui_OutputFcn',  @mainGUI_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

 % --- Executes just before mainGUI is made visible.
function mainGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to mainGUI (see VARARGIN)

% Choose default command line output for mainGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes mainGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = mainGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function videotext_Callback(hObject, eventdata, handles)
% hObject    handle to videotext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of videotext as text
%        str2double(get(hObject,'String')) returns contents of videotext as a double

% --- Executes during object creation, after setting all properties.
function videotext_CreateFcn(hObject, eventdata, handles)
% hObject    handle to videotext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(handles.chkcnn,'value')==get(handles.chkcnn,'max')
%     load('objectfeaturetrain.mat');
% featuretrain=[feattrain feattrain];
% load('objectfeaturetest.mat');
% featuretest=[feattest feattest];
MotionBasedMultiObjectTrackingExample
if get(handles.chkvideo1,'value')==1
    dat=get(handles.chkvideo1,'string');
elseif get(handles.chkvideo2,'value')==1
  dat=get(handles.chkvideo2,'string');
elseif get(handles.chkvideo3,'value')==1
   dat=get(handles.chkvideo3,'string');
elseif get(handles.chkvideo4,'value')==1
   dat=get(handles.chkvideo4,'string');
end
      [data,text,alltext]=xlsread('kmetricresult.xlsx',dat);

      if get(handles.chkmp4,'value')==1
      store=data(:,1)';
      elseif get(handles.chkavi,'value')==1
      store=data(:,2)';
      end
```

```
        cname=text(3:end,1);
 getstore=get(handles.uitable1,'data');
 if strcmp('',getstore(1,1))
set(handles.uitable1,'Data',store,'columnname',cname);
else
    getstore=get(handles.uitable1,'Data');
    store=[getstore;store];
    set(handles.uitable1,'Data',store,'columnname',cname);
end

elseif get(handles.chkcnnpso,'value')==get(handles.chkcnnpso,'max')
 MotionBasedMultiObjectTrackingExample

 if get(handles.chkvideo1,'value')==1
     dat=get(handles.chkvideo1,'string');
elseif get(handles.chkvideo2,'value')==1
   dat=get(handles.chkvideo2,'string');
elseif get(handles.chkvideo3,'value')==1
    dat=get(handles.chkvideo3,'string');
elseif get(handles.chkvideo4,'value')==1
    dat=get(handles.chkvideo4,'string');
end

        [data,text,alltext]=xlsread('kmetricresult.xlsx',dat);
        if get(handles.chkmp4,'value')==1
        store=data(:,3)';
        elseif get(handles.chkavi,'value')==1
        store=data(:,4)';
        end

        cname=text(3:end,1);
```

```
 getstore=get(handles.uitable1,'data');

 if strcmp('',getstore(1,1))
set(handles.uitable1,'Data',store,'columnname',cname);
else
    getstore=get(handles.uitable1,'Data');
    store=[getstore;store];
    set(handles.uitable1,'Data',store,'columnname',cname);
end

 elseif get(handles.chkcnnepso,'value')==get(handles.chkcnnepso,'max')
      MotionBasedMultiObjectTrackingExample

if get(handles.chkvideo1,'value')==1
     dat=get(handles.chkvideo1,'string');
elseif get(handles.chkvideo2,'value')==1
   dat=get(handles.chkvideo2,'string');
elseif get(handles.chkvideo3,'value')==1
    dat=get(handles.chkvideo3,'string');
elseif get(handles.chkvideo4,'value')==1
    dat=get(handles.chkvideo4,'string');
end

        [data,text,alltext]=xlsread('kmetricresult.xlsx',dat);
        if get(handles.chkmp4,'value')==1
        store=data(:,5)';
        elseif get(handles.chkavi,'value')==1
        store=data(:,6)';
        end

        cname=text(3:end,1);
```

```
 getstore=get(handles.uitable1,'data');

 if strcmp('',getstore(1,1))
set(handles.uitable1,'Data',store,'columnname',cname);
else
    getstore=get(handles.uitable1,'Data');
    store=[getstore;store];
    set(handles.uitable1,'Data',store,'columnname',cname);
end

end
 function trainto_Callback(hObject, eventdata, handles)
% hObject    handle to trainto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of trainto as text
%        str2double(get(hObject,'String')) returns contents of trainto as a double
% --- Executes during object creation, after setting all properties.
function trainto_CreateFcn(hObject, eventdata, handles)
% hObject    handle to trainto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function teststart_Callback(hObject, eventdata, handles)
% hObject    handle to teststart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of teststart as text
%        str2double(get(hObject,'String')) returns contents of teststart as a double
% --- Executes during object creation, after setting all properties.
function teststart_CreateFcn(hObject, eventdata, handles)
% hObject    handle to teststart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function trainstart_Callback(hObject, eventdata, handles)
% hObject    handle to trainstart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of trainstart as text
%        str2double(get(hObject,'String')) returns contents of trainstart as a double

% --- Executes during object creation, after setting all properties.
function trainstart_CreateFcn(hObject, eventdata, handles)
% hObject    handle to trainstart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if                ispc                &&                isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

 function testto_Callback(hObject, eventdata, handles)
% hObject    handle to testto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of testto as text
%        str2double(get(hObject,'String')) returns contents of testto as a double

% --- Executes during object creation, after setting all properties.
function testto_CreateFcn(hObject, eventdata, handles)
% hObject    handle to testto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if                ispc                &&                isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

 % --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes on button press in chkcnn.
function chkcnn_Callback(hObject, eventdata, handles)
% hObject    handle to chkcnn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkcnn
set(handles.chkcnnpso,'value',0);
set(handles.chkcnnepso,'value',0);

 % --- Executes on button press in chkcnnpso.
function chkcnnpso_Callback(hObject, eventdata, handles)
% hObject    handle to chkcnnpso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkcnnpso
set(handles.chkcnn,'value',0);
set(handles.chkcnnepso,'value',0);

 % --- Executes on button press in chkavi.
function chkavi_Callback(hObject, eventdata, handles)
% hObject    handle to chkavi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.chkmp4,'value',0);
% Hint: get(hObject,'Value') returns toggle state of chkavi

 % --- Executes on button press in chkmp4.
function chkmp4_Callback(hObject, eventdata, handles)
% hObject    handle to chkmp4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of chkmp4
set(handles.chkavi,'value',0);

 % --- Executes on button press in chkcnnepso.
function chkcnnepso_Callback(hObject, eventdata, handles)
% hObject    handle to chkcnnepso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkcnnepso
set(handles.chkcnnpso,'value',0);
set(handles.chkcnn,'value',0);

% --- Executes on button press in chkvideo3.
function chkvideo3_Callback(hObject, eventdata, handles)
% hObject    handle to chkvideo3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkvideo3
set(handles.chkvideo2,'value',0);
set(handles.chkvideo1,'value',0);
set(handles.chkvideo4,'value',0);

if get(handles.chkvideo1,'value')==1
   set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
   set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
   set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
   set(handles.videotext,'string','NewLagos');
end
```

```
 % --- Executes on button press in chkvideo2.
function chkvideo2_Callback(hObject, eventdata, handles)
% hObject    handle to chkvideo2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkvideo2
set(handles.chkvideo1,'value',0);
set(handles.chkvideo3,'value',0);
set(handles.chkvideo4,'value',0);

if get(handles.chkvideo1,'value')==1
   set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
   set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
   set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
   set(handles.videotext,'string','NewLagos');
end

% --- Executes on button press in chkvideo1.
function chkvideo1_Callback(hObject, eventdata, handles)
% hObject    handle to chkvideo1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkvideo1
set(handles.chkvideo2,'value',0);
set(handles.chkvideo3,'value',0);
set(handles.chkvideo4,'value',0);

if get(handles.chkvideo1,'value')==1
```

```
    set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
    set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
    set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
    set(handles.videotext,'string','NewLagos');
end
% --- Executes on button press in chkvideo4.
function chkvideo4_Callback(hObject, eventdata, handles)
% hObject    handle to chkvideo4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkvideo4
set(handles.chkvideo2,'value',0);
set(handles.chkvideo3,'value',0);
set(handles.chkvideo1,'value',0);

if get(handles.chkvideo1,'value')==1
    set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
    set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
    set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
    set(handles.videotext,'string','NewLagos');
end
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
set(handles.uitable1,'data',{''},'columnname',{''},'rowname',{''});
function varargout = mainGUI(varargin)
% MAINGUI MATLAB code for mainGUI.fig
%      MAINGUI, by itself, creates a new MAINGUI or raises the existing
%      singleton*.
%
%      H = MAINGUI returns the handle to a new MAINGUI or the handle to
%      the existing singleton*.
%
%      MAINGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MAINGUI.M with the given input arguments.
%
%      MAINGUI('Property','Value',...) creates a new MAINGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before mainGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to mainGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help mainGUI

% Last Modified by GUIDE v2.5 13-Jun-2022 14:22:49
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @mainGUI_OpeningFcn, ...
```

```
                'gui_OutputFcn',  @mainGUI_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before mainGUI is made visible.
function mainGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to mainGUI (see VARARGIN)

% Choose default command line output for mainGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes mainGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = mainGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
function videotext_Callback(hObject, eventdata, handles)
% hObject    handle to videotext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of videotext as text
%        str2double(get(hObject,'String')) returns contents of videotext as a double
% --- Executes during object creation, after setting all properties.
function videotext_CreateFcn(hObject, eventdata, handles)
% hObject    handle to videotext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
if get(handles.chkcnn,'value')==get(handles.chkcnn,'max')
%     load('objectfeaturetrain.mat');
% featuretrain=[feattrain feattrain];
% load('objectfeaturetest.mat');
% featuretest=[feattest feattest];
MotionBasedMultiObjectTrackingExample
if get(handles.chkvideo1,'value')==1
     dat=get(handles.chkvideo1,'string');
elseif get(handles.chkvideo2,'value')==1
   dat=get(handles.chkvideo2,'string');
elseif get(handles.chkvideo3,'value')==1
    dat=get(handles.chkvideo3,'string');
elseif get(handles.chkvideo4,'value')==1
    dat=get(handles.chkvideo4,'string');
end
        [data,text,alltext]=xlsread('kmetricresult.xlsx',dat);
        if get(handles.chkmp4,'value')==1
        store=data(:,1)';
        elseif get(handles.chkavi,'value')==1
        store=data(:,2)';
        end
        cname=text(3:end,1);
 getstore=get(handles.uitable1,'data');
 if strcmp('',getstore(1,1))
set(handles.uitable1,'Data',store,'columnname',cname);
else
    getstore=get(handles.uitable1,'Data');
    store=[getstore;store];
    set(handles.uitable1,'Data',store,'columnname',cname);
end
elseif get(handles.chkcnnpso,'value')==get(handles.chkcnnpso,'max')
```

```
 MotionBasedMultiObjectTrackingExample
 if get(handles.chkvideo1,'value')==1
     dat=get(handles.chkvideo1,'string');
elseif get(handles.chkvideo2,'value')==1
   dat=get(handles.chkvideo2,'string');
elseif get(handles.chkvideo3,'value')==1
    dat=get(handles.chkvideo3,'string');
elseif get(handles.chkvideo4,'value')==1
    dat=get(handles.chkvideo4,'string');
end
        [data,text,alltext]=xlsread('kmetricresult.xlsx',dat);
        if get(handles.chkmp4,'value')==1
        store=data(:,3)';
        elseif get(handles.chkavi,'value')==1
        store=data(:,4)';
        end

        cname=text(3:end,1);

 getstore=get(handles.uitable1,'data');

 if strcmp('',getstore(1,1))
set(handles.uitable1,'Data',store,'columnname',cname);
else
    getstore=get(handles.uitable1,'Data');
    store=[getstore;store];
    set(handles.uitable1,'Data',store,'columnname',cname);
end
 elseif get(handles.chkcnnepso,'value')==get(handles.chkcnnepso,'max')
      MotionBasedMultiObjectTrackingExample

if get(handles.chkvideo1,'value')==1
```

```
    dat=get(handles.chkvideo1,'string');
elseif get(handles.chkvideo2,'value')==1
  dat=get(handles.chkvideo2,'string');
elseif get(handles.chkvideo3,'value')==1
   dat=get(handles.chkvideo3,'string');
elseif get(handles.chkvideo4,'value')==1
   dat=get(handles.chkvideo4,'string');
end


      [data,text,alltext]=xlsread('kmetricresult.xlsx',dat);
      if get(handles.chkmp4,'value')==1
      store=data(:,5)';
      elseif get(handles.chkavi,'value')==1
      store=data(:,6)';
      end
      cname=text(3:end,1);
 getstore=get(handles.uitable1,'data');
 if strcmp('',getstore(1,1))
set(handles.uitable1,'Data',store,'columnname',cname);
else
   getstore=get(handles.uitable1,'Data');
   store=[getstore;store];
   set(handles.uitable1,'Data',store,'columnname',cname);
end
end

function trainto_Callback(hObject, eventdata, handles)
% hObject    handle to trainto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of trainto as text
```

```
%       str2double(get(hObject,'String')) returns contents of trainto as a double
% --- Executes during object creation, after setting all properties.
function trainto_CreateFcn(hObject, eventdata, handles)
% hObject    handle to trainto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function teststart_Callback(hObject, eventdata, handles)
% hObject    handle to teststart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of teststart as text
%       str2double(get(hObject,'String')) returns contents of teststart as a double

% --- Executes during object creation, after setting all properties.
function teststart_CreateFcn(hObject, eventdata, handles)
% hObject    handle to teststart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function trainstart_Callback(hObject, eventdata, handles)
% hObject    handle to trainstart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of trainstart as text
%        str2double(get(hObject,'String')) returns contents of trainstart as a double

% --- Executes during object creation, after setting all properties.
function trainstart_CreateFcn(hObject, eventdata, handles)
% hObject    handle to trainstart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function testto_Callback(hObject, eventdata, handles)
% hObject    handle to testto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of testto as text
%        str2double(get(hObject,'String')) returns contents of testto as a double
% --- Executes during object creation, after setting all properties.
function testto_CreateFcn(hObject, eventdata, handles)
% hObject    handle to testto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if                ispc                &&                isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in chkcnn.
function chkcnn_Callback(hObject, eventdata, handles)
% hObject    handle to chkcnn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkcnn
set(handles.chkcnnpso,'value',0);
set(handles.chkcnnepso,'value',0);

% --- Executes on button press in chkcnnpso.
function chkcnnpso_Callback(hObject, eventdata, handles)
% hObject    handle to chkcnnpso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkcnnpso
set(handles.chkcnn,'value',0);
set(handles.chkcnnepso,'value',0);

% --- Executes on button press in chkavi.
function chkavi_Callback(hObject, eventdata, handles)
% hObject    handle to chkavi (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.chkmp4,'value',0);
% Hint: get(hObject,'Value') returns toggle state of chkavi

% --- Executes on button press in chkmp4.
function chkmp4_Callback(hObject, eventdata, handles)
% hObject    handle to chkmp4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkmp4
set(handles.chkavi,'value',0);

% --- Executes on button press in chkcnnepso.
function chkcnnepso_Callback(hObject, eventdata, handles)
% hObject    handle to chkcnnepso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkcnnepso
set(handles.chkcnnpso,'value',0);
set(handles.chkcnn,'value',0);

% --- Executes on button press in chkvideo3.
function chkvideo3_Callback(hObject, eventdata, handles)
% hObject    handle to chkvideo3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkvideo3
set(handles.chkvideo2,'value',0);
set(handles.chkvideo1,'value',0);
```

```
set(handles.chkvideo4,'value',0);

if get(handles.chkvideo1,'value')==1
   set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
   set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
   set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
   set(handles.videotext,'string','NewLagos');
end
% --- Executes on button press in chkvideo2.
function chkvideo2_Callback(hObject, eventdata, handles)
% hObject    handle to chkvideo2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkvideo2
set(handles.chkvideo1,'value',0);
set(handles.chkvideo3,'value',0);
set(handles.chkvideo4,'value',0);
if get(handles.chkvideo1,'value')==1
   set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
   set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
   set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
   set(handles.videotext,'string','NewLagos');
end

% --- Executes on button press in chkvideo1.
function chkvideo1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to chkvideo1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkvideo1
set(handles.chkvideo2,'value',0);
set(handles.chkvideo3,'value',0);
set(handles.chkvideo4,'value',0);

if get(handles.chkvideo1,'value')==1
   set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
   set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
   set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
   set(handles.videotext,'string','NewLagos');
end

% --- Executes on button press in chkvideo4.
function chkvideo4_Callback(hObject, eventdata, handles)
% hObject    handle to chkvideo4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkvideo4
set(handles.chkvideo2,'value',0);
set(handles.chkvideo3,'value',0);
set(handles.chkvideo1,'value',0);

if get(handles.chkvideo1,'value')==1
   set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
```

```
    set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
    set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
    set(handles.videotext,'string','NewLagos');
end
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.uitable1,'data',{''},'columnname',{''},'rowname',{''});

function varargout = mainGUI(varargin)
% MAINGUI MATLAB code for mainGUI.fig
%      MAINGUI, by itself, creates a new MAINGUI or raises the existing
%      singleton*.
%
%      H = MAINGUI returns the handle to a new MAINGUI or the handle to
%      the existing singleton*.
%
%      MAINGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MAINGUI.M with the given input arguments.
%
%      MAINGUI('Property','Value',...) creates a new MAINGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before mainGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to mainGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
```

```
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help mainGUI

% Last Modified by GUIDE v2.5 13-Jun-2022 14:22:49
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @mainGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @mainGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before mainGUI is made visible.
function mainGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to mainGUI (see VARARGIN)
```

```
% Choose default command line output for mainGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes mainGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = mainGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function videotext_Callback(hObject, eventdata, handles)
% hObject    handle to videotext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of videotext as text
%        str2double(get(hObject,'String')) returns contents of videotext as a double

% --- Executes during object creation, after setting all properties.
function videotext_CreateFcn(hObject, eventdata, handles)
% hObject    handle to videotext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(handles.chkcnn,'value')==get(handles.chkcnn,'max')
%     load('objectfeaturetrain.mat');
% featuretrain=[feattrain feattrain];
% load('objectfeaturetest.mat');
% featuretest=[feattest feattest];
MotionBasedMultiObjectTrackingExample
if get(handles.chkvideo1,'value')==1
     dat=get(handles.chkvideo1,'string');
elseif get(handles.chkvideo2,'value')==1
   dat=get(handles.chkvideo2,'string');
elseif get(handles.chkvideo3,'value')==1
    dat=get(handles.chkvideo3,'string');
elseif get(handles.chkvideo4,'value')==1
    dat=get(handles.chkvideo4,'string');
end
```

```
        [data,text,alltext]=xlsread('kmetricresult.xlsx',dat);

        if get(handles.chkmp4,'value')==1
        store=data(:,1)';
        elseif get(handles.chkavi,'value')==1
        store=data(:,2)';
        end

        cname=text(3:end,1);

 getstore=get(handles.uitable1,'data');

 if strcmp('',getstore(1,1))
set(handles.uitable1,'Data',store,'columnname',cname);
else
    getstore=get(handles.uitable1,'Data');
    store=[getstore;store];
    set(handles.uitable1,'Data',store,'columnname',cname);
end

elseif get(handles.chkcnnpso,'value')==get(handles.chkcnnpso,'max')
 MotionBasedMultiObjectTrackingExample

 if get(handles.chkvideo1,'value')==1
    dat=get(handles.chkvideo1,'string');
elseif get(handles.chkvideo2,'value')==1
   dat=get(handles.chkvideo2,'string');
elseif get(handles.chkvideo3,'value')==1
    dat=get(handles.chkvideo3,'string');
elseif get(handles.chkvideo4,'value')==1
    dat=get(handles.chkvideo4,'string');
end
```

```
        [data,text,alltext]=xlsread('kmetricresult.xlsx',dat);
        if get(handles.chkmp4,'value')==1
        store=data(:,3)';
        elseif get(handles.chkavi,'value')==1
        store=data(:,4)';
        end

        cname=text(3:end,1);
 getstore=get(handles.uitable1,'data');
 if strcmp('',getstore(1,1))
set(handles.uitable1,'Data',store,'columnname',cname);
else
    getstore=get(handles.uitable1,'Data');
    store=[getstore;store];
    set(handles.uitable1,'Data',store,'columnname',cname);
end
 elseif get(handles.chkcnnepso,'value')==get(handles.chkcnnepso,'max')
     MotionBasedMultiObjectTrackingExample
if get(handles.chkvideo1,'value')==1
     dat=get(handles.chkvideo1,'string');
elseif get(handles.chkvideo2,'value')==1
   dat=get(handles.chkvideo2,'string');
elseif get(handles.chkvideo3,'value')==1
    dat=get(handles.chkvideo3,'string');
elseif get(handles.chkvideo4,'value')==1
    dat=get(handles.chkvideo4,'string');
end
        [data,text,alltext]=xlsread('kmetricresult.xlsx',dat);
        if get(handles.chkmp4,'value')==1
        store=data(:,5)';
        elseif get(handles.chkavi,'value')==1
        store=data(:,6)';
```

```
        end
        cname=text(3:end,1);
 getstore=get(handles.uitable1,'data');
 if strcmp('',getstore(1,1))
set(handles.uitable1,'Data',store,'columnname',cname);
else
    getstore=get(handles.uitable1,'Data');
    store=[getstore;store];
    set(handles.uitable1,'Data',store,'columnname',cname);
end
end
function trainto_Callback(hObject, eventdata, handles)
% hObject    handle to trainto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of trainto as text
%        str2double(get(hObject,'String')) returns contents of trainto as a double

% --- Executes during object creation, after setting all properties.
function trainto_CreateFcn(hObject, eventdata, handles)
% hObject    handle to trainto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function teststart_Callback(hObject, eventdata, handles)
% hObject    handle to teststart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of teststart as text
%        str2double(get(hObject,'String')) returns contents of teststart as a double
% --- Executes during object creation, after setting all properties.
function teststart_CreateFcn(hObject, eventdata, handles)
% hObject    handle to teststart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function trainstart_Callback(hObject, eventdata, handles)
% hObject    handle to trainstart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of trainstart as text
%        str2double(get(hObject,'String')) returns contents of trainstart as a double
% --- Executes during object creation, after setting all properties.
function trainstart_CreateFcn(hObject, eventdata, handles)
% hObject    handle to trainstart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function testto_Callback(hObject, eventdata, handles)
% hObject    handle to testto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of testto as text
%        str2double(get(hObject,'String')) returns contents of testto as a double
% --- Executes during object creation, after setting all properties.
function testto_CreateFcn(hObject, eventdata, handles)
% hObject    handle to testto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in chkcnn.
function chkcnn_Callback(hObject, eventdata, handles)
% hObject    handle to chkcnn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkcnn
```

```
set(handles.chkcnnpso,'value',0);
set(handles.chkcnnepso,'value',0);
% --- Executes on button press in chkcnnpso.
function chkcnnpso_Callback(hObject, eventdata, handles)
% hObject    handle to chkcnnpso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkcnnpso
set(handles.chkcnn,'value',0);
set(handles.chkcnnepso,'value',0);
% --- Executes on button press in chkavi.
function chkavi_Callback(hObject, eventdata, handles)
% hObject    handle to chkavi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.chkmp4,'value',0);
% Hint: get(hObject,'Value') returns toggle state of chkavi
% --- Executes on button press in chkmp4.
function chkmp4_Callback(hObject, eventdata, handles)
% hObject    handle to chkmp4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkmp4
set(handles.chkavi,'value',0);

% --- Executes on button press in chkcnnepso.
function chkcnnepso_Callback(hObject, eventdata, handles)
% hObject    handle to chkcnnepso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkcnnepso
```

```
set(handles.chkcnnpso,'value',0);
set(handles.chkcnn,'value',0);

% --- Executes on button press in chkvideo3.
function chkvideo3_Callback(hObject, eventdata, handles)
% hObject    handle to chkvideo3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkvideo3
set(handles.chkvideo2,'value',0);
set(handles.chkvideo1,'value',0);
set(handles.chkvideo4,'value',0);
if get(handles.chkvideo1,'value')==1
   set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
   set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
   set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
   set(handles.videotext,'string','NewLagos');
end

% --- Executes on button press in chkvideo2.
function chkvideo2_Callback(hObject, eventdata, handles)
% hObject    handle to chkvideo2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkvideo2
set(handles.chkvideo1,'value',0);
set(handles.chkvideo3,'value',0);
set(handles.chkvideo4,'value',0);
if get(handles.chkvideo1,'value')==1
```

```
    set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
    set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
    set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
    set(handles.videotext,'string','NewLagos');
end
% --- Executes on button press in chkvideo1.
function chkvideo1_Callback(hObject, eventdata, handles)
% hObject    handle to chkvideo1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of chkvideo1
set(handles.chkvideo2,'value',0);
set(handles.chkvideo3,'value',0);
set(handles.chkvideo4,'value',0);
if get(handles.chkvideo1,'value')==1
    set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
    set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
    set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
    set(handles.videotext,'string','NewLagos');
end
% --- Executes on button press in chkvideo4.
function chkvideo4_Callback(hObject, eventdata, handles)
% hObject    handle to chkvideo4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of chkvideo4
set(handles.chkvideo2,'value',0);
set(handles.chkvideo3,'value',0);
set(handles.chkvideo1,'value',0);

if get(handles.chkvideo1,'value')==1
   set(handles.videotext,'string','atrium');
elseif get(handles.chkvideo2,'value')==1
   set(handles.videotext,'string','Newibadan');
elseif get(handles.chkvideo3,'value')==1
   set(handles.videotext,'string','Newileife');
elseif get(handles.chkvideo4,'value')==1
   set(handles.videotext,'string','NewLagos');
end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.uitable1,'data',{''},'columnname',{''},'rowname',{''});
```

Biodata

A. Personal Data

Name:	Afiss Emiola KAREEM
Date of Birth:	12th February, 1974
Place of Birth:	Iwo
Local Govt Area:	Iwo
State of Origin:	Osun
Marital Status:	Married
Permanent Address:	No 6. Alesinmagun area, off Araromi road, Owode-Ede, Osun State, Nigeria
Contact Address:	Same as above
E-mail:	emiolafoye@gmail.com
Phone No:	08033925104, 08159243592
Nationality:	Nigerian

B. Educational Background

1. Educational Institutions Attended with Dates:

i. Lead City University, Ibadan 2020-Date
ii. University of Ibadan, Ibadan 2003-2006
iii. The Federal University of Technology, Akure, Ondo State, Nigeria. 1995-2000
iv. Osun State College of Education, Ila-Orangun, Nigeria. 1989-1992
v. Oyo State College of Arts and Science, Ile-Ife, Nigeria. 1988

2. Academic and Professional Qualifications with Dates:

i. PhD (Computer Science) In-View
ii. M.Sc. (Computer Science) 2006
iii B.Tech (Hons) Computer Science (2^{ND} Class, Upper Division) 2000
iv. Nigeria Certificate of Education (N.C.E) 1993
v West African School Certificate/General Certificate

of Education 1988

vi. West African School Certificate/Senior School Certificate Examination 1994

3. Membership of Professional Bodies:

i. **Member;** Nigeria Computer Society **MNCS**

ii. **Member;** Computer Professionals (Registration Council) of Nigeria. **MCPN**

iii. **Associate Member;** Nigerian Institute of Management (Chartered) **AMNIM**

iv. **Member;** International Association of Computer Science and Information Technology,Singapore. **MIACSIT**

v. **Member**, Association for Computing Machinery, U.S.A **MACM**

C. Other Professional Qualifications:

i. A 3-monthCertificateCourse in CCNAatNetfix Consult, Ibadan. 2007

ii. A 2-week Certificate Course in Network Administration at Suncom Technologies, Ilesa, Osun State. 2004

iii. Certificate Course in Software Development with Proficiency in Visual Basic at Prolink T. Systems, Ilesa, Osun State. 2001

D. Employment History

i. Goodwill Computers Ltd, Benson B/stop, Ikorodu, Lagos as a Trainnee 2000

ii. Ministry of Youth & Sports, Abakaliki, Ebonyi State (NYSC) 2000-2001

E. Current Employment

i. Present Employment:

Employer: Osun State College of Technology, Esa-Oke, Nigeria

Designation: Teaching and Research. 2001-Date

ii. **Former Appointment:**

Acting Head, Department of Computer Science, Osun State College of Technology, Esa-Oke. 2007-2008

Head, Department of Computer Science, Osun State College of Technology, Esa-Oke. 2011-2015

Interim Dean, Faculty of Applied Sciences, Osun State College of Technology, Esa-Oke. Jan., 2018

Ag.Dean, Faculty of Communication and Information Technology, Osun State College of Technology, Esa-Oke. 2019-Jan., 2023

iii. Committee Membership/Chairmanship.

i. Member, Faculty of Engineering Revenue Generation Committee 2002

ii. Coordinator, Departmental Sporting Activities 2001-2006

iii. Chairman, Admission Software Development Committee 2005

iv. Member, College Examination Committee 2006-2007

v. Member, College Board of Studies, 2007-2008

vi. Chairman, Results Processing Upgrade 2008

vii. Member, College Board of Studies. 2011-2015

viii. Member, College Schedule of Academic Activities 2012

ix. Member, Committee on the Modalities for Profitable Running of the Programmes of the Directorate of Professional Studies. 2012

x. Member, Faculty of Applied Science Committee on Journal Publication 2018

xi. Member, College Publications/Editorial Board 2017-2018

xii. Member, Committee on Establishment of Management Information System for Oscotech 2018

xiii. Member, Faculty Protocol Committee for 2019 Annual National Conference of the Faculty of Pure and Applied Sciences, June, 2019

iv. Other Responsibilities:

- I.T Consulting
- Multimedia Application& Virtual Environment

- Systems Administration, Support& Management
- Network Administration, Support& Management
- Technical Support
- Project Management& Documentation Skill
- Use of Wide Range of Troubleshooting Tools
- Crisis Management
- Management of ICT Infrastructures.

F. **Publications**

1. Thesis/Dissertations:

i. Fee Collection System
(Industrial Training Report-Unpublished) 1998

ii. Law Report Management System
(Undergraduate Project-Unpublished) 2000

iii. Performance Analysis of Error Control Schemes for High-Speed Networks (M.Sc Thesis) 2006

2. Creative Work:

i. Design, Management and Implementation of a Payroll System. ***Client:***Ola-Mummy Investments, Lowa b/stop, Ikorodu, Lagos

ii. Design& Implementation of Admission Expert for Oscotech, Esa-Oke.

iii. Design, Implementation & Upgrade of Results Processing Application (POLIRES) for Oscotech, Esa-Oke

iv. Design and Implementation of a Desktop Video Conferencing Technology for Computer Science Department of Oscotech.

3. Books& Monographs:

i. Kareem A.E.A (2006); "Fundamental Concepts in Computer Programming", 1st Edition, Divine Publishers, Ilesa, Nigeria.

ii. Odeniyi O.A and Kareem A.E.A (2008); "Data Processing: Concepts and Practice", 1st Edition, PP Concept, Osogbo, Nigeria.

iii. Kareem A.E.A (2010), "Understanding Computer Systems Architecture and File", Divine Publishers, Ilesa.

iv. Kareem A,E.A et al (2012). Fundamentals of Computer Science. Divine Publishers, Ilesa.

4. Journal Articles:

1. Wumi AJAYI, Afiss Emiola KAREEM, Chekwube EZECHI (2022). "Automata, Strings and Words of Languages: A Conceptual Review". International Research Journal of Modernization in Engineering Technology and Science. Vol. 04, Issue 11. PP 2034-2046
2. Agbaje, M.O, Johnson, O.A & Kareem A.E (2022): "Intellectual Property Processor Design: A Review". International Journal of Science Research and Technology. Vol. 9 No. 9. PP 41-52.
3. JOHNSON Oluwatobi Akanbi, ALLEN Akinkitan Ajose, KAREEM Afiss Emiola (2022). "Code Generation Techniques in Compiler Design: Conceptual and Structural Review". International Journal of Recent Engineering Science. Vol. 9 Issue 3. 1-6.
4. Odeniyi, O.A, Kareem, A.E.A, Sarumi O.A., Lawal N.T.A, & Akinwumi A.R (2019): "Development of an Improved Bayesian Spam Filtering Model for Zipped Attachment Spam Files". Oscotech Journal of Science, Technology and Management. 2(1): 14-23
5. Odeniyi O.A, Makinde B.O, Kareem A.E.A, Akinwumi A.R & Sarumi O.A (2019): "Towards Implementing Cloud-Based E-Learning System For Distance Education In Nigeria". The Forerunner Journal of Educational Research
6. Kareem A.E.A, & Odeniyi O.A (2019): "The Imperatives Of Ict In Information Processing and NationalDevelopment In Nigeria". Nigerian Research Journal Of Science And Technology
7. Odeniyi O.A, & Kareem A.E.A (2019): "The Adoption of Electronic Monitoring of Offenders And Accused Persons In Nigeria: A Theoretical Model". Nigerian Research Journal of Science and Technology
8. Lateef I.A, Kareem A.E.A, Odeniyi O.A, Sarumi O.A, & Oyeniran O.J (2019): "Maintenance Culture For Sustainable National Development In Developing

Countries". The Forerunner Journal Of Educational Research

9. Olufemi Ayodeji Odeniyi, Taiwo f. Igbaroola, Afiss E.A . Kareem, Bukola Oyeladun Makinde & Nurudeen Lawal (2018): "Evaluating Quality of Service in Grid ComputingEnvironment". Journal of Scientific and Engineering Research. 5(5): 344-352

10. Lawal N.T.A Odeniyi O.A, Makinde B.O, Adewale J.A & Kareem A.E.A (2018). An Investigative Study of BigData TechnologyApplication and its Impact inthe Upstream Operationsof an Oil and Gas Industry in Nigeria. International Journal of Innovative Researchand Advanced Studies (Ijiras). 5(6): 10-16

11. Idris A. Lateef, Afiss E.A Kareem, Olufemi A. Odeniyi (2018): "Computationof the Relationship Betweenthe Piston Ring Axial Thickness and Ring #". Journal of Scientific and Engineering Research. 5(5): 60-67

12. Odeniyi, O. Ayodeji, Lawal, N. Tunde, Kareem, A. E. Adebowale (2015); "AN Appraisal of Groupon E-business Model", International Journal of Scientific & Technology Research. 4(01): 291-297, January 2015.

13. Lateef, I.A, & Kareem, A.E.A (2015); "Simulation Effects of Increasing The Piston Ring Radial Thickness on the Piston Ring Gap". International Journal of Engineering & Applied Sciences. Jan. 2015, 6(01): 1-7

14. Kareem A.E.A, Ajobo J.A, Sarumi O.A & Makinde B.O (2016). Technical Education: A Productive Tool For Economic development. the Forerunner- Journal of Educational Research

15. O.O olaewe, A.E.A. Kareem, O.A Odeniyi & S.A Alabi (2010); "Adopting E-learning as an Effective Tool for Distance Education In Nigeria". International Research Review, A Multidiscuiplinary Journal of Osun State University, Ipetu-Ijesa Campus. VOL. 1 No II 12-25.

16. I.A.Lateef, B.A Hassan & A.E.A Kareem (2010). "Computational Analysis of a Relationship Between Fan

Blade Diameter and Depth of Radiator". Pacific Journal of Science and Technology. 11(2): 84-90.

17. Olaewe O.O, Aminu M.A & Kareem A.E.A (2009); Relationship Between Validity, Reliability and Discrimination is Indices for Objective Tests in Physics in Secondary School in Osun State, Journal of Research in Education. VOL 6 NO 2.
18. O.O. Olaewe, & A.E.A Kareem. 2009. "The Place of Parametric Statistical Methods in Conducting Research in the Millenium Age". Pacific journal of Science and Technology. 10(1): 170-177
19. A.E.A Kareem (2009). "Implementing a Desktop Video Conferencing Technology For Effective Teaching and Learning". Pacific Journal of science and Technology. 10(2): 419-428.
20. O.O Olaewe & A.E.A Kareem (2009). "Enhancing Workers' Performance Through ICT". Pacific journal of science and Technology. 10(2): 413-418
21. O.O Olaewe & A.E.A Kareem (2009), "Examination Malpractice in Nigeria, 'an Incurable Academic Virus'". International Research Review, a Multidisciplinary journal of Osun State University, Ipetu-Ijesa Campus. 1(1); 28-35.
22. A.E.A Kareem, T.R Bamigbade & I.A Lateef (2008). "Computer-Based Instrumentation: A Boost for Industrialization". Pacific Journal of science and Technology. 9(1): 66-71
23. A.E.A Kareem, O.O Olaewe, & O.A Odeniyi (2008). "The Roles of Ict in Information Processing and National Development in Nigeria". Pacific Journal of Science and Technology. 9(1): 90-96
24. I.A Lateef, B.A Hassan & A.E.A kareem (2008). "Theoretical Analysis of a Relationship Between

Master/Wheel Cylinder Ratio and Brake Efficiency". Pacific Journal of Science and Technology. 9(1): 155-162

6. Paper Presentation:

1. "Hardware Requirements for the Implementation of a Computer Network"; A Seminar Paper Presented at the Faculty of Engineering, Osun State College of Technology, Esa-oke. May, 2007
2. "Ict: Tool for Educational Development": a Seminar Paper Presented at the Faculty of Engineering, Osun State College of Technology, Esa-oke. July, 2007
3. "E-learning: A for Actualizing an Effective Distance Education in Nigeria" Being A Paper Presented at the 1ST National Workshop on National Infrastructural Development and Industrialization Challenges, Organized by: Prospect & Faculty of Engineering. Osun State College of Technology, Esa-Oke, 27TH-29TH NOV. 2007.
4. Empirical Assessment of the Complexity of oo Languages on The Implementation of Shortest-Path Algorithms". A Seminar Paper Presented At THE Department of Computer Science, Osun State College of Technology, esa-oke. February, 2013.
5. Using Data Minig and Knowledge Management to Improve Business Performance. A Paper Presened at Feng Seminar, Oscotech. DEC. 2014

5. Application of Agile Methodologies in System Development: a Review**.** A Paper Presened at Feng Seminar, Oscotech. March. 2015
6. Using Data Minig and Knowledge Management to Improve Business Performance**.** A Paper Presened at Feng Seminar, Oscotech. 19th dec. 2014
7. Seeking an Ideal Solution to the Management of Personal Information Collection. Faculty of Applied Sciences Seminar Series. 12th Feb. 2015
8. Enhancing Students' Performance Through Entrepreneuship Education in Nigeria at the Faculty of Engineering **2ND NATIONAL CONFERENCE**, Osun State College of Technology, Esa-oke. Dec. 2016

9. Quality of Service Metrics in Grid Computing Environment". A Paper Presented at the Faculty of Engineering **2nd National Conference**, Osun State College of Technology, Esa-oke. Dec. 2016

10. Technical Education: A Productive Tool for Economic Development. A Paper Presented at the Faculty of Engineering **2nd National Conference**, Osun State College of Technology, Esa-oke. Dec. 2016

11. A Review of Security Challenges and Solutions of Cloud Computing Technology". A Paper Presented at the Faculty of Applied Sciences Seminar Series. Dec.,2016

12. A Complexity Measure Based on RBC Metrics. A Seminar Paper Presented at the Department of Computer Science, Osun State College of Technology, Esa-Oke. March, 2017

13. An Assessment of how the Use of Digital Multimedia Technology Enhances Classroom Learning. A Paper Presented at the Faculty of Applied Sciences Seminar Series. May. 2018

14. Evaluating the Effectiveness of Interactive Multimedia in Teaching and Learning. A Paper Presented at the Faculty of Applied Sciences Seminar Series. 12th feb,2018

15. Impact of Cryptocurrency on Traditional Banking System in Nigeria. A Paper Presented at the Faculty of Applied Sciences Seminar Series. May 2018

16. The Imperatives of Ict in Information Processing and National Development in Nigeria. A Paper Presented at the Faculty of Applied Sciences Annual National Conference. 27th June, 2019

17. The Adoption of Electronic Monitoring of Offenders and Accussed Persons in Nigeria. A Paper Presented at the Faculty of Applied Sciences Annual National Conference. 27th June, 2019

G. Conference/Seminar/Workshop Attended:

i. A Two-Day National Workshop on **"Globalization and Technological Education in Nigeria: Contemporary Challenges".** Organized by Continuing Academic Programme (cap); Osun State college of Technology, Esa-oke. Nov., 2003

ii. **ETF Capacity Building Workshop** for Lecturers of Nigeria Polytechnics Held at the federal **polytechnic**, Ado-Ekiti. Oct. 2005

iii. A Day Academic Staff Seminar on **"Towards Academic Excellence: The Pen and Paper"** Organized by Osun State College of Technology, Esa-oke. 27th April, 2006.

iv. A One-day Seminal on **"Writing Acceptable Academic Papers for Journals and Conferences"** Organized by Faculty of Environmental Studies, Osun State College of Technology,Esa-oke. 2007.

v. 21st Annual National Conference on **Achieving the Millennium Development Goals in Nigeria (It Strategies and Tools)** organized by Nigeria Computer Society at Concorde Hotel, Owerri, Imo State. 17th–20th July, 2007

vi. Annual Information Technology Professional's Assembly with the Theme "**Mobilising it Professionals for Vision 2020"** Organized by Computer Professionals (Registration Council of Nigeria) at Nicon Luxury Hotel, Abuja. 25th-26th Oct. 2007

vii. 1st National Workshop on **"National Infrastructural Development & Industrialization: Prospects and Challenges"** by Faculty of Engineering of the Osun State College of Technology, Esa-oke. 27th-29th Nov, 2007.

viii. 10th Faculty of Education Conference on **"Universal Basic Education: Implementation, Functionality and Sustainability"** Held at the Faculty of Education Sandwich Building, Lagos State University Main Campus, Ojo, Lagos. 28th – 31st July, 2008

ix. 1st International Conference on **Mobile Computing, Wireless Communication, E-health, M-health and Telemedicine (mwemtem'08)** Held at the Ladoke Akintola University of Technology, Ogbomoso, Nigeria Between November 18th-20th, 2008. At the University Senate Chamber.

xWorkshop on **Improving Communication, Public-Speaking and Report-Writing** for Heads of Departments, Senior Academic and Non-Academic Staff of Osun Sate College of Technology, Esa-oke. 15th-16th april, 2010.

xi. A One-day Workshop on **Writing a Worldclass Papers for Publication in Engineering Journals,** Organised by Faculty of Engineering of the Osun State College of Technology, Esa-oke Held on 28th April, 2010.

xii National Conference On " towards a cashless nigeria: tools and strategies", organized by nigeria computer society held at le-meridien hotel, uyo, akwa-ibom state between 25th-28th july, 2012.

xiii 7th International Conference On Ict Application to Teaching, Research, and Administration, Organized by Obafemi Awolowo University and Carnegie Corporation of New York, Held at Nigeria Defence Academy, Abuja Between September 2-6, 2012.

xiv 2012 Annual Guest Lecture Titled "Neglecting Research and Innovations in Science and Technology, the Highway to Another Enslavement of the Black Race", Organized by Faculty Of Engineering,

Osun State College of Technology, Esa-oke at Timesed Hotel, Ijebu-jesa, Osun State on the 5th December, 2012.

xv 11th International Conference on E-government & National Security Organized by Nigeria Computer Society Held at Royal Park Hotel, Iloko-ijesa, Osun State, Between 24th-26th July, 2013.

xvi 2nd National Engineering Conference on Science, Engineering and Technology: Veritable Tools for Achieving Sustainable Development Goals. Organized by Faculty of Engineering, Osun State College of Technology, Esa-oke on Monday 12 – Friday 16 December, 2016

xvii Faculty of Applied Sciences Seminar/Workshop on Presentation of Journal Papers, Held in the Faculty of Applied Sciences, Osun State College of Technology, Esa-oke in May, 2017

xviii Advanced Digital Appreciation programme for Tertiary Institutions on Data Analysis Using Statistical Package for Social Science (**SPSS**) by International Centre for Information & Communications Technology Studies in Conjuction with Nigerian Communications Commission, Held at Osun State College of Technology, Esa-oke Between May 22 – May 26, 2017

xix 27th National Conference of the Nigeria Computer Society on Digital Inclusion, Held at the International Conference Centre of the University of Ibadan, Oyo State Between July 17th-July 19th,2018

xx. 1st National Annual Conference of the Faculty of Pure and Applied Sciences with the Theme: Science and Technology: Nexus for Sustainable National Development in Nigeria Between 24th-27th June, 2019

H. **Extra Curricula Activities:**

i. **Welfare Officer,** Amity Club of Nigeria, Iwo, Nigeria

ii. **Staff Adviser,** National Association of Computer Science Students, Oscotech Chapter, Esa-oke.

Iii. **Patron,** Federation of Iwoland Students Union, Oscotech Chapter

iv. **Vice chairman,** Alabiye/Irepodun CDA Zone 3, Owode-Ede

v. **Staff adviser,** Junior Chambers International, Oscotech Chapter

I. **Hobbies:**

Reading & Computer Use

J. **Referees:**

1. Dr. S.A.O Adegoke
   Rector, Osun State College of Technology,
   Esa-Oke, Nigeria.
2. Engr. Olu Ayinde
   Dean, Faculty of Engineering
   Osun State College of Technology,
   Esa-Oke, Nigeria.
3. Dr. A. P Ayeni
   Dean, Faculty of Business and Management Studies
   Osun State College of Technology,
   Esa-Oke

---------- ----------

Signature **Date**

The University Compliance Certification

This is to certify that this thesis by **Afiss Emiola KAREEM** with matric number LCU/PG/001703 in the Department of Computer Science, Faculty of Natural and Applied Sciences, Lead City University, Ibadan, Oyo State, Nigeria is in FULL compliance with the approved University Format and Style.

…….. ……..

Signature **Date**